Claude Code

Dokumentasi Lengkap

By Anthropic

Tanggal produksi：2026-03-17

Alat pengkodean agentik di terminal Anda,
memahami basis kode Anda, dan membantu Anda membuat kode lebih cepat.

Dikompilasi dari code.claude.com/docs/id
Untuk versi terbaru, kunjungi situs web.

Generated by: github.com/bepsvpt/cc-docs

Table of Contents

Part 1: Getting Started

Ikhtisar Claude Code

> *Claude Code adalah alat pengkodean agentic yang membaca basis kode Anda, mengedit file, menjalankan perintah, dan terintegrasi dengan alat pengembangan Anda. Tersedia di terminal, IDE, aplikasi desktop, dan browser Anda.*

Claude Code adalah asisten pengkodean bertenaga AI yang membantu Anda membangun fitur, memperbaiki bug, dan mengotomatisasi tugas pengembangan. Ini memahami seluruh basis kode Anda dan dapat bekerja di berbagai file dan alat untuk menyelesaikan pekerjaan.

Memulai

Pilih lingkungan Anda untuk memulai. Sebagian besar permukaan memerlukan langganan Claude atau akun Konsol Anthropic. Terminal CLI dan VS Code juga mendukung penyedia pihak ketiga.

Terminal

CLI lengkap untuk bekerja dengan Claude Code langsung di terminal Anda. Edit file, jalankan perintah, dan kelola seluruh proyek Anda dari baris perintah.

To install Claude Code, use one of the following methods:

Native Install (Recommended)

macOS, Linux, WSL:

```
curl -fsSL https://claude.ai/install.sh | bash
```

Windows PowerShell:

```
irm https://claude.ai/install.ps1 | iex
```

Windows CMD:

```
curl -fsSL https://claude.ai/install.cmd -o install.cmd && install.cmd && del
install.cmd
```

Windows requires Git for Windows. Install it first if you don't have it.

> **Info:**
>
> Native installations automatically update in the background to keep you on the latest version.

Homebrew

```
brew install --cask claude-code
```

> **Info:**
>
> Homebrew installations do not auto-update. Run `brew upgrade claude-code` periodically to get the latest features and security fixes.

WinGet

```
winget install Anthropic.ClaudeCode
```

> **Info:**
>
> WinGet installations do not auto-update. Run `winget upgrade Anthropic.ClaudeCode` periodically to get the latest features and security fixes.

Kemudian mulai Claude Code di proyek apa pun:

```
cd your-project
claude
```

Anda akan diminta untuk masuk pada penggunaan pertama. Itu saja! Lanjutkan dengan Quickstart →

> **Tip:**

> Lihat pengaturan lanjutan untuk opsi instalasi, pembaruan manual, atau instruksi penghapusan. Kunjungi pemecahan masalah jika Anda mengalami masalah.

VS Code

Ekstensi VS Code menyediakan diff inline, @-mentions, tinjauan rencana, dan riwayat percakapan langsung di editor Anda.

- Instal untuk VS Code
- Instal untuk Cursor

Atau cari "Claude Code" di tampilan Ekstensi (`Cmd+Shift+X` di Mac, `Ctrl+Shift+X` di Windows/Linux). Setelah menginstal, buka Palet Perintah (`Cmd+Shift+P` / `Ctrl+Shift+P`), ketik "Claude Code", dan pilih **Buka di Tab Baru**.

Mulai dengan VS Code →

Aplikasi desktop

Aplikasi mandiri untuk menjalankan Claude Code di luar IDE atau terminal Anda. Tinjau diff secara visual, jalankan beberapa sesi berdampingan, jadwalkan tugas berulang, dan mulai sesi cloud.

Unduh dan instal:

- macOS (Intel dan Apple Silicon)
- Windows (x64)
- Windows ARM64 (hanya sesi jarak jauh)

Setelah menginstal, luncurkan Claude, masuk, dan klik tab **Code** untuk mulai pengkodean. Langganan berbayar diperlukan.

Pelajari lebih lanjut tentang aplikasi desktop →

Web

Jalankan Claude Code di browser Anda tanpa pengaturan lokal. Mulai tugas yang berjalan lama dan periksa kembali saat selesai, bekerja pada repo yang tidak Anda miliki secara lokal, atau jalankan beberapa tugas secara paralel. Tersedia di browser desktop dan aplikasi Claude iOS.

Mulai pengkodean di claude.ai/code.

Mulai di web →

JetBrains

Plugin untuk IntelliJ IDEA, PyCharm, WebStorm, dan IDE JetBrains lainnya dengan tampilan diff interaktif dan berbagi konteks seleksi.

Instal plugin Claude Code dari Marketplace JetBrains dan mulai ulang IDE Anda.

Mulai dengan JetBrains →

Apa yang dapat Anda lakukan

Berikut adalah beberapa cara Anda dapat menggunakan Claude Code:

Claude Code menangani tugas-tugas membosankan yang menghabiskan hari Anda: menulis tes untuk kode yang tidak diuji, memperbaiki kesalahan lint di seluruh proyek, menyelesaikan konflik penggabungan, memperbarui dependensi, dan menulis catatan rilis.

```
claude "write tests for the auth module, run them, and fix any failures"
```

Jelaskan apa yang Anda inginkan dalam bahasa biasa. Claude Code merencanakan pendekatan, menulis kode di berbagai file, dan memverifikasi bahwa itu berfungsi.

Untuk bug, tempel pesan kesalahan atau jelaskan gejalanya. Claude Code melacak masalah melalui basis kode Anda, mengidentifikasi akar penyebabnya, dan menerapkan perbaikan. Lihat alur kerja umum untuk contoh lebih lanjut.

Claude Code bekerja langsung dengan git. Ini menampilkan perubahan, menulis pesan commit, membuat cabang, dan membuka pull request.

```
claude "commit my changes with a descriptive message"
```

Di CI, Anda dapat mengotomatisasi tinjauan kode dan triase masalah dengan GitHub Actions atau GitLab CI/CD.

Model Context Protocol (MCP) adalah standar terbuka untuk menghubungkan alat AI ke sumber data eksternal. Dengan MCP, Claude Code dapat membaca dokumen desain Anda di Google Drive, memperbarui tiket di Jira, menarik data dari Slack, atau menggunakan alat kustom Anda sendiri.

`CLAUDE.md` adalah file markdown yang Anda tambahkan ke root proyek Anda yang dibaca Claude Code di awal setiap sesi. Gunakan untuk menetapkan standar pengkodean, keputusan arsitektur, perpustakaan pilihan, dan daftar periksa tinjauan. Claude juga membangun memori otomatis saat bekerja, menyimpan pembelajaran seperti perintah build dan wawasan debugging di seluruh sesi tanpa Anda menulis apa pun.

Buat perintah kustom untuk mengemas alur kerja yang dapat diulang yang dapat dibagikan tim Anda, seperti `/review-pr` atau `/deploy-staging`.

Hooks memungkinkan Anda menjalankan perintah shell sebelum atau sesudah tindakan Claude Code, seperti pemformatan otomatis setelah setiap pengeditan file atau menjalankan lint sebelum commit.

Spawn beberapa agen Claude Code yang bekerja pada bagian berbeda dari tugas secara bersamaan. Agen utama mengoordinasikan pekerjaan, menetapkan subtask, dan menggabungkan hasil.

Untuk alur kerja yang sepenuhnya kustom, Agent SDK memungkinkan Anda membangun agen Anda sendiri yang didukung oleh alat dan kemampuan Claude Code, dengan kontrol penuh atas orkestrasi, akses alat, dan izin.

Claude Code dapat dikomposisi dan mengikuti filosofi Unix. Pipa log ke dalamnya, jalankan di CI, atau rantai dengan alat lain:

```
## Monitor logs and get alerted
tail -f app.log | claude -p "Slack me if you see any anomalies"

## Automate translations in CI
claude -p "translate new strings into French and raise a PR for review"

## Bulk operations across files
git diff main --name-only | claude -p "review these changed files for security
issues"
```

Lihat referensi CLI untuk set lengkap perintah dan flag.

Sesi tidak terikat pada satu permukaan. Pindahkan pekerjaan antar lingkungan saat konteks Anda berubah:

- Tinggalkan meja Anda dan terus bekerja dari ponsel Anda atau browser apa pun dengan Remote Control
- Mulai tugas yang berjalan lama di web atau aplikasi iOS, kemudian tariknya ke terminal Anda dengan `/teleport`
- Serahkan sesi terminal ke aplikasi Desktop dengan `/desktop` untuk tinjauan diff visual
- Rute tugas dari obrolan tim: sebutkan `@Claude` di Slack dengan laporan bug dan dapatkan pull request kembali

Gunakan Claude Code di mana saja

Setiap permukaan terhubung ke mesin Claude Code yang mendasar yang sama, jadi file CLAUDE.md, pengaturan, dan server MCP Anda bekerja di semua permukaan.

Selain lingkungan Terminal, VS Code, JetBrains, Desktop, dan Web di atas, Claude Code terintegrasi dengan alur kerja CI/CD, obrolan, dan browser:

Saya ingin...	Opsi terbaik
Lanjutkan sesi lokal dari ponsel atau perangkat lain saya	Remote Control
Mulai tugas secara lokal, lanjutkan di mobile	Web atau aplikasi Claude iOS
Otomatisasi tinjauan PR dan triase masalah	GitHub Actions atau GitLab CI/CD
Dapatkan tinjauan kode otomatis di setiap PR	GitHub Code Review
Rute laporan bug dari Slack ke pull request	Slack
Debug aplikasi web langsung	Chrome
Bangun agen kustom untuk alur kerja Anda sendiri	Agent SDK

Langkah berikutnya

Setelah Anda menginstal Claude Code, panduan ini membantu Anda menggali lebih dalam.

- Quickstart: berjalan melalui tugas nyata pertama Anda, dari menjelajahi basis kode hingga melakukan perbaikan
- Simpan instruksi dan memori: berikan Claude instruksi persisten dengan file CLAUDE.md dan memori otomatis
- Alur kerja umum dan praktik terbaik: pola untuk mendapatkan hasil maksimal dari Claude Code
- Pengaturan: sesuaikan Claude Code untuk alur kerja Anda
- Pemecahan masalah: solusi untuk masalah umum

- code.claude.com: demo, harga, dan detail produk

Panduan Cepat

> *Selamat datang di Claude Code!*

Panduan cepat ini akan membuat Anda menggunakan bantuan coding bertenaga AI dalam beberapa menit. Di akhir panduan, Anda akan memahami cara menggunakan Claude Code untuk tugas-tugas pengembangan umum.

Sebelum Anda memulai

Pastikan Anda memiliki:

- Terminal atau command prompt yang terbuka
  - Jika Anda belum pernah menggunakan terminal sebelumnya, lihat panduan terminal
- Proyek kode untuk dikerjakan
- Langganan Claude (Pro, Max, Teams, atau Enterprise), akun Claude Console, atau akses melalui penyedia cloud yang didukung

> **Note:**
>
> Panduan ini mencakup CLI terminal. Claude Code juga tersedia di web, sebagai aplikasi desktop, di VS Code dan IDE JetBrains, di Slack, dan di CI/CD dengan GitHub Actions dan GitLab. Lihat semua antarmuka.

Langkah 1: Instal Claude Code

To install Claude Code, use one of the following methods:

Native Install (Recommended)

macOS, Linux, WSL:

```
curl -fsSL https://claude.ai/install.sh | bash
```

Windows PowerShell:

```
irm https://claude.ai/install.ps1 | iex
```

Windows CMD:

```
curl -fsSL https://claude.ai/install.cmd -o install.cmd && install.cmd && del install.cmd
```

Windows requires Git for Windows. Install it first if you don't have it.

> **Info:**
>
> Native installations automatically update in the background to keep you on the latest version.

Homebrew

```
brew install --cask claude-code
```

> **Info:**
>
> Homebrew installations do not auto-update. Run `brew upgrade claude-code` periodically to get the latest features and security fixes.

WinGet

```
winget install Anthropic.ClaudeCode
```

> **Info:**
>
> WinGet installations do not auto-update. Run `winget upgrade Anthropic.ClaudeCode` periodically to get the latest features and security fixes.

Langkah 2: Masuk ke akun Anda

Claude Code memerlukan akun untuk digunakan. Ketika Anda memulai sesi interaktif dengan perintah `claude`, Anda perlu masuk:

```
claude
## Anda akan diminta untuk masuk pada penggunaan pertama
```

```
/login
## Ikuti petunjuk untuk masuk dengan akun Anda
```

Anda dapat masuk menggunakan salah satu jenis akun ini:

- Claude Pro, Max, Teams, atau Enterprise (direkomendasikan)
- Claude Console (akses API dengan kredit prabayar). Pada login pertama, workspace "Claude Code" secara otomatis dibuat di Console untuk pelacakan biaya terpusat.
- Amazon Bedrock, Google Vertex AI, atau Microsoft Foundry (penyedia cloud enterprise)

Setelah masuk, kredensial Anda disimpan dan Anda tidak perlu masuk lagi. Untuk beralih akun nanti, gunakan perintah `/login`.

Langkah 3: Mulai sesi pertama Anda

Buka terminal Anda di direktori proyek mana pun dan mulai Claude Code:

```
cd /path/to/your/project
claude
```

Anda akan melihat layar sambutan Claude Code dengan informasi sesi, percakapan terbaru, dan pembaruan terbaru. Ketik `/help` untuk perintah yang tersedia atau `/resume` untuk melanjutkan percakapan sebelumnya.

> **Tip:**
>
> Setelah masuk (Langkah 2), kredensial Anda disimpan di sistem Anda. Pelajari lebih lanjut di Manajemen Kredensial.

Langkah 4: Ajukan pertanyaan pertama Anda

Mari kita mulai dengan memahami basis kode Anda. Coba salah satu perintah ini:

```
apa yang dilakukan proyek ini?
```

Claude akan menganalisis file Anda dan memberikan ringkasan. Anda juga dapat mengajukan pertanyaan yang lebih spesifik:

```
teknologi apa yang digunakan proyek ini?
```

```
di mana titik masuk utama?
```

```
jelaskan struktur folder
```

Anda juga dapat bertanya kepada Claude tentang kemampuannya sendiri:

```
apa yang dapat dilakukan Claude Code?
```

```
bagaimana cara membuat skills kustom di Claude Code?
```

```
bisakah Claude Code bekerja dengan Docker?
```

> **Note:**
>
> Claude Code membaca file proyek Anda sesuai kebutuhan. Anda tidak perlu menambahkan konteks secara manual.

Langkah 5: Buat perubahan kode pertama Anda

Sekarang mari buat Claude Code melakukan beberapa coding sebenarnya. Coba tugas sederhana:

```
tambahkan fungsi hello world ke file utama
```

Claude Code akan:

1. Menemukan file yang sesuai
2. Menampilkan perubahan yang diusulkan
3. Meminta persetujuan Anda
4. Membuat edit

> **Note:**
>
> Claude Code selalu meminta izin sebelum memodifikasi file. Anda dapat menyetujui perubahan individual atau mengaktifkan mode "Terima semua" untuk sesi.

Langkah 6: Gunakan Git dengan Claude Code

Claude Code membuat operasi Git menjadi percakapan:

```
file apa yang telah saya ubah?
```

```
commit perubahan saya dengan pesan deskriptif
```

Anda juga dapat meminta operasi Git yang lebih kompleks:

```
buat cabang baru bernama feature/quickstart
```

```
tunjukkan 5 commit terakhir saya
```

```
bantu saya menyelesaikan konflik merge
```

Langkah 7: Perbaiki bug atau tambahkan fitur

Claude mahir dalam debugging dan implementasi fitur.

Jelaskan apa yang Anda inginkan dalam bahasa alami:

```
tambahkan validasi input ke formulir pendaftaran pengguna
```

Atau perbaiki masalah yang ada:

```
ada bug di mana pengguna dapat mengirimkan formulir kosong - perbaiki
```

Claude Code akan:

- Menemukan kode yang relevan
- Memahami konteksnya
- Menerapkan solusi
- Menjalankan tes jika tersedia

Langkah 8: Coba alur kerja umum lainnya

Ada beberapa cara untuk bekerja dengan Claude:

Refactor kode

```
refactor modul autentikasi untuk menggunakan async/await alih-alih callbacks
```

Tulis tes

```
tulis unit test untuk fungsi kalkulator
```

Perbarui dokumentasi

```
perbarui README dengan instruksi instalasi
```

Tinjauan kode

```
tinjau perubahan saya dan sarankan perbaikan
```

> **Tip:**

> Berbicara dengan Claude seperti Anda berbicara dengan rekan kerja yang membantu. Jelaskan apa yang ingin Anda capai, dan itu akan membantu Anda sampai ke sana.

Perintah penting

Berikut adalah perintah paling penting untuk penggunaan sehari-hari:

Perintah	Apa yang dilakukannya	Contoh
`claude`	Mulai mode interaktif	`claude`
`claude "task"`	Jalankan tugas satu kali	`claude "perbaiki kesalahan build"`
`claude -p "query"`	Jalankan kueri sekali jalan, lalu keluar	`claude -p "jelaskan fungsi ini"`
`claude -c`	Lanjutkan percakapan terbaru di direktori saat ini	`claude -c`
`claude -r`	Lanjutkan percakapan sebelumnya	`claude -r`
`claude commit`	Buat commit Git	`claude commit`
`/clear`	Hapus riwayat percakapan	`/clear`
`/help`	Tampilkan perintah yang tersedia	`/help`
`exit` atau Ctrl+C	Keluar dari Claude Code	`exit`

Lihat referensi CLI untuk daftar lengkap perintah.

Tips pro untuk pemula

Untuk informasi lebih lanjut, lihat praktik terbaik dan alur kerja umum.

Alih-alih: “perbaiki bug”

Coba: “perbaiki bug login di mana pengguna melihat layar kosong setelah memasukkan kredensial yang salah”

Pecah tugas kompleks menjadi langkah-langkah:

```
1. buat tabel database baru untuk profil pengguna
2. buat endpoint API untuk mendapatkan dan memperbarui profil pengguna
3. bangun halaman web yang memungkinkan pengguna melihat dan mengedit informasi
mereka
```

Sebelum membuat perubahan, biarkan Claude memahami kode Anda:

```
analisis skema database
```

```
bangun dasbor yang menampilkan produk yang paling sering dikembalikan oleh
pelanggan Inggris kami
```

- Tekan `?` untuk melihat semua pintasan keyboard yang tersedia
- Gunakan Tab untuk penyelesaian perintah
- Tekan ↑ untuk riwayat perintah
- Ketik `/` untuk melihat semua perintah dan skills

Apa selanjutnya?

Sekarang yang Anda telah mempelajari dasar-dasarnya, jelajahi fitur-fitur yang lebih canggih:

- Cara kerja Claude Code Pahami loop agentic, alat bawaan, dan cara Claude Code berinteraksi dengan proyek Anda
- Praktik terbaik Dapatkan hasil yang lebih baik dengan prompting yang efektif dan pengaturan proyek
- Alur kerja umum Panduan langkah demi langkah untuk tugas-tugas umum
- Perluas Claude Code Sesuaikan dengan CLAUDE.md, skills, hooks, MCP, dan lainnya

Mendapatkan bantuan

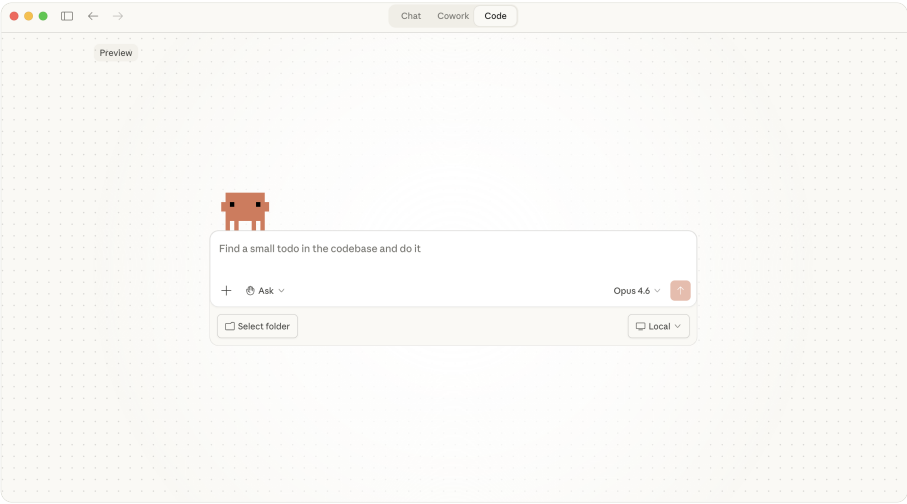
- **Di Claude Code**: Ketik `/help` atau tanya “bagaimana cara saya…”
- **Dokumentasi**: Anda di sini! Jelajahi panduan lainnya
- **Komunitas**: Bergabunglah dengan Discord kami untuk tips dan dukungan

Memulai dengan aplikasi desktop

> *Instal Claude Code di desktop dan mulai sesi coding pertama Anda*

Aplikasi desktop memberi Anda Claude Code dengan antarmuka grafis: tinjauan diff visual, pratinjau aplikasi langsung, pemantauan GitHub PR dengan penggabungan otomatis, sesi paralel dengan isolasi Git worktree, tugas terjadwal, dan kemampuan untuk menjalankan tugas dari jarak jauh. Tidak perlu terminal.

Halaman ini memandu Anda melalui instalasi aplikasi dan memulai sesi pertama Anda. Jika Anda sudah siap, lihat Gunakan Claude Code Desktop untuk referensi lengkap.



Antarmuka Claude Code Desktop menampilkan tab Code yang dipilih, dengan kotak prompt, pemilih mode izin diatur ke Minta izin, pemilih model, pemilih folder, dan opsi Lingkungan Lokal

Aplikasi desktop memiliki tiga tab:

- **Chat**: Percakapan umum tanpa akses file, mirip dengan claude.ai.
- **Cowork**: Agen latar belakang otonom yang bekerja pada tugas di VM cloud dengan lingkungannya sendiri. Dapat berjalan secara independen saat Anda melakukan pekerjaan lain.
- **Code**: Asisten coding interaktif dengan akses langsung ke file lokal Anda. Anda meninjau dan menyetujui setiap perubahan secara real-time.

Chat dan Cowork tercakup dalam artikel dukungan Claude Desktop. Halaman ini berfokus pada tab **Code**.

> **Note:**
>
> Claude Code memerlukan langganan Pro, Max, Teams, atau Enterprise.

Instal

Step 1: Unduh aplikasi

Unduh Claude untuk platform Anda.

- macOS Build universal untuk Intel dan Apple Silicon
- Windows Untuk prosesor x64

Untuk Windows ARM64, unduh di sini.

Linux saat ini tidak didukung.

Step 2: Masuk

Luncurkan Claude dari folder Aplikasi Anda (macOS) atau menu Start (Windows). Masuk dengan akun Anthropic Anda.

Step 3: Buka tab Code

Klik tab **Code** di bagian atas tengah. Jika mengklik Code meminta Anda untuk upgrade, Anda perlu berlangganan paket berbayar terlebih dahulu. Jika meminta Anda untuk masuk online, selesaikan masuk dan mulai ulang aplikasi. Jika Anda melihat kesalahan 403, lihat pemecahan masalah autentikasi.

Aplikasi desktop menyertakan Claude Code. Anda tidak perlu menginstal Node.js atau CLI secara terpisah. Untuk menggunakan `claude` dari terminal, instal CLI secara terpisah. Lihat Memulai dengan CLI.

Mulai sesi pertama Anda

Dengan tab Code terbuka, pilih proyek dan beri Claude sesuatu untuk dikerjakan.

Step 1: Pilih lingkungan dan folder

Pilih **Local** untuk menjalankan Claude di mesin Anda menggunakan file Anda secara langsung. Klik **Select folder** dan pilih direktori proyek Anda.

> **Tip:**
>
> Mulai dengan proyek kecil yang Anda kenal dengan baik. Ini adalah cara tercepat untuk melihat apa yang dapat dilakukan Claude Code. Di Windows, Git harus diinstal agar sesi lokal berfungsi. Sebagian besar Mac menyertakan Git secara default.

Anda juga dapat memilih:

- **Remote**: Jalankan sesi pada infrastruktur cloud Anthropic yang berlanjut bahkan jika Anda menutup aplikasi. Sesi remote menggunakan infrastruktur yang sama dengan Claude Code di web.
- **SSH**: Terhubung ke mesin jarak jauh melalui SSH (server Anda sendiri, VM cloud, atau dev containers). Claude Code harus diinstal di mesin jarak jauh.

Step 2: Pilih model

Pilih model dari dropdown di sebelah tombol kirim. Lihat models untuk perbandingan Opus, Sonnet, dan Haiku. Anda tidak dapat mengubah model setelah sesi dimulai.

Step 3: Beri tahu Claude apa yang harus dilakukan

Ketik apa yang ingin Anda lakukan Claude:

- `Find a TODO comment and fix it`
- `Add tests for the main function`
- `Create a CLAUDE.md with instructions for this codebase`

Sesi adalah percakapan dengan Claude tentang kode Anda. Setiap sesi melacak konteks dan perubahannya sendiri, sehingga Anda dapat bekerja pada beberapa tugas tanpa saling mengganggu.

Step 4: Tinjau dan terima perubahan

Secara default, tab Code dimulai dalam mode Minta izin, di mana Claude mengusulkan perubahan dan menunggu persetujuan Anda sebelum menerapkannya. Anda akan melihat:

1. Tampilan diff menunjukkan dengan tepat apa yang akan berubah di setiap file
2. Tombol Terima/Tolak untuk menyetujui atau menolak setiap perubahan
3. Pembaruan real-time saat Claude menyelesaikan permintaan Anda

Jika Anda menolak perubahan, Claude akan bertanya bagaimana Anda ingin melanjutkan dengan cara yang berbeda. File Anda tidak dimodifikasi sampai Anda menerima.

Sekarang apa?

Anda telah membuat edit pertama Anda. Untuk referensi lengkap tentang semua yang dapat dilakukan Desktop, lihat Gunakan Claude Code Desktop. Berikut adalah beberapa hal yang dapat dicoba selanjutnya.

Interupsi dan arahkan. Anda dapat menghentikan Claude kapan saja. Jika itu menuju jalan yang salah, klik tombol stop atau ketik koreksi Anda dan tekan **Enter**. Claude berhenti melakukan apa yang sedang dilakukannya dan menyesuaikan berdasarkan input Anda. Anda tidak perlu menunggu sampai selesai atau memulai dari awal.

Beri Claude lebih banyak konteks. Ketik `@filename` di kotak prompt untuk menarik file tertentu ke dalam percakapan, lampirkan gambar dan PDF menggunakan tombol lampiran, atau seret dan lepas file langsung ke prompt. Semakin banyak konteks yang dimiliki Claude, semakin baik hasilnya. Lihat Tambahkan file dan konteks.

Gunakan skills untuk tugas yang dapat diulang. Ketik `/` atau klik **+** → **Slash commands** untuk menjelajahi perintah bawaan, skills kustom, dan skills plugin. Skills adalah prompt yang dapat digunakan kembali yang dapat Anda panggil kapan pun Anda membutuhkannya, seperti daftar periksa tinjauan kode atau langkah penyebaran.

Tinjau perubahan sebelum melakukan commit. Setelah Claude mengedit file, indikator `+12 -1` muncul. Klik untuk membuka tampilan diff, tinjau modifikasi file demi file, dan beri komentar pada baris tertentu. Claude membaca komentar Anda dan merevisi. Klik **Review code** untuk membuat Claude mengevaluasi diff itu sendiri dan meninggalkan saran inline.

Sesuaikan berapa banyak kontrol yang Anda miliki. Mode izin Anda mengontrol keseimbangan. Minta izin (default) memerlukan persetujuan sebelum setiap edit. Auto accept edits secara otomatis menerima edit file untuk iterasi yang lebih cepat. Plan mode memungkinkan Claude memetakan pendekatan tanpa menyentuh file apa pun, yang berguna sebelum refactor besar.

Tambahkan plugins untuk kemampuan lebih. Klik tombol **+** di sebelah kotak prompt dan pilih **Plugins** untuk menjelajahi dan menginstal plugins yang menambahkan skills, agents, MCP servers, dan lainnya.

Pratinjau aplikasi Anda. Klik dropdown **Preview** untuk menjalankan dev server Anda langsung di desktop. Claude dapat melihat aplikasi yang berjalan, menguji endpoint, memeriksa log, dan melakukan iterasi pada apa yang dilihatnya. Lihat Pratinjau aplikasi Anda.

Lacak pull request Anda. Setelah membuka PR, Claude Code memantau hasil pemeriksaan CI dan dapat secara otomatis memperbaiki kegagalan atau menggabungkan PR setelah semua pemeriksaan lulus. Lihat Pantau status pull request.

Letakkan Claude pada jadwal. Atur tugas terjadwal untuk menjalankan Claude secara otomatis secara berulang: tinjauan kode harian setiap pagi, audit dependensi mingguan, atau briefing yang menarik dari alat yang terhubung.

Skalakan ketika Anda siap. Buka sesi paralel dari sidebar untuk bekerja pada beberapa tugas sekaligus, masing-masing di Git worktree-nya sendiri. Kirim pekerjaan jangka panjang ke cloud sehingga terus berlanjut bahkan jika Anda menutup aplikasi, atau lanjutkan sesi di web atau di IDE Anda jika tugas memakan waktu lebih lama dari yang diharapkan. Hubungkan alat eksternal seperti GitHub, Slack, dan Linear untuk menyatukan alur kerja Anda.

Datang dari CLI?

Desktop menjalankan mesin yang sama dengan CLI dengan antarmuka grafis. Anda dapat menjalankan keduanya secara bersamaan pada proyek yang sama, dan mereka berbagi konfigurasi (file CLAUDE.md, MCP servers, hooks, skills, dan settings). Untuk perbandingan lengkap fitur, setara flag, dan apa yang tidak tersedia di Desktop, lihat perbandingan CLI.

Apa selanjutnya

- Gunakan Claude Code Desktop: mode izin, sesi paralel, tampilan diff, konektor, dan konfigurasi enterprise
- Pemecahan masalah: solusi untuk kesalahan umum dan masalah setup
- Praktik terbaik: tips untuk menulis prompt yang efektif dan mendapatkan hasil maksimal dari Claude Code
- Alur kerja umum: tutorial untuk debugging, refactoring, testing, dan lainnya

Siapkan Claude Code

> *Instal, autentikasi, dan mulai menggunakan Claude Code di mesin pengembangan Anda.*

Persyaratan sistem

- **Sistem Operasi**:
  - macOS 13.0+
  - Windows 10 1809+ atau Windows Server 2019+ (lihat catatan setup)
  - Ubuntu 20.04+
  - Debian 10+
  - Alpine Linux 3.19+ (dependensi tambahan diperlukan)
- **Hardware**: RAM 4 GB+
- **Jaringan**: Koneksi internet diperlukan (lihat konfigurasi jaringan)
- **Shell**: Bekerja terbaik di Bash atau Zsh
- **Lokasi**: Negara yang didukung Anthropic

Dependensi tambahan

- **ripgrep**: Biasanya disertakan dengan Claude Code. Jika pencarian gagal, lihat pemecahan masalah pencarian.
- **Node.js 18+**: Hanya diperlukan untuk instalasi npm yang sudah usang

Instalasi

To install Claude Code, use one of the following methods:

Native Install (Recommended)

macOS, Linux, WSL:

```
curl -fsSL https://claude.ai/install.sh | bash
```

Windows PowerShell:

```
irm https://claude.ai/install.ps1 | iex
```

Windows CMD:

```
curl -fsSL https://claude.ai/install.cmd -o install.cmd && install.cmd && del
install.cmd
```

Windows requires Git for Windows. Install it first if you don't have it.

> **Info:**
>
> Native installations automatically update in the background to keep you on the latest version.

Homebrew

```
brew install --cask claude-code
```

> **Info:**
>
> Homebrew installations do not auto-update. Run `brew upgrade claude-code` periodically to get the latest features and security fixes.

WinGet

```
winget install Anthropic.ClaudeCode
```

> **Info:**
>
> WinGet installations do not auto-update. Run `winget upgrade Anthropic.ClaudeCode` periodically to get the latest features and security fixes.

Setelah proses instalasi selesai, navigasikan ke proyek Anda dan mulai Claude Code:

```
cd your-awesome-project
claude
```

Jika Anda mengalami masalah apa pun selama instalasi, konsultasikan panduan pemecahan masalah.

> **Tip:**
>
> Jalankan `claude doctor` setelah instalasi untuk memeriksa jenis dan versi instalasi Anda.

Setup khusus platform

Windows: Jalankan Claude Code secara native (memerlukan Git Bash) atau di dalam WSL. Baik WSL 1 maupun WSL 2 didukung, tetapi WSL 1 memiliki dukungan terbatas dan tidak mendukung fitur seperti sandboxing alat Bash.

Alpine Linux dan distribusi berbasis musl/uClibc lainnya:

Installer native di Alpine dan distribusi berbasis musl/uClibc lainnya memerlukan `libgcc`, `libstdc++`, dan `ripgrep`. Instal ini menggunakan manajer paket distribusi Anda, kemudian atur `USE_BUILTIN_RIPGREP=0`.

Di Alpine:

```
apk add libgcc libstdc++ ripgrep
```

Autentikasi

Untuk individu

1. **Paket Claude Pro atau Max** (direkomendasikan): Berlangganan paket Pro atau Max Claude untuk langganan terpadu yang mencakup Claude Code dan Claude di web. Kelola akun Anda di satu tempat dan masuk dengan akun Claude.ai Anda.
2. **Claude Console**: Terhubung melalui Claude Console dan selesaikan proses OAuth. Memerlukan penagihan aktif di Anthropic Console. Ruang kerja "Claude Code" dibuat secara otomatis untuk pelacakan penggunaan dan manajemen biaya. Anda tidak dapat membuat kunci API untuk ruang kerja Claude Code; itu didedikasikan secara eksklusif untuk penggunaan Claude Code.

Untuk tim dan organisasi

1. **Claude untuk Tim atau Enterprise** (direkomendasikan): Berlangganan Claude untuk Tim atau Claude untuk Enterprise untuk penagihan terpusat, manajemen tim, dan akses ke Claude Code dan Claude di web. Anggota tim masuk dengan akun Claude.ai mereka.

2. **Claude Console dengan penagihan tim**: Siapkan organisasi Claude Console bersama dengan penagihan tim. Undang anggota tim dan tetapkan peran untuk pelacakan penggunaan.
3. **Penyedia cloud**: Konfigurasikan Claude Code untuk menggunakan Amazon Bedrock, Google Vertex AI, atau Microsoft Foundry untuk penyebaran dengan infrastruktur cloud yang ada.

Instal versi tertentu

Installer native menerima nomor versi tertentu atau saluran rilis (`latest` atau `stable`). Saluran yang Anda pilih saat instalasi menjadi default untuk pembaruan otomatis. Lihat Konfigurasikan saluran rilis untuk informasi lebih lanjut.

Untuk menginstal versi terbaru (default):

macOS, Linux, WSL

```
curl -fsSL https://claude.ai/install.sh | bash
```

Windows PowerShell

```
irm https://claude.ai/install.ps1 | iex
```

Windows CMD

```
curl -fsSL https://claude.ai/install.cmd -o install.cmd && install.cmd && del
install.cmd
```

Untuk menginstal versi stabil:

macOS, Linux, WSL

```
curl -fsSL https://claude.ai/install.sh | bash -s stable
```

Windows PowerShell

```
& ([scriptblock]::Create((irm https://claude.ai/install.ps1))) stable
```

Windows CMD

```
curl -fsSL https://claude.ai/install.cmd -o install.cmd && install.cmd stable &&
del install.cmd
```

Untuk menginstal nomor versi tertentu:

macOS, Linux, WSL

```
curl -fsSL https://claude.ai/install.sh | bash -s 1.0.58
```

Windows PowerShell

```
& ([scriptblock]::Create((irm https://claude.ai/install.ps1))) 1.0.58
```

Windows CMD

```
curl -fsSL https://claude.ai/install.cmd -o install.cmd && install.cmd 1.0.58 &&
del install.cmd
```

Integritas biner dan penandatanganan kode

- Checksum SHA256 untuk semua platform dipublikasikan dalam manifes rilis, saat ini berlokasi di `https://storage.googleapis.com/claude-code-dist-86c565f3-f756-42ad-8dfa-d59b1c096819/claude-code-releases/{VERSION}/manifest.json` (contoh: ganti `{VERSION}` dengan `2.0.30`)
- Biner yang ditandatangani didistribusikan untuk platform berikut:
  - macOS: Ditandatangani oleh “Anthropic PBC” dan dinotarisi oleh Apple
  - Windows: Ditandatangani oleh “Anthropic, PBC”

Instalasi NPM (usang)

Instalasi NPM sudah usang. Gunakan metode instalasi native jika memungkinkan. Untuk memigrasikan instalasi npm yang ada ke native, jalankan `claude install`.

Instalasi npm global

```
npm install -g @anthropic-ai/claude-code
```

> **Warning:**
>
> JANGAN gunakan `sudo npm install -g` karena ini dapat menyebabkan masalah izin dan risiko keamanan. Jika Anda mengalami kesalahan izin, lihat pemecahan masalah kesalahan izin untuk solusi yang direkomendasikan.

Setup Windows

Opsi 1: Claude Code dalam WSL

- Baik WSL 1 maupun WSL 2 didukung
- WSL 2 mendukung sandboxing untuk keamanan yang ditingkatkan. WSL 1 tidak mendukung sandboxing.

Opsi 2: Claude Code di Windows native dengan Git Bash

- Memerlukan Git untuk Windows
- Untuk instalasi Git portabel, tentukan jalur ke `bash.exe` Anda:

```
$env:CLAUDE_CODE_GIT_BASH_PATH="C:\Program Files\Git\bin\bash.exe"
```

Perbarui Claude Code

Pembaruan otomatis

Claude Code secara otomatis tetap terbaru untuk memastikan Anda memiliki fitur terbaru dan perbaikan keamanan.

- **Pemeriksaan pembaruan**: Dilakukan saat startup dan secara berkala saat berjalan
- **Proses pembaruan**: Mengunduh dan menginstal secara otomatis di latar belakang
- **Notifikasi**: Anda akan melihat notifikasi ketika pembaruan diinstal
- **Menerapkan pembaruan**: Pembaruan berlaku saat Anda memulai Claude Code berikutnya

> **Note:**
>
> Instalasi Homebrew dan WinGet tidak auto-update. Gunakan `brew upgrade claude-code` atau `winget upgrade Anthropic.ClaudeCode` untuk memperbarui secara manual.
>
> **Masalah yang diketahui:** Claude Code mungkin memberi tahu Anda tentang pembaruan sebelum versi baru tersedia di manajer paket ini. Jika upgrade gagal, tunggu dan coba lagi nanti.

Konfigurasikan saluran rilis

Konfigurasikan saluran rilis mana yang diikuti Claude Code untuk pembaruan otomatis dan `claude update` dengan pengaturan `autoUpdatesChannel`:

- `"latest"` (default): Terima fitur baru segera setelah dirilis
- `"stable"`: Gunakan versi yang biasanya sekitar satu minggu lama, lewati rilis dengan regresi besar

Konfigurasikan ini melalui `/config` → **Saluran auto-update**, atau tambahkan ke file settings.json Anda:

```
{
  "autoUpdatesChannel": "stable"
}
```

Untuk penyebaran enterprise, Anda dapat memberlakukan saluran rilis yang konsisten di seluruh organisasi Anda menggunakan pengaturan terkelola.

Nonaktifkan pembaruan otomatis

Atur variabel lingkungan `DISABLE_AUTOUPDATER` di shell Anda atau file settings.json:

```
export DISABLE_AUTOUPDATER=1
```

Perbarui secara manual

```
claude update
```

Copot Claude Code

Jika Anda perlu mencopot Claude Code, ikuti instruksi untuk metode instalasi Anda.

Instalasi native

Hapus biner Claude Code dan file versi:

macOS, Linux, WSL:

```
rm -f ~/.local/bin/claude
rm -rf ~/.local/share/claude
```

Windows PowerShell:

```
Remove-Item -Path "$env:USERPROFILE\.local\bin\claude.exe" -Force
Remove-Item -Path "$env:USERPROFILE\.local\share\claude" -Recurse -Force
```

Windows CMD:

```
del "%USERPROFILE%\.local\bin\claude.exe"
rmdir /s /q "%USERPROFILE%\.local\share\claude"
```

Instalasi Homebrew

```
brew uninstall --cask claude-code
```

Instalasi WinGet

```
winget uninstall Anthropic.ClaudeCode
```

Instalasi NPM

```
npm uninstall -g @anthropic-ai/claude-code
```

Bersihkan file konfigurasi (opsional)

> **Warning:**
>
> Menghapus file konfigurasi akan menghapus semua pengaturan, alat yang diizinkan, konfigurasi server MCP, dan riwayat sesi Anda.

Untuk menghapus pengaturan Claude Code dan data cache:

macOS, Linux, WSL:

```
## Hapus pengaturan pengguna dan status
rm -rf ~/.claude
rm ~/.claude.json

## Hapus pengaturan khusus proyek (jalankan dari direktori proyek Anda)
rm -rf .claude
rm -f .mcp.json
```

Windows PowerShell:

```
## Hapus pengaturan pengguna dan status
Remove-Item -Path "$env:USERPROFILE\.claude" -Recurse -Force
Remove-Item -Path "$env:USERPROFILE\.claude.json" -Force

## Hapus pengaturan khusus proyek (jalankan dari direktori proyek Anda)
Remove-Item -Path ".claude" -Recurse -Force
Remove-Item -Path ".mcp.json" -Force
```

Windows CMD:

```
REM Hapus pengaturan pengguna dan status
rmdir /s /q "%USERPROFILE%\.claude"
del "%USERPROFILE%\.claude.json"

REM Hapus pengaturan khusus proyek (jalankan dari direktori proyek Anda)
rmdir /s /q ".claude"
del ".mcp.json"
```

Cara Kerja Claude Code

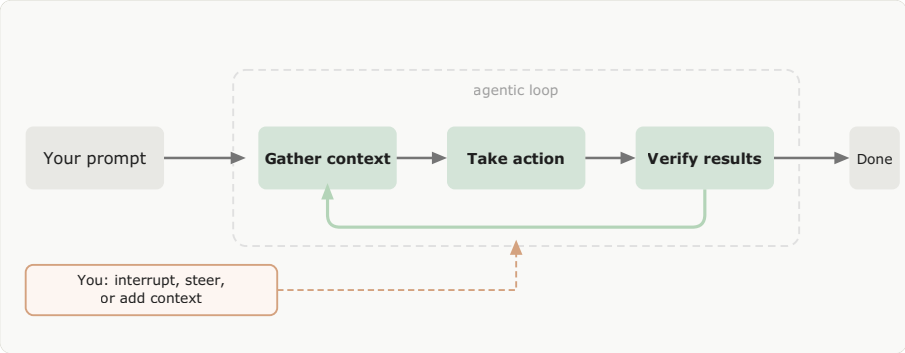
> *Pahami loop agentic, tools bawaan, dan bagaimana Claude Code berinteraksi dengan proyek Anda.*

Claude Code adalah asisten agentic yang berjalan di terminal Anda. Meskipun unggul dalam coding, Claude Code dapat membantu dengan apa pun yang dapat Anda lakukan dari command line: menulis dokumentasi, menjalankan build, mencari file, meneliti topik, dan banyak lagi.

Panduan ini mencakup arsitektur inti, kemampuan bawaan, dan tips untuk bekerja secara efektif dengan Claude Code. Untuk panduan langkah demi langkah, lihat Common workflows. Untuk fitur extensibility seperti skills, MCP, dan hooks, lihat Extend Claude Code.

Loop agentic

Ketika Anda memberikan tugas kepada Claude, Claude bekerja melalui tiga fase: **mengumpulkan konteks**, **mengambil tindakan**, dan **memverifikasi hasil**. Fase-fase ini berpadu bersama. Claude menggunakan tools di seluruh proses, baik mencari file untuk memahami kode Anda, mengedit untuk membuat perubahan, atau menjalankan test untuk memeriksa pekerjaannya.



Loop agentic: Prompt Anda mengarah ke Claude mengumpulkan konteks, mengambil tindakan, memverifikasi hasil, dan mengulangi sampai tugas selesai. Anda dapat mengganggu kapan saja.

Loop beradaptasi dengan apa yang Anda minta. Pertanyaan tentang codebase Anda mungkin hanya memerlukan pengumpulan konteks. Perbaikan bug melakukan siklus melalui ketiga fase berulang kali. Refactor mungkin melibatkan verifikasi ekstensif. Claude

memutuskan apa yang setiap langkah butuhkan berdasarkan apa yang dipelajarinya dari langkah sebelumnya, menghubungkan puluhan tindakan bersama-sama dan melakukan koreksi jalur di sepanjang jalan.

Anda juga bagian dari loop ini. Anda dapat mengganggu kapan saja untuk mengarahkan Claude ke arah yang berbeda, memberikan konteks tambahan, atau memintanya mencoba pendekatan yang berbeda. Claude bekerja secara otonom tetapi tetap responsif terhadap input Anda.

Loop agentic didukung oleh dua komponen: models yang bernalar dan tools yang bertindak. Claude Code berfungsi sebagai **agentic harness** di sekitar Claude: Claude Code menyediakan tools, manajemen konteks, dan lingkungan eksekusi yang mengubah model bahasa menjadi agen coding yang mampu.

Models

Claude Code menggunakan model Claude untuk memahami kode Anda dan bernalar tentang tugas. Claude dapat membaca kode dalam bahasa apa pun, memahami bagaimana komponen terhubung, dan mengetahui apa yang perlu berubah untuk mencapai tujuan Anda. Untuk tugas kompleks, Claude memecah pekerjaan menjadi langkah-langkah, menjalankannya, dan menyesuaikan berdasarkan apa yang dipelajarinya.

Multiple models tersedia dengan trade-off yang berbeda. Sonnet menangani sebagian besar tugas coding dengan baik. Opus memberikan penalaran yang lebih kuat untuk keputusan arsitektur yang kompleks. Beralih dengan `/model` selama sesi atau mulai dengan `claude --model <name>`.

Ketika panduan ini mengatakan “Claude memilih” atau “Claude memutuskan,” itu adalah model yang melakukan penalaran.

Tools

Tools adalah apa yang membuat Claude Code agentic. Tanpa tools, Claude hanya dapat merespons dengan teks. Dengan tools, Claude dapat bertindak: membaca kode Anda, mengedit file, menjalankan perintah, mencari web, dan berinteraksi dengan layanan eksternal. Setiap penggunaan tool mengembalikan informasi yang umpan balik ke dalam loop, menginformasikan keputusan Claude berikutnya.

Tools bawaan umumnya terbagi menjadi lima kategori, masing-masing mewakili jenis agency yang berbeda.

Kategori	Apa yang dapat dilakukan Claude
File operations	Membaca file, mengedit kode, membuat file baru, mengganti nama dan mengorganisir ulang
Search	Menemukan file berdasarkan pola, mencari konten dengan regex, menjelajahi codebase
Execution	Menjalankan perintah shell, memulai server, menjalankan test, menggunakan git
Web	Mencari web, mengambil dokumentasi, mencari pesan error
Code intelligence	Melihat error tipe dan peringatan setelah edit, melompat ke definisi, menemukan referensi (memerlukan code intelligence plugins)

Ini adalah kemampuan utama. Claude juga memiliki tools untuk spawning subagents, mengajukan pertanyaan kepada Anda, dan tugas orchestration lainnya. Lihat Tools available to Claude untuk daftar lengkap.

Claude memilih tools mana yang akan digunakan berdasarkan prompt Anda dan apa yang dipelajarinya di sepanjang jalan. Ketika Anda mengatakan "perbaiki test yang gagal," Claude mungkin:

1. Menjalankan test suite untuk melihat apa yang gagal
2. Membaca output error
3. Mencari file sumber yang relevan
4. Membaca file tersebut untuk memahami kode
5. Mengedit file untuk memperbaiki masalah
6. Menjalankan test lagi untuk memverifikasi

Setiap penggunaan tool memberikan Claude informasi baru yang menginformasikan langkah berikutnya. Ini adalah loop agentic dalam aksi.

Memperluas kemampuan dasar: Tools bawaan adalah fondasi. Anda dapat memperluas apa yang diketahui Claude dengan skills, terhubung ke layanan eksternal dengan MCP, mengotomatisasi workflow dengan hooks, dan mendelegasikan tugas ke subagents. Ekstensi ini membentuk lapisan di atas loop agentic inti. Lihat Extend Claude Code untuk panduan memilih ekstensi yang tepat untuk kebutuhan Anda.

Apa yang dapat diakses Claude

Panduan ini berfokus pada terminal. Claude Code juga berjalan di VS Code, JetBrains IDEs, dan lingkungan lainnya.

Ketika Anda menjalankan `claude` di direktori, Claude Code mendapatkan akses ke:

- **Proyek Anda.** File di direktori dan subdirektori Anda, ditambah file di tempat lain dengan izin Anda.
- **Terminal Anda.** Perintah apa pun yang dapat Anda jalankan: build tools, git, package managers, system utilities, scripts. Jika Anda dapat melakukannya dari command line, Claude juga dapat.
- **Status git Anda.** Branch saat ini, perubahan yang belum di-commit, dan riwayat commit terbaru.
- **CLAUDE.md Anda.** File markdown tempat Anda menyimpan instruksi khusus proyek, konvensi, dan konteks yang harus diketahui Claude setiap sesi.
- **Auto memory.** Pembelajaran yang disimpan Claude secara otomatis saat Anda bekerja, seperti pola proyek dan preferensi Anda. 200 baris pertama MEMORY.md dimuat di awal setiap sesi.
- **Ekstensi yang Anda konfigurasi.** MCP servers untuk layanan eksternal, skills untuk workflow, subagents untuk pekerjaan yang didelegasikan, dan Claude in Chrome untuk interaksi browser.

Karena Claude melihat seluruh proyek Anda, Claude dapat bekerja di seluruhnya. Ketika Anda meminta Claude untuk "perbaiki bug autentikasi," Claude mencari file yang relevan, membaca beberapa file untuk memahami konteks, membuat edit terkoordinasi di seluruhnya, menjalankan test untuk memverifikasi perbaikan, dan melakukan commit perubahan jika Anda meminta. Ini berbeda dari asisten kode inline yang hanya melihat file saat ini.

Lingkungan dan antarmuka

Loop agentic, tools, dan kemampuan yang dijelaskan di atas sama di mana pun Anda menggunakan Claude Code. Apa yang berubah adalah di mana kode dieksekusi dan bagaimana Anda berinteraksi dengannya.

Lingkungan eksekusi

Claude Code berjalan di tiga lingkungan, masing-masing dengan trade-off berbeda untuk di mana kode Anda dieksekusi.

Lingkungan	Di mana kode berjalan	Use case
Local	Mesin Anda	Default. Akses penuh ke file, tools, dan lingkungan Anda
Cloud	VM yang dikelola Anthropic	Mendelegasikan tugas, bekerja pada repo yang tidak Anda miliki secara lokal
Remote Control	Mesin Anda, dikendalikan dari browser	Gunakan web UI sambil menjaga semuanya tetap lokal

Antarmuka

Anda dapat mengakses Claude Code melalui terminal, desktop app, IDE extensions, claude.ai/code, Remote Control, Slack, dan CI/CD pipelines. Antarmuka menentukan bagaimana Anda melihat dan berinteraksi dengan Claude, tetapi loop agentic yang mendasarinya identik. Lihat Use Claude Code everywhere untuk daftar lengkap.

Bekerja dengan sesi

Claude Code menyimpan percakapan Anda secara lokal saat Anda bekerja. Setiap pesan, penggunaan tool, dan hasil disimpan, yang memungkinkan rewinding, resuming, dan forking sesi. Sebelum Claude membuat perubahan kode, Claude juga membuat snapshot file yang terpengaruh sehingga Anda dapat mengembalikan jika diperlukan.

Sesi bersifat independen. Setiap sesi baru dimulai dengan context window segar, tanpa riwayat percakapan dari sesi sebelumnya. Claude dapat mempertahankan pembelajaran di seluruh sesi menggunakan auto memory, dan Anda dapat menambahkan instruksi persisten Anda sendiri di CLAUDE.md.

Bekerja di seluruh branch

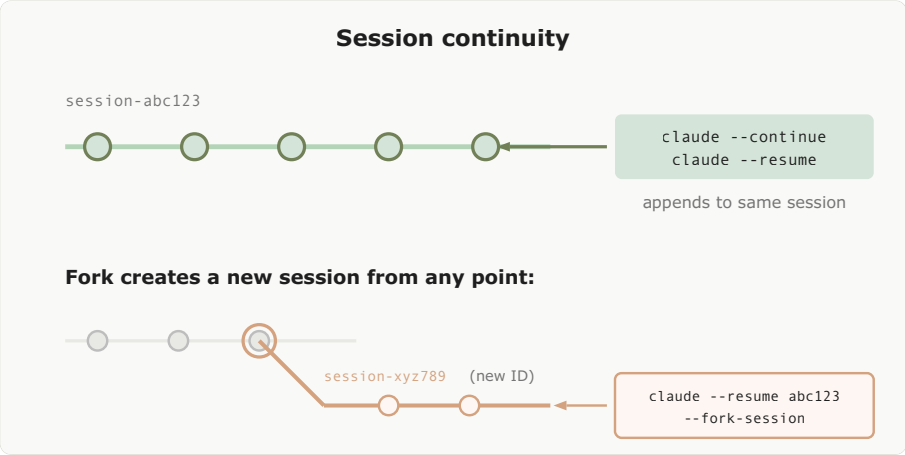
Setiap percakapan Claude Code adalah sesi yang terikat pada direktori saat ini Anda. Ketika Anda melanjutkan, Anda hanya melihat sesi dari direktori itu.

Claude melihat file branch saat ini Anda. Ketika Anda beralih branch, Claude melihat file branch baru, tetapi riwayat percakapan Anda tetap sama. Claude mengingat apa yang Anda diskusikan bahkan setelah beralih.

Karena sesi terikat pada direktori, Anda dapat menjalankan sesi Claude paralel dengan menggunakan git worktrees, yang membuat direktori terpisah untuk branch individual.

Resume atau fork sesi

Ketika Anda melanjutkan sesi dengan `claude --continue` atau `claude --resume`, Anda melanjutkan dari tempat Anda berhenti menggunakan ID sesi yang sama. Pesan baru ditambahkan ke percakapan yang ada. Riwayat percakapan lengkap Anda dipulihkan, tetapi izin yang dibatasi sesi tidak. Anda perlu menyetujui ulang.



Kontinuitas sesi: resume melanjutkan sesi yang sama, fork membuat branch baru dengan ID baru.

Untuk membuat cabang dan mencoba pendekatan berbeda tanpa mempengaruhi sesi asli, gunakan flag `--fork-session`:

```
claude --continue --fork-session
```

Ini membuat ID sesi baru sambil mempertahankan riwayat percakapan hingga titik itu. Sesi asli tetap tidak berubah. Seperti resume, sesi yang di-fork tidak mewarisi izin yang dibatasi sesi.

Sesi yang sama di beberapa terminal: Jika Anda melanjutkan sesi yang sama di beberapa terminal, kedua terminal menulis ke file sesi yang sama. Pesan dari keduanya saling tumpang tindih, seperti dua orang menulis di notebook yang sama. Tidak ada yang

rusak, tetapi percakapan menjadi kacau. Setiap terminal hanya melihat pesan miliknya sendiri selama sesi, tetapi jika Anda melanjutkan sesi itu nanti, Anda akan melihat semuanya saling tumpang tindih. Untuk pekerjaan paralel dari titik awal yang sama, gunakan `--fork-session` untuk memberikan setiap terminal sesi bersihnya sendiri.

Context window

Context window Claude menampung riwayat percakapan Anda, konten file, output perintah, CLAUDE.md, skill yang dimuat, dan instruksi sistem. Saat Anda bekerja, konteks terisi. Claude melakukan compacting secara otomatis, tetapi instruksi dari awal percakapan dapat hilang. Letakkan aturan persisten di CLAUDE.md, dan jalankan `/context` untuk melihat apa yang menggunakan ruang.

Ketika context terisi

Claude Code mengelola konteks secara otomatis saat Anda mendekati batas. Claude menghapus output tool yang lebih lama terlebih dahulu, kemudian merangkum percakapan jika diperlukan. Permintaan Anda dan snippet kode kunci dipertahankan; instruksi terperinci dari awal percakapan mungkin hilang. Letakkan aturan persisten di CLAUDE.md daripada mengandalkan riwayat percakapan.

Untuk mengontrol apa yang dipertahankan selama compacting, tambahkan bagian "Compact Instructions" ke CLAUDE.md atau jalankan `/compact` dengan fokus (seperti `/compact focus on the API changes`).

Jalankan `/context` untuk melihat apa yang menggunakan ruang. MCP servers menambahkan definisi tool ke setiap permintaan, jadi beberapa server dapat mengonsumsi konteks yang signifikan sebelum Anda mulai bekerja. Jalankan `/mcp` untuk memeriksa biaya per-server.

Kelola konteks dengan skills dan subagents

Selain compacting, Anda dapat menggunakan fitur lain untuk mengontrol apa yang dimuat ke dalam konteks.

Skills dimuat sesuai permintaan. Claude melihat deskripsi skill di awal sesi, tetapi konten lengkap hanya dimuat ketika skill digunakan. Untuk skill yang Anda panggil secara manual, atur `disable-model-invocation: true` untuk menjaga deskripsi keluar dari konteks sampai Anda membutuhkannya.

Subagents mendapatkan konteks segar mereka sendiri, sepenuhnya terpisah dari percakapan utama Anda. Pekerjaan mereka tidak membengkak konteks Anda. Ketika selesai, mereka mengembalikan ringkasan. Isolasi ini adalah alasan mengapa subagents membantu dengan sesi panjang.

Lihat context costs untuk apa yang setiap fitur biayai, dan reduce token usage untuk tips mengelola konteks.

Tetap aman dengan checkpoints dan permissions

Claude memiliki dua mekanisme keamanan: checkpoints memungkinkan Anda membatalkan perubahan file, dan permissions mengontrol apa yang dapat dilakukan Claude tanpa bertanya.

Batalkan perubahan dengan checkpoints

Setiap edit file dapat dibalikkan. Sebelum Claude mengedit file apa pun, Claude membuat snapshot konten saat ini. Jika ada yang salah, tekan `Esc` dua kali untuk kembali ke status sebelumnya, atau minta Claude untuk membatalkan.

Checkpoints bersifat lokal untuk sesi Anda, terpisah dari git. Mereka hanya mencakup perubahan file. Tindakan yang mempengaruhi sistem jarak jauh (database, API, deployment) tidak dapat di-checkpoint, itulah mengapa Claude bertanya sebelum menjalankan perintah dengan efek samping eksternal.

Kontrol apa yang dapat dilakukan Claude

Tekan `Shift+Tab` untuk melakukan siklus melalui mode permission:

- **Default**: Claude bertanya sebelum edit file dan perintah shell
- **Auto-accept edits**: Claude mengedit file tanpa bertanya, masih bertanya untuk perintah
- **Plan mode**: Claude hanya menggunakan tools read-only, membuat rencana yang dapat Anda setujui sebelum eksekusi

Anda juga dapat mengizinkan perintah tertentu di `.claude/settings.json` sehingga Claude tidak bertanya setiap kali. Ini berguna untuk perintah terpercaya seperti `npm test` atau `git status`. Settings dapat dibatasi dari kebijakan organisasi-luas hingga preferensi pribadi. Lihat Permissions untuk detail.

Bekerja secara efektif dengan Claude Code

Tips ini membantu Anda mendapatkan hasil yang lebih baik dari Claude Code.

Minta bantuan Claude Code

Claude Code dapat mengajarkan Anda cara menggunakannya. Ajukan pertanyaan seperti "bagaimana cara mengatur hooks?" atau "apa cara terbaik untuk menyusun CLAUDE.md saya?" dan Claude akan menjelaskan.

Perintah bawaan juga memandu Anda melalui setup:

- `/init` memandu Anda membuat CLAUDE.md untuk proyek Anda
- `/agents` membantu Anda mengonfigurasi subagents kustom
- `/doctor` mendiagnosis masalah umum dengan instalasi Anda

Ini adalah percakapan

Claude Code bersifat conversational. Anda tidak memerlukan prompt yang sempurna. Mulai dengan apa yang Anda inginkan, kemudian perbaiki:

```
Perbaiki bug login
```

[Claude menyelidiki, mencoba sesuatu]

```
Itu tidak cukup benar. Masalahnya ada di session handling.
```

[Claude menyesuaikan pendekatan]

Ketika upaya pertama tidak benar, Anda tidak memulai dari awal. Anda melakukan iterasi.

Ganggu dan arahkan

Anda dapat mengganggu Claude kapan saja. Jika Claude sedang menuju jalur yang salah, cukup ketik koreksi Anda dan tekan Enter. Claude akan berhenti melakukan apa yang sedang dilakukan dan menyesuaikan pendekatannya berdasarkan input Anda. Anda tidak harus menunggu sampai selesai atau memulai dari awal.

Jadilah spesifik di awal

Semakin presisi prompt awal Anda, semakin sedikit koreksi yang Anda butuhkan. Referensikan file tertentu, sebutkan batasan, dan tunjukkan pola contoh.

```
Alur checkout rusak untuk pengguna dengan kartu yang kadaluarsa.
Periksa src/payments/ untuk masalahnya, terutama token refresh.
Tulis test yang gagal terlebih dahulu, kemudian perbaiki.
```

Prompt yang samar-samar berfungsi, tetapi Anda akan menghabiskan lebih banyak waktu untuk mengarahkan. Prompt spesifik seperti yang di atas sering berhasil pada upaya pertama.

Berikan Claude sesuatu untuk diverifikasi

Claude berkinerja lebih baik ketika dapat memeriksa pekerjaannya sendiri. Sertakan test case, tempel screenshot UI yang diharapkan, atau tentukan output yang Anda inginkan.

```
Implementasikan validateEmail. Test case: 'user@example.com' → true,
'invalid' → false, 'user@.com' → false. Jalankan test setelahnya.
```

Untuk pekerjaan visual, tempel screenshot desain dan minta Claude membandingkan implementasinya dengannya.

Jelajahi sebelum mengimplementasikan

Untuk masalah kompleks, pisahkan penelitian dari coding. Gunakan plan mode (`Shift+Tab` dua kali) untuk menganalisis codebase terlebih dahulu:

```
Baca src/auth/ dan pahami bagaimana kami menangani session.
Kemudian buat rencana untuk menambahkan dukungan OAuth.
```

Tinjau rencana, perbaiki melalui percakapan, kemudian biarkan Claude mengimplementasikan. Pendekatan dua fase ini menghasilkan hasil yang lebih baik daripada langsung melompat ke kode.

Delegasikan, jangan mendikte

Pikirkan mendelegasikan kepada rekan kerja yang mampu. Berikan konteks dan arah, kemudian percayai Claude untuk mengetahui detail:

```
Alur checkout rusak untuk pengguna dengan kartu yang kadaluarsa.
Kode yang relevan ada di src/payments/. Bisakah Anda menyelidiki dan
memperbaikinya?
```

Anda tidak perlu menentukan file mana yang harus dibaca atau perintah mana yang harus dijalankan. Claude mengetahui itu.

Apa selanjutnya

- Perluas dengan fitur Tambahkan Skills, koneksi MCP, dan perintah kustom
- Common workflows Panduan langkah demi langkah untuk tugas khas

Part 2: Core Usage

Perluas Claude Code

> *Pahami kapan menggunakan CLAUDE.md, Skills, subagents, hooks, MCP, dan plugins.*

Claude Code menggabungkan model yang bernalar tentang kode Anda dengan alat bawaan untuk operasi file, pencarian, eksekusi, dan akses web. Alat bawaan mencakup sebagian besar tugas pengkodean. Panduan ini mencakup lapisan ekstensi: fitur yang Anda tambahkan untuk menyesuaikan apa yang Claude ketahui, menghubungkannya ke layanan eksternal, dan mengotomatisasi alur kerja.

> **Note:**
>
> Untuk cara loop agentic inti bekerja, lihat Cara Claude Code Bekerja.

Baru di Claude Code? Mulai dengan CLAUDE.md untuk konvensi proyek. Tambahkan ekstensi lain sesuai kebutuhan Anda.

Ikhtisar

Ekstensi terhubung ke bagian berbeda dari loop agentic:

- **CLAUDE.md** menambahkan konteks persisten yang Claude lihat setiap sesi
- **Skills** menambahkan pengetahuan yang dapat digunakan kembali dan alur kerja yang dapat dipanggil
- **MCP** menghubungkan Claude ke layanan dan alat eksternal
- **Subagents** menjalankan loop mereka sendiri dalam konteks terisolasi, mengembalikan ringkasan
- **Agent teams** mengoordinasikan beberapa sesi independen dengan tugas bersama dan pesan peer-to-peer
- **Hooks** berjalan di luar loop sepenuhnya sebagai skrip deterministik
- **Plugins** dan **marketplaces** mengemas dan mendistribusikan fitur-fitur ini

Skills adalah ekstensi paling fleksibel. Skill adalah file markdown yang berisi pengetahuan, alur kerja, atau instruksi. Anda dapat memanggil skills dengan perintah seperti `/deploy`, atau Claude dapat memuatnya secara otomatis ketika relevan. Skills dapat berjalan dalam percakapan Anda saat ini atau dalam konteks terisolasi melalui subagents.

Cocokkan fitur dengan tujuan Anda

Fitur berkisar dari konteks yang selalu aktif yang Claude lihat setiap sesi, hingga kemampuan on-demand yang dapat Anda atau Claude panggil, hingga otomasi latar belakang yang berjalan pada acara tertentu. Tabel di bawah menunjukkan apa yang tersedia dan kapan masing-masing masuk akal.

Fitur	Apa yang dilakukannya	Kapan menggunakannya	Contoh
CLAUDE.md	Konteks persisten dimuat setiap percakapan	Konvensi proyek, aturan "selalu lakukan X"	"Gunakan pnpm, bukan npm. Jalankan tes sebelum commit."
Skill	Instruksi, pengetahuan, dan alur kerja yang dapat digunakan Claude	Konten yang dapat digunakan kembali, dokumen referensi, tugas yang dapat diulang	`/deploy` menjalankan daftar periksa deployment Anda; skill dokumen API dengan pola endpoint
Subagent	Konteks eksekusi terisolasi yang mengembalikan hasil ringkasan	Isolasi konteks, tugas paralel, pekerja khusus	Tugas penelitian yang membaca banyak file tetapi hanya mengembalikan temuan kunci
Agent teams	Mengoordinasikan beberapa sesi Claude Code independen	Penelitian paralel, pengembangan fitur baru, debugging dengan hipotesis bersaing	Spawn reviewer untuk memeriksa keamanan, performa, dan tes secara bersamaan
MCP	Terhubung ke layanan eksternal	Data atau tindakan eksternal	Kueri database Anda, posting ke Slack, kontrol browser
Hook	Skrip deterministik yang berjalan pada acara	Otomasi yang dapat diprediksi, tidak ada LLM yang terlibat	Jalankan ESLint setelah setiap edit file

Plugins adalah lapisan pengemasan. Plugin menggabungkan skills, hooks, subagents, dan MCP servers menjadi satu unit yang dapat diinstal. Plugin skills memiliki namespace (seperti `/my-plugin:review`) sehingga beberapa plugin dapat hidup berdampingan. Gunakan plugins ketika Anda ingin menggunakan kembali setup yang sama di beberapa repositori atau mendistribusikan ke orang lain melalui **marketplace**.

Bandingkan fitur serupa

Beberapa fitur dapat terlihat serupa. Berikut cara membedakannya.

Skill vs Subagent

Skills dan subagents menyelesaikan masalah yang berbeda:

- **Skills** adalah konten yang dapat digunakan kembali yang dapat Anda muat ke konteks apa pun
- **Subagents** adalah pekerja terisolasi yang berjalan terpisah dari percakapan utama Anda

Aspek	Skill	Subagent
Apa itu	Instruksi, pengetahuan, atau alur kerja yang dapat digunakan kembali	Pekerja terisolasi dengan konteksnya sendiri
Manfaat utama	Bagikan konten di seluruh konteks	Isolasi konteks. Pekerjaan terjadi secara terpisah, hanya ringkasan yang kembali
Terbaik untuk	Materi referensi, alur kerja yang dapat dipanggil	Tugas yang membaca banyak file, pekerjaan paralel, pekerja khusus

Skills dapat berupa referensi atau tindakan. Skills referensi memberikan pengetahuan yang Claude gunakan sepanjang sesi Anda (seperti panduan gaya API Anda). Skills tindakan memberi tahu Claude untuk melakukan sesuatu yang spesifik (seperti `/deploy` yang menjalankan alur kerja deployment Anda).

Gunakan subagent ketika Anda membutuhkan isolasi konteks atau ketika jendela konteks Anda penuh. Subagent mungkin membaca puluhan file atau menjalankan pencarian ekstensif, tetapi percakapan utama Anda hanya menerima ringkasan. Karena pekerjaan subagent tidak mengonsumsi konteks utama Anda, ini juga berguna ketika Anda tidak memerlukan pekerjaan perantara untuk tetap terlihat. Subagents kustom dapat memiliki instruksi mereka sendiri dan dapat memuat skills sebelumnya.

Mereka dapat digabungkan. Subagent dapat memuat skills tertentu sebelumnya (field `skills:`). Skill dapat berjalan dalam konteks terisolasi menggunakan `context: fork`. Lihat Skills untuk detail.

CLAUDE.md vs Skill

Keduanya menyimpan instruksi, tetapi mereka dimuat secara berbeda dan melayani tujuan yang berbeda.

Aspek	CLAUDE.md	Skill
Dimuat	Setiap sesi, secara otomatis	On demand
Dapat menyertakan file	Ya, dengan impor `@path`	Ya, dengan impor `@path`
Dapat memicu alur kerja	Tidak	Ya, dengan `/<name>`
Terbaik untuk	Aturan "selalu lakukan X"	Materi referensi, alur kerja yang dapat dipanggil

Letakkan di CLAUDE.md jika Claude harus selalu mengetahuinya: konvensi pengkodean, perintah build, struktur proyek, aturan "jangan pernah lakukan X".

Letakkan di skill jika itu materi referensi yang Claude butuhkan kadang-kadang (dokumen API, panduan gaya) atau alur kerja yang Anda picu dengan `/<name>` (deploy, review, release).

Aturan praktis: Jaga CLAUDE.md di bawah 200 baris. Jika berkembang, pindahkan konten referensi ke skills atau pisahkan ke file `.claude/rules/`.

CLAUDE.md vs Rules vs Skills

Ketiganya menyimpan instruksi, tetapi mereka dimuat secara berbeda:

Aspek	CLAUDE.md	`.claude/rules/`	Skill
Dimuat	Setiap sesi	Setiap sesi, atau ketika file yang cocok dibuka	On demand, ketika dipanggil atau relevan
Cakupan	Seluruh proyek	Dapat dibatasi ke jalur file	Spesifik tugas

Aspek	CLAUDE.md	`.claude/rules/`	Skill
Terbaik untuk	Konvensi inti dan perintah build	Panduan spesifik bahasa atau direktori	Materi referensi, alur kerja yang dapat diulang

Gunakan CLAUDE.md untuk instruksi yang setiap sesi butuhkan: perintah build, konvensi tes, arsitektur proyek.

Gunakan rules untuk menjaga CLAUDE.md tetap fokus. Rules dengan `paths` frontmatter hanya dimuat ketika Claude bekerja dengan file yang cocok, menghemat konteks.

Gunakan skills untuk konten yang Claude hanya butuhkan kadang-kadang, seperti dokumentasi API atau daftar periksa deployment yang Anda picu dengan `/<name>`.

Subagent vs Agent team

Keduanya melakukan paralelisasi pekerjaan, tetapi mereka secara arsitektur berbeda:

- **Subagents** berjalan di dalam sesi Anda dan melaporkan hasil kembali ke konteks utama Anda
- **Agent teams** adalah sesi Claude Code independen yang berkomunikasi satu sama lain

Aspek	Subagent	Agent team
Konteks	Jendela konteks sendiri; hasil kembali ke pemanggil	Jendela konteks sendiri; sepenuhnya independen
Komunikasi	Melaporkan hasil kembali ke agen utama saja	Rekan kerja saling mengirim pesan secara langsung
Koordinasi	Agen utama mengelola semua pekerjaan	Daftar tugas bersama dengan koordinasi diri
Terbaik untuk	Tugas terfokus di mana hanya hasil yang penting	Pekerjaan kompleks yang memerlukan diskusi dan kolaborasi
Biaya token	Lebih rendah: hasil diringkas kembali ke konteks utama	Lebih tinggi: setiap rekan kerja adalah instans Claude terpisah

Gunakan subagent ketika Anda membutuhkan pekerja cepat dan terfokus: teliti pertanyaan, verifikasi klaim, tinjau file. Subagent melakukan pekerjaan dan mengembalikan ringkasan. Percakapan utama Anda tetap bersih.

Gunakan agent team ketika rekan kerja perlu berbagi temuan, menantang satu sama lain, dan berkoordinasi secara independen. Agent teams terbaik untuk penelitian dengan hipotesis bersaing, tinjauan kode paralel, dan pengembangan fitur baru di mana setiap rekan kerja memiliki bagian terpisah.

Titik transisi: Jika Anda menjalankan subagents paralel tetapi mencapai batas konteks, atau jika subagents Anda perlu berkomunikasi satu sama lain, agent teams adalah langkah alami berikutnya.

> **Note:**
>
> Agent teams bersifat eksperimental dan dinonaktifkan secara default. Lihat agent teams untuk setup dan batasan saat ini.

MCP vs Skill

MCP menghubungkan Claude ke layanan eksternal. Skills memperluas apa yang Claude ketahui, termasuk cara menggunakan layanan tersebut secara efektif.

Aspek	MCP	Skill
Apa itu	Protokol untuk terhubung ke layanan eksternal	Pengetahuan, alur kerja, dan materi referensi
Menyediakan	Akses alat dan data	Pengetahuan, alur kerja, materi referensi
Contoh	Integrasi Slack, kueri database, kontrol browser	Daftar periksa tinjauan kode, alur kerja deploy, panduan gaya API

Ini menyelesaikan masalah yang berbeda dan bekerja dengan baik bersama:

MCP memberi Claude kemampuan untuk berinteraksi dengan sistem eksternal. Tanpa MCP, Claude tidak dapat mengueri database Anda atau posting ke Slack.

Skills memberi Claude pengetahuan tentang cara menggunakan alat tersebut secara efektif, ditambah alur kerja yang dapat Anda picu dengan `/<name>`. Skill mungkin menyertakan skema database tim Anda dan pola kueri, atau alur kerja `/post-to-slack` dengan aturan pemformatan pesan tim Anda.

Contoh: Server MCP menghubungkan Claude ke database Anda. Skill mengajarkan Claude model data Anda, pola kueri umum, dan tabel mana yang digunakan untuk tugas berbeda.

Pahami bagaimana fitur berlapis

Fitur dapat didefinisikan di beberapa tingkat: seluruh pengguna, per-proyek, melalui plugins, atau melalui kebijakan terkelola. Anda juga dapat menyarangkan file CLAUDE.md di subdirektori atau menempatkan skills di paket tertentu dari monorepo. Ketika fitur yang sama ada di beberapa tingkat, berikut cara mereka berlapis:

- **File CLAUDE.md** bersifat aditif: semua tingkat berkontribusi konten ke konteks Claude secara bersamaan. File dari direktori kerja Anda dan di atas dimuat saat peluncuran; subdirektori dimuat saat Anda bekerja di dalamnya. Ketika instruksi bertentangan, Claude menggunakan penilaian untuk merekonsiliasi mereka, dengan instruksi yang lebih spesifik biasanya mengambil alih. Lihat bagaimana file CLAUDE.md dimuat.
- **Skills dan subagents** menimpa berdasarkan nama: ketika nama yang sama ada di beberapa tingkat, satu definisi menang berdasarkan prioritas (terkelola > pengguna > proyek untuk skills; terkelola > bendera CLI > proyek > pengguna > plugin untuk subagents). Plugin skills adalah namespaced untuk menghindari konflik. Lihat penemuan skill dan cakupan subagent.
- **Server MCP** menimpa berdasarkan nama: lokal > proyek > pengguna. Lihat cakupan MCP.
- **Hooks** bergabung: semua hook terdaftar api untuk acara pencocokan mereka terlepas dari sumber. Lihat hooks.

Gabungkan fitur

Setiap ekstensi menyelesaikan masalah yang berbeda: CLAUDE.md menangani konteks yang selalu aktif, skills menangani pengetahuan dan alur kerja on-demand, MCP menangani koneksi eksternal, subagents menangani isolasi, dan hooks menangani otomasi. Setup nyata menggabungkan mereka berdasarkan alur kerja Anda.

Misalnya, Anda mungkin menggunakan CLAUDE.md untuk konvensi proyek, skill untuk alur kerja deployment Anda, MCP untuk terhubung ke database Anda, dan hook untuk menjalankan linting setelah setiap edit. Setiap fitur menangani apa yang terbaik.

Pola	Cara kerjanya	Contoh
Skill + MCP	MCP menyediakan koneksi; skill mengajarkan Claude cara menggunakannya dengan baik	MCP terhubung ke database Anda, skill mendokumentasikan skema dan pola kueri Anda

Pola	Cara kerjanya	Contoh
Skill + Subagent	Skill menspawn subagents untuk pekerjaan paralel	Skill `/audit` memulai subagents keamanan, performa, dan gaya yang bekerja dalam konteks terisolasi
CLAUDE.md + Skills	CLAUDE.md menyimpan aturan yang selalu aktif; skills menyimpan materi referensi yang dimuat on-demand	CLAUDE.md mengatakan "ikuti konvensi API kami," skill berisi panduan gaya API lengkap
Hook + MCP	Hook memicu tindakan eksternal melalui MCP	Hook pasca-edit mengirim notifikasi Slack ketika Claude memodifikasi file kritis

Pahami biaya konteks

Setiap fitur yang Anda tambahkan mengonsumsi beberapa konteks Claude. Terlalu banyak dapat mengisi jendela konteks Anda, tetapi juga dapat menambah kebisingan yang membuat Claude kurang efektif; skills mungkin tidak dipicu dengan benar, atau Claude mungkin kehilangan jejak konvensi Anda. Memahami trade-off ini membantu Anda membangun setup yang efektif.

Biaya konteks berdasarkan fitur

Setiap fitur memiliki strategi pemuatan dan biaya konteks yang berbeda:

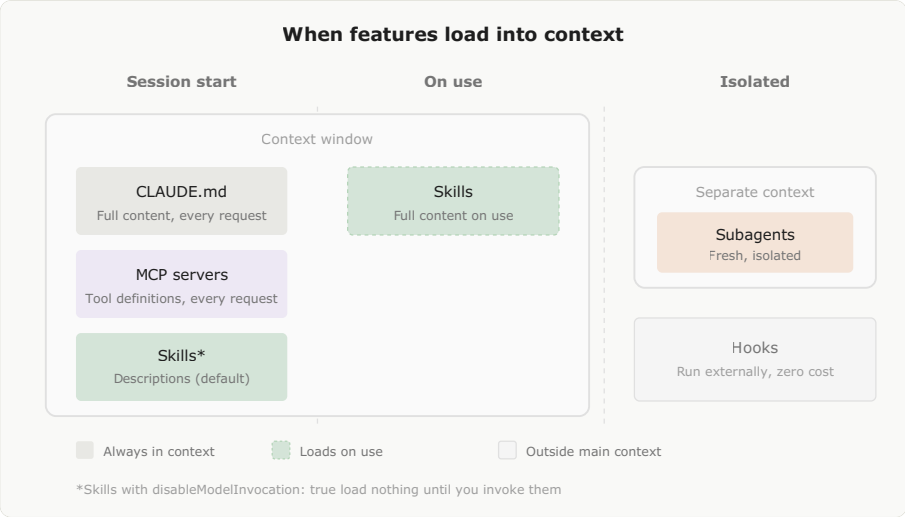
Fitur	Kapan dimuat	Apa yang dimuat	Biaya konteks
CLAUDE.md	Awal sesi	Konten penuh	Setiap permintaan
Skills	Awal sesi + ketika digunakan	Deskripsi di awal, konten penuh ketika digunakan	Rendah (deskripsi setiap permintaan)*
Server MCP	Awal sesi	Semua definisi alat dan skema	Setiap permintaan
Subagents	Ketika dispawn	Konteks segar dengan skills yang ditentukan	Terisolasi dari sesi utama

Fitur	Kapan dimuat	Apa yang dimuat	Biaya konteks
Hooks	Saat dipicu	Tidak ada (berjalan secara eksternal)	Nol, kecuali hook mengembalikan konteks tambahan

*Secara default, deskripsi skill dimuat saat awal sesi sehingga Claude dapat memutuskan kapan menggunakannya. Atur `disable-model-invocation: true` di frontmatter skill untuk menyembunyikannya dari Claude sepenuhnya sampai Anda memanggilnya secara manual. Ini mengurangi biaya konteks menjadi nol untuk skills yang hanya Anda picu sendiri.

Pahami bagaimana fitur dimuat

Setiap fitur dimuat pada titik berbeda dalam sesi Anda. Tab di bawah menjelaskan kapan masing-masing dimuat dan apa yang masuk ke konteks.



Pemuatan konteks: CLAUDE.md dan MCP dimuat saat awal sesi dan tetap di setiap permintaan. Skills memuat deskripsi di awal, konten penuh saat invokasi. Subagents mendapat konteks terisolasi. Hooks berjalan secara eksternal.

CLAUDE.md

Kapan: Awal sesi

Apa yang dimuat: Konten penuh semua file CLAUDE.md (tingkat terkelola, pengguna, dan proyek).

Warisan: Claude membaca file CLAUDE.md dari direktori kerja Anda hingga ke root, dan menemukan yang tersarang di subdirektori saat mengakses file tersebut. Lihat Bagaimana file CLAUDE.md dimuat untuk detail.

> **Tip:**
>
> Jaga CLAUDE.md di bawah ~500 baris. Pindahkan materi referensi ke skills, yang dimuat on-demand.

Skills

Skills adalah kemampuan tambahan dalam toolkit Claude. Mereka dapat berupa materi referensi (seperti panduan gaya API) atau alur kerja yang dapat dipanggil yang Anda picu dengan `/<name>` (seperti `/deploy`). Claude Code dilengkapi dengan skills bundel seperti `/simplify`, `/batch`, dan `/debug` yang bekerja langsung. Anda juga dapat membuat yang Anda sendiri. Claude menggunakan skills ketika sesuai, atau Anda dapat memanggil satu secara langsung.

Kapan: Tergantung pada konfigurasi skill. Secara default, deskripsi dimuat saat awal sesi dan konten penuh dimuat ketika digunakan. Untuk skills hanya pengguna (`disable-model-invocation: true`), tidak ada yang dimuat sampai Anda memanggilnya.

Apa yang dimuat: Untuk skills yang dapat dipanggil model, Claude melihat nama dan deskripsi di setiap permintaan. Ketika Anda memanggil skill dengan `/<name>` atau Claude memuatnya secara otomatis, konten penuh dimuat ke percakapan Anda.

Bagaimana Claude memilih skills: Claude mencocokkan tugas Anda terhadap deskripsi skill untuk memutuskan mana yang relevan. Jika deskripsi samar atau tumpang tindih, Claude mungkin memuat skill yang salah atau melewatkan yang akan membantu. Untuk memberi tahu Claude menggunakan skill tertentu, panggilnya dengan `/<name>`. Skills dengan `disable-model-invocation: true` tidak terlihat oleh Claude sampai Anda memanggilnya.

Biaya konteks: Rendah sampai digunakan. Skills hanya pengguna memiliki biaya nol sampai dipanggil.

Di subagents: Skills bekerja berbeda di subagents. Alih-alih pemuatan on-demand, skills yang dilewatkan ke subagent sepenuhnya dimuat sebelumnya ke konteksnya saat peluncuran. Subagents tidak mewarisi skills dari sesi utama; Anda harus menentukannya secara eksplisit.

> **Tip:**

> Gunakan `disable-model-invocation: true` untuk skills dengan efek samping. Ini menghemat konteks dan memastikan hanya Anda yang memicunya.

Server MCP

Kapan: Awal sesi.

Apa yang dimuat: Semua definisi alat dan skema JSON dari server yang terhubung.

Biaya konteks: Pencarian alat (diaktifkan secara default) memuat alat MCP hingga 10% konteks dan menunda sisanya sampai diperlukan.

Catatan keandalan: Koneksi MCP dapat gagal secara diam-diam di tengah sesi. Jika server terputus, alatnya hilang tanpa peringatan. Claude mungkin mencoba menggunakan alat yang tidak lagi ada. Jika Anda melihat Claude gagal menggunakan alat MCP yang sebelumnya dapat diakses, periksa koneksi dengan `/mcp`.

> **Tip:**
>
> Jalankan `/mcp` untuk melihat biaya token per server. Putuskan server yang tidak Anda gunakan secara aktif.

Subagents

Kapan: On demand, ketika Anda atau Claude menspawn satu untuk tugas.

Apa yang dimuat: Konteks segar dan terisolasi yang berisi:

- Prompt sistem (dibagikan dengan induk untuk efisiensi cache)
- Konten penuh skills yang tercantum di field `skills:` agen
- CLAUDE.md dan status git (diwarisi dari induk)
- Apa pun konteks yang agen utama lewatkan dalam prompt

Biaya konteks: Terisolasi dari sesi utama. Subagents tidak mewarisi riwayat percakapan Anda atau skills yang dipanggil.

> **Tip:**
>
> Gunakan subagents untuk pekerjaan yang tidak memerlukan konteks percakapan penuh Anda. Isolasi mereka mencegah mengembang sesi utama Anda.

Hooks

Kapan: Saat dipicu. Hooks api pada acara siklus hidup tertentu seperti eksekusi alat, batas sesi, pengajuan prompt, permintaan izin, dan pemadatan. Lihat Hooks untuk daftar lengkap.

Apa yang dimuat: Tidak ada secara default. Hooks berjalan sebagai skrip eksternal.

Biaya konteks: Nol, kecuali hook mengembalikan output yang ditambahkan sebagai pesan ke percakapan Anda.

> **Tip:**
>
> Hooks ideal untuk efek samping (linting, logging) yang tidak perlu mempengaruhi konteks Claude.

Pelajari lebih lanjut

Setiap fitur memiliki panduan sendiri dengan instruksi setup, contoh, dan opsi konfigurasi.

- CLAUDE.md Simpan konteks proyek, konvensi, dan instruksi
- Skills Berikan Claude keahlian domain dan alur kerja yang dapat digunakan kembali
- Subagents Alihkan pekerjaan ke konteks terisolasi
- Agent teams Koordinasikan beberapa sesi yang bekerja secara paralel
- MCP Hubungkan Claude ke layanan eksternal
- Hooks Otomatisasi alur kerja dengan hooks
- Plugins Bundel dan bagikan set fitur
- Marketplaces Host dan distribusikan koleksi plugin

Mode interaktif

> *Referensi lengkap untuk pintasan keyboard, mode input, dan fitur interaktif dalam sesi Claude Code.*

Pintasan keyboard

Note:

Pintasan keyboard mungkin berbeda menurut platform dan terminal. Tekan `?` untuk melihat pintasan yang tersedia untuk lingkungan Anda.

Pengguna macOS: Pintasan tombol Option/Alt (`Alt+B` , `Alt+F` , `Alt+Y` , `Alt+M` , `Alt+P`) memerlukan konfigurasi Option sebagai Meta di terminal Anda:

- **iTerm2**: settings → Profiles → Keys → atur Left/Right Option key ke “Esc+”
- **Terminal.app**: settings → Profiles → Keyboard → centang “Use Option as Meta Key”
- **VS Code**: settings → Profiles → Keys → atur Left/Right Option key ke “Esc+”

Lihat Konfigurasi terminal untuk detail.

Kontrol umum

Pintasan	Deskripsi	Konteks
`Ctrl+C`	Batalkan input atau generasi saat ini	Interupsi standar
`Ctrl+F`	Matikan semua agen latar belakang. Tekan dua kali dalam 3 detik untuk mengonfirmasi	Kontrol agen latar belakang
`Ctrl+D`	Keluar dari sesi Claude Code	Sinyal EOF
`Ctrl+G`	Buka di editor teks default	Edit prompt atau respons kustom Anda di editor teks default
`Ctrl+L`	Bersihkan layar terminal	Menjaga riwayat percakapan

Pintasan	Deskripsi	Konteks
`Ctrl+O`	Alihkan output verbose	Menampilkan penggunaan alat dan eksekusi terperinci
`Ctrl+R`	Pencarian riwayat perintah terbalik	Cari melalui perintah sebelumnya secara interaktif
`Ctrl+V` atau `Cmd+V` (iTerm2) atau `Alt+V` (Windows)	Tempel gambar dari clipboard	Menempel gambar atau jalur ke file gambar
`Ctrl+B`	Tugas yang berjalan di latar belakang	Menjalankan perintah bash dan agen di latar belakang. Pengguna Tmux tekan dua kali
`Ctrl+T`	Alihkan daftar tugas	Tampilkan atau sembunyikan daftar tugas di area status terminal
`Left/Right arrows`	Siklus melalui tab dialog	Navigasi antar tab dalam dialog izin dan menu
`Up/Down arrows`	Navigasi riwayat perintah	Ingat kembali input sebelumnya
`Esc` + `Esc`	Putar ulang atau ringkas	Pulihkan kode dan/atau percakapan ke titik sebelumnya, atau ringkas dari pesan yang dipilih
`Shift+Tab` atau `Alt+M` (beberapa konfigurasi)	Alihkan mode izin	Beralih antara Mode Auto-Accept, Plan Mode, dan mode normal.
`Option+P` (macOS) atau `Alt+P` (Windows/Linux)	Alihkan model	Alihkan model tanpa menghapus prompt Anda
`Option+T` (macOS) atau `Alt+T` (Windows/Linux)	Alihkan pemikiran yang diperluas	Aktifkan atau nonaktifkan mode pemikiran yang diperluas. Jalankan `/terminal-setup` terlebih dahulu untuk mengaktifkan pintasan ini

Pengeditan teks

Pintasan	Deskripsi	Konteks
`Ctrl+K`	Hapus hingga akhir baris	Menyimpan teks yang dihapus untuk ditempel
`Ctrl+U`	Hapus seluruh baris	Menyimpan teks yang dihapus untuk ditempel
`Ctrl+Y`	Tempel teks yang dihapus	Tempel teks yang dihapus dengan `Ctrl+K` atau `Ctrl+U`
`Alt+Y` (setelah `Ctrl+Y`)	Siklus riwayat tempel	Setelah menempel, siklus melalui teks yang dihapus sebelumnya. Memerlukan Option sebagai Meta di macOS
`Alt+B`	Pindahkan kursor kembali satu kata	Navigasi kata. Memerlukan Option sebagai Meta di macOS
`Alt+F`	Pindahkan kursor maju satu kata	Navigasi kata. Memerlukan Option sebagai Meta di macOS

Tema dan tampilan

Pintasan	Deskripsi	Konteks
`Ctrl+T`	Alihkan penyorotan sintaks untuk blok kode	Hanya berfungsi di dalam menu pemilih `/theme`. Mengontrol apakah kode dalam respons Claude menggunakan pewarnaan sintaks

> **Note:**
>
> Penyorotan sintaks hanya tersedia dalam build asli Claude Code.

Input multiline

Metode	Pintasan	Konteks
Escape cepat	`\` + `Enter`	Berfungsi di semua terminal
Default macOS	`Option+Enter`	Default di macOS
Shift+Enter	`Shift+Enter`	Berfungsi langsung di iTerm2, WezTerm, Ghostty, Kitty
Urutan kontrol	`Ctrl+J`	Karakter line feed untuk multiline
Mode tempel	Tempel langsung	Untuk blok kode, log

> **Tip:**
>
> Shift+Enter berfungsi tanpa konfigurasi di iTerm2, WezTerm, Ghostty, dan Kitty. Untuk terminal lain (VS Code, Alacritty, Zed, Warp), jalankan `/terminal-setup` untuk memasang binding.

Perintah cepat

Pintasan	Deskripsi	Catatan
`/` di awal	Perintah atau skill	Lihat perintah bawaan dan skills
`!` di awal	Mode Bash	Jalankan perintah langsung dan tambahkan output eksekusi ke sesi
`@`	Penyebutan jalur file	Picu pelengkapan otomatis jalur file

Perintah bawaan

Ketik `/` di Claude Code untuk melihat semua perintah yang tersedia, atau ketik `/` diikuti huruf apa pun untuk memfilter. Tidak semua perintah terlihat oleh setiap pengguna. Beberapa tergantung pada platform, paket, atau lingkungan Anda. Misalnya, `/desktop`

hanya muncul di macOS dan Windows, `/upgrade` dan `/privacy-settings` hanya tersedia di paket Pro dan Max, dan `/terminal-setup` disembunyikan ketika terminal Anda secara asli mendukung pintasan kuncinya.

Claude Code juga dilengkapi dengan skills bundel seperti `/simplify`, `/batch`, dan `/debug` yang muncul bersama perintah bawaan saat Anda mengetik `/`. Untuk membuat perintah Anda sendiri, lihat skills.

Dalam tabel di bawah, `<arg>` menunjukkan argumen yang diperlukan dan `[arg]` menunjukkan argumen opsional.

Perintah	Tujuan
`/add-dir <path>`	Tambahkan direktori kerja baru ke sesi saat ini
`/agents`	Kelola konfigurasi agent
`/btw <question>`	Ajukan pertanyaan sampingan cepat tanpa menambah percakapan
`/chrome`	Konfigurasi pengaturan Claude di Chrome
`/clear`	Bersihkan riwayat percakapan dan bebaskan konteks. Alias: `/reset`, `/new`
`/compact [instructions]`	Kompak percakapan dengan instruksi fokus opsional
`/config`	Buka antarmuka Pengaturan untuk menyesuaikan tema, model, gaya output, dan preferensi lainnya. Alias: `/settings`
`/context`	Visualisasikan penggunaan konteks saat ini sebagai grid berwarna
`/copy`	Salin respons asisten terakhir ke clipboard. Ketika blok kode ada, menampilkan pemilih interaktif untuk memilih blok individual atau respons lengkap
`/cost`	Tampilkan statistik penggunaan token. Lihat panduan pelacakan biaya untuk detail khusus langganan
`/desktop`	Lanjutkan sesi saat ini di aplikasi Claude Code Desktop. Hanya macOS dan Windows. Alias: `/app`
`/diff`	Buka penampil diff interaktif yang menampilkan perubahan yang belum dilakukan dan diff per-turn. Gunakan panah kiri/kanan untuk beralih antara diff git saat ini dan turn Claude individual, dan atas/bawah untuk menelusuri file

Perintah	Tujuan
`/doctor`	Diagnosa dan verifikasi instalasi dan pengaturan Claude Code Anda
`/exit`	Keluar dari CLI. Alias: `/quit`
`/export [filename]`	Ekspor percakapan saat ini sebagai teks biasa. Dengan nama file, menulis langsung ke file itu. Tanpa, membuka dialog untuk menyalin ke clipboard atau menyimpan ke file
`/extra-usage`	Konfigurasi penggunaan ekstra untuk terus bekerja ketika batas laju tercapai
`/fast [on|off]`	Alihkan mode cepat aktif atau nonaktif
`/feedback [report]`	Kirimkan umpan balik tentang Claude Code. Alias: `/bug`
`/fork [name]`	Buat fork percakapan saat ini pada titik ini
`/help`	Tampilkan bantuan dan perintah yang tersedia
`/hooks`	Kelola konfigurasi hook untuk peristiwa alat
`/ide`	Kelola integrasi IDE dan tampilkan status
`/init`	Inisialisasi proyek dengan panduan `CLAUDE.md`
`/insights`	Hasilkan laporan yang menganalisis sesi Claude Code Anda, termasuk area proyek, pola interaksi, dan titik gesekan
`/install-github-app`	Atur aplikasi Claude GitHub Actions untuk repositori. Memandu Anda memilih repo dan mengonfigurasi integrasi
`/install-slack-app`	Instal aplikasi Claude Slack. Membuka browser untuk menyelesaikan alur OAuth
`/keybindings`	Buka atau buat file konfigurasi pintasan keyboard Anda
`/login`	Masuk ke akun Anthropic Anda
`/logout`	Keluar dari akun Anthropic Anda
`/mcp`	Kelola koneksi server MCP dan autentikasi OAuth
`/memory`	Edit file memori `CLAUDE.md`, aktifkan atau nonaktifkan auto-memory, dan lihat entri auto-memory

Perintah	Tujuan
`/mobile`	Tampilkan kode QR untuk mengunduh aplikasi mobile Claude. Alias: `/ios`, `/android`
`/model [model]`	Pilih atau ubah model AI. Untuk model yang mendukungnya, gunakan panah kiri/kanan untuk menyesuaikan tingkat upaya. Perubahan berlaku segera tanpa menunggu respons saat ini selesai
`/passes`	Bagikan minggu gratis Claude Code dengan teman. Hanya terlihat jika akun Anda memenuhi syarat
`/permissions`	Lihat atau perbarui izin. Alias: `/allowed-tools`
`/plan`	Masuk mode rencana langsung dari prompt
`/plugin`	Kelola plugin Claude Code
`/pr-comments [PR]`	Ambil dan tampilkan komentar dari permintaan tarik GitHub. Secara otomatis mendeteksi PR untuk cabang saat ini, atau lewatkan URL atau nomor PR. Memerlukan CLI `gh`
`/privacy-settings`	Lihat dan perbarui pengaturan privasi Anda. Hanya tersedia untuk pelanggan paket Pro dan Max
`/release-notes`	Lihat changelog lengkap, dengan versi terbaru paling dekat dengan prompt Anda
`/reload-plugins`	Muat ulang semua plugin aktif untuk menerapkan perubahan yang tertunda tanpa memulai ulang. Melaporkan apa yang dimuat dan mencatat perubahan apa pun yang memerlukan restart
`/remote-control`	Buat sesi ini tersedia untuk kontrol jarak jauh dari claude.ai. Alias: `/rc`
`/remote-env`	Konfigurasi lingkungan jarak jauh default untuk sesi teleport
`/rename [name]`	Ubah nama sesi saat ini. Tanpa nama, secara otomatis menghasilkan satu dari riwayat percakapan
`/resume [session]`	Lanjutkan percakapan berdasarkan ID atau nama, atau buka pemilih sesi. Alias: `/continue`
`/review`	Tidak digunakan lagi. Instal plugin `code-review` sebagai gantinya: `claude plugin install code-review@claude-code-marketplace`

Perintah	Tujuan
`/rewind`	Putar ulang percakapan dan/atau kode ke titik sebelumnya, atau ringkas dari pesan yang dipilih. Lihat checkpointing. Alias: `/checkpoint`
`/sandbox`	Alihkan mode sandbox. Tersedia hanya di platform yang didukung
`/security-review`	Analisis perubahan yang tertunda di cabang saat ini untuk kerentanan keamanan. Meninjau diff git dan mengidentifikasi risiko seperti injeksi, masalah auth, dan paparan data
`/skills`	Daftar skills yang tersedia
`/stats`	Visualisasikan penggunaan harian, riwayat sesi, streak, dan preferensi model
`/status`	Buka antarmuka Pengaturan (tab Status) yang menampilkan versi, model, akun, dan konektivitas
`/statusline`	Konfigurasi status line Claude Code. Jelaskan apa yang Anda inginkan, atau jalankan tanpa argumen untuk auto-configure dari prompt shell Anda
`/stickers`	Pesan stiker Claude Code
`/tasks`	Daftar dan kelola tugas latar belakang
`/terminal-setup`	Konfigurasi pintasan keyboard terminal untuk Shift+Enter dan pintasan lainnya. Hanya terlihat di terminal yang membutuhkannya, seperti VS Code, Alacritty, atau Warp
`/theme`	Ubah tema warna. Mencakup varian terang dan gelap, tema yang dapat diakses buta warna (daltonized), dan tema ANSI yang menggunakan palet warna terminal Anda
`/upgrade`	Buka halaman upgrade untuk beralih ke tingkat paket yang lebih tinggi
`/usage`	Tampilkan batas penggunaan paket dan status batas laju
`/vim`	Alihkan antara mode pengeditan Vim dan Normal

Prompt MCP

Server MCP dapat mengekspos prompt yang muncul sebagai perintah. Ini menggunakan format `/mcp__<server>__<prompt>` dan secara dinamis ditemukan dari server yang terhubung. Lihat prompt MCP untuk detail.

Mode editor Vim

Aktifkan pengeditan gaya vim dengan perintah `/vim` atau konfigurasi secara permanen melalui `/config`.

Pengalihan mode

Perintah	Tindakan	Dari mode
`Esc`	Masuk mode NORMAL	INSERT
`i`	Sisipkan sebelum kursor	NORMAL
`I`	Sisipkan di awal baris	NORMAL
`a`	Sisipkan setelah kursor	NORMAL
`A`	Sisipkan di akhir baris	NORMAL
`o`	Buka baris di bawah	NORMAL
`O`	Buka baris di atas	NORMAL

Navigasi (mode NORMAL)

Perintah	Tindakan
`h` / `j` / `k` / `l`	Pindah kiri/bawah/atas/kanan
`w`	Kata berikutnya
`e`	Akhir kata
`b`	Kata sebelumnya
`0`	Awal baris
`$`	Akhir baris
`^`	Karakter non-blank pertama
`gg`	Awal input
`G`	Akhir input
`f{char}`	Lompat ke kemunculan berikutnya dari karakter
`F{char}`	Lompat ke kemunculan sebelumnya dari karakter

Perintah	Tindakan
`t{char}`	Lompat ke tepat sebelum kemunculan berikutnya dari karakter
`T{char}`	Lompat ke tepat setelah kemunculan sebelumnya dari karakter
`;`	Ulangi gerakan f/F/t/T terakhir
`,`	Ulangi gerakan f/F/t/T terakhir dalam urutan terbalik

> **Note:**
>
> Dalam mode normal vim, jika kursor berada di awal atau akhir input dan tidak dapat bergerak lebih jauh, tombol panah menavigasi riwayat perintah sebagai gantinya.

Pengeditan (mode NORMAL)

Perintah	Tindakan
`x`	Hapus karakter
`dd`	Hapus baris
`D`	Hapus hingga akhir baris
`dw` / `de` / `db`	Hapus kata/hingga akhir/kembali
`cc`	Ubah baris
`C`	Ubah hingga akhir baris
`cw` / `ce` / `cb`	Ubah kata/hingga akhir/kembali
`yy` / `Y`	Yank (salin) baris
`yw` / `ye` / `yb`	Yank kata/hingga akhir/kembali
`p`	Tempel setelah kursor
`P`	Tempel sebelum kursor
`>>`	Indentasi baris
`<<`	Dedentasi baris
`J`	Gabungkan baris

Perintah	Tindakan
`.`	Ulangi perubahan terakhir

Objek teks (mode NORMAL)

Objek teks bekerja dengan operator seperti `d`, `c`, dan `y`:

Perintah	Tindakan
`iw` / `aw`	Kata dalam/sekitar
`iW` / `aW`	WORD dalam/sekitar (dibatasi whitespace)
`i"` / `a"`	Dalam/sekitar tanda kutip ganda
`i'` / `a'`	Dalam/sekitar tanda kutip tunggal
`i(` / `a(`	Dalam/sekitar tanda kurung
`i[` / `a[`	Dalam/sekitar tanda kurung siku
`i{` / `a{`	Dalam/sekitar tanda kurung kurawal

Riwayat perintah

Claude Code mempertahankan riwayat perintah untuk sesi saat ini:

- Riwayat input disimpan per direktori kerja
- Riwayat input direset ketika Anda menjalankan `/clear` untuk memulai sesi baru. Percakapan sesi sebelumnya dipertahankan dan dapat dilanjutkan.
- Gunakan panah Atas/Bawah untuk menavigasi (lihat pintasan keyboard di atas)
- **Catatan**: ekspansi riwayat (`!`) dinonaktifkan secara default

Pencarian terbalik dengan Ctrl+R

Tekan `Ctrl+R` untuk secara interaktif mencari melalui riwayat perintah Anda:

1. **Mulai pencarian**: tekan `Ctrl+R` untuk mengaktifkan pencarian riwayat terbalik
2. **Ketik kueri**: masukkan teks untuk dicari di perintah sebelumnya. Istilah pencarian disorot dalam hasil yang cocok
3. **Navigasi kecocokan**: tekan `Ctrl+R` lagi untuk siklus melalui kecocokan yang lebih lama

4. **Terima kecocokan**:
   - Tekan `Tab` atau `Esc` untuk menerima kecocokan saat ini dan lanjutkan pengeditan
   - Tekan `Enter` untuk menerima dan menjalankan perintah segera
5. **Batalkan pencarian**:
   - Tekan `Ctrl+C` untuk membatalkan dan mengembalikan input asli Anda
   - Tekan `Backspace` pada pencarian kosong untuk membatalkan

Pencarian menampilkan perintah yang cocok dengan istilah pencarian disorot, sehingga Anda dapat menemukan dan menggunakan kembali input sebelumnya.

Perintah bash latar belakang

Claude Code mendukung menjalankan perintah bash di latar belakang, memungkinkan Anda untuk terus bekerja sementara proses yang berjalan lama dieksekusi.

Cara backgrounding bekerja

Ketika Claude Code menjalankan perintah di latar belakang, ia menjalankan perintah secara asinkron dan segera mengembalikan ID tugas latar belakang. Claude Code dapat merespons prompt baru sementara perintah terus dieksekusi di latar belakang.

Untuk menjalankan perintah di latar belakang, Anda dapat:

- Minta Claude Code untuk menjalankan perintah di latar belakang
- Tekan Ctrl+B untuk memindahkan invokasi alat Bash biasa ke latar belakang. (Pengguna Tmux harus menekan Ctrl+B dua kali karena kunci awalan tmux.)

Fitur utama:

- Output di-buffer dan Claude dapat mengambilnya menggunakan alat TaskOutput
- Tugas latar belakang memiliki ID unik untuk pelacakan dan pengambilan output
- Tugas latar belakang secara otomatis dibersihkan ketika Claude Code keluar

Untuk menonaktifkan semua fungsionalitas tugas latar belakang, atur variabel lingkungan `CLAUDE_CODE_DISABLE_BACKGROUND_TASKS` ke `1`. Lihat Variabel lingkungan untuk detail.

Perintah yang sering di-background:

- Alat build (webpack, vite, make)
- Manajer paket (npm, yarn, pnpm)
- Pelari tes (jest, pytest)

- Server pengembangan
- Proses yang berjalan lama (docker, terraform)

Mode Bash dengan awalan `!`

Jalankan perintah bash langsung tanpa melalui Claude dengan menambahkan awalan input Anda dengan `!`:

```
! npm test
! git status
! ls -la
```

Mode Bash:

- Menambahkan perintah dan outputnya ke konteks percakapan
- Menampilkan kemajuan dan output real-time
- Mendukung backgrounding `Ctrl+B` yang sama untuk perintah yang berjalan lama
- Tidak memerlukan Claude untuk menginterpretasi atau menyetujui perintah
- Mendukung pelengkapan otomatis berbasis riwayat: ketik perintah parsial dan tekan **Tab** untuk melengkapi dari perintah `!` sebelumnya di proyek saat ini
- Keluar dengan `Escape`, `Backspace`, atau `Ctrl+U` pada prompt kosong

Ini berguna untuk operasi shell cepat sambil mempertahankan konteks percakapan.

Saran prompt

Ketika Anda pertama kali membuka sesi, perintah contoh yang digelapkan muncul di input prompt untuk membantu Anda memulai. Claude Code memilih ini dari riwayat git proyek Anda, jadi ini mencerminkan file yang telah Anda kerjakan baru-baru ini.

Setelah Claude merespons, saran terus muncul berdasarkan riwayat percakapan Anda, seperti langkah lanjutan dari permintaan multi-bagian atau kelanjutan alami dari alur kerja Anda.

- Tekan **Tab** untuk menerima saran, atau tekan **Enter** untuk menerima dan mengirimkan
- Mulai mengetik untuk menolaknya

Saran berjalan sebagai permintaan latar belakang yang menggunakan kembali cache prompt percakapan induk, jadi biaya tambahan minimal. Claude Code melewati pembuatan saran ketika cache dingin untuk menghindari biaya yang tidak perlu.

Saran secara otomatis dilewati setelah turn pertama percakapan, dalam mode non-interaktif, dan dalam mode rencana.

Untuk menonaktifkan saran prompt sepenuhnya, atur variabel lingkungan atau alihkan pengaturan di `/config`:

```
export CLAUDE_CODE_ENABLE_PROMPT_SUGGESTION=false
```

Pertanyaan sampingan dengan /btw

Gunakan `/btw` untuk mengajukan pertanyaan cepat tentang pekerjaan saat ini tanpa menambah riwayat percakapan. Ini berguna ketika Anda menginginkan jawaban cepat tetapi tidak ingin mengacaukan konteks utama atau mengalihkan Claude dari tugas yang berjalan lama.

```
/btw what was the name of that config file again?
```

Pertanyaan sampingan memiliki visibilitas penuh ke percakapan saat ini, jadi Anda dapat bertanya tentang kode yang telah dibaca Claude, keputusan yang dibuatnya sebelumnya, atau apa pun dari sesi. Pertanyaan dan jawaban bersifat sementara: mereka muncul dalam overlay yang dapat ditutup dan tidak pernah memasuki riwayat percakapan.

- **Tersedia saat Claude sedang bekerja**: Anda dapat menjalankan `/btw` bahkan saat Claude memproses respons. Pertanyaan sampingan berjalan secara independen dan tidak mengganggu turn utama.
- **Tidak ada akses alat**: pertanyaan sampingan hanya menjawab dari apa yang sudah ada dalam konteks. Claude tidak dapat membaca file, menjalankan perintah, atau mencari saat menjawab pertanyaan sampingan.
- **Respons tunggal**: tidak ada turn lanjutan. Jika Anda memerlukan bolak-balik, gunakan prompt normal sebagai gantinya.
- **Biaya rendah**: pertanyaan sampingan menggunakan kembali cache prompt percakapan induk, jadi biaya tambahan minimal.

Tekan **Space**, **Enter**, atau **Escape** untuk menolak jawaban dan kembali ke prompt.

`/btw` adalah kebalikan dari subagent: ia melihat percakapan lengkap Anda tetapi tidak memiliki alat, sementara subagent memiliki alat lengkap tetapi dimulai dengan konteks kosong. Gunakan `/btw` untuk bertanya tentang apa yang sudah diketahui Claude dari sesi ini; gunakan subagent untuk menemukan sesuatu yang baru.

Daftar tugas

Ketika mengerjakan pekerjaan yang kompleks dan multi-langkah, Claude membuat daftar tugas untuk melacak kemajuan. Tugas muncul di area status terminal Anda dengan indikator yang menunjukkan apa yang tertunda, sedang berlangsung, atau selesai.

- Tekan `Ctrl+T` untuk mengalihkan tampilan daftar tugas. Tampilan menampilkan hingga 10 tugas sekaligus
- Untuk melihat semua tugas atau menghapusnya, minta Claude secara langsung: “show me all tasks” atau “clear all tasks”
- Tugas bertahan di seluruh pemadatan konteks, membantu Claude tetap terorganisir pada proyek yang lebih besar
- Untuk berbagi daftar tugas di seluruh sesi, atur `CLAUDE_CODE_TASK_LIST_ID` untuk menggunakan direktori bernama di `~/.claude/tasks/` : `CLAUDE_CODE_TASK_LIST_ID=my-project claude`
- Untuk kembali ke daftar TODO sebelumnya, atur `CLAUDE_CODE_ENABLE_TASKS=false` .

Status tinjauan PR

Ketika bekerja pada cabang dengan permintaan tarik terbuka, Claude Code menampilkan tautan PR yang dapat diklik di footer (misalnya, “PR #446”). Tautan memiliki garis bawah berwarna yang menunjukkan status tinjauan:

- Hijau: disetujui
- Kuning: menunggu tinjauan
- Merah: perubahan diminta
- Abu-abu: draft
- Ungu: digabungkan

`Cmd+click` (Mac) atau `Ctrl+click` (Windows/Linux) tautan untuk membuka permintaan tarik di browser Anda. Status diperbarui secara otomatis setiap 60 detik.

> **Note:**
>
> Status PR memerlukan CLI `gh` untuk diinstal dan diautentikasi (`gh auth login`).

Lihat juga

- Skills - Prompt dan alur kerja kustom
- Checkpointing - Putar ulang pengeditan Claude dan pulihkan status sebelumnya

- Referensi CLI - Bendera dan opsi baris perintah
- Pengaturan - Opsi konfigurasi
- Manajemen memori - Mengelola file CLAUDE.md

Praktik Terbaik untuk Claude Code

> *Tips dan pola untuk memaksimalkan Claude Code, dari mengonfigurasi lingkungan Anda hingga menskalakan di seluruh sesi paralel.*

Claude Code adalah lingkungan pengkodean yang bersifat agentic. Tidak seperti chatbot yang menjawab pertanyaan dan menunggu, Claude Code dapat membaca file Anda, menjalankan perintah, membuat perubahan, dan bekerja secara mandiri melalui masalah sambil Anda menonton, mengarahkan, atau sepenuhnya menjauh.

Ini mengubah cara Anda bekerja. Alih-alih menulis kode sendiri dan meminta Claude untuk meninjau, Anda menjelaskan apa yang Anda inginkan dan Claude mengetahui cara membangunnya. Claude mengeksplorasi, merencanakan, dan mengimplementasikan.

Namun otonomi ini masih datang dengan kurva pembelajaran. Claude bekerja dalam batasan tertentu yang perlu Anda pahami.

Panduan ini mencakup pola yang telah terbukti efektif di seluruh tim internal Anthropic dan untuk insinyur yang menggunakan Claude Code di berbagai basis kode, bahasa, dan lingkungan. Untuk cara loop agentic bekerja di balik layar, lihat Cara Claude Code Bekerja.

Sebagian besar praktik terbaik didasarkan pada satu batasan: jendela konteks Claude terisi dengan cepat, dan kinerja menurun saat terisi.

Jendela konteks Claude menyimpan seluruh percakapan Anda, termasuk setiap pesan, setiap file yang dibaca Claude, dan setiap output perintah. Namun, ini dapat terisi dengan cepat. Sesi debugging tunggal atau eksplorasi basis kode mungkin menghasilkan dan mengonsumsi puluhan ribu token.

Ini penting karena kinerja LLM menurun saat konteks terisi. Ketika jendela konteks hampir penuh, Claude mungkin mulai “lupa” instruksi sebelumnya atau membuat lebih banyak kesalahan. Jendela konteks adalah sumber daya paling penting untuk dikelola. Lacak penggunaan konteks secara berkelanjutan dengan baris status khusus, dan lihat Kurangi penggunaan token untuk strategi mengurangi penggunaan token.

Berikan Claude cara untuk memverifikasi pekerjaannya

> **Tip:**
>
> Sertakan tes, tangkapan layar, atau output yang diharapkan sehingga Claude dapat memeriksa dirinya sendiri. Ini adalah hal dengan leverage tertinggi yang dapat Anda lakukan.

Claude berkinerja jauh lebih baik ketika dapat memverifikasi pekerjaannya sendiri, seperti menjalankan tes, membandingkan tangkapan layar, dan memvalidasi output.

Tanpa kriteria kesuksesan yang jelas, mungkin menghasilkan sesuatu yang terlihat benar tetapi sebenarnya tidak berfungsi. Anda menjadi satu-satunya loop umpan balik, dan setiap kesalahan memerlukan perhatian Anda.

Strategi	Sebelum	Sesudah
Berikan kriteria verifikasi	*"implementasikan fungsi yang memvalidasi alamat email"*	*"tulis fungsi validateEmail. contoh kasus uji: user@example.com adalah true, invalid adalah false, user@.com adalah false. jalankan tes setelah mengimplementasikan"*
Verifikasi perubahan UI secara visual	*"buat dashboard terlihat lebih baik"*	*"[tempel tangkapan layar] implementasikan desain ini. ambil tangkapan layar hasilnya dan bandingkan dengan yang asli. daftar perbedaan dan perbaiki"*
Tangani penyebab akar, bukan gejala	*"build gagal"*	*"build gagal dengan kesalahan ini: [tempel kesalahan]. perbaiki dan verifikasi build berhasil. tangani penyebab akar, jangan tekan kesalahan"*

Perubahan UI dapat diverifikasi menggunakan ekstensi Claude di Chrome. Ini membuka tab baru di browser Anda, menguji UI, dan melakukan iterasi hingga kode berfungsi.

Verifikasi Anda juga dapat berupa rangkaian tes, linter, atau perintah Bash yang memeriksa output. Investasikan dalam membuat verifikasi Anda sangat solid.

Jelajahi terlebih dahulu, kemudian rencanakan, kemudian kode

> **Tip:**
>
> Pisahkan penelitian dan perencanaan dari implementasi untuk menghindari menyelesaikan masalah yang salah.

Membiarkan Claude langsung melompat ke pengkodean dapat menghasilkan kode yang menyelesaikan masalah yang salah. Gunakan Plan Mode untuk memisahkan eksplorasi dari eksekusi.

Alur kerja yang direkomendasikan memiliki empat fase:

Step 1: Jelajahi

Masukkan Plan Mode. Claude membaca file dan menjawab pertanyaan tanpa membuat perubahan.

```
read /src/auth and understand how we handle sessions and login.
also look at how we manage environment variables for secrets.
```

Step 2: Rencanakan

Minta Claude untuk membuat rencana implementasi terperinci.

```
I want to add Google OAuth. What files need to change?
What's the session flow? Create a plan.
```

Tekan `Ctrl+G` untuk membuka rencana di editor teks Anda untuk pengeditan langsung sebelum Claude melanjutkan.

Step 3: Implementasikan

Beralih kembali ke Normal Mode dan biarkan Claude kode, memverifikasi terhadap rencananya.

```
implement the OAuth flow from your plan. write tests for the
callback handler, run the test suite and fix any failures.
```

Step 4: Komit

Minta Claude untuk melakukan komit dengan pesan deskriptif dan membuat PR.

```
commit with a descriptive message and open a PR
```

Plan Mode berguna, tetapi juga menambah overhead.

Untuk tugas di mana cakupannya jelas dan perbaikannya kecil (seperti memperbaiki typo, menambahkan baris log, atau mengganti nama variabel) minta Claude untuk melakukannya secara langsung.

Perencanaan paling berguna ketika Anda tidak yakin tentang pendekatannya, ketika perubahan memodifikasi beberapa file, atau ketika Anda tidak terbiasa dengan kode yang dimodifikasi. Jika Anda dapat menjelaskan diff dalam satu kalimat, lewati rencana.

Berikan konteks spesifik dalam prompt Anda

> **Tip:**
>
> Semakin tepat instruksi Anda, semakin sedikit koreksi yang Anda butuhkan.

Claude dapat menyimpulkan niat, tetapi tidak dapat membaca pikiran Anda. Referensikan file spesifik, sebutkan batasan, dan tunjukkan pola contoh.

Strategi	Sebelum	Sesudah
Batasi tugas. Tentukan file mana, skenario apa, dan preferensi pengujian.	*"tambahkan tes untuk foo.py"*	*"tulis tes untuk foo.py yang mencakup kasus tepi di mana pengguna logout. hindari mock."*
Tunjukkan sumber. Arahkan Claude ke sumber yang dapat menjawab pertanyaan.	*"mengapa ExecutionFactory memiliki api yang aneh?"*	*"lihat melalui riwayat git ExecutionFactory dan ringkas bagaimana api-nya menjadi seperti ini"*

Strategi	Sebelum	Sesudah
Referensikan pola yang ada. Tunjukkan Claude pola dalam basis kode Anda.	*“tambahkan widget kalender”*	*“lihat bagaimana widget yang ada diimplementasikan di halaman beranda untuk memahami pola. HotDogWidget.php adalah contoh yang baik. ikuti pola untuk mengimplementasikan widget kalender baru yang memungkinkan pengguna memilih bulan dan paginate maju/mundur untuk memilih tahun. bangun dari awal tanpa perpustakaan selain yang sudah digunakan dalam basis kode.”*
Jelaskan gejala. Berikan gejala, lokasi yang mungkin, dan apa “diperbaiki” terlihat seperti.	*“perbaiki bug login”*	*“pengguna melaporkan bahwa login gagal setelah timeout sesi. periksa alur auth di src/auth/, terutama penyegaran token. tulis tes yang gagal yang mereproduksi masalah, kemudian perbaiki”*

Prompt yang samar dapat berguna ketika Anda mengeksplorasi dan dapat mengubah arah. Prompt seperti `"apa yang akan Anda tingkatkan dalam file ini?"` dapat mengungkap hal-hal yang tidak akan Anda pikirkan untuk ditanyakan.

Berikan konten kaya

> **Tip:**
>
> Gunakan `@` untuk mereferensikan file, tempel tangkapan layar/gambar, atau pipa data secara langsung.

Anda dapat memberikan data kaya kepada Claude dalam beberapa cara:

- **Referensikan file dengan** `@` alih-alih menjelaskan di mana kode berada. Claude membaca file sebelum merespons.

- **Tempel gambar secara langsung**. Salin/tempel atau seret dan lepas gambar ke dalam prompt.
- **Berikan URL** untuk dokumentasi dan referensi API. Gunakan `/permissions` untuk allowlist domain yang sering digunakan.
- **Pipa data** dengan menjalankan `cat error.log | claude` untuk mengirim konten file secara langsung.
- **Biarkan Claude mengambil apa yang dibutuhkan**. Beri tahu Claude untuk menarik konteks sendiri menggunakan perintah Bash, alat MCP, atau dengan membaca file.

Konfigurasi lingkungan Anda

Beberapa langkah setup membuat Claude Code jauh lebih efektif di semua sesi Anda. Untuk gambaran lengkap fitur ekstensi dan kapan menggunakan masing-masing, lihat Perluas Claude Code.

Tulis CLAUDE.md yang efektif

> **Tip:**
>
> Jalankan `/init` untuk menghasilkan file CLAUDE.md pemula berdasarkan struktur proyek Anda saat ini, kemudian perbaiki seiring waktu.

CLAUDE.md adalah file khusus yang dibaca Claude di awal setiap percakapan. Sertakan perintah Bash, gaya kode, dan aturan alur kerja. Ini memberikan Claude konteks persisten yang tidak dapat disimpulkan dari kode saja.

Perintah `/init` menganalisis basis kode Anda untuk mendeteksi sistem build, kerangka kerja tes, dan pola kode, memberikan Anda fondasi solid untuk disempurnakan.

Tidak ada format yang diperlukan untuk file CLAUDE.md, tetapi tetap singkat dan mudah dibaca manusia. Sebagai contoh:

```
## Code style
- Use ES modules (import/export) syntax, not CommonJS (require)
- Destructure imports when possible (eg. import { foo } from 'bar')

## Workflow
- Be sure to typecheck when you're done making a series of code changes
- Prefer running single tests, and not the whole test suite, for performance
```

CLAUDE.md dimuat setiap sesi, jadi hanya sertakan hal-hal yang berlaku secara luas. Untuk pengetahuan domain atau alur kerja yang hanya relevan kadang-kadang, gunakan skills sebagai gantinya. Claude memuat mereka sesuai permintaan tanpa membengkak setiap percakapan.

Tetap ringkas. Untuk setiap baris, tanyakan: *"Apakah menghapus ini akan menyebabkan Claude membuat kesalahan?"* Jika tidak, potong. File CLAUDE.md yang membengkak menyebabkan Claude mengabaikan instruksi aktual Anda!

✅ Sertakan	❌ Kecualikan
Perintah Bash yang tidak dapat ditebak Claude	Apa pun yang dapat diketahui Claude dengan membaca kode
Aturan gaya kode yang berbeda dari default	Konvensi bahasa standar yang sudah diketahui Claude
Instruksi pengujian dan test runner pilihan	Dokumentasi API terperinci (tautkan ke dokumen sebagai gantinya)
Etiket repositori (penamaan cabang, konvensi PR)	Informasi yang berubah sering
Keputusan arsitektur khusus untuk proyek Anda	Penjelasan panjang atau tutorial
Keanehan lingkungan pengembang (variabel env yang diperlukan)	Praktik yang jelas sendiri seperti "tulis kode yang bersih"
Gotcha umum atau perilaku yang tidak jelas	Deskripsi file demi file dari basis kode

Jika Claude terus melakukan sesuatu yang tidak Anda inginkan meskipun memiliki aturan melawannya, file mungkin terlalu panjang dan aturan hilang. Jika Claude mengajukan pertanyaan yang dijawab di CLAUDE.md, frasenya mungkin ambigu. Perlakukan CLAUDE.md seperti kode: tinjau saat ada yang salah, pangkas secara teratur, dan uji perubahan dengan mengamati apakah perilaku Claude benar-benar bergeser.

Anda dapat menyesuaikan instruksi dengan menambahkan penekanan (misalnya, "PENTING" atau "ANDA HARUS") untuk meningkatkan kepatuhan. Periksa CLAUDE.md ke dalam git sehingga tim Anda dapat berkontribusi. File ini meningkat nilainya seiring waktu.

File CLAUDE.md dapat mengimpor file tambahan menggunakan sintaks `@path/to/import`:

```
See @README.md for project overview and @package.json for available npm commands.

## Additional Instructions
- Git workflow: @docs/git-instructions.md
- Personal overrides: @~/.claude/my-project-instructions.md
```

Anda dapat menempatkan file CLAUDE.md di beberapa lokasi:

- **Folder home (`~/.claude/CLAUDE.md`)**: berlaku untuk semua sesi Claude
- **Root proyek (`./CLAUDE.md`)**: periksa ke dalam git untuk dibagikan dengan tim Anda
- **Direktori induk**: berguna untuk monorepo di mana `root/CLAUDE.md` dan `root/foo/CLAUDE.md` ditarik secara otomatis
- **Direktori anak**: Claude menarik file CLAUDE.md anak sesuai permintaan saat bekerja dengan file di direktori tersebut

Konfigurasi izin

> **Tip:**
>
> Gunakan `/permissions` untuk allowlist perintah aman atau `/sandbox` untuk isolasi tingkat OS. Ini mengurangi gangguan sambil membuat Anda tetap mengendalikan.

Secara default, Claude Code meminta izin untuk tindakan yang mungkin memodifikasi sistem Anda: penulisan file, perintah Bash, alat MCP, dll. Ini aman tetapi membosankan. Setelah persetujuan kesepuluh Anda tidak benar-benar meninjau lagi, Anda hanya mengklik. Ada dua cara untuk mengurangi gangguan ini:

- **Allowlist izin**: izinkan alat spesifik yang Anda tahu aman (seperti `npm run lint` atau `git commit`)
- **Sandboxing**: aktifkan isolasi tingkat OS yang membatasi akses sistem file dan jaringan, memungkinkan Claude bekerja lebih bebas dalam batas yang ditentukan

Alternatifnya, gunakan `--dangerously-skip-permissions` untuk melewati semua pemeriksaan izin untuk alur kerja yang terkandung seperti memperbaiki kesalahan lint atau menghasilkan boilerplate.

> **Warning:**
>
> Membiarkan Claude menjalankan perintah arbitrer dapat menghasilkan kehilangan data, kerusakan sistem, atau exfiltrasi data melalui prompt injection. Hanya gunakan `--dangerously-skip-permissions` di sandbox tanpa akses internet.

Baca lebih lanjut tentang mengonfigurasi izin dan mengaktifkan sandboxing.

Gunakan alat CLI

> **Tip:**
>
> Beri tahu Claude Code untuk menggunakan alat CLI seperti `gh`, `aws`, `gcloud`, dan `sentry-cli` saat berinteraksi dengan layanan eksternal.

Alat CLI adalah cara paling efisien konteks untuk berinteraksi dengan layanan eksternal. Jika Anda menggunakan GitHub, instal CLI `gh`. Claude tahu cara menggunakannya untuk membuat masalah, membuka pull request, dan membaca komentar. Tanpa `gh`, Claude masih dapat menggunakan GitHub API, tetapi permintaan yang tidak diautentikasi sering kali mencapai batas laju.

Claude juga efektif dalam mempelajari alat CLI yang tidak diketahuinya. Coba prompt seperti `Use 'foo-cli-tool --help' to learn about foo tool, then use it to solve A, B, C.`

Hubungkan server MCP

> **Tip:**
>
> Jalankan `claude mcp add` untuk menghubungkan alat eksternal seperti Notion, Figma, atau database Anda.

Dengan server MCP, Anda dapat meminta Claude untuk mengimplementasikan fitur dari pelacak masalah, query database, menganalisis data pemantauan, mengintegrasikan desain dari Figma, dan mengotomatisasi alur kerja.

Atur hooks

> **Tip:**
>
> Gunakan hooks untuk tindakan yang harus terjadi setiap kali tanpa pengecualian.

Hooks menjalankan skrip secara otomatis pada titik tertentu dalam alur kerja Claude. Tidak seperti instruksi CLAUDE.md yang bersifat penasihat, hooks bersifat deterministik dan menjamin tindakan terjadi.

Claude dapat menulis hooks untuk Anda. Coba prompt seperti *"Tulis hook yang menjalankan eslint setelah setiap pengeditan file"* atau *"Tulis hook yang memblokir penulisan ke folder migrasi."* Jalankan `/hooks` untuk konfigurasi interaktif, atau edit `.claude/settings.json` secara langsung.

Buat skills

> **Tip:**
>
> Buat file `SKILL.md` di `.claude/skills/` untuk memberikan Claude pengetahuan domain dan alur kerja yang dapat digunakan kembali.

Skills memperluas pengetahuan Claude dengan informasi khusus untuk proyek, tim, atau domain Anda. Claude menerapkannya secara otomatis saat relevan, atau Anda dapat menginvokannya secara langsung dengan `/skill-name`.

Buat skill dengan menambahkan direktori dengan `SKILL.md` ke `.claude/skills/`:

```
---
name: api-conventions
description: REST API design conventions for our services
---
## API Conventions
- Use kebab-case for URL paths
- Use camelCase for JSON properties
- Always include pagination for list endpoints
- Version APIs in the URL path (/v1/, /v2/)
```

Skills juga dapat mendefinisikan alur kerja yang dapat digunakan kembali yang Anda panggil secara langsung:

```
---
name: fix-issue
description: Fix a GitHub issue
disable-model-invocation: true
---
Analyze and fix the GitHub issue: $ARGUMENTS.

1. Use `gh issue view` to get the issue details
2. Understand the problem described in the issue
3. Search the codebase for relevant files
4. Implement the necessary changes to fix the issue
5. Write and run tests to verify the fix
6. Ensure code passes linting and type checking
7. Create a descriptive commit message
8. Push and create a PR
```

Jalankan `/fix-issue 1234` untuk menginvokannya. Gunakan `disable-model-invocation: true` untuk alur kerja dengan efek samping yang ingin Anda picu secara manual.

Buat subagent khusus

> **Tip:**

> Tentukan asisten khusus di `.claude/agents/` yang dapat didelegasikan Claude untuk tugas terisolasi.

Subagents berjalan dalam konteks mereka sendiri dengan set alat yang diizinkan mereka sendiri. Mereka berguna untuk tugas yang membaca banyak file atau memerlukan fokus khusus tanpa mengacaukan percakapan utama Anda.

```
---
name: security-reviewer
description: Reviews code for security vulnerabilities
tools: Read, Grep, Glob, Bash
model: opus
---
You are a senior security engineer. Review code for:
- Injection vulnerabilities (SQL, XSS, command injection)
- Authentication and authorization flaws
- Secrets or credentials in code
- Insecure data handling

Provide specific line references and suggested fixes.
```

Beri tahu Claude untuk menggunakan subagent secara eksplisit: *"Gunakan subagent untuk meninjau kode ini untuk masalah keamanan."*

Instal plugins

> **Tip:**
>
> Jalankan `/plugin` untuk menjelajahi marketplace. Plugin menambahkan skills, alat, dan integrasi tanpa konfigurasi.

Plugins menggabungkan skills, hooks, subagent, dan server MCP menjadi satu unit yang dapat diinstal dari komunitas dan Anthropic. Jika Anda bekerja dengan bahasa yang diketik, instal plugin code intelligence untuk memberikan Claude navigasi simbol presisi dan deteksi kesalahan otomatis setelah pengeditan.

Untuk panduan memilih antara skills, subagent, hooks, dan MCP, lihat Perluas Claude Code.

Berkomunikasi secara efektif

Cara Anda berkomunikasi dengan Claude Code secara signifikan mempengaruhi kualitas hasil.

Tanyakan pertanyaan basis kode

> **Tip:**
>
> Tanyakan Claude pertanyaan yang akan Anda tanyakan kepada insinyur senior.

Saat onboarding ke basis kode baru, gunakan Claude Code untuk pembelajaran dan eksplorasi. Anda dapat mengajukan Claude pertanyaan yang sama seperti yang Anda tanyakan kepada insinyur lain:

- Bagaimana cara logging bekerja?
- Bagaimana cara membuat endpoint API baru?
- Apa yang dilakukan `async move { ... }` pada baris 134 dari `foo.rs`?
- Kasus tepi apa yang ditangani `CustomerOnboardingFlowImpl`?
- Mengapa kode ini memanggil `foo()` alih-alih `bar()` pada baris 333?

Menggunakan Claude Code dengan cara ini adalah alur kerja onboarding yang efektif, meningkatkan waktu ramp-up dan mengurangi beban pada insinyur lain. Tidak ada prompt khusus yang diperlukan: tanyakan pertanyaan secara langsung.

Biarkan Claude mewawancarai Anda

> **Tip:**
>
> Untuk fitur yang lebih besar, biarkan Claude mewawancarai Anda terlebih dahulu. Mulai dengan prompt minimal dan minta Claude untuk mewawancarai Anda menggunakan alat `AskUserQuestion`.

Claude menanyakan tentang hal-hal yang mungkin belum Anda pertimbangkan, termasuk implementasi teknis, UI/UX, kasus tepi, dan trade-off.

```
I want to build [brief description]. Interview me in detail using the
AskUserQuestion tool.

Ask about technical implementation, UI/UX, edge cases, concerns, and tradeoffs.
Don't ask obvious questions, dig into the hard parts I might not have considered.

Keep interviewing until we've covered everything, then write a complete spec to
SPEC.md.
```

Setelah spesifikasi selesai, mulai sesi segar untuk menjalankannya. Sesi baru memiliki konteks bersih yang fokus sepenuhnya pada implementasi, dan Anda memiliki spesifikasi tertulis untuk direferensikan.

Kelola sesi Anda

Percakapan bersifat persisten dan dapat dibalik. Gunakan ini untuk keuntungan Anda!

Perbaiki arah dengan cepat dan sering

> **Tip:**
>
> Perbaiki Claude segera setelah Anda melihatnya keluar jalur.

Hasil terbaik datang dari loop umpan balik yang ketat. Meskipun Claude kadang-kadang menyelesaikan masalah dengan sempurna pada upaya pertama, memperbaikinya dengan cepat umumnya menghasilkan solusi yang lebih baik lebih cepat.

- `Esc` : hentikan Claude di tengah-tindakan dengan tombol `Esc` . Konteks dipertahankan, jadi Anda dapat mengarahkan kembali.
- `Esc + Esc` **atau** `/rewind` : tekan `Esc` dua kali atau jalankan `/rewind` untuk membuka menu rewind dan mengembalikan percakapan dan status kode sebelumnya, atau ringkas dari pesan yang dipilih.
- `"Undo that"` : biarkan Claude mengembalikan perubahannya.
- `/clear` : atur ulang konteks antara tugas yang tidak terkait. Sesi panjang dengan konteks yang tidak relevan dapat mengurangi kinerja.

Jika Anda telah memperbaiki Claude lebih dari dua kali pada masalah yang sama dalam satu sesi, konteks penuh dengan pendekatan yang gagal. Jalankan `/clear` dan mulai segar dengan prompt yang lebih spesifik yang menggabungkan apa yang Anda pelajari. Sesi bersih dengan prompt yang lebih baik hampir selalu mengungguli sesi panjang dengan koreksi terakumulasi.

Kelola konteks secara agresif

> **Tip:**
>
> Jalankan `/clear` antara tugas yang tidak terkait untuk mengatur ulang konteks.

Claude Code secara otomatis mengompaksi riwayat percakapan saat Anda mendekati batas konteks, yang mempertahankan kode dan keputusan penting sambil membebaskan ruang.

Selama sesi panjang, jendela konteks Claude dapat terisi dengan percakapan yang tidak relevan, konten file, dan perintah. Ini dapat mengurangi kinerja dan kadang-kadang mengalihkan Claude.

- Gunakan `/clear` sering antara tugas untuk mengatur ulang jendela konteks sepenuhnya
- Ketika auto compaction dipicu, Claude meringkas apa yang paling penting, termasuk pola kode, status file, dan keputusan kunci
- Untuk kontrol lebih, jalankan `/compact <instructions>`, seperti `/compact Focus on the API changes`
- Untuk mengompaksi hanya bagian dari percakapan, gunakan `Esc + Esc` atau `/rewind`, pilih checkpoint pesan, dan pilih **Summarize from here**. Ini mengondensasi pesan dari titik itu maju sambil menjaga konteks awal tetap utuh.
- Sesuaikan perilaku compaction di CLAUDE.md dengan instruksi seperti `"When compacting, always preserve the full list of modified files and any test commands"` untuk memastikan konteks kritis bertahan dari ringkasan
- Untuk pertanyaan cepat yang tidak perlu tetap dalam konteks, gunakan `/btw`. Jawabannya muncul dalam overlay yang dapat ditutup dan tidak pernah memasuki riwayat percakapan, jadi Anda dapat memeriksa detail tanpa menumbuhkan konteks.

Gunakan subagent untuk investigasi

> **Tip:**

> Delegasikan penelitian dengan `"use subagents to investigate X"`. Mereka mengeksplorasi dalam konteks terpisah, menjaga percakapan utama Anda bersih untuk implementasi.

Karena konteks adalah batasan fundamental Anda, subagent adalah salah satu alat paling kuat yang tersedia. Ketika Claude meneliti basis kode, ia membaca banyak file, semuanya mengonsumsi konteks Anda. Subagent berjalan dalam jendela konteks terpisah dan melaporkan kembali ringkasan:

```
Use subagents to investigate how our authentication system handles token
refresh, and whether we have any existing OAuth utilities I should reuse.
```

Subagent mengeksplorasi basis kode, membaca file yang relevan, dan melaporkan kembali dengan temuan, semuanya tanpa mengacaukan percakapan utama Anda.

Anda juga dapat menggunakan subagent untuk verifikasi setelah Claude mengimplementasikan sesuatu:

```
use a subagent to review this code for edge cases
```

Rewind dengan checkpoint

> **Tip:**
>
> Setiap tindakan yang dilakukan Claude membuat checkpoint. Anda dapat mengembalikan percakapan, kode, atau keduanya ke checkpoint sebelumnya.

Claude secara otomatis membuat checkpoint sebelum perubahan. Tekan Escape dua kali atau jalankan `/rewind` untuk membuka menu rewind. Anda dapat mengembalikan percakapan saja, mengembalikan kode saja, mengembalikan keduanya, atau meringkas dari pesan yang dipilih. Lihat Checkpointing untuk detail.

Alih-alih merencanakan setiap langkah dengan hati-hati, Anda dapat memberi tahu Claude untuk mencoba sesuatu yang berisiko. Jika tidak berhasil, rewind dan coba pendekatan berbeda. Checkpoint bertahan di seluruh sesi, jadi Anda dapat menutup terminal dan masih rewind nanti.

> **Warning:**

> Checkpoint hanya melacak perubahan yang dibuat *oleh Claude*, bukan proses eksternal. Ini bukan pengganti git.

Lanjutkan percakapan

> **Tip:**
>
> Jalankan `claude --continue` untuk melanjutkan dari mana Anda tinggalkan, atau `--resume` untuk memilih dari sesi terbaru.

Claude Code menyimpan percakapan secara lokal. Ketika tugas mencakup beberapa sesi, Anda tidak harus menjelaskan ulang konteksnya:

```
claude --continue    # Resume the most recent conversation
claude --resume      # Select from recent conversations
```

Gunakan `/rename` untuk memberikan sesi nama deskriptif seperti `"oauth-migration"` atau `"debugging-memory-leak"` sehingga Anda dapat menemukannya nanti. Perlakukan sesi seperti cabang: alur kerja yang berbeda dapat memiliki konteks terpisah dan persisten.

Otomatisasi dan skalakan

Setelah Anda efektif dengan satu Claude, kalikan output Anda dengan sesi paralel, mode non-interaktif, dan pola fan-out.

Semuanya sejauh ini mengasumsikan satu manusia, satu Claude, dan satu percakapan. Tetapi Claude Code skalakan secara horizontal. Teknik di bagian ini menunjukkan bagaimana Anda dapat melakukan lebih banyak.

Jalankan mode non-interaktif

> **Tip:**
>
> Gunakan `claude -p "prompt"` di CI, pre-commit hooks, atau skrip. Tambahkan `--output-format stream-json` untuk output JSON streaming.

Dengan `claude -p "your prompt"`, Anda dapat menjalankan Claude secara non-interaktif, tanpa sesi. Mode non-interaktif adalah cara Anda mengintegrasikan Claude ke dalam pipeline CI, pre-commit hooks, atau alur kerja otomatis apa pun. Format output memungkinkan Anda mengurai hasil secara terprogram: teks biasa, JSON, atau JSON streaming.

```
## One-off queries
claude -p "Explain what this project does"

## Structured output for scripts
claude -p "List all API endpoints" --output-format json

## Streaming for real-time processing
claude -p "Analyze this log file" --output-format stream-json
```

Jalankan beberapa sesi Claude

> **Tip:**
>
> Jalankan beberapa sesi Claude secara paralel untuk mempercepat pengembangan, menjalankan eksperimen terisolasi, atau memulai alur kerja kompleks.

Ada tiga cara utama untuk menjalankan sesi paralel:

- Aplikasi desktop Claude Code: Kelola beberapa sesi lokal secara visual. Setiap sesi mendapat worktree terisolasi sendiri.
- Claude Code di web: Jalankan di infrastruktur cloud aman Anthropic dalam VM terisolasi.
- Tim agen: Koordinasi otomatis dari beberapa sesi dengan tugas bersama, pesan, dan pemimpin tim.

Selain paralelisasi pekerjaan, beberapa sesi memungkinkan alur kerja yang berfokus pada kualitas. Konteks segar meningkatkan tinjauan kode karena Claude tidak akan bias terhadap kode yang baru saja ditulisnya.

Sebagai contoh, gunakan pola Writer/Reviewer:

Sesi A (Penulis)	Sesi B (Peninjau)
`Implementasikan rate limiter untuk endpoint API kami`	
	`Tinjau implementasi rate limiter di @src/middleware/rateLimiter.ts. Cari kasus tepi, kondisi race, dan konsistensi dengan pola middleware yang ada.`
`Berikut adalah umpan balik tinjauan: [output Sesi B]. Tangani masalah ini.`	

Anda dapat melakukan sesuatu yang serupa dengan tes: biarkan satu Claude menulis tes, kemudian yang lain menulis kode untuk lulus.

Fan out di seluruh file

> **Tip:**
>
> Loop melalui tugas memanggil `claude -p` untuk masing-masing. Gunakan `--allowedTools` untuk cakupan izin untuk operasi batch.

Untuk migrasi besar atau analisis, Anda dapat mendistribusikan pekerjaan di seluruh banyak invokasi Claude paralel:

Step 1: Hasilkan daftar tugas

Biarkan Claude membuat daftar semua file yang perlu dimigrasikan (misalnya, `list all 2,000 Python files that need migrating`)

Step 2: Tulis skrip untuk loop melalui daftar

```
for file in $(cat files.txt); do
  claude -p "Migrate $file from React to Vue. Return OK or FAIL." \
    --allowedTools "Edit,Bash(git commit *)"
done
```

Step 3: Uji pada beberapa file, kemudian jalankan dalam skala

Perbaiki prompt Anda berdasarkan apa yang salah dengan 2-3 file pertama, kemudian jalankan pada set lengkap. Bendera `--allowedTools` membatasi apa yang dapat dilakukan Claude, yang penting ketika Anda menjalankan tanpa pengawasan.

Anda juga dapat mengintegrasikan Claude ke dalam pipeline pemrosesan/data yang ada:

```
claude -p "<your prompt>" --output-format json | your_command
```

Gunakan `--verbose` untuk debugging selama pengembangan, dan matikan dalam produksi.

Hindari pola kegagalan umum

Ini adalah kesalahan umum. Mengenalinya lebih awal menghemat waktu:

- **Sesi kitchen sink.** Anda mulai dengan satu tugas, kemudian meminta Claude sesuatu yang tidak terkait, kemudian kembali ke tugas pertama. Konteks penuh dengan informasi yang tidak relevan. > **Perbaikan**: `/clear` antara tugas yang tidak terkait.
- **Mengoreksi berulang kali.** Claude melakukan sesuatu yang salah, Anda memperbaikinya, masih salah, Anda memperbaiki lagi. Konteks tercemar dengan pendekatan yang gagal. > **Perbaikan**: Setelah dua koreksi yang gagal, `/clear` dan tulis prompt awal yang lebih baik menggabungkan apa yang Anda pelajari.
- **CLAUDE.md yang terlalu spesifik.** Jika CLAUDE.md Anda terlalu panjang, Claude mengabaikan setengahnya karena aturan penting hilang dalam kebisingan. > **Perbaikan**: Pangkas tanpa ampun. Jika Claude sudah melakukan sesuatu dengan benar tanpa instruksi, hapus atau ubah menjadi hook.
- **Kesenjangan kepercayaan-kemudian-verifikasi.** Claude menghasilkan implementasi yang terlihat masuk akal tetapi tidak menangani kasus tepi. > **Perbaikan**: Selalu berikan verifikasi (tes, skrip, tangkapan layar). Jika Anda tidak dapat memverifikasinya, jangan kirimkan.
- **Eksplorasi tak terbatas.** Anda meminta Claude untuk "menyelidiki" sesuatu tanpa membatasinya. Claude membaca ratusan file, mengisi konteks. > **Perbaikan**: Batasi investigasi secara sempit atau gunakan subagent sehingga eksplorasi tidak mengonsumsi konteks utama Anda.

Kembangkan intuisi Anda

Pola dalam panduan ini bukan batu loncatan. Mereka adalah titik awal yang bekerja dengan baik secara umum, tetapi mungkin tidak optimal untuk setiap situasi.

Kadang-kadang Anda *harus* membiarkan konteks terakumulasi karena Anda mendalam dalam satu masalah kompleks dan riwayat berharga. Kadang-kadang Anda harus melewati perencanaan dan membiarkan Claude mengetahuinya karena tugas bersifat eksplorasi. Kadang-kadang prompt yang samar adalah tepat karena Anda ingin melihat bagaimana Claude menafsirkan masalah sebelum membatasinya.

Perhatikan apa yang berhasil. Ketika Claude menghasilkan output yang hebat, perhatikan apa yang Anda lakukan: struktur prompt, konteks yang Anda berikan, mode yang Anda gunakan. Ketika Claude berjuang, tanyakan mengapa. Apakah konteksnya terlalu bising? Prompt terlalu samar? Tugas terlalu besar untuk satu pass?

Seiring waktu, Anda akan mengembangkan intuisi yang tidak dapat ditangkap oleh panduan apa pun. Anda akan tahu kapan harus spesifik dan kapan harus terbuka, kapan harus merencanakan dan kapan harus mengeksplorasi, kapan harus menghapus konteks dan kapan harus membiarkannya terakumulasi.

Sumber daya terkait

- Cara Claude Code Bekerja: loop agentic, alat, dan manajemen konteks
- Perluas Claude Code: skills, hooks, MCP, subagent, dan plugins
- Alur kerja umum: resep langkah demi langkah untuk debugging, pengujian, PR, dan lainnya
- CLAUDE.md: simpan konvensi proyek dan konteks persisten

Alur kerja umum

> *Panduan langkah demi langkah untuk menjelajahi basis kode, memperbaiki bug, refactoring, pengujian, dan tugas sehari-hari lainnya dengan Claude Code.*

Halaman ini mencakup alur kerja praktis untuk pengembangan sehari-hari: menjelajahi kode yang tidak familiar, debugging, refactoring, menulis tes, membuat PR, dan mengelola sesi. Setiap bagian mencakup contoh prompt yang dapat Anda sesuaikan dengan proyek Anda sendiri. Untuk pola dan tips tingkat yang lebih tinggi, lihat Best practices.

Pahami basis kode baru

Dapatkan gambaran umum basis kode dengan cepat

Misalkan Anda baru saja bergabung dengan proyek baru dan perlu memahami strukturnya dengan cepat.

Step 1: Navigasi ke direktori root proyek

```
cd /path/to/project
```

Step 2: Mulai Claude Code

```
claude
```

Step 3: Minta gambaran umum tingkat tinggi

```
give me an overview of this codebase
```

Step 4: Selami komponen spesifik lebih dalam

```
explain the main architecture patterns used here
```

```
what are the key data models?
```

```
how is authentication handled?
```

> **Tip:**
>
> Tips:
>
> - Mulai dengan pertanyaan luas, kemudian fokus ke area spesifik
> - Tanyakan tentang konvensi koding dan pola yang digunakan dalam proyek
> - Minta glosarium istilah khusus proyek

Temukan kode yang relevan

Misalkan Anda perlu menemukan kode yang terkait dengan fitur atau fungsionalitas tertentu.

Step 1: Minta Claude untuk menemukan file yang relevan

```
find the files that handle user authentication
```

Step 2: Dapatkan konteks tentang cara komponen berinteraksi

```
how do these authentication files work together?
```

Step 3: Pahami alur eksekusi

```
trace the login process from front-end to database
```

> **Tip:**
>
> Tips:
>
> - Jadilah spesifik tentang apa yang Anda cari
> - Gunakan bahasa domain dari proyek
> - Instal code intelligence plugin untuk bahasa Anda untuk memberikan Claude navigasi “go to definition” dan “find references” yang presisi

---

Perbaiki bug secara efisien

Misalkan Anda telah mengalami pesan kesalahan dan perlu menemukan dan memperbaiki sumbernya.

Step 1: Bagikan kesalahan dengan Claude

```
I'm seeing an error when I run npm test
```

Step 2: Minta rekomendasi perbaikan

```
suggest a few ways to fix the @ts-ignore in user.ts
```

Step 3: Terapkan perbaikan

```
update user.ts to add the null check you suggested
```

> **Tip:**
>
> Tips:
>
> - Beri tahu Claude perintah untuk mereproduksi masalah dan dapatkan stack trace
> - Sebutkan langkah apa pun untuk mereproduksi kesalahan
> - Beri tahu Claude jika kesalahan bersifat intermiten atau konsisten

---

Refactor kode

Misalkan Anda perlu memperbarui kode lama untuk menggunakan pola dan praktik modern.

Step 1: Identifikasi kode legacy untuk refactoring

```
find deprecated API usage in our codebase
```

Step 2: Dapatkan rekomendasi refactoring

```
suggest how to refactor utils.js to use modern JavaScript features
```

Step 3: Terapkan perubahan dengan aman

```
refactor utils.js to use ES2024 features while maintaining the same behavior
```

Step 4: Verifikasi refactoring

```
run tests for the refactored code
```

> **Tip:**
>
> Tips:
>
> - Minta Claude untuk menjelaskan manfaat pendekatan modern
> - Minta agar perubahan mempertahankan kompatibilitas backward ketika diperlukan
> - Lakukan refactoring dalam increment kecil yang dapat diuji

---

Gunakan subagents khusus

Misalkan Anda ingin menggunakan subagents AI khusus untuk menangani tugas spesifik dengan lebih efektif.

Step 1: Lihat subagents yang tersedia

```
/agents
```

Ini menampilkan semua subagents yang tersedia dan memungkinkan Anda membuat yang baru.

Step 2: Gunakan subagents secara otomatis

Claude Code secara otomatis mendelegasikan tugas yang sesuai ke subagents khusus:

```
review my recent code changes for security issues
```

```
run all tests and fix any failures
```

Step 3: Minta subagents spesifik secara eksplisit

```
use the code-reviewer subagent to check the auth module
```

```
have the debugger subagent investigate why users can't log in
```

Step 4: Buat subagents kustom untuk alur kerja Anda

```
/agents
```

Kemudian pilih “Create New subagent” dan ikuti prompt untuk mendefinisikan:

- Pengenal unik yang menggambarkan tujuan subagent (misalnya, `code-reviewer`, `api-designer`).
- Kapan Claude harus menggunakan agen ini
- Alat mana yang dapat diaksesnya
- Prompt sistem yang menggambarkan peran dan perilaku agen

> **Tip:**
>
> Tips:
>
> - Buat subagents khusus proyek di `.claude/agents/` untuk berbagi tim
> - Gunakan field `description` deskriptif untuk mengaktifkan delegasi otomatis
> - Batasi akses alat ke apa yang benar-benar dibutuhkan setiap subagent
> - Periksa dokumentasi subagents untuk contoh terperinci

Gunakan Plan Mode untuk analisis kode yang aman

Plan Mode menginstruksikan Claude untuk membuat rencana dengan menganalisis basis kode dengan operasi read-only, sempurna untuk menjelajahi basis kode, merencanakan perubahan kompleks, atau meninjau kode dengan aman. Dalam Plan Mode, Claude menggunakan `AskUserQuestion` untuk mengumpulkan persyaratan dan memperjelas tujuan Anda sebelum mengusulkan rencana.

Kapan menggunakan Plan Mode

- **Implementasi multi-langkah**: Ketika fitur Anda memerlukan pengeditan ke banyak file
- **Eksplorasi kode**: Ketika Anda ingin meneliti basis kode secara menyeluruh sebelum mengubah apa pun
- **Pengembangan interaktif**: Ketika Anda ingin mengulangi arah dengan Claude

Cara menggunakan Plan Mode

Aktifkan Plan Mode selama sesi

Anda dapat beralih ke Plan Mode selama sesi menggunakan **Shift+Tab** untuk bersiklus melalui mode izin.

Jika Anda berada dalam Normal Mode, **Shift+Tab** pertama kali beralih ke Auto-Accept Mode, ditunjukkan oleh `⏵⏵ accept edits on` di bagian bawah terminal. **Shift+Tab** berikutnya akan beralih ke Plan Mode, ditunjukkan oleh `⏸ plan mode on`.

Mulai sesi baru dalam Plan Mode

Untuk memulai sesi baru dalam Plan Mode, gunakan flag `--permission-mode plan`:

```
claude --permission-mode plan
```

Jalankan query "headless" dalam Plan Mode

Anda juga dapat menjalankan query dalam Plan Mode secara langsung dengan `-p` (yaitu, dalam "headless mode"):

```
claude --permission-mode plan -p "Analyze the authentication system and suggest improvements"
```

Contoh: Merencanakan refactor kompleks

```
claude --permission-mode plan
```

```
I need to refactor our authentication system to use OAuth2. Create a detailed migration plan.
```

Claude menganalisis implementasi saat ini dan membuat rencana komprehensif. Perbaiki dengan tindak lanjut:

```
What about backward compatibility?
```

```
How should we handle database migration?
```

> **Tip:**
>
> Tekan `Ctrl+G` untuk membuka rencana di editor teks default Anda, di mana Anda dapat mengeditnya secara langsung sebelum Claude melanjutkan.

Konfigurasikan Plan Mode sebagai default

```
// .claude/settings.json
{
  "permissions": {
    "defaultMode": "plan"
  }
}
```

Lihat dokumentasi settings untuk opsi konfigurasi lainnya.

Bekerja dengan tes

Misalkan Anda perlu menambahkan tes untuk kode yang tidak tercakup.

Step 1: Identifikasi kode yang tidak diuji

```
find functions in NotificationsService.swift that are not covered by tests
```

Step 2: Hasilkan scaffolding tes

```
add tests for the notification service
```

Step 3: Tambahkan kasus tes yang bermakna

```
add test cases for edge conditions in the notification service
```

Step 4: Jalankan dan verifikasi tes

```
run the new tests and fix any failures
```

Claude dapat menghasilkan tes yang mengikuti pola dan konvensi yang ada dalam proyek Anda. Saat meminta tes, jadilah spesifik tentang perilaku apa yang ingin Anda verifikasi. Claude memeriksa file tes yang ada untuk mencocokkan gaya, framework, dan pola pernyataan yang sudah digunakan.

Untuk cakupan komprehensif, minta Claude untuk mengidentifikasi kasus tepi yang mungkin Anda lewatkan. Claude dapat menganalisis jalur kode Anda dan menyarankan tes untuk kondisi kesalahan, nilai batas, dan input yang tidak terduga yang mudah diabaikan.

Buat pull request

Anda dapat membuat pull request dengan meminta Claude secara langsung (“create a pr for my changes”), atau memandu Claude melaluinya langkah demi langkah:

Step 1: Ringkas perubahan Anda

```
summarize the changes I've made to the authentication module
```

Step 2: Hasilkan pull request

```
create a pr
```

Step 3: Tinjau dan perbaiki

```
enhance the PR description with more context about the security improvements
```

Ketika Anda membuat PR menggunakan `gh pr create`, sesi secara otomatis ditautkan ke PR tersebut. Anda dapat melanjutkannya nanti dengan `claude --from-pr <number>`.

> **Tip:**
>
> Tinjau PR yang dihasilkan Claude sebelum mengirimkan dan minta Claude untuk menyoroti risiko atau pertimbangan potensial.

Tangani dokumentasi

Misalkan Anda perlu menambah atau memperbarui dokumentasi untuk kode Anda.

Step 1: Identifikasi kode yang tidak terdokumentasi

```
find functions without proper JSDoc comments in the auth module
```

Step 2: Hasilkan dokumentasi

```
add JSDoc comments to the undocumented functions in auth.js
```

Step 3: Tinjau dan tingkatkan

```
improve the generated documentation with more context and examples
```

Step 4: Verifikasi dokumentasi

```
check if the documentation follows our project standards
```

> **Tip:**
>
> Tips:
>
> - Tentukan gaya dokumentasi yang Anda inginkan (JSDoc, docstrings, dll.)
> - Minta contoh dalam dokumentasi
> - Minta dokumentasi untuk API publik, antarmuka, dan logika kompleks

---

Bekerja dengan gambar

Misalkan Anda perlu bekerja dengan gambar dalam basis kode Anda, dan Anda ingin bantuan Claude dalam menganalisis konten gambar.

Step 1: Tambahkan gambar ke percakapan

Anda dapat menggunakan salah satu metode ini:

1. Seret dan lepas gambar ke jendela Claude Code
2. Salin gambar dan tempel ke CLI dengan ctrl+v (Jangan gunakan cmd+v)
3. Berikan jalur gambar ke Claude. Misalnya, “Analyze this image: /path/to/your/image.png”

Step 2: Minta Claude untuk menganalisis gambar

```
What does this image show?
```

```
Describe the UI elements in this screenshot
```

```
Are there any problematic elements in this diagram?
```

Step 3: Gunakan gambar untuk konteks

```
Here's a screenshot of the error. What's causing it?
```

```
This is our current database schema. How should we modify it for the new feature?
```

Step 4: Dapatkan saran kode dari konten visual

```
Generate CSS to match this design mockup
```

```
What HTML structure would recreate this component?
```

> **Tip:**
>
> Tips:
>
> - Gunakan gambar ketika deskripsi teks akan tidak jelas atau merepotkan
> - Sertakan tangkapan layar kesalahan, desain UI, atau diagram untuk konteks yang lebih baik

- Anda dapat bekerja dengan beberapa gambar dalam percakapan
- Analisis gambar bekerja dengan diagram, tangkapan layar, mockup, dan lainnya
- Ketika Claude mereferensikan gambar (misalnya, `[Image #1]`), `Cmd+Click` (Mac) atau `Ctrl+Click` (Windows/Linux) tautan untuk membuka gambar di penampil default Anda

---

File dan direktori referensi

Gunakan @ untuk dengan cepat menyertakan file atau direktori tanpa menunggu Claude membacanya.

Step 1: Referensikan file tunggal

```
Explain the logic in @src/utils/auth.js
```

Ini menyertakan konten lengkap file dalam percakapan.

Step 2: Referensikan direktori

```
What's the structure of @src/components?
```

Ini menyediakan daftar direktori dengan informasi file.

Step 3: Referensikan sumber daya MCP

```
Show me the data from @github:repos/owner/repo/issues
```

Ini mengambil data dari server MCP yang terhubung menggunakan format @server:resource. Lihat sumber daya MCP untuk detail.

Tip:

Tips:

- Jalur file dapat relatif atau absolut
- Referensi file @ menambahkan `CLAUDE.md` di direktori file dan direktori induk ke konteks
- Referensi direktori menampilkan daftar file, bukan konten
- Anda dapat mereferensikan beberapa file dalam satu pesan (misalnya, “@file1.js and @file2.js”)

Gunakan extended thinking (thinking mode)

Extended thinking diaktifkan secara default, memberikan Claude ruang untuk bernalar melalui masalah kompleks langkah demi langkah sebelum merespons. Penalaran ini terlihat dalam verbose mode, yang dapat Anda aktifkan dengan `Ctrl+O`.

Selain itu, Opus 4.6 memperkenalkan adaptive reasoning: alih-alih anggaran token thinking yang tetap, model secara dinamis mengalokasikan thinking berdasarkan pengaturan effort level Anda. Extended thinking dan adaptive reasoning bekerja bersama untuk memberi Anda kontrol atas seberapa dalam Claude bernalar sebelum merespons.

Extended thinking sangat berharga untuk keputusan arsitektur kompleks, bug yang menantang, perencanaan implementasi multi-langkah, dan mengevaluasi trade-off antara pendekatan yang berbeda.

> **Note:**
>
> Frasa seperti "think", "think hard", dan "think more" diinterpretasikan sebagai instruksi prompt reguler dan tidak mengalokasikan token thinking.

Konfigurasikan thinking mode

Thinking diaktifkan secara default, tetapi Anda dapat menyesuaikan atau menonaktifkannya.

Scope	Cara mengkonfigurasi	Detail
Effort level	Sesuaikan di `/model` atau atur `CLAUDE_CODE_EFFORT_LEVEL`	Kontrol kedalaman thinking untuk Opus 4.6 dan Sonnet 4.6: low, medium, high. Lihat Adjust effort level
Kata kunci `ultrathink`	Sertakan "ultrathink" di mana saja dalam prompt Anda	Menetapkan effort ke high untuk giliran itu pada Opus 4.6 dan Sonnet 4.6. Berguna untuk tugas sekali jadi yang memerlukan penalaran mendalam tanpa mengubah pengaturan effort Anda secara permanen

Scope	Cara mengkonfigurasi	Detail
Pintasan toggle	Tekan `Option+T` (macOS) atau `Alt+T` (Windows/Linux)	Toggle thinking on/off untuk sesi saat ini (semua model). Mungkin memerlukan konfigurasi terminal untuk mengaktifkan pintasan tombol Option
Default global	Gunakan `/config` untuk toggle thinking mode	Menetapkan default Anda di semua proyek (semua model). Disimpan sebagai `alwaysThinkingEnabled` di `~/.claude/settings.json`
Batasi anggaran token	Atur variabel lingkungan `MAX_THINKING_TOKENS`	Batasi anggaran thinking ke jumlah token tertentu (diabaikan pada Opus 4.6 kecuali diatur ke 0). Contoh: `export MAX_THINKING_TOKENS=10000`

Untuk melihat proses thinking Claude, tekan `Ctrl+O` untuk toggle verbose mode dan lihat penalaran internal ditampilkan sebagai teks italic abu-abu.

Cara kerja extended thinking

Extended thinking mengontrol berapa banyak penalaran internal yang dilakukan Claude sebelum merespons. Lebih banyak thinking memberikan lebih banyak ruang untuk menjelajahi solusi, menganalisis kasus tepi, dan memperbaiki kesalahan sendiri.

Dengan Opus 4.6, thinking menggunakan adaptive reasoning: model secara dinamis mengalokasikan token thinking berdasarkan effort level yang Anda pilih (low, medium, high). Ini adalah cara yang direkomendasikan untuk menyesuaikan trade-off antara kecepatan dan kedalaman penalaran.

Dengan model lain, thinking menggunakan anggaran tetap hingga 31.999 token dari anggaran output Anda. Anda dapat membatasinya dengan variabel lingkungan `MAX_THINKING_TOKENS`, atau menonaktifkan thinking sepenuhnya melalui `/config` atau toggle `Option+T` / `Alt+T`.

`MAX_THINKING_TOKENS` diabaikan pada Opus 4.6 dan Sonnet 4.6, karena adaptive reasoning mengontrol kedalaman thinking sebagai gantinya. Satu pengecualian: mengatur `MAX_THINKING_TOKENS=0` masih menonaktifkan thinking sepenuhnya pada model apa pun. Untuk menonaktifkan adaptive thinking dan kembali ke anggaran thinking tetap, atur `CLAUDE_CODE_DISABLE_ADAPTIVE_THINKING=1` . Lihat variabel lingkungan.

> **Warning:**
>
> Anda dikenakan biaya untuk semua token thinking yang digunakan, meskipun model Claude 4 menampilkan thinking yang diringkas

---

Lanjutkan percakapan sebelumnya

Saat memulai Claude Code, Anda dapat melanjutkan sesi sebelumnya:

- `claude --continue` melanjutkan percakapan terbaru di direktori saat ini
- `claude --resume` membuka pemilih percakapan atau melanjutkan berdasarkan nama
- `claude --from-pr 123` melanjutkan sesi yang ditautkan ke pull request tertentu

Dari dalam sesi aktif, gunakan `/resume` untuk beralih ke percakapan berbeda.

Sesi disimpan per direktori proyek. Pemilih `/resume` menampilkan sesi dari repositori git yang sama, termasuk worktrees.

Beri nama sesi Anda

Berikan sesi nama deskriptif untuk menemukannya nanti. Ini adalah praktik terbaik saat bekerja pada beberapa tugas atau fitur.

Step 1: Beri nama sesi saat ini

Gunakan `/rename` selama sesi untuk memberikannya nama yang mudah diingat:

```
/rename auth-refactor
```

Anda juga dapat mengganti nama sesi apa pun dari pemilih: jalankan `/resume` , navigasi ke sesi, dan tekan `R` .

Step 2: Lanjutkan berdasarkan nama nanti

Dari baris perintah:

```
claude --resume auth-refactor
```

Atau dari dalam sesi aktif:

```
/resume auth-refactor
```

Gunakan pemilih sesi

Perintah `/resume` (atau `claude --resume` tanpa argumen) membuka pemilih sesi interaktif dengan fitur-fitur ini:

Pintasan keyboard dalam pemilih:

Pintasan	Tindakan
`↑` / `↓`	Navigasi antar sesi
`→` / `←`	Perluas atau tutup sesi yang dikelompokkan
`Enter`	Pilih dan lanjutkan sesi yang disorot
`P`	Pratinjau konten sesi
`R`	Ganti nama sesi yang disorot
`/`	Cari untuk memfilter sesi
`A`	Toggle antara direktori saat ini dan semua proyek
`B`	Filter ke sesi dari cabang git saat ini Anda
`Esc`	Keluar dari pemilih atau mode pencarian

Organisasi sesi:

Pemilih menampilkan sesi dengan metadata yang membantu:

- Nama sesi atau prompt awal
- Waktu yang telah berlalu sejak aktivitas terakhir
- Jumlah pesan
- Cabang Git (jika berlaku)

Sesi yang di-fork (dibuat dengan `/rewind` atau `--fork-session`) dikelompokkan bersama di bawah sesi root mereka, memudahkan menemukan percakapan terkait.

Tip:

Tips:

- **Beri nama sesi lebih awal**: Gunakan `/rename` saat memulai pekerjaan pada tugas yang berbeda—jauh lebih mudah menemukan "payment-integration" daripada "explain this function" nanti
- Gunakan `--continue` untuk akses cepat ke percakapan terbaru Anda di direktori saat ini
- Gunakan `--resume session-name` ketika Anda tahu sesi mana yang Anda butuhkan
- Gunakan `--resume` (tanpa nama) ketika Anda perlu menjelajahi dan memilih
- Untuk skrip, gunakan `claude --continue --print "prompt"` untuk melanjutkan dalam mode non-interaktif
- Tekan `P` dalam pemilih untuk melihat pratinjau sesi sebelum melanjutkannya
- Percakapan yang dilanjutkan dimulai dengan model dan konfigurasi yang sama dengan yang asli

Cara kerjanya:

1. **Penyimpanan Percakapan**: Semua percakapan secara otomatis disimpan secara lokal dengan riwayat pesan lengkap mereka
2. **Deserialisasi Pesan**: Saat melanjutkan, seluruh riwayat pesan dipulihkan untuk mempertahankan konteks
3. **Status Alat**: Penggunaan alat dan hasil dari percakapan sebelumnya dipertahankan
4. **Pemulihan Konteks**: Percakapan dilanjutkan dengan semua konteks sebelumnya utuh

---

Jalankan sesi Claude Code paralel dengan Git worktrees

Saat bekerja pada beberapa tugas sekaligus, Anda memerlukan setiap sesi Claude untuk memiliki salinan basis kode sendiri sehingga perubahan tidak bertabrakan. Git worktrees menyelesaikan ini dengan membuat direktori kerja terpisah yang masing-masing memiliki file dan cabang sendiri, sambil berbagi riwayat repositori dan koneksi remote yang sama. Ini berarti Anda dapat memiliki Claude bekerja pada fitur di satu worktree sambil memperbaiki bug di worktree lain, tanpa sesi apa pun mengganggu yang lain.

Gunakan flag `--worktree` (`-w`) untuk membuat worktree terisolasi dan memulai Claude di dalamnya. Nilai yang Anda berikan menjadi nama direktori worktree dan nama cabang:

```
## Mulai Claude dalam worktree bernama "feature-auth"
## Membuat .claude/worktrees/feature-auth/ dengan cabang baru
claude --worktree feature-auth

## Mulai sesi lain dalam worktree terpisah
claude --worktree bugfix-123
```

Jika Anda menghilangkan nama, Claude secara otomatis menghasilkan nama acak:

```
## Auto-generates a name like "bright-running-fox"
claude --worktree
```

Worktrees dibuat di `<repo>/.claude/worktrees/<name>` dan bercabang dari cabang remote default. Cabang worktree dinamai `worktree-<name>`.

Anda juga dapat meminta Claude untuk "work in a worktree" atau "start a worktree" selama sesi, dan itu akan membuat satu secara otomatis.

Worktrees subagent

Subagents juga dapat menggunakan isolasi worktree untuk bekerja secara paralel tanpa konflik. Minta Claude untuk "use worktrees for your agents" atau konfigurasikan di custom subagent dengan menambahkan `isolation: worktree` ke frontmatter agen. Setiap subagent mendapatkan worktree sendiri yang secara otomatis dibersihkan ketika subagent selesai tanpa perubahan.

Pembersihan worktree

Saat Anda keluar dari sesi worktree, Claude menangani pembersihan berdasarkan apakah Anda membuat perubahan:

- **Tidak ada perubahan**: worktree dan cabangnya dihapus secara otomatis
- **Perubahan atau commit ada**: Claude meminta Anda untuk menyimpan atau menghapus worktree. Menyimpan mempertahankan direktori dan cabang sehingga Anda dapat kembali nanti. Menghapus menghapus direktori worktree dan cabangnya, membuang semua perubahan yang tidak dilakukan dan commit

Untuk membersihkan worktrees di luar sesi Claude, gunakan manajemen worktree manual.

> **Tip:**
>
> Tambahkan `.claude/worktrees/` ke `.gitignore` Anda untuk mencegah konten worktree muncul sebagai file yang tidak dilacak dalam repositori utama Anda.

Kelola worktrees secara manual

Untuk kontrol lebih besar atas lokasi worktree dan konfigurasi cabang, buat worktrees dengan Git secara langsung. Ini berguna ketika Anda perlu checkout cabang yang ada tertentu atau menempatkan worktree di luar repositori.

```
## Buat worktree dengan cabang baru
git worktree add ../project-feature-a -b feature-a

## Buat worktree dengan cabang yang ada
git worktree add ../project-bugfix bugfix-123

## Mulai Claude dalam worktree
cd ../project-feature-a && claude

## Bersihkan saat selesai
git worktree list
git worktree remove ../project-feature-a
```

Pelajari lebih lanjut di dokumentasi Git worktree resmi.

> **Tip:**
>
> Ingat untuk menginisialisasi lingkungan pengembangan Anda di setiap worktree baru sesuai dengan setup proyek Anda. Tergantung pada stack Anda, ini mungkin termasuk menjalankan instalasi dependensi (`npm install` , `yarn`), menyiapkan lingkungan virtual, atau mengikuti proses setup standar proyek Anda.

Kontrol versi non-git

Isolasi worktree bekerja dengan git secara default. Untuk sistem kontrol versi lain seperti SVN, Perforce, atau Mercurial, konfigurasikan hook WorktreeCreate dan WorktreeRemove untuk menyediakan logika pembuatan dan pembersihan worktree kustom. Ketika dikonfigurasi, hook ini menggantikan perilaku git default saat Anda menggunakan `--worktree` .

Untuk koordinasi otomatis sesi paralel dengan tugas bersama dan pesan, lihat agent teams.

Dapatkan notifikasi ketika Claude membutuhkan perhatian Anda

Ketika Anda memulai tugas yang berjalan lama dan beralih ke jendela lain, Anda dapat menyiapkan notifikasi desktop sehingga Anda tahu ketika Claude selesai atau membutuhkan input Anda. Ini menggunakan event hook `Notification`, yang diaktifkan setiap kali Claude menunggu izin, idle dan siap untuk prompt baru, atau menyelesaikan autentikasi.

Step 1: Buka menu hooks

Ketik `/hooks` dan pilih `Notification` dari daftar event.

Step 2: Konfigurasikan matcher

Pilih `+ Match all (no filter)` untuk diaktifkan pada semua jenis notifikasi. Untuk memberi tahu hanya untuk event spesifik, pilih `+ Add new matcher…` dan masukkan salah satu nilai ini:

Matcher	Diaktifkan ketika
`permission_prompt`	Claude membutuhkan Anda untuk menyetujui penggunaan alat
`idle_prompt`	Claude selesai dan menunggu prompt berikutnya Anda
`auth_success`	Autentikasi selesai
`elicitation_dialog`	Claude mengajukan pertanyaan kepada Anda

Step 3: Tambahkan perintah notifikasi Anda

Pilih `+ Add new hook…` dan masukkan perintah untuk OS Anda:

macOS

Menggunakan `osascript` untuk memicu notifikasi macOS asli melalui AppleScript:

```
osascript -e 'display notification "Claude Code needs your attention" with title
"Claude Code"'
```

Linux

Menggunakan `notify-send`, yang sudah diinstal sebelumnya di sebagian besar desktop Linux dengan daemon notifikasi:

```
notify-send 'Claude Code' 'Claude Code needs your attention'
```

Windows (PowerShell)

Menggunakan PowerShell untuk menampilkan kotak pesan asli melalui Windows Forms .NET:

```
powershell.exe -Command
"[System.Reflection.Assembly]::LoadWithPartialName('System.Windows.Forms');
[System.Windows.Forms.MessageBox]::Show('Claude Code needs your attention',
'Claude Code')"
```

Step 4: Simpan ke pengaturan pengguna

Pilih `User settings` untuk menerapkan notifikasi di semua proyek Anda.

Untuk panduan lengkap dengan contoh konfigurasi JSON, lihat Automate workflows with hooks. Untuk skema event lengkap dan jenis notifikasi, lihat referensi Notification.

Gunakan Claude sebagai utilitas gaya unix

Tambahkan Claude ke proses verifikasi Anda

Misalkan Anda ingin menggunakan Claude Code sebagai linter atau code reviewer.

Tambahkan Claude ke skrip build Anda:

```
// package.json
{
    ...
    "scripts": {
        ...
        "lint:claude": "claude -p 'you are a linter. please look at the changes vs. main and report any issues related to typos. report the filename and line number on one line, and a description of the issue on the second line. do not return any other text.'"
    }
}
```

Tip:

Tips:

- Gunakan Claude untuk code review otomatis dalam pipeline CI/CD Anda
- Sesuaikan prompt untuk memeriksa masalah spesifik yang relevan dengan proyek Anda
- Pertimbangkan membuat beberapa skrip untuk jenis verifikasi yang berbeda

Pipe in, pipe out

Misalkan Anda ingin pipe data ke Claude, dan dapatkan kembali data dalam format ter-struktur.

Pipe data melalui Claude:

```
cat build-error.txt | claude -p 'concisely explain the root cause of this build error' > output.txt
```

Tip:

Tips:

- Gunakan pipe untuk mengintegrasikan Claude ke dalam skrip shell yang ada
- Gabungkan dengan alat Unix lain untuk alur kerja yang kuat
- Pertimbangkan menggunakan –output-format untuk output terstruktur

Kontrol format output

Misalkan Anda memerlukan output Claude dalam format tertentu, terutama saat mengintegrasikan Claude Code ke dalam skrip atau alat lain.

Step 1: Gunakan format teks (default)

```
cat data.txt | claude -p 'summarize this data' --output-format text > summary.txt
```

Ini menampilkan hanya respons teks biasa Claude (perilaku default).

Step 2: Gunakan format JSON

```
cat code.py | claude -p 'analyze this code for bugs' --output-format json > analysis.json
```

Ini menampilkan array JSON pesan dengan metadata termasuk biaya dan durasi.

Step 3: Gunakan format streaming JSON

```
cat log.txt | claude -p 'parse this log file for errors' --output-format stream-json
```

Ini menampilkan serangkaian objek JSON secara real-time saat Claude memproses permintaan. Setiap pesan adalah objek JSON yang valid, tetapi seluruh output bukan JSON yang valid jika digabungkan.

> **Tip:**
>
> Tips:
>
> - Gunakan `--output-format text` untuk integrasi sederhana di mana Anda hanya memerlukan respons Claude
> - Gunakan `--output-format json` ketika Anda memerlukan log percakapan lengkap
> - Gunakan `--output-format stream-json` untuk output real-time dari setiap giliran percakapan

Tanyakan Claude tentang kemampuannya

Claude memiliki akses bawaan ke dokumentasinya dan dapat menjawab pertanyaan tentang fitur dan keterbatasannya sendiri.

Contoh pertanyaan

```
can Claude Code create pull requests?
```

```
how does Claude Code handle permissions?
```

```
what skills are available?
```

```
how do I use MCP with Claude Code?
```

```
how do I configure Claude Code for Amazon Bedrock?
```

```
what are the limitations of Claude Code?
```

Note:

Claude memberikan jawaban berbasis dokumentasi untuk pertanyaan-pertanyaan ini. Untuk contoh yang dapat dieksekusi dan demonstrasi langsung, lihat bagian alur kerja spesifik di atas.

Tip:

Tips:

- Claude selalu memiliki akses ke dokumentasi Claude Code terbaru, terlepas dari versi yang Anda gunakan
- Ajukan pertanyaan spesifik untuk mendapatkan jawaban terperinci
- Claude dapat menjelaskan fitur kompleks seperti integrasi MCP, konfigurasi enterprise, dan alur kerja lanjutan

Langkah berikutnya

- Best practices Pola untuk mendapatkan hasil maksimal dari Claude Code
- Cara kerja Claude Code Pahami loop agentic dan manajemen konteks
- Perluas Claude Code Tambahkan skills, hooks, MCP, subagents, dan plugins
- Implementasi referensi Clone implementasi referensi container pengembangan kami

Part 3: Commands & Reference

Referensi CLI

> *Referensi lengkap untuk antarmuka baris perintah Claude Code, termasuk perintah dan flag.*

Perintah CLI

Anda dapat memulai sesi, menyalurkan konten, melanjutkan percakapan, dan mengelola pembaruan dengan perintah-perintah ini:

Perintah	Deskripsi	Contoh
`claude`	Mulai sesi interaktif	`claude`
`claude "query"`	Mulai sesi interaktif dengan prompt awal	`claude "explain this project"`
`claude -p "query"`	Kueri melalui SDK, kemudian keluar	`claude -p "explain this function"`
`cat file | claude -p "query"`	Proses konten yang disalurkan	`cat logs.txt | claude -p "explain"`
`claude -c`	Lanjutkan percakapan terbaru di direktori saat ini	`claude -c`
`claude -c -p "query"`	Lanjutkan melalui SDK	`claude -c -p "Check for type errors"`
`claude -r "<session>" "query"`	Lanjutkan sesi berdasarkan ID atau nama	`claude -r "auth-refactor" "Finish this PR"`
`claude update`	Perbarui ke versi terbaru	`claude update`
`claude auth login`	Masuk ke akun Anthropic Anda. Gunakan `--email` untuk mengisi email Anda sebelumnya dan `--sso` untuk memaksa autentikasi SSO	`claude auth login --email user@example.com --sso`

Perintah	Deskripsi	Contoh
`claude auth logout`	Keluar dari akun Anthropic Anda	`claude auth logout`
`claude auth status`	Tampilkan status autentikasi sebagai JSON. Gunakan `--text` untuk output yang dapat dibaca manusia. Keluar dengan kode 0 jika masuk, 1 jika tidak	`claude auth status`
`claude agents`	Daftar semua subagents yang dikonfigurasi, dikelompokkan berdasarkan sumber	`claude agents`
`claude mcp`	Konfigurasi server Model Context Protocol (MCP)	Lihat dokumentasi Claude Code MCP.
`claude remote-control`	Mulai sesi Remote Control untuk mengontrol Claude Code dari Claude.ai atau aplikasi Claude sambil berjalan secara lokal. Lihat Remote Control untuk flag	`claude remote-control`

Flag CLI

Sesuaikan perilaku Claude Code dengan flag baris perintah ini:

Flag	Deskripsi	Contoh
`--add-dir`	Tambahkan direktori kerja tambahan untuk Claude akses (memvalidasi setiap jalur ada sebagai direktori)	`claude --add-dir ../apps ../lib`
`--agent`	Tentukan agen untuk sesi saat ini (menimpa pengaturan `agent`)	`claude --agent my-custom-agent`
`--agents`	Tentukan subagents kustom secara dinamis melalui JSON (lihat di bawah untuk format)	`claude --agents '{"reviewer": {"description":"Reviews code","prompt":"You are a code reviewer"}}'`

Flag	Deskripsi	Contoh
`--allow-dangerously-skip-permissions`	Aktifkan bypass izin sebagai opsi tanpa langsung mengaktifkannya. Memungkinkan komposisi dengan `--permission-mode` (gunakan dengan hati-hati)	`claude --permission-mode plan --allow-dangerously-skip-permissions`
`--allowedTools`	Alat yang dijalankan tanpa meminta izin. Lihat sintaks aturan izin untuk pencocokan pola. Untuk membatasi alat mana yang tersedia, gunakan `--tools` sebagai gantinya	`"Bash(git log *)" "Bash(git diff *)" "Read"`
`--append-system-prompt`	Tambahkan teks kustom ke akhir prompt sistem default	`claude --append-system-prompt "Always use TypeScript"`
`--append-system-prompt-file`	Muat teks prompt sistem tambahan dari file dan tambahkan ke prompt default	`claude --append-system-prompt-file ./extra-rules.txt`
`--betas`	Header beta untuk disertakan dalam permintaan API (hanya pengguna kunci API)	`claude --betas interleaved-thinking`
`--chrome`	Aktifkan integrasi browser Chrome untuk otomasi web dan pengujian	`claude --chrome`
`--continue`, `-c`	Muat percakapan terbaru di direktori saat ini	`claude --continue`
`--dangerously-skip-permissions`	Lewati semua prompt izin (gunakan dengan hati-hati)	`claude --dangerously-skip-permissions`
`--debug`	Aktifkan mode debug dengan penyaringan kategori opsional (misalnya, `"api,hooks"` atau `"!statsig,!file"`)	`claude --debug "api,mcp"`
`--disable-slash-commands`	Nonaktifkan semua skills dan perintah untuk sesi ini	`claude --disable-slash-commands`
`--disallowedTools`	Alat yang dihapus dari konteks model dan tidak dapat digunakan	`"Bash(git log *)" "Bash(git diff *)" "Edit"`

Flag	Deskripsi	Contoh
`--fallback-model`	Aktifkan fallback otomatis ke model yang ditentukan ketika model default kelebihan beban (mode cetak saja)	`claude -p --fallback-model sonnet "query"`
`--fork-session`	Saat melanjutkan, buat ID sesi baru alih-alih menggunakan kembali yang asli (gunakan dengan `--resume` atau `--continue`)	`claude --resume abc123 --fork-session`
`--from-pr`	Lanjutkan sesi yang ditautkan ke PR GitHub tertentu. Menerima nomor PR atau URL. Sesi secara otomatis ditautkan saat dibuat melalui `gh pr create`	`claude --from-pr 123`
`--ide`	Terhubung secara otomatis ke IDE saat startup jika tepat satu IDE valid tersedia	`claude --ide`
`--init`	Jalankan hook inisialisasi dan mulai mode interaktif	`claude --init`
`--init-only`	Jalankan hook inisialisasi dan keluar (tidak ada sesi interaktif)	`claude --init-only`
`--include-partial-messages`	Sertakan peristiwa streaming parsial dalam output (memerlukan `--print` dan `--output-format=stream-json`)	`claude -p --output-format stream-json --include-partial-messages "query"`
`--input-format`	Tentukan format input untuk mode cetak (opsi: `text`, `stream-json`)	`claude -p --output-format json --input-format stream-json`
`--json-schema`	Dapatkan output JSON yang divalidasi sesuai dengan JSON Schema setelah agen menyelesaikan alurnya (mode cetak saja, lihat structured outputs)	`claude -p --json-schema '{"type":"object","properties":{...}}' "query"`
`--maintenance`	Jalankan hook pemeliharaan dan keluar	`claude --maintenance`

Flag	Deskripsi	Contoh
`--max-budget-usd`	Jumlah dolar maksimum untuk dihabiskan pada panggilan API sebelum berhenti (mode cetak saja)	`claude -p --max-budget-usd 5.00 "query"`
`--max-turns`	Batasi jumlah putaran agentic (mode cetak saja). Keluar dengan kesalahan saat batas tercapai. Tidak ada batas secara default	`claude -p --max-turns 3 "query"`
`--mcp-config`	Muat server MCP dari file JSON atau string (dipisahkan spasi)	`claude --mcp-config ./mcp.json`
`--model`	Menetapkan model untuk sesi saat ini dengan alias untuk model terbaru (`sonnet` atau `opus`) atau nama lengkap model	`claude --model claude-sonnet-4-6`
`--no-chrome`	Nonaktifkan integrasi browser Chrome untuk sesi ini	`claude --no-chrome`
`--no-session-persistence`	Nonaktifkan persistensi sesi sehingga sesi tidak disimpan ke disk dan tidak dapat dilanjutkan (mode cetak saja)	`claude -p --no-session-persistence "query"`
`--output-format`	Tentukan format output untuk mode cetak (opsi: `text` , `json` , `stream-json`)	`claude -p "query" --output-format json`
`--permission-mode`	Mulai dalam mode izin yang ditentukan	`claude --permission-mode plan`
`--permission-prompt-tool`	Tentukan alat MCP untuk menangani prompt izin dalam mode non-interaktif	`claude -p --permission-prompt-tool mcp_auth_tool "query"`
`--plugin-dir`	Muat plugin dari direktori untuk sesi ini saja (dapat diulang)	`claude --plugin-dir ./my-plugins`
`--print` , `-p`	Cetak respons tanpa mode interaktif (lihat dokumentasi Agent SDK untuk detail penggunaan programatik)	`claude -p "query"`

Flag	Deskripsi	Contoh
`--remote`	Buat sesi web baru di claude.ai dengan deskripsi tugas yang disediakan	`claude --remote "Fix the login bug"`
`--resume`, `-r`	Lanjutkan sesi tertentu berdasarkan ID atau nama, atau tampilkan pemilih interaktif untuk memilih sesi	`claude --resume auth-refactor`
`--session-id`	Gunakan ID sesi tertentu untuk percakapan (harus UUID yang valid)	`claude --session-id "550e8400-e29b-41d4-a716-446655440000"`
`--setting-sources`	Daftar sumber pengaturan yang dipisahkan koma untuk dimuat (`user`, `project`, `local`)	`claude --setting-sources user,project`
`--settings`	Jalur ke file JSON pengaturan atau string JSON untuk memuat pengaturan tambahan dari	`claude --settings ./settings.json`
`--strict-mcp-config`	Hanya gunakan server MCP dari `--mcp-config`, abaikan semua konfigurasi MCP lainnya	`claude --strict-mcp-config --mcp-config ./mcp.json`
`--system-prompt`	Ganti seluruh prompt sistem dengan teks kustom	`claude --system-prompt "You are a Python expert"`
`--system-prompt-file`	Muat prompt sistem dari file, menggantiprompt default	`claude --system-prompt-file ./custom-prompt.txt`
`--teleport`	Lanjutkan sesi web di terminal lokal Anda	`claude --teleport`
`--teammate-mode`	Atur bagaimana tim agen rekan kerja ditampilkan: `auto` (default), `in-process`, atau `tmux`. Lihat atur tim agen	`claude --teammate-mode in-process`

Flag	Deskripsi	Contoh
`--tools`	Batasi alat bawaan mana yang dapat digunakan Claude. Gunakan `""` untuk menonaktifkan semua, `"default"` untuk semua, atau nama alat seperti `"Bash,Edit,Read"`	`claude --tools "Bash,Edit,Read"`
`--verbose`	Aktifkan logging verbose, menampilkan output putaran penuh	`claude --verbose`
`--version`, `-v`	Keluarkan nomor versi	`claude -v`
`--worktree`, `-w`	Mulai Claude dalam git worktree terisolasi di `<repo>/.claude/worktrees/<name>`. Jika tidak ada nama yang diberikan, satu akan dibuat secara otomatis	`claude -w feature-auth`

> **Tip:**
>
> Flag `--output-format json` sangat berguna untuk skrip dan otomasi, memungkinkan Anda mengurai respons Claude secara programatik.

Format flag agents

Flag `--agents` menerima objek JSON yang mendefinisikan satu atau lebih subagents kustom. Setiap subagent memerlukan nama unik (sebagai kunci) dan objek definisi dengan bidang berikut:

Bidang	Diperlukan	Deskripsi
`description`	Ya	Deskripsi bahasa alami tentang kapan subagent harus dipanggil
`prompt`	Ya	Prompt sistem yang memandu perilaku subagent

Bidang	Diperlukan	Deskripsi
`tools`	Tidak	Array alat spesifik yang dapat digunakan subagent, misalnya `["Read", "Edit", "Bash"]`. Jika dihilangkan, mewarisi semua alat. Mendukung sintaks `Agent(agent_type)`
`disallowedTools`	Tidak	Array nama alat untuk secara eksplisit menolak untuk subagent ini
`model`	Tidak	Alias model untuk digunakan: `sonnet`, `opus`, `haiku`, atau `inherit`. Jika dihilangkan, default ke `inherit`
`skills`	Tidak	Array nama skill untuk dimuat sebelumnya ke dalam konteks subagent
`mcpServers`	Tidak	Array server MCP untuk subagent ini. Setiap entri adalah string nama server atau objek `{name: config}`
`maxTurns`	Tidak	Jumlah maksimum putaran agentic sebelum subagent berhenti

Contoh:

```
claude --agents '{
  "code-reviewer": {
    "description": "Expert code reviewer. Use proactively after code changes.",
    "prompt": "You are a senior code reviewer. Focus on code quality, security,
and best practices.",
    "tools": ["Read", "Grep", "Glob", "Bash"],
    "model": "sonnet"
  },
  "debugger": {
    "description": "Debugging specialist for errors and test failures.",
    "prompt": "You are an expert debugger. Analyze errors, identify root causes,
and provide fixes."
  }
}'
```

Untuk detail lebih lanjut tentang membuat dan menggunakan subagents, lihat dokumentasi subagents.

Flag prompt sistem

Claude Code menyediakan empat flag untuk menyesuaikan prompt sistem. Keempat flag bekerja dalam mode interaktif dan non-interaktif.

Flag	Perilaku	Kasus penggunaan
`--system-prompt`	**Mengganti** seluruh prompt default	Kontrol lengkap atas perilaku dan instruksi Claude
`--system-prompt-file`	**Mengganti** dengan konten file	Muat prompt dari file untuk reproduksibilitas dan kontrol versi
`--append-system-prompt`	**Menambahkan** ke prompt default	Tambahkan instruksi spesifik sambil mempertahankan perilaku Claude Code default
`--append-system-prompt-file`	**Menambahkan** konten file ke prompt default	Muat instruksi tambahan dari file sambil mempertahankan default

Kapan menggunakan masing-masing:

- `--system-prompt` : gunakan ketika Anda memerlukan kontrol lengkap atas prompt sistem Claude. Ini menghapus semua instruksi Claude Code default, memberikan Anda kanvas kosong.

```
claude --system-prompt "You are a Python expert who only writes type-annotated code"
```

- `--system-prompt-file` : gunakan ketika Anda ingin memuat prompt kustom dari file, berguna untuk konsistensi tim atau template prompt yang dikontrol versi.

```
claude --system-prompt-file ./prompts/code-review.txt
```

- `--append-system-prompt` : gunakan ketika Anda ingin menambahkan instruksi spesifik sambil mempertahankan kemampuan default Claude Code. Ini adalah opsi paling aman untuk sebagian besar kasus penggunaan.

```
claude --append-system-prompt "Always use TypeScript and include JSDoc comments"
```

- `--append-system-prompt-file` : gunakan ketika Anda ingin menambahkan instruksi dari file sambil mempertahankan default Claude Code. Berguna untuk penambahan yang dikontrol versi.

```
claude --append-system-prompt-file ./prompts/style-rules.txt
```

`--system-prompt` dan `--system-prompt-file` saling eksklusif. Flag append dapat digunakan bersama dengan flag penggantian apa pun.

Untuk sebagian besar kasus penggunaan, `--append-system-prompt` atau `--append-system-prompt-file` direkomendasikan karena mereka mempertahankan kemampuan bawaan Claude Code sambil menambahkan persyaratan kustom Anda. Gunakan `--system-prompt` atau `--system-prompt-file` hanya ketika Anda memerlukan kontrol lengkap atas prompt sistem.

Lihat juga

- Ekstensi Chrome - Otomasi browser dan pengujian web

- Mode interaktif - Pintasan, mode input, dan fitur interaktif
- Panduan quickstart - Memulai dengan Claude Code
- Alur kerja umum - Alur kerja dan pola lanjutan
- Pengaturan - Opsi konfigurasi
- Dokumentasi Agent SDK - Penggunaan programatik dan integrasi

Sesuaikan pintasan keyboard

> *Sesuaikan pintasan keyboard di Claude Code dengan file konfigurasi keybindings.*

> **Note:**
>
> Pintasan keyboard yang dapat disesuaikan memerlukan Claude Code v2.1.18 atau lebih baru. Periksa versi Anda dengan `claude --version`.

Claude Code mendukung pintasan keyboard yang dapat disesuaikan. Jalankan `/keybindings` untuk membuat atau membuka file konfigurasi Anda di `~/.claude/keybindings.json`.

File konfigurasi

File konfigurasi keybindings adalah objek dengan array `bindings`. Setiap blok menentukan konteks dan peta dari keystroke ke tindakan.

> **Note:**
>
> Perubahan pada file keybindings secara otomatis terdeteksi dan diterapkan tanpa perlu memulai ulang Claude Code.

Field	Deskripsi
`$schema`	URL JSON Schema opsional untuk penyelesaian otomatis editor
`$docs`	URL dokumentasi opsional
`bindings`	Array blok binding berdasarkan konteks

Contoh ini mengikat `Ctrl+E` untuk membuka editor eksternal dalam konteks chat, dan membatalkan ikatan `Ctrl+U`:

```
{
  "$schema": "https://www.schemastore.org/claude-code-keybindings.json",
  "$docs": "https://code.claude.com/docs/id/keybindings",
  "bindings": [
    {
      "context": "Chat",
      "bindings": {
        "ctrl+e": "chat:externalEditor",
        "ctrl+u": null
      }
    }
  ]
}
```

Konteks

Setiap blok binding menentukan **konteks** di mana binding berlaku:

Konteks	Deskripsi
`Global`	Berlaku di mana saja dalam aplikasi
`Chat`	Area input chat utama
`Autocomplete`	Menu penyelesaian otomatis terbuka
`Settings`	Menu pengaturan (dismiss hanya dengan escape)
`Confirmation`	Dialog izin dan konfirmasi
`Tabs`	Komponen navigasi tab
`Help`	Menu bantuan terlihat
`Transcript`	Penampil transkrip
`HistorySearch`	Mode pencarian riwayat (Ctrl+R)
`Task`	Tugas latar belakang sedang berjalan
`ThemePicker`	Dialog pemilih tema
`Attachments`	Navigasi bilah gambar/lampiran

Konteks	Deskripsi
`Footer`	Navigasi indikator footer (tugas, tim, diff)
`MessageSelector`	Pemilihan pesan dialog rewind dan ringkasan
`DiffDialog`	Navigasi penampil diff
`ModelPicker`	Tingkat upaya pemilih model
`Select`	Komponen select/list generik
`Plugin`	Dialog plugin (jelajahi, temukan, kelola)

Tindakan yang tersedia

Tindakan mengikuti format `namespace:action`, seperti `chat:submit` untuk mengirim pesan atau `app:toggleTodos` untuk menampilkan daftar tugas. Setiap konteks memiliki tindakan spesifik yang tersedia.

Tindakan aplikasi

Tindakan yang tersedia dalam konteks `Global`:

Tindakan	Default	Deskripsi
`app:interrupt`	Ctrl+C	Batalkan operasi saat ini
`app:exit`	Ctrl+D	Keluar dari Claude Code
`app:toggleTodos`	Ctrl+T	Alihkan visibilitas daftar tugas
`app:toggleTranscript`	Ctrl+O	Alihkan transkrip verbose

Tindakan riwayat

Tindakan untuk menavigasi riwayat perintah:

Tindakan	Default	Deskripsi
`history:search`	Ctrl+R	Buka pencarian riwayat
`history:previous`	Up	Item riwayat sebelumnya
`history:next`	Down	Item riwayat berikutnya

Tindakan chat

Tindakan yang tersedia dalam konteks `Chat`:

Tindakan	Default	Deskripsi
`chat:cancel`	Escape	Batalkan input saat ini
`chat:cycleMode`	Shift+Tab*	Mode izin siklus
`chat:modelPicker`	Cmd+P / Meta+P	Buka pemilih model
`chat:thinkingToggle`	Cmd+T / Meta+T	Alihkan pemikiran yang diperluas
`chat:submit`	Enter	Kirim pesan
`chat:undo`	Ctrl+_	Batalkan tindakan terakhir
`chat:externalEditor`	Ctrl+G	Buka di editor eksternal
`chat:stash`	Ctrl+S	Simpan prompt saat ini
`chat:imagePaste`	Ctrl+V (Alt+V di Windows)	Tempel gambar

*Di Windows tanpa mode VT (Node <24.2.0/<22.17.0, Bun <1.2.23), default ke Meta+M.

Tindakan penyelesaian otomatis

Tindakan yang tersedia dalam konteks `Autocomplete`:

Tindakan	Default	Deskripsi
`autocomplete:accept`	Tab	Terima saran
`autocomplete:dismiss`	Escape	Tutup menu
`autocomplete:previous`	Up	Saran sebelumnya
`autocomplete:next`	Down	Saran berikutnya

Tindakan konfirmasi

Tindakan yang tersedia dalam konteks `Confirmation`:

Tindakan	Default	Deskripsi
`confirm:yes`	Y, Enter	Konfirmasi tindakan

Tindakan	Default	Deskripsi
`confirm:no`	N, Escape	Tolak tindakan
`confirm:previous`	Up	Opsi sebelumnya
`confirm:next`	Down	Opsi berikutnya
`confirm:nextField`	Tab	Bidang berikutnya
`confirm:previousField`	(unbound)	Bidang sebelumnya
`confirm:cycleMode`	Shift+Tab	Mode izin siklus
`confirm:toggleExplanation`	Ctrl+E	Alihkan penjelasan izin

Tindakan izin

Tindakan yang tersedia dalam konteks `Confirmation` untuk dialog izin:

Tindakan	Default	Deskripsi
`permission:toggleDebug`	Ctrl+D	Alihkan info debug izin

Tindakan transkrip

Tindakan yang tersedia dalam konteks `Transcript`:

Tindakan	Default	Deskripsi
`transcript:toggleShowAll`	Ctrl+E	Alihkan tampilkan semua konten
`transcript:exit`	Ctrl+C, Escape	Keluar dari tampilan transkrip

Tindakan pencarian riwayat

Tindakan yang tersedia dalam konteks `HistorySearch`:

Tindakan	Default	Deskripsi
`historySearch:next`	Ctrl+R	Kecocokan berikutnya
`historySearch:accept`	Escape, Tab	Terima pilihan
`historySearch:cancel`	Ctrl+C	Batalkan pencarian

Tindakan	Default	Deskripsi
`historySearch:execute`	Enter	Jalankan perintah yang dipilih

Tindakan tugas

Tindakan yang tersedia dalam konteks `Task`:

Tindakan	Default	Deskripsi
`task:background`	Ctrl+B	Tugas latar belakang saat ini

Tindakan tema

Tindakan yang tersedia dalam konteks `ThemePicker`:

Tindakan	Default	Deskripsi
`theme:toggleSyntaxHighlighting`	Ctrl+T	Alihkan penyorotan sintaks

Tindakan bantuan

Tindakan yang tersedia dalam konteks `Help`:

Tindakan	Default	Deskripsi
`help:dismiss`	Escape	Tutup menu bantuan

Tindakan tab

Tindakan yang tersedia dalam konteks `Tabs`:

Tindakan	Default	Deskripsi
`tabs:next`	Tab, Right	Tab berikutnya
`tabs:previous`	Shift+Tab, Left	Tab sebelumnya

Tindakan lampiran

Tindakan yang tersedia dalam konteks `Attachments`:

Tindakan	Default	Deskripsi
`attachments:next`	Right	Lampiran berikutnya
`attachments:previous`	Left	Lampiran sebelumnya
`attachments:remove`	Backspace, Delete	Hapus lampiran yang dipi-lih
`attachments:exit`	Down, Escape	Keluar dari bilah lampiran

Tindakan footer

Tindakan yang tersedia dalam konteks `Footer`:

Tindakan	Default	Deskripsi
`footer:next`	Right	Item footer berikutnya
`footer:previous`	Left	Item footer sebelumnya
`footer:openSelected`	Enter	Buka item footer yang dipilih
`footer:clearSelection`	Escape	Hapus pilihan footer

Tindakan pemilih pesan

Tindakan yang tersedia dalam konteks `MessageSelector`:

Tindakan	Default	Deskripsi
`messageSelector:up`	Up, K, Ctrl+P	Naik dalam daftar
`messageSelector:down`	Down, J, Ctrl+N	Turun dalam daftar
`messageSelector:top`	Ctrl+Up, Shift+Up, Meta+Up, Shift+K	Lompat ke atas
`messageSelector:bottom`	Ctrl+Down, Shift+Down, Meta+Down, Shift+J	Lompat ke bawah
`messageSelector:select`	Enter	Pilih pesan

Tindakan diff

Tindakan yang tersedia dalam konteks `DiffDialog`:

Tindakan	Default	Deskripsi
`diff:dismiss`	Escape	Tutup penampil diff
`diff:previousSource`	Left	Sumber diff sebelumnya
`diff:nextSource`	Right	Sumber diff berikutnya
`diff:previousFile`	Up	File sebelumnya dalam diff
`diff:nextFile`	Down	File berikutnya dalam diff
`diff:viewDetails`	Enter	Lihat detail diff
`diff:back`	(context-specific)	Kembali dalam penampil diff

Tindakan pemilih model

Tindakan yang tersedia dalam konteks `ModelPicker`:

Tindakan	Default	Deskripsi
`modelPicker:decreaseEffort`	Left	Kurangi tingkat upaya
`modelPicker:increaseEffort`	Right	Tingkatkan tingkat upaya

Tindakan pilih

Tindakan yang tersedia dalam konteks `Select`:

Tindakan	Default	Deskripsi
`select:next`	Down, J, Ctrl+N	Opsi berikutnya
`select:previous`	Up, K, Ctrl+P	Opsi sebelumnya
`select:accept`	Enter	Terima pilihan
`select:cancel`	Escape	Batalkan pilihan

Tindakan plugin

Tindakan yang tersedia dalam konteks `Plugin`:

Tindakan	Default	Deskripsi
`plugin:toggle`	Space	Alihkan pemilihan plugin
`plugin:install`	I	Instal plugin yang dipilih

Tindakan pengaturan

Tindakan yang tersedia dalam konteks `Settings`:

Tindakan	Default	Deskripsi
`settings:search`	/	Masuk mode pencarian
`settings:retry`	R	Coba muat ulang data penggunaan (saat terjadi kesalahan)

Sintaks keystroke

Pengubah

Gunakan tombol pengubah dengan pemisah `+`:

- `ctrl` atau `control` - Tombol Control
- `alt`, `opt`, atau `option` - Tombol Alt/Option
- `shift` - Tombol Shift
- `meta`, `cmd`, atau `command` - Tombol Meta/Command

Sebagai contoh:

```
ctrl+k          Tombol tunggal dengan pengubah
shift+tab       Shift + Tab
meta+p          Command/Meta + P
ctrl+shift+c    Pengubah ganda
```

Huruf besar

Huruf besar yang berdiri sendiri menyiratkan Shift. Sebagai contoh, `K` setara dengan `shift+k`. Ini berguna untuk binding gaya vim di mana kunci huruf besar dan kecil memiliki arti berbeda.

Huruf besar dengan pengubah (misalnya, `ctrl+K`) diperlakukan sebagai gaya dan **tidak** menyiratkan Shift — `ctrl+K` sama dengan `ctrl+k`.

Chord

Chord adalah urutan keystroke yang dipisahkan oleh spasi:

```
ctrl+k ctrl+s   Tekan Ctrl+K, lepaskan, lalu Ctrl+S
```

Tombol khusus

- `escape` atau `esc` - Tombol Escape
- `enter` atau `return` - Tombol Enter
- `tab` - Tombol Tab
- `space` - Bilah spasi
- `up`, `down`, `left`, `right` - Tombol panah
- `backspace`, `delete` - Tombol hapus

Batalkan pintasan default

Atur tindakan ke `null` untuk membatalkan ikatan pintasan default:

```
{
  "bindings": [
    {
      "context": "Chat",
      "bindings": {
        "ctrl+s": null
      }
    }
  ]
}
```

Pintasan yang dicadangkan

Pintasan ini tidak dapat diikat ulang:

Pintasan	Alasan
Ctrl+C	Interrupt/cancel yang dikodekan keras

Pintasan	Alasan
Ctrl+D	Exit yang dikodekan keras

Konflik terminal

Beberapa pintasan mungkin bertentangan dengan multiplexer terminal:

Pintasan	Konflik
Ctrl+B	Awalan tmux (tekan dua kali untuk mengirim)
Ctrl+A	Awalan GNU screen
Ctrl+Z	Suspend proses Unix (SIGTSTP)

Interaksi mode vim

Ketika mode vim diaktifkan (`/vim`), keybindings dan mode vim beroperasi secara independen:

- **Mode vim** menangani input pada tingkat input teks (gerakan kursor, mode, motions)
- **Keybindings** menangani tindakan pada tingkat komponen (alihkan todos, kirim, dll.)
- Tombol Escape dalam mode vim beralih dari INSERT ke mode NORMAL; itu tidak memicu `chat:cancel`
- Sebagian besar pintasan Ctrl+key melewati mode vim ke sistem keybinding
- Dalam mode NORMAL vim, `?` menampilkan menu bantuan (perilaku vim)

Validasi

Claude Code memvalidasi keybindings Anda dan menampilkan peringatan untuk:

- Parse errors (JSON atau struktur tidak valid)
- Nama konteks tidak valid
- Konflik pintasan yang dicadangkan
- Konflik multiplexer terminal
- Binding duplikat dalam konteks yang sama

Jalankan `/doctor` untuk melihat peringatan keybinding apa pun.

Part 4: Configuration

Konfigurasi izin

> *Kontrol apa yang dapat diakses Claude Code dan lakukan dengan aturan izin terperinci, mode, dan kebijakan terkelola.*

Claude Code mendukung izin terperinci sehingga Anda dapat menentukan dengan tepat apa yang diizinkan dilakukan oleh agen dan apa yang tidak. Pengaturan izin dapat diperiksa ke dalam kontrol versi dan didistribusikan ke semua pengembang di organisasi Anda, serta disesuaikan oleh pengembang individual.

Sistem izin

Claude Code menggunakan sistem izin berjenjang untuk menyeimbangkan kekuatan dan keamanan:

Jenis alat	Contoh	Persetujuan diperlukan	Perilaku "Ya, jangan tanya lagi"
Hanya baca	Pembacaan file, Grep	Tidak	T/A
Perintah Bash	Eksekusi shell	Ya	Secara permanen per direktori proyek dan perintah
Modifikasi file	Edit/tulis file	Ya	Hingga akhir sesi

Kelola izin

Anda dapat melihat dan mengelola izin alat Claude Code dengan `/permissions`. UI ini mencantumkan semua aturan izin dan file settings.json tempat mereka bersumber.

- Aturan **Allow** memungkinkan Claude Code menggunakan alat yang ditentukan tanpa persetujuan manual.
- Aturan **Ask** meminta konfirmasi setiap kali Claude Code mencoba menggunakan alat yang ditentukan.
- Aturan **Deny** mencegah Claude Code menggunakan alat yang ditentukan.

Aturan dievaluasi secara berurutan: **deny -> ask -> allow**. Aturan pertama yang cocok menang, jadi aturan deny selalu memiliki prioritas.

Mode izin

Claude Code mendukung beberapa mode izin yang mengontrol bagaimana alat disetujui. Atur `defaultMode` dalam file pengaturan Anda:

Mode	Deskripsi
`default`	Perilaku standar: meminta izin pada penggunaan pertama setiap alat
`acceptEdits`	Secara otomatis menerima izin edit file untuk sesi
`plan`	Plan Mode: Claude dapat menganalisis tetapi tidak memodifikasi file atau menjalankan perintah
`dontAsk`	Secara otomatis menolak alat kecuali pra-disetujui melalui `/permissions` atau aturan `permissions.allow`
`bypassPermissions`	Melewati semua prompt izin (memerlukan lingkungan yang aman, lihat peringatan di bawah)

> **Warning:**
>
> Mode `bypassPermissions` menonaktifkan semua pemeriksaan izin. Hanya gunakan ini di lingkungan terisolasi seperti kontainer atau VM tempat Claude Code tidak dapat menyebabkan kerusakan. Administrator dapat mencegah mode ini dengan mengatur `disableBypassPermissionsMode` ke `"disable"` dalam pengaturan terkelola.

Sintaks aturan izin

Aturan izin mengikuti format `Tool` atau `Tool(specifier)`.

Cocokkan semua penggunaan alat

Untuk mencocokkan semua penggunaan alat, gunakan hanya nama alat tanpa tanda kurung:

Aturan	Efek
`Bash`	Mencocokkan semua perintah Bash
`WebFetch`	Mencocokkan semua permintaan pengambilan web

Aturan	Efek
`Read`	Mencocokkan semua pembacaan file

`Bash(*)` setara dengan `Bash` dan mencocokkan semua perintah Bash.

Gunakan specifier untuk kontrol terperinci

Tambahkan specifier dalam tanda kurung untuk mencocokkan penggunaan alat tertentu:

Aturan	Efek
`Bash(npm run build)`	Mencocokkan perintah yang tepat `npm run build`
`Read(./.env)`	Mencocokkan pembacaan file `.env` di direktori saat ini
`WebFetch(domain:example.com)`	Mencocokkan permintaan pengambilan ke example.com

Pola wildcard

Aturan Bash mendukung pola glob dengan `*`. Wildcard dapat muncul di posisi mana pun dalam perintah. Konfigurasi ini memungkinkan perintah npm dan git commit sambil memblokir git push:

```
{
  "permissions": {
    "allow": [
      "Bash(npm run *)",
      "Bash(git commit *)",
      "Bash(git * main)",
      "Bash(* --version)",
      "Bash(* --help *)"
    ],
    "deny": [
      "Bash(git push *)"
    ]
  }
}
```

Spasi sebelum `*` penting: `Bash(ls *)` mencocokkan `ls -la` tetapi bukan `lsof`, sementara `Bash(ls*)` mencocokkan keduanya. Sintaks akhiran `:*` warisan setara dengan `*` tetapi sudah usang.

Aturan izin khusus alat

Bash

Aturan izin Bash mendukung pencocokan wildcard dengan `*`. Wildcard dapat muncul di posisi mana pun dalam perintah, termasuk di awal, tengah, atau akhir:

- `Bash(npm run build)` mencocokkan perintah Bash yang tepat `npm run build`
- `Bash(npm run test *)` mencocokkan perintah Bash yang dimulai dengan `npm run test`
- `Bash(npm *)` mencocokkan perintah apa pun yang dimulai dengan `npm`
- `Bash(* install)` mencocokkan perintah apa pun yang berakhir dengan `install`
- `Bash(git * main)` mencocokkan perintah seperti `git checkout main`, `git merge main`

Ketika `*` muncul di akhir dengan spasi sebelumnya (seperti `Bash(ls *)`), ini memberlakukan batas kata, memerlukan awalan diikuti oleh spasi atau akhir string. Misalnya, `Bash(ls *)` mencocokkan `ls -la` tetapi bukan `lsof`. Sebaliknya, `Bash(ls*)` tanpa spasi mencocokkan `ls -la` dan `lsof` karena tidak ada batasan batas kata.

> **Tip:**
>
> Claude Code menyadari operator shell (seperti `&&`) jadi aturan pencocokan awalan seperti `Bash(safe-cmd *)` tidak akan memberinya izin untuk menjalankan perintah `safe-cmd && other-cmd`.

> **Warning:**
>
> Pola izin Bash yang mencoba membatasi argumen perintah rapuh. Misalnya, `Bash(curl http://github.com/ *)` dimaksudkan untuk membatasi curl ke URL GitHub, tetapi tidak akan mencocokkan variasi seperti:
>
> - Opsi sebelum URL: `curl -X GET http://github.com/...`
> - Protokol berbeda: `curl https://github.com/...`
> - Pengalihan: `curl -L http://bit.ly/xyz` (pengalihan ke github)
> - Variabel: `URL=http://github.com && curl $URL`
> - Spasi ekstra: `curl http://github.com`
>
> Untuk penyaringan URL yang lebih andal, pertimbangkan:
>
> - **Batasi alat jaringan Bash**: gunakan aturan deny untuk memblokir `curl`, `wget`, dan perintah serupa, kemudian gunakan alat WebFetch dengan izin `WebFetch(domain:github.com)` untuk domain yang diizinkan

- **Gunakan hook PreToolUse**: implementasikan hook yang memvalidasi URL dalam perintah Bash dan memblokir domain yang tidak diizinkan
- Menginstruksikan Claude Code tentang pola curl yang diizinkan Anda melalui CLAUDE.md

Perhatikan bahwa menggunakan WebFetch saja tidak mencegah akses jaringan. Jika Bash diizinkan, Claude masih dapat menggunakan `curl`, `wget`, atau alat lain untuk menjangkau URL apa pun.

Read dan Edit

Aturan `Edit` berlaku untuk semua alat bawaan yang mengedit file. Claude membuat upaya terbaik untuk menerapkan aturan `Read` ke semua alat bawaan yang membaca file seperti Grep dan Glob.

Aturan Read dan Edit keduanya mengikuti spesifikasi gitignore dengan empat jenis pola yang berbeda:

Pola	Arti	Contoh	Cocok
`//path`	Jalur **absolut** dari akar sistem file	`Read(//Users/alice/secrets/**)`	`/Users/alice/secrets/**`
`~/path`	Jalur dari direktori **home**	`Read(~/Documents/*.pdf)`	`/Users/alice/Documents/*.pdf`
`/path`	Jalur **relatif terhadap akar proyek**	`Edit(/src/**/*.ts)`	`<project root>/src/**/*.ts`
`path` atau `./path`	Jalur **relatif terhadap direktori saat ini**	`Read(*.env)`	`<cwd>/*.env`

Warning:

Pola seperti `/Users/alice/file` BUKAN jalur absolut. Ini relatif terhadap akar proyek. Gunakan `//Users/alice/file` untuk jalur absolut.

Contoh:

- `Edit(/docs/**)`: edit di `<project>/docs/` (BUKAN `/docs/` dan BUKAN `<project>/.claude/docs/`)
- `Read(~/.zshrc)`: membaca `.zshrc` direktori home Anda
- `Edit(//tmp/scratch.txt)`: edit jalur absolut `/tmp/scratch.txt`

- `Read(src/**)` : membaca dari `<current-directory>/src/`

> **Note:**
>
> Dalam pola gitignore, `*` mencocokkan file dalam satu direktori sementara `**` mencocokkan secara rekursif di seluruh direktori. Untuk memungkinkan semua akses file, gunakan hanya nama alat tanpa tanda kurung: `Read` , `Edit` , atau `Write` .

WebFetch

- `WebFetch(domain:example.com)` mencocokkan permintaan pengambilan ke example.com

MCP

- `mcp__puppeteer` mencocokkan alat apa pun yang disediakan oleh server `puppeteer` (nama dikonfigurasi di Claude Code)
- `mcp__puppeteer__*` sintaks wildcard yang juga mencocokkan semua alat dari server `puppeteer`
- `mcp__puppeteer__puppeteer_navigate` mencocokkan alat `puppeteer_navigate` yang disediakan oleh server `puppeteer`

Agent (subagents)

Gunakan aturan `Agent(AgentName)` untuk mengontrol subagents mana yang dapat digunakan Claude:

- `Agent(Explore)` mencocokkan subagent Explore
- `Agent(Plan)` mencocokkan subagent Plan
- `Agent(my-custom-agent)` mencocokkan subagent kustom bernama `my-custom-agent`

Tambahkan aturan ini ke array `deny` dalam pengaturan Anda atau gunakan flag CLI `--disallowedTools` untuk menonaktifkan agen tertentu. Untuk menonaktifkan agen Explore:

```
{
  "permissions": {
    "deny": ["Agent(Explore)"]
  }
}
```

Perluas izin dengan hook

Hook Claude Code menyediakan cara untuk mendaftarkan perintah shell kustom guna melakukan evaluasi izin saat runtime. Ketika Claude Code membuat panggilan alat, hook PreToolUse berjalan sebelum sistem izin, dan output hook dapat menentukan apakah akan menyetujui atau menolak panggilan alat sebagai pengganti sistem izin.

Direktori kerja

Secara default, Claude memiliki akses ke file di direktori tempat diluncurkan. Anda dapat memperluas akses ini:

- **Saat startup**: gunakan argumen CLI `--add-dir <path>`
- **Selama sesi**: gunakan perintah `/add-dir`
- **Konfigurasi persisten**: tambahkan ke `additionalDirectories` dalam file pengaturan

File di direktori tambahan mengikuti aturan izin yang sama dengan direktori kerja asli: mereka menjadi dapat dibaca tanpa prompt, dan izin edit file mengikuti mode izin saat ini.

Bagaimana izin berinteraksi dengan sandboxing

Izin dan sandboxing adalah lapisan keamanan pelengkap:

- **Izin** mengontrol alat mana yang dapat digunakan Claude Code dan file atau domain mana yang dapat diaksesnya. Mereka berlaku untuk semua alat (Bash, Read, Edit, WebFetch, MCP, dan lainnya).
- **Sandboxing** menyediakan penegakan tingkat OS yang membatasi akses sistem file dan jaringan alat Bash. Ini hanya berlaku untuk perintah Bash dan proses anak mereka.

Gunakan keduanya untuk pertahanan berlapis:

- Aturan deny izin memblokir Claude dari bahkan mencoba mengakses sumber daya terbatas
- Pembatasan sandbox mencegah perintah Bash menjangkau sumber daya di luar batas yang ditentukan, bahkan jika injeksi prompt melewati pengambilan keputusan Claude
- Pembatasan sistem file di sandbox menggunakan aturan deny Read dan Edit, bukan konfigurasi sandbox terpisah
- Pembatasan jaringan menggabungkan aturan izin WebFetch dengan daftar `allowedDomains` sandbox

Pengaturan terkelola

Untuk organisasi yang memerlukan kontrol terpusat atas konfigurasi Claude Code, administrator dapat menerapkan pengaturan terkelola yang tidak dapat ditimpa oleh pengaturan pengguna atau proyek. Pengaturan kebijakan ini mengikuti format yang sama dengan file pengaturan reguler dan dapat dikirimkan melalui kebijakan MDM/tingkat OS, file pengaturan terkelola, atau pengaturan yang dikelola server. Lihat file pengaturan untuk mekanisme pengiriman dan lokasi file.

Pengaturan khusus terkelola

Beberapa pengaturan hanya efektif dalam pengaturan terkelola:

Pengaturan	Deskripsi
`disableBypassPermissionsMode`	Atur ke `"disable"` untuk mencegah mode `bypassPermissions` dan flag `--dangerously-skip-permissions`
`allowManagedPermissionRulesOnly`	Ketika `true`, mencegah pengaturan pengguna dan proyek dari mendefinisikan aturan izin `allow`, `ask`, atau `deny`. Hanya aturan dalam pengaturan terkelola yang berlaku
`allowManagedHooksOnly`	Ketika `true`, mencegah pemuatan hook pengguna, proyek, dan plugin. Hanya hook terkelola dan hook SDK yang diizinkan
`allowManagedMcpServersOnly`	Ketika `true`, hanya `allowedMcpServers` dari pengaturan terkelola yang dihormati. `deniedMcpServers` masih digabung dari semua sumber. Lihat Konfigurasi MCP terkelola
`blockedMarketplaces`	Daftar blokir sumber marketplace. Sumber yang diblokir diperiksa sebelum mengunduh, jadi mereka tidak pernah menyentuh sistem file. Lihat pembatasan marketplace terkelola

Pengaturan	Deskripsi
`sandbox.network.allowManagedDomainsOnly`	Ketika `true`, hanya `allowedDomains` dan aturan allow `WebFetch(domain:...)` dari pengaturan terkelola yang dihormati. Domain yang tidak diizinkan diblokir secara otomatis tanpa meminta pengguna. Domain yang ditolak masih digabung dari semua sumber
`strictKnownMarketplaces`	Mengontrol marketplace plugin mana yang dapat ditambahkan pengguna. Lihat pembatasan marketplace terkelola
`allow_remote_sessions`	Ketika `true`, memungkinkan pengguna memulai Remote Control dan sesi web. Default ke `true`. Atur ke `false` untuk mencegah akses sesi jarak jauh

Prioritas pengaturan

Aturan izin mengikuti prioritas pengaturan yang sama dengan semua pengaturan Claude Code lainnya:

1. **Pengaturan terkelola**: tidak dapat ditimpa oleh tingkat lain apa pun, termasuk argumen baris perintah
2. **Argumen baris perintah**: penggantian sesi sementara
3. **Pengaturan proyek lokal** (`.claude/settings.local.json`)
4. **Pengaturan proyek bersama** (`.claude/settings.json`)
5. **Pengaturan pengguna** (`~/.claude/settings.json`)

Jika alat ditolak di tingkat mana pun, tidak ada tingkat lain yang dapat mengizinkannya. Misalnya, deny pengaturan terkelola tidak dapat ditimpa oleh `--allowedTools`, dan `--disallowedTools` dapat menambahkan pembatasan di luar apa yang ditentukan pengaturan terkelola.

Jika izin diizinkan dalam pengaturan pengguna tetapi ditolak dalam pengaturan proyek, pengaturan proyek memiliki prioritas dan izin diblokir.

Contoh konfigurasi

Repositori ini mencakup konfigurasi pengaturan pemula untuk skenario penerapan umum. Gunakan ini sebagai titik awal dan sesuaikan dengan kebutuhan Anda.

Lihat juga

- Pengaturan: referensi konfigurasi lengkap termasuk tabel pengaturan izin
- Sandboxing: isolasi sistem file dan jaringan tingkat OS untuk perintah Bash
- Autentikasi: atur akses pengguna ke Claude Code
- Keamanan: perlindungan keamanan dan praktik terbaik
- Hook: otomatisasi alur kerja dan perluas evaluasi izin

Bagaimana Claude mengingat proyek Anda

> *Berikan Claude instruksi persisten dengan file CLAUDE.md, dan biarkan Claude mengumpulkan pembelajaran secara otomatis dengan auto memory.*

Setiap sesi Claude Code dimulai dengan context window yang segar. Dua mekanisme membawa pengetahuan lintas sesi:

- **File CLAUDE.md**: instruksi yang Anda tulis untuk memberikan Claude konteks persisten
- **Auto memory**: catatan yang Claude tulis sendiri berdasarkan koreksi dan preferensi Anda

Halaman ini mencakup cara untuk:

- Menulis dan mengorganisir file CLAUDE.md
- Membatasi aturan ke tipe file tertentu dengan `.claude/rules/`
- Mengonfigurasi auto memory agar Claude membuat catatan secara otomatis
- Memecahkan masalah ketika instruksi tidak diikuti

CLAUDE.md vs auto memory

Claude Code memiliki dua sistem memori yang saling melengkapi. Keduanya dimuat di awal setiap percakapan. Claude memperlakukan mereka sebagai konteks, bukan konfigurasi yang ditegakkan. Semakin spesifik dan ringkas instruksi Anda, semakin konsisten Claude mengikutinya.

	File CLAUDE.md	Auto memory
Siapa yang menulisnya	Anda	Claude
Apa yang dikandungnya	Instruksi dan aturan	Pembelajaran dan pola
Cakupan	Proyek, pengguna, atau organisasi	Per working tree
Dimuat ke dalam	Setiap sesi	Setiap sesi (200 baris pertama)

	File CLAUDE.md	Auto memory
Gunakan untuk	Standar pengkodean, alur kerja, arsitektur proyek	Perintah build, wawasan debugging, preferensi yang Claude temukan

Gunakan file CLAUDE.md ketika Anda ingin memandu perilaku Claude. Auto memory memungkinkan Claude belajar dari koreksi Anda tanpa usaha manual.

Subagents juga dapat mempertahankan auto memory mereka sendiri. Lihat konfigurasi subagent untuk detail.

File CLAUDE.md

File CLAUDE.md adalah file markdown yang memberikan Claude instruksi persisten untuk proyek, alur kerja pribadi Anda, atau seluruh organisasi Anda. Anda menulis file ini dalam teks biasa; Claude membacanya di awal setiap sesi.

Pilih di mana menempatkan file CLAUDE.md

File CLAUDE.md dapat berada di beberapa lokasi, masing-masing dengan cakupan yang berbeda. Lokasi yang lebih spesifik memiliki prioritas lebih tinggi daripada yang lebih luas.

Cakupan	Lokasi	Tujuan	Contoh kasus penggunaan	Dibagikan dengan
Kebijakan terkelola	• macOS: `/Library/Application Support/ClaudeCode/CLAUDE.md` • Linux dan WSL: `/etc/claude-code/CLAUDE.md` • Windows: `C:\Program Files\ClaudeCode\CLAUDE.md`	Instruksi di seluruh organisasi yang dikelola oleh IT/DevOps	Standar pengkodean perusahaan, kebijakan keamanan, persyaratan kepatuhan	Semua pengguna dalam organisasi
Instruksi proyek	`./CLAUDE.md` atau `./.claude/CLAUDE.md`	Instruksi bersama tim untuk proyek	Arsitektur proyek, standar pengkodean, alur kerja umum	Anggota tim melalui kontrol sumber

Cakupan	Lokasi	Tujuan	Contoh kasus penggunaan	Dibagikan dengan
Instruksi pengguna	`~/.claude/CLAUDE.md`	Preferensi pribadi untuk semua proyek	Preferensi gaya kode, pintasan alat pribadi	Hanya Anda (semua proyek)

File CLAUDE.md dalam hierarki direktori di atas direktori kerja dimuat sepenuhnya saat peluncuran. File CLAUDE.md di subdirektori dimuat sesuai permintaan ketika Claude membaca file di direktori tersebut. Lihat Bagaimana file CLAUDE.md dimuat untuk urutan resolusi lengkap.

Untuk proyek besar, Anda dapat memecah instruksi menjadi file khusus topik menggunakan aturan proyek. Aturan memungkinkan Anda membatasi instruksi ke tipe file atau subdirektori tertentu.

Siapkan CLAUDE.md proyek

CLAUDE.md proyek dapat disimpan di `./CLAUDE.md` atau `./.claude/CLAUDE.md`. Buat file ini dan tambahkan instruksi yang berlaku untuk siapa pun yang bekerja pada proyek: perintah build dan test, standar pengkodean, keputusan arsitektur, konvensi penamaan, dan alur kerja umum. Instruksi ini dibagikan dengan tim Anda melalui kontrol versi, jadi fokus pada standar tingkat proyek daripada preferensi pribadi.

> **Tip:**
>
> Jalankan `/init` untuk menghasilkan CLAUDE.md awal secara otomatis. Claude menganalisis basis kode Anda dan membuat file dengan perintah build, instruksi test, dan konvensi proyek yang ditemukannya. Jika CLAUDE.md sudah ada, `/init` menyarankan perbaikan daripada menimpanya. Perbaiki dari sana dengan instruksi yang Claude tidak akan temukan sendiri.

Tulis instruksi yang efektif

File CLAUDE.md dimuat ke dalam context window di awal setiap sesi, mengonsumsi token bersama percakapan Anda. Karena mereka adalah konteks daripada konfigurasi yang ditegakkan, cara Anda menulis instruksi mempengaruhi seberapa andal Claude mengikutinya. Instruksi yang spesifik, ringkas, dan terstruktur dengan baik bekerja paling baik.

Ukuran: targetkan di bawah 200 baris per file CLAUDE.md. File yang lebih panjang mengonsumsi lebih banyak konteks dan mengurangi kepatuhan. Jika instruksi Anda berkembang besar, pisahkan menggunakan impor atau file `.claude/rules/`.

Struktur: gunakan header markdown dan bullet untuk mengelompokkan instruksi terkait. Claude memindai struktur dengan cara yang sama seperti pembaca: bagian yang terorganisir lebih mudah diikuti daripada paragraf padat.

Spesifisitas: tulis instruksi yang cukup konkret untuk diverifikasi. Misalnya:

- "Gunakan indentasi 2 spasi" daripada "Format kode dengan baik"
- "Jalankan `npm test` sebelum commit" daripada "Uji perubahan Anda"
- "Handler API berada di `src/api/handlers/` " daripada "Jaga file tetap terorganisir"

Konsistensi: jika dua aturan saling bertentangan, Claude mungkin memilih satu secara sembarangan. Tinjau file CLAUDE.md Anda, file CLAUDE.md bersarang di subdirektori, dan file `.claude/rules/` secara berkala untuk menghapus instruksi yang ketinggalan zaman atau bertentangan. Dalam monorepo, gunakan `claudeMdExcludes` untuk melewati file CLAUDE.md dari tim lain yang tidak relevan dengan pekerjaan Anda.

Impor file tambahan

File CLAUDE.md dapat mengimpor file tambahan menggunakan sintaks `@path/to/import` . File yang diimpor diperluas dan dimuat ke dalam konteks saat peluncuran bersama CLAUDE.md yang mereferensikannya.

Jalur relatif dan absolut diizinkan. Jalur relatif diselesaikan relatif terhadap file yang berisi impor, bukan direktori kerja. File yang diimpor dapat secara rekursif mengimpor file lain, dengan kedalaman maksimal lima hop.

Untuk menarik README, package.json, dan panduan alur kerja, referensikan mereka dengan sintaks `@` di mana saja di CLAUDE.md Anda:

```
Lihat @README untuk gambaran umum proyek dan @package.json untuk perintah npm yang
tersedia untuk proyek ini.

## Instruksi Tambahan
- alur kerja git @docs/git-instructions.md
```

Untuk preferensi pribadi yang tidak ingin Anda periksa, impor file dari direktori home Anda. Impor masuk ke CLAUDE.md bersama, tetapi file yang ditunjuknya tetap di mesin Anda:

```
## Preferensi Individu
- @~/.claude/my-project-instructions.md
```

> **Warning:**
>
> Pertama kali Claude Code menemukan impor eksternal dalam proyek, itu menampilkan dialog persetujuan yang mencantumkan file. Jika Anda menolak, impor tetap dinonaktifkan dan dialog tidak muncul lagi.

Untuk pendekatan yang lebih terstruktur untuk mengorganisir instruksi, lihat `.claude/rules/`.

Bagaimana file CLAUDE.md dimuat

Claude Code membaca file CLAUDE.md dengan berjalan naik pohon direktori dari direktori kerja saat ini, memeriksa setiap direktori di sepanjang jalan. Ini berarti jika Anda menjalankan Claude Code di `foo/bar/`, itu memuat instruksi dari `foo/bar/CLAUDE.md` dan `foo/CLAUDE.md`.

Claude juga menemukan file CLAUDE.md di subdirektori di bawah direktori kerja saat ini. Alih-alih memuatnya saat peluncuran, mereka disertakan ketika Claude membaca file di subdirektori tersebut.

Jika Anda bekerja di monorepo besar di mana file CLAUDE.md tim lain diambil, gunakan `claudeMdExcludes` untuk melewatinya.

Muat dari direktori tambahan

Flag `--add-dir` memberikan Claude akses ke direktori tambahan di luar direktori kerja utama Anda. Secara default, file CLAUDE.md dari direktori ini tidak dimuat.

Untuk juga memuat file CLAUDE.md dari direktori tambahan, termasuk `CLAUDE.md`, `.claude/CLAUDE.md`, dan `.claude/rules/*.md`, atur variabel lingkungan `CLAUDE_CODE_ADDITIONAL_DIRECTORIES_CLAUDE_MD`:

```
CLAUDE_CODE_ADDITIONAL_DIRECTORIES_CLAUDE_MD=1 claude --add-dir ../shared-config
```

Organisir aturan dengan `.claude/rules/`

Untuk proyek yang lebih besar, Anda dapat mengorganisir instruksi menjadi beberapa file menggunakan direktori `.claude/rules/`. Ini membuat instruksi modular dan lebih mudah bagi tim untuk dipertahankan. Aturan juga dapat dibatasi ke jalur file tertentu, sehingga mereka hanya dimuat ke dalam konteks ketika Claude bekerja dengan file yang cocok, mengurangi kebisingan dan menghemat ruang konteks.

> **Note:**
>
> Aturan dimuat ke dalam konteks setiap sesi atau ketika file yang cocok dibuka. Untuk instruksi khusus tugas yang tidak perlu berada dalam konteks sepanjang waktu, gunakan skills sebagai gantinya, yang hanya dimuat ketika Anda menginvokasinya atau ketika Claude menentukan mereka relevan dengan prompt Anda.

Siapkan aturan

Tempatkan file markdown di direktori `.claude/rules/` proyek Anda. Setiap file harus mencakup satu topik, dengan nama file deskriptif seperti `testing.md` atau `api-design.md`. Semua file `.md` ditemukan secara rekursif, jadi Anda dapat mengorganisir aturan ke dalam subdirektori seperti `frontend/` atau `backend/`:

```
your-project/
├── .claude/
│   ├── CLAUDE.md           # Instruksi proyek utama
│   └── rules/
│       ├── code-style.md   # Pedoman gaya kode
│       ├── testing.md      # Konvensi pengujian
│       └── security.md     # Persyaratan keamanan
```

Aturan tanpa frontmatter `paths` dimuat saat peluncuran dengan prioritas yang sama seperti `.claude/CLAUDE.md`.

Aturan khusus jalur

Aturan dapat dibatasi ke file tertentu menggunakan frontmatter YAML dengan bidang `paths`. Aturan bersyarat ini hanya berlaku ketika Claude bekerja dengan file yang cocok dengan pola yang ditentukan.

```
---
paths:
  - "src/api/**/*.ts"
---

## Aturan Pengembangan API

- Semua endpoint API harus menyertakan validasi input
- Gunakan format respons kesalahan standar
- Sertakan komentar dokumentasi OpenAPI
```

Aturan tanpa bidang `paths` dimuat tanpa syarat dan berlaku untuk semua file. Aturan bersyarat jalur dipicu ketika Claude membaca file yang cocok dengan pola, bukan pada setiap penggunaan alat.

Gunakan pola glob di bidang `paths` untuk mencocokkan file berdasarkan ekstensi, direktori, atau kombinasi apa pun:

Pola	Cocok dengan
`**/*.ts`	Semua file TypeScript di direktori apa pun
`src/**/*`	Semua file di bawah direktori `src/`
`*.md`	File Markdown di root proyek
`src/components/*.tsx`	Komponen React di direktori tertentu

Anda dapat menentukan beberapa pola dan menggunakan ekspansi kurung untuk mencocokkan beberapa ekstensi dalam satu pola:

```
---
paths:
  - "src/**/*.{ts,tsx}"
  - "lib/**/*.ts"
  - "tests/**/*.test.ts"
---
```

Bagikan aturan lintas proyek dengan symlink

Direktori `.claude/rules/` mendukung symlink, jadi Anda dapat mempertahankan set aturan bersama dan menautkannya ke beberapa proyek. Symlink diselesaikan dan dimuat secara normal, dan symlink melingkar terdeteksi dan ditangani dengan anggun.

Contoh ini menautkan direktori bersama dan file individual:

```
ln -s ~/shared-claude-rules .claude/rules/shared
ln -s ~/company-standards/security.md .claude/rules/security.md
```

Aturan tingkat pengguna

Aturan pribadi di `~/.claude/rules/` berlaku untuk setiap proyek di mesin Anda. Gunakan untuk preferensi yang bukan khusus proyek:

```
~/.claude/rules/
├── preferences.md    # Preferensi pengkodean pribadi Anda
└── workflows.md      # Alur kerja pilihan Anda
```

Aturan tingkat pengguna dimuat sebelum aturan proyek, memberikan aturan proyek prioritas lebih tinggi.

Kelola CLAUDE.md untuk tim besar

Untuk organisasi yang menerapkan Claude Code di seluruh tim, Anda dapat memusatkan instruksi dan mengontrol file CLAUDE.md mana yang dimuat.

Terapkan CLAUDE.md di seluruh organisasi

Organisasi dapat menerapkan CLAUDE.md yang dikelola secara terpusat yang berlaku untuk semua pengguna di mesin. File ini tidak dapat dikecualikan oleh pengaturan individual.

Step 1: Buat file di lokasi kebijakan terkelola

- macOS: `/Library/Application Support/ClaudeCode/CLAUDE.md`
- Linux dan WSL: `/etc/claude-code/CLAUDE.md`
- Windows: `C:\Program Files\ClaudeCode\CLAUDE.md`

Step 2: Terapkan dengan sistem manajemen konfigurasi Anda

Gunakan MDM, Group Policy, Ansible, atau alat serupa untuk mendistribusikan file di seluruh mesin pengembang. Lihat pengaturan terkelola untuk opsi konfigurasi di seluruh organisasi lainnya.

Kecualikan file CLAUDE.md tertentu

Dalam monorepo besar, file CLAUDE.md leluhur mungkin berisi instruksi yang tidak relevan dengan pekerjaan Anda. Pengaturan `claudeMdExcludes` memungkinkan Anda melewati file tertentu berdasarkan jalur atau pola glob.

Contoh ini mengecualikan CLAUDE.md tingkat atas dan direktori aturan dari folder induk. Tambahkan ke `.claude/settings.local.json` agar pengecualian tetap lokal ke mesin Anda:

```
{
  "claudeMdExcludes": [
    "**/monorepo/CLAUDE.md",
    "/home/user/monorepo/other-team/.claude/rules/**"
  ]
}
```

Pola dicocokkan dengan jalur file absolut menggunakan sintaks glob. Anda dapat mengonfigurasi `claudeMdExcludes` di lapisan pengaturan apa pun: pengguna, proyek, lokal, atau kebijakan terkelola. Array digabungkan di seluruh lapisan.

File CLAUDE.md kebijakan terkelola tidak dapat dikecualikan. Ini memastikan instruksi di seluruh organisasi selalu berlaku terlepas dari pengaturan individual.

Auto memory

Auto memory memungkinkan Claude mengumpulkan pengetahuan lintas sesi tanpa Anda menulis apa pun. Claude menyimpan catatan untuk dirinya sendiri saat bekerja: perintah build, wawasan debugging, catatan arsitektur, preferensi gaya kode, dan kebiasaan alur kerja. Claude tidak menyimpan sesuatu setiap sesi. Itu memutuskan apa yang layak diingat berdasarkan apakah informasi akan berguna dalam percakapan masa depan.

> **Note:**
>
> Auto memory memerlukan Claude Code v2.1.59 atau lebih baru. Periksa versi Anda dengan `claude --version`.

Aktifkan atau nonaktifkan auto memory

Auto memory aktif secara default. Untuk mengalihkannya, buka `/memory` dalam sesi dan gunakan toggle auto memory, atau atur `autoMemoryEnabled` dalam pengaturan proyek Anda:

```
{
  "autoMemoryEnabled": false
}
```

Untuk menonaktifkan auto memory melalui variabel lingkungan, atur `CLAUDE_CODE_DISABLE_AUTO_MEMORY=1`.

Lokasi penyimpanan

Setiap proyek mendapatkan direktori memori sendiri di `~/.claude/projects/<project>/memory/`. Jalur `<project>` berasal dari repositori git, jadi semua worktrees dan subdirektori dalam repo yang sama berbagi satu direktori auto memory. Di luar repo git, root proyek digunakan sebagai gantinya.

Untuk menyimpan auto memory di lokasi berbeda, atur `autoMemoryDirectory` dalam pengaturan pengguna atau lokal Anda:

```
{
  "autoMemoryDirectory": "~/my-custom-memory-dir"
}
```

Pengaturan ini diterima dari pengaturan kebijakan, lokal, dan pengguna. Itu tidak diterima dari pengaturan proyek (`.claude/settings.json`) untuk mencegah proyek bersama mengarahkan ulang penulisan auto memory ke lokasi sensitif.

Direktori berisi titik masuk `MEMORY.md` dan file topik opsional:

```
~/.claude/projects/<project>/memory/
├── MEMORY.md          # Indeks ringkas, dimuat ke dalam setiap sesi
├── debugging.md       # Catatan terperinci tentang pola debugging
├── api-conventions.md # Keputusan desain API
└── ...                # File topik lain yang Claude buat
```

`MEMORY.md` bertindak sebagai indeks direktori memori. Claude membaca dan menulis file di direktori ini sepanjang sesi Anda, menggunakan `MEMORY.md` untuk melacak apa yang disimpan di mana.

Auto memory adalah mesin-lokal. Semua worktrees dan subdirektori dalam repositori git yang sama berbagi satu direktori auto memory. File tidak dibagikan di seluruh mesin atau lingkungan cloud.

Bagaimana cara kerjanya

200 baris pertama `MEMORY.md` dimuat di awal setiap percakapan. Konten di luar baris 200 tidak dimuat saat awal sesi. Claude membuat `MEMORY.md` ringkas dengan memindahkan catatan terperinci ke file topik terpisah.

Batas 200 baris ini hanya berlaku untuk `MEMORY.md` . File CLAUDE.md dimuat sepenuhnya terlepas dari panjangnya, meskipun file yang lebih pendek menghasilkan kepatuhan yang lebih baik.

File topik seperti `debugging.md` atau `patterns.md` tidak dimuat saat startup. Claude membacanya sesuai permintaan menggunakan alat file standarnya ketika membutuhkan informasi.

Claude membaca dan menulis file memori selama sesi Anda. Ketika Anda melihat “Writing memory” atau “Recalled memory” di antarmuka Claude Code, Claude secara aktif memperbarui atau membaca dari `~/.claude/projects/<project>/memory/` .

Audit dan edit memori Anda

File auto memory adalah markdown biasa yang dapat Anda edit atau hapus kapan saja. Jalankan `/memory` untuk menelusuri dan membuka file memori dari dalam sesi.

Lihat dan edit dengan `/memory`

Perintah `/memory` mencantumkan semua file CLAUDE.md dan rules yang dimuat dalam sesi saat ini, memungkinkan Anda mengalihkan auto memory on atau off, dan menyediakan tautan untuk membuka folder auto memory. Pilih file apa pun untuk membukanya di editor Anda.

Ketika Anda meminta Claude untuk mengingat sesuatu, seperti “selalu gunakan pnpm, bukan npm” atau “ingat bahwa tes API memerlukan instans Redis lokal,” Claude menyimpannya ke auto memory. Untuk menambahkan instruksi ke CLAUDE.md sebagai gantinya, minta Claude secara langsung, seperti “tambahkan ini ke CLAUDE.md,” atau edit file sendiri melalui `/memory` .

Memecahkan masalah memori

Ini adalah masalah paling umum dengan CLAUDE.md dan auto memory, bersama dengan langkah-langkah untuk men-debug mereka.

Claude tidak mengikuti CLAUDE.md saya

CLAUDE.md adalah konteks, bukan penegakan. Claude membacanya dan mencoba mengikutinya, tetapi tidak ada jaminan kepatuhan ketat, terutama untuk instruksi yang samar atau bertentangan.

Untuk men-debug:

- Jalankan `/memory` untuk memverifikasi file CLAUDE.md Anda dimuat. Jika file tidak terdaftar, Claude tidak dapat melihatnya.
- Periksa bahwa CLAUDE.md yang relevan berada di lokasi yang dimuat untuk sesi Anda (lihat Pilih di mana menempatkan file CLAUDE.md).
- Buat instruksi lebih spesifik. “Gunakan indentasi 2 spasi” bekerja lebih baik daripada “format kode dengan baik.”
- Cari instruksi yang bertentangan di seluruh file CLAUDE.md. Jika dua file memberikan panduan berbeda untuk perilaku yang sama, Claude mungkin memilih satu secara sembarangan.

> **Tip:**
>
> Gunakan hook `InstructionsLoaded` untuk mencatat dengan tepat file instruksi mana yang dimuat, kapan mereka dimuat, dan mengapa. Ini berguna untuk men-debug aturan khusus jalur atau file yang dimuat malas di subdirektori.

Saya tidak tahu apa yang disimpan auto memory

Jalankan `/memory` dan pilih folder auto memory untuk menelusuri apa yang Claude simpan. Semuanya adalah markdown biasa yang dapat Anda baca, edit, atau hapus.

CLAUDE.md saya terlalu besar

File di atas 200 baris mengonsumsi lebih banyak konteks dan dapat mengurangi kepatuhan. Pindahkan konten terperinci ke file terpisah yang direferensikan dengan impor `@path` (lihat Impor file tambahan), atau pisahkan instruksi Anda di seluruh file `.claude/rules/`.

Instruksi tampak hilang setelah `/compact`

CLAUDE.md sepenuhnya bertahan dari pemadatan. Setelah `/compact`, Claude membaca ulang CLAUDE.md Anda dari disk dan menyuntikkannya kembali segar ke dalam sesi. Jika instruksi hilang setelah pemadatan, itu diberikan hanya dalam percakapan, bukan ditulis ke CLAUDE.md. Tambahkan ke CLAUDE.md untuk membuatnya bertahan lintas sesi.

Lihat Tulis instruksi yang efektif untuk panduan tentang ukuran, struktur, dan spesifisitas.

Sumber daya terkait

- Skills: paket alur kerja yang dapat diulang yang dimuat sesuai permintaan
- Settings: konfigurasi perilaku Claude Code dengan file pengaturan
- Kelola sesi: kelola konteks, lanjutkan percakapan, dan jalankan sesi paralel
- Memori subagent: biarkan subagents mempertahankan auto memory mereka sendiri

Pengaturan Claude Code

> *Konfigurasikan Claude Code dengan pengaturan global dan tingkat proyek, serta variabel lingkungan.*

Claude Code menawarkan berbagai pengaturan untuk mengonfigurasi perilakunya sesuai kebutuhan Anda. Anda dapat mengonfigurasi Claude Code dengan menjalankan perintah `/config` saat menggunakan REPL interaktif, yang membuka antarmuka Pengaturan bertab di mana Anda dapat melihat informasi status dan memodifikasi opsi konfigurasi.

Cakupan konfigurasi

Claude Code menggunakan **sistem cakupan** untuk menentukan di mana konfigurasi berlaku dan siapa yang membagikannya. Memahami cakupan membantu Anda memutuskan cara mengonfigurasi Claude Code untuk penggunaan pribadi, kolaborasi tim, atau penyebaran perusahaan.

Cakupan yang tersedia

Cakupan	Lokasi	Siapa yang terpengaruh	Dibagikan dengan tim?
Managed	Pengaturan yang dikelola server, plist / registry, atau `managed-settings.json` tingkat sistem	Semua pengguna di mesin	Ya (digunakan oleh IT)
User	Direktori `~/.claude/`	Anda, di semua proyek	Tidak
Project	`.claude/` di repositori	Semua kolaborator di repositori ini	Ya (dikomit ke git)
Local	`.claude/settings.local.json`	Anda, hanya di repositori ini	Tidak (diabaikan git)

Kapan menggunakan setiap cakupan

Cakupan Managed adalah untuk:

- Kebijakan keamanan yang harus diterapkan di seluruh organisasi

- Persyaratan kepatuhan yang tidak dapat ditimpa
- Konfigurasi standar yang digunakan oleh IT/DevOps

Cakupan User paling baik untuk:

- Preferensi pribadi yang Anda inginkan di mana-mana (tema, pengaturan editor)
- Alat dan plugin yang Anda gunakan di semua proyek
- Kunci API dan autentikasi (disimpan dengan aman)

Cakupan Project paling baik untuk:

- Pengaturan bersama tim (izin, hooks, MCP servers)
- Plugin yang harus dimiliki seluruh tim
- Standardisasi alat di seluruh kolaborator

Cakupan Local paling baik untuk:

- Penggantian pribadi untuk proyek tertentu
- Pengaturan pengujian sebelum dibagikan dengan tim
- Pengaturan spesifik mesin yang tidak akan berfungsi untuk orang lain

Bagaimana cakupan berinteraksi

Ketika pengaturan yang sama dikonfigurasi dalam beberapa cakupan, cakupan yang lebih spesifik memiliki prioritas:

1. **Managed** (tertinggi) - tidak dapat ditimpa oleh apa pun
2. **Argumen baris perintah** - penggantian sesi sementara
3. **Local** - menimpa pengaturan proyek dan pengguna
4. **Project** - menimpa pengaturan pengguna
5. **User** (terendah) - berlaku ketika tidak ada yang menentukan pengaturan

Misalnya, jika izin diizinkan dalam pengaturan pengguna tetapi ditolak dalam pengaturan proyek, pengaturan proyek memiliki prioritas dan izin diblokir.

Apa yang menggunakan cakupan

Cakupan berlaku untuk banyak fitur Claude Code:

Fitur	Lokasi pengguna	Lokasi proyek	Lokasi lokal
Settings	`~/.claude/settings.json`	`.claude/settings.json`	`.claude/settings.local.json`

Fitur	Lokasi pengguna	Lokasi proyek	Lokasi lokal
Subagen-ts	`~/.claude/agents/`	`.claude/agents/`	—
MCP ser-vers	`~/.claude.json`	`.mcp.json`	`~/.claude.json` (per-proyek)
Plugins	`~/.claude/settings.json`	`.claude/settings.json`	`.claude/settings.local.json`
CLAU-DE.md	`~/.claude/CLAUDE.md`	`CLAUDE.md` atau `.claude/CLAUDE.md`	—

File pengaturan

File `settings.json` adalah mekanisme resmi kami untuk mengonfigurasi Claude Code melalui pengaturan hierarki:

- **Pengaturan pengguna** didefinisikan dalam `~/.claude/settings.json` dan berlaku untuk semua proyek.
- **Pengaturan proyek** disimpan di direktori proyek Anda:
  - `.claude/settings.json` untuk pengaturan yang diperiksa ke dalam kontrol sumber dan dibagikan dengan tim Anda
  - `.claude/settings.local.json` untuk pengaturan yang tidak diperiksa, berguna untuk preferensi pribadi dan eksperimen. Claude Code akan mengonfigurasi git untuk mengabaikan `.claude/settings.local.json` saat dibuat.
- **Pengaturan Managed**: Untuk organisasi yang memerlukan kontrol terpusat, Claude Code mendukung beberapa mekanisme pengiriman untuk pengaturan yang dikelola. Semua menggunakan format JSON yang sama dan tidak dapat ditimpa oleh pengaturan pengguna atau proyek:
  - **Pengaturan yang dikelola server**: dikirimkan dari server Anthropic melalui konsol admin Claude.ai. Lihat pengaturan yang dikelola server.
  - **Kebijakan tingkat MDM/OS**: dikirimkan melalui manajemen perangkat asli di macOS dan Windows:
  - macOS: domain preferensi terkelola `com.anthropic.claudecode` (digunakan melalui profil konfigurasi di Jamf, Kandji, atau alat MDM lainnya)

- Windows: kunci registry `HKLM\SOFTWARE\Policies\ClaudeCode` dengan nilai `Settings` (REG_SZ atau REG_EXPAND_SZ) yang berisi JSON (digunakan melalui Group Policy atau Intune)
- Windows (tingkat pengguna): `HKCU\SOFTWARE\Policies\ClaudeCode` (prioritas kebijakan terendah, hanya digunakan ketika tidak ada sumber tingkat admin)
- **Berbasis file**: `managed-settings.json` dan `managed-mcp.json` digunakan ke direktori sistem:
- macOS: `/Library/Application Support/ClaudeCode/`
- Linux dan WSL: `/etc/claude-code/`
- Windows: `C:\Program Files\ClaudeCode\`

Lihat pengaturan yang dikelola dan Konfigurasi MCP Managed untuk detail.

> **Note:**
>
> Penyebaran yang dikelola juga dapat membatasi **penambahan marketplace plugin** menggunakan `strictKnownMarketplaces`. Untuk informasi lebih lanjut, lihat Pembatasan marketplace yang dikelola.

- **Konfigurasi lainnya** disimpan dalam `~/.claude.json`. File ini berisi preferensi Anda (tema, pengaturan notifikasi, mode editor), sesi OAuth, konfigurasi MCP server untuk cakupan pengguna dan lokal, status per-proyek (alat yang diizinkan, pengaturan kepercayaan), dan berbagai cache. MCP server dengan cakupan proyek disimpan secara terpisah dalam `.mcp.json`.

> **Note:**
>
> Claude Code secara otomatis membuat cadangan file konfigurasi dengan stempel waktu dan menyimpan lima cadangan terbaru untuk mencegah kehilangan data.

```
{
  "$schema": "https://json.schemastore.org/claude-code-settings.json",
  "permissions": {
    "allow": [
      "Bash(npm run lint)",
      "Bash(npm run test *)",
      "Read(~/.zshrc)"
    ],
    "deny": [
      "Bash(curl *)",
      "Read(./.env)",
      "Read(./.env.*)",
      "Read(./secrets/**)"
    ]
  },
  "env": {
    "CLAUDE_CODE_ENABLE_TELEMETRY": "1",
    "OTEL_METRICS_EXPORTER": "otlp"
  },
  "companyAnnouncements": [
    "Welcome to Acme Corp! Review our code guidelines at docs.acme.com",
    "Reminder: Code reviews required for all PRs",
    "New security policy in effect"
  ]
}
```

Baris `$schema` dalam contoh di atas menunjuk ke skema JSON resmi untuk pengaturan Claude Code. Menambahkannya ke `settings.json` Anda memungkinkan pelengkapan otomatis dan validasi inline di VS Code, Cursor, dan editor lain yang mendukung validasi skema JSON.

Pengaturan yang tersedia

`settings.json` mendukung sejumlah opsi:

Kunci	Deskripsi	Contoh
`apiKeyHelper`	Skrip khusus, yang akan dieksekusi dalam `/bin/sh`, untuk menghasilkan nilai auth. Nilai ini akan dikirim sebagai header `X-Api-Key` dan `Authorization: Bearer` untuk permintaan model	`/bin/generate_temp_api_key.sh`
`cleanupPeriodDays`	Sesi yang tidak aktif lebih lama dari periode ini dihapus saat startup. Mengatur ke `0` segera menghapus semua sesi. (default: 30 hari)	`20`
`companyAnnouncements`	Pengumuman untuk ditampilkan kepada pengguna saat startup. Jika beberapa pengumuman disediakan, mereka akan diputar secara acak.	`["Welcome to Acme Corp! Review our code guidelines at docs.acme.com"]`
`env`	Variabel lingkungan yang akan diterapkan ke setiap sesi	`{"FOO": "bar"}`
`attribution`	Sesuaikan atribusi untuk komit git dan permintaan tarik. Lihat Pengaturan atribusi	`{"commit": "🤖 Generated with Claude Code", "pr": ""}`
`includeCoAuthoredBy`	**Tidak direkomendasikan**: Gunakan `attribution` sebagai gantinya. Apakah akan menyertakan baris `co-authored-by Claude` dalam komit git dan permintaan tarik (default: `true`)	`false`
`includeGitInstructions`	Sertakan instruksi alur kerja komit dan PR bawaan dalam prompt sistem Claude (default: `true`). Atur ke `false` untuk menghapus instruksi ini, misalnya saat menggunakan skill alur kerja git Anda sendiri. Variabel lingkungan `CLAUDE_CODE_DISABLE_GIT_INSTRUCTIONS` memiliki prioritas atas pengaturan ini saat diatur	`false`

Kunci	Deskripsi	Contoh
`permissions`	Lihat tabel di bawah untuk struktur izin.	
`hooks`	Konfigurasikan perintah khusus untuk dijalankan pada acara siklus hidup. Lihat dokumentasi hooks untuk format	Lihat hooks
`disableAllHooks`	Nonaktifkan semua hooks dan status line khusus apa pun	`true`
`allowManagedHooksOnly`	(Hanya pengaturan yang dikelola) Cegah pemuatan hook pengguna, proyek, dan plugin. Hanya memungkinkan hook yang dikelola dan hook SDK. Lihat Konfigurasi Hook	`true`
`allowedHttpHookUrls`	Daftar putih pola URL yang dapat ditargetkan hook HTTP. Mendukung `*` sebagai wildcard. Saat diatur, hook dengan URL yang tidak cocok diblokir. Tidak terdefinisi = tidak ada pembatasan, array kosong = blokir semua hook HTTP. Array digabungkan di seluruh sumber pengaturan. Lihat Konfigurasi Hook	`["https://hooks.example.com/*"]`
`httpHookAllowedEnvVars`	Daftar putih nama variabel lingkungan yang dapat diinterpolasi hook HTTP ke dalam header. Saat diatur, `allowedEnvVars` efektif setiap hook adalah persimpangan dengan daftar ini. Tidak terdefinisi = tidak ada pembatasan. Array digabungkan di seluruh sumber pengaturan. Lihat Konfigurasi Hook	`["MY_TOKEN", "HOOK_SECRET"]`

Kunci	Deskripsi	Contoh
`allowManagedPermissionRulesOnly`	(Hanya pengaturan yang dikelola) Cegah pengaturan pengguna dan proyek dari mendefinisikan aturan izin `allow`, `ask`, atau `deny`. Hanya aturan dalam pengaturan yang dikelola yang berlaku. Lihat Pengaturan khusus yang dikelola	`true`
`allowManagedMcpServersOnly`	(Hanya pengaturan yang dikelola) Hanya `allowedMcpServers` dari pengaturan yang dikelola yang dihormati. `deniedMcpServers` masih digabungkan dari semua sumber. Pengguna masih dapat menambahkan MCP servers, tetapi hanya daftar putih yang ditentukan admin yang berlaku. Lihat Konfigurasi MCP Managed	`true`
`model`	Timpa model default untuk digunakan untuk Claude Code	`"claude-sonnet-4-6"`
`availableModels`	Batasi model mana yang dapat dipilih pengguna melalui `/model`, `--model`, alat Config, atau `ANTHROPIC_MODEL`. Tidak mempengaruhi opsi Default. Lihat Batasi pemilihan model	`["sonnet", "haiku"]`
`modelOverrides`	Petakan ID model Anthropic ke ID model spesifik penyedia seperti ARN profil inferensi Bedrock. Setiap entri pemilih model menggunakan nilai yang dipetakan saat memanggil API penyedia. Lihat Timpa ID model per versi	`{"claude-opus-4-6": "arn:aws:bedrock:..."}`
`otelHeadersHelper`	Skrip untuk menghasilkan header OpenTelemetry dinamis. Berjalan saat startup dan secara berkala (lihat Header dinamis)	`/bin/generate_otel_headers.sh`

Kunci	Deskripsi	Contoh
`statusLine`	Konfigurasikan status line khusus untuk menampilkan konteks. Lihat dokumentasi `statusLine`	`{"type": "command", "command": "~/.claude/statusline.sh"}`
`fileSuggestion`	Konfigurasikan skrip khusus untuk pelengkapan otomatis file `@`. Lihat Pengaturan saran file	`{"type": "command", "command": "~/.claude/file-suggestion.sh"}`
`respectGitignore`	Kontrol apakah pemilih file `@` menghormati pola `.gitignore`. Saat `true` (default), file yang cocok dengan pola `.gitignore` dikecualikan dari saran	`false`
`outputStyle`	Konfigurasikan gaya output untuk menyesuaikan prompt sistem. Lihat dokumentasi gaya output	`"Explanatory"`
`forceLoginMethod`	Gunakan `claudeai` untuk membatasi login ke akun Claude.ai, `console` untuk membatasi login ke akun Claude Console (penagihan penggunaan API)	`claudeai`
`forceLoginOrgUUID`	Tentukan UUID organisasi untuk secara otomatis memilihnya selama login, melewati langkah pemilihan organisasi. Memerlukan `forceLoginMethod` untuk diatur	`"xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"`
`enableAllProjectMcpServers`	Secara otomatis menyetujui semua MCP servers yang ditentukan dalam file `.mcp.json` proyek	`true`
`enabledMcpjsonServers`	Daftar MCP servers tertentu dari file `.mcp.json` untuk disetujui	`["memory", "github"]`
`disabledMcpjsonServers`	Daftar MCP servers tertentu dari file `.mcp.json` untuk ditolak	`["filesystem"]`

Kunci	Deskripsi	Contoh
`allowedMcpServers`	Saat diatur dalam managed-settings.json, daftar putih MCP servers yang dapat dikonfigurasi pengguna. Tidak terdefinisi = tidak ada pembatasan, array kosong = lockdown. Berlaku untuk semua cakupan. Daftar hitam memiliki prioritas. Lihat Konfigurasi MCP Managed	`[{ "serverName": "github" }]`
`deniedMcpServers`	Saat diatur dalam managed-settings.json, daftar hitam MCP servers yang secara eksplisit diblokir. Berlaku untuk semua cakupan termasuk server yang dikelola. Daftar hitam memiliki prioritas atas daftar putih. Lihat Konfigurasi MCP Managed	`[{ "serverName": "filesystem" }]`
`strictKnownMarketplaces`	Saat diatur dalam managed-settings.json, daftar putih marketplace plugin yang dapat ditambahkan pengguna. Tidak terdefinisi = tidak ada pembatasan, array kosong = lockdown. Berlaku hanya untuk penambahan marketplace. Lihat Pembatasan marketplace yang dikelola	`[{ "source": "github", "repo": "acme-corp/plugins" }]`
`blockedMarketplaces`	(Hanya pengaturan yang dikelola) Daftar hitam sumber marketplace. Sumber yang diblokir diperiksa sebelum mengunduh, jadi mereka tidak pernah menyentuh sistem file. Lihat Pembatasan marketplace yang dikelola	`[{ "source": "github", "repo": "untrusted/plugins" }]`

Kunci	Deskripsi	Contoh
`pluginTrustMessage`	(Hanya pengaturan yang dikelola) Pesan khusus ditambahkan ke peringatan kepercayaan plugin yang ditampilkan sebelum instalasi. Gunakan ini untuk menambahkan konteks spesifik organisasi, misalnya untuk mengonfirmasi bahwa plugin dari marketplace internal Anda telah disaring.	`"All plugins from our marketplace are approved by IT"`
`awsAuthRefresh`	Skrip khusus yang memodifikasi direktori `.aws` (lihat konfigurasi kredensial lanjutan)	`aws sso login --profile myprofile`
`awsCredentialExport`	Skrip khusus yang menampilkan JSON dengan kredensial AWS (lihat konfigurasi kredensial lanjutan)	`/bin/generate_aws_grant.sh`
`alwaysThinkingEnabled`	Aktifkan extended thinking secara default untuk semua sesi. Biasanya dikonfigurasi melalui perintah `/config` daripada mengedit langsung	`true`
`plansDirectory`	Sesuaikan di mana file rencana disimpan. Jalur relatif terhadap akar proyek. Default: `~/.claude/plans`	`"./plans"`
`showTurnDuration`	Tampilkan pesan durasi giliran setelah respons (misalnya, “Cooked for 1m 6s”). Atur ke `false` untuk menyembunyikan pesan ini	`true`
`spinnerVerbs`	Sesuaikan kata kerja aksi yang ditampilkan dalam spinner dan pesan durasi giliran. Atur `mode` ke `"replace"` untuk menggunakan hanya kata kerja Anda, atau `"append"` untuk menambahkannya ke default	`{"mode": "append", "verbs": ["Pondering", "Crafting"]}`

Kunci	Deskripsi	Contoh
`language`	Konfigurasikan bahasa respons pilihan Claude (misalnya, `"japanese"`, `"spanish"`, `"french"`). Claude akan merespons dalam bahasa ini secara default	`"japanese"`
`autoUpdatesChannel`	Saluran rilis untuk diikuti untuk pembaruan. Gunakan `"stable"` untuk versi yang biasanya sekitar satu minggu lama dan melewati versi dengan regresi besar, atau `"latest"` (default) untuk rilis terbaru	`"stable"`
`spinnerTipsEnabled`	Tampilkan tips dalam spinner saat Claude bekerja. Atur ke `false` untuk menonaktifkan tips (default: `true`)	`false`
`spinnerTipsOverride`	Timpa tips spinner dengan string khusus. `tips`: array string tip. `excludeDefault`: jika `true`, hanya tampilkan tip khusus; jika `false` atau tidak ada, tip khusus digabungkan dengan tip bawaan	`{ "excludeDefault": true, "tips": ["Use our internal tool X"] }`
`terminalProgressBarEnabled`	Aktifkan bilah kemajuan terminal yang menampilkan kemajuan di terminal yang didukung seperti Windows Terminal dan iTerm2 (default: `true`)	`false`
`prefersReducedMotion`	Kurangi atau nonaktifkan animasi UI (spinner, shimmer, efek flash) untuk aksesibilitas	`true`

Kunci	Deskripsi	Contoh
`fastModePerSessionOptIn`	Saat `true`, mode cepat tidak bertahan di seluruh sesi. Setiap sesi dimulai dengan mode cepat mati, memerlukan pengguna untuk mengaktifkannya dengan `/fast`. Preferensi mode cepat pengguna masih disimpan. Lihat Memerlukan opt-in per sesi	`true`
`teammateMode`	Bagaimana rekan tim agent ditampilkan: `auto` (memilih panel terpisah di tmux atau iTerm2, in-process sebaliknya), `in-process`, atau `tmux`. Lihat atur agent teams	`"in-process"`

Pengaturan izin

Kunci	Deskripsi	Contoh
`allow`	Array aturan izin untuk memungkinkan penggunaan alat. Lihat Sintaks aturan izin di bawah untuk detail pencocokan pola	`[ "Bash(git diff *)" ]`
`ask`	Array aturan izin untuk meminta konfirmasi saat penggunaan alat. Lihat Sintaks aturan izin di bawah	`[ "Bash(git push *)" ]`
`deny`	Array aturan izin untuk menolak penggunaan alat. Gunakan ini untuk mengecualikan file sensitif dari akses Claude Code. Lihat Sintaks aturan izin dan Batasan izin Bash	`[ "WebFetch", "Bash(curl *)", "Read(./.env)", "Read(./secrets/**)" ]`
`additionalDirectories`	Direktori kerja tambahan yang dapat diakses Claude	`[ "../docs/" ]`
`defaultMode`	Mode izin default saat membuka Claude Code	`"acceptEdits"`

Kunci	Deskripsi	Contoh
`disableBypassPermissionsMode`	Atur ke `"disable"` untuk mencegah mode `bypassPermissions` diaktifkan. Ini menonaktifkan bendera baris perintah `--dangerously-skip-permissions`. Lihat pengaturan yang dikelola	`"disable"`

Sintaks aturan izin

Aturan izin mengikuti format `Tool` atau `Tool(specifier)`. Aturan dievaluasi secara berurutan: aturan deny terlebih dahulu, kemudian ask, kemudian allow. Aturan pertama yang cocok menang.

Contoh cepat:

Aturan	Efek
`Bash`	Cocok dengan semua perintah Bash
`Bash(npm run *)`	Cocok dengan perintah yang dimulai dengan `npm run`
`Read(./.env)`	Cocok dengan membaca file `.env`
`WebFetch(domain:example.com)`	Cocok dengan permintaan fetch ke example.com

Untuk referensi sintaks aturan lengkap, termasuk perilaku wildcard, pola spesifik alat untuk Read, Edit, WebFetch, MCP, dan aturan Agent, dan batasan keamanan pola Bash, lihat Sintaks aturan izin.

Pengaturan sandbox

Konfigurasikan perilaku sandboxing lanjutan. Sandboxing mengisolasi perintah bash dari sistem file dan jaringan Anda. Lihat Sandboxing untuk detail.

Kunci	Deskripsi	Contoh
`enabled`	Aktifkan bash sandboxing (macOS, Linux, dan WSL2). Default: false	`true`
`autoAllowBashIfSandboxed`	Persetujuan otomatis perintah bash saat sandboxed. Default: true	`true`

Kunci	Deskripsi	Contoh
`excludedCommands`	Perintah yang harus dijalankan di luar sandbox	`["git", "docker"]`
`allowUnsandboxedCommands`	Izinkan perintah untuk dijalankan di luar sandbox melalui parameter `dangerouslyDisableSandbox`. Saat diatur ke `false`, pintu keluar `dangerouslyDisableSandbox` sepenuhnya dinonaktifkan dan semua perintah harus dijalankan sandboxed (atau berada dalam `excludedCommands`). Berguna untuk kebijakan perusahaan yang memerlukan sandboxing ketat. Default: true	`false`
`filesystem.allowWrite`	Jalur tambahan di mana perintah sandboxed dapat menulis. Array digabungkan di semua cakupan pengaturan: jalur pengguna, proyek, dan yang dikelola digabungkan, bukan diganti. Juga digabungkan dengan jalur dari aturan izin `Edit(...)` allow. Lihat prefiks jalur sandbox di bawah.	`["//tmp/build", "~/.kube"]`
`filesystem.denyWrite`	Jalur di mana perintah sandboxed tidak dapat menulis. Array digabungkan di semua cakupan pengaturan. Juga digabungkan dengan jalur dari aturan izin `Edit(...)` deny.	`["//etc", "//usr/local/bin"]`
`filesystem.denyRead`	Jalur di mana perintah sandboxed tidak dapat membaca. Array digabungkan di semua cakupan pengaturan. Juga digabungkan dengan jalur dari aturan izin `Read(...)` deny.	`["~/.aws/credentials"]`
`network.allowUnixSockets`	Jalur soket Unix yang dapat diakses dalam sandbox (untuk agen SSH, dll.)	`["~/.ssh/agent-socket"]`

Kunci	Deskripsi	Contoh
`network.allowAllUnixSockets`	Izinkan semua koneksi soket Unix dalam sandbox. Default: false	`true`
`network.allowLocalBinding`	Izinkan pengikatan ke port localhost (hanya macOS). Default: false	`true`
`network.allowedDomains`	Array domain untuk memungkinkan lalu lintas jaringan keluar. Mendukung wildcard (misalnya, `*.example.com`).	`["github.com", "*.npmjs.org"]`
`network.allowManagedDomainsOnly`	(Hanya pengaturan yang dikelola) Hanya `allowedDomains` dan aturan allow `WebFetch(domain:...)` dari pengaturan yang dikelola yang dihormati. Domain dari pengaturan pengguna, proyek, dan lokal diabaikan. Domain yang tidak diizinkan diblokir secara otomatis tanpa meminta pengguna. Domain yang ditolak masih dihormati dari semua sumber. Default: false	`true`
`network.httpProxyPort`	Port proxy HTTP yang digunakan jika Anda ingin membawa proxy Anda sendiri. Jika tidak ditentukan, Claude akan menjalankan proxy-nya sendiri.	`8080`
`network.socksProxyPort`	Port proxy SOCKS5 yang digunakan jika Anda ingin membawa proxy Anda sendiri. Jika tidak ditentukan, Claude akan menjalankan proxy-nya sendiri.	`8081`
`enableWeakerNestedSandbox`	Aktifkan sandbox yang lebih lemah untuk lingkungan Docker yang tidak istimewa (hanya Linux dan WSL2). **Mengurangi keamanan.** Default: false	`true`

Kunci	Deskripsi	Contoh
`enableWeakerNetworkIsolation`	(Hanya macOS) Izinkan akses ke layanan kepercayaan TLS sistem (`com.apple.trustd.agent`) dalam sandbox. Diperlukan untuk alat berbasis Go seperti `gh`, `gcloud`, dan `terraform` untuk memverifikasi sertifikat TLS saat menggunakan `httpProxyPort` dengan proxy MITM dan CA khusus. **Mengurangi keamanan** dengan membuka jalur eksfiltrasi data potensial. Default: false	`true`

Prefiks jalur sandbox

Jalur dalam `filesystem.allowWrite`, `filesystem.denyWrite`, dan `filesystem.denyRead` mendukung prefiks ini:

Prefiks	Arti	Contoh
`//`	Jalur absolut dari akar sistem file	`//tmp/build` menjadi `/tmp/build`
`~/`	Relatif terhadap direktori home	`~/.kube` menjadi `$HOME/.kube`
`/`	Relatif terhadap direktori file pengaturan	`/build` menjadi `$SETTINGS_DIR/build`
`./` atau tidak ada prefiks	Jalur relatif (diselesaikan oleh runtime sandbox)	`./output`

Contoh konfigurasi:

```
{
  "sandbox": {
    "enabled": true,
    "autoAllowBashIfSandboxed": true,
    "excludedCommands": ["docker"],
    "filesystem": {
      "allowWrite": ["//tmp/build", "~/.kube"],
      "denyRead": ["~/.aws/credentials"]
    },
    "network": {
      "allowedDomains": ["github.com", "*.npmjs.org", "registry.yarnpkg.com"],
      "allowUnixSockets": [
        "/var/run/docker.sock"
      ],
      "allowLocalBinding": true
    }
  }
}
```

Pembatasan sistem file dan jaringan dapat dikonfigurasi dalam dua cara yang digabungkan bersama:

- **Pengaturan** `sandbox.filesystem` (ditampilkan di atas): Kontrol jalur pada batas sandbox tingkat OS. Pembatasan ini berlaku untuk semua perintah subprocess (misalnya, `kubectl`, `terraform`, `npm`), bukan hanya alat file Claude.
- **Aturan izin**: Gunakan aturan allow/deny `Edit` untuk mengontrol akses alat file Claude, aturan deny `Read` untuk memblokir pembacaan, dan aturan allow/deny `WebFetch` untuk mengontrol domain jaringan. Jalur dari aturan ini juga digabungkan ke dalam konfigurasi sandbox.

Pengaturan atribusi

Claude Code menambahkan atribusi ke komit git dan permintaan tarik. Ini dikonfigurasi secara terpisah:

- Komit menggunakan git trailers (seperti `Co-Authored-By`) secara default, yang dapat disesuaikan atau dinonaktifkan
- Deskripsi permintaan tarik adalah teks biasa

Kunci	Deskripsi
`commit`	Atribusi untuk komit git, termasuk trailer apa pun. String kosong menyembunyikan atribusi komit
`pr`	Atribusi untuk deskripsi permintaan tarik. String kosong menyembunyikan atribusi permintaan tarik

Atribusi komit default:

```
🤖 Generated with [Claude Code](https://claude.com/claude-code)

   Co-Authored-By: Claude Sonnet 4.6 <noreply@anthropic.com>
```

Atribusi permintaan tarik default:

```
🤖 Generated with [Claude Code](https://claude.com/claude-code)
```

Contoh:

```
{
  "attribution": {
    "commit": "Generated with AI\n\nCo-Authored-By: AI <ai@example.com>",
    "pr": ""
  }
}
```

> **Note:**
>
> Pengaturan `attribution` memiliki prioritas atas pengaturan `includeCoAuthoredBy` yang tidak direkomendasikan. Untuk menyembunyikan semua atribusi, atur `commit` dan `pr` ke string kosong.

Pengaturan saran file

Konfigurasikan perintah khusus untuk pelengkapan otomatis jalur file `@`. Saran file bawaan menggunakan traversal sistem file cepat, tetapi monorepo besar mungkin mendapat manfaat dari pengindeksan spesifik proyek seperti indeks file yang telah dibangun sebelumnya atau alat khusus.

```
{
  "fileSuggestion": {
    "type": "command",
    "command": "~/.claude/file-suggestion.sh"
  }
}
```

Perintah berjalan dengan variabel lingkungan yang sama dengan hooks, termasuk `CLAUDE_PROJECT_DIR`. Ini menerima JSON melalui stdin dengan bidang `query`:

```
{"query": "src/comp"}
```

Keluarkan jalur file yang dipisahkan baris baru ke stdout (saat ini dibatasi hingga 15):

```
src/components/Button.tsx
src/components/Modal.tsx
src/components/Form.tsx
```

Contoh:

```
#!/bin/bash
query=$(cat | jq -r '.query')
your-repo-file-index --query "$query" | head -20
```

Konfigurasi hook

Pengaturan ini mengontrol hook mana yang diizinkan untuk dijalankan dan apa yang dapat diakses hook HTTP. Pengaturan `allowManagedHooksOnly` hanya dapat dikonfigurasi dalam pengaturan yang dikelola. Daftar putih URL dan env var dapat diatur di tingkat pengaturan apa pun dan digabungkan di seluruh sumber.

Perilaku saat `allowManagedHooksOnly` adalah `true`:

- Hook yang dikelola dan hook SDK dimuat
- Hook pengguna, hook proyek, dan hook plugin diblokir

Batasi URL hook HTTP:

Batasi URL mana yang dapat ditargetkan hook HTTP. Mendukung `*` sebagai wildcard untuk pencocokan. Saat array didefinisikan, hook HTTP yang menargetkan URL yang tidak cocok diblokir secara diam-diam.

```
{
  "allowedHttpHookUrls": ["https://hooks.example.com/*", "http://localhost:*"]
}
```

Batasi variabel lingkungan hook HTTP:

Batasi nama variabel lingkungan mana yang dapat diinterpolasi hook HTTP ke dalam nilai header. `allowedEnvVars` efektif setiap hook adalah persimpangan dari daftar sendiri dan pengaturan ini.

```
{
  "httpHookAllowedEnvVars": ["MY_TOKEN", "HOOK_SECRET"]
}
```

Preseden pengaturan

Pengaturan berlaku dalam urutan preseden. Dari tertinggi ke terendah:

1. **Pengaturan yang dikelola** (yang dikelola server, kebijakan tingkat MDM/OS, atau pengaturan yang dikelola)
   - Kebijakan yang digunakan oleh IT melalui pengiriman server, profil konfigurasi MDM, kebijakan registry, atau file pengaturan yang dikelola
   - Tidak dapat ditimpa oleh tingkat lain apa pun, termasuk argumen baris perintah
   - Dalam tingkat yang dikelola, preseden adalah: yang dikelola server > kebijakan tingkat MDM/OS > `managed-settings.json` > registry HKCU (hanya Windows). Hanya satu sumber yang dikelola yang digunakan; sumber tidak digabungkan.
2. **Argumen baris perintah**
   - Penggantian sementara untuk sesi tertentu
3. **Pengaturan proyek lokal** (`.claude/settings.local.json`)
   - Pengaturan proyek pribadi
4. **Pengaturan proyek bersama** (`.claude/settings.json`)
   - Pengaturan proyek bersama tim dalam kontrol sumber

5. **Pengaturan pengguna** (`~/.claude/settings.json`)
    - Pengaturan global pribadi

Hierarki ini memastikan bahwa kebijakan organisasi selalu diterapkan sambil tetap memungkinkan tim dan individu untuk menyesuaikan pengalaman mereka.

Misalnya, jika pengaturan pengguna Anda memungkinkan `Bash(npm run *)` tetapi pengaturan bersama proyek menolaknya, pengaturan proyek memiliki prioritas dan perintah diblokir.

> **Note:**
>
> **Pengaturan array digabungkan di seluruh cakupan.** Ketika pengaturan bernilai array yang sama (seperti `sandbox.filesystem.allowWrite` atau `permissions.allow`) muncul dalam beberapa cakupan, array **digabungkan dan dideduplikasi**, bukan diganti. Ini berarti cakupan prioritas lebih rendah dapat menambahkan entri tanpa menimpa yang ditetapkan oleh cakupan prioritas lebih tinggi, dan sebaliknya. Misalnya, jika pengaturan yang dikelola menetapkan `allowWrite` ke `["//opt/company-tools"]` dan pengguna menambahkan `["~/.kube"]` , kedua jalur disertakan dalam konfigurasi akhir.

Verifikasi pengaturan aktif

Jalankan `/status` di dalam Claude Code untuk melihat sumber pengaturan mana yang aktif dan dari mana asalnya. Output menampilkan setiap lapisan konfigurasi (managed, user, project) bersama dengan asalnya, seperti `Enterprise managed settings (remote)` , `Enterprise managed settings (plist)` , `Enterprise managed settings (HKLM)` , atau `Enterprise managed settings (file)` . Jika file pengaturan berisi kesalahan, `/status` melaporkan masalah sehingga Anda dapat memperbaikinya.

Poin kunci tentang sistem konfigurasi

- **File memori (`CLAUDE.md`)**: Berisi instruksi dan konteks yang dimuat Claude saat startup
- **File pengaturan (JSON)**: Konfigurasikan izin, variabel lingkungan, dan perilaku alat
- **Skills**: Prompt khusus yang dapat dipanggil dengan `/skill-name` atau dimuat oleh Claude secara otomatis
- **MCP servers**: Perluas Claude Code dengan alat dan integrasi tambahan
- **Preseden**: Konfigurasi tingkat lebih tinggi (Managed) menimpa yang tingkat lebih rendah (User/Project)
- **Warisan**: Pengaturan digabungkan, dengan pengaturan yang lebih spesifik menambah atau menimpa yang lebih luas

Prompt sistem

Prompt sistem internal Claude Code tidak dipublikasikan. Untuk menambahkan instruksi khusus, gunakan file `CLAUDE.md` atau bendera `--append-system-prompt`.

Mengecualikan file sensitif

Untuk mencegah Claude Code mengakses file yang berisi informasi sensitif seperti kunci API, rahasia, dan file lingkungan, gunakan pengaturan `permissions.deny` dalam file `.claude/settings.json` Anda:

```
{
  "permissions": {
    "deny": [
      "Read(./.env)",
      "Read(./.env.*)",
      "Read(./secrets/**)",
      "Read(./config/credentials.json)",
      "Read(./build)"
    ]
  }
}
```

Ini menggantikan konfigurasi `ignorePatterns` yang tidak direkomendasikan. File yang cocok dengan pola ini dikecualikan dari penemuan file dan hasil pencarian, dan operasi baca pada file ini ditolak.

Konfigurasi subagent

Claude Code mendukung subagent AI khusus yang dapat dikonfigurasi di tingkat pengguna dan proyek. Subagent ini disimpan sebagai file Markdown dengan frontmatter YAML:

- **Subagent pengguna**: `~/.claude/agents/` - Tersedia di semua proyek Anda
- **Subagent proyek**: `.claude/agents/` - Spesifik untuk proyek Anda dan dapat dibagikan dengan tim Anda

File subagent mendefinisikan asisten AI khusus dengan prompt khusus dan izin alat. Pelajari lebih lanjut tentang membuat dan menggunakan subagent dalam dokumentasi subagent.

Konfigurasi plugin

Claude Code mendukung sistem plugin yang memungkinkan Anda memperluas fungsionalitas dengan skills, agents, hooks, dan MCP servers. Plugin didistribusikan melalui marketplace dan dapat dikonfigurasi di tingkat pengguna dan repositori.

Pengaturan plugin

Pengaturan terkait plugin dalam `settings.json`:

```
{
  "enabledPlugins": {
    "formatter@acme-tools": true,
    "deployer@acme-tools": true,
    "analyzer@security-plugins": false
  },
  "extraKnownMarketplaces": {
    "acme-tools": {
      "source": "github",
      "repo": "acme-corp/claude-plugins"
    }
  }
}
```

`enabledPlugins`

Mengontrol plugin mana yang diaktifkan. Format: `"plugin-name@marketplace-name": true/false`

Cakupan:

- **Pengaturan pengguna** (`~/.claude/settings.json`): Preferensi plugin pribadi
- **Pengaturan proyek** (`.claude/settings.json`): Plugin spesifik proyek yang dibagikan dengan tim
- **Pengaturan lokal** (`.claude/settings.local.json`): Penggantian per-mesin (tidak dikomit)

Contoh:

```
{
  "enabledPlugins": {
    "code-formatter@team-tools": true,
    "deployment-tools@team-tools": true,
    "experimental-features@personal": false
  }
}
```

`extraKnownMarketplaces`

Mendefinisikan marketplace tambahan yang harus tersedia untuk repositori. Biasanya digunakan dalam pengaturan tingkat repositori untuk memastikan anggota tim memiliki akses ke sumber plugin yang diperlukan.

Ketika repositori menyertakan `extraKnownMarketplaces`:

1. Anggota tim diminta untuk menginstal marketplace saat mereka mempercayai folder
2. Anggota tim kemudian diminta untuk menginstal plugin dari marketplace itu
3. Pengguna dapat melewati marketplace atau plugin yang tidak diinginkan (disimpan dalam pengaturan pengguna)
4. Instalasi menghormati batas kepercayaan dan memerlukan persetujuan eksplisit

Contoh:

```
{
  "extraKnownMarketplaces": {
    "acme-tools": {
      "source": {
        "source": "github",
        "repo": "acme-corp/claude-plugins"
      }
    },
    "security-plugins": {
      "source": {
        "source": "git",
        "url": "https://git.example.com/security/plugins.git"
      }
    }
  }
}
```

Jenis sumber marketplace:

- `github` : Repositori GitHub (menggunakan `repo`)
- `git` : URL git apa pun (menggunakan `url`)
- `directory` : Jalur sistem file lokal (menggunakan `path` , hanya untuk pengembangan)
- `hostPattern` : Pola regex untuk mencocokkan host marketplace (menggunakan `hostPattern`)

`strictKnownMarketplaces`

Hanya pengaturan yang dikelola: Mengontrol marketplace plugin mana yang diizinkan pengguna untuk ditambahkan. Pengaturan ini hanya dapat dikonfigurasi dalam pengaturan yang dikelola dan memberikan administrator kontrol ketat atas sumber marketplace.

Lokasi file pengaturan yang dikelola:

- **macOS**: `/Library/Application Support/ClaudeCode/managed-settings.json`
- **Linux dan WSL**: `/etc/claude-code/managed-settings.json`
- **Windows**: `C:\Program Files\ClaudeCode\managed-settings.json`

Karakteristik kunci:

- Hanya tersedia dalam pengaturan yang dikelola (`managed-settings.json`)

- Tidak dapat ditimpa oleh pengaturan pengguna atau proyek (preseden tertinggi)
- Diterapkan SEBELUM operasi jaringan/sistem file (sumber yang diblokir tidak pernah dieksekusi)
- Menggunakan pencocokan tepat untuk spesifikasi sumber (termasuk `ref`, `path` untuk sumber git), kecuali `hostPattern`, yang menggunakan pencocokan regex

Perilaku daftar putih:

- `undefined` (default): Tidak ada pembatasan - pengguna dapat menambahkan marketplace apa pun
- Array kosong `[]`: Lockdown lengkap - pengguna tidak dapat menambahkan marketplace baru apa pun
- Daftar sumber: Pengguna hanya dapat menambahkan marketplace yang cocok dengan tepat

Semua jenis sumber yang didukung:

Daftar putih mendukung tujuh jenis sumber marketplace. Sebagian besar sumber menggunakan pencocokan tepat, sementara `hostPattern` menggunakan pencocokan regex terhadap host marketplace.

1. **Repositori GitHub**:

```
{ "source": "github", "repo": "acme-corp/approved-plugins" }
{ "source": "github", "repo": "acme-corp/security-tools", "ref": "v2.0" }
{ "source": "github", "repo": "acme-corp/plugins", "ref": "main", "path": "marketp
lace" }
```

Bidang: `repo` (diperlukan), `ref` (opsional: cabang/tag/SHA), `path` (opsional: subdirektori)

1. **Repositori Git**:

```
{ "source": "git", "url": "https://gitlab.example.com/tools/plugins.git" }
{ "source": "git", "url": "https://bitbucket.org/acme-corp/plugins.git", "ref": "p
roduction" }
{ "source": "git", "url": "ssh://git@git.example.com/plugins.git", "ref": "v3.1",
"path": "approved" }
```

Bidang: `url` (diperlukan), `ref` (opsional: cabang/tag/SHA), `path` (opsional: subdirektori)

1. **Marketplace berbasis URL**:

```
{ "source": "url", "url": "https://plugins.example.com/marketplace.json" }
{ "source": "url", "url": "https://cdn.example.com/marketplace.json", "headers":
{ "Authorization": "Bearer ${TOKEN}" } }
```

Bidang: `url` (diperlukan), `headers` (opsional: header HTTP untuk akses terautentikasi)

> **Note:**
>
> Marketplace berbasis URL hanya mengunduh file `marketplace.json`. Mereka tidak mengunduh file plugin dari server. Plugin dalam marketplace berbasis URL harus menggunakan sumber eksternal (GitHub, npm, atau URL git) daripada jalur relatif. Untuk plugin dengan jalur relatif, gunakan marketplace berbasis Git sebagai gantinya. Lihat Troubleshooting untuk detail.

1. **Paket NPM**:

```
{ "source": "npm", "package": "@acme-corp/claude-plugins" }
{ "source": "npm", "package": "@acme-corp/approved-marketplace" }
```

Bidang: `package` (diperlukan, mendukung paket yang diberi cakupan)

1. **Jalur file**:

```
{ "source": "file", "path": "/usr/local/share/claude/acme-marketplace.json" }
{ "source": "file", "path": "/opt/acme-corp/plugins/marketplace.json" }
```

Bidang: `path` (diperlukan: jalur absolut ke file marketplace.json)

1. **Jalur direktori**:

```
{ "source": "directory", "path": "/usr/local/share/claude/acme-plugins" }
{ "source": "directory", "path": "/opt/acme-corp/approved-marketplaces" }
```

Bidang: `path` (diperlukan: jalur absolut ke direktori yang berisi `.claude-plugin/marketplace.json`)

1. **Pencocokan pola host**:

```
{ "source": "hostPattern", "hostPattern": "^github\\.example\\.com$" }
{ "source": "hostPattern", "hostPattern": "^gitlab\\.internal\\.example\\.com$" }
```

Bidang: `hostPattern` (diperlukan: pola regex untuk mencocokkan terhadap host marketplace)

Gunakan pencocokan pola host saat Anda ingin memungkinkan semua marketplace dari host tertentu tanpa menghitung setiap repositori secara individual. Ini berguna untuk organisasi dengan server GitHub Enterprise atau GitLab internal di mana pengembang membuat marketplace mereka sendiri.

Ekstraksi host berdasarkan jenis sumber:

- `github`: selalu cocok dengan `github.com`
- `git`: mengekstrak nama host dari URL (mendukung format HTTPS dan SSH)
- `url`: mengekstrak nama host dari URL
- `npm`, `file`, `directory`: tidak didukung untuk pencocokan pola host

Contoh konfigurasi:

Contoh: izinkan marketplace tertentu saja:

```
{
  "strictKnownMarketplaces": [
    {
      "source": "github",
      "repo": "acme-corp/approved-plugins"
    },
    {
      "source": "github",
      "repo": "acme-corp/security-tools",
      "ref": "v2.0"
    },
    {
      "source": "url",
      "url": "https://plugins.example.com/marketplace.json"
    },
    {
      "source": "npm",
      "package": "@acme-corp/compliance-plugins"
    }
  ]
}
```

Contoh - Nonaktifkan semua penambahan marketplace:

```
{
  "strictKnownMarketplaces": []
}
```

Contoh: izinkan semua marketplace dari server git internal:

```
{
  "strictKnownMarketplaces": [
    {
      "source": "hostPattern",
      "hostPattern": "^github\\.example\\.com$"
    }
  ]
}
```

Persyaratan pencocokan tepat:

Sumber marketplace harus cocok **dengan tepat** agar penambahan pengguna diizinkan. Untuk sumber berbasis git (`github` dan `git`), ini termasuk semua bidang opsional:

- `repo` atau `url` harus cocok dengan tepat
- Bidang `ref` harus cocok dengan tepat (atau keduanya tidak terdefinisi)
- Bidang `path` harus cocok dengan tepat (atau keduanya tidak terdefinisi)

Contoh sumber yang **TIDAK cocok**:

```
// Ini adalah sumber BERBEDA:
{ "source": "github", "repo": "acme-corp/plugins" }
{ "source": "github", "repo": "acme-corp/plugins", "ref": "main" }

// Ini juga BERBEDA:
{ "source": "github", "repo": "acme-corp/plugins", "path": "marketplace" }
{ "source": "github", "repo": "acme-corp/plugins" }
```

Perbandingan dengan `extraKnownMarketplaces`:

Aspek	`strictKnownMarketplaces`	`extraKnownMarketplaces`
Tujuan	Penegakan kebijakan organisasi	Kenyamanan tim
File pengaturan	Hanya `managed-settings.json`	File pengaturan apa pun
Perilaku	Blokir penambahan yang tidak ada dalam daftar putih	Instal otomatis marketplace yang hilang

Aspek	`strictKnownMarketplaces`	`extraKnownMarketplaces`
Saat diterapkan	Sebelum operasi jaringan/sistem file	Setelah prompt kepercayaan pengguna
Dapat ditimpa	Tidak (preseden tertinggi)	Ya (oleh pengaturan prioritas lebih tinggi)
Format sumber	Objek sumber langsung	Marketplace bernama dengan sumber bersarang
Kasus penggunaan	Kepatuhan, pembatasan keamanan	Onboarding, standardisasi

Perbedaan format:

`strictKnownMarketplaces` menggunakan objek sumber langsung:

```
{
  "strictKnownMarketplaces": [
    { "source": "github", "repo": "acme-corp/plugins" }
  ]
}
```

`extraKnownMarketplaces` memerlukan marketplace bernama:

```
{
  "extraKnownMarketplaces": {
    "acme-tools": {
      "source": { "source": "github", "repo": "acme-corp/plugins" }
    }
  }
}
```

Catatan penting:

- Pembatasan diperiksa SEBELUM permintaan jaringan atau operasi sistem file apa pun
- Saat diblokir, pengguna melihat pesan kesalahan yang jelas yang menunjukkan sumber diblokir oleh kebijakan yang dikelola

- Pembatasan hanya berlaku untuk menambahkan marketplace BARU; marketplace yang sebelumnya diinstal tetap dapat diakses
- Pengaturan yang dikelola memiliki preseden tertinggi dan tidak dapat ditimpa

Lihat Pembatasan marketplace yang dikelola untuk dokumentasi yang menghadap pengguna.

Mengelola plugin

Gunakan perintah `/plugin` untuk mengelola plugin secara interaktif:

- Jelajahi plugin yang tersedia dari marketplace
- Instal/copot plugin
- Aktifkan/nonaktifkan plugin
- Lihat detail plugin (perintah, agent, hook yang disediakan)
- Tambah/hapus marketplace

Pelajari lebih lanjut tentang sistem plugin dalam dokumentasi plugin.

Variabel lingkungan

Claude Code mendukung variabel lingkungan berikut untuk mengontrol perilakunya:

> **Note:**
>
> Semua variabel lingkungan juga dapat dikonfigurasi dalam `settings.json`. Ini berguna sebagai cara untuk secara otomatis menetapkan variabel lingkungan untuk setiap sesi, atau untuk meluncurkan serangkaian variabel lingkungan untuk seluruh tim atau organisasi Anda.

Variabel	Tujuan	
`ANTHROPIC_API_KEY`	Kunci API dikirim sebagai header `X-Api-Key`, biasanya untuk SDK Claude (untuk penggunaan interaktif, jalankan `/login`)	
`ANTHROPIC_AUTH_TOKEN`	Nilai khusus untuk header `Authorization` (nilai yang Anda atur di sini akan diawali dengan `Bearer`)	

Variabel	Tujuan	
`ANTHROPIC_CUSTOM_HEADERS`	Header khusus untuk ditambahkan ke permintaan (format `Name: Value`, dipisahkan baris baru untuk beberapa header)	
`ANTHROPIC_DEFAULT_HAIKU_MODEL`	Lihat Konfigurasi Model	
`ANTHROPIC_DEFAULT_OPUS_MODEL`	Lihat Konfigurasi Model	
`ANTHROPIC_DEFAULT_SONNET_MODEL`	Lihat Konfigurasi Model	
`ANTHROPIC_FOUNDRY_API_KEY`	Kunci API untuk autentikasi Microsoft Foundry (lihat Microsoft Foundry)	
`ANTHROPIC_FOUNDRY_BASE_URL`	URL dasar lengkap untuk sumber daya Foundry (misalnya, `https://my-resource.services.ai.azure.com/anthropic`). Alternatif untuk `ANTHROPIC_FOUNDRY_RESOURCE` (lihat Microsoft Foundry)	
`ANTHROPIC_FOUNDRY_RESOURCE`	Nama sumber daya Foundry (misalnya, `my-resource`). Diperlukan jika `ANTHROPIC_FOUNDRY_BASE_URL` tidak diatur (lihat Microsoft Foundry)	
`ANTHROPIC_MODEL`	Nama pengaturan model untuk digunakan (lihat Konfigurasi Model)	
`ANTHROPIC_SMALL_FAST_MODEL`	[TIDAK DIREKOMENDASIKAN] Nama model kelas Haiku untuk tugas latar belakang	
`ANTHROPIC_SMALL_FAST_MODEL_AWS_REGION`	Timpa wilayah AWS untuk model kelas Haiku saat menggunakan Bedrock	
`AWS_BEARER_TOKEN_BEDROCK`	Kunci API Bedrock untuk autentikasi (lihat Kunci API Bedrock)	
`BASH_DEFAULT_TIMEOUT_MS`	Timeout default untuk perintah bash yang berjalan lama	

Variabel	Tujuan	
`BASH_MAX_OUTPUT_LENGTH`	Jumlah maksimum karakter dalam output bash sebelum dipotong di tengah	
`BASH_MAX_TIMEOUT_MS`	Timeout maksimum yang dapat ditetapkan model untuk perintah bash yang berjalan lama	
`CLAUDE_AUTOCOMPACT_PCT_OVERRIDE`	Atur persentase kapasitas konteks (1-100) di mana auto-compaction dipicu. Secara default, auto-compaction dipicu pada kapasitas sekitar 95%. Gunakan nilai lebih rendah seperti `50` untuk compact lebih awal. Nilai di atas ambang default tidak berpengaruh. Berlaku untuk percakapan utama dan subagent. Persentase ini selaras dengan bidang `context_window.used_percentage` yang tersedia dalam status line	
`CLAUDE_BASH_MAINTAIN_PROJECT_WORKING_DIR`	Kembali ke direktori kerja asli setelah setiap perintah Bash	
`CLAUDE_CODE_ACCOUNT_UUID`	UUID akun untuk pengguna yang terautentikasi. Digunakan oleh pemanggil SDK untuk memberikan informasi akun secara sinkron, menghindari kondisi balapan di mana acara telemetri awal kekurangan metadata akun. Memerlukan `CLAUDE_CODE_USER_EMAIL` dan `CLAUDE_CODE_ORGANIZATION_UUID` juga diatur	
`CLAUDE_CODE_ADDITIONAL_DIRECTORIES_CLAUDE_MD`	Atur ke `1` untuk memuat file CLAUDE.md dari direktori yang ditentukan dengan `--add-dir`. Secara default, direktori tambahan tidak memuat file memori	`1`

Variabel	Tujuan	
`CLAUDE_CODE_API_KEY_HELPER_TTL_MS`	Interval dalam milidetik di mana kredensial harus disegarkan (saat menggunakan `apiKeyHelper`)	
`CLAUDE_CODE_CLIENT_CERT`	Jalur ke file sertifikat klien untuk autentikasi mTLS	
`CLAUDE_CODE_CLIENT_KEY`	Jalur ke file kunci pribadi klien untuk autentikasi mTLS	
`CLAUDE_CODE_CLIENT_KEY_PASSPHRASE`	Frasa sandi untuk `CLAUDE_CODE_CLIENT_KEY` terenkripsi (opsional)	
`CLAUDE_CODE_DISABLE_1M_CONTEXT`	Atur ke `1` untuk menonaktifkan dukungan jendela konteks 1M. Saat diatur, varian model 1M tidak tersedia dalam pemilih model. Berguna untuk lingkungan perusahaan dengan persyaratan kepatuhan	
`CLAUDE_CODE_DISABLE_ADAPTIVE_THINKING`	Atur ke `1` untuk menonaktifkan penalaran adaptif untuk Opus 4.6 dan Sonnet 4.6. Saat dinonaktifkan, model ini kembali ke anggaran pemikiran tetap yang dikendalikan oleh `MAX_THINKING_TOKENS`	
`CLAUDE_CODE_DISABLE_AUTO_MEMORY`	Atur ke `1` untuk menonaktifkan auto memory. Atur ke `0` untuk memaksa auto memory selama peluncuran bertahap. Saat dinonaktifkan, Claude tidak membuat atau memuat file auto memory	
`CLAUDE_CODE_DISABLE_GIT_INSTRUCTIONS`	Atur ke `1` untuk menghapus instruksi alur kerja komit dan PR bawaan dari prompt sistem Claude. Berguna saat menggunakan skill alur kerja git Anda sendiri. Memiliki prioritas atas pengaturan `includeGitInstructions` saat diatur	

Variabel	Tujuan	
`CLAUDE_CODE_DISABLE_BACKGROUND_TASKS`	Atur ke `1` untuk menonaktifkan semua fungsionalitas tugas latar belakang, termasuk parameter `run_in_background` pada alat Bash dan subagent, auto-backgrounding, dan pintasan Ctrl+B	
`CLAUDE_CODE_DISABLE_CRON`	Atur ke `1` untuk menonaktifkan tugas terjadwal. Skill `/loop` dan alat cron menjadi tidak tersedia dan tugas yang sudah dijadwalkan berhenti berfungsi, termasuk tugas yang sudah berjalan di tengah sesi	
`CLAUDE_CODE_DISABLE_EXPERIMENTAL_BETAS`	Atur ke `1` untuk menonaktifkan header `anthropic-beta` spesifik API Anthropic. Gunakan ini jika mengalami masalah seperti “Unexpected value(s) for the `anthropic-beta` header” saat menggunakan gateway LLM dengan penyedia pihak ketiga	
`CLAUDE_CODE_DISABLE_FAST_MODE`	Atur ke `1` untuk menonaktifkan mode cepat	
`CLAUDE_CODE_DISABLE_FEEDBACK_SURVEY`	Atur ke `1` untuk menonaktifkan survei kualitas sesi “How is Claude doing?”. Juga dinonaktifkan saat menggunakan penyedia pihak ketiga atau saat telemetri dinonaktifkan. Lihat Survei kualitas sesi	
`CLAUDE_CODE_DISABLE_NONESSENTIAL_TRAFFIC`	Setara dengan pengaturan `DISABLE_AUTOUPDATER`, `DISABLE_BUG_COMMAND`, `DISABLE_ERROR_REPORTING`, dan `DISABLE_TELEMETRY`	
`CLAUDE_CODE_DISABLE_TERMINAL_TITLE`	Atur ke `1` untuk menonaktifkan pembaruan judul terminal otomatis berdasarkan konteks percakapan	

Variabel	Tujuan	
`CLAUDE_CODE_EFFORT_LEVEL`	Atur tingkat upaya untuk model yang didukung. Nilai: `low`, `medium`, `high`. Upaya lebih rendah lebih cepat dan lebih murah, upaya lebih tinggi memberikan penalaran lebih dalam. Didukung pada Opus 4.6 dan Sonnet 4.6. Lihat Sesuaikan tingkat upaya	
`CLAUDE_CODE_ENABLE_PROMPT_SUGGESTION`	Atur ke `false` untuk menonaktifkan saran prompt (toggle "Prompt suggestions" dalam `/config`). Ini adalah prediksi yang diarsir yang muncul dalam input prompt Anda setelah Claude merespons. Lihat Saran prompt	
`CLAUDE_CODE_ENABLE_TASKS`	Atur ke `false` untuk sementara kembali ke daftar TODO sebelumnya daripada sistem pelacakan tugas. Default: `true`. Lihat Daftar tugas	
`CLAUDE_CODE_ENABLE_TELEMETRY`	Atur ke `1` untuk mengaktifkan pengumpulan data OpenTelemetry untuk metrik dan logging. Diperlukan sebelum mengonfigurasi pengekspor OTel. Lihat Monitoring	
`CLAUDE_CODE_EXIT_AFTER_STOP_DELAY`	Waktu dalam milidetik untuk menunggu setelah loop kueri menjadi idle sebelum keluar secara otomatis. Berguna untuk alur kerja otomatis dan skrip menggunakan mode SDK	
`CLAUDE_CODE_EXPERIMENTAL_AGENT_TEAMS`	Atur ke `1` untuk mengaktifkan agent teams. Agent teams bersifat eksperimental dan dinonaktifkan secara default	
`CLAUDE_CODE_FILE_READ_MAX_OUTPUT_TOKENS`	Timpa batas token default untuk pembacaan file. Berguna saat Anda perlu membaca file yang lebih besar secara penuh	

Variabel	Tujuan	
`CLAUDE_CODE_HIDE_ACCOUNT_INFO`	Atur ke `1` untuk menyembunyikan alamat email dan nama organisasi Anda dari UI Claude Code. Berguna saat streaming atau merekam	
`CLAUDE_CODE_IDE_SKIP_AUTO_INSTALL`	Lewati instalasi otomatis ekstensi IDE	
`CLAUDE_CODE_MAX_OUTPUT_TOKENS`	Atur jumlah maksimum token output untuk sebagian besar permintaan. Default: 32,000. Maksimum: 64,000. Meningkatkan nilai ini mengurangi jendela konteks efektif yang tersedia sebelum auto-compaction dipicu.	
`CLAUDE_CODE_ORGANIZATION_UUID`	UUID organisasi untuk pengguna yang terautentikasi. Digunakan oleh pemanggil SDK untuk memberikan informasi akun secara sinkron. Memerlukan `CLAUDE_CODE_ACCOUNT_UUID` dan `CLAUDE_CODE_USER_EMAIL` juga diatur	
`CLAUDE_CODE_OTEL_HEADERS_HELPER_DEBOUNCE_MS`	Interval untuk menyegarkan header OpenTelemetry dinamis dalam milidetik (default: 1740000 / 29 menit). Lihat Header dinamis	
`CLAUDE_CODE_PLAN_MODE_REQUIRED`	Secara otomatis diatur ke `true` pada agent team rekan kerja yang memerlukan persetujuan rencana. Baca-saja: diatur oleh Claude Code saat menghasilkan rekan kerja. Lihat memerlukan persetujuan rencana	
`CLAUDE_CODE_PLUGIN_GIT_TIMEOUT_MS`	Timeout dalam milidetik untuk operasi git saat menginstal atau memperbarui plugin (default: 120000). Tingkatkan nilai ini untuk repositori besar atau koneksi jaringan lambat. Lihat Operasi Git habis waktu	

Variabel	Tujuan	
`CLAUDE_CODE_PROXY_RESOLVES_HOSTS`	Atur ke `true` untuk memungkinkan proxy melakukan resolusi DNS daripada pemanggil. Opt-in untuk lingkungan di mana proxy harus menangani resolusi nama host	
`CLAUDE_CODE_SHELL`	Timpa deteksi shell otomatis. Berguna saat shell login Anda berbeda dari shell kerja pilihan Anda (misalnya, `bash` vs `zsh`)	
`CLAUDE_CODE_SHELL_PREFIX`	Prefiks perintah untuk membungkus semua perintah bash (misalnya, untuk logging atau auditing). Contoh: `/path/to/logger.sh` akan mengeksekusi `/path/to/logger.sh <command>`	
`CLAUDE_CODE_SIMPLE`	Atur ke `1` untuk menjalankan dengan prompt sistem minimal dan hanya alat Bash, pembacaan file, dan pengeditan file. Menonaktifkan alat MCP, lampiran, hook, dan file CLAUDE.md	
`CLAUDE_CODE_SKIP_BEDROCK_AUTH`	Lewati autentikasi AWS untuk Bedrock (misalnya, saat menggunakan gateway LLM)	
`CLAUDE_CODE_SKIP_FOUNDRY_AUTH`	Lewati autentikasi Azure untuk Microsoft Foundry (misalnya, saat menggunakan gateway LLM)	
`CLAUDE_CODE_SKIP_VERTEX_AUTH`	Lewati autentikasi Google untuk Vertex (misalnya, saat menggunakan gateway LLM)	
`CLAUDE_CODE_SUBAGENT_MODEL`	Lihat Konfigurasi Model	
`CLAUDE_CODE_TASK_LIST_ID`	Bagikan daftar tugas di seluruh sesi. Atur ID yang sama dalam beberapa instans Claude Code untuk berkoordinasi pada daftar tugas bersama. Lihat Daftar tugas	

Variabel	Tujuan	
`CLAUDE_CODE_TEAM_NAME`	Nama agent team yang dimiliki rekan kerja ini. Diatur secara otomatis pada anggota agent team	
`CLAUDE_CODE_TMPDIR`	Timpa direktori temp yang digunakan untuk file temp internal. Claude Code menambahkan `/claude/` ke jalur ini. Default: `/tmp` pada Unix/macOS, `os.tmpdir()` pada Windows	
`CLAUDE_CODE_USER_EMAIL`	Alamat email untuk pengguna yang terautentikasi. Digunakan oleh pemanggil SDK untuk memberikan informasi akun secara sinkron. Memerlukan `CLAUDE_CODE_ACCOUNT_UUID` dan `CLAUDE_CODE_ORGANIZATION_UUID` juga diatur	
`CLAUDE_CODE_USE_BEDROCK`	Gunakan Bedrock	
`CLAUDE_CODE_USE_FOUNDRY`	Gunakan Microsoft Foundry	
`CLAUDE_CODE_USE_VERTEX`	Gunakan Vertex	
`CLAUDE_CONFIG_DIR`	Sesuaikan di mana Claude Code menyimpan file konfigurasi dan datanya	
`DISABLE_AUTOUPDATER`	Atur ke `1` untuk menonaktifkan pembaruan otomatis.	
`DISABLE_BUG_COMMAND`	Atur ke `1` untuk menonaktifkan perintah `/bug`	
`DISABLE_COST_WARNINGS`	Atur ke `1` untuk menonaktifkan pesan peringatan biaya	
`DISABLE_ERROR_REPORTING`	Atur ke `1` untuk keluar dari pelaporan kesalahan Sentry	
`DISABLE_INSTALLATION_CHECKS`	Atur ke `1` untuk menonaktifkan peringatan instalasi. Gunakan hanya saat mengelola lokasi instalasi secara manual, karena ini dapat menyembunyikan masalah dengan instalasi standar	

Variabel	Tujuan	
`DISABLE_NON_ESSENTIAL_MODEL_CALLS`	Atur ke `1` untuk menonaktifkan panggilan model untuk jalur non-kritis seperti teks flavor	
`DISABLE_PROMPT_CACHING`	Atur ke `1` untuk menonaktifkan prompt caching untuk semua model (memiliki prioritas atas pengaturan per-model)	
`DISABLE_PROMPT_CACHING_HAIKU`	Atur ke `1` untuk menonaktifkan prompt caching untuk model Haiku	
`DISABLE_PROMPT_CACHING_OPUS`	Atur ke `1` untuk menonaktifkan prompt caching untuk model Opus	
`DISABLE_PROMPT_CACHING_SONNET`	Atur ke `1` untuk menonaktifkan prompt caching untuk model Sonnet	
`DISABLE_TELEMETRY`	Atur ke `1` untuk keluar dari telemetri Statsig (perhatikan bahwa acara Statsig tidak menyertakan data pengguna seperti kode, jalur file, atau perintah bash)	
`ENABLE_CLAUDEAI_MCP_SERVERS`	Atur ke `false` untuk menonaktifkan MCP servers claude.ai dalam Claude Code. Diaktifkan secara default untuk pengguna yang masuk	
`ENABLE_TOOL_SEARCH`	Mengontrol pencarian alat MCP. Nilai: `auto` (default, mengaktifkan pada 10% konteks), `auto:N` (ambang batas khusus, misalnya `auto:5` untuk 5%), `true` (selalu aktif), `false` (dinonaktifkan)	
`FORCE_AUTOUPDATE_PLUGINS`	Atur ke `true` untuk memaksa pembaruan otomatis plugin bahkan saat auto-updater utama dinonaktifkan melalui `DISABLE_AUTOUPDATER`	
`HTTP_PROXY`	Tentukan server proxy HTTP untuk koneksi jaringan	

Variabel	Tujuan	
`HTTPS_PROXY`	Tentukan server proxy HTTPS untuk koneksi jaringan	
`IS_DEMO`	Atur ke `true` untuk mengaktifkan mode demo: menyembunyikan email dan organisasi dari UI, melewati onboarding, dan menyembunyikan perintah internal. Berguna untuk streaming atau merekam sesi	
`MAX_MCP_OUTPUT_TOKENS`	Jumlah maksimum token yang diizinkan dalam respons alat MCP. Claude Code menampilkan peringatan saat output melebihi 10,000 token (default: 25000)	
`MAX_THINKING_TOKENS`	Timpa anggaran token extended thinking. Pemikiran diaktifkan pada anggaran maksimum (31,999 token) secara default. Gunakan ini untuk membatasi anggaran (misalnya, `MAX_THINKING_TOKENS=10000`) atau menonaktifkan pemikiran sepenuhnya (`MAX_THINKING_TOKENS=0`). Untuk Opus 4.6, kedalaman pemikiran dikendalikan oleh tingkat upaya sebagai gantinya, dan variabel ini diabaikan kecuali diatur ke `0` untuk menonaktifkan pemikiran.	
`MCP_CLIENT_SECRET`	Rahasia klien OAuth untuk MCP servers yang memerlukan kredensial yang dikonfigurasi sebelumnya. Menghindari prompt interaktif saat menambahkan server dengan `--client-secret`	
`MCP_OAUTH_CALLBACK_PORT`	Port tetap untuk callback pengalihan OAuth, sebagai alternatif untuk `--callback-port` saat menambahkan MCP server dengan kredensial yang dikonfigurasi sebelumnya	

Variabel	Tujuan	
`MCP_TIMEOUT`	Timeout dalam milidetik untuk startup server MCP	
`MCP_TOOL_TIMEOUT`	Timeout dalam milidetik untuk eksekusi alat MCP	
`NO_PROXY`	Daftar domain dan IP ke mana permintaan akan dikeluarkan secara langsung, melewati proxy	
`SLASH_COMMAND_TOOL_CHAR_BUDGET`	Timpa anggaran karakter untuk metadata skill yang ditampilkan ke alat Skill. Anggaran diskalakan secara dinamis pada 2% dari jendela konteks, dengan fallback 16,000 karakter. Nama warisan disimpan untuk kompatibilitas mundur	
`USE_BUILTIN_RIPGREP`	Atur ke `0` untuk menggunakan `rg` yang diinstal sistem daripada `rg` yang disertakan dengan Claude Code	
`VERTEX_REGION_CLAUDE_3_5_HAIKU`	Timpa wilayah untuk Claude 3.5 Haiku saat menggunakan Vertex AI	
`VERTEX_REGION_CLAUDE_3_7_SONNET`	Timpa wilayah untuk Claude 3.7 Sonnet saat menggunakan Vertex AI	
`VERTEX_REGION_CLAUDE_4_0_OPUS`	Timpa wilayah untuk Claude 4.0 Opus saat menggunakan Vertex AI	
`VERTEX_REGION_CLAUDE_4_0_SONNET`	Timpa wilayah untuk Claude 4.0 Sonnet saat menggunakan Vertex AI	
`VERTEX_REGION_CLAUDE_4_1_OPUS`	Timpa wilayah untuk Claude 4.1 Opus saat menggunakan Vertex AI	

Alat yang tersedia untuk Claude

Claude Code memiliki akses ke serangkaian alat canggih yang membantu memahami dan memodifikasi basis kode Anda:

Alat	Deskripsi	Izin Diperlukan
Agent	Menghasilkan subagent dengan jendela konteks sendiri untuk menangani tugas	Tidak
AskUserQuestion	Mengajukan pertanyaan pilihan ganda untuk mengumpulkan persyaratan atau mengklarifikasi ambiguitas	Tidak
Bash	Mengeksekusi perintah shell di lingkungan Anda. Lihat Perilaku alat Bash	Ya
CronCreate	Menjadwalkan prompt berulang atau satu kali dalam sesi saat ini (hilang saat Claude keluar). Lihat tugas terjadwal	Tidak
CronDelete	Membatalkan tugas terjadwal berdasarkan ID	Tidak
CronList	Mencantumkan semua tugas terjadwal dalam sesi	Tidak
Edit	Membuat pengeditan bertarget ke file tertentu	Ya
EnterPlanMode	Beralih ke mode rencana untuk merancang pendekatan sebelum coding	Tidak
EnterWorktree	Membuat git worktree terisolasi dan beralih ke dalamnya	Tidak
ExitPlanMode	Menyajikan rencana untuk persetujuan dan keluar dari mode rencana	Ya
ExitWorktree	Keluar dari sesi worktree dan kembali ke direktori asli	Tidak
Glob	Menemukan file berdasarkan pencocokan pola	Tidak
Grep	Mencari pola dalam konten file	Tidak

Alat	Deskripsi	Izin Diperlukan
ListMcpResourcesTool	Mencantumkan sumber daya yang diekspos oleh MCP servers yang terhubung	Tidak
LSP	Intelijen kode melalui server bahasa. Melaporkan kesalahan tipe dan peringatan secara otomatis setelah pengeditan file. Juga mendukung operasi navigasi: lompat ke definisi, temukan referensi, dapatkan info tipe, daftar simbol, temukan implementasi, lacak hierarki panggilan. Memerlukan plugin intelijen kode dan biner server bahasanya	Tidak
NotebookEdit	Memodifikasi sel notebook Jupyter	Ya
Read	Membaca konten file	Tidak
ReadMcpResourceTool	Membaca sumber daya MCP tertentu berdasarkan URI	Tidak
Skill	Mengeksekusi skill dalam percakapan utama	Ya
TaskCreate	Membuat tugas baru dalam daftar tugas	Tidak
TaskGet	Mengambil detail lengkap untuk tugas tertentu	Tidak
TaskList	Mencantumkan semua tugas dengan status saat ini mereka	Tidak
TaskOutput	Mengambil output dari tugas latar belakang	Tidak
TaskStop	Membunuh tugas latar belakang yang sedang berjalan berdasarkan ID	Tidak
TaskUpdate	Memperbarui status tugas, dependensi, detail, atau menghapus tugas	Tidak

Alat	Deskripsi	Izin Diperlukan
TodoWrite	Mengelola daftar tugas sesi. Tersedia dalam mode non-interaktif dan Agent SDK; sesi interaktif menggunakan TaskCreate, TaskGet, TaskList, dan TaskUpdate sebagai gantinya	Tidak
ToolSearch	Mencari dan memuat alat yang ditangguhkan saat pencarian alat diaktifkan	Tidak
WebFetch	Mengambil konten dari URL yang ditentukan	Ya
WebSearch	Melakukan pencarian web	Ya
Write	Membuat atau menimpa file	Ya

Aturan izin dapat dikonfigurasi menggunakan `/allowed-tools` atau dalam pengaturan izin. Lihat juga Aturan izin spesifik alat.

Perilaku alat Bash

Alat Bash mengeksekusi perintah shell dengan perilaku persistensi berikut:

- **Direktori kerja bertahan**: Ketika Claude mengubah direktori kerja (misalnya, `cd /path/to/dir`), perintah Bash berikutnya akan dieksekusi di direktori itu. Anda dapat menggunakan `CLAUDE_BASH_MAINTAIN_PROJECT_WORKING_DIR=1` untuk kembali ke direktori proyek setelah setiap perintah.
- **Variabel lingkungan TIDAK bertahan**: Variabel lingkungan yang ditetapkan dalam satu perintah Bash (misalnya, `export MY_VAR=value`) **tidak** tersedia dalam perintah Bash berikutnya. Setiap perintah Bash berjalan di lingkungan shell yang segar.

Untuk membuat variabel lingkungan tersedia dalam perintah Bash, Anda memiliki **tiga opsi**:

Opsi 1: Aktifkan lingkungan sebelum memulai Claude Code (pendekatan paling sederhana)

Aktifkan lingkungan virtual Anda di terminal sebelum meluncurkan Claude Code:

```
conda activate myenv
## atau: source /path/to/venv/bin/activate
claude
```

Ini berfungsi untuk lingkungan shell tetapi variabel lingkungan yang ditetapkan dalam perintah Bash Claude tidak akan bertahan di antara perintah.

Opsi 2: Atur CLAUDE_ENV_FILE sebelum memulai Claude Code (pengaturan lingkungan yang persisten)

Ekspor jalur ke skrip shell yang berisi pengaturan lingkungan Anda:

```
export CLAUDE_ENV_FILE=/path/to/env-setup.sh
claude
```

Di mana `/path/to/env-setup.sh` berisi:

```
conda activate myenv
## atau: source /path/to/venv/bin/activate
## atau: export MY_VAR=value
```

Claude Code akan membuat sumber file ini sebelum setiap perintah Bash, membuat lingkungan persisten di semua perintah.

Opsi 3: Gunakan hook SessionStart (konfigurasi spesifik proyek)

Konfigurasikan dalam `.claude/settings.json` :

```
{
  "hooks": {
    "SessionStart": [{
      "matcher": "startup",
      "hooks": [{
        "type": "command",
        "command": "echo 'conda activate myenv' >> \"$CLAUDE_ENV_FILE\""
      }]
    }]
  }
}
```

Hook menulis ke `$CLAUDE_ENV_FILE`, yang kemudian dibuat sumber sebelum setiap perintah Bash. Ini ideal untuk konfigurasi proyek bersama tim.

Lihat hook SessionStart untuk detail lebih lanjut tentang Opsi 3.

Memperluas alat dengan hooks

Anda dapat menjalankan perintah khusus sebelum atau sesudah alat apa pun dieksekusi menggunakan hook Claude Code.

Misalnya, Anda dapat secara otomatis menjalankan formatter Python setelah Claude memodifikasi file Python, atau mencegah modifikasi file konfigurasi produksi dengan memblokir operasi Write ke jalur tertentu.

Lihat juga

- Permissions: sistem izin, sintaks aturan, pola spesifik alat, dan kebijakan yang dikelola
- Authentication: atur akses pengguna ke Claude Code
- Troubleshooting: solusi untuk masalah konfigurasi umum

Konfigurasi model

> *Pelajari tentang konfigurasi model Claude Code, termasuk alias model seperti* `opusplan`

Model yang tersedia

Untuk pengaturan `model` di Claude Code, Anda dapat mengonfigurasi salah satu dari:

- Sebuah **alias model**
- Sebuah **nama model**
  - Anthropic API: Sebuah **nama model** lengkap
  - Bedrock: ARN profil inferensi
  - Foundry: nama deployment
  - Vertex: nama versi

Alias model

Alias model menyediakan cara yang nyaman untuk memilih pengaturan model tanpa perlu mengingat nomor versi yang tepat:

Alias model	Perilaku
`default`	Pengaturan model yang direkomendasikan, tergantung pada jenis akun Anda
`sonnet`	Menggunakan model Sonnet terbaru (saat ini Sonnet 4.6) untuk tugas coding sehari-hari
`opus`	Menggunakan model Opus terbaru (saat ini Opus 4.6) untuk tugas penalaran kompleks
`haiku`	Menggunakan model Haiku yang cepat dan efisien untuk tugas sederhana
`sonnet[1m]`	Menggunakan Sonnet dengan jendela konteks 1 juta token untuk sesi panjang
`opusplan`	Mode khusus yang menggunakan `opus` selama plan mode, kemudian beralih ke `sonnet` untuk eksekusi

Alias selalu menunjuk ke versi terbaru. Untuk menetapkan ke versi tertentu, gunakan nama model lengkap (misalnya, `claude-opus-4-6`) atau atur variabel lingkungan yang sesuai seperti `ANTHROPIC_DEFAULT_OPUS_MODEL`.

Mengatur model Anda

Anda dapat mengonfigurasi model Anda dengan beberapa cara, yang tercantum dalam urutan prioritas:

1. **Selama sesi** - Gunakan `/model <alias|name>` untuk beralih model di tengah sesi
2. **Saat startup** - Luncurkan dengan `claude --model <alias|name>`
3. **Variabel lingkungan** - Atur `ANTHROPIC_MODEL=<alias|name>`
4. **Pengaturan** - Konfigurasi secara permanen di file pengaturan Anda menggunakan bidang `model`.

Contoh penggunaan:

```
## Mulai dengan Opus
claude --model opus

## Beralih ke Sonnet selama sesi
/model sonnet
```

Contoh file pengaturan:

```
{
    "permissions": {
        ...
    },
    "model": "opus"
}
```

Batasi pemilihan model

Administrator enterprise dapat menggunakan `availableModels` dalam pengaturan terkelola atau kebijakan untuk membatasi model mana yang dapat dipilih pengguna.

Ketika `availableModels` diatur, pengguna tidak dapat beralih ke model yang tidak ada dalam daftar melalui `/model`, flag `--model`, alat Config, atau variabel lingkungan `ANTHROPIC_MODEL`.

```
{
  "availableModels": ["sonnet", "haiku"]
}
```

Perilaku model default

Opsi Default di pemilih model tidak dipengaruhi oleh `availableModels`. Opsi ini selalu tetap tersedia dan mewakili default runtime sistem berdasarkan tingkat langganan pengguna.

Bahkan dengan `availableModels: []`, pengguna masih dapat menggunakan Claude Code dengan model Default untuk tingkat mereka.

Kontrol model yang dijalankan pengguna

Untuk sepenuhnya mengontrol pengalaman model, gunakan `availableModels` bersama dengan pengaturan `model`:

- **availableModels**: membatasi apa yang dapat dialihkan pengguna
- **model**: menetapkan penggantian model eksplisit, mengambil alih dari Default

Contoh ini memastikan semua pengguna menjalankan Sonnet 4.6 dan hanya dapat memilih antara Sonnet dan Haiku:

```
{
  "model": "sonnet",
  "availableModels": ["sonnet", "haiku"]
}
```

Perilaku penggabungan

Ketika `availableModels` diatur di beberapa tingkat, seperti pengaturan pengguna dan pengaturan proyek, array digabungkan dan dideduplikasi. Untuk memberlakukan allowlist ketat, atur `availableModels` dalam pengaturan terkelola atau kebijakan yang memiliki prioritas tertinggi.

Perilaku model khusus

Pengaturan model `default`

Perilaku `default` tergantung pada jenis akun Anda:

- **Max dan Team Premium**: default ke Opus 4.6

- **Pro dan Team Standard**: default ke Sonnet 4.6
- **Enterprise**: Opus 4.6 tersedia tetapi bukan default

Claude Code dapat secara otomatis kembali ke Sonnet jika Anda mencapai ambang penggunaan dengan Opus.

Pengaturan model `opusplan`

Alias model `opusplan` menyediakan pendekatan hibrida otomatis:

- **Dalam plan mode** - Menggunakan `opus` untuk penalaran kompleks dan keputusan arsitektur
- **Dalam execution mode** - Secara otomatis beralih ke `sonnet` untuk pembuatan kode dan implementasi

Ini memberi Anda yang terbaik dari kedua dunia: penalaran superior Opus untuk perencanaan, dan efisiensi Sonnet untuk eksekusi.

Sesuaikan tingkat usaha

Tingkat usaha mengontrol penalaran adaptif, yang secara dinamis mengalokasikan pemikiran berdasarkan kompleksitas tugas. Usaha lebih rendah lebih cepat dan lebih murah untuk tugas-tugas langsung, sementara usaha lebih tinggi memberikan penalaran lebih dalam untuk masalah kompleks.

Tiga tingkat tersedia: **low**, **medium**, dan **high**. Opus 4.6 default ke medium effort untuk pelanggan Max dan Team.

Mengatur usaha:

- **Dalam `/model`**: gunakan tombol panah kiri/kanan untuk menyesuaikan slider usaha saat memilih model
- **Variabel lingkungan**: atur `CLAUDE_CODE_EFFORT_LEVEL=low|medium|high`
- **Pengaturan**: atur `effortLevel` di file pengaturan Anda

Usaha didukung pada Opus 4.6 dan Sonnet 4.6. Slider usaha muncul dalam `/model` ketika model yang didukung dipilih. Tingkat usaha saat ini juga ditampilkan di sebelah logo dan spinner (misalnya, “with low effort”), sehingga Anda dapat mengkonfirmasi pengaturan mana yang aktif tanpa membuka `/model`.

Untuk menonaktifkan penalaran adaptif pada Opus 4.6 dan Sonnet 4.6 dan kembali ke anggaran pemikiran tetap sebelumnya, atur `CLAUDE_CODE_DISABLE_ADAPTIVE_THINKING=1`. Ketika dinonaktifkan, model ini menggunakan anggaran tetap yang dikendalikan oleh `MAX_THINKING_TOKENS`. Lihat variabel lingkungan.

Konteks diperluas

Opus 4.6 dan Sonnet 4.6 mendukung jendela konteks 1 juta token untuk sesi panjang dengan basis kode besar.

> **Note:**
>
> Jendela konteks 1M saat ini dalam beta. Fitur, harga, dan ketersediaan dapat berubah.

Konteks diperluas tersedia untuk:

- **Pengguna API dan pay-as-you-go**: akses penuh ke konteks 1M
- **Pelanggan Pro, Max, Teams, dan Enterprise**: tersedia dengan penggunaan ekstra diaktifkan

Untuk menonaktifkan konteks 1M sepenuhnya, atur `CLAUDE_CODE_DISABLE_1M_CONTEXT=1`. Ini menghapus varian model 1M dari pemilih model. Lihat variabel lingkungan.

Memilih model 1M tidak segera mengubah penagihan. Sesi Anda menggunakan tarif standar hingga melebihi 200K token konteks. Melampaui 200K token, permintaan dikenakan biaya pada harga konteks panjang dengan batas laju khusus. Untuk pelanggan, token melampaui 200K ditagih sebagai penggunaan ekstra daripada melalui langganan.

Jika akun Anda mendukung konteks 1M, opsi muncul di pemilih model (`/model`) dalam versi terbaru Claude Code. Jika Anda tidak melihatnya, coba mulai ulang sesi Anda.

Anda juga dapat menggunakan akhiran `[1m]` dengan alias model atau nama model lengkap:

```
## Gunakan alias sonnet[1m]
/model sonnet[1m]

## Atau tambahkan [1m] ke nama model lengkap
/model claude-sonnet-4-6[1m]
```

Memeriksa model Anda saat ini

Anda dapat melihat model mana yang sedang Anda gunakan dengan beberapa cara:

1. Dalam baris status (jika dikonfigurasi)
2. Dalam `/status`, yang juga menampilkan informasi akun Anda.

Variabel lingkungan

Anda dapat menggunakan variabel lingkungan berikut, yang harus berupa **nama model** lengkap (atau setara untuk penyedia API Anda), untuk mengontrol nama model yang dipetakan alias.

Variabel lingkungan	Deskripsi
`ANTHROPIC_DEFAULT_OPUS_MODEL`	Model yang digunakan untuk `opus`, atau untuk `opusplan` ketika Plan Mode aktif.
`ANTHROPIC_DEFAULT_SONNET_MODEL`	Model yang digunakan untuk `sonnet`, atau untuk `opusplan` ketika Plan Mode tidak aktif.
`ANTHROPIC_DEFAULT_HAIKU_MODEL`	Model yang digunakan untuk `haiku`, atau fungsionalitas latar belakang
`CLAUDE_CODE_SUBAGENT_MODEL`	Model yang digunakan untuk subagents

Catatan: `ANTHROPIC_SMALL_FAST_MODEL` sudah usang dan digantikan oleh `ANTHROPIC_DEFAULT_HAIKU_MODEL`.

Tetapkan model untuk deployment pihak ketiga

Saat menerapkan Claude Code melalui Bedrock, Vertex AI, atau Foundry, tetapkan versi model sebelum meluncurkan ke pengguna.

Tanpa penetapan, Claude Code menggunakan alias model (`sonnet`, `opus`, `haiku`) yang diselesaikan ke versi terbaru. Ketika Anthropic merilis model baru, pengguna yang akunnya tidak memiliki versi baru yang diaktifkan akan rusak secara diam-diam.

> **Warning:**
>
> Atur ketiga variabel lingkungan model ke ID versi spesifik sebagai bagian dari pengaturan awal Anda. Melewatkan langkah ini berarti pembaruan Claude Code dapat merusak pengguna Anda tanpa tindakan apa pun dari pihak Anda.

Gunakan variabel lingkungan berikut dengan ID model spesifik versi untuk penyedia Anda:

Penyedia	Contoh
Bedrock	`export ANTHROPIC_DEFAULT_OPUS_MODEL='us.anthropic.claude-opus-4-6-v1'`
Vertex AI	`export ANTHROPIC_DEFAULT_OPUS_MODEL='claude-opus-4-6'`
Foundry	`export ANTHROPIC_DEFAULT_OPUS_MODEL='claude-opus-4-6'`

Terapkan pola yang sama untuk `ANTHROPIC_DEFAULT_SONNET_MODEL` dan `ANTHROPIC_DEFAULT_HAIKU_MODEL`. Untuk ID model saat ini dan warisan di semua penyedia, lihat Ikhtisar Model. Untuk meningkatkan pengguna ke versi model baru, perbarui variabel lingkungan ini dan terapkan kembali.

> **Note:**
>
> Allowlist `settings.availableModels` masih berlaku saat menggunakan penyedia pihak ketiga. Penyaringan cocok pada alias model (`opus` , `sonnet` , `haiku`), bukan ID model spesifik penyedia.

Ganti ID model per versi

Variabel lingkungan tingkat keluarga di atas mengonfigurasi satu ID model per alias keluarga. Jika Anda perlu memetakan beberapa versi dalam keluarga yang sama ke ID penyedia yang berbeda, gunakan pengaturan `modelOverrides` sebagai gantinya.

`modelOverrides` memetakan ID model Anthropic individual ke string spesifik penyedia yang dikirim Claude Code ke API penyedia Anda. Ketika pengguna memilih model yang dipetakan di pemilih `/model`, Claude Code menggunakan nilai yang dikonfigurasi Anda alih-alih default bawaan.

Ini memungkinkan administrator enterprise untuk merutekan setiap versi model ke ARN profil inferensi Bedrock spesifik, nama versi Vertex AI, atau nama deployment Foundry untuk tata kelola, alokasi biaya, atau perutean regional.

Atur `modelOverrides` dalam file pengaturan Anda:

```
{
  "modelOverrides": {
    "claude-opus-4-6": "arn:aws:bedrock:us-east-2:123456789012:application-inference-profile/opus-prod",
    "claude-opus-4-5-20251101": "arn:aws:bedrock:us-east-2:123456789012:application-inference-profile/opus-45-prod",
    "claude-sonnet-4-6": "arn:aws:bedrock:us-east-2:123456789012:application-inference-profile/sonnet-prod"
  }
}
```

Kunci harus berupa ID model Anthropic seperti yang tercantum dalam Ikhtisar Model. Untuk ID model bertanggal, sertakan akhiran tanggal persis seperti yang muncul di sana. Kunci yang tidak dikenal diabaikan.

Penggantian menggantikan ID model bawaan yang mendukung setiap entri di pemilih `/model`. Di Bedrock, penggantian mengambil alih profil inferensi apa pun yang ditemukan Claude Code secara otomatis saat startup. Nilai yang Anda berikan langsung melalui `ANTHROPIC_MODEL`, `--model`, atau variabel lingkungan `ANTHROPIC_DEFAULT_*_MODEL` diteruskan ke penyedia apa adanya dan tidak diubah oleh `modelOverrides`.

`modelOverrides` bekerja bersama `availableModels`. Allowlist dievaluasi terhadap ID model Anthropic, bukan nilai penggantian, jadi entri seperti `"opus"` dalam `availableModels` terus cocok bahkan ketika versi Opus dipetakan ke ARN.

Konfigurasi prompt caching

Claude Code secara otomatis menggunakan prompt caching untuk mengoptimalkan kinerja dan mengurangi biaya. Anda dapat menonaktifkan prompt caching secara global atau untuk tingkat model tertentu:

Variabel lingkungan	Deskripsi
`DISABLE_PROMPT_CACHING`	Atur ke `1` untuk menonaktifkan prompt caching untuk semua model (mengambil alih pengaturan per-model)
`DISABLE_PROMPT_CACHING_HAIKU`	Atur ke `1` untuk menonaktifkan prompt caching hanya untuk model Haiku
`DISABLE_PROMPT_CACHING_SONNET`	Atur ke `1` untuk menonaktifkan prompt caching hanya untuk model Sonnet

Variabel lingkungan	Deskripsi
`DISABLE_PROMPT_CACHING_OPUS`	Atur ke `1` untuk menonaktifkan prompt caching hanya untuk model Opus

Variabel lingkungan ini memberi Anda kontrol terperinci atas perilaku prompt caching. Pengaturan global `DISABLE_PROMPT_CACHING` mengambil alih pengaturan spesifik model, memungkinkan Anda dengan cepat menonaktifkan semua caching saat diperlukan. Pengaturan per-model berguna untuk kontrol selektif, seperti saat men-debug model tertentu atau bekerja dengan penyedia cloud yang mungkin memiliki implementasi caching yang berbeda.

Output styles

> *Sesuaikan Claude Code untuk penggunaan di luar rekayasa perangkat lunak*

Output styles memungkinkan Anda menggunakan Claude Code sebagai jenis agen apa pun sambil mempertahankan kemampuan intinya, seperti menjalankan skrip lokal, membaca/menulis file, dan melacak TODOs.

Built-in output styles

Default output style Claude Code adalah system prompt yang ada, dirancang untuk membantu Anda menyelesaikan tugas-tugas rekayasa perangkat lunak secara efisien.

Ada dua built-in output styles tambahan yang berfokus pada pengajaran Anda tentang codebase dan cara Claude beroperasi:

- **Explanatory**: Menyediakan "Insights" edukatif di antara membantu Anda menyelesaikan tugas-tugas rekayasa perangkat lunak. Membantu Anda memahami pilihan implementasi dan pola codebase.
- **Learning**: Mode kolaboratif belajar-dengan-melakukan di mana Claude tidak hanya akan berbagi "Insights" saat coding, tetapi juga meminta Anda untuk berkontribusi dengan potongan kode kecil dan strategis sendiri. Claude Code akan menambahkan penanda `TODO(human)` dalam kode Anda untuk Anda implementasikan.

Cara kerja output styles

Output styles secara langsung memodifikasi system prompt Claude Code.

- Semua output styles mengecualikan instruksi untuk output yang efisien (seperti merespons secara ringkas).
- Custom output styles mengecualikan instruksi untuk coding (seperti memverifikasi kode dengan tes), kecuali `keep-coding-instructions` bernilai true.
- Semua output styles memiliki instruksi kustom mereka sendiri yang ditambahkan ke akhir system prompt.
- Semua output styles memicu pengingat bagi Claude untuk mematuhi instruksi output style selama percakapan.

Ubah output style Anda

Jalankan `/config` dan pilih **Output style** untuk memilih style dari menu. Pilihan Anda disimpan ke `.claude/settings.local.json` di tingkat proyek lokal.

Untuk menetapkan style tanpa menu, edit field `outputStyle` secara langsung dalam file settings:

```
{
  "outputStyle": "Explanatory"
}
```

Karena output style ditetapkan dalam system prompt saat awal sesi, perubahan berlaku saat Anda memulai sesi baru. Ini menjaga system prompt tetap stabil sepanjang percakapan sehingga prompt caching dapat mengurangi latensi dan biaya.

Buat custom output style

Custom output styles adalah file Markdown dengan frontmatter dan teks yang akan ditambahkan ke system prompt:

```
---
name: My Custom Style
description:
  A brief description of what this style does, to be displayed to the user
---

## Custom Style Instructions

You are an interactive CLI tool that helps users with software engineering
tasks. [Your custom instructions here...]

### Specific Behaviors

[Define how the assistant should behave in this style...]
```

Anda dapat menyimpan file-file ini di tingkat pengguna (`~/.claude/output-styles`) atau tingkat proyek (`.claude/output-styles`).

Frontmatter

File output style mendukung frontmatter untuk menentukan metadata:

Frontmatter	Tujuan	Default
`name`	Nama output style, jika bukan nama file	Mewarisi dari nama file
`description`	Deskripsi output style, ditampilkan dalam picker `/config`	Tidak ada
`keep-coding-instructions`	Apakah akan mempertahankan bagian-bagian dari system prompt Claude Code yang terkait dengan coding.	false

Perbandingan dengan fitur terkait

Output Styles vs. CLAUDE.md vs. –append-system-prompt

Output styles sepenuhnya "mematikan" bagian-bagian dari default system prompt Claude Code yang spesifik untuk rekayasa perangkat lunak. Baik CLAUDE.md maupun `--append-system-prompt` tidak mengedit default system prompt Claude Code. CLAUDE.md menambahkan konten sebagai pesan pengguna *setelah* default system prompt Claude Code. `--append-system-prompt` menambahkan konten ke system prompt.

Output Styles vs. Agents

Output styles secara langsung mempengaruhi loop agen utama dan hanya mempengaruhi system prompt. Agents dipanggil untuk menangani tugas-tugas spesifik dan dapat mencakup pengaturan tambahan seperti model yang akan digunakan, tools yang tersedia bagi mereka, dan beberapa konteks tentang kapan menggunakan agent.

Output Styles vs. Skills

Output styles memodifikasi cara Claude merespons (pemformatan, nada, struktur) dan selalu aktif setelah dipilih. Skills adalah prompts khusus tugas yang Anda panggil dengan `/skill-name` atau yang Claude muat secara otomatis saat relevan. Gunakan output styles untuk preferensi pemformatan yang konsisten; gunakan skills untuk alur kerja dan tugas yang dapat digunakan kembali.

Percepat respons dengan mode cepat

> *Dapatkan respons Opus 4.6 yang lebih cepat di Claude Code dengan mengaktifkan mode cepat.*

Note:

Mode cepat berada dalam pratinjau penelitian. Fitur, harga, dan ketersediaan dapat berubah berdasarkan umpan balik.

Mode cepat adalah konfigurasi kecepatan tinggi untuk Claude Opus 4.6, membuat model 2,5x lebih cepat dengan biaya per token yang lebih tinggi. Aktifkan dengan `/fast` ketika Anda membutuhkan kecepatan untuk pekerjaan interaktif seperti iterasi cepat atau debugging langsung, dan nonaktifkan ketika biaya lebih penting daripada latensi.

Mode cepat bukan model yang berbeda. Mode ini menggunakan Opus 4.6 yang sama dengan konfigurasi API berbeda yang memprioritaskan kecepatan daripada efisiensi biaya. Anda mendapatkan kualitas dan kemampuan yang identik, hanya respons yang lebih cepat.

Note:

Mode cepat memerlukan Claude Code v2.1.36 atau lebih baru. Periksa versi Anda dengan `claude --version`.

Yang perlu diketahui:

- Gunakan `/fast` untuk mengaktifkan mode cepat di Claude Code CLI. Juga tersedia melalui `/fast` di Ekstensi Claude Code VS Code.
- Harga mode cepat untuk Opus 4.6 dimulai dari $30/150 MTok. Mode cepat tersedia dengan diskon 50% untuk semua paket hingga 23:59 PT pada 16 Februari.
- Tersedia untuk semua pengguna Claude Code pada paket berlangganan (Pro/Max/Team/Enterprise) dan Claude Console.
- Untuk pengguna Claude Code pada paket berlangganan (Pro/Max/Team/Enterprise), mode cepat tersedia hanya melalui penggunaan tambahan dan tidak termasuk dalam batas laju penggunaan berlangganan.

Halaman ini mencakup cara mengaktifkan mode cepat, pertukaran biayanya, kapan menggunakannya, persyaratan, opt-in per sesi, dan perilaku batas laju.

Aktifkan mode cepat

Aktifkan mode cepat dengan salah satu cara berikut:

- Ketik `/fast` dan tekan Tab untuk mengaktifkan atau menonaktifkan
- Atur `"fastMode": true` di file pengaturan pengguna Anda

Secara default, mode cepat bertahan di seluruh sesi. Administrator dapat mengonfigurasi mode cepat untuk disetel ulang setiap sesi. Lihat require per-session opt-in untuk detail.

Untuk efisiensi biaya terbaik, aktifkan mode cepat di awal sesi daripada beralih di tengah percakapan. Lihat understand the cost tradeoff untuk detail.

Ketika Anda mengaktifkan mode cepat:

- Jika Anda berada di model yang berbeda, Claude Code secara otomatis beralih ke Opus 4.6
- Anda akan melihat pesan konfirmasi: “Fast mode ON”
- Ikon kecil `↯` muncul di sebelah prompt saat mode cepat aktif
- Jalankan `/fast` lagi kapan saja untuk memeriksa apakah mode cepat aktif atau tidak

Ketika Anda menonaktifkan mode cepat dengan `/fast` lagi, Anda tetap berada di Opus 4.6. Model tidak kembali ke model sebelumnya. Untuk beralih ke model yang berbeda, gunakan `/model`.

Pahami pertukaran biaya

Mode cepat memiliki harga per-token yang lebih tinggi daripada Opus 4.6 standar:

Mode	Input (MTok)	Output (MTok)
Mode cepat di Opus 4.6 (<200K)	$30	$150
Mode cepat di Opus 4.6 (>200K)	$60	$225

Mode cepat kompatibel dengan jendela konteks yang diperluas 1M token.

Ketika Anda beralih ke mode cepat di tengah percakapan, Anda membayar harga token input mode cepat tanpa cache penuh untuk seluruh konteks percakapan. Ini lebih mahal daripada jika Anda telah mengaktifkan mode cepat dari awal.

Tentukan kapan menggunakan mode cepat

Mode cepat terbaik untuk pekerjaan interaktif di mana latensi respons lebih penting daripada biaya:

- Iterasi cepat pada perubahan kode
- Sesi debugging langsung
- Pekerjaan sensitif waktu dengan tenggat waktu ketat

Mode standar lebih baik untuk:

- Tugas otonom jangka panjang di mana kecepatan kurang penting
- Pemrosesan batch atau pipeline CI/CD
- Beban kerja sensitif biaya

Mode cepat vs tingkat usaha

Mode cepat dan tingkat usaha keduanya mempengaruhi kecepatan respons, tetapi dengan cara yang berbeda:

Pengaturan	Efek
Mode cepat	Kualitas model yang sama, latensi lebih rendah, biaya lebih tinggi
Tingkat usaha lebih rendah	Waktu pemikiran lebih sedikit, respons lebih cepat, potensi kualitas lebih rendah pada tugas kompleks

Anda dapat menggabungkan keduanya: gunakan mode cepat dengan tingkat usaha yang lebih rendah untuk kecepatan maksimal pada tugas yang mudah.

Persyaratan

Mode cepat memerlukan semua hal berikut:

- **Tidak tersedia di penyedia cloud pihak ketiga**: mode cepat tidak tersedia di Amazon Bedrock, Google Vertex AI, atau Microsoft Azure Foundry. Mode cepat tersedia melalui API Konsol Anthropic dan untuk paket berlangganan Claude menggunakan penggunaan tambahan.
- **Penggunaan tambahan diaktifkan**: akun Anda harus memiliki penggunaan tambahan diaktifkan, yang memungkinkan penagihan di luar penggunaan yang disertakan dalam paket Anda. Untuk akun individual, aktifkan ini di pengaturan penagihan Konsol Anda. Untuk Teams dan Enterprise, admin harus mengaktifkan penggunaan tambahan untuk organisasi.

> **Note:**
>
> Penggunaan mode cepat ditagih langsung ke penggunaan tambahan, bahkan jika Anda memiliki penggunaan yang tersisa di paket Anda. Ini berarti token mode cepat tidak dihitung terhadap penggunaan yang disertakan dalam paket Anda dan dikenakan biaya dengan tarif mode cepat dari token pertama.

- **Aktivasi admin untuk Teams dan Enterprise**: mode cepat dinonaktifkan secara default untuk organisasi Teams dan Enterprise. Admin harus secara eksplisit mengaktifkan mode cepat sebelum pengguna dapat mengaksesnya.

> **Note:**
>
> Jika admin Anda belum mengaktifkan mode cepat untuk organisasi Anda, perintah `/fast` akan menampilkan "Fast mode has been disabled by your organization."

Aktifkan mode cepat untuk organisasi Anda

Admin dapat mengaktifkan mode cepat di:

- **Console** (pelanggan API): preferensi Claude Code
- **Claude AI** (Teams dan Enterprise): Admin Settings > Claude Code

Opsi lain untuk menonaktifkan mode cepat sepenuhnya adalah mengatur `CLAUDE_CODE_DISABLE_FAST_MODE=1`. Lihat Variabel lingkungan.

Require per-session opt-in

Secara default, mode cepat bertahan di seluruh sesi: jika pengguna mengaktifkan mode cepat, mode ini tetap aktif di sesi mendatang. Administrator pada paket Teams atau Enterprise dapat mencegah ini dengan mengatur `fastModePerSessionOptIn` ke `true` di pengaturan terkelola atau pengaturan yang dikelola server. Ini menyebabkan setiap sesi dimulai dengan mode cepat mati, memerlukan pengguna untuk secara eksplisit mengaktifkannya dengan `/fast`.

```
{
  "fastModePerSessionOptIn": true
}
```

Ini berguna untuk mengontrol biaya di organisasi di mana pengguna menjalankan beberapa sesi bersamaan. Pengguna masih dapat mengaktifkan mode cepat dengan `/fast` ketika mereka membutuhkan kecepatan, tetapi mode ini disetel ulang di awal setiap sesi baru. Preferensi mode cepat pengguna masih disimpan, jadi menghapus pengaturan ini mengembalikan perilaku persisten default.

Tangani batas laju

Mode cepat memiliki batas laju terpisah dari Opus 4.6 standar. Ketika Anda mencapai batas laju mode cepat atau kehabisan kredit penggunaan tambahan:

1. Mode cepat secara otomatis kembali ke Opus 4.6 standar
2. Ikon `↯` berubah menjadi abu-abu untuk menunjukkan cooldown
3. Anda terus bekerja dengan kecepatan dan harga standar
4. Ketika cooldown berakhir, mode cepat secara otomatis diaktifkan kembali

Untuk menonaktifkan mode cepat secara manual daripada menunggu cooldown, jalankan `/fast` lagi.

Pratinjau penelitian

Mode cepat adalah fitur pratinjau penelitian. Ini berarti:

- Fitur dapat berubah berdasarkan umpan balik
- Ketersediaan dan harga dapat berubah
- Konfigurasi API yang mendasar dapat berkembang

Laporkan masalah atau umpan balik melalui saluran dukungan Anthropic biasa Anda.

Lihat juga

- Konfigurasi model: beralih model dan sesuaikan tingkat usaha
- Kelola biaya secara efektif: lacak penggunaan token dan kurangi biaya
- Konfigurasi baris status: tampilkan informasi model dan konteks

Kelola biaya secara efektif

> *Lacak penggunaan token, tetapkan batas pengeluaran tim, dan kurangi biaya Claude Code dengan manajemen konteks, pemilihan model, pengaturan pemikiran yang diperluas, dan hook prapemrosesan.*

Claude Code mengonsumsi token untuk setiap interaksi. Biaya bervariasi berdasarkan ukuran basis kode, kompleksitas kueri, dan panjang percakapan. Biaya rata-rata adalah $6 per pengembang per hari, dengan biaya harian tetap di bawah $12 untuk 90% pengguna.

Untuk penggunaan tim, Claude Code mengenakan biaya berdasarkan konsumsi token API. Rata-rata, Claude Code berharga ~$100-200/pengembang per bulan dengan Sonnet 4.6 meskipun ada varians besar tergantung pada berapa banyak instans yang dijalankan pengguna dan apakah mereka menggunakannya dalam otomasi.

Halaman ini mencakup cara melacak biaya Anda, mengelola biaya untuk tim, dan mengurangi penggunaan token.

Lacak biaya Anda

Menggunakan perintah `/cost`

> **Note:**
>
> Perintah `/cost` menampilkan penggunaan token API dan dimaksudkan untuk pengguna API. Pelanggan Claude Max dan Pro memiliki penggunaan yang disertakan dalam langganan mereka, jadi data `/cost` tidak relevan untuk tujuan penagihan. Pelanggan dapat menggunakan `/stats` untuk melihat pola penggunaan.

Perintah `/cost` menyediakan statistik penggunaan token terperinci untuk sesi Anda saat ini:

```
Total cost:            $0.55
Total duration (API):  6m 19.7s
Total duration (wall): 6h 33m 10.2s
Total code changes:    0 lines added, 0 lines removed
```

Mengelola biaya untuk tim

Saat menggunakan Claude API, Anda dapat menetapkan batas pengeluaran ruang kerja pada total pengeluaran ruang kerja Claude Code. Admin dapat melihat pelaporan biaya dan penggunaan di Konsol.

> **Note:**
>
> Ketika Anda pertama kali mengautentikasi Claude Code dengan akun Claude Console Anda, ruang kerja yang disebut "Claude Code" secara otomatis dibuat untuk Anda. Ruang kerja ini menyediakan pelacakan dan manajemen biaya terpusat untuk semua penggunaan Claude Code di organisasi Anda. Anda tidak dapat membuat kunci API untuk ruang kerja ini; ini secara eksklusif untuk autentikasi dan penggunaan Claude Code.

Di Bedrock, Vertex, dan Foundry, Claude Code tidak mengirim metrik dari cloud Anda. Untuk mendapatkan metrik biaya, beberapa perusahaan besar melaporkan menggunakan LiteLLM, yang merupakan alat sumber terbuka yang membantu perusahaan melacak pengeluaran berdasarkan kunci. Proyek ini tidak berafiliasi dengan Anthropic dan belum diaudit untuk keamanan.

Rekomendasi batas laju

Saat menyiapkan Claude Code untuk tim, pertimbangkan rekomendasi Token Per Minute (TPM) dan Request Per Minute (RPM) per pengguna ini berdasarkan ukuran organisasi Anda:

Ukuran tim	TPM per pengguna	RPM per pengguna
1-5 pengguna	200k-300k	5-7
5-20 pengguna	100k-150k	2.5-3.5
20-50 pengguna	50k-75k	1.25-1.75
50-100 pengguna	25k-35k	0.62-0.87
100-500 pengguna	15k-20k	0.37-0.47
500+ pengguna	10k-15k	0.25-0.35

Misalnya, jika Anda memiliki 200 pengguna, Anda mungkin meminta 20k TPM untuk setiap pengguna, atau 4 juta total TPM (200*20.000 = 4 juta).

TPM per pengguna menurun seiring pertumbuhan ukuran tim karena lebih sedikit pengguna yang cenderung menggunakan Claude Code secara bersamaan di organisasi yang lebih besar. Batas laju ini berlaku di tingkat organisasi, bukan per pengguna individual, yang berarti pengguna individual dapat sementara mengonsumsi lebih dari bagian yang dihitung mereka ketika orang lain tidak secara aktif menggunakan layanan.

> **Note:**
>
> Jika Anda mengantisipasi skenario dengan penggunaan bersamaan yang tidak biasa tinggi (seperti sesi pelatihan langsung dengan kelompok besar), Anda mungkin memerlukan alokasi TPM yang lebih tinggi per pengguna.

Biaya token tim agen

Tim agen menjalankan beberapa instans Claude Code, masing-masing dengan jendela konteks sendiri. Penggunaan token diskalakan dengan jumlah rekan kerja aktif dan berapa lama masing-masing berjalan.

Untuk menjaga biaya tim agen tetap dapat dikelola:

- Gunakan Sonnet untuk rekan kerja. Ini menyeimbangkan kemampuan dan biaya untuk tugas koordinasi.
- Jaga tim tetap kecil. Setiap rekan kerja menjalankan jendela konteks sendiri, jadi penggunaan token kira-kira sebanding dengan ukuran tim.
- Jaga prompt spawn tetap fokus. Rekan kerja memuat CLAUDE.md, server MCP, dan skills secara otomatis, tetapi semuanya dalam prompt spawn menambah konteks mereka dari awal.
- Bersihkan tim ketika pekerjaan selesai. Rekan kerja aktif terus mengonsumsi token bahkan jika menganggur.
- Tim agen dinonaktifkan secara default. Atur `CLAUDE_CODE_EXPERIMENTAL_AGENT_TEAMS=1` di settings.json atau lingkungan Anda untuk mengaktifkannya. Lihat aktifkan tim agen.

Kurangi penggunaan token

Biaya token diskalakan dengan ukuran konteks: semakin banyak konteks yang diproses Claude, semakin banyak token yang Anda gunakan. Claude Code secara otomatis mengoptimalkan biaya melalui prompt caching (yang mengurangi biaya untuk konten berulang seperti prompt sistem) dan auto-compact (yang merangkum riwayat percakapan saat mendekati batas konteks).

Strategi berikut membantu Anda menjaga konteks tetap kecil dan mengurangi biaya per pesan.

Kelola konteks secara proaktif

Gunakan `/cost` untuk memeriksa penggunaan token Anda saat ini, atau konfigurasi baris status Anda untuk menampilkannya secara berkelanjutan.

- **Bersihkan antar tugas**: Gunakan `/clear` untuk memulai segar saat beralih ke pekerjaan yang tidak terkait. Konteks basi membuang token pada setiap pesan berikutnya. Gunakan `/rename` sebelum membersihkan sehingga Anda dapat dengan mudah menemukan sesi nanti, kemudian `/resume` untuk kembali ke sana.
- **Tambahkan instruksi compaction kustom**: `/compact Focus on code samples and API usage` memberi tahu Claude apa yang harus dipertahankan selama perangkuman.

Anda juga dapat menyesuaikan perilaku compaction di CLAUDE.md Anda:

```
## Compact instructions

When you are using compact, please focus on test output and code changes
```

Pilih model yang tepat

Sonnet menangani sebagian besar tugas pengkodean dengan baik dan biayanya lebih rendah dari Opus. Cadangkan Opus untuk keputusan arsitektur yang kompleks atau penalaran multi-langkah. Gunakan `/model` untuk beralih model di tengah sesi, atau atur default di `/config`. Untuk tugas subagent sederhana, tentukan `model: haiku` di konfigurasi subagent Anda.

Kurangi overhead server MCP

Setiap server MCP menambahkan definisi alat ke konteks Anda, bahkan saat menganggur. Jalankan `/context` untuk melihat apa yang mengonsumsi ruang.

- **Lebih suka alat CLI jika tersedia**: Alat seperti `gh`, `aws`, `gcloud`, dan `sentry-cli` lebih efisien konteks daripada server MCP karena mereka tidak menambahkan definisi alat yang persisten. Claude dapat menjalankan perintah CLI secara langsung tanpa overhead.
- **Nonaktifkan server yang tidak digunakan**: Jalankan `/mcp` untuk melihat server yang dikonfigurasi dan nonaktifkan yang tidak Anda gunakan secara aktif.

- **Pencarian alat bersifat otomatis**: Ketika deskripsi alat MCP melebihi 10% dari jendela konteks Anda, Claude Code secara otomatis menunda mereka dan memuat alat sesuai permintaan melalui pencarian alat. Karena alat yang ditunda hanya memasuki konteks saat benar-benar digunakan, ambang batas yang lebih rendah berarti lebih sedikit definisi alat menganggur yang mengonsumsi ruang. Atur ambang batas yang lebih rendah dengan `ENABLE_TOOL_SEARCH=auto:<N>` (misalnya, `auto:5` memicu ketika alat melebihi 5% dari jendela konteks Anda).

Instal plugin kecerdasan kode untuk bahasa yang diketik

Plugin kecerdasan kode memberi Claude navigasi simbol yang tepat daripada pencarian berbasis teks, mengurangi pembacaan file yang tidak perlu saat menjelajahi kode yang tidak dikenal. Satu panggilan "go to definition" menggantikan apa yang mungkin merupakan grep diikuti dengan membaca beberapa file kandidat. Server bahasa yang diinstal juga melaporkan kesalahan tipe secara otomatis setelah pengeditan, jadi Claude menangkap kesalahan tanpa menjalankan compiler.

Offload pemrosesan ke hooks dan skills

Hooks kustom dapat memproses data sebelum Claude melihatnya. Alih-alih Claude membaca file log 10.000 baris untuk menemukan kesalahan, hook dapat grep untuk `ERROR` dan mengembalikan hanya baris yang cocok, mengurangi konteks dari puluhan ribu token menjadi ratusan.

Skill dapat memberi Claude pengetahuan domain sehingga tidak harus menjelajahi. Misalnya, skill "codebase-overview" dapat mendeskripsikan arsitektur proyek Anda, direktori kunci, dan konvensi penamaan. Ketika Claude memanggil skill, ia mendapatkan konteks ini segera daripada menghabiskan token membaca beberapa file untuk memahami struktur.

Misalnya, hook PreToolUse ini memfilter output tes untuk menampilkan hanya kegagalan:

settings.json

Tambahkan ini ke settings.json Anda untuk menjalankan hook sebelum setiap perintah Bash:

```
{
  "hooks": {
    "PreToolUse": [
      {
        "matcher": "Bash",
        "hooks": [
          {
            "type": "command",
            "command": "~/.claude/hooks/filter-test-output.sh"
          }
        ]
      }
    ]
  }
}
```

filter-test-output.sh

Hook memanggil skrip ini, yang memeriksa apakah perintah adalah test runner dan memodifikasinya untuk menampilkan hanya kegagalan:

```
#!/bin/bash
input=$(cat)
cmd=$(echo "$input" | jq -r '.tool_input.command')

## If running tests, filter to show only failures
if [[ "$cmd" =~ ^(npm test|pytest|go test) ]]; then
  filtered_cmd="$cmd 2>&1 | grep -A 5 -E '(FAIL|ERROR|error:)' | head -100"
  echo "{\"hookSpecificOutput\":{\"hookEventName\":\"PreToolUse\",\"permissionDeci
sion\":\"allow\",\"updatedInput\":{\"command\":\"$filtered_cmd\"}}}"
else
  echo "{}"
fi
```

Pindahkan instruksi dari CLAUDE.md ke skills

File CLAUDE.md Anda dimuat ke konteks saat awal sesi. Jika berisi instruksi terperinci untuk alur kerja spesifik (seperti ulasan PR atau migrasi database), token tersebut ada bahkan ketika Anda melakukan pekerjaan yang tidak terkait. Skills dimuat sesuai permin-

taan hanya saat dipanggil, jadi memindahkan instruksi khusus ke skills menjaga konteks dasar Anda tetap lebih kecil. Bertujuan untuk menjaga CLAUDE.md di bawah ~500 baris dengan hanya menyertakan hal-hal penting.

Sesuaikan pemikiran yang diperluas

Pemikiran yang diperluas diaktifkan secara default dengan anggaran 31.999 token karena secara signifikan meningkatkan kinerja pada tugas perencanaan dan penalaran yang kompleks. Namun, token pemikiran ditagih sebagai token output, jadi untuk tugas yang lebih sederhana di mana penalaran mendalam tidak diperlukan, Anda dapat mengurangi biaya dengan menurunkan tingkat upaya di `/model` untuk Opus 4.6, menonaktifkan pemikiran di `/config`, atau menurunkan anggaran (misalnya, `MAX_THINKING_TOKENS=8000`).

Delegasikan operasi verbose ke subagents

Menjalankan tes, mengambil dokumentasi, atau memproses file log dapat mengonsumsi konteks yang signifikan. Delegasikan ini ke subagents sehingga output verbose tetap dalam konteks subagent sementara hanya ringkasan yang kembali ke percakapan utama Anda.

Kelola biaya tim agen

Tim agen menggunakan sekitar 7x lebih banyak token daripada sesi standar ketika rekan kerja berjalan dalam plan mode, karena setiap rekan kerja mempertahankan jendela konteks sendiri dan berjalan sebagai instans Claude terpisah. Jaga tugas tim tetap kecil dan mandiri untuk membatasi penggunaan token per rekan kerja. Lihat tim agen untuk detail.

Tulis prompt spesifik

Permintaan yang tidak jelas seperti “tingkatkan basis kode ini” memicu pemindaian luas. Permintaan spesifik seperti “tambahkan validasi input ke fungsi login di auth.ts” memungkinkan Claude bekerja secara efisien dengan pembacaan file minimal.

Bekerja secara efisien pada tugas yang kompleks

Untuk pekerjaan yang lebih lama atau lebih kompleks, kebiasaan ini membantu menghindari token yang terbuang dari mengambil jalan yang salah:

- **Gunakan plan mode untuk tugas yang kompleks**: Tekan Shift+Tab untuk memasuki plan mode sebelum implementasi. Claude menjelajahi basis kode dan mengusulkan pendekatan untuk persetujuan Anda, mencegah pekerjaan ulang yang mahal ketika arah awal salah.

- **Koreksi kursus lebih awal**: Jika Claude mulai menuju arah yang salah, tekan Escape untuk berhenti segera. Gunakan `/rewind` atau tekan dua kali Escape untuk mengembalikan percakapan dan kode ke checkpoint sebelumnya.
- **Berikan target verifikasi**: Sertakan kasus uji, tempel tangkapan layar, atau tentukan output yang diharapkan dalam prompt Anda. Ketika Claude dapat memverifikasi pekerjaan sendiri, ia menangkap masalah sebelum Anda perlu meminta perbaikan.
- **Uji secara bertahap**: Tulis satu file, uji, kemudian lanjutkan. Ini menangkap masalah lebih awal ketika murah untuk diperbaiki.

Penggunaan token latar belakang

Claude Code menggunakan token untuk beberapa fungsi latar belakang bahkan saat menganggur:

- **Perangkuman percakapan**: Pekerjaan latar belakang yang merangkum percakapan sebelumnya untuk fitur `claude --resume`
- **Pemrosesan perintah**: Beberapa perintah seperti `/cost` dapat menghasilkan permintaan untuk memeriksa status

Proses latar belakang ini mengonsumsi sejumlah kecil token (biasanya di bawah $0,04 per sesi) bahkan tanpa interaksi aktif.

Memahami perubahan dalam perilaku Claude Code

Claude Code secara teratur menerima pembaruan yang dapat mengubah cara fitur bekerja, termasuk pelaporan biaya. Jalankan `claude --version` untuk memeriksa versi Anda saat ini. Untuk pertanyaan penagihan spesifik, hubungi dukungan Anthropic melalui akun Konsol Anda. Untuk penyebaran tim, mulai dengan kelompok pilot kecil untuk membangun pola penggunaan sebelum peluncuran yang lebih luas.

Part 5: Extensibility

Referensi Hooks

> *Referensi untuk event hook Claude Code, skema konfigurasi, format JSON input/output, kode keluar, hooks asinkron, hooks HTTP, hooks prompt, dan hooks alat MCP.*

> **Tip:**
>
> Untuk panduan quickstart dengan contoh, lihat Otomatisasi alur kerja dengan hooks.

Hooks adalah perintah shell yang ditentukan pengguna, endpoint HTTP, atau prompt LLM yang dijalankan secara otomatis pada titik-titik tertentu dalam siklus hidup Claude Code. Gunakan referensi ini untuk mencari skema event, opsi konfigurasi, format JSON input/output, dan fitur lanjutan seperti hooks asinkron, hooks HTTP, dan hooks alat MCP. Jika Anda menyiapkan hooks untuk pertama kalinya, mulai dengan panduan sebagai gantinya.

Siklus hidup hook

Hooks dijalankan pada titik-titik tertentu selama sesi Claude Code. Ketika event dijalankan dan matcher cocok, Claude Code meneruskan konteks JSON tentang event ke handler hook Anda. Untuk command hooks, input tiba di stdin. Untuk HTTP hooks, input tiba sebagai badan permintaan POST. Handler Anda kemudian dapat memeriksa input, mengambil tindakan, dan secara opsional mengembalikan keputusan. Beberapa event dijalankan sekali per sesi, sementara yang lain dijalankan berulang kali di dalam loop agentic:

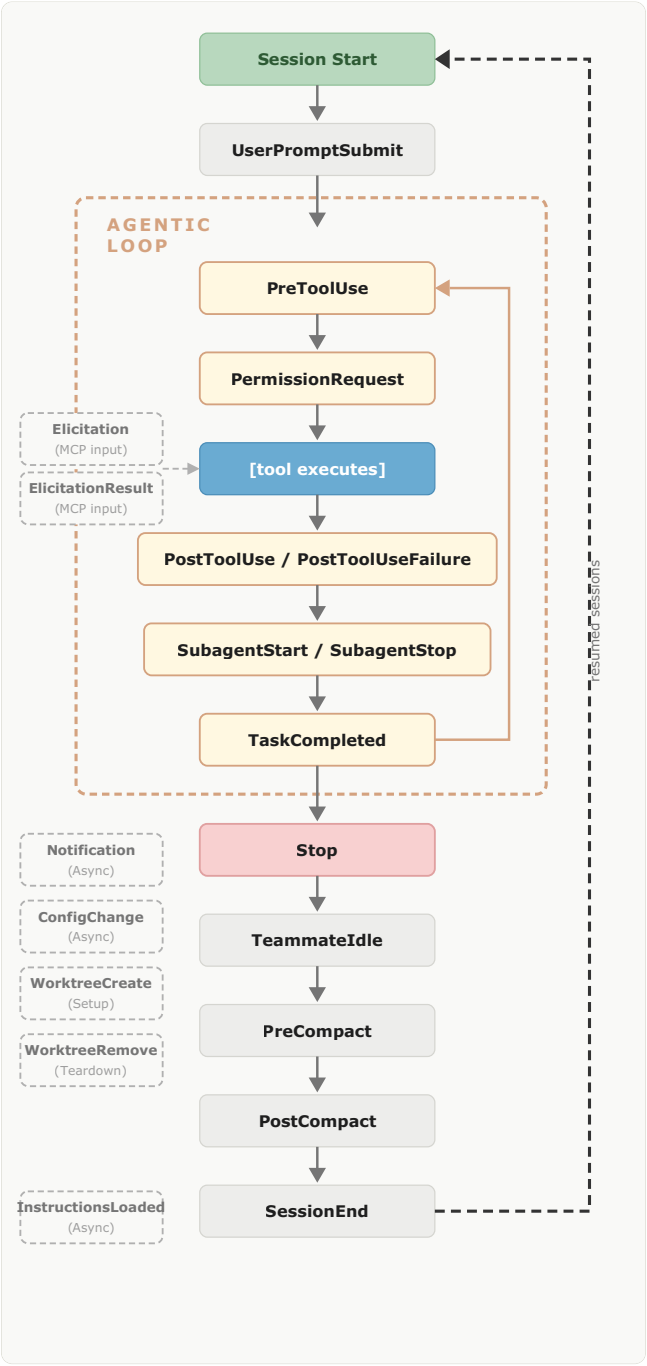

Session Start
UserPromptSubmit
AGENTIC LOOP
PreToolUse
PermissionRequest
Elicitation
(MCP input)
[tool executes]
ElicitationResult
(MCP input)
PostToolUse / PostToolUseFailure
SubagentStart / SubagentStop
TaskCompleted
Stop
Notification
(Async)
ConfigChange
(Async)
TeammateIdle
WorktreeCreate
(Setup)
PreCompact
WorktreeRemove
(Teardown)
PostCompact
InstructionsLoaded
(Async)
SessionEnd
resumed sessions

Diagram siklus hidup hook menunjukkan urutan hooks dari SessionStart melalui loop agentic ke SessionEnd, dengan WorktreeCreate, WorktreeRemove, dan InstructionsLoaded sebagai event asinkron mandiri

Tabel di bawah merangkum kapan setiap event dijalankan. Bagian Hook events mendokumentasikan skema input lengkap dan opsi kontrol keputusan untuk masing-masing.

Event	When it fires
`SessionStart`	When a session begins or resumes
`UserPromptSubmit`	When you submit a prompt, before Claude processes it
`PreToolUse`	Before a tool call executes. Can block it
`PermissionRequest`	When a permission dialog appears
`PostToolUse`	After a tool call succeeds
`PostToolUseFailure`	After a tool call fails
`Notification`	When Claude Code sends a notification
`SubagentStart`	When a subagent is spawned
`SubagentStop`	When a subagent finishes
`Stop`	When Claude finishes responding
`TeammateIdle`	When an agent team teammate is about to go idle
`TaskCompleted`	When a task is being marked as completed
`InstructionsLoaded`	When a CLAUDE.md or `.claude/rules/*.md` file is loaded into context. Fires at session start and when files are lazily loaded during a session
`ConfigChange`	When a configuration file changes during a session
`WorktreeCreate`	When a worktree is being created via `--worktree` or `isolation: "worktree"`. Replaces default git behavior
`WorktreeRemove`	When a worktree is being removed, either at session exit or when a subagent finishes
`PreCompact`	Before context compaction
`PostCompact`	After context compaction completes

Event	When it fires
`Elicitation`	When an MCP server requests user input during a tool call
`ElicitationResult`	After a user responds to an MCP elicitation, before the response is sent back to the server
`SessionEnd`	When a session terminates

Bagaimana hook diselesaikan

Untuk melihat bagaimana potongan-potongan ini cocok bersama, pertimbangkan hook `PreToolUse` ini yang memblokir perintah shell yang merusak. Hook menjalankan `block-rm.sh` sebelum setiap pemanggilan alat Bash:

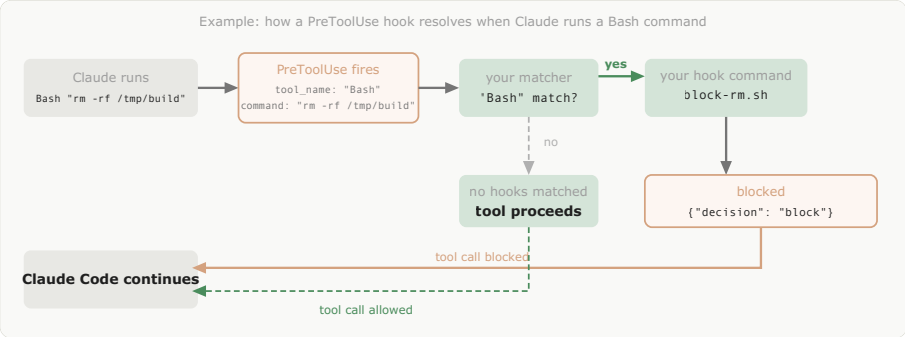
```
{
  "hooks": {
    "PreToolUse": [
      {
        "matcher": "Bash",
        "hooks": [
          {
            "type": "command",
            "command": ".claude/hooks/block-rm.sh"
          }
        ]
      }
    ]
  }
}
```

Skrip membaca input JSON dari stdin, mengekstrak perintah, dan mengembalikan `permissionDecision` dari `"deny"` jika berisi `rm -rf`:

```
#!/bin/bash
## .claude/hooks/block-rm.sh
COMMAND=$(jq -r '.tool_input.command')

if echo "$COMMAND" | grep -q 'rm -rf'; then
  jq -n '{
    hookSpecificOutput: {
      hookEventName: "PreToolUse",
      permissionDecision: "deny",
      permissionDecisionReason: "Destructive command blocked by hook"
    }
  }'
else
  exit 0  # allow the command
fi
```

Sekarang anggaplah Claude Code memutuskan untuk menjalankan `Bash "rm -rf /tmp/build"`. Inilah yang terjadi:



Alur resolusi hook: event PreToolUse dijalankan, matcher memeriksa kecocokan Bash, handler hook dijalankan, hasil dikembalikan ke Claude Code

Step 1: Event dijalankan

Event `PreToolUse` dijalankan. Claude Code mengirimkan input alat sebagai JSON di stdin ke hook:

```
{ "tool_name": "Bash", "tool_input": { "command": "rm -rf /tmp/build" }, ... }
```

Step 2: Matcher memeriksa

Matcher `"Bash"` cocok dengan nama alat, jadi `block-rm.sh` dijalankan. Jika Anda menghilangkan matcher atau menggunakan `"*"`, hook dijalankan pada setiap kemunculan event. Hooks hanya dilewati ketika matcher didefinisikan dan tidak cocok.

Step 3: Handler hook dijalankan

Skrip mengekstrak `"rm -rf /tmp/build"` dari input dan menemukan `rm -rf`, jadi itu mencetak keputusan ke stdout:

```
{
  "hookSpecificOutput": {
    "hookEventName": "PreToolUse",
    "permissionDecision": "deny",
    "permissionDecisionReason": "Destructive command blocked by hook"
  }
}
```

Jika perintah telah aman (seperti `npm test`), skrip akan mencapai `exit 0` sebagai gantinya, yang memberitahu Claude Code untuk mengizinkan pemanggilan alat tanpa tindakan lebih lanjut.

Step 4: Claude Code bertindak atas hasil

Claude Code membaca keputusan JSON, memblokir pemanggilan alat, dan menunjukkan alasannya kepada Claude.

Bagian Configuration di bawah mendokumentasikan skema lengkap, dan setiap bagian hook event mendokumentasikan input apa yang diterima perintah Anda dan output apa yang dapat dikembalikan.

Konfigurasi

Hooks didefinisikan dalam file pengaturan JSON. Konfigurasi memiliki tiga tingkat nesting:

1. Pilih hook event untuk merespons, seperti `PreToolUse` atau `Stop`
2. Tambahkan matcher group untuk memfilter kapan dijalankan, seperti “hanya untuk alat Bash”
3. Tentukan satu atau lebih hook handlers untuk dijalankan saat cocok

Lihat Bagaimana hook diselesaikan di atas untuk panduan lengkap dengan contoh beranotasi.

> **Note:**
>
> Halaman ini menggunakan istilah spesifik untuk setiap tingkat: **hook event** untuk titik siklus hidup, **matcher group** untuk filter, dan **hook handler** untuk perintah shell, endpoint HTTP, prompt, atau agen yang dijalankan. "Hook" sendiri merujuk pada fitur umum.

Lokasi hook

Tempat Anda mendefinisikan hook menentukan cakupannya:

Lokasi	Cakupan	Dapat Dibagikan
`~/.claude/settings.json`	Semua proyek Anda	Tidak, lokal ke mesin Anda
`.claude/settings.json`	Proyek tunggal	Ya, dapat dikomit ke repo
`.claude/settings.local.json`	Proyek tunggal	Tidak, diabaikan git
Pengaturan kebijakan terkelola	Seluruh organisasi	Ya, dikendalikan admin
Plugin `hooks/hooks.json`	Ketika plugin diaktifkan	Ya, dibundel dengan plugin
Skill atau agent frontmatter	Saat komponen aktif	Ya, didefinisikan dalam file komponen

Untuk detail tentang resolusi file pengaturan, lihat settings. Administrator enterprise dapat menggunakan `allowManagedHooksOnly` untuk memblokir hooks pengguna, proyek, dan plugin. Lihat Hook configuration.

Pola matcher

Bidang `matcher` adalah string regex yang memfilter kapan hooks dijalankan. Gunakan `"*"`, `""`, atau hilangkan `matcher` sepenuhnya untuk mencocokkan semua kemunculan. Setiap tipe event cocok pada bidang yang berbeda:

Event	Apa yang difilter matcher	Contoh nilai matcher
`PreToolUse`, `PostToolUse`, `PostToolUseFailure`, `PermissionRequest`	nama alat	`Bash`, `Edit|Write`, `mcp__.*`
`SessionStart`	bagaimana sesi dimulai	`startup`, `resume`, `clear`, `compact`
`SessionEnd`	mengapa sesi berakhir	`clear`, `logout`, `prompt_input_exit`, `bypass_permissions_disabled`, `other`
`Notification`	tipe notifikasi	`permission_prompt`, `idle_prompt`, `auth_success`, `elicitation_dialog`
`SubagentStart`	tipe agen	`Bash`, `Explore`, `Plan`, atau nama agen kustom
`PreCompact`	apa yang memicu pemadatan	`manual`, `auto`
`SubagentStop`	tipe agen	nilai yang sama seperti `SubagentStart`
`ConfigChange`	sumber konfigurasi	`user_settings`, `project_settings`, `local_settings`, `policy_settings`, `skills`
`UserPromptSubmit`, `Stop`, `TeammateIdle`, `TaskCompleted`, `WorktreeCreate`, `WorktreeRemove`, `InstructionsLoaded`	tidak ada dukungan matcher	selalu dijalankan pada setiap kemunculan

Matcher adalah regex, jadi `Edit|Write` cocok dengan alat mana pun dan `Notebook.*` cocok dengan alat apa pun yang dimulai dengan Notebook. Matcher dijalankan terhadap bidang dari JSON input yang Claude Code kirimkan ke hook Anda di stdin. Untuk event alat, bidang itu adalah `tool_name`. Setiap bagian hook event mencantumkan set lengkap nilai matcher dan skema input untuk event itu.

Contoh ini menjalankan skrip linting hanya ketika Claude menulis atau mengedit file:

```
{
  "hooks": {
    "PostToolUse": [
      {
        "matcher": "Edit|Write",
        "hooks": [
          {
            "type": "command",
            "command": "/path/to/lint-check.sh"
          }
        ]
      }
    ]
  }
}
```

`UserPromptSubmit`, `Stop`, `TeammateIdle`, `TaskCompleted`, `WorktreeCreate`, `WorktreeRemove`, dan `InstructionsLoaded` tidak mendukung matchers dan selalu dijalankan pada setiap kemunculan. Jika Anda menambahkan bidang `matcher` ke event ini, itu akan diabaikan secara diam-diam.

Cocokkan alat MCP

Alat server MCP muncul sebagai alat biasa dalam event alat (`PreToolUse`, `PostToolUse`, `PostToolUseFailure`, `PermissionRequest`), jadi Anda dapat mencocokkannya dengan cara yang sama seperti Anda mencocokkan nama alat lainnya.

Alat MCP mengikuti pola penamaan `mcp__<server>__<tool>`, misalnya:

- `mcp__memory__create_entities`: alat create entities dari server Memory
- `mcp__filesystem__read_file`: alat read file dari server Filesystem
- `mcp__github__search_repositories`: alat search dari server GitHub

Gunakan pola regex untuk menargetkan alat MCP tertentu atau grup alat:

- `mcp__memory__.*` cocok dengan semua alat dari server `memory`
- `mcp__.*__write.*` cocok dengan alat apa pun yang berisi "write" dari server apa pun

Contoh ini mencatat semua operasi server memory dan memvalidasi operasi write dari server MCP apa pun:

```
{
  "hooks": {
    "PreToolUse": [
      {
        "matcher": "mcp__memory__.*",
        "hooks": [
          {
            "type": "command",
            "command": "echo 'Memory operation initiated' >> ~/mcp-operations.log"
          }
        ]
      },
      {
        "matcher": "mcp__.*__write.*",
        "hooks": [
          {
            "type": "command",
            "command": "/home/user/scripts/validate-mcp-write.py"
          }
        ]
      }
    ]
  }
}
```

Bidang hook handler

Setiap objek dalam array `hooks` inner adalah hook handler: perintah shell, endpoint HTTP, prompt LLM, atau agen yang dijalankan saat matcher cocok. Ada empat tipe:

- **Command hooks** (`type: "command"`): jalankan perintah shell. Skrip Anda menerima JSON input event di stdin dan mengkomunikasikan hasil kembali melalui kode keluar dan stdout.

- **HTTP hooks** (`type: "http"`): kirimkan JSON input event sebagai permintaan HTTP POST ke URL. Endpoint mengkomunikasikan hasil kembali melalui badan respons menggunakan format JSON output yang sama seperti command hooks.
- **Prompt hooks** (`type: "prompt"`): kirimkan prompt ke model Claude untuk evaluasi single-turn. Model mengembalikan keputusan yes/no sebagai JSON. Lihat Prompt-based hooks.
- **Agent hooks** (`type: "agent"`): spawn subagent yang dapat menggunakan alat seperti Read, Grep, dan Glob untuk memverifikasi kondisi sebelum mengembalikan keputusan. Lihat Agent-based hooks.

Bidang umum

Bidang-bidang ini berlaku untuk semua tipe hook:

Bidang	Diperlukan	Deskripsi
`type`	ya	`"command"` , `"http"` , `"prompt"` , atau `"agent"`
`timeout`	tidak	Detik sebelum membatalkan. Default: 600 untuk command, 30 untuk prompt, 60 untuk agent
`statusMessage`	tidak	Pesan spinner kustom ditampilkan saat hook dijalankan
`once`	tidak	Jika `true` , dijalankan hanya sekali per sesi kemudian dihapus. Hanya skills, bukan agents. Lihat Hooks in skills and agents

Bidang command hook

Selain bidang umum, command hooks menerima bidang-bidang ini:

Bidang	Diperlukan	Deskripsi
`command`	ya	Perintah shell untuk dijalankan

Bidang	Diperlukan	Deskripsi
`async`	tidak	Jika `true`, dijalankan di latar belakang tanpa memblokir. Lihat Run hooks in the background

Bidang HTTP hook

Selain bidang umum, HTTP hooks menerima bidang-bidang ini:

Bidang	Diperlukan	Deskripsi
`url`	ya	URL untuk mengirimkan permintaan POST ke
`headers`	tidak	Header HTTP tambahan sebagai pasangan kunci-nilai. Nilai mendukung interpolasi variabel lingkungan menggunakan sintaks `$VAR_NAME` atau `${VAR_NAME}`. Hanya variabel yang tercantum dalam `allowedEnvVars` yang diselesaikan
`allowedEnvVars`	tidak	Daftar nama variabel lingkungan yang dapat diinterpolasi ke dalam nilai header. Referensi ke variabel yang tidak tercantum diganti dengan string kosong. Diperlukan untuk interpolasi variabel env apa pun untuk bekerja

Claude Code mengirimkan JSON input hook sebagai badan permintaan POST dengan `Content-Type: application/json`. Badan respons menggunakan format JSON output yang sama seperti command hooks.

Penanganan kesalahan berbeda dari command hooks: respons non-2xx, kegagalan koneksi, dan timeout semuanya menghasilkan kesalahan non-blocking yang memungkinkan eksekusi untuk melanjutkan. Untuk memblokir pemanggilan alat atau menolak izin, kembalikan respons 2xx dengan badan JSON yang berisi `decision: "block"` atau `hookSpecificOutput` dengan `permissionDecision: "deny"`.

Contoh ini mengirimkan event `PreToolUse` ke layanan validasi lokal, mengautentikasi dengan token dari variabel lingkungan `MY_TOKEN`:

```
{
  "hooks": {
    "PreToolUse": [
      {
        "matcher": "Bash",
        "hooks": [
          {
            "type": "http",
            "url": "http://localhost:8080/hooks/pre-tool-use",
            "timeout": 30,
            "headers": {
              "Authorization": "Bearer $MY_TOKEN"
            },
            "allowedEnvVars": ["MY_TOKEN"]
          }
        ]
      }
    ]
  }
}
```

> **Note:**
>
> HTTP hooks harus dikonfigurasi dengan mengedit JSON pengaturan secara langsung. Menu `/hooks` interaktif hanya mendukung penambahan command hooks.

Bidang prompt dan agent hook

Selain bidang umum, prompt dan agent hooks menerima bidang-bidang ini:

Bidang	Diperlukan	Deskripsi
`prompt`	ya	Teks prompt untuk dikirim ke model. Gunakan `$ARGUMENTS` sebagai placeholder untuk JSON input hook
`model`	tidak	Model untuk digunakan untuk evaluasi. Default ke model cepat

Semua matching hooks dijalankan secara paralel, dan handler identik dideduplikasi secara otomatis. Command hooks dideduplikasi berdasarkan string perintah, dan HTTP hooks dideduplikasi berdasarkan URL. Handler dijalankan di direktori saat ini dengan lingkungan Claude Code. Variabel lingkungan `$CLAUDE_CODE_REMOTE` diatur ke `"true"` di lingkungan web jarak jauh dan tidak diatur di CLI lokal.

Referensi skrip berdasarkan path

Gunakan variabel lingkungan untuk mereferensikan skrip hook relatif terhadap akar proyek atau plugin, terlepas dari direktori kerja saat hook dijalankan:

- `$CLAUDE_PROJECT_DIR` : akar proyek. Bungkus dalam tanda kutip untuk menangani path dengan spasi.
- `${CLAUDE_PLUGIN_ROOT}` : direktori akar plugin, untuk skrip yang dibundel dengan plugin.

Skrip proyek

Contoh ini menggunakan `$CLAUDE_PROJECT_DIR` untuk menjalankan pemeriksa gaya dari direktori `.claude/hooks/` proyek setelah pemanggilan alat `Write` atau `Edit` apa pun:

```
{
  "hooks": {
    "PostToolUse": [
      {
        "matcher": "Write|Edit",
        "hooks": [
          {
            "type": "command",
            "command": "\"$CLAUDE_PROJECT_DIR\"/.claude/hooks/check-style.sh"
          }
        ]
      }
    ]
  }
}
```

Skrip plugin

Tentukan hooks plugin dalam `hooks/hooks.json` dengan bidang `description` tingkat atas opsional. Ketika plugin diaktifkan, hooks-nya bergabung dengan hooks pengguna dan proyek Anda.

Contoh ini menjalankan skrip pemformatan yang dibundel dengan plugin:

```
{
  "description": "Automatic code formatting",
  "hooks": {
    "PostToolUse": [
      {
        "matcher": "Write|Edit",
        "hooks": [
          {
            "type": "command",
            "command": "${CLAUDE_PLUGIN_ROOT}/scripts/format.sh",
            "timeout": 30
          }
        ]
      }
    ]
  }
}
```

Lihat plugin components reference untuk detail tentang membuat hooks plugin.

Hooks dalam skills dan agents

Selain file pengaturan dan plugin, hooks dapat didefinisikan langsung dalam skills dan subagents menggunakan frontmatter. Hooks ini dibatasi pada siklus hidup komponen dan hanya dijalankan ketika komponen itu aktif.

Semua hook events didukung. Untuk subagents, hooks `Stop` secara otomatis dikonversi ke `SubagentStop` karena itu adalah event yang dijalankan ketika subagent selesai.

Hooks menggunakan format konfigurasi yang sama seperti hooks berbasis pengaturan tetapi dibatasi pada masa hidup komponen dan dibersihkan saat selesai.

Skill ini mendefinisikan hook `PreToolUse` yang menjalankan skrip validasi keamanan sebelum setiap perintah `Bash`:

```
---
name: secure-operations
description: Perform operations with security checks
hooks:
  PreToolUse:
    - matcher: "Bash"
      hooks:
        - type: command
          command: "./scripts/security-check.sh"
---
```

Agents menggunakan format yang sama dalam frontmatter YAML mereka.

Menu `/hooks`

Ketik `/hooks` di Claude Code untuk membuka manajer hooks interaktif, di mana Anda dapat melihat, menambah, dan menghapus hooks tanpa mengedit file pengaturan secara langsung. Untuk panduan langkah demi langkah, lihat Set up your first hook dalam panduan.

Setiap hook dalam menu diberi label dengan awalan bracket yang menunjukkan sumbernya:

- `[User]` : dari `~/.claude/settings.json`
- `[Project]` : dari `.claude/settings.json`
- `[Local]` : dari `.claude/settings.local.json`
- `[Plugin]` : dari `hooks/hooks.json` plugin, hanya baca

Nonaktifkan atau hapus hooks

Untuk menghapus hook, hapus entrinya dari file JSON pengaturan, atau gunakan menu `/hooks` dan pilih hook untuk menghapusnya.

Untuk menonaktifkan semua hooks sementara tanpa menghapusnya, atur `"disableAllHooks": true` dalam file pengaturan Anda atau gunakan toggle dalam menu `/hooks`. Tidak ada cara untuk menonaktifkan hook individual sambil menyimpannya dalam konfigurasi.

Pengaturan `disableAllHooks` menghormati hierarki pengaturan terkelola. Jika administrator telah mengonfigurasi hooks melalui pengaturan kebijakan terkelola, `disableAllHooks` yang diatur dalam pengaturan pengguna, proyek, atau lokal tidak dapat menonaktifkan hooks terkelola tersebut. Hanya `disableAllHooks` yang diatur pada tingkat pengaturan terkelola yang dapat menonaktifkan hooks terkelola.

Pengeditan langsung ke hooks dalam file pengaturan tidak berlaku segera. Claude Code menangkap snapshot hooks saat startup dan menggunakannya sepanjang sesi. Ini mencegah modifikasi hook yang berbahaya atau tidak disengaja dari berlaku di tengah-sesi tanpa ulasan Anda. Jika hooks dimodifikasi secara eksternal, Claude Code memperingatkan Anda dan memerlukan ulasan dalam menu `/hooks` sebelum perubahan diterapkan.

Hook input dan output

Command hooks menerima data JSON melalui stdin dan mengkomunikasikan hasil melalui kode keluar, stdout, dan stderr. HTTP hooks menerima JSON yang sama sebagai badan permintaan POST dan mengkomunikasikan hasil melalui badan respons HTTP. Bagian ini mencakup bidang dan perilaku yang umum untuk semua event. Setiap bagian event di bawah Hook events mencakup skema input spesifiknya dan opsi kontrol keputusan.

Bidang input umum

Semua hook events menerima bidang-bidang ini sebagai JSON, selain bidang spesifik event yang didokumentasikan di setiap bagian hook event. Untuk command hooks, JSON ini tiba melalui stdin. Untuk HTTP hooks, itu tiba sebagai badan permintaan POST.

Bidang	Deskripsi
`session_id`	Pengenal sesi saat ini
`transcript_path`	Path ke JSON percakapan
`cwd`	Direktori kerja saat hook dipanggil
`permission_mode`	Mode izin saat ini: `"default"`, `"plan"`, `"acceptEdits"`, `"dontAsk"`, atau `"bypassPermissions"`
`hook_event_name`	Nama event yang dijalankan

Saat berjalan dengan `--agent` atau di dalam subagent, dua bidang tambahan disertakan:

Bidang	Deskripsi
`agent_id`	Pengenal unik untuk subagent. Hadir hanya ketika hook dijalankan di dalam pemanggilan subagent. Gunakan ini untuk membedakan pemanggilan hook subagent dari pemanggilan thread utama.
`agent_type`	Nama agen (misalnya, `"Explore"` atau `"security-reviewer"`). Hadir ketika sesi menggunakan `--agent` atau hook dijalankan di dalam subagent. Untuk subagents, tipe subagent mengambil alih nilai `--agent` sesi.

Misalnya, hook `PreToolUse` untuk perintah Bash menerima ini di stdin:

```
{
  "session_id": "abc123",
  "transcript_path": "/home/user/.claude/projects/.../transcript.jsonl",
  "cwd": "/home/user/my-project",
  "permission_mode": "default",
  "hook_event_name": "PreToolUse",
  "tool_name": "Bash",
  "tool_input": {
    "command": "npm test"
  }
}
```

Bidang `tool_name` dan `tool_input` spesifik untuk event. Setiap bagian hook event mendokumentasikan bidang tambahan untuk event itu.

Output kode keluar

Kode keluar dari perintah hook Anda memberitahu Claude Code apakah tindakan harus dilanjutkan, diblokir, atau diabaikan.

Exit 0 berarti sukses. Claude Code mengurai stdout untuk bidang output JSON. Output JSON hanya diproses pada exit 0. Untuk sebagian besar event, stdout hanya ditampilkan dalam mode verbose (`Ctrl+O`). Pengecualiannya adalah `UserPromptSubmit` dan `SessionStart`, di mana stdout ditambahkan sebagai konteks yang dapat dilihat dan ditindaklanjuti oleh Claude.

Exit 2 berarti kesalahan blocking. Claude Code mengabaikan stdout dan JSON apa pun di dalamnya. Sebagai gantinya, teks stderr diumpankan kembali ke Claude sebagai pesan kesalahan. Efeknya tergantung pada event: `PreToolUse` memblokir pemanggilan alat, `UserPromptSubmit` menolak prompt, dan sebagainya. Lihat exit code 2 behavior untuk daftar lengkap.

Kode keluar lainnya adalah kesalahan non-blocking. stderr ditampilkan dalam mode verbose (`Ctrl+O`) dan eksekusi berlanjut.

Misalnya, skrip perintah hook yang memblokir perintah Bash yang berbahaya:

```
#!/bin/bash
## Membaca input JSON dari stdin, memeriksa perintah
command=$(jq -r '.tool_input.command' < /dev/stdin)

if [[ "$command" == rm* ]]; then
  echo "Blocked: rm commands are not allowed" >&2
  exit 2  # Blocking error: tool call is prevented
fi

exit 0  # Success: tool call proceeds
```

Perilaku exit code 2 per event

Exit code 2 adalah cara hook menandakan “berhenti, jangan lakukan ini.” Efeknya tergantung pada event, karena beberapa event mewakili tindakan yang dapat diblokir (seperti pemanggilan alat yang belum terjadi) dan yang lain mewakili hal-hal yang sudah terjadi atau tidak dapat dicegah.

Hook event	Dapat diblokir?	Apa yang terjadi pada exit 2
`PreToolUse`	Ya	Memblokir pemanggilan alat
`PermissionRequest`	Ya	Menolak izin
`UserPromptSubmit`	Ya	Memblokir pemrosesan prompt dan menghapus prompt

Hook event	Dapat diblokir?	Apa yang terjadi pada exit 2
`Stop`	Ya	Mencegah Claude berhenti, melanjutkan percakapan
`SubagentStop`	Ya	Mencegah subagent berhenti
`TeammateIdle`	Ya	Mencegah rekan kerja menjadi idle (rekan kerja terus bekerja)
`TaskCompleted`	Ya	Mencegah tugas ditandai sebagai selesai
`ConfigChange`	Ya	Memblokir perubahan konfigurasi dari berlaku (kecuali `policy_settings`)
`PostToolUse`	Tidak	Menampilkan stderr ke Claude (alat sudah berjalan)
`PostToolUseFailure`	Tidak	Menampilkan stderr ke Claude (alat sudah gagal)
`Notification`	Tidak	Menampilkan stderr ke pengguna saja
`SubagentStart`	Tidak	Menampilkan stderr ke pengguna saja
`SessionStart`	Tidak	Menampilkan stderr ke pengguna saja
`SessionEnd`	Tidak	Menampilkan stderr ke pengguna saja
`PreCompact`	Tidak	Menampilkan stderr ke pengguna saja

Hook event	Dapat diblokir?	Apa yang terjadi pada exit 2
`WorktreeCreate`	Ya	Kode keluar non-zero apa pun menyebabkan pembuatan worktree gagal
`WorktreeRemove`	Tidak	Kegagalan dicatat dalam mode debug saja
`InstructionsLoaded`	Tidak	Kode keluar diabaikan

Penanganan respons HTTP

HTTP hooks menggunakan kode status HTTP dan badan respons alih-alih kode keluar dan stdout:

- **2xx dengan badan kosong**: sukses, setara dengan exit code 0 tanpa output
- **2xx dengan badan teks biasa**: sukses, teks ditambahkan sebagai konteks
- **2xx dengan badan JSON**: sukses, diurai menggunakan skema JSON output yang sama seperti command hooks
- **Status non-2xx**: kesalahan non-blocking, eksekusi berlanjut
- **Kegagalan koneksi atau timeout**: kesalahan non-blocking, eksekusi berlanjut

Tidak seperti command hooks, HTTP hooks tidak dapat menandakan kesalahan blocking hanya melalui kode status. Untuk memblokir pemanggilan alat atau menolak izin, kembalikan respons 2xx dengan badan JSON yang berisi bidang keputusan yang sesuai.

Output JSON

Kode keluar memungkinkan Anda mengizinkan atau memblokir, tetapi output JSON memberikan kontrol yang lebih halus. Alih-alih keluar dengan kode 2 untuk memblokir, keluar 0 dan cetak objek JSON ke stdout. Claude Code membaca bidang tertentu dari JSON itu untuk mengontrol perilaku, termasuk decision control untuk memblokir, mengizinkan, atau meningkatkan ke pengguna.

> **Note:**
>
> Anda harus memilih satu pendekatan per hook, bukan keduanya: gunakan kode keluar saja untuk signaling, atau keluar 0 dan cetak JSON untuk kontrol terstruktur. Claude Code hanya memproses JSON pada exit 0. Jika Anda keluar 2, JSON apa pun diabaikan.

Stdout hook Anda harus berisi hanya objek JSON. Jika profil shell Anda mencetak teks saat startup, itu dapat mengganggu parsing JSON. Lihat JSON validation failed dalam panduan troubleshooting.

Objek JSON mendukung tiga jenis bidang:

- **Bidang universal** seperti `continue` bekerja di semua event. Ini tercantum dalam tabel di bawah.
- **Top-level `decision` dan `reason`** digunakan oleh beberapa event untuk memblokir atau memberikan umpan balik.
- `hookSpecificOutput` adalah objek bersarang untuk event yang memerlukan kontrol yang lebih kaya. Ini memerlukan bidang `hookEventName` yang diatur ke nama event.

Bidang	Default	Deskripsi
`continue`	`true`	Jika `false`, Claude berhenti memproses sepenuhnya setelah hook dijalankan. Mengambil alih bidang keputusan spesifik event apa pun
`stopReason`	tidak ada	Pesan ditampilkan ke pengguna saat `continue` adalah `false`. Tidak ditampilkan ke Claude
`suppressOutput`	`false`	Jika `true`, menyembunyikan stdout dari output mode verbose
`systemMessage`	tidak ada	Pesan peringatan ditampilkan ke pengguna

Untuk menghentikan Claude sepenuhnya terlepas dari tipe event:

```
{ "continue": false, "stopReason": "Build failed, fix errors before continuing" }
```

Kontrol keputusan

Tidak setiap event mendukung pemblokiran atau kontrol perilaku melalui JSON. Event yang melakukannya masing-masing menggunakan set bidang yang berbeda untuk mengekspresikan keputusan itu. Gunakan tabel ini sebagai referensi cepat sebelum menulis hook:

Events	Pola keputusan	Bidang kunci
UserPromptSubmit, PostToolUse, PostToolUseFailure, Stop, SubagentStop, ConfigChange	Top-level `decision`	`decision: "block"`, `reason`
TeammateIdle, TaskCompleted	Kode keluar atau `continue: false`	Exit code 2 memblokir tindakan dengan umpan balik stderr. JSON `{"continue": false, "stopReason": "..."}` juga menghentikan rekan kerja sepenuhnya, cocok dengan perilaku hook `Stop`
PreToolUse	`hookSpecificOutput`	`permissionDecision` (allow/deny/ask), `permissionDecisionReason`
PermissionRequest	`hookSpecificOutput`	`decision.behavior` (allow/deny)
WorktreeCreate	stdout path	Hook mencetak path absolut ke direktori worktree yang dibuat. Exit non-zero gagal membuat
WorktreeRemove, Notification, SessionEnd, PreCompact, InstructionsLoaded	Tidak ada	Tidak ada kontrol keputusan. Digunakan untuk efek samping seperti logging atau cleanup

Berikut adalah contoh setiap pola dalam aksi:

Top-level decision

Digunakan oleh `UserPromptSubmit`, `PostToolUse`, `PostToolUseFailure`, `Stop`, `SubagentStop`, dan `ConfigChange`. Satu-satunya nilai adalah `"block"`. Untuk mengizinkan tindakan dilanjutkan, hilangkan `decision` dari JSON Anda, atau keluar 0 tanpa JSON apa pun:

```
{
  "decision": "block",
  "reason": "Test suite must pass before proceeding"
}
```

PreToolUse

Menggunakan `hookSpecificOutput` untuk kontrol yang lebih kaya: izinkan, tolak, atau tingkatkan ke pengguna. Anda juga dapat memodifikasi input alat sebelum dijalankan atau menyuntikkan konteks tambahan untuk Claude. Lihat PreToolUse decision control untuk set lengkap opsi.

```
{
  "hookSpecificOutput": {
    "hookEventName": "PreToolUse",
    "permissionDecision": "deny",
    "permissionDecisionReason": "Database writes are not allowed"
  }
}
```

PermissionRequest

Menggunakan `hookSpecificOutput` untuk mengizinkan atau menolak permintaan izin atas nama pengguna. Saat mengizinkan, Anda juga dapat memodifikasi input alat atau menerapkan aturan izin sehingga pengguna tidak diminta lagi. Lihat PermissionRequest decision control untuk set lengkap opsi.

```
{
  "hookSpecificOutput": {
    "hookEventName": "PermissionRequest",
    "decision": {
      "behavior": "allow",
      "updatedInput": {
        "command": "npm run lint"
      }
    }
  }
}
```

Untuk contoh yang diperluas termasuk validasi perintah Bash, pemfilteran prompt, dan skrip persetujuan otomatis, lihat What you can automate dalam panduan dan Bash command validator reference implementation.

Hook events

Setiap event sesuai dengan titik dalam siklus hidup Claude Code di mana hooks dapat dijalankan. Bagian-bagian di bawah diurutkan untuk mencocokkan siklus hidup: dari pengaturan sesi melalui loop agentic ke akhir sesi. Setiap bagian menjelaskan kapan event dijalankan, matcher apa yang didukungnya, JSON input yang diterima, dan cara mengontrol perilaku melalui output.

SessionStart

Dijalankan ketika Claude Code memulai sesi baru atau melanjutkan sesi yang ada. Berguna untuk memuat konteks pengembangan seperti masalah yang ada atau perubahan terbaru pada codebase Anda, atau menyiapkan variabel lingkungan. Untuk konteks statis yang tidak memerlukan skrip, gunakan CLAUDE.md sebagai gantinya.

SessionStart dijalankan pada setiap sesi, jadi jaga hooks ini tetap cepat. Hanya hooks `type: "command"` yang didukung.

Nilai matcher sesuai dengan cara sesi dimulai:

Matcher	Kapan dijalankan
`startup`	Sesi baru
`resume`	`--resume`, `--continue`, atau `/resume`

Matcher	Kapan dijalankan
`clear`	`/clear`
`compact`	Pemadatan otomatis atau manual

Input SessionStart

Selain bidang input umum, hooks SessionStart menerima `source`, `model`, dan secara opsional `agent_type`. Bidang `source` menunjukkan bagaimana sesi dimulai: `"startup"` untuk sesi baru, `"resume"` untuk sesi yang dilanjutkan, `"clear"` setelah `/clear`, atau `"compact"` setelah pemadatan. Bidang `model` berisi pengenal model. Jika Anda memulai Claude Code dengan `claude --agent <name>`, bidang `agent_type` berisi nama agen.

```
{
  "session_id": "abc123",
  "transcript_path": "/Users/.../.claude/projects/.../
00893aaf-19fa-41d2-8238-13269b9b3ca0.jsonl",
  "cwd": "/Users/...",
  "permission_mode": "default",
  "hook_event_name": "SessionStart",
  "source": "startup",
  "model": "claude-sonnet-4-6"
}
```

Kontrol keputusan SessionStart

Teks apa pun yang dicetak skrip hook ke stdout ditambahkan sebagai konteks untuk Claude. Selain bidang output JSON yang tersedia untuk semua hooks, Anda dapat mengembalikan bidang spesifik event ini:

Bidang	Deskripsi
`additionalContext`	String ditambahkan ke konteks Claude. Nilai hooks berganda digabungkan

```
{
  "hookSpecificOutput": {
    "hookEventName": "SessionStart",
    "additionalContext": "My additional context here"
  }
}
```

Pertahankan variabel lingkungan

Hooks SessionStart memiliki akses ke variabel lingkungan `CLAUDE_ENV_FILE`, yang menyediakan path file di mana Anda dapat mempertahankan variabel lingkungan untuk perintah Bash berikutnya.

Untuk menetapkan variabel lingkungan individual, tulis pernyataan `export` ke `CLAUDE_ENV_FILE`. Gunakan append (`>>`) untuk mempertahankan variabel yang ditetapkan oleh hooks lain:

```
#!/bin/bash

if [ -n "$CLAUDE_ENV_FILE" ]; then
  echo 'export NODE_ENV=production' >> "$CLAUDE_ENV_FILE"
  echo 'export DEBUG_LOG=true' >> "$CLAUDE_ENV_FILE"
  echo 'export PATH="$PATH:./node_modules/.bin"' >> "$CLAUDE_ENV_FILE"
fi

exit 0
```

Untuk menangkap semua perubahan lingkungan dari perintah setup, bandingkan variabel yang diekspor sebelum dan sesudah:

```
#!/bin/bash

ENV_BEFORE=$(export -p | sort)

## Jalankan perintah setup Anda yang memodifikasi lingkungan
source ~/.nvm/nvm.sh
nvm use 20

if [ -n "$CLAUDE_ENV_FILE" ]; then
  ENV_AFTER=$(export -p | sort)
  comm -13 <(echo "$ENV_BEFORE") <(echo "$ENV_AFTER") >> "$CLAUDE_ENV_FILE"
fi

exit 0
```

Variabel apa pun yang ditulis ke file ini akan tersedia di semua perintah Bash berikutnya yang dijalankan Claude Code selama sesi.

> **Note:**
>
> `CLAUDE_ENV_FILE` tersedia untuk hooks SessionStart. Tipe hook lainnya tidak memiliki akses ke variabel ini.

InstructionsLoaded

Dijalankan ketika file `CLAUDE.md` atau `.claude/rules/*.md` dimuat ke dalam konteks. Event ini dijalankan saat startup sesi untuk file yang dimuat dengan eager dan lagi nanti ketika file dimuat dengan lazy, misalnya ketika Claude mengakses subdirektori yang berisi `CLAUDE.md` bersarang atau ketika aturan bersyarat dengan frontmatter `paths:` cocok. Hook tidak mendukung pemblokiran atau kontrol keputusan. Itu dijalankan secara asinkron untuk tujuan observabilitas.

InstructionsLoaded tidak mendukung matchers dan dijalankan pada setiap kemunculan load.

Input InstructionsLoaded

Selain bidang input umum, hooks InstructionsLoaded menerima bidang-bidang ini:

Bidang	Deskripsi
`file_path`	Path absolut ke file instruksi yang dimuat
`memory_type`	Cakupan file: `"User"`, `"Project"`, `"Local"`, atau `"Managed"`
`load_reason`	Mengapa file dimuat: `"session_start"`, `"nested_traversal"`, `"path_glob_match"`, atau `"include"`
`globs`	Pola glob path dari frontmatter `paths:` file, jika ada. Hadir hanya untuk load `path_glob_match`
`trigger_file_path`	Path ke file yang akses memicu load ini, untuk lazy loads
`parent_file_path`	Path ke file instruksi induk yang menyertakan ini, untuk load `include`

```
{
  "session_id": "abc123",
  "transcript_path": "/Users/.../.claude/projects/.../transcript.jsonl",
  "cwd": "/Users/my-project",
  "permission_mode": "default",
  "hook_event_name": "InstructionsLoaded",
  "file_path": "/Users/my-project/CLAUDE.md",
  "memory_type": "Project",
  "load_reason": "session_start"
}
```

Kontrol keputusan InstructionsLoaded

Hooks InstructionsLoaded tidak memiliki kontrol keputusan. Mereka tidak dapat memblokir atau memodifikasi pemuatan instruksi. Gunakan event ini untuk audit logging, compliance tracking, atau observabilitas.

UserPromptSubmit

Dijalankan ketika pengguna mengirimkan prompt, sebelum Claude memproses. Ini memungkinkan Anda menambahkan konteks tambahan berdasarkan prompt/percakapan, memvalidasi prompts, atau memblokir jenis prompts tertentu.

Input UserPromptSubmit

Selain bidang input umum, hooks UserPromptSubmit menerima bidang `prompt` yang berisi teks yang dikirimkan pengguna.

```
{
  "session_id": "abc123",
  "transcript_path": "/Users/.../.claude/projects/.../
00893aaf-19fa-41d2-8238-13269b9b3ca0.jsonl",
  "cwd": "/Users/...",
  "permission_mode": "default",
  "hook_event_name": "UserPromptSubmit",
  "prompt": "Write a function to calculate the factorial of a number"
}
```

Kontrol keputusan UserPromptSubmit

Hooks `UserPromptSubmit` dapat mengontrol apakah prompt pengguna diproses dan menambahkan konteks. Semua bidang output JSON tersedia.

Ada dua cara untuk menambahkan konteks ke percakapan pada exit code 0:

- **Plain text stdout**: teks non-JSON apa pun yang ditulis ke stdout ditambahkan sebagai konteks
- **JSON dengan `additionalContext`**: gunakan format JSON di bawah untuk kontrol lebih. Bidang `additionalContext` ditambahkan sebagai konteks

Plain stdout ditampilkan sebagai output hook dalam transkrip. Bidang `additionalContext` ditambahkan lebih diskrit.

Untuk memblokir prompt, kembalikan objek JSON dengan `decision` diatur ke `"block"`:

Bidang	Deskripsi
`decision`	`"block"` mencegah prompt diproses dan menghapusnya dari konteks. Hilangkan untuk mengizinkan prompt dilanjutkan
`reason`	Ditampilkan ke pengguna saat `decision` adalah `"block"`. Tidak ditambahkan ke konteks
`additionalContext`	String ditambahkan ke konteks Claude

```
{
  "decision": "block",
  "reason": "Explanation for decision",
  "hookSpecificOutput": {
    "hookEventName": "UserPromptSubmit",
    "additionalContext": "My additional context here"
  }
}
```

> **Note:**
>
> Format JSON tidak diperlukan untuk kasus penggunaan sederhana. Untuk menambahkan konteks, Anda dapat mencetak teks biasa ke stdout dengan exit code 0. Gunakan JSON ketika Anda perlu memblokir prompts atau menginginkan kontrol yang lebih terstruktur.

PreToolUse

Dijalankan setelah Claude membuat parameter alat dan sebelum memproses pemanggilan alat. Cocok pada nama alat: `Bash`, `Edit`, `Write`, `Read`, `Glob`, `Grep`, `Agent`, `WebFetch`, `WebSearch`, dan nama alat MCP apa pun.

Gunakan PreToolUse decision control untuk mengizinkan, menolak, atau meminta izin untuk menggunakan alat.

Input PreToolUse

Selain bidang input umum, hooks PreToolUse menerima `tool_name`, `tool_input`, dan `tool_use_id`. Bidang `tool_input` tergantung pada alat:

Bash

Menjalankan perintah shell.

Bidang	Tipe	Contoh	Deskripsi
`command`	string	`"npm test"`	Perintah shell untuk dijalankan
`description`	string	`"Run test suite"`	Deskripsi opsional tentang apa yang dilakukan perintah

Bidang	Tipe	Contoh	Deskripsi
`timeout`	number	`120000`	Timeout opsional dalam milidetik
`run_in_background`	boolean	`false`	Apakah menjalankan perintah di latar belakang

Write

Membuat atau menimpa file.

Bidang	Tipe	Contoh	Deskripsi
`file_path`	string	`"/path/to/file.txt"`	Path absolut ke file untuk ditulis
`content`	string	`"file content"`	Konten untuk ditulis ke file

Edit

Mengganti string dalam file yang ada.

Bidang	Tipe	Contoh	Deskripsi
`file_path`	string	`"/path/to/file.txt"`	Path absolut ke file untuk diedit
`old_string`	string	`"original text"`	Teks untuk dicari dan diganti
`new_string`	string	`"replacement text"`	Teks pengganti
`replace_all`	boolean	`false`	Apakah mengganti semua kemunculan

Read

Membaca konten file.

Bidang	Tipe	Contoh	Deskripsi
`file_path`	string	`"/path/to/file.txt"`	Path absolut ke file untuk dibaca
`offset`	number	`10`	Nomor baris opsional untuk mulai membaca dari
`limit`	number	`50`	Jumlah baris opsional untuk dibaca

Glob

Menemukan file yang cocok dengan pola glob.

Bidang	Tipe	Contoh	Deskripsi
`pattern`	string	`"**/*.ts"`	Pola glob untuk mencocokkan file terhadap
`path`	string	`"/path/to/dir"`	Direktori opsional untuk dicari. Default ke direktori kerja saat ini

Grep

Mencari konten file dengan ekspresi reguler.

Bidang	Tipe	Contoh	Deskripsi
`pattern`	string	`"TODO.*fix"`	Pola ekspresi reguler untuk dicari
`path`	string	`"/path/to/dir"`	File atau direktori opsional untuk dicari
`glob`	string	`"*.ts"`	Pola glob opsional untuk memfilter file

Bidang	Tipe	Contoh	Deskripsi
`output_mode`	string	`"content"`	`"content"`, `"files_with_matches"`, atau `"count"`. Default ke `"files_with_matches"`
`-i`	boolean	`true`	Pencarian case insensitive
`multiline`	boolean	`false`	Aktifkan pencocokan multiline

WebFetch

Mengambil dan memproses konten web.

Bidang	Tipe	Contoh	Deskripsi
`url`	string	`"https://example.com/api"`	URL untuk mengambil konten dari
`prompt`	string	`"Extract the API endpoints"`	Prompt untuk dijalankan pada konten yang diambil

WebSearch

Mencari web.

Bidang	Tipe	Contoh	Deskripsi
`query`	string	`"react hooks best practices"`	Kueri pencarian
`allowed_domains`	array	`["docs.example.com"]`	Opsional: hanya sertakan hasil dari domain ini
`blocked_domains`	array	`["spam.example.com"]`	Opsional: kecualikan hasil dari domain ini

Agent

Spawn subagent.

Bidang	Tipe	Contoh	Deskripsi
`prompt`	string	`"Find all API endpoints"`	Tugas untuk agen lakukan
`description`	string	`"Find API endpoints"`	Deskripsi singkat tugas
`subagent_type`	string	`"Explore"`	Tipe agen khusus untuk digunakan
`model`	string	`"sonnet"`	Alias model opsional untuk menimpa default

Kontrol keputusan PreToolUse

Hooks `PreToolUse` dapat mengontrol apakah pemanggilan alat dilanjutkan. Tidak seperti hooks lain yang menggunakan bidang `decision` tingkat atas, PreToolUse mengembalikan keputusannya di dalam objek `hookSpecificOutput`. Ini memberikannya kontrol yang lebih kaya: tiga hasil (izinkan, tolak, atau tanya) ditambah kemampuan untuk memodifikasi input alat sebelum eksekusi.

Bidang	Deskripsi
`permissionDecision`	`"allow"` melewati sistem izin, `"deny"` mencegah pemanggilan alat, `"ask"` meminta pengguna untuk mengkonfirmasi
`permissionDecisionReason`	Untuk `"allow"` dan `"ask"`, ditampilkan ke pengguna tetapi bukan Claude. Untuk `"deny"`, ditampilkan ke Claude
`updatedInput`	Memodifikasi parameter input alat sebelum eksekusi. Gabungkan dengan `"allow"` untuk persetujuan otomatis, atau `"ask"` untuk menampilkan input yang dimodifikasi ke pengguna
`additionalContext`	String ditambahkan ke konteks Claude sebelum alat dijalankan

```
{
  "hookSpecificOutput": {
    "hookEventName": "PreToolUse",
    "permissionDecision": "allow",
    "permissionDecisionReason": "My reason here",
    "updatedInput": {
      "field_to_modify": "new value"
    },
    "additionalContext": "Current environment: production. Proceed with caution."
  }
}
```

> **Note:**
>
> PreToolUse sebelumnya menggunakan bidang `decision` dan `reason` tingkat atas, tetapi ini sudah usang untuk event ini. Gunakan `hookSpecificOutput.permissionDecision` dan `hookSpecificOutput.permissionDecisionReason` sebagai gantinya. Nilai usang `"approve"` dan `"block"` memetakan ke `"allow"` dan `"deny"` masing-masing. Event lain seperti PostToolUse dan Stop terus menggunakan `decision` dan `reason` tingkat atas sebagai format saat ini mereka.

PermissionRequest

Dijalankan ketika pengguna ditampilkan dialog izin. Gunakan PermissionRequest decision control untuk mengizinkan atau menolak atas nama pengguna.

Cocok pada nama alat, nilai yang sama seperti PreToolUse.

Input PermissionRequest

Hooks PermissionRequest menerima bidang `tool_name` dan `tool_input` seperti hooks PreToolUse, tetapi tanpa `tool_use_id`. Array `permission_suggestions` opsional berisi opsi "selalu izinkan" yang biasanya dilihat pengguna dalam dialog izin. Perbedaannya adalah kapan hook dijalankan: hooks PermissionRequest dijalankan ketika dialog izin akan ditampilkan kepada pengguna, sementara hooks PreToolUse dijalankan sebelum eksekusi alat terlepas dari status izin.

```
{
  "session_id": "abc123",
  "transcript_path": "/Users/.../.claude/projects/.../
00893aaf-19fa-41d2-8238-13269b9b3ca0.jsonl",
  "cwd": "/Users/...",
  "permission_mode": "default",
  "hook_event_name": "PermissionRequest",
  "tool_name": "Bash",
  "tool_input": {
    "command": "rm -rf node_modules",
    "description": "Remove node_modules directory"
  },
  "permission_suggestions": [
    { "type": "toolAlwaysAllow", "tool": "Bash" }
  ]
}
```

Kontrol keputusan PermissionRequest

Hooks `PermissionRequest` dapat mengizinkan atau menolak permintaan izin. Selain bidang output JSON yang tersedia untuk semua hooks, skrip hook Anda dapat mengembalikan objek `decision` dengan bidang spesifik event ini:

Bidang	Deskripsi
`behavior`	`"allow"` memberikan izin, `"deny"` menolaknya
`updatedInput`	Untuk `"allow"` saja: memodifikasi parameter input alat sebelum eksekusi
`updatedPermissions`	Untuk `"allow"` saja: menerapkan pembaruan aturan izin, setara dengan pengguna memilih opsi “selalu izinkan”
`message`	Untuk `"deny"` saja: memberitahu Claude mengapa izin ditolak
`interrupt`	Untuk `"deny"` saja: jika `true`, menghentikan Claude

```
{
  "hookSpecificOutput": {
    "hookEventName": "PermissionRequest",
    "decision": {
      "behavior": "allow",
      "updatedInput": {
        "command": "npm run lint"
      }
    }
  }
}
```

PostToolUse

Dijalankan segera setelah alat selesai dengan sukses.

Cocok pada nama alat, nilai yang sama seperti PreToolUse.

Input PostToolUse

Hooks `PostToolUse` dijalankan setelah alat sudah dijalankan dengan sukses. Input mencakup `tool_input`, argumen yang dikirim ke alat, dan `tool_response`, hasil yang dikembalikan. Skema yang tepat untuk keduanya tergantung pada alat.

```
{
  "session_id": "abc123",
  "transcript_path": "/Users/.../.claude/projects/.../
00893aaf-19fa-41d2-8238-13269b9b3ca0.jsonl",
  "cwd": "/Users/...",
  "permission_mode": "default",
  "hook_event_name": "PostToolUse",
  "tool_name": "Write",
  "tool_input": {
    "file_path": "/path/to/file.txt",
    "content": "file content"
  },
  "tool_response": {
    "filePath": "/path/to/file.txt",
    "success": true
  },
  "tool_use_id": "toolu_01ABC123..."
}
```

Kontrol keputusan PostToolUse

Hooks `PostToolUse` dapat memberikan umpan balik ke Claude setelah eksekusi alat. Selain bidang output JSON yang tersedia untuk semua hooks, skrip hook Anda dapat mengembalikan bidang spesifik event ini:

Bidang	Deskripsi
`decision`	`"block"` meminta Claude dengan `reason`. Hilangkan untuk mengizinkan tindakan dilanjutkan
`reason`	Penjelasan ditampilkan ke Claude saat `decision` adalah `"block"`
`additionalContext`	Konteks tambahan untuk Claude pertimbangkan
`updatedMCPToolOutput`	Untuk alat MCP saja: mengganti output alat dengan nilai yang disediakan

```
{
  "decision": "block",
  "reason": "Explanation for decision",
  "hookSpecificOutput": {
    "hookEventName": "PostToolUse",
    "additionalContext": "Additional information for Claude"
  }
}
```

PostToolUseFailure

Dijalankan ketika eksekusi alat gagal. Event ini dijalankan untuk pemanggilan alat yang melempar kesalahan atau mengembalikan hasil kegagalan. Gunakan ini untuk mencatat kegagalan, mengirim alert, atau memberikan umpan balik korektif ke Claude.

Cocok pada nama alat, nilai yang sama seperti PreToolUse.

Input PostToolUseFailure

Hooks PostToolUseFailure menerima bidang `tool_name` dan `tool_input` yang sama seperti PostToolUse, bersama dengan informasi kesalahan sebagai bidang tingkat atas:

```
{
  "session_id": "abc123",
  "transcript_path": "/Users/.../.claude/projects/.../
00893aaf-19fa-41d2-8238-13269b9b3ca0.jsonl",
  "cwd": "/Users/...",
  "permission_mode": "default",
  "hook_event_name": "PostToolUseFailure",
  "tool_name": "Bash",
  "tool_input": {
    "command": "npm test",
    "description": "Run test suite"
  },
  "tool_use_id": "toolu_01ABC123...",
  "error": "Command exited with non-zero status code 1",
  "is_interrupt": false
}
```

Bidang	Deskripsi
`error`	String menjelaskan apa yang salah
`is_interrupt`	Boolean opsional menunjukkan apakah kegagalan disebabkan oleh interupsi pengguna

Kontrol keputusan PostToolUseFailure

Hooks `PostToolUseFailure` dapat memberikan konteks ke Claude setelah kegagalan alat. Selain bidang output JSON yang tersedia untuk semua hooks, skrip hook Anda dapat mengembalikan bidang spesifik event ini:

Bidang	Deskripsi
`additionalContext`	Konteks tambahan untuk Claude pertimbangkan bersama kesalahan

```
{
  "hookSpecificOutput": {
    "hookEventName": "PostToolUseFailure",
    "additionalContext": "Additional information about the failure for Claude"
  }
}
```

Notification

Dijalankan ketika Claude Code mengirimkan notifikasi. Cocok pada tipe notifikasi: `permission_prompt`, `idle_prompt`, `auth_success`, `elicitation_dialog`. Hilangkan matcher untuk menjalankan hooks untuk semua tipe notifikasi.

Gunakan matchers terpisah untuk menjalankan handler berbeda tergantung pada tipe notifikasi. Konfigurasi ini memicu skrip alert khusus izin ketika Claude memerlukan persetujuan izin dan notifikasi berbeda ketika Claude telah idle:

```
{
  "hooks": {
    "Notification": [
      {
        "matcher": "permission_prompt",
        "hooks": [
          {
            "type": "command",
            "command": "/path/to/permission-alert.sh"
          }
        ]
      },
      {
        "matcher": "idle_prompt",
        "hooks": [
          {
            "type": "command",
            "command": "/path/to/idle-notification.sh"
          }
        ]
      }
    ]
  }
}
```

Input Notification

Selain bidang input umum, hooks Notification menerima `message` dengan teks notifikasi, `title` opsional, dan `notification_type` menunjukkan tipe mana yang dijalankan.

```
{
  "session_id": "abc123",
  "transcript_path": "/Users/.../.claude/projects/.../
00893aaf-19fa-41d2-8238-13269b9b3ca0.jsonl",
  "cwd": "/Users/...",
  "permission_mode": "default",
  "hook_event_name": "Notification",
  "message": "Claude needs your permission to use Bash",
  "title": "Permission needed",
  "notification_type": "permission_prompt"
}
```

Hooks Notification tidak dapat memblokir atau memodifikasi notifikasi. Selain bidang output JSON yang tersedia untuk semua hooks, Anda dapat mengembalikan `additionalContext` untuk menambahkan konteks ke percakapan:

Bidang	Deskripsi
`additionalContext`	String ditambahkan ke konteks Claude

SubagentStart

Dijalankan ketika subagent Claude Code dispawn melalui alat Agent. Mendukung matchers untuk memfilter berdasarkan nama tipe agen (agen built-in seperti `Bash`, `Explore`, `Plan`, atau nama agen kustom dari `.claude/agents/`).

Input SubagentStart

Selain bidang input umum, hooks SubagentStart menerima `agent_id` dengan pengenal unik untuk subagent dan `agent_type` dengan nama agen (agen built-in seperti `"Bash"`, `"Explore"`, `"Plan"`, atau nama agen kustom).

```
{
  "session_id": "abc123",
  "transcript_path": "/Users/.../.claude/projects/.../
00893aaf-19fa-41d2-8238-13269b9b3ca0.jsonl",
  "cwd": "/Users/...",
  "permission_mode": "default",
  "hook_event_name": "SubagentStart",
  "agent_id": "agent-abc123",
  "agent_type": "Explore"
}
```

Hooks SubagentStart tidak dapat memblokir pembuatan subagent, tetapi mereka dapat menyuntikkan konteks ke dalam subagent. Selain bidang output JSON yang tersedia untuk semua hooks, Anda dapat mengembalikan:

Bidang	Deskripsi
`additionalContext`	String ditambahkan ke konteks subagent

```
{
  "hookSpecificOutput": {
    "hookEventName": "SubagentStart",
    "additionalContext": "Follow security guidelines for this task"
  }
}
```

SubagentStop

Dijalankan ketika subagent Claude Code telah selesai merespons. Cocok pada tipe agen, nilai yang sama seperti SubagentStart.

Input SubagentStop

Selain bidang input umum, hooks SubagentStop menerima `stop_hook_active`, `agent_id`, `agent_type`, `agent_transcript_path`, dan `last_assistant_message`. Bidang `agent_type` adalah nilai yang digunakan untuk pemfilteran matcher. `transcript_path` adalah transkrip sesi utama, sementara `agent_transcript_path` adalah transkrip suba-

gent sendiri yang disimpan dalam folder `subagents/` bersarang. Bidang `last_assistant_message` berisi konten teks respons akhir subagent, jadi hooks dapat mengaksesnya tanpa mengurai file transkrip.

```
{
  "session_id": "abc123",
  "transcript_path": "~/.claude/projects/.../abc123.jsonl",
  "cwd": "/Users/...",
  "permission_mode": "default",
  "hook_event_name": "SubagentStop",
  "stop_hook_active": false,
  "agent_id": "def456",
  "agent_type": "Explore",
  "agent_transcript_path": "~/.claude/projects/.../abc123/subagents/agent-
def456.jsonl",
  "last_assistant_message": "Analysis complete. Found 3 potential issues..."
}
```

Hooks SubagentStop menggunakan format kontrol keputusan yang sama seperti hooks Stop.

Stop

Dijalankan ketika agen Claude Code utama telah selesai merespons. Tidak dijalankan jika penghentian terjadi karena interupsi pengguna.

Input Stop

Selain bidang input umum, hooks Stop menerima `stop_hook_active` dan `last_assistant_message`. Bidang `stop_hook_active` adalah `true` ketika Claude Code sudah melanjutkan sebagai hasil dari stop hook. Periksa nilai ini atau proses transkrip untuk mencegah Claude Code berjalan tanpa batas. Bidang `last_assistant_message` berisi konten teks respons akhir Claude, jadi hooks dapat mengaksesnya tanpa mengurai file transkrip.

```
{
  "session_id": "abc123",
  "transcript_path": "~/.claude/projects/.../
00893aaf-19fa-41d2-8238-13269b9b3ca0.jsonl",
  "cwd": "/Users/...",
  "permission_mode": "default",
  "hook_event_name": "Stop",
  "stop_hook_active": true,
  "last_assistant_message": "I've completed the refactoring. Here's a summary..."
}
```

Kontrol keputusan Stop

Hooks `Stop` dan `SubagentStop` dapat mengontrol apakah Claude melanjutkan. Selain bidang output JSON yang tersedia untuk semua hooks, skrip hook Anda dapat mengembalikan bidang spesifik event ini:

Bidang	Deskripsi
`decision`	`"block"` mencegah Claude berhenti. Hilangkan untuk mengizinkan Claude berhenti
`reason`	Diperlukan saat `decision` adalah `"block"`. Memberitahu Claude mengapa itu harus melanjutkan

```
{
  "decision": "block",
  "reason": "Must be provided when Claude is blocked from stopping"
}
```

TeammateIdle

Dijalankan ketika rekan kerja agent team akan menjadi idle setelah menyelesaikan giliran. Gunakan ini untuk menegakkan quality gates sebelum rekan kerja berhenti bekerja, seperti memerlukan passing lint checks atau memverifikasi bahwa file output ada.

Ketika hook `TeammateIdle` keluar dengan kode 2, rekan kerja menerima pesan stderr sebagai umpan balik dan terus bekerja alih-alih menjadi idle. Untuk menghentikan rekan kerja sepenuhnya alih-alih menjalankannya kembali, kembalikan JSON dengan `{"continue": false, "stopReason": "..."}`. Hooks TeammateIdle tidak mendukung matchers dan dijalankan pada setiap kemunculan.

Input TeammateIdle

Selain bidang input umum, hooks TeammateIdle menerima `teammate_name` dan `team_name`.

```
{
  "session_id": "abc123",
  "transcript_path": "/Users/.../.claude/projects/.../
00893aaf-19fa-41d2-8238-13269b9b3ca0.jsonl",
  "cwd": "/Users/...",
  "permission_mode": "default",
  "hook_event_name": "TeammateIdle",
  "teammate_name": "researcher",
  "team_name": "my-project"
}
```

Bidang	Deskripsi
`teammate_name`	Nama rekan kerja yang akan menjadi idle
`team_name`	Nama tim

Kontrol keputusan TeammateIdle

Hooks TeammateIdle mendukung dua cara untuk mengontrol perilaku rekan kerja:

- **Exit code 2**: rekan kerja menerima pesan stderr sebagai umpan balik dan terus bekerja alih-alih menjadi idle.
- **JSON** `{"continue": false, "stopReason": "..."}`: menghentikan rekan kerja sepenuhnya, cocok dengan perilaku hook `Stop`. `stopReason` ditampilkan ke pengguna.

Contoh ini memeriksa bahwa artefak build ada sebelum mengizinkan rekan kerja menjadi idle:

```
#!/bin/bash

if [ ! -f "./dist/output.js" ]; then
  echo "Build artifact missing. Run the build before stopping." >&2
  exit 2
fi

exit 0
```

TaskCompleted

Dijalankan ketika tugas ditandai sebagai selesai. Ini dijalankan dalam dua situasi: ketika agen apa pun secara eksplisit menandai tugas sebagai selesai melalui alat TaskUpdate, atau ketika rekan kerja agent team menyelesaikan giliran dengan tugas yang sedang berlangsung. Gunakan ini untuk menegakkan kriteria penyelesaian seperti passing tests atau lint checks sebelum tugas dapat ditutup.

Ketika hook `TaskCompleted` keluar dengan kode 2, tugas tidak ditandai sebagai selesai dan pesan stderr diumpankan kembali ke model sebagai umpan balik. Untuk menghentikan rekan kerja sepenuhnya alih-alih menjalankannya kembali, kembalikan JSON dengan `{"continue": false, "stopReason": "..."}`. Hooks TaskCompleted tidak mendukung matchers dan dijalankan pada setiap kemunculan.

Input TaskCompleted

Selain bidang input umum, hooks TaskCompleted menerima `task_id`, `task_subject`, dan secara opsional `task_description`, `teammate_name`, dan `team_name`.

```
{
  "session_id": "abc123",
  "transcript_path": "/Users/.../.claude/projects/.../
00893aaf-19fa-41d2-8238-13269b9b3ca0.jsonl",
  "cwd": "/Users/...",
  "permission_mode": "default",
  "hook_event_name": "TaskCompleted",
  "task_id": "task-001",
  "task_subject": "Implement user authentication",
  "task_description": "Add login and signup endpoints",
  "teammate_name": "implementer",
  "team_name": "my-project"
}
```

Bidang	Deskripsi
`task_id`	Pengenal tugas yang sedang diselesaikan
`task_subject`	Judul tugas
`task_description`	Deskripsi terperinci tugas. Mungkin tidak ada
`teammate_name`	Nama rekan kerja yang menyelesaikan tugas. Mungkin tidak ada
`team_name`	Nama tim. Mungkin tidak ada

Kontrol keputusan TaskCompleted

Hooks TaskCompleted mendukung dua cara untuk mengontrol penyelesaian tugas:

- **Exit code 2**: tugas tidak ditandai sebagai selesai dan pesan stderr diumpankan kembali ke model sebagai umpan balik.
- **JSON** `{"continue": false, "stopReason": "..."}`: menghentikan rekan kerja sepenuhnya, cocok dengan perilaku hook `Stop`. `stopReason` ditampilkan ke pengguna.

Contoh ini menjalankan tests dan memblokir penyelesaian tugas jika gagal:

```
#!/bin/bash
INPUT=$(cat)
TASK_SUBJECT=$(echo "$INPUT" | jq -r '.task_subject')

## Jalankan test suite
if ! npm test 2>&1; then
  echo "Tests not passing. Fix failing tests before completing: $TASK_SUBJECT" >&2
  exit 2
fi

exit 0
```

ConfigChange

Dijalankan ketika file konfigurasi berubah selama sesi. Gunakan ini untuk mengaudit perubahan pengaturan, menegakkan kebijakan keamanan, atau memblokir modifikasi tidak sah ke file konfigurasi.

Hooks ConfigChange dijalankan untuk perubahan ke file pengaturan, pengaturan kebijakan terkelola, dan file skill. Bidang `source` dalam input memberitahu Anda tipe konfigurasi mana yang berubah, dan bidang `file_path` opsional menyediakan path ke file yang berubah.

Matcher memfilter pada sumber konfigurasi:

Matcher	Kapan dijalankan
`user_settings`	`~/.claude/settings.json` berubah
`project_settings`	`.claude/settings.json` berubah
`local_settings`	`.claude/settings.local.json` berubah
`policy_settings`	Pengaturan kebijakan terkelola berubah
`skills`	File skill dalam `.claude/skills/` berubah

Contoh ini mencatat semua perubahan konfigurasi untuk audit keamanan:

```
{
  "hooks": {
    "ConfigChange": [
      {
        "hooks": [
          {
            "type": "command",
            "command": "\"$CLAUDE_PROJECT_DIR\"/.claude/hooks/audit-config-
change.sh"
          }
        ]
      }
    ]
  }
}
```

Input ConfigChange

Selain bidang input umum, hooks ConfigChange menerima `source` dan secara opsional `file_path`. Bidang `source` menunjukkan tipe konfigurasi mana yang berubah, dan `file_path` menyediakan path ke file spesifik yang dimodifikasi.

```
{
  "session_id": "abc123",
  "transcript_path": "/Users/.../.claude/projects/.../
00893aaf-19fa-41d2-8238-13269b9b3ca0.jsonl",
  "cwd": "/Users/...",
  "permission_mode": "default",
  "hook_event_name": "ConfigChange",
  "source": "project_settings",
  "file_path": "/Users/.../my-project/.claude/settings.json"
}
```

Kontrol keputusan ConfigChange

Hooks ConfigChange dapat memblokir perubahan konfigurasi dari berlaku. Gunakan exit code 2 atau JSON `decision` untuk mencegah perubahan. Ketika diblokir, pengaturan baru tidak diterapkan ke sesi yang sedang berjalan.

Bidang	Deskripsi
`decision`	`"block"` mencegah perubahan konfigurasi diterapkan. Hilangkan untuk mengizinkan perubahan
`reason`	Penjelasan ditampilkan ke pengguna saat `decision` adalah `"block"`

```
{
  "decision": "block",
  "reason": "Configuration changes to project settings require admin approval"
}
```

Perubahan `policy_settings` tidak dapat diblokir. Hooks masih dijalankan untuk sumber `policy_settings`, jadi Anda dapat menggunakannya untuk audit logging, tetapi keputusan blocking apa pun diabaikan. Ini memastikan pengaturan yang dikelola enterprise selalu berlaku.

WorktreeCreate

Ketika Anda menjalankan `claude --worktree` atau subagent menggunakan `isolation: "worktree"`, Claude Code membuat salinan kerja terisolasi menggunakan `git worktree`. Jika Anda mengonfigurasi hook WorktreeCreate, itu menggantikan perilaku git default, memungkinkan Anda menggunakan sistem kontrol versi berbeda seperti SVN, Perforce, atau Mercurial.

Hook harus mencetak path absolut ke direktori worktree yang dibuat di stdout. Claude Code menggunakan path ini sebagai direktori kerja untuk sesi terisolasi.

Contoh ini membuat salinan kerja SVN dan mencetak path untuk Claude Code gunakan. Ganti URL repositori dengan milik Anda sendiri:

```
{
  "hooks": {
    "WorktreeCreate": [
      {
        "hooks": [
          {
            "type": "command",
            "command": "bash -c 'NAME=$(jq -r .name); DIR=\"$HOME/.claude/
worktrees/$NAME\"; svn checkout https://svn.example.com/repo/trunk
\"$DIR\" >&2 && echo \"$DIR\"'"
          }
        ]
      }
    ]
  }
}
```

Hook membaca `name` worktree dari input JSON di stdin, melakukan checkout salinan segar ke direktori baru, dan mencetak path direktori. `echo` pada baris terakhir adalah apa yang Claude Code baca sebagai path worktree. Alihkan output lainnya ke stderr sehingga tidak mengganggu path.

Input WorktreeCreate

Selain bidang input umum, hooks WorktreeCreate menerima bidang `name`. Ini adalah pengenal slug untuk worktree baru, baik ditentukan oleh pengguna atau auto-generated (misalnya, `bold-oak-a3f2`).

```
{
  "session_id": "abc123",
  "transcript_path": "/Users/.../.claude/projects/.../
00893aaf-19fa-41d2-8238-13269b9b3ca0.jsonl",
  "cwd": "/Users/...",
  "hook_event_name": "WorktreeCreate",
  "name": "feature-auth"
}
```

Output WorktreeCreate

Hook harus mencetak path absolut ke direktori worktree yang dibuat di stdout. Jika hook gagal atau tidak menghasilkan output, pembuatan worktree gagal dengan kesalahan.

Hooks WorktreeCreate tidak menggunakan model keputusan allow/block standar. Sebaliknya, kesuksesan atau kegagalan hook menentukan hasil. Hanya hooks `type: "command"` yang didukung.

WorktreeRemove

Pasangan cleanup untuk WorktreeCreate. Hook ini dijalankan ketika worktree sedang dihapus, baik ketika Anda keluar dari sesi `--worktree` dan memilih untuk menghapusnya, atau ketika subagent dengan `isolation: "worktree"` selesai. Untuk worktrees berbasis git, Claude menangani cleanup secara otomatis dengan `git worktree remove`. Jika Anda mengonfigurasi hook WorktreeCreate untuk sistem kontrol versi non-git, pasangkan dengan hook WorktreeRemove untuk menangani cleanup. Tanpanya, direktori worktree ditinggalkan di disk.

Claude Code meneruskan path yang dicetak WorktreeCreate di stdout sebagai `worktree_path` dalam input hook. Contoh ini membaca path itu dan menghapus direktori:

```
{
  "hooks": {
    "WorktreeRemove": [
      {
        "hooks": [
          {
            "type": "command",
            "command": "bash -c 'jq -r .worktree_path | xargs rm -rf'"
          }
        ]
      }
    ]
  }
}
```

Input WorktreeRemove

Selain bidang input umum, hooks WorktreeRemove menerima bidang `worktree_path`, yang merupakan path absolut ke worktree yang sedang dihapus.

```
{
  "session_id": "abc123",
  "transcript_path": "/Users/.../.claude/projects/.../
00893aaf-19fa-41d2-8238-13269b9b3ca0.jsonl",
  "cwd": "/Users/...",
  "hook_event_name": "WorktreeRemove",
  "worktree_path": "/Users/.../my-project/.claude/worktrees/feature-auth"
}
```

Hooks WorktreeRemove tidak memiliki kontrol keputusan. Mereka tidak dapat memblokir penghapusan worktree tetapi dapat melakukan tugas cleanup seperti menghapus status kontrol versi atau mengarsipkan perubahan. Kegagalan hook dicatat dalam mode debug saja. Hanya hooks `type: "command"` yang didukung.

PreCompact

Dijalankan sebelum Claude Code akan menjalankan operasi compact.

Nilai matcher menunjukkan apakah pemadatan dipicu secara manual atau otomatis:

Matcher	Kapan dijalankan
`manual`	`/compact`
`auto`	Auto-compact ketika context window penuh

Input PreCompact

Selain bidang input umum, hooks PreCompact menerima `trigger` dan `custom_instructions`. Untuk `manual`, `custom_instructions` berisi apa yang diteruskan pengguna ke `/compact`. Untuk `auto`, `custom_instructions` kosong.

```
{
  "session_id": "abc123",
  "transcript_path": "/Users/.../.claude/projects/.../
00893aaf-19fa-41d2-8238-13269b9b3ca0.jsonl",
  "cwd": "/Users/...",
  "permission_mode": "default",
  "hook_event_name": "PreCompact",
  "trigger": "manual",
  "custom_instructions": ""
}
```

SessionEnd

Dijalankan ketika sesi Claude Code berakhir. Berguna untuk tugas cleanup, logging statistik sesi, atau menyimpan status sesi. Mendukung matchers untuk memfilter berdasarkan alasan keluar.

Bidang `reason` dalam input hook menunjukkan mengapa sesi berakhir:

Alasan	Deskripsi
`clear`	Sesi dihapus dengan perintah `/clear`
`logout`	Pengguna logout
`prompt_input_exit`	Pengguna keluar saat input prompt terlihat
`bypass_permissions_disabled`	Mode bypass permissions dinonaktifkan
`other`	Alasan keluar lainnya

Input SessionEnd

Selain bidang input umum, hooks SessionEnd menerima bidang `reason` menunjukkan mengapa sesi berakhir. Lihat tabel reason di atas untuk semua nilai.

```
{
  "session_id": "abc123",
  "transcript_path": "/Users/.../.claude/projects/.../
00893aaf-19fa-41d2-8238-13269b9b3ca0.jsonl",
  "cwd": "/Users/...",
  "permission_mode": "default",
  "hook_event_name": "SessionEnd",
  "reason": "other"
}
```

Hooks SessionEnd tidak memiliki kontrol keputusan. Mereka tidak dapat memblokir penghentian sesi tetapi dapat melakukan tugas cleanup.

Prompt-based hooks

Selain command dan HTTP hooks, Claude Code mendukung prompt-based hooks (`type: "prompt"`) yang menggunakan LLM untuk mengevaluasi apakah akan mengizinkan atau memblokir tindakan, dan agent hooks (`type: "agent"`) yang spawn agentic verifier dengan akses alat. Tidak semua event mendukung setiap tipe hook.

Events yang mendukung semua empat tipe hook (`command` , `http` , `prompt` , dan `agent`):

- `PermissionRequest`
- `PostToolUse`
- `PostToolUseFailure`
- `PreToolUse`
- `Stop`
- `SubagentStop`
- `TaskCompleted`
- `UserPromptSubmit`

Events yang hanya mendukung hooks `type: "command"` :

- `ConfigChange`
- `InstructionsLoaded`
- `Notification`
- `PreCompact`
- `SessionEnd`

- `SessionStart`
- `SubagentStart`
- `TeammateIdle`
- `WorktreeCreate`
- `WorktreeRemove`

Bagaimana prompt-based hooks bekerja

Alih-alih menjalankan perintah Bash, prompt-based hooks:

1. Mengirimkan input hook dan prompt Anda ke model Claude, Haiku secara default
2. LLM merespons dengan JSON terstruktur yang berisi keputusan
3. Claude Code memproses keputusan secara otomatis

Konfigurasi prompt hook

Atur `type` ke `"prompt"` dan sediakan string `prompt` alih-alih `command`. Gunakan placeholder `$ARGUMENTS` untuk menyuntikkan data JSON input hook ke dalam teks prompt Anda. Claude Code mengirimkan prompt gabungan dan input ke model Claude cepat, yang mengembalikan keputusan JSON.

Hook `Stop` ini meminta LLM untuk mengevaluasi apakah Claude harus berhenti sebelum mengizinkan Claude selesai:

```
{
  "hooks": {
    "Stop": [
      {
        "hooks": [
          {
            "type": "prompt",
            "prompt": "Evaluate if Claude should stop: $ARGUMENTS. Check if all
tasks are complete."
          }
        ]
      }
    ]
  }
}
```

Bidang	Diperlukan	Deskripsi
`type`	ya	Harus `"prompt"`
`prompt`	ya	Teks prompt untuk dikirim ke LLM. Gunakan `$ARGUMENTS` sebagai placeholder untuk JSON input hook. Jika `$ARGUMENTS` tidak ada, JSON input ditambahkan ke prompt
`model`	tidak	Model untuk digunakan untuk evaluasi. Default ke model cepat
`timeout`	tidak	Timeout dalam detik. Default: 30

Skema respons

LLM harus merespons dengan JSON yang berisi:

```
{
  "ok": true | false,
  "reason": "Explanation for the decision"
}
```

Bidang	Deskripsi
`ok`	`true` mengizinkan tindakan, `false` mencegahnya
`reason`	Diperlukan saat `ok` adalah `false`. Penjelasan ditampilkan ke Claude

Contoh: Multi-criteria Stop hook

Hook `Stop` ini menggunakan prompt terperinci untuk memeriksa tiga kondisi sebelum mengizinkan Claude berhenti. Jika `"ok"` adalah `false`, Claude terus bekerja dengan alasan yang disediakan sebagai instruksi berikutnya. Hooks `SubagentStop` menggunakan format yang sama untuk mengevaluasi apakah subagent harus berhenti:

```
{
  "hooks": {
    "Stop": [
      {
        "hooks": [
          {
            "type": "prompt",
            "prompt": "You are evaluating whether Claude should stop working.
Context: $ARGUMENTS\n\nAnalyze the conversation and determine if:\n1. All user-
requested tasks are complete\n2. Any errors need to be addressed\n3. Follow-up
work is needed\n\nRespond with JSON: {\"ok\": true} to allow stopping, or
{\"ok\": false, \"reason\": \"your explanation\"} to continue working.",
            "timeout": 30
          }
        ]
      }
    ]
  }
}
```

Agent-based hooks

Agent-based hooks (`type: "agent"`) seperti prompt-based hooks tetapi dengan akses alat multi-turn. Alih-alih pemanggilan LLM tunggal, hook agent spawn subagent yang dapat membaca file, mencari kode, dan memeriksa codebase untuk memverifikasi kondisi. Agent hooks mendukung event yang sama seperti prompt-based hooks.

Bagaimana agent hooks bekerja

Ketika hook agent dijalankan:

1. Claude Code spawn subagent dengan prompt Anda dan JSON input hook
2. Subagent dapat menggunakan alat seperti Read, Grep, dan Glob untuk menyelidiki
3. Setelah hingga 50 turn, subagent mengembalikan keputusan terstruktur `{ "ok": true/false }`
4. Claude Code memproses keputusan dengan cara yang sama seperti prompt hook

Agent hooks berguna ketika verifikasi memerlukan memeriksa file aktual atau output test, bukan hanya mengevaluasi data input hook saja.

Konfigurasi agent hook

Atur `type` ke `"agent"` dan sediakan string `prompt`. Bidang konfigurasi sama seperti prompt hooks, dengan timeout default yang lebih lama:

Bidang	Diperlukan	Deskripsi
`type`	ya	Harus `"agent"`
`prompt`	ya	Prompt menjelaskan apa yang diverifikasi. Gunakan `$ARGUMENTS` sebagai placeholder untuk JSON input hook
`model`	tidak	Model untuk digunakan. Default ke model cepat
`timeout`	tidak	Timeout dalam detik. Default: 60

Skema respons sama seperti prompt hooks: `{ "ok": true }` untuk mengizinkan atau `{ "ok": false, "reason": "..." }` untuk memblokir.

Hook `Stop` ini memverifikasi bahwa semua unit tests lulus sebelum mengizinkan Claude selesai:

```
{
  "hooks": {
    "Stop": [
      {
        "hooks": [
          {
            "type": "agent",
            "prompt": "Verify that all unit tests pass. Run the test suite and
check the results. $ARGUMENTS",
            "timeout": 120
          }
        ]
      }
    ]
  }
}
```

Jalankan hooks di latar belakang

Secara default, hooks memblokir eksekusi Claude sampai selesai. Untuk tugas yang berjalan lama seperti deployments, test suites, atau panggilan API eksternal, atur `"async": true` untuk menjalankan hook di latar belakang sementara Claude terus bekerja. Async hooks tidak dapat memblokir atau mengontrol perilaku Claude: bidang respons seperti `decision`, `permissionDecision`, dan `continue` tidak berpengaruh, karena tindakan yang akan mereka kontrol sudah selesai.

Konfigurasi async hook

Tambahkan `"async": true` ke konfigurasi command hook untuk menjalankannya di latar belakang tanpa memblokir Claude. Bidang ini hanya tersedia pada hooks `type: "command"`.

Hook ini menjalankan skrip test setelah setiap pemanggilan alat `Write`. Claude terus bekerja segera sementara `run-tests.sh` dijalankan hingga 120 detik. Ketika skrip selesai, outputnya disampaikan pada turn percakapan berikutnya:

```
{
  "hooks": {
    "PostToolUse": [
      {
        "matcher": "Write",
        "hooks": [
          {
            "type": "command",
            "command": "/path/to/run-tests.sh",
            "async": true,
            "timeout": 120
          }
        ]
      }
    ]
  }
}
```

Bidang `timeout` menetapkan waktu maksimum dalam detik untuk proses latar belakang. Jika tidak ditentukan, async hooks menggunakan default 10 menit yang sama seperti sync hooks.

Bagaimana async hooks dijalankan

Ketika hook async dijalankan, Claude Code memulai proses hook dan segera melanjutkan tanpa menunggu selesai. Hook menerima JSON input yang sama melalui stdin seperti hook sinkron.

Setelah proses latar belakang keluar, jika hook menghasilkan respons JSON dengan bidang `systemMessage` atau `additionalContext`, konten itu disampaikan ke Claude sebagai konteks pada turn percakapan berikutnya.

Contoh: jalankan tests setelah perubahan file

Hook ini memulai test suite di latar belakang setiap kali Claude menulis file, kemudian melaporkan hasil kembali ke Claude ketika tests selesai. Simpan skrip ini ke `.claude/hooks/run-tests-async.sh` dalam proyek Anda dan buat dapat dijalankan dengan `chmod +x`:

```
#!/bin/bash
## run-tests-async.sh

## Baca hook input dari stdin
INPUT=$(cat)
FILE_PATH=$(echo "$INPUT" | jq -r '.tool_input.file_path // empty')

## Hanya jalankan tests untuk file sumber
if [[ "$FILE_PATH" =~ *.ts && "$FILE_PATH" =~ *.js ]]; then
  exit 0
fi

## Jalankan tests dan laporkan hasil melalui systemMessage
RESULT=$(npm test 2>&1)
EXIT_CODE=$?

if [ $EXIT_CODE -eq 0 ]; then
  echo "{\"systemMessage\": \"Tests passed after editing $FILE_PATH\"}"
else
  echo "{\"systemMessage\": \"Tests failed after editing $FILE_PATH: $RESULT\"}"
fi
```

Kemudian tambahkan konfigurasi ini ke `.claude/settings.json` dalam akar proyek Anda. Flag `async: true` memungkinkan Claude terus bekerja sementara tests dijalankan:

```
{
  "hooks": {
    "PostToolUse": [
      {
        "matcher": "Write|Edit",
        "hooks": [
          {
            "type": "command",
            "command": "\"$CLAUDE_PROJECT_DIR\"/.claude/hooks/run-tests-async.sh",
            "async": true,
            "timeout": 300
          }
        ]
      }
    ]
  }
}
```

Keterbatasan

Async hooks memiliki beberapa batasan dibandingkan dengan hooks sinkron:

- Hanya hooks `type: "command"` yang mendukung `async`. Prompt-based hooks tidak dapat dijalankan secara asinkron.
- Async hooks tidak dapat memblokir pemanggilan alat atau mengembalikan keputusan. Pada saat hook selesai, tindakan pemicu sudah dilanjutkan.
- Output hook disampaikan pada turn percakapan berikutnya. Jika sesi idle, respons menunggu sampai interaksi pengguna berikutnya.
- Setiap eksekusi membuat proses latar belakang terpisah. Tidak ada deduplikasi di seluruh beberapa penjalankan hook async yang sama.

Pertimbangan keamanan

Penafian

Command hooks dijalankan dengan izin pengguna sistem penuh Anda.

> **Warning:**

> Command hooks menjalankan perintah shell dengan izin pengguna penuh Anda. Mereka dapat memodifikasi, menghapus, atau mengakses file apa pun yang dapat diakses akun pengguna Anda. Tinjau dan uji semua perintah hook sebelum menambahkannya ke konfigurasi Anda.

Praktik terbaik keamanan

Ingat praktik-praktik ini saat menulis hooks:

- **Validasi dan sanitasi input**: jangan pernah mempercayai data input secara membabi buta
- **Selalu kutip variabel shell**: gunakan `"$VAR"` bukan `$VAR`
- **Blokir path traversal**: periksa `..` dalam path file
- **Gunakan path absolut**: tentukan path lengkap untuk skrip, menggunakan `"$CLAUDE_PROJECT_DIR"` untuk akar proyek
- **Lewati file sensitif**: hindari `.env`, `.git/`, keys, dll.

Debug hooks

Jalankan `claude --debug` untuk melihat detail eksekusi hook, termasuk hooks mana yang cocok, kode keluar mereka, dan output. Toggle mode verbose dengan `Ctrl+O` untuk melihat progress hook dalam transkrip.

```
[DEBUG] Executing hooks for PostToolUse:Write
[DEBUG] Getting matching hook commands for PostToolUse with query: Write
[DEBUG] Found 1 hook matchers in settings
[DEBUG] Matched 1 hooks for query "Write"
[DEBUG] Found 1 hook commands to execute
[DEBUG] Executing hook command: <Your command> with timeout 600000ms
[DEBUG] Hook command completed with status 0: <Your stdout>
```

Untuk troubleshooting masalah umum seperti hooks tidak dijalankan, infinite Stop hook loops, atau kesalahan konfigurasi, lihat Limitations and troubleshooting dalam panduan.

Otomatisasi alur kerja dengan hooks

> *Jalankan perintah shell secara otomatis ketika Claude Code mengedit file, menyelesaikan tugas, atau memerlukan input. Format kode, kirim notifikasi, validasi perintah, dan terapkan aturan proyek.*

Hooks adalah perintah shell yang ditentukan pengguna yang dijalankan pada titik-titik spesifik dalam siklus hidup Claude Code. Mereka memberikan kontrol deterministik atas perilaku Claude Code, memastikan tindakan tertentu selalu terjadi daripada mengandalkan LLM untuk memilih menjalankannya. Gunakan hooks untuk menegakkan aturan proyek, mengotomatisasi tugas berulang, dan mengintegrasikan Claude Code dengan alat yang sudah ada.

Untuk keputusan yang memerlukan penilaian daripada aturan deterministik, Anda juga dapat menggunakan hooks berbasis prompt atau hooks berbasis agent yang menggunakan model Claude untuk mengevaluasi kondisi.

Untuk cara lain memperluas Claude Code, lihat skills untuk memberikan Claude instruksi tambahan dan perintah yang dapat dieksekusi, subagents untuk menjalankan tugas dalam konteks terisolasi, dan plugins untuk mengemas ekstensi untuk dibagikan di seluruh proyek.

> **Tip:**
>
> Panduan ini mencakup kasus penggunaan umum dan cara memulai. Untuk skema acara lengkap, format input/output JSON, dan fitur lanjutan seperti hooks asinkron dan hooks alat MCP, lihat referensi Hooks.

Siapkan hook pertama Anda

Cara tercepat untuk membuat hook adalah melalui menu interaktif `/hooks` di Claude Code. Panduan ini membuat hook notifikasi desktop, sehingga Anda mendapat peringatan kapan pun Claude menunggu input Anda daripada menonton terminal.

Step 1: Buka menu hooks

Ketik `/hooks` di CLI Claude Code. Anda akan melihat daftar semua acara hook yang tersedia, ditambah opsi untuk menonaktifkan semua hooks. Setiap acara sesuai dengan titik dalam siklus hidup Claude di mana Anda dapat menjalankan kode khusus. Pilih `Notification` untuk membuat hook yang aktif ketika Claude memerlukan perhatian Anda.

Step 2: Konfigurasi matcher

Menu menampilkan daftar matcher, yang memfilter kapan hook aktif. Atur matcher ke `*` untuk aktif pada semua jenis notifikasi. Anda dapat mempersempit nanti dengan mengubah matcher ke nilai spesifik seperti `permission_prompt` atau `idle_prompt`.

Step 3: Tambahkan perintah Anda

Pilih `+ Add new hook…`. Menu meminta Anda untuk perintah shell yang akan dijalankan ketika acara aktif. Hooks menjalankan perintah shell apa pun yang Anda berikan, jadi Anda dapat menggunakan alat notifikasi bawaan platform Anda. Salin perintah untuk OS Anda:

macOS

Menggunakan `osascript` untuk memicu notifikasi macOS asli melalui AppleScript:

```
osascript -e 'display notification "Claude Code needs your attention" with title "Claude Code"'
```

Linux

Menggunakan `notify-send`, yang sudah diinstal sebelumnya di sebagian besar desktop Linux dengan daemon notifikasi:

```
notify-send 'Claude Code' 'Claude Code needs your attention'
```

Windows (PowerShell)

Menggunakan PowerShell untuk menampilkan kotak pesan asli melalui Windows Forms .NET:

```
powershell.exe -Command "[System.Reflection.Assembly]::LoadWithPartialName('System.Windows.Forms'); [System.Windows.Forms.MessageBox]::Show('Claude Code needs your attention', 'Claude Code')"
```

Step 4: Pilih lokasi penyimpanan

Menu menanyakan di mana menyimpan konfigurasi hook. Pilih `User settings` untuk menyimpannya di `~/.claude/settings.json`, yang menerapkan hook ke semua proyek Anda. Anda juga dapat memilih `Project settings` untuk membatasi ke proyek saat ini. Lihat Konfigurasi lokasi hook untuk semua cakupan yang tersedia.

Step 5: Uji hook

Tekan `Esc` untuk kembali ke CLI. Minta Claude untuk melakukan sesuatu yang memerlukan izin, kemudian beralih dari terminal. Anda harus menerima notifikasi desktop.

Apa yang dapat Anda otomatisasi

Hooks memungkinkan Anda menjalankan kode pada titik-titik kunci dalam siklus hidup Claude Code: format file setelah edit, blokir perintah sebelum dijalankan, kirim notifikasi ketika Claude memerlukan input, injeksi konteks saat awal sesi, dan banyak lagi. Untuk daftar lengkap acara hook, lihat referensi Hooks.

Setiap contoh mencakup blok konfigurasi siap pakai yang Anda tambahkan ke file pengaturan. Pola paling umum:

- Dapatkan notifikasi ketika Claude memerlukan input
- Format kode otomatis setelah edit
- Blokir edit ke file yang dilindungi
- Injeksi ulang konteks setelah pemadatan
- Audit perubahan konfigurasi

Dapatkan notifikasi ketika Claude memerlukan input

Dapatkan notifikasi desktop kapan pun Claude selesai bekerja dan memerlukan input Anda, sehingga Anda dapat beralih ke tugas lain tanpa memeriksa terminal.

Hook ini menggunakan acara `Notification`, yang aktif ketika Claude menunggu input atau izin. Setiap tab di bawah menggunakan perintah notifikasi asli platform. Tambahkan ini ke `~/.claude/settings.json`, atau gunakan panduan interaktif di atas untuk mengonfigurasinya dengan `/hooks`:

macOS

```
{
  "hooks": {
    "Notification": [
      {
        "matcher": "",
        "hooks": [
          {
            "type": "command",
            "command": "osascript -e 'display notification \"Claude Code needs
your attention\" with title \"Claude Code\"'"
          }
        ]
      }
    ]
  }
}
```

Linux

```
{
  "hooks": {
    "Notification": [
      {
        "matcher": "",
        "hooks": [
          {
            "type": "command",
            "command": "notify-send 'Claude Code' 'Claude Code needs your
attention'"
          }
        ]
      }
    ]
  }
}
```

Windows (PowerShell)

```
{
  "hooks": {
    "Notification": [
      {
        "matcher": "",
        "hooks": [
          {
            "type": "command",
            "command": "powershell.exe -Command \"[System.Reflection.Assembly]::Lo
adWithPartialName('System.Windows.Forms');
[System.Windows.Forms.MessageBox]::Show('Claude Code needs your attention',
'Claude Code')\""
          }
        ]
      }
    ]
  }
}
```

Format kode otomatis setelah edit

Jalankan Prettier secara otomatis pada setiap file yang Claude edit, sehingga pemformatan tetap konsisten tanpa intervensi manual.

Hook ini menggunakan acara `PostToolUse` dengan matcher `Edit|Write`, sehingga hanya berjalan setelah alat pengeditan file. Perintah mengekstrak jalur file yang diedit dengan `jq` dan meneruskannya ke Prettier. Tambahkan ini ke `.claude/settings.json` di akar proyek Anda:

```
{
  "hooks": {
    "PostToolUse": [
      {
        "matcher": "Edit|Write",
        "hooks": [
          {
            "type": "command",
            "command": "jq -r '.tool_input.file_path' | xargs npx prettier --
write"
          }
        ]
      }
    ]
  }
}
```

> **Note:**
>
> Contoh Bash di halaman ini menggunakan `jq` untuk penguraian JSON. Instal dengan `brew install jq` (macOS), `apt-get install jq` (Debian/Ubuntu), atau lihat unduhan `jq`.

Blokir edit ke file yang dilindungi

Cegah Claude dari memodifikasi file sensitif seperti `.env`, `package-lock.json`, atau apa pun di `.git/`. Claude menerima umpan balik yang menjelaskan mengapa edit diblokir, sehingga dapat menyesuaikan pendekatannya.

Contoh ini menggunakan file skrip terpisah yang dipanggil oleh hook. Skrip memeriksa jalur file target terhadap daftar pola yang dilindungi dan keluar dengan kode 2 untuk memblokir edit.

Step 1: Buat skrip hook

Simpan ini ke `.claude/hooks/protect-files.sh`:

```
#!/bin/bash
## protect-files.sh

INPUT=$(cat)
FILE_PATH=$(echo "$INPUT" | jq -r '.tool_input.file_path // empty')

PROTECTED_PATTERNS=(".env" "package-lock.json" ".git/")

for pattern in "${PROTECTED_PATTERNS[@]}"; do
  if [[ "$FILE_PATH" == *"$pattern"* ]]; then
    echo "Blocked: $FILE_PATH matches protected pattern '$pattern'" >&2
    exit 2
  fi
done

exit 0
```

Step 2: Buat skrip dapat dieksekusi (macOS/Linux)

Skrip hook harus dapat dieksekusi agar Claude Code dapat menjalankannya:

```
chmod +x .claude/hooks/protect-files.sh
```

Step 3: Daftarkan hook

Tambahkan hook `PreToolUse` ke `.claude/settings.json` yang menjalankan skrip sebelum panggilan alat `Edit` atau `Write` apa pun:

```
{
  "hooks": {
    "PreToolUse": [
      {
        "matcher": "Edit|Write",
        "hooks": [
          {
            "type": "command",
            "command": "\"$CLAUDE_PROJECT_DIR\"/.claude/hooks/protect-files.sh"
          }
        ]
      }
    ]
  }
}
```

Injeksi ulang konteks setelah pemadatan

Ketika jendela konteks Claude penuh, pemadatan merangkum percakapan untuk membebaskan ruang. Ini dapat kehilangan detail penting. Gunakan hook `SessionStart` dengan matcher `compact` untuk injeksi ulang konteks kritis setelah setiap pemadatan.

Teks apa pun yang ditulis perintah Anda ke stdout ditambahkan ke konteks Claude. Contoh ini mengingatkan Claude tentang konvensi proyek dan pekerjaan terbaru. Tambahkan ini ke `.claude/settings.json` di akar proyek Anda:

```
{
  "hooks": {
    "SessionStart": [
      {
        "matcher": "compact",
        "hooks": [
          {
            "type": "command",
            "command": "echo 'Reminder: use Bun, not npm. Run bun test before
committing. Current sprint: auth refactor.'"
          }
        ]
      }
    ]
  }
}
```

Anda dapat mengganti `echo` dengan perintah apa pun yang menghasilkan output dinamis, seperti `git log --oneline -5` untuk menampilkan commit terbaru. Untuk injeksi konteks pada setiap awal sesi, pertimbangkan menggunakan CLAUDE.md sebagai gantinya. Untuk variabel lingkungan, lihat `CLAUDE_ENV_FILE` dalam referensi.

Audit perubahan konfigurasi

Lacak ketika file pengaturan atau skills berubah selama sesi. Acara `ConfigChange` aktif ketika proses eksternal atau editor memodifikasi file konfigurasi, sehingga Anda dapat mencatat perubahan untuk kepatuhan atau memblokir modifikasi yang tidak sah.

Contoh ini menambahkan setiap perubahan ke log audit. Tambahkan ini ke `~/.claude/settings.json`:

```
{
  "hooks": {
    "ConfigChange": [
      {
        "matcher": "",
        "hooks": [
          {
            "type": "command",
            "command": "jq -c '{timestamp: now | todate, source: .source,
file: .file_path}' >> ~/claude-config-audit.log"
          }
        ]
      }
    ]
  }
}
```

Matcher memfilter berdasarkan jenis konfigurasi: `user_settings`, `project_settings`, `local_settings`, `policy_settings`, atau `skills`. Untuk memblokir perubahan agar tidak berlaku, keluar dengan kode 2 atau kembalikan `{"decision": "block"}`. Lihat referensi ConfigChange untuk skema input lengkap.

Cara kerja hooks

Acara hook aktif pada titik-titik siklus hidup spesifik di Claude Code. Ketika acara aktif, semua hook yang cocok berjalan secara paralel, dan perintah hook yang identik secara otomatis dideduplikasi. Tabel di bawah menunjukkan setiap acara dan kapan dipicu:

Event	When it fires
`SessionStart`	When a session begins or resumes
`UserPromptSubmit`	When you submit a prompt, before Claude processes it
`PreToolUse`	Before a tool call executes. Can block it
`PermissionRequest`	When a permission dialog appears
`PostToolUse`	After a tool call succeeds
`PostToolUseFailure`	After a tool call fails

Event	When it fires
`Notification`	When Claude Code sends a notification
`SubagentStart`	When a subagent is spawned
`SubagentStop`	When a subagent finishes
`Stop`	When Claude finishes responding
`TeammateIdle`	When an agent team teammate is about to go idle
`TaskCompleted`	When a task is being marked as completed
`InstructionsLoaded`	When a CLAUDE.md or `.claude/rules/*.md` file is loaded into context. Fires at session start and when files are lazily loaded during a session
`ConfigChange`	When a configuration file changes during a session
`WorktreeCreate`	When a worktree is being created via `--worktree` or `isolation: "worktree"`. Replaces default git behavior
`WorktreeRemove`	When a worktree is being removed, either at session exit or when a subagent finishes
`PreCompact`	Before context compaction
`PostCompact`	After context compaction completes
`Elicitation`	When an MCP server requests user input during a tool call
`ElicitationResult`	After a user responds to an MCP elicitation, before the response is sent back to the server
`SessionEnd`	When a session terminates

Setiap hook memiliki `type` yang menentukan cara menjalankannya. Sebagian besar hooks menggunakan `"type": "command"`, yang menjalankan perintah shell. Tiga jenis lainnya tersedia:

- `"type": "http"`: POST data acara ke URL. Lihat HTTP hooks.
- `"type": "prompt"`: evaluasi LLM satu putaran. Lihat Hooks berbasis prompt.
- `"type": "agent"`: verifikasi multi-putaran dengan akses alat. Lihat Hooks berbasis agent.

Baca input dan kembalikan output

Hooks berkomunikasi dengan Claude Code melalui stdin, stdout, stderr, dan kode keluar. Ketika acara aktif, Claude Code meneruskan data khusus acara sebagai JSON ke stdin skrip Anda. Skrip Anda membaca data itu, melakukan pekerjaan, dan memberi tahu Claude Code apa yang harus dilakukan selanjutnya melalui kode keluar.

Input hook

Setiap acara mencakup bidang umum seperti `session_id` dan `cwd`, tetapi setiap jenis acara menambahkan data berbeda. Misalnya, ketika Claude menjalankan perintah Bash, hook `PreToolUse` menerima sesuatu seperti ini di stdin:

```
{
  "session_id": "abc123",           // unique ID for this session
  "cwd": "/Users/sarah/myproject", // working directory when the event fired
  "hook_event_name": "PreToolUse", // which event triggered this hook
  "tool_name": "Bash",              // the tool Claude is about to use
  "tool_input": {                   // the arguments Claude passed to the tool
    "command": "npm test"           // for Bash, this is the shell command
  }
}
```

Skrip Anda dapat menguraikan JSON itu dan bertindak atas bidang apa pun. Hook `UserPromptSubmit` mendapatkan teks `prompt` sebagai gantinya, hook `SessionStart` mendapatkan `source` (startup, resume, clear, compact), dan seterusnya. Lihat Bidang input umum dalam referensi untuk bidang bersama, dan bagian setiap acara untuk skema khusus acara.

Output hook

Skrip Anda memberi tahu Claude Code apa yang harus dilakukan selanjutnya dengan menulis ke stdout atau stderr dan keluar dengan kode spesifik. Misalnya, hook `PreToolUse` yang ingin memblokir perintah:

```
#!/bin/bash
INPUT=$(cat)
COMMAND=$(echo "$INPUT" | jq -r '.tool_input.command')

if echo "$COMMAND" | grep -q "drop table"; then
  echo "Blocked: dropping tables is not allowed" >&2  # stderr becomes Claude's
feedback
  exit 2 # exit 2 = block the action
fi

exit 0  # exit 0 = let it proceed
```

Kode keluar menentukan apa yang terjadi selanjutnya:

- **Exit 0**: tindakan berlanjut. Untuk hook `UserPromptSubmit` dan `SessionStart`, apa pun yang Anda tulis ke stdout ditambahkan ke konteks Claude.
- **Exit 2**: tindakan diblokir. Tulis alasan ke stderr, dan Claude menerimanya sebagai umpan balik sehingga dapat menyesuaikan.
- **Kode keluar lainnya**: tindakan berlanjut. Stderr dicatat tetapi tidak ditampilkan ke Claude. Alihkan mode verbose dengan `Ctrl+O` untuk melihat pesan ini dalam transkrip.

Output JSON terstruktur

Kode keluar memberi Anda dua opsi: izinkan atau blokir. Untuk kontrol lebih, keluar 0 dan cetak objek JSON ke stdout sebagai gantinya.

> **Note:**
>
> Gunakan exit 2 untuk memblokir dengan pesan stderr, atau exit 0 dengan JSON untuk kontrol terstruktur. Jangan campur: Claude Code mengabaikan JSON ketika Anda exit 2.

Misalnya, hook `PreToolUse` dapat menolak panggilan alat dan memberi tahu Claude mengapa, atau meningkatkannya ke pengguna untuk persetujuan:

```
{
  "hookSpecificOutput": {
    "hookEventName": "PreToolUse",
    "permissionDecision": "deny",
    "permissionDecisionReason": "Use rg instead of grep for better performance"
  }
}
```

Claude Code membaca `permissionDecision` dan membatalkan panggilan alat, kemudian memberi umpan balik `permissionDecisionReason` kembali ke Claude. Tiga opsi ini khusus untuk `PreToolUse`:

- `"allow"`: lanjutkan tanpa menampilkan prompt izin
- `"deny"`: batalkan panggilan alat dan kirim alasan ke Claude
- `"ask"`: tampilkan prompt izin kepada pengguna seperti biasa

Acara lain menggunakan pola keputusan berbeda. Misalnya, hook `PostToolUse` dan `Stop` menggunakan bidang `decision: "block"` tingkat atas, sementara `PermissionRequest` menggunakan `hookSpecificOutput.decision.behavior`. Lihat tabel ringkasan dalam referensi untuk rincian lengkap menurut acara.

Untuk hook `UserPromptSubmit`, gunakan `additionalContext` sebagai gantinya untuk injeksi teks ke dalam konteks Claude. Hooks berbasis prompt (`type: "prompt"`) menangani output secara berbeda: lihat Hooks berbasis prompt.

Filter hooks dengan matchers

Tanpa matcher, hook aktif pada setiap kemunculan acaranya. Matchers memungkinkan Anda mempersempit itu. Misalnya, jika Anda ingin menjalankan formatter hanya setelah edit file (bukan setelah setiap panggilan alat), tambahkan matcher ke hook `PostToolUse` Anda:

```
{
  "hooks": {
    "PostToolUse": [
      {
        "matcher": "Edit|Write",
        "hooks": [
          { "type": "command", "command": "prettier --write ..." }
        ]
      }
    ]
  }
}
```

Matcher `"Edit|Write"` adalah pola regex yang cocok dengan nama alat. Hook hanya aktif ketika Claude menggunakan alat `Edit` atau `Write`, bukan ketika menggunakan `Bash`, `Read`, atau alat lainnya.

Setiap jenis acara cocok pada bidang spesifik. Matchers mendukung string tepat dan pola regex:

Acara	Apa yang difilter matcher	Contoh nilai matcher
`PreToolUse`, `PostToolUse`, `PostToolUseFailure`, `PermissionRequest`	nama alat	`Bash`, `Edit|Write`, `mcp__.*`
`SessionStart`	cara sesi dimulai	`startup`, `resume`, `clear`, `compact`
`SessionEnd`	mengapa sesi berakhir	`clear`, `logout`, `prompt_input_exit`, `bypass_permissions_disabled`, `other`
`Notification`	jenis notifikasi	`permission_prompt`, `idle_prompt`, `auth_success`, `elicitation_dialog`
`SubagentStart`	jenis agent	`Bash`, `Explore`, `Plan`, atau nama agent khusus

Acara	Apa yang difilter matcher	Contoh nilai matcher
`PreCompact`	apa yang memicu pemadatan	`manual`, `auto`
`SubagentStop`	jenis agent	nilai yang sama seperti `SubagentStart`
`ConfigChange`	sumber konfigurasi	`user_settings`, `project_settings`, `local_settings`, `policy_settings`, `skills`
`UserPromptSubmit`, `Stop`, `TeammateIdle`, `TaskCompleted`, `WorktreeCreate`, `WorktreeRemove`	tidak ada dukungan matcher	selalu aktif pada setiap kemunculan

Beberapa contoh lagi menunjukkan matchers pada jenis acara berbeda:

Catat setiap perintah Bash

Cocokkan hanya panggilan alat `Bash` dan catat setiap perintah ke file. Acara `PostToolUse` aktif setelah perintah selesai, jadi `tool_input.command` berisi apa yang berjalan. Hook menerima data acara sebagai JSON di stdin, dan `jq -r '.tool_input.command'` mengekstrak hanya string perintah, yang `>>` tambahkan ke file log:

```
{
  "hooks": {
    "PostToolUse": [
      {
        "matcher": "Bash",
        "hooks": [
          {
            "type": "command",
            "command": "jq -r '.tool_input.command' >> ~/.claude/command-log.txt"
          }
        ]
      }
    ]
  }
}
```

Cocokkan alat MCP

Alat MCP menggunakan konvensi penamaan berbeda dari alat bawaan: `mcp__<server>__<tool>`, di mana `<server>` adalah nama server MCP dan `<tool>` adalah alat yang disediakannya. Misalnya, `mcp__github__search_repositories` atau `mcp__filesystem__read_file`. Gunakan matcher regex untuk menargetkan semua alat dari server spesifik, atau cocokkan di seluruh server dengan pola seperti `mcp__.*__write.*`. Lihat Cocokkan alat MCP dalam referensi untuk daftar lengkap contoh.

Perintah di bawah mengekstrak nama alat dari input JSON hook dengan `jq` dan menulisnya ke stderr, di mana muncul dalam mode verbose (`Ctrl+O`):

```
{
  "hooks": {
    "PreToolUse": [
      {
        "matcher": "mcp__github__.*",
        "hooks": [
          {
            "type": "command",
            "command": "echo \"GitHub tool called: $(jq -r '.tool_name')\" >&2"
          }
        ]
      }
    ]
  }
}
```

Bersihkan saat akhir sesi

Acara `SessionEnd` mendukung matchers pada alasan sesi berakhir. Hook ini hanya aktif pada `clear` (ketika Anda menjalankan `/clear`), bukan pada keluar normal:

```
{
  "hooks": {
    "SessionEnd": [
      {
        "matcher": "clear",
        "hooks": [
          {
            "type": "command",
            "command": "rm -f /tmp/claude-scratch-*.txt"
          }
        ]
      }
    ]
  }
}
```

Untuk sintaks matcher lengkap, lihat referensi Hooks.

Konfigurasi lokasi hook

Di mana Anda menambahkan hook menentukan cakupannya:

Lokasi	Cakupan	Dapat Dibagikan
`~/.claude/settings.json`	Semua proyek Anda	Tidak, lokal ke mesin Anda
`.claude/settings.json`	Proyek tunggal	Ya, dapat dikomit ke repo
`.claude/settings.local.json`	Proyek tunggal	Tidak, gitignored
Pengaturan kebijakan terkelola	Seluruh organisasi	Ya, dikendalikan admin
Plugin `hooks/hooks.json`	Ketika plugin diaktifkan	Ya, dikemas dengan plugin
Skill atau agent frontmatter	Saat skill atau agent aktif	Ya, didefinisikan dalam file komponen

Anda juga dapat menggunakan menu `/hooks` di Claude Code untuk menambah, menghapus, dan melihat hooks secara interaktif. Untuk menonaktifkan semua hooks sekaligus, gunakan toggle di bagian bawah menu `/hooks` atau atur `"disableAllHooks": true` dalam file pengaturan Anda.

Hooks yang ditambahkan melalui menu `/hooks` berlaku segera. Jika Anda mengedit file pengaturan secara langsung saat Claude Code berjalan, perubahan tidak akan berlaku sampai Anda meninjau dalam menu `/hooks` atau memulai ulang sesi Anda.

Hooks berbasis prompt

Untuk keputusan yang memerlukan penilaian daripada aturan deterministik, gunakan hooks `type: "prompt"`. Daripada menjalankan perintah shell, Claude Code mengirim prompt Anda dan data input hook ke model Claude (Haiku secara default) untuk membuat keputusan. Anda dapat menentukan model berbeda dengan bidang `model` jika Anda memerlukan kemampuan lebih.

Satu-satunya pekerjaan model adalah mengembalikan keputusan ya/tidak sebagai JSON:

- `"ok": true`: tindakan berlanjut

- `"ok": false`: tindakan diblokir. `"reason"` model diberi umpan balik ke Claude sehingga dapat menyesuaikan.

Contoh ini menggunakan hook `Stop` untuk menanyakan kepada model apakah semua tugas yang diminta selesai. Jika model mengembalikan `"ok": false`, Claude terus bekerja dan menggunakan `reason` sebagai instruksi berikutnya:

```
{
  "hooks": {
    "Stop": [
      {
        "hooks": [
          {
            "type": "prompt",
            "prompt": "Check if all tasks are complete. If not, respond with
{\"ok\": false, \"reason\": \"what remains to be done\"}."
          }
        ]
      }
    ]
  }
}
```

Untuk opsi konfigurasi lengkap, lihat Hooks berbasis prompt dalam referensi.

Hooks berbasis agent

Ketika verifikasi memerlukan inspeksi file atau menjalankan perintah, gunakan hooks `type: "agent"`. Tidak seperti hooks prompt yang membuat panggilan LLM tunggal, hooks agent menelurkan subagent yang dapat membaca file, mencari kode, dan menggunakan alat lain untuk memverifikasi kondisi sebelum mengembalikan keputusan.

Hooks agent menggunakan format respons `"ok"` / `"reason"` yang sama seperti hooks prompt, tetapi dengan timeout default lebih lama 60 detik dan hingga 50 putaran penggunaan alat.

Contoh ini memverifikasi bahwa tes lulus sebelum memungkinkan Claude berhenti:

```
{
  "hooks": {
    "Stop": [
      {
        "hooks": [
          {
            "type": "agent",
            "prompt": "Verify that all unit tests pass. Run the test suite and
check the results. $ARGUMENTS",
            "timeout": 120
          }
        ]
      }
    ]
  }
}
```

Gunakan hooks prompt ketika data input hook saja cukup untuk membuat keputusan. Gunakan hooks agent ketika Anda perlu memverifikasi sesuatu terhadap keadaan aktual codebase.

Untuk opsi konfigurasi lengkap, lihat Hooks berbasis agent dalam referensi.

HTTP hooks

Gunakan hooks `type: "http"` untuk POST data acara ke endpoint HTTP daripada menjalankan perintah shell. Endpoint menerima JSON yang sama yang diterima hook perintah di stdin, dan mengembalikan hasil melalui badan respons HTTP menggunakan format JSON yang sama.

HTTP hooks berguna ketika Anda ingin server web, fungsi cloud, atau layanan eksternal menangani logika hook: misalnya, layanan audit bersama yang mencatat acara penggunaan alat di seluruh tim.

Contoh ini memposting setiap penggunaan alat ke layanan logging lokal:

```
{
  "hooks": {
    "PostToolUse": [
      {
        "hooks": [
          {
            "type": "http",
            "url": "http://localhost:8080/hooks/tool-use",
            "headers": {
              "Authorization": "Bearer $MY_TOKEN"
            },
            "allowedEnvVars": ["MY_TOKEN"]
          }
        ]
      }
    ]
  }
}
```

Endpoint harus mengembalikan badan respons JSON menggunakan format output yang sama seperti hooks perintah. Untuk memblokir panggilan alat, kembalikan respons 2xx dengan bidang `hookSpecificOutput` yang sesuai. Kode status HTTP saja tidak dapat memblokir tindakan.

Nilai header mendukung interpolasi variabel lingkungan menggunakan sintaks `$VAR_NAME` atau `${VAR_NAME}`. Hanya variabel yang tercantum dalam array `allowedEnvVars` yang diselesaikan; semua referensi `$VAR` lainnya tetap kosong.

> **Note:**
>
> HTTP hooks harus dikonfigurasi dengan mengedit JSON pengaturan Anda secara langsung. Menu interaktif `/hooks` hanya mendukung penambahan hooks perintah.

Untuk opsi konfigurasi lengkap dan penanganan respons, lihat HTTP hooks dalam referensi.

Batasan dan pemecahan masalah

Batasan

- Hooks perintah berkomunikasi melalui stdout, stderr, dan kode keluar saja. Mereka tidak dapat memicu perintah atau panggilan alat secara langsung. HTTP hooks berkomunikasi melalui badan respons sebagai gantinya.
- Timeout hook adalah 10 menit secara default, dapat dikonfigurasi per hook dengan bidang `timeout` (dalam detik).
- Hook `PostToolUse` tidak dapat membatalkan tindakan karena alat sudah dieksekusi.
- Hook `PermissionRequest` tidak aktif dalam mode non-interaktif (`-p`). Gunakan hooks `PreToolUse` untuk keputusan izin otomatis.
- Hook `Stop` aktif kapan pun Claude selesai merespons, bukan hanya saat penyelesaian tugas. Mereka tidak aktif pada interupsi pengguna.

Hook tidak aktif

Hook dikonfigurasi tetapi tidak pernah dieksekusi.

- Jalankan `/hooks` dan konfirmasi hook muncul di bawah acara yang benar
- Periksa bahwa pola matcher cocok dengan nama alat dengan tepat (matchers peka huruf besar-kecil)
- Verifikasi Anda memicu jenis acara yang benar (misalnya, `PreToolUse` aktif sebelum eksekusi alat, `PostToolUse` aktif setelah)
- Jika menggunakan hooks `PermissionRequest` dalam mode non-interaktif (`-p`), beralih ke `PreToolUse` sebagai gantinya

Kesalahan hook dalam output

Anda melihat pesan seperti “PreToolUse hook error: …” dalam transkrip.

- Skrip Anda keluar dengan kode non-nol secara tidak terduga. Uji secara manual dengan menyalurkan JSON sampel:

```
echo '{"tool_name":"Bash","tool_input":{"command":"ls"}}' | ./my-hook.sh
echo $?  # Check the exit code
```

- Jika Anda melihat “command not found”, gunakan jalur absolut atau `$CLAUDE_PROJECT_DIR` untuk mereferensikan skrip
- Jika Anda melihat “jq: command not found”, instal `jq` atau gunakan Python/Node.js untuk penguraian JSON

- Jika skrip tidak berjalan sama sekali, buatnya dapat dieksekusi: `chmod +x ./my-hook.sh`

`/hooks` menunjukkan tidak ada hooks yang dikonfigurasi

Anda mengedit file pengaturan tetapi hooks tidak muncul dalam menu.

- Mulai ulang sesi Anda atau buka `/hooks` untuk memuat ulang. Hooks yang ditambahkan melalui menu `/hooks` berlaku segera, tetapi edit file manual memerlukan reload.
- Verifikasi JSON Anda valid (koma trailing dan komentar tidak diizinkan)
- Konfirmasi file pengaturan berada di lokasi yang benar: `.claude/settings.json` untuk hooks proyek, `~/.claude/settings.json` untuk hooks global

Hook Stop berjalan selamanya

Claude terus bekerja dalam loop tak terbatas daripada berhenti.

Skrip hook Stop Anda perlu memeriksa apakah sudah memicu kelanjutan. Uraikan bidang `stop_hook_active` dari input JSON dan keluar lebih awal jika `true`:

```
#!/bin/bash
INPUT=$(cat)
if [ "$(echo "$INPUT" | jq -r '.stop_hook_active')" = "true" ]; then
  exit 0  # Allow Claude to stop
fi
## ... rest of your hook logic
```

Validasi JSON gagal

Claude Code menampilkan kesalahan penguraian JSON meskipun skrip hook Anda menampilkan JSON yang valid.

Ketika Claude Code menjalankan hook, ia menelurkan shell yang bersumber dari profil Anda (`~/.zshrc` atau `~/.bashrc`). Jika profil Anda berisi pernyataan `echo` tanpa syarat, output itu ditambahkan ke JSON hook Anda:

```
Shell ready on arm64
{"decision": "block", "reason": "Not allowed"}
```

Claude Code mencoba menguraikan ini sebagai JSON dan gagal. Untuk memperbaiki ini, bungkus pernyataan echo dalam profil shell Anda sehingga hanya berjalan di shell interaktif:

```
## In ~/.zshrc or ~/.bashrc
if [[ $- == *i* ]]; then
  echo "Shell ready"
fi
```

Variabel `$-` berisi flag shell, dan `i` berarti interaktif. Hooks berjalan di shell non-interaktif, jadi echo dilewati.

Teknik debug

Alihkan mode verbose dengan `Ctrl+O` untuk melihat output hook dalam transkrip, atau jalankan `claude --debug` untuk detail eksekusi lengkap termasuk hooks mana yang cocok dan kode keluar mereka.

Pelajari lebih lanjut

- Referensi Hooks: skema acara lengkap, format output JSON, hooks asinkron, dan hooks alat MCP
- Pertimbangan keamanan: tinjau sebelum menerapkan hooks di lingkungan bersama atau produksi
- Contoh validator perintah Bash: implementasi referensi lengkap

Hubungkan Claude Code ke alat melalui MCP

> *Pelajari cara menghubungkan Claude Code ke alat Anda dengan Model Context Protocol.*

Claude Code dapat terhubung ke ratusan alat eksternal dan sumber data melalui Model Context Protocol (MCP), standar sumber terbuka untuk integrasi AI-alat. Server MCP memberikan Claude Code akses ke alat, database, dan API Anda.

Apa yang dapat Anda lakukan dengan MCP

Dengan server MCP yang terhubung, Anda dapat meminta Claude Code untuk:

- **Menerapkan fitur dari pelacak masalah**: "Tambahkan fitur yang dijelaskan dalam masalah JIRA ENG-4521 dan buat PR di GitHub."
- **Menganalisis data pemantauan**: "Periksa Sentry dan Statsig untuk memeriksa penggunaan fitur yang dijelaskan dalam ENG-4521."
- **Menanyakan database**: "Temukan email 10 pengguna acak yang menggunakan fitur ENG-4521, berdasarkan database PostgreSQL kami."
- **Mengintegrasikan desain**: "Perbarui template email standar kami berdasarkan desain Figma baru yang diposting di Slack"
- **Mengotomatisasi alur kerja**: "Buat draf Gmail mengundang 10 pengguna ini ke sesi umpan balik tentang fitur baru."

Server MCP populer

Berikut adalah beberapa server MCP yang umum digunakan yang dapat Anda hubungkan ke Claude Code:

> **Warning:**
>
> Gunakan server MCP pihak ketiga dengan risiko Anda sendiri - Anthropic belum memverifikasi kebenaran atau keamanan semua server ini. Pastikan Anda mempercayai server MCP yang Anda instal. Berhati-hatilah terutama saat menggunakan server MCP yang dapat mengambil konten yang tidak dipercaya, karena ini dapat mengekspos Anda ke risiko injeksi prompt.

Server	Description	Command
Notion	Connect your Notion workspace to search, update, and power workflows across tools	`claude mcp add --transport http notion https://mcp.notion.com/mcp`
Canva	Search, create, autofill, and export Canva designs	`claude mcp add --transport http canva https://mcp.canva.com/mcp`
Figma	Generate diagrams and better code from Figma context	`claude mcp add --transport http figma-remote-mcp https://mcp.figma.com/mcp`
Atlassian	Access Jira & Confluence from Claude	`claude mcp add --transport http atlassian https://mcp.atlassian.com/v1/mcp`
Linear	Manage issues, projects & team workflows in Linear	`claude mcp add --transport http linear https://mcp.linear.app/mcp`
monday.com	Manage projects, boards, and workflows in monday.com	`claude mcp add --transport http monday https://mcp.monday.com/mcp`
Intercom	Access to Intercom data for better customer insights	`claude mcp add --transport http intercom https://mcp.intercom.com/mcp`
Vercel	Analyze, debug, and manage projects and deployments	`claude mcp add --transport http vercel https://mcp.vercel.com`

Server	Description	Command
Granola	The AI notepad for meetings	`claude mcp add --transport http granola https://mcp.granola.ai/mcp`
Asana	Connect to Asana to coordinate tasks, projects, and goals	`claude mcp add --transport streamable-http asana https://mcp.asana.com/v2/mcp`
Miro	Access and create new content on Miro boards	`claude mcp add --transport http miro https://mcp.miro.com/`
Sentry	Search, query, and debug errors intelligently	`claude mcp add --transport http sentry https://mcp.sentry.dev/mcp`
Supabase	Manage databases, authentication, and storage	`claude mcp add --transport http supabase https://mcp.supabase.com/mcp`
Hugging Face	Access the Hugging Face Hub and thousands of Gradio Apps	`claude mcp add --transport http hugging-face https://huggingface.co/mcp`
Context7	Up-to-date docs for LLMs and AI code editors	`claude mcp add --transport http context7 https://mcp.context7.com/mcp`
Stripe	Payment processing and financial infrastructure tools	`claude mcp add --transport http stripe https://mcp.stripe.com`

Server	Description	Command
Microsoft Learn	Search trusted Microsoft docs to power your development	`claude mcp add --transport http microsoft-learn https://learn.microsoft.com/api/mcp`
Clay	Find prospects. Research accounts. Personalize outreach	`claude mcp add --transport http clay https://api.clay.com/v3/mcp`
Webflow	Manage Webflow CMS, pages, assets and sites	`claude mcp add --transport http webflow https://mcp.webflow.com/mcp`
Cloudflare	Build applications with compute, storage, and AI	`claude mcp add --transport http cloudflare https://bindings.mcp.cloudflare.com/mcp`
Ramp	Search, access, and analyze your Ramp financial data	`claude mcp add --transport http ramp https://ramp-mcp-remote.ramp.com/mcp`
ZoomInfo	Enrich contacts & accounts with GTM intelligence	`claude mcp add --transport http zoominfo https://mcp.zoominfo.com/mcp`
Netlify	Create, deploy, manage, and secure websites on Netlify	`claude mcp add --transport http netlify https://netlify-mcp.netlify.app/mcp`
Make	Run Make scenarios and manage your Make account	`claude mcp add --transport http make https://mcp.make.com`

Server	Description	Command
GoDaddy	Search domains and check availability	`claude mcp add --transport http godaddy https://api.godaddy.com/v1/domains/mcp`
Google Cloud BigQuery	BigQuery: Advanced analytical insights for agents	`claude mcp add --transport http bigquery https://bigquery.googleapis.com/mcp`
PayPal	Access PayPal payments platform	`claude mcp add --transport http paypal https://mcp.paypal.com/mcp`
PostHog	Query, analyze, and manage your PostHog insights	`claude mcp add --transport http posthog https://mcp.posthog.com/mcp`
Similarweb	Real time web, mobile app, and market data	`claude mcp add --transport http similarweb https://mcp.similarweb.com`
Crypto.com	Real time prices, orders, charts, and more for crypto	`claude mcp add --transport http crypto.com https://mcp.crypto.com/market-data/mcp`
Attio	Search, manage, and update your Attio CRM from Claude	`claude mcp add --transport http attio https://mcp.attio.com/mcp`
Trivago	Find your ideal hotel at the best price	`claude mcp add --transport http trivago https://mcp.trivago.com/mcp`

Server	Description	Command
Jam	Record screen and collect automatic context for issues	`claude mcp add --transport http jam https://mcp.jam.dev/mcp`
Consensus	Explore scientific research	`claude mcp add --transport http consensus https://mcp.consensus.app/mcp`
Clockwise	Advanced scheduling and time management for work	`claude mcp add --transport http clockwise https://mcp.getclockwise.com/mcp`
Square	Search and manage transaction, merchant, and payment data	`claude mcp add --transport sse square https://mcp.squareup.com/sse`
Egnyte	Securely access and analyze Egnyte content	`claude mcp add --transport http egnyte https://mcp-server.egnyte.com/mcp`
Pylon	Search and manage Pylon support issues	`claude mcp add --transport http pylon https://mcp.usepylon.com/`
Honeycomb	Query and explore observability data and SLOs	`claude mcp add --transport http honeycomb https://mcp.honeycomb.io/mcp`

Note:

Membutuhkan integrasi spesifik? Temukan ratusan server MCP lainnya di GitHub, atau buat server Anda sendiri menggunakan MCP SDK.

Menginstal server MCP

Server MCP dapat dikonfigurasi dengan tiga cara berbeda tergantung pada kebutuhan Anda:

Opsi 1: Tambahkan server HTTP jarak jauh

Server HTTP adalah opsi yang direkomendasikan untuk terhubung ke server MCP jarak jauh. Ini adalah transport yang paling banyak didukung untuk layanan berbasis cloud.

```
## Sintaks dasar
claude mcp add --transport http <name> <url>

## Contoh nyata: Hubungkan ke Notion
claude mcp add --transport http notion https://mcp.notion.com/mcp

## Contoh dengan token Bearer
claude mcp add --transport http secure-api https://api.example.com/mcp \
  --header "Authorization: Bearer your-token"
```

Opsi 2: Tambahkan server SSE jarak jauh

> **Warning:**
>
> Transport SSE (Server-Sent Events) sudah usang. Gunakan server HTTP sebagai gantinya, jika tersedia.

```
## Sintaks dasar
claude mcp add --transport sse <name> <url>

## Contoh nyata: Hubungkan ke Asana
claude mcp add --transport sse asana https://mcp.asana.com/sse

## Contoh dengan header autentikasi
claude mcp add --transport sse private-api https://api.company.com/sse \
  --header "X-API-Key: your-key-here"
```

Opsi 3: Tambahkan server stdio lokal

Server stdio berjalan sebagai proses lokal di mesin Anda. Mereka ideal untuk alat yang memerlukan akses sistem langsung atau skrip khusus.

```
## Sintaks dasar
claude mcp add [options] <name> -- <command> [args...]

## Contoh nyata: Tambahkan server Airtable
claude mcp add --transport stdio --env AIRTABLE_API_KEY=YOUR_KEY airtable \
  -- npx -y airtable-mcp-server
```

> **Note:**
>
> **Penting: Urutan opsi**
>
> Semua opsi (`--transport` , `--env` , `--scope` , `--header`) harus datang **sebelum** nama server. `--` (tanda hubung ganda) kemudian memisahkan nama server dari perintah dan argumen yang diteruskan ke server MCP.
>
> Sebagai contoh:
>
> - `claude mcp add --transport stdio myserver -- npx server` → menjalankan `npx server`
> - `claude mcp add --transport stdio --env KEY=value myserver -- python server.py --port 8080` → menjalankan `python server.py --port 8080` dengan `KEY=value` di lingkungan
>
> Ini mencegah konflik antara flag Claude dan flag server.

Mengelola server Anda

Setelah dikonfigurasi, Anda dapat mengelola server MCP Anda dengan perintah ini:

```
## Daftar semua server yang dikonfigurasi
claude mcp list

## Dapatkan detail untuk server tertentu
claude mcp get github

## Hapus server
claude mcp remove github

## (dalam Claude Code) Periksa status server
/mcp
```

Pembaruan alat dinamis

Claude Code mendukung notifikasi `list_changed` MCP, memungkinkan server MCP untuk secara dinamis memperbarui alat, prompt, dan sumber daya yang tersedia tanpa memerlukan Anda untuk memutuskan dan menghubungkan kembali. Ketika server MCP mengirim notifikasi `list_changed`, Claude Code secara otomatis menyegarkan kemampuan yang tersedia dari server tersebut.

Tip:

Tips:

- Gunakan flag `--scope` untuk menentukan di mana konfigurasi disimpan:
- `local` (default): Hanya tersedia untuk Anda di proyek saat ini (disebut `project` di versi yang lebih lama)
- `project`: Dibagikan dengan semua orang di proyek melalui file `.mcp.json`
- `user`: Tersedia untuk Anda di semua proyek (disebut `global` di versi yang lebih lama)
- Atur variabel lingkungan dengan flag `--env` (misalnya, `--env KEY=value`)
- Konfigurasi waktu tunggu startup server MCP menggunakan variabel lingkungan MCP_TIMEOUT (misalnya, `MCP_TIMEOUT=10000 claude` menetapkan waktu tunggu 10 detik)
- Claude Code akan menampilkan peringatan ketika output alat MCP melebihi 10.000 token. Untuk meningkatkan batas ini, atur variabel lingkungan `MAX_MCP_OUTPUT_TOKENS` (misalnya, `MAX_MCP_OUTPUT_TOKENS=50000`)
- Gunakan `/mcp` untuk autentikasi dengan server jarak jauh yang memerlukan autentikasi OAuth 2.0

Warning:

Pengguna Windows: Pada Windows asli (bukan WSL), server MCP lokal yang menggunakan `npx` memerlukan pembungkus `cmd /c` untuk memastikan eksekusi yang tepat.

```
## Ini membuat command="cmd" yang dapat dieksekusi Windows
claude mcp add --transport stdio my-server -- cmd /c npx -y @some/package
```

Tanpa pembungkus `cmd /c`, Anda akan mengalami kesalahan "Connection closed" karena Windows tidak dapat langsung menjalankan `npx`. (Lihat catatan di atas untuk penjelasan parameter `--`.)

Server MCP yang disediakan plugin

Plugin dapat menggabungkan server MCP, secara otomatis menyediakan alat dan integrasi ketika plugin diaktifkan. Server MCP plugin bekerja identik dengan server yang dikonfigurasi pengguna.

Cara kerja server MCP plugin:

- Plugin menentukan server MCP dalam `.mcp.json` di root plugin atau inline dalam `plugin.json`
- Ketika plugin diaktifkan, server MCP-nya dimulai secara otomatis
- Alat MCP plugin muncul bersama alat MCP yang dikonfigurasi secara manual
- Server plugin dikelola melalui instalasi plugin (bukan perintah `/mcp`)

Contoh konfigurasi MCP plugin:

Dalam `.mcp.json` di root plugin:

```
{
  "database-tools": {
    "command": "${CLAUDE_PLUGIN_ROOT}/servers/db-server",
    "args": ["--config", "${CLAUDE_PLUGIN_ROOT}/config.json"],
    "env": {
      "DB_URL": "${DB_URL}"
    }
  }
}
```

Atau inline dalam `plugin.json`:

```
{
  "name": "my-plugin",
  "mcpServers": {
    "plugin-api": {
      "command": "${CLAUDE_PLUGIN_ROOT}/servers/api-server",
      "args": ["--port", "8080"]
    }
  }
}
```

Fitur MCP plugin:

- **Siklus hidup otomatis**: Server dimulai ketika plugin diaktifkan, tetapi Anda harus memulai ulang Claude Code untuk menerapkan perubahan server MCP (mengaktifkan atau menonaktifkan)
- **Variabel lingkungan**: Gunakan `${CLAUDE_PLUGIN_ROOT}` untuk jalur relatif plugin
- **Akses lingkungan pengguna**: Akses ke variabel lingkungan yang sama seperti server yang dikonfigurasi secara manual
- **Jenis transport berganda**: Dukungan transport stdio, SSE, dan HTTP (dukungan transport dapat bervariasi menurut server)

Melihat server MCP plugin:

```
## Dalam Claude Code, lihat semua server MCP termasuk yang dari plugin
/mcp
```

Server plugin muncul dalam daftar dengan indikator yang menunjukkan mereka berasal dari plugin.

Manfaat server MCP plugin:

- **Distribusi bundel**: Alat dan server dikemas bersama
- **Pengaturan otomatis**: Tidak perlu konfigurasi MCP manual
- **Konsistensi tim**: Semua orang mendapatkan alat yang sama ketika plugin diinstal

Lihat referensi komponen plugin untuk detail tentang penggabungan server MCP dengan plugin.

Cakupan instalasi MCP

Server MCP dapat dikonfigurasi pada tiga tingkat cakupan yang berbeda, masing-masing melayani tujuan yang berbeda untuk mengelola aksesibilitas server dan berbagi. Memahami cakupan ini membantu Anda menentukan cara terbaik untuk mengonfigurasi server sesuai kebutuhan spesifik Anda.

Cakupan lokal

Server dengan cakupan lokal mewakili tingkat konfigurasi default dan disimpan dalam `~/.claude.json` di bawah jalur proyek Anda. Server ini tetap pribadi untuk Anda dan hanya dapat diakses saat bekerja dalam direktori proyek saat ini. Cakupan ini ideal untuk server pengembangan pribadi, konfigurasi eksperimental, atau server yang berisi kredensial sensitif yang tidak boleh dibagikan.

> **Note:**
>
> Istilah "cakupan lokal" untuk server MCP berbeda dari pengaturan lokal umum. Server MCP dengan cakupan lokal disimpan dalam `~/.claude.json` (direktori home Anda), sementara pengaturan lokal umum menggunakan `.claude/settings.local.json` (di direktori proyek). Lihat Pengaturan untuk detail tentang lokasi file pengaturan.

```
## Tambahkan server dengan cakupan lokal (default)
claude mcp add --transport http stripe https://mcp.stripe.com

## Tentukan cakupan lokal secara eksplisit
claude mcp add --transport http stripe --scope local https://mcp.stripe.com
```

Cakupan proyek

Server dengan cakupan proyek memungkinkan kolaborasi tim dengan menyimpan konfigurasi dalam file `.mcp.json` di direktori root proyek Anda. File ini dirancang untuk diperiksa ke dalam kontrol versi, memastikan semua anggota tim memiliki akses ke alat dan layanan MCP yang sama. Ketika Anda menambahkan server dengan cakupan proyek, Claude Code secara otomatis membuat atau memperbarui file ini dengan struktur konfigurasi yang sesuai.

```
## Tambahkan server dengan cakupan proyek
claude mcp add --transport http paypal --scope project https://mcp.paypal.com/mcp
```

File `.mcp.json` yang dihasilkan mengikuti format standar:

```
{
  "mcpServers": {
    "shared-server": {
      "command": "/path/to/server",
      "args": [],
      "env": {}
    }
  }
}
```

Untuk alasan keamanan, Claude Code meminta persetujuan sebelum menggunakan server dengan cakupan proyek dari file `.mcp.json`. Jika Anda perlu mengatur ulang pilihan persetujuan ini, gunakan perintah `claude mcp reset-project-choices`.

Cakupan pengguna

Server dengan cakupan pengguna disimpan dalam `~/.claude.json` dan menyediakan aksesibilitas lintas proyek, menjadikannya tersedia di semua proyek di mesin Anda sambil tetap pribadi untuk akun pengguna Anda. Cakupan ini bekerja dengan baik untuk server utilitas pribadi, alat pengembangan, atau layanan yang sering Anda gunakan di berbagai proyek.

```
## Tambahkan server pengguna
claude mcp add --transport http hubspot --scope user https://mcp.hubspot.com/
anthropic
```

Memilih cakupan yang tepat

Pilih cakupan Anda berdasarkan:

- **Cakupan lokal**: Server pribadi, konfigurasi eksperimental, atau kredensial sensitif khusus untuk satu proyek
- **Cakupan proyek**: Server bersama tim, alat khusus proyek, atau layanan yang diperlukan untuk kolaborasi
- **Cakupan pengguna**: Utilitas pribadi yang diperlukan di berbagai proyek, alat pengembangan, atau layanan yang sering digunakan

Note:

Di mana server MCP disimpan?

- **Cakupan pengguna dan lokal**: `~/.claude.json` (dalam field `mcpServers` atau di bawah jalur proyek)
- **Cakupan proyek**: `.mcp.json` di root proyek Anda (diperiksa ke dalam kontrol sumber)
- **Dikelola**: `managed-mcp.json` di direktori sistem (lihat Konfigurasi MCP yang dikelola)

Hierarki cakupan dan prioritas

Konfigurasi server MCP mengikuti hierarki prioritas yang jelas. Ketika server dengan nama yang sama ada di berbagai cakupan, sistem menyelesaikan konflik dengan memprioritaskan server dengan cakupan lokal terlebih dahulu, diikuti oleh server dengan cakupan proyek, dan akhirnya server dengan cakupan pengguna. Desain ini memastikan bahwa konfigurasi pribadi dapat mengganti yang dibagikan jika diperlukan.

Ekspansi variabel lingkungan dalam `.mcp.json`

Claude Code mendukung ekspansi variabel lingkungan dalam file `.mcp.json`, memungkinkan tim untuk berbagi konfigurasi sambil mempertahankan fleksibilitas untuk jalur spesifik mesin dan nilai sensitif seperti kunci API.

Sintaks yang didukung:

- `${VAR}` - Berkembang menjadi nilai variabel lingkungan `VAR`
- `${VAR:-default}` - Berkembang menjadi `VAR` jika diatur, jika tidak menggunakan `default`

Lokasi ekspansi: Variabel lingkungan dapat berkembang dalam:

- `command` - Jalur executable server
- `args` - Argumen baris perintah
- `env` - Variabel lingkungan yang diteruskan ke server
- `url` - Untuk jenis server HTTP
- `headers` - Untuk autentikasi server HTTP

Contoh dengan ekspansi variabel:

```
{
  "mcpServers": {
    "api-server": {
      "type": "http",
      "url": "${API_BASE_URL:-https://api.example.com}/mcp",
      "headers": {
        "Authorization": "Bearer ${API_KEY}"
      }
    }
  }
}
```

Jika variabel lingkungan yang diperlukan tidak diatur dan tidak memiliki nilai default, Claude Code akan gagal mengurai konfigurasi.

Contoh praktis

{/* ### Contoh: Otomatisasi pengujian browser dengan Playwright

```
claude mcp add --transport stdio playwright -- npx -y @playwright/mcp@latest
```

Kemudian tulis dan jalankan tes browser:

```
Test if the login flow works with test@example.com
```

```
Take a screenshot of the checkout page on mobile
```

```
  Verify that the search feature returns results
  ``` */}

Contoh: Pantau kesalahan dengan Sentry

```bash  theme={null}
claude mcp add --transport http sentry https://mcp.sentry.dev/mcp
```

Autentikasi dengan akun Sentry Anda:

```
/mcp
```

Kemudian debug masalah produksi:

```
What are the most common errors in the last 24 hours?
```

```
Show me the stack trace for error ID abc123
```

```
Which deployment introduced these new errors?
```

Contoh: Hubungkan ke GitHub untuk tinjauan kode

```
claude mcp add --transport http github https://api.githubcopilot.com/mcp/
```

Autentikasi jika diperlukan dengan memilih “Authenticate” untuk GitHub:

```
/mcp
```

Kemudian bekerja dengan GitHub:

```
Review PR #456 and suggest improvements
```

```
Create a new issue for the bug we just found
```

```
Show me all open PRs assigned to me
```

Contoh: Tanyakan database PostgreSQL Anda

```
claude mcp add --transport stdio db -- npx -y @bytebase/dbhub \
  --dsn "postgresql://readonly:pass@prod.db.com:5432/analytics"
```

Kemudian tanyakan database Anda secara alami:

```
What's our total revenue this month?
```

```
Show me the schema for the orders table
```

```
Find customers who haven't made a purchase in 90 days
```

Autentikasi dengan server MCP jarak jauh

Banyak server MCP berbasis cloud memerlukan autentikasi. Claude Code mendukung OAuth 2.0 untuk koneksi yang aman.

Step 1: Tambahkan server yang memerlukan autentikasi

Sebagai contoh:

```
claude mcp add --transport http sentry https://mcp.sentry.dev/mcp
```

Step 2: Gunakan perintah /mcp dalam Claude Code

Dalam Claude Code, gunakan perintah:

```
/mcp
```

Kemudian ikuti langkah-langkah di browser Anda untuk login.

> **Tip:**
>
> Tips:
>
> - Token autentikasi disimpan dengan aman dan disegarkan secara otomatis
> - Gunakan “Clear authentication” dalam menu `/mcp` untuk mencabut akses

- Jika browser Anda tidak terbuka secara otomatis, salin URL yang disediakan dan buka secara manual
- Jika pengalihan browser gagal dengan kesalahan koneksi setelah autentikasi, tempel URL callback lengkap dari bilah alamat browser Anda ke prompt URL yang muncul di Claude Code
- Autentikasi OAuth bekerja dengan server HTTP

Gunakan port callback OAuth tetap

Beberapa server MCP memerlukan URI pengalihan tertentu yang terdaftar sebelumnya. Secara default, Claude Code memilih port acak yang tersedia untuk callback OAuth. Gunakan `--callback-port` untuk memperbaiki port sehingga cocok dengan URI pengalihan yang telah terdaftar sebelumnya dalam bentuk `http://localhost:PORT/callback`.

Anda dapat menggunakan `--callback-port` sendiri (dengan pendaftaran klien dinamis) atau bersama dengan `--client-id` (dengan kredensial yang telah dikonfigurasi sebelumnya).

```
## Port callback tetap dengan pendaftaran klien dinamis
claude mcp add --transport http \
  --callback-port 8080 \
  my-server https://mcp.example.com/mcp
```

Gunakan kredensial OAuth yang telah dikonfigurasi sebelumnya

Beberapa server MCP tidak mendukung pengaturan OAuth otomatis. Jika Anda melihat kesalahan seperti “Incompatible auth server: does not support dynamic client registration,” server memerlukan kredensial yang telah dikonfigurasi sebelumnya. Daftarkan aplikasi OAuth melalui portal pengembang server terlebih dahulu, kemudian berikan kredensial saat menambahkan server.

Step 1: Daftarkan aplikasi OAuth dengan server

Buat aplikasi melalui portal pengembang server dan catat ID klien dan rahasia klien Anda.

Banyak server juga memerlukan URI pengalihan. Jika demikian, pilih port dan daftarkan URI pengalihan dalam format `http://localhost:PORT/callback`. Gunakan port yang sama dengan `--callback-port` di langkah berikutnya.

Step 2: Tambahkan server dengan kredensial Anda

Pilih salah satu metode berikut. Port yang digunakan untuk `--callback-port` dapat berupa port apa pun yang tersedia. Itu hanya perlu cocok dengan URI pengalihan yang Anda daftarkan di langkah sebelumnya.

claude mcp add

Gunakan `--client-id` untuk meneruskan ID klien aplikasi Anda. Flag `--client-secret` meminta rahasia dengan input yang disembunyikan:

```
claude mcp add --transport http \
  --client-id your-client-id --client-secret --callback-port 8080 \
  my-server https://mcp.example.com/mcp
```

claude mcp add-json

Sertakan objek `oauth` dalam konfigurasi JSON dan teruskan `--client-secret` sebagai flag terpisah:

```
claude mcp add-json my-server \
  '{"type":"http","url":"https://mcp.example.com/mcp","oauth":{"clientId":"your-client-id","callbackPort":8080}}' \
  --client-secret
```

claude mcp add-json (callback port only)

Gunakan `--callback-port` tanpa ID klien untuk memperbaiki port sambil menggunakan pendaftaran klien dinamis:

```
claude mcp add-json my-server \
  '{"type":"http","url":"https://mcp.example.com/mcp","oauth":{"callbackPort":8080}}'
```

CI / env var

Atur rahasia melalui variabel lingkungan untuk melewati prompt interaktif:

```
MCP_CLIENT_SECRET=your-secret claude mcp add --transport http \
  --client-id your-client-id --client-secret --callback-port 8080 \
  my-server https://mcp.example.com/mcp
```

Step 3: Autentikasi di Claude Code

Jalankan `/mcp` di Claude Code dan ikuti alur login browser.

> **Tip:**
>
> Tips:
>
> - Rahasia klien disimpan dengan aman di keychain sistem Anda (macOS) atau file kredensial, bukan di konfigurasi Anda
> - Jika server menggunakan klien OAuth publik tanpa rahasia, gunakan hanya `--client-id` tanpa `--client-secret`
> - `--callback-port` dapat digunakan dengan atau tanpa `--client-id`
> - Flag ini hanya berlaku untuk transport HTTP dan SSE. Mereka tidak berpengaruh pada server stdio
> - Gunakan `claude mcp get <name>` untuk memverifikasi bahwa kredensial OAuth dikonfigurasi untuk server

Ganti penemuan metadata OAuth

Jika server MCP Anda mengembalikan kesalahan pada endpoint metadata OAuth standar (`/.well-known/oauth-authorization-server`) tetapi mengekspos endpoint OIDC yang berfungsi, Anda dapat memberi tahu Claude Code untuk mengambil metadata OAuth langsung dari URL yang Anda tentukan, melewati rantai penemuan standar.

Atur `authServerMetadataUrl` dalam objek `oauth` dari konfigurasi server Anda di `.mcp.json`:

```
{
  "mcpServers": {
    "my-server": {
      "type": "http",
      "url": "https://mcp.example.com/mcp",
      "oauth": {
        "authServerMetadataUrl": "https://auth.example.com/.well-known/openid-
configuration"
      }
    }
  }
}
```

URL harus menggunakan `https://`. Opsi ini memerlukan Claude Code v2.1.64 atau lebih baru.

Tambahkan server MCP dari konfigurasi JSON

Jika Anda memiliki konfigurasi JSON untuk server MCP, Anda dapat menambahkannya secara langsung:

Step 1: Tambahkan server MCP dari JSON

```
## Sintaks dasar
claude mcp add-json <name> '<json>'

## Contoh: Menambahkan server HTTP dengan konfigurasi JSON
claude mcp add-json weather-api '{"type":"http","url":"https://api.weather.com/
mcp","headers":{"Authorization":"Bearer token"}}'

## Contoh: Menambahkan server stdio dengan konfigurasi JSON
claude mcp add-json local-weather '{"type":"stdio","command":"/path/to/weather-
cli","args":["--api-key","abc123"],"env":{"CACHE_DIR":"/tmp"}}'

## Contoh: Menambahkan server HTTP dengan kredensial OAuth yang telah
dikonfigurasi sebelumnya
claude mcp add-json my-server '{"type":"http","url":"https://mcp.example.com/
mcp","oauth":{"clientId":"your-client-id","callbackPort":8080}}' --client-secret
```

Step 2: Verifikasi server ditambahkan

```
claude mcp get weather-api
```

Tip:

Tips:

- Pastikan JSON diloloskan dengan benar di shell Anda
- JSON harus sesuai dengan skema konfigurasi server MCP
- Anda dapat menggunakan `--scope user` untuk menambahkan server ke konfigurasi pengguna Anda alih-alih yang spesifik proyek

Impor server MCP dari Claude Desktop

Jika Anda telah mengonfigurasi server MCP di Claude Desktop, Anda dapat mengimpornya:

Step 1: Impor server dari Claude Desktop

```
## Sintaks dasar
claude mcp add-from-claude-desktop
```

Step 2: Pilih server mana yang akan diimpor

Setelah menjalankan perintah, Anda akan melihat dialog interaktif yang memungkinkan Anda memilih server mana yang ingin Anda impor.

Step 3: Verifikasi server diimpor

```
claude mcp list
```

Tip:

Tips:

- Fitur ini hanya bekerja di macOS dan Windows Subsystem for Linux (WSL)
- Ini membaca file konfigurasi Claude Desktop dari lokasi standarnya di platform tersebut
- Gunakan flag `--scope user` untuk menambahkan server ke konfigurasi pengguna Anda
- Server yang diimpor akan memiliki nama yang sama seperti di Claude Desktop
- Jika server dengan nama yang sama sudah ada, mereka akan mendapatkan akhiran numerik (misalnya, `server_1`)

Gunakan server MCP dari Claude.ai

Jika Anda telah masuk ke Claude Code dengan akun Claude.ai, server MCP yang telah Anda tambahkan di Claude.ai secara otomatis tersedia di Claude Code:

Step 1: Konfigurasi server MCP di Claude.ai

Tambahkan server di claude.ai/settings/connectors. Pada paket Tim dan Enterprise, hanya admin yang dapat menambahkan server.

Step 2: Autentikasi server MCP

Selesaikan langkah autentikasi yang diperlukan di Claude.ai.

Step 3: Lihat dan kelola server di Claude Code

Di Claude Code, gunakan perintah:

```
/mcp
```

Server Claude.ai muncul dalam daftar dengan indikator yang menunjukkan mereka berasal dari Claude.ai.

Untuk menonaktifkan server MCP claude.ai di Claude Code, atur variabel lingkungan `ENABLE_CLAUDEAI_MCP_SERVERS` ke `false`:

```
ENABLE_CLAUDEAI_MCP_SERVERS=false claude
```

Gunakan Claude Code sebagai server MCP

Anda dapat menggunakan Claude Code itu sendiri sebagai server MCP yang dapat terhubung oleh aplikasi lain:

```
## Mulai Claude sebagai server MCP stdio
claude mcp serve
```

Anda dapat menggunakan ini di Claude Desktop dengan menambahkan konfigurasi ini ke claude_desktop_config.json:

```
{
  "mcpServers": {
    "claude-code": {
      "type": "stdio",
      "command": "claude",
      "args": ["mcp", "serve"],
      "env": {}
    }
  }
}
```

> **Warning:**

Mengonfigurasi jalur executable: Field `command` harus mereferensikan executable Claude Code. Jika perintah `claude` tidak ada di PATH sistem Anda, Anda perlu menentukan jalur lengkap ke executable.

Untuk menemukan jalur lengkap:

```
which claude
```

Kemudian gunakan jalur lengkap dalam konfigurasi Anda:

```
{
  "mcpServers": {
    "claude-code": {
      "type": "stdio",
      "command": "/full/path/to/claude",
      "args": ["mcp", "serve"],
      "env": {}
    }
  }
}
```

Tanpa jalur executable yang benar, Anda akan mengalami kesalahan seperti `spawn claude ENOENT`.

Tip:

Tips:

- Server menyediakan akses ke alat Claude seperti View, Edit, LS, dll.
- Di Claude Desktop, coba minta Claude untuk membaca file di direktori, membuat edit, dan lainnya.
- Perhatikan bahwa server MCP ini hanya mengekspos alat Claude Code ke klien MCP Anda, jadi klien Anda sendiri bertanggung jawab untuk menerapkan konfirmasi pengguna untuk panggilan alat individual.

Batas output MCP dan peringatan

Ketika alat MCP menghasilkan output besar, Claude Code membantu mengelola penggunaan token untuk mencegah membanjiri konteks percakapan Anda:

- **Ambang batas peringatan output**: Claude Code menampilkan peringatan ketika output alat MCP apa pun melebihi 10.000 token

- **Batas yang dapat dikonfigurasi**: Anda dapat menyesuaikan token output MCP maksimum yang diizinkan menggunakan variabel lingkungan `MAX_MCP_OUTPUT_TOKENS`
- **Batas default**: Maksimum default adalah 25.000 token

Untuk meningkatkan batas untuk alat yang menghasilkan output besar:

```
## Atur batas lebih tinggi untuk output alat MCP
export MAX_MCP_OUTPUT_TOKENS=50000
claude
```

Ini sangat berguna saat bekerja dengan server MCP yang:

- Menanyakan dataset atau database besar
- Menghasilkan laporan atau dokumentasi terperinci
- Memproses file log atau informasi debugging yang luas

> **Warning:**
>
> Jika Anda sering mengalami peringatan output dengan server MCP tertentu, pertimbangkan untuk meningkatkan batas atau mengonfigurasi server untuk membuat halaman atau memfilter responsnya.

Gunakan sumber daya MCP

Server MCP dapat mengekspos sumber daya yang dapat Anda referensikan menggunakan penyebutan @, mirip dengan cara Anda mereferensikan file.

Referensikan sumber daya MCP

Step 1: Daftar sumber daya yang tersedia

Ketik `@` dalam prompt Anda untuk melihat sumber daya yang tersedia dari semua server MCP yang terhubung. Sumber daya muncul bersama file dalam menu pelengkapan otomatis.

Step 2: Referensikan sumber daya tertentu

Gunakan format `@server:protocol://resource/path` untuk mereferensikan sumber daya:

```
Can you analyze @github:issue://123 and suggest a fix?
```

```
Please review the API documentation at @docs:file://api/authentication
```

Step 3: Referensi sumber daya berganda

Anda dapat mereferensikan beberapa sumber daya dalam satu prompt:

```
Compare @postgres:schema://users with @docs:file://database/user-model
```

> **Tip:**
>
> Tips:
>
> - Sumber daya secara otomatis diambil dan disertakan sebagai lampiran saat direferensikan
> - Jalur sumber daya dapat dicari fuzzy dalam pelengkapan otomatis penyebutan @
> - Claude Code secara otomatis menyediakan alat untuk membuat daftar dan membaca sumber daya MCP ketika server mendukungnya
> - Sumber daya dapat berisi jenis konten apa pun yang disediakan server MCP (teks, JSON, data terstruktur, dll.)

Skala dengan Pencarian Alat MCP

Ketika Anda memiliki banyak server MCP yang dikonfigurasi, definisi alat dapat mengonsumsi sebagian besar jendela konteks Anda. Pencarian Alat MCP menyelesaikan ini dengan memuat alat secara dinamis sesuai permintaan alih-alih memuat semuanya sebelumnya.

Cara kerjanya

Claude Code secara otomatis mengaktifkan Pencarian Alat ketika deskripsi alat MCP Anda akan mengonsumsi lebih dari 10% jendela konteks. Anda dapat menyesuaikan ambang batas ini atau menonaktifkan pencarian alat sepenuhnya. Ketika dipicu:

1. Alat MCP ditangguhkan daripada dimuat ke konteks sebelumnya
2. Claude menggunakan alat pencarian untuk menemukan alat MCP yang relevan saat diperlukan
3. Hanya alat yang benar-benar dibutuhkan Claude yang dimuat ke konteks
4. Alat MCP terus bekerja persis seperti sebelumnya dari perspektif Anda

Untuk penulis server MCP

Jika Anda membangun server MCP, field instruksi server menjadi lebih berguna dengan Pencarian Alat yang diaktifkan. Instruksi server membantu Claude memahami kapan harus mencari alat Anda, mirip dengan cara skills bekerja.

Tambahkan instruksi server yang jelas dan deskriptif yang menjelaskan:

- Kategori tugas apa yang ditangani alat Anda
- Kapan Claude harus mencari alat Anda
- Kemampuan utama yang disediakan server Anda

Konfigurasi pencarian alat

Pencarian alat berjalan dalam mode otomatis secara default, artinya diaktifkan hanya ketika definisi alat MCP Anda melebihi ambang batas konteks. Jika Anda memiliki beberapa alat, mereka dimuat secara normal tanpa pencarian alat. Fitur ini memerlukan model yang mendukung blok `tool_reference`: Sonnet 4 dan lebih baru, atau Opus 4 dan lebih baru. Model Haiku tidak mendukung pencarian alat.

Kontrol perilaku pencarian alat dengan variabel lingkungan `ENABLE_TOOL_SEARCH`:

Nilai	Perilaku
`auto`	Diaktifkan ketika alat MCP melebihi 10% konteks (default)
`auto:<N>`	Diaktifkan pada ambang batas khusus, di mana `<N>` adalah persentase (misalnya, `auto:5` untuk 5%)
`true`	Selalu diaktifkan
`false`	Dinonaktifkan, semua alat MCP dimuat sebelumnya

```
## Gunakan ambang batas khusus 5%
ENABLE_TOOL_SEARCH=auto:5 claude

## Nonaktifkan pencarian alat sepenuhnya
ENABLE_TOOL_SEARCH=false claude
```

Atau atur nilai dalam field `env` settings.json Anda.

Anda juga dapat menonaktifkan alat MCPSearch secara khusus menggunakan pengaturan `disallowedTools`:

```
{
  "permissions": {
    "deny": ["MCPSearch"]
  }
}
```

Gunakan prompt MCP sebagai perintah

Server MCP dapat mengekspos prompt yang menjadi tersedia sebagai perintah di Claude Code.

Jalankan prompt MCP

Step 1: Temukan prompt yang tersedia

Ketik `/` untuk melihat semua perintah yang tersedia, termasuk yang dari server MCP. Prompt MCP muncul dengan format `/mcp__servername__promptname`.

Step 2: Jalankan prompt tanpa argumen

```
/mcp__github__list_prs
```

Step 3: Jalankan prompt dengan argumen

Banyak prompt menerima argumen. Teruskan mereka dipisahkan spasi setelah perintah:

```
/mcp__github__pr_review 456
```

```
/mcp__jira__create_issue "Bug in login flow" high
```

Tip:

Tips:

- Prompt MCP ditemukan secara dinamis dari server yang terhubung
- Argumen diurai berdasarkan parameter yang ditentukan prompt
- Hasil prompt disuntikkan langsung ke dalam percakapan
- Nama server dan prompt dinormalisasi (spasi menjadi garis bawah)

Konfigurasi MCP yang dikelola

Untuk organisasi yang memerlukan kontrol terpusat atas server MCP, Claude Code mendukung dua opsi konfigurasi:

1. **Kontrol eksklusif dengan** `managed-mcp.json`: Terapkan set server MCP tetap yang tidak dapat dimodifikasi atau diperluas pengguna
2. **Kontrol berbasis kebijakan dengan daftar putih/daftar hitam**: Izinkan pengguna menambahkan server mereka sendiri, tetapi batasi server mana yang diizinkan

Opsi ini memungkinkan administrator IT untuk:

- **Kontrol server MCP mana yang dapat diakses karyawan**: Terapkan set server MCP yang disetujui secara standar di seluruh organisasi
- **Cegah server MCP yang tidak sah**: Batasi pengguna dari menambahkan server MCP yang tidak disetujui
- **Nonaktifkan MCP sepenuhnya**: Hapus fungsionalitas MCP sepenuhnya jika diperlukan

Opsi 1: Kontrol eksklusif dengan managed-mcp.json

Ketika Anda menerapkan file `managed-mcp.json`, file tersebut mengambil **kontrol eksklusif** atas semua server MCP. Pengguna tidak dapat menambahkan, memodifikasi, atau menggunakan server MCP apa pun selain yang ditentukan dalam file ini. Ini adalah pendekatan paling sederhana untuk organisasi yang menginginkan kontrol penuh.

Administrator sistem menerapkan file konfigurasi ke direktori sistem:

- macOS: `/Library/Application Support/ClaudeCode/managed-mcp.json`
- Linux dan WSL: `/etc/claude-code/managed-mcp.json`
- Windows: `C:\Program Files\ClaudeCode\managed-mcp.json`

> **Note:**
>
> Ini adalah jalur sistem (bukan direktori home pengguna seperti `~/Library/...`) yang memerlukan hak istimewa administrator. Mereka dirancang untuk diterapkan oleh administrator IT.

File `managed-mcp.json` menggunakan format yang sama seperti file `.mcp.json` standar:

```
{
  "mcpServers": {
    "github": {
      "type": "http",
      "url": "https://api.githubcopilot.com/mcp/"
    },
    "sentry": {
      "type": "http",
      "url": "https://mcp.sentry.dev/mcp"
    },
    "company-internal": {
      "type": "stdio",
      "command": "/usr/local/bin/company-mcp-server",
      "args": ["--config", "/etc/company/mcp-config.json"],
      "env": {
        "COMPANY_API_URL": "https://internal.company.com"
      }
    }
  }
}
```

Opsi 2: Kontrol berbasis kebijakan dengan daftar putih dan daftar hitam

Alih-alih mengambil kontrol eksklusif, administrator dapat mengizinkan pengguna mengonfigurasi server MCP mereka sendiri sambil memberlakukan pembatasan pada server mana yang diizinkan. Pendekatan ini menggunakan `allowedMcpServers` dan `deniedMcpServers` dalam file pengaturan yang dikelola.

> **Note:**
>
> **Memilih antara opsi**: Gunakan Opsi 1 (`managed-mcp.json`) ketika Anda ingin menerapkan set server tetap tanpa kustomisasi pengguna. Gunakan Opsi 2 (daftar putih/daftar hitam) ketika Anda ingin mengizinkan pengguna menambahkan server mereka sendiri dalam batasan kebijakan.

Opsi pembatasan

Setiap entri dalam daftar putih atau daftar hitam dapat membatasi server dengan tiga cara:

1. **Berdasarkan nama server** (`serverName`): Cocok dengan nama server yang dikonfigurasi
2. **Berdasarkan perintah** (`serverCommand`): Cocok dengan perintah dan argumen yang tepat yang digunakan untuk memulai server stdio
3. **Berdasarkan pola URL** (`serverUrl`): Cocok dengan URL server jarak jauh dengan dukungan wildcard

Penting: Setiap entri harus memiliki tepat satu dari `serverName` , `serverCommand` , atau `serverUrl` .

Contoh konfigurasi

```
{
  "allowedMcpServers": [
    // Izinkan berdasarkan nama server
    { "serverName": "github" },
    { "serverName": "sentry" },

    // Izinkan berdasarkan perintah yang tepat (untuk server stdio)
    { "serverCommand": ["npx", "-y", "@modelcontextprotocol/server-filesystem"] },
    { "serverCommand": ["python", "/usr/local/bin/approved-server.py"] },

    // Izinkan berdasarkan pola URL (untuk server jarak jauh)
    { "serverUrl": "https://mcp.company.com/*" },
    { "serverUrl": "https://*.internal.corp/*" }
  ],
  "deniedMcpServers": [
    // Blokir berdasarkan nama server
    { "serverName": "dangerous-server" },

    // Blokir berdasarkan perintah yang tepat (untuk server stdio)
    { "serverCommand": ["npx", "-y", "unapproved-package"] },

    // Blokir berdasarkan pola URL (untuk server jarak jauh)
    { "serverUrl": "https://*.untrusted.com/*" }
  ]
}
```

Cara kerja pembatasan berbasis perintah

Pencocokan yang tepat:

- Array perintah harus cocok **persis** - baik perintah maupun semua argumen dalam urutan yang benar
- Contoh: `["npx", "-y", "server"]` akan TIDAK cocok dengan `["npx", "server"]` atau `["npx", "-y", "server", "--flag"]`

Perilaku server stdio:

- Ketika daftar putih berisi **entri** `serverCommand` apa pun, server stdio **harus** cocok dengan salah satu perintah tersebut

- Server stdio tidak dapat lulus berdasarkan nama saja ketika pembatasan perintah ada
- Ini memastikan administrator dapat memberlakukan perintah mana yang diizinkan untuk dijalankan

Perilaku server non-stdio:

- Server jarak jauh (HTTP, SSE, WebSocket) menggunakan pencocokan berbasis URL ketika entri `serverUrl` ada dalam daftar putih
- Jika tidak ada entri URL, server jarak jauh kembali ke pencocokan berbasis nama
- Pembatasan perintah tidak berlaku untuk server jarak jauh

Cara kerja pembatasan berbasis URL

Pola URL mendukung wildcard menggunakan `*` untuk mencocokkan urutan karakter apa pun. Ini berguna untuk mengizinkan seluruh domain atau subdomain.

Contoh wildcard:

- `https://mcp.company.com/*` - Izinkan semua jalur di domain tertentu
- `https://*.example.com/*` - Izinkan subdomain apa pun dari example.com
- `http://localhost:*/*` - Izinkan port apa pun di localhost

Perilaku server jarak jauh:

- Ketika daftar putih berisi **entri** `serverUrl` apa pun, server jarak jauh **harus** cocok dengan salah satu pola URL tersebut
- Server jarak jauh tidak dapat lulus berdasarkan nama saja ketika pembatasan URL ada
- Ini memastikan administrator dapat memberlakukan endpoint jarak jauh mana yang diizinkan

```
{
  "allowedMcpServers": [
    { "serverUrl": "https://mcp.company.com/*" },
    { "serverUrl": "https://*.internal.corp/*" }
  ]
}
```

Hasil:

- Server HTTP di `https://mcp.company.com/api` : ✅ Diizinkan (cocok dengan pola URL)

- Server HTTP di `https://api.internal.corp/mcp`: ✅ Diizinkan (cocok dengan wildcard subdomain)
- Server HTTP di `https://external.com/mcp`: ❌ Diblokir (tidak cocok dengan pola URL apa pun)
- Server stdio dengan perintah apa pun: ❌ Diblokir (tidak ada entri nama atau perintah untuk dicocokkan)

```
{
  "allowedMcpServers": [
    { "serverCommand": ["npx", "-y", "approved-package"] }
  ]
}
```

Hasil:

- Server stdio dengan `["npx", "-y", "approved-package"]`: ✅ Diizinkan (cocok dengan perintah)
- Server stdio dengan `["node", "server.js"]`: ❌ Diblokir (tidak cocok dengan perintah)
- Server HTTP bernama “my-api”: ❌ Diblokir (tidak ada entri nama untuk dicocokkan)

```
{
  "allowedMcpServers": [
    { "serverName": "github" },
    { "serverCommand": ["npx", "-y", "approved-package"] }
  ]
}
```

Hasil:

- Server stdio bernama “local-tool” dengan `["npx", "-y", "approved-package"]`: ✅ Diizinkan (cocok dengan perintah)
- Server stdio bernama “local-tool” dengan `["node", "server.js"]`: ❌ Diblokir (entri perintah ada tetapi tidak cocok)
- Server stdio bernama “github” dengan `["node", "server.js"]`: ❌ Diblokir (server stdio harus cocok dengan perintah ketika entri perintah ada)
- Server HTTP bernama “github”: ✅ Diizinkan (cocok dengan nama)
- Server HTTP bernama “other-api”: ❌ Diblokir (nama tidak cocok)

```
{
  "allowedMcpServers": [
    { "serverName": "github" },
    { "serverName": "internal-tool" }
  ]
}
```

Hasil:

- Server stdio bernama “github” dengan perintah apa pun: ✅ Diizinkan (tidak ada pembatasan perintah)
- Server stdio bernama “internal-tool” dengan perintah apa pun: ✅ Diizinkan (tidak ada pembatasan perintah)
- Server HTTP bernama “github”: ✅ Diizinkan (cocok dengan nama)
- Server apa pun bernama “other”: ❌ Diblokir (nama tidak cocok)

Perilaku daftar putih (`allowedMcpServers`)

- `undefined` (default): Tidak ada pembatasan - pengguna dapat mengonfigurasi server MCP apa pun
- Array kosong `[]` : Penguncian lengkap - pengguna tidak dapat mengonfigurasi server MCP apa pun
- Daftar entri: Pengguna hanya dapat mengonfigurasi server yang cocok berdasarkan nama, perintah, atau pola URL

Perilaku daftar hitam (`deniedMcpServers`)

- `undefined` (default): Tidak ada server yang diblokir
- Array kosong `[]` : Tidak ada server yang diblokir
- Daftar entri: Server yang ditentukan secara eksplisit diblokir di semua cakupan

Catatan penting

- **Opsi 1 dan Opsi 2 dapat digabungkan**: Jika `managed-mcp.json` ada, file tersebut memiliki kontrol eksklusif dan pengguna tidak dapat menambahkan server. Daftar putih/daftar hitam masih berlaku untuk server yang dikelola itu sendiri.
- **Daftar hitam mengambil prioritas absolut**: Jika server cocok dengan entri daftar hitam (berdasarkan nama, perintah, atau URL), server akan diblokir bahkan jika ada di daftar putih

- Pembatasan berbasis nama, berbasis perintah, dan berbasis URL bekerja bersama: server lulus jika cocok dengan **entri** nama, entri perintah, atau pola URL (kecuali diblokir oleh daftar hitam)

> **Note:**
>
> **Saat menggunakan** `managed-mcp.json` : Pengguna tidak dapat menambahkan server MCP melalui `claude mcp add` atau file konfigurasi. Pengaturan `allowedMcpServers` dan `deniedMcpServers` masih berlaku untuk memfilter server yang dikelola mana yang benar-benar dimuat.

Perluas Claude dengan skills

> *Buat, kelola, dan bagikan skills untuk memperluas kemampuan Claude di Claude Code. Termasuk perintah kustom dan skills bundel.*

Skills memperluas apa yang dapat dilakukan Claude. Buat file `SKILL.md` dengan instruksi, dan Claude menambahkannya ke toolkit-nya. Claude menggunakan skills ketika relevan, atau Anda dapat menginvokasinya secara langsung dengan `/skill-name`.

> **Note:**
>
> Untuk perintah bawaan seperti `/help` dan `/compact`, lihat mode interaktif.
>
> **Perintah kustom telah digabungkan ke dalam skills.** File di `.claude/commands/deploy.md` dan skill di `.claude/skills/deploy/SKILL.md` keduanya membuat `/deploy` dan bekerja dengan cara yang sama. File `.claude/commands/` yang ada tetap berfungsi. Skills menambahkan fitur opsional: direktori untuk file pendukung, frontmatter untuk mengontrol apakah Anda atau Claude menginvokasinya, dan kemampuan bagi Claude untuk memuatnya secara otomatis ketika relevan.

Skills Claude Code mengikuti standar terbuka Agent Skills, yang bekerja di berbagai alat AI. Claude Code memperluas standar dengan fitur tambahan seperti kontrol invokasi, eksekusi subagent, dan injeksi konteks dinamis.

Skills bundel

Skills bundel dikirim dengan Claude Code dan tersedia di setiap sesi. Tidak seperti perintah bawaan, yang menjalankan logika tetap secara langsung, skills bundel berbasis prompt: mereka memberikan Claude playbook terperinci dan membiarkannya mengorkestrasi pekerjaan menggunakan tools-nya. Ini berarti skills bundel dapat menelurkan agen paralel, membaca file, dan beradaptasi dengan codebase Anda.

Anda menginvokasinya skills bundel dengan cara yang sama seperti skill lainnya: ketik `/` diikuti dengan nama skill.

- `/simplify`: meninjau file yang baru-baru ini diubah untuk masalah penggunaan kembali kode, kualitas, dan efisiensi, kemudian memperbaikinya. Jalankan setelah mengimplementasikan fitur atau perbaikan bug untuk membersihkan pekerjaan Anda. Ini menelurkan tiga agen review secara paralel (penggunaan kembali kode, kualitas kode, efisiensi), mengagregasi temuan mereka, dan menerapkan perbaikan. Lewatkan teks

opsional untuk fokus pada kekhawatiran tertentu: `/simplify focus on memory efficiency`.

- `/batch <instruction>`: mengorkestrasi perubahan skala besar di seluruh codebase secara paralel. Berikan deskripsi perubahan dan `/batch` meneliti codebase, menguraikan pekerjaan menjadi 5 hingga 30 unit independen, dan menyajikan rencana untuk persetujuan Anda. Setelah disetujui, ia menelurkan satu agen latar belakang per unit, masing-masing dalam git worktree yang terisolasi. Setiap agen mengimplementasikan unitnya, menjalankan tes, dan membuka pull request. Memerlukan repositori git. Contoh: `/batch migrate src/ from Solid to React`.
- `/debug [description]`: memecahkan masalah sesi Claude Code Anda saat ini dengan membaca log debug sesi. Secara opsional jelaskan masalahnya untuk fokus analisis.
- `/loop [interval] <prompt>`: menjalankan prompt berulang kali pada interval sambil sesi tetap terbuka. Claude menguraikan interval, menjadwalkan tugas cron berulang, dan mengonfirmasi ritme. Berguna untuk polling deployment, mengasuh PR, atau menjalankan kembali skill lain secara berkala. Contoh: `/loop 5m check if the deploy finished`. Lihat Jalankan prompt sesuai jadwal.
- `/claude-api`: memuat materi referensi Claude API untuk bahasa proyek Anda (Python, TypeScript, Java, Go, Ruby, C#, PHP, atau cURL) dan referensi Agent SDK untuk Python dan TypeScript. Mencakup tool use, streaming, batches, structured outputs, dan pitfalls umum. Juga diaktifkan secara otomatis ketika kode Anda mengimpor `anthropic`, `@anthropic-ai/sdk`, atau `claude_agent_sdk`.

Memulai

Buat skill pertama Anda

Contoh ini membuat skill yang mengajarkan Claude menjelaskan kode menggunakan diagram visual dan analogi. Karena menggunakan frontmatter default, Claude dapat memuatnya secara otomatis ketika Anda bertanya bagaimana sesuatu bekerja, atau Anda dapat menginvokasinya secara langsung dengan `/explain-code`.

Step 1: Buat direktori skill

Buat direktori untuk skill di folder skills pribadi Anda. Skills pribadi tersedia di semua proyek Anda.

```
mkdir -p ~/.claude/skills/explain-code
```

Step 2: Tulis SKILL.md

Setiap skill memerlukan file `SKILL.md` dengan dua bagian: frontmatter YAML (antara penanda `---`) yang memberi tahu Claude kapan menggunakan skill, dan konten markdown dengan instruksi yang diikuti Claude ketika skill diinvokasinya. Bidang `name` menjadi `/slash-command`, dan `description` membantu Claude memutuskan kapan memuatnya secara otomatis.

Buat `~/.claude/skills/explain-code/SKILL.md`:

```
---
name: explain-code
description: Menjelaskan kode dengan diagram visual dan analogi. Gunakan saat menjelaskan cara kerja kode, mengajar tentang codebase, atau ketika pengguna bertanya "bagaimana cara kerjanya?"
---

Saat menjelaskan kode, selalu sertakan:

1. **Mulai dengan analogi**: Bandingkan kode dengan sesuatu dari kehidupan sehari-hari
2. **Gambar diagram**: Gunakan ASCII art untuk menunjukkan alur, struktur, atau hubungan
3. **Jelajahi kode**: Jelaskan langkah demi langkah apa yang terjadi
4. **Sorot gotcha**: Apa kesalahan umum atau kesalahpahaman?

Jaga penjelasan tetap percakapan. Untuk konsep kompleks, gunakan beberapa analogi.
```

Step 3: Uji skill

Anda dapat mengujinya dengan dua cara:

Biarkan Claude menginvokasinya secara otomatis dengan bertanya sesuatu yang cocok dengan deskripsi:

```
Bagaimana cara kerja kode ini?
```

Atau invokasinya secara langsung dengan nama skill:

```
/explain-code src/auth/login.ts
```

Baik cara mana pun, Claude harus menyertakan analogi dan diagram ASCII dalam penjelasannya.

Tempat skills berada

Tempat Anda menyimpan skill menentukan siapa yang dapat menggunakannya:

Lokasi	Path	Berlaku untuk
Enterprise	Lihat pengaturan terkelola	Semua pengguna di organisasi Anda
Pribadi	`~/.claude/skills/<skill-name>/SKILL.md`	Semua proyek Anda
Proyek	`.claude/skills/<skill-name>/SKILL.md`	Proyek ini saja
Plugin	`<plugin>/skills/<skill-name>/SKILL.md`	Tempat plugin diaktifkan

Ketika skills berbagi nama yang sama di berbagai level, lokasi prioritas lebih tinggi menang: enterprise > pribadi > proyek. Skills plugin menggunakan namespace `plugin-name:skill-name`, jadi mereka tidak dapat bertentangan dengan level lain. Jika Anda memiliki file di `.claude/commands/`, file tersebut bekerja dengan cara yang sama, tetapi jika skill dan perintah berbagi nama yang sama, skill mengambil alih.

Penemuan otomatis dari direktori bersarang

Ketika Anda bekerja dengan file di subdirektori, Claude Code secara otomatis menemukan skills dari direktori `.claude/skills/` bersarang. Misalnya, jika Anda mengedit file di `packages/frontend/`, Claude Code juga mencari skills di `packages/frontend/.claude/skills/`. Ini mendukung pengaturan monorepo di mana paket memiliki skills mereka sendiri.

Setiap skill adalah direktori dengan `SKILL.md` sebagai titik masuk:

```
my-skill/
├── SKILL.md           # Instruksi utama (diperlukan)
├── template.md        # Template untuk Claude isi
├── examples/
│   └── sample.md      # Contoh output menunjukkan format yang diharapkan
└── scripts/
    └── validate.sh    # Script Claude dapat jalankan
```

`SKILL.md` berisi instruksi utama dan diperlukan. File lainnya opsional dan memungkinkan Anda membangun skills yang lebih kuat: template untuk Claude isi, contoh output menunjukkan format yang diharapkan, script Claude dapat jalankan, atau dokumentasi referensi terperinci. Referensikan file pendukung ini dari `SKILL.md` Anda sehingga Claude tahu apa yang mereka berisi dan kapan memuatnya. Lihat Tambahkan file pendukung untuk detail lebih lanjut.

> **Note:**
>
> File di `.claude/commands/` masih bekerja dan mendukung frontmatter yang sama. Skills direkomendasikan karena mendukung fitur tambahan seperti file pendukung.

Skills dari direktori tambahan

Skills yang didefinisikan di `.claude/skills/` dalam direktori yang ditambahkan melalui `--add-dir` dimuat secara otomatis dan diambil oleh deteksi perubahan langsung, jadi Anda dapat mengeditnya selama sesi tanpa memulai ulang.

> **Note:**
>
> File CLAUDE.md dari direktori `--add-dir` tidak dimuat secara default. Untuk memuatnya, atur `CLAUDE_CODE_ADDITIONAL_DIRECTORIES_CLAUDE_MD=1` . Lihat Muat dari direktori tambahan.

Konfigurasi skills

Skills dikonfigurasi melalui frontmatter YAML di bagian atas `SKILL.md` dan konten markdown yang mengikutinya.

Jenis konten skill

File skill dapat berisi instruksi apa pun, tetapi memikirkan tentang cara Anda ingin menginvokasinya membantu memandu apa yang harus disertakan:

Konten referensi menambahkan pengetahuan yang diterapkan Claude pada pekerjaan Anda saat ini. Konvensi, pola, panduan gaya, pengetahuan domain. Konten ini berjalan inline sehingga Claude dapat menggunakannya bersama konteks percakapan Anda.

```
---
name: api-conventions
description: Pola desain API untuk codebase ini
---

Saat menulis endpoint API:
- Gunakan konvensi penamaan RESTful
- Kembalikan format kesalahan yang konsisten
- Sertakan validasi permintaan
```

Konten tugas memberikan Claude instruksi langkah demi langkah untuk tindakan tertentu, seperti deployment, commit, atau pembuatan kode. Ini sering kali tindakan yang ingin Anda invokasinya secara langsung dengan `/skill-name` daripada membiarkan Claude memutuskan kapan menjalankannya. Tambahkan `disable-model-invocation: true` untuk mencegah Claude memicunya secara otomatis.

```
---
name: deploy
description: Deployment aplikasi ke produksi
context: fork
disable-model-invocation: true
---

Deployment aplikasi:
1. Jalankan test suite
2. Build aplikasi
3. Push ke target deployment
```

`SKILL.md` Anda dapat berisi apa pun, tetapi memikirkan tentang cara Anda ingin skill di-invokasinya (oleh Anda, oleh Claude, atau keduanya) dan di mana Anda ingin menjalankannya (inline atau di subagent) membantu memandu apa yang harus disertakan. Untuk skills kompleks, Anda juga dapat menambahkan file pendukung untuk menjaga skill utama tetap fokus.

Referensi frontmatter

Selain konten markdown, Anda dapat mengonfigurasi perilaku skill menggunakan bidang frontmatter YAML antara penanda `---` di bagian atas file `SKILL.md` Anda:

```
---
name: my-skill
description: Apa yang dilakukan skill ini
disable-model-invocation: true
allowed-tools: Read, Grep
---

Instruksi skill Anda di sini...
```

Semua bidang opsional. Hanya `description` yang direkomendasikan sehingga Claude tahu kapan menggunakan skill.

Bidang	Diperlukan	Deskripsi
`name`	Tidak	Nama tampilan untuk skill. Jika dihilangkan, menggunakan nama direktori. Huruf kecil, angka, dan tanda hubung saja (maks 64 karakter).
`description`	Direkomendasikan	Apa yang dilakukan skill dan kapan menggunakannya. Claude menggunakan ini untuk memutuskan kapan menerapkan skill. Jika dihilangkan, menggunakan paragraf pertama konten markdown.

Bidang	Diperlukan	Deskripsi
`argument-hint`	Tidak	Petunjuk ditampilkan selama autocomplete untuk menunjukkan argumen yang diharapkan. Contoh: `[issue-number]` atau `[filename] [format]`.
`disable-model-invocation`	Tidak	Atur ke `true` untuk mencegah Claude memuatnya secara otomatis. Gunakan untuk alur kerja yang ingin Anda picu secara manual dengan `/name`. Default: `false`.
`user-invocable`	Tidak	Atur ke `false` untuk menyembunyikan dari menu `/`. Gunakan untuk pengetahuan latar belakang yang tidak boleh diinvokasinya pengguna secara langsung. Default: `true`.
`allowed-tools`	Tidak	Tools yang dapat digunakan Claude tanpa meminta izin ketika skill ini aktif.
`model`	Tidak	Model yang digunakan ketika skill ini aktif.
`context`	Tidak	Atur ke `fork` untuk menjalankan dalam konteks subagent yang di-fork.
`agent`	Tidak	Jenis subagent mana yang digunakan ketika `context: fork` diatur.
`hooks`	Tidak	Hooks yang dibatasi pada siklus hidup skill ini. Lihat Hooks dalam skills dan agents untuk format konfigurasi.

Substitusi string yang tersedia

Skills mendukung substitusi string untuk nilai dinamis dalam konten skill:

Variabel	Deskripsi
`$ARGUMENTS`	Semua argumen yang dilewatkan saat menginvokasinya skill. Jika `$ARGUMENTS` tidak ada dalam konten, argumen ditambahkan sebagai `ARGUMENTS: <value>`.
`$ARGUMENTS[N]`	Akses argumen tertentu berdasarkan indeks berbasis 0, seperti `$ARGUMENTS[0]` untuk argumen pertama.
`$N`	Singkat untuk `$ARGUMENTS[N]`, seperti `$0` untuk argumen pertama atau `$1` untuk argumen kedua.
`${CLAUDE_SESSION_ID}`	ID sesi saat ini. Berguna untuk logging, membuat file khusus sesi, atau menghubungkan output skill dengan sesi.
`${CLAUDE_SKILL_DIR}`	Direktori yang berisi file `SKILL.md` skill. Untuk skills plugin, ini adalah subdirektori skill dalam plugin, bukan root plugin. Gunakan ini dalam perintah injeksi bash untuk mereferensikan script atau file yang disertakan dengan skill, terlepas dari direktori kerja saat ini.

Contoh menggunakan substitusi:

```
---
name: session-logger
description: Log aktivitas untuk sesi ini
---

Log berikut ke logs/${CLAUDE_SESSION_ID}.log:

$ARGUMENTS
```

Tambahkan file pendukung

Skills dapat menyertakan beberapa file di direktorinya. Ini menjaga `SKILL.md` fokus pada hal-hal penting sambil membiarkan Claude mengakses materi referensi terperinci hanya saat diperlukan. Dokumen referensi besar, spesifikasi API, atau koleksi contoh tidak perlu dimuat ke dalam konteks setiap kali skill berjalan.

```
my-skill/
├── SKILL.md (diperlukan - gambaran umum dan navigasi)
├── reference.md (dokumen API terperinci - dimuat saat diperlukan)
├── examples.md (contoh penggunaan - dimuat saat diperlukan)
└── scripts/
    └── helper.py (script utilitas - dijalankan, tidak dimuat)
```

Referensikan file pendukung dari `SKILL.md` Anda sehingga Claude tahu apa yang berisi setiap file dan kapan memuatnya:

```
### Sumber daya tambahan

- Untuk detail API lengkap, lihat [reference.md](reference.md)
- Untuk contoh penggunaan, lihat [examples.md](examples.md)
```

> **Tip:**
>
> Jaga `SKILL.md` di bawah 500 baris. Pindahkan materi referensi terperinci ke file terpisah.

Kontrol siapa yang menginvokasinya skill

Secara default, baik Anda maupun Claude dapat menginvokasinya skill apa pun. Anda dapat mengetik `/skill-name` untuk menginvokasinya secara langsung, dan Claude dapat memuatnya secara otomatis ketika relevan dengan percakapan Anda. Dua bidang frontmatter memungkinkan Anda membatasi ini:

- `disable-model-invocation: true` : Hanya Anda yang dapat menginvokasinya skill. Gunakan ini untuk alur kerja dengan efek samping atau yang ingin Anda kontrol waktunya, seperti `/commit` , `/deploy` , atau `/send-slack-message` . Anda tidak ingin Claude memutuskan untuk deploy karena kode Anda terlihat siap.
- `user-invocable: false` : Hanya Claude yang dapat menginvokasinya skill. Gunakan ini untuk pengetahuan latar belakang yang tidak dapat ditindaklanjuti sebagai perintah. Skill `legacy-system-context` menjelaskan cara kerja sistem lama. Claude harus tahu ini ketika relevan, tetapi `/legacy-system-context` bukan tindakan bermakna bagi pengguna untuk diambil.

Contoh ini membuat skill deploy yang hanya dapat Anda picu. Bidang `disable-model-invocation: true` mencegah Claude menjalankannya secara otomatis:

```
---
name: deploy
description: Deployment aplikasi ke produksi
disable-model-invocation: true
---

Deployment $ARGUMENTS ke produksi:

1. Jalankan test suite
2. Build aplikasi
3. Push ke target deployment
4. Verifikasi deployment berhasil
```

Berikut adalah cara dua bidang mempengaruhi invokasi dan pemuatan konteks:

Front-matter	Anda dapat menginvoka-sinya	Claude dapat menginvokasi-nya	Kapan dimuat ke dalam konteks
(default)	Ya	Ya	Deskripsi selalu dalam konteks, skill penuh dimuat saat diinvokasinya
`disable-model-invocation: true`	Ya	Tidak	Deskripsi tidak dalam konteks, skill penuh dimuat saat Anda menginvokasinya
`user-invocable: false`	Tidak	Ya	Deskripsi selalu dalam konteks, skill penuh dimuat saat diinvokasinya

> **Note:**
>
> Dalam sesi reguler, deskripsi skill dimuat ke dalam konteks sehingga Claude tahu apa yang tersedia, tetapi konten skill penuh hanya dimuat saat diinvokasinya. Subagents dengan skills yang dimuat sebelumnya bekerja berbeda: konten skill penuh disuntikkan saat startup.

Batasi akses tool

Gunakan bidang `allowed-tools` untuk membatasi tools mana yang dapat digunakan Claude ketika skill aktif. Skill ini membuat mode baca-saja di mana Claude dapat menjelajahi file tetapi tidak memodifikasinya:

```
---
name: safe-reader
description: Baca file tanpa membuat perubahan
allowed-tools: Read, Grep, Glob
---
```

Lewatkan argumen ke skills

Baik Anda maupun Claude dapat melewatkan argumen saat menginvokasinya skill. Argumen tersedia melalui placeholder `$ARGUMENTS`.

Skill ini memperbaiki masalah GitHub berdasarkan nomor. Placeholder `$ARGUMENTS` diganti dengan apa pun yang mengikuti nama skill:

```
---
name: fix-issue
description: Perbaiki masalah GitHub
disable-model-invocation: true
---

Perbaiki masalah GitHub $ARGUMENTS mengikuti standar coding kami.

1. Baca deskripsi masalah
2. Pahami persyaratan
3. Implementasikan perbaikan
4. Tulis tes
5. Buat commit
```

Ketika Anda menjalankan `/fix-issue 123`, Claude menerima "Perbaiki masalah GitHub 123 mengikuti standar coding kami..."

Jika Anda menginvokasinya skill dengan argumen tetapi skill tidak menyertakan `$ARGUMENTS`, Claude Code menambahkan `ARGUMENTS: <your input>` ke akhir konten skill sehingga Claude masih melihat apa yang Anda ketik.

Untuk mengakses argumen individual berdasarkan posisi, gunakan `$ARGUMENTS[N]` atau yang lebih pendek `$N`:

```
---
name: migrate-component
description: Migrasikan komponen dari satu framework ke framework lain
---

Migrasikan komponen $ARGUMENTS[0] dari $ARGUMENTS[1] ke $ARGUMENTS[2].
Pertahankan semua perilaku dan tes yang ada.
```

Menjalankan `/migrate-component SearchBar React Vue` mengganti `$ARGUMENTS[0]` dengan `SearchBar`, `$ARGUMENTS[1]` dengan `React`, dan `$ARGUMENTS[2]` dengan `Vue`. Skill yang sama menggunakan singkat `$N`:

```
---
name: migrate-component
description: Migrasikan komponen dari satu framework ke framework lain
---

Migrasikan komponen $0 dari $1 ke $2.
Pertahankan semua perilaku dan tes yang ada.
```

Pola lanjutan

Injeksi konteks dinamis

Sintaks `!` command`` menjalankan perintah shell sebelum konten skill dikirim ke Claude. Output perintah mengganti placeholder, jadi Claude menerima data aktual, bukan perintah itu sendiri.

Skill ini merangkum pull request dengan mengambil data PR langsung dengan GitHub CLI. Perintah `!` gh pr diff`` dan lainnya berjalan terlebih dahulu, dan output mereka dimasukkan ke dalam prompt:

```
---
name: pr-summary
description: Rangkum perubahan dalam pull request
context: fork
agent: Explore
allowed-tools: Bash(gh *)
---

### Konteks pull request
- PR diff: !`gh pr diff`
- Komentar PR: !`gh pr view --comments`
- File yang diubah: !`gh pr diff --name-only`

### Tugas Anda
Rangkum pull request ini...
```

Ketika skill ini berjalan:

1. Setiap `!` command`` dijalankan segera (sebelum Claude melihat apa pun)
2. Output mengganti placeholder dalam konten skill
3. Claude menerima prompt yang sepenuhnya dirender dengan data PR aktual

Ini adalah preprocessing, bukan sesuatu yang dijalankan Claude. Claude hanya melihat hasil akhir.

> **Tip:**
>
> Untuk mengaktifkan extended thinking dalam skill, sertakan kata "ultrathink" di mana pun dalam konten skill Anda.

Jalankan skills dalam subagent

Tambahkan `context: fork` ke frontmatter Anda ketika Anda ingin skill berjalan dalam isolasi. Konten skill menjadi prompt yang mendorong subagent. Ini tidak akan memiliki akses ke riwayat percakapan Anda.

> **Warning:**

> `context: fork` hanya masuk akal untuk skills dengan instruksi eksplisit. Jika skill Anda berisi pedoman seperti "gunakan konvensi API ini" tanpa tugas, subagent menerima pedoman tetapi tidak ada prompt yang dapat ditindaklanjuti, dan kembali tanpa output bermakna.

Skills dan subagents bekerja bersama dalam dua arah:

Pende-katan	System prompt	Tugas	Juga memuat
Skill deng-an `context: fork`	Dari jenis agent (`Explore` , `Plan` , dll)	Konten SKILL.md	CLAUDE.md
Subagent dengan bi-dang `skills`	Badan markdown subagent	Pesan delegasi Claude	Skills yang dimuat se-belumnya + CLAU-DE.md

Dengan `context: fork` , Anda menulis tugas dalam skill Anda dan memilih jenis agent untuk menjalankannya. Untuk kebalikannya (mendefinisikan subagent kustom yang menggunakan skills sebagai materi referensi), lihat Subagents.

Contoh: Skill penelitian menggunakan agen Explore

Skill ini menjalankan penelitian dalam agen Explore yang di-fork. Konten skill menjadi tugas, dan agen menyediakan tools baca-saja yang dioptimalkan untuk eksplorasi code-base:

```
---
name: deep-research
description: Teliti topik secara menyeluruh
context: fork
agent: Explore
---

Teliti $ARGUMENTS secara menyeluruh:

1. Temukan file relevan menggunakan Glob dan Grep
2. Baca dan analisis kode
3. Rangkum temuan dengan referensi file tertentu
```

Ketika skill ini berjalan:

1. Konteks terisolasi baru dibuat
2. Subagent menerima konten skill sebagai promptnya ("Teliti $ARGUMENTS secara menyeluruh...")
3. Bidang `agent` menentukan lingkungan eksekusi (model, tools, dan izin)
4. Hasil dirangkum dan dikembalikan ke percakapan utama Anda

Bidang `agent` menentukan konfigurasi subagent mana yang digunakan. Opsi termasuk agen bawaan (`Explore` , `Plan` , `general-purpose`) atau subagent kustom apa pun dari `.claude/agents/` . Jika dihilangkan, menggunakan `general-purpose` .

Batasi akses skill Claude

Secara default, Claude dapat menginvokasinya skill apa pun yang tidak memiliki `disable-model-invocation: true` diatur. Skills yang mendefinisikan `allowed-tools` memberikan Claude akses ke tools tersebut tanpa persetujuan per-penggunaan ketika skill aktif. Pengaturan izin Anda masih mengatur perilaku persetujuan baseline untuk semua tools lainnya. Perintah bawaan seperti `/compact` dan `/init` tidak tersedia melalui tool Skill.

Tiga cara untuk mengontrol skills mana yang dapat diinvokasinya Claude:

Nonaktifkan semua skills dengan menolak tool Skill dalam `/permissions` :

```
## Tambahkan ke aturan penolakan:
Skill
```

Izinkan atau tolak skills tertentu menggunakan aturan izin:

```
## Izinkan hanya skills tertentu
Skill(commit)
Skill(review-pr *)

## Tolak skills tertentu
Skill(deploy *)
```

Sintaks izin: `Skill(name)` untuk kecocokan tepat, `Skill(name *)` untuk kecocokan awalan dengan argumen apa pun.

Sembunyikan skills individual dengan menambahkan `disable-model-invocation: true` ke frontmatter mereka. Ini menghapus skill dari konteks Claude sepenuhnya.

> **Note:**
>
> Bidang `user-invocable` hanya mengontrol visibilitas menu, bukan akses tool Skill. Gunakan `disable-model-invocation: true` untuk memblokir invokasi pemrograman.

Bagikan skills

Skills dapat didistribusikan pada cakupan berbeda tergantung pada audiens Anda:

- **Skills proyek**: Commit `.claude/skills/` ke kontrol versi
- **Plugins**: Buat direktori `skills/` dalam plugin Anda
- **Terkelola**: Deployment di seluruh organisasi melalui pengaturan terkelola

Hasilkan output visual

Skills dapat membundel dan menjalankan script dalam bahasa apa pun, memberikan Claude kemampuan di luar apa yang mungkin dalam prompt tunggal. Satu pola yang kuat adalah menghasilkan output visual: file HTML interaktif yang terbuka di browser Anda untuk menjelajahi data, debugging, atau membuat laporan.

Contoh ini membuat penjelajah codebase: tampilan pohon interaktif di mana Anda dapat memperluas dan menciutkan direktori, melihat ukuran file sekilas, dan mengidentifikasi jenis file berdasarkan warna.

Buat direktori Skill:

```
mkdir -p ~/.claude/skills/codebase-visualizer/scripts
```

Buat `~/.claude/skills/codebase-visualizer/SKILL.md`. Deskripsi memberi tahu Claude kapan mengaktifkan Skill ini, dan instruksi memberi tahu Claude untuk menjalankan script bundel:

```
---
name: codebase-visualizer
description: Hasilkan visualisasi pohon interaktif yang dapat diciutkan dari
codebase Anda. Gunakan saat menjelajahi repo baru, memahami struktur proyek, atau
mengidentifikasi file besar.
allowed-tools: Bash(python *)
---

## Codebase Visualizer

Hasilkan tampilan pohon HTML interaktif yang menunjukkan struktur file proyek Anda
dengan direktori yang dapat diciutkan.

### Penggunaan

Jalankan script visualisasi dari root proyek Anda:

```bash
python ~/.claude/skills/codebase-visualizer/scripts/visualize.py .
```text

Ini membuat `codebase-map.html` di direktori saat ini dan membukanya di browser
default Anda.

### Apa yang ditampilkan visualisasi

- **Direktori yang dapat diciutkan**: Klik folder untuk memperluas/menciutkan
- **Ukuran file**: Ditampilkan di sebelah setiap file
- **Warna**: Warna berbeda untuk jenis file berbeda
- **Total direktori**: Menunjukkan ukuran agregat setiap folder
```

Buat `~/.claude/skills/codebase-visualizer/scripts/visualize.py`. Script ini memindai pohon direktori dan menghasilkan file HTML mandiri dengan:

- **Sidebar ringkasan** menampilkan jumlah file, jumlah direktori, ukuran total, dan jumlah jenis file
- **Bagan batang** memecah codebase berdasarkan jenis file (8 teratas berdasarkan ukuran)
- **Pohon yang dapat diciutkan** di mana Anda dapat memperluas dan menciutkan direktori, dengan indikator jenis file berkode warna

Script memerlukan Python tetapi hanya menggunakan library bawaan, jadi tidak ada paket untuk diinstal:

```
#!/usr/bin/env python3
"""Hasilkan visualisasi pohon interaktif yang dapat diciutkan dari codebase."""

from pathlib import Path
from collections import Counter

IGNORE = {'.git', 'node_modules', '__pycache__', '.venv', 'venv', 'dist', 'build'}

def scan(path: Path, stats: dict) -> dict:
    result = {"name": path.name, "children": [], "size": 0}
    try:
        for item in sorted(path.iterdir()):
            if item.name in IGNORE or item.name.startswith('.'):
                continue
            if item.is_file():
                size = item.stat().st_size
                ext = item.suffix.lower() or '(no ext)'
                result["children"].append({"name": item.name, "size": size,
"ext": ext})
                result["size"] += size
                stats["files"] += 1
                stats["extensions"][ext] += 1
                stats["ext_sizes"][ext] += size
            elif item.is_dir():
                stats["dirs"] += 1
                child = scan(item, stats)
                if child["children"]:
                    result["children"].append(child)
                    result["size"] += child["size"]
    except PermissionError:
        pass
    return result

def generate_html(data: dict, stats: dict, output: Path) -> None:
    ext_sizes = stats["ext_sizes"]
    total_size = sum(ext_sizes.values()) or 1
    sorted_exts = sorted(ext_sizes.items(), key=lambda x: -x[1])[:8]
    colors = {
```

```
        '.js': '#f7df1e', '.ts': '#3178c6', '.py': '#3776ab', '.go': '#00add8',
        '.rs': '#dea584', '.rb': '#cc342d', '.css': '#264de4', '.html': '#e34c26',
        '.json': '#6b7280', '.md': '#083fa1', '.yaml': '#cb171e', '.yml': '#cb171e
',
        '.mdx': '#083fa1', '.tsx': '#3178c6', '.jsx': '#61dafb', '.sh': '#4eaa25',
    }
    lang_bars = "".join(
        f'<div class="bar-row"><span class="bar-label">{ext}</span>'
        f'<div class="bar" style="width:{(size/total_size)*100}%;background:{color
s.get(ext,"#6b7280")}"></div>'
        f'<span class="bar-pct">{(size/total_size)*100:.1f}%</span></div>'
        for ext, size in sorted_exts
    )
    def fmt(b):
        if b < 1024: return f"{b} B"
        if b < 1048576: return f"{b/1024:.1f} KB"
        return f"{b/1048576:.1f} MB"

    html = f'''<!DOCTYPE html>
<html><head>
  <meta charset="utf-8"><title>Codebase Explorer</title>
  <style>
    body {{ font: 14px/1.5 system-ui, sans-serif; margin: 0; background: #1a1a2e;
color: #eee; }}
    .container {{ display: flex; height: 100vh; }}
    .sidebar {{ width: 280px; background: #252542; padding: 20px; border-right:
1px solid #3d3d5c; overflow-y: auto; flex-shrink: 0; }}
    .main {{ flex: 1; padding: 20px; overflow-y: auto; }}
    h1 {{ margin: 0 0 10px 0; font-size: 18px; }}
    h2 {{ margin: 20px 0 10px 0; font-size: 14px; color: #888; text-transform:
uppercase; }}
    .stat {{ display: flex; justify-content: space-between; padding: 8px 0;
border-bottom: 1px solid #3d3d5c; }}
    .stat-value {{ font-weight: bold; }}
    .bar-row {{ display: flex; align-items: center; margin: 6px 0; }}
    .bar-label {{ width: 55px; font-size: 12px; color: #aaa; }}
    .bar {{ height: 18px; border-radius: 3px; }}
    .bar-pct {{ margin-left: 8px; font-size: 12px; color: #666; }}
    .tree {{ list-style: none; padding-left: 20px; }}
```

```
    details {{ cursor: pointer; }}
    summary {{ padding: 4px 8px; border-radius: 4px; }}
    summary:hover {{ background: #2d2d44; }}
    .folder {{ color: #ffd700; }}
    .file {{ display: flex; align-items: center; padding: 4px 8px; border-radius:
4px; }}
    .file:hover {{ background: #2d2d44; }}
    .size {{ color: #888; margin-left: auto; font-size: 12px; }}
    .dot {{ width: 8px; height: 8px; border-radius: 50%; margin-right: 8px; }}
  </style>
</head><body>
  <div class="container">
    <div class="sidebar">
      <h1>📊 Summary</h1>
      <div class="stat"><span>Files</span><span class="stat-
value">{stats["files"]:,}</span></div>
      <div class="stat"><span>Directories</span><span class="stat-value">{stats["d
irs"]:,}</span></div>
      <div class="stat"><span>Total size</span><span class="stat-
value">{fmt(data["size"])}</span></div>
      <div class="stat"><span>File types</span><span class="stat-value">{len(stats
["extensions"])}</span></div>
      <h2>By file type</h2>
      {lang_bars}
    </div>
    <div class="main">
      <h1>📁 {data["name"]}</h1>
      <ul class="tree" id="root"></ul>
    </div>
  </div>
  <script>
    const data = {json.dumps(data)};
    const colors = {json.dumps(colors)};
    function fmt(b) {{ if (b < 1024) return b + ' B'; if (b < 1048576) return (b/
1024).toFixed(1) + ' KB'; return (b/1048576).toFixed(1) + ' MB'; }}
    function render(node, parent) {{
      if (node.children) {{
        const det = document.createElement('details');
        det.open = parent === document.getElementById('root');
```

```
        det.innerHTML = `<summary><span class="folder">📁 ${{node.name}}</
span><span class="size">${{fmt(node.size)}}</span></summary>`;
        const ul = document.createElement('ul'); ul.className = 'tree';
        node.children.sort((a,b) => (b.children?1:0)-(a.children?1:0) ||
a.name.localeCompare(b.name));
        node.children.forEach(c => render(c, ul));
        det.appendChild(ul);
        const li = document.createElement('li'); li.appendChild(det);
parent.appendChild(li);
      }} else {{
        const li = document.createElement('li'); li.className = 'file';
        li.innerHTML = `<span class="dot" style="background:${{colors[node.ext]||'
#6b7280'}}"></span>${{node.name}}<span class="size">${{fmt(node.size)}}</span>`;
        parent.appendChild(li);
      }}
    }}
    data.children.forEach(c => render(c, document.getElementById('root')));
  </script>
</body></html>'''
    output.write_text(html)

if __name__ == '__main__':
    target = Path(sys.argv[1] if len(sys.argv) > 1 else '.').resolve()
    stats = {"files": 0, "dirs": 0, "extensions": Counter(), "ext_sizes":
Counter()}
    data = scan(target, stats)
    out = Path('codebase-map.html')
    generate_html(data, stats, out)
    print(f'Generated {out.absolute()}')
    webbrowser.open(f'file://{out.absolute()}')
```

Untuk menguji, buka Claude Code di proyek apa pun dan tanya “Visualisasi codebase ini.” Claude menjalankan script, menghasilkan `codebase-map.html`, dan membukanya di browser Anda.

Pola ini bekerja untuk output visual apa pun: grafik dependensi, laporan cakupan tes, dokumentasi API, atau visualisasi skema database. Script bundel melakukan pekerjaan berat sementara Claude menangani orkestrasi.

Troubleshooting

Skill tidak terpicu

Jika Claude tidak menggunakan skill Anda saat diharapkan:

1. Periksa deskripsi mencakup kata kunci yang akan dikatakan pengguna secara alami
2. Verifikasi skill muncul dalam `What skills are available?`
3. Coba rephrase permintaan Anda agar lebih cocok dengan deskripsi
4. Invokasinya secara langsung dengan `/skill-name` jika skill dapat diinvokasinya pengguna

Skill terpicu terlalu sering

Jika Claude menggunakan skill Anda saat Anda tidak menginginkannya:

1. Buat deskripsi lebih spesifik
2. Tambahkan `disable-model-invocation: true` jika Anda hanya menginginkan invokasi manual

Claude tidak melihat semua skills saya

Deskripsi skill dimuat ke dalam konteks sehingga Claude tahu apa yang tersedia. Jika Anda memiliki banyak skills, mereka mungkin melebihi anggaran karakter. Anggaran diskalakan secara dinamis pada 2% dari jendela konteks, dengan fallback 16.000 karakter. Jalankan `/context` untuk memeriksa peringatan tentang skills yang dikecualikan.

Untuk mengganti batas, atur variabel lingkungan `SLASH_COMMAND_TOOL_CHAR_BUDGET`.

Sumber daya terkait

- **Subagents**: delegasikan tugas ke agen khusus
- **Plugins**: paket dan distribusikan skills dengan ekstensi lainnya
- **Hooks**: otomatisasi alur kerja di sekitar peristiwa tool
- **Memory**: kelola file CLAUDE.md untuk konteks persisten
- **Mode interaktif**: perintah bawaan dan pintasan
- **Permissions**: kontrol akses tool dan skill

Buat plugins

> *Buat plugins kustom untuk memperluas Claude Code dengan skills, agents, hooks, dan MCP servers.*

Plugins memungkinkan Anda memperluas Claude Code dengan fungsionalitas kustom yang dapat dibagikan di seluruh proyek dan tim. Panduan ini mencakup pembuatan plugins Anda sendiri dengan skills, agents, hooks, dan MCP servers.

Mencari untuk memasang plugins yang sudah ada? Lihat Temukan dan pasang plugins. Untuk spesifikasi teknis lengkap, lihat Referensi plugins.

Kapan menggunakan plugins vs konfigurasi standalone

Claude Code mendukung dua cara untuk menambahkan skills, agents, dan hooks kustom:

Pendekatan	Nama skill	Terbaik untuk
Standalone (direktori `.claude/`)	`/hello`	Alur kerja pribadi, kustomisasi khusus proyek, eksperimen cepat
Plugins (direktori dengan `.claude-plugin/plugin.json`)	`/plugin-name:hello`	Berbagi dengan rekan kerja, distribusi ke komunitas, rilis dengan versi, dapat digunakan kembali di seluruh proyek

Gunakan konfigurasi standalone ketika:

- Anda menyesuaikan Claude Code untuk satu proyek
- Konfigurasi bersifat pribadi dan tidak perlu dibagikan
- Anda bereksperimen dengan skills atau hooks sebelum mengemas mereka
- Anda menginginkan nama skill pendek seperti `/hello` atau `/deploy`

Gunakan plugins ketika:

- Anda ingin berbagi fungsionalitas dengan tim atau komunitas Anda
- Anda memerlukan skills/agents yang sama di seluruh beberapa proyek
- Anda menginginkan kontrol versi dan pembaruan mudah untuk ekstensi Anda

- Anda mendistribusikan melalui marketplace
- Anda tidak keberatan dengan skills yang diberi namespace seperti `/my-plugin:hello` (namespace mencegah konflik antara plugins)

> **Tip:**
>
> Mulai dengan konfigurasi standalone di `.claude/` untuk iterasi cepat, kemudian konversi ke plugin ketika Anda siap untuk berbagi.

Quickstart

Quickstart ini memandu Anda melalui pembuatan plugin dengan skill kustom. Anda akan membuat manifest (file konfigurasi yang mendefinisikan plugin Anda), menambahkan skill, dan mengujinya secara lokal menggunakan flag `--plugin-dir`.

Prasyarat

- Claude Code diinstal dan diautentikasi
- Claude Code versi 1.0.33 atau lebih baru (jalankan `claude --version` untuk memeriksa)

> **Note:**
>
> Jika Anda tidak melihat perintah `/plugin`, perbarui Claude Code ke versi terbaru. Lihat Troubleshooting untuk instruksi upgrade.

Buat plugin pertama Anda

Step 1: Buat direktori plugin

Setiap plugin berada di direktorinya sendiri yang berisi manifest dan skills, agents, atau hooks Anda. Buat satu sekarang:

```
mkdir my-first-plugin
```

Step 2: Buat manifest plugin

File manifest di `.claude-plugin/plugin.json` mendefinisikan identitas plugin Anda: nama, deskripsi, dan versinya. Claude Code menggunakan metadata ini untuk menampilkan plugin Anda di plugin manager.

Buat direktori `.claude-plugin` di dalam folder plugin Anda:

```
mkdir my-first-plugin/.claude-plugin
```

Kemudian buat `my-first-plugin/.claude-plugin/plugin.json` dengan konten ini:

```
{
"name": "my-first-plugin",
"description": "A greeting plugin to learn the basics",
"version": "1.0.0",
"author": {
"name": "Your Name"
}
}
```

Field	Tujuan
`name`	Pengidentifikasi unik dan namespace skill. Skills diawali dengan ini (misalnya, `/my-first-plugin:hello`).
`description`	Ditampilkan di plugin manager saat menjelajahi atau memasang plugins.
`version`	Lacak rilis menggunakan semantic versioning.
`author`	Opsional. Membantu untuk atribusi.

Untuk field tambahan seperti `homepage`, `repository`, dan `license`, lihat skema manifest lengkap.

Step 3: Tambahkan skill

Skills berada di direktori `skills/`. Setiap skill adalah folder yang berisi file `SKILL.md`. Nama folder menjadi nama skill, diawali dengan namespace plugin (`hello/` dalam plugin bernama `my-first-plugin` membuat `/my-first-plugin:hello`).

Buat direktori skill di folder plugin Anda:

```
mkdir -p my-first-plugin/skills/hello
```

Kemudian buat `my-first-plugin/skills/hello/SKILL.md` dengan konten ini:

```
---
description: Greet the user with a friendly message
disable-model-invocation: true
---

Greet the user warmly and ask how you can help them today.
```

Step 4: Uji plugin Anda

Jalankan Claude Code dengan flag `--plugin-dir` untuk memuat plugin Anda:

```
claude --plugin-dir ./my-first-plugin
```

Setelah Claude Code dimulai, coba skill baru Anda:

```
/my-first-plugin:hello
```

Anda akan melihat Claude merespons dengan salam. Jalankan `/help` untuk melihat skill Anda terdaftar di bawah namespace plugin.

> **Note:**
>
> **Mengapa namespace?** Plugin skills selalu diberi namespace (seperti `/greet:hello`) untuk mencegah konflik ketika beberapa plugins memiliki skills dengan nama yang sama.
>
> Untuk mengubah awalan namespace, perbarui field `name` di `plugin.json`.

Step 5: Tambahkan argumen skill

Buat skill Anda dinamis dengan menerima input pengguna. Placeholder `$ARGUMENTS` menangkap teks apa pun yang disediakan pengguna setelah nama skill.

Perbarui file `SKILL.md` Anda:

```
---
description: Greet the user with a personalized message
---

## Hello Skill

Greet the user named "$ARGUMENTS" warmly and ask how you can help them today. Make
the greeting personal and encouraging.
```

Jalankan `/reload-plugins` untuk mengambil perubahan, kemudian coba skill dengan nama Anda:

```
/my-first-plugin:hello Alex
```

Claude akan menyapa Anda dengan nama. Untuk lebih lanjut tentang meneruskan argumen ke skills, lihat Skills.

Anda telah berhasil membuat dan menguji plugin dengan komponen kunci ini:

- **Plugin manifest** (`.claude-plugin/plugin.json`): menjelaskan metadata plugin Anda
- **Direktori skills** (`skills/`): berisi skills kustom Anda
- **Argumen skill** (`$ARGUMENTS`): menangkap input pengguna untuk perilaku dinamis

> **Tip:**
>
> Flag `--plugin-dir` berguna untuk pengembangan dan pengujian. Ketika Anda siap untuk berbagi plugin Anda dengan orang lain, lihat Buat dan distribusikan marketplace plugin.

Ikhtisar struktur plugin

Anda telah membuat plugin dengan skill, tetapi plugins dapat mencakup banyak hal lagi: agents kustom, hooks, MCP servers, dan LSP servers.

> **Warning:**

Kesalahan umum: Jangan letakkan `commands/`, `agents/`, `skills/`, atau `hooks/` di dalam direktori `.claude-plugin/`. Hanya `plugin.json` yang masuk ke dalam `.claude-plugin/`. Semua direktori lainnya harus berada di tingkat root plugin.

Direktori	Lokasi	Tujuan
`.claude-plugin/`	Root plugin	Berisi manifest `plugin.json` (opsional jika komponen menggunakan lokasi default)
`commands/`	Root plugin	Skills sebagai file Markdown
`agents/`	Root plugin	Definisi agent kustom
`skills/`	Root plugin	Agent Skills dengan file `SKILL.md`
`hooks/`	Root plugin	Event handlers di `hooks.json`
`.mcp.json`	Root plugin	Konfigurasi MCP server
`.lsp.json`	Root plugin	Konfigurasi LSP server untuk code intelligence
`settings.json`	Root plugin	Settings default yang diterapkan ketika plugin diaktifkan

Note:

Langkah berikutnya: Siap menambahkan lebih banyak fitur? Lompat ke Kembangkan plugins yang lebih kompleks untuk menambahkan agents, hooks, MCP servers, dan LSP servers. Untuk spesifikasi teknis lengkap dari semua komponen plugin, lihat Referensi plugins.

Kembangkan plugins yang lebih kompleks

Setelah Anda nyaman dengan plugins dasar, Anda dapat membuat ekstensi yang lebih canggih.

Tambahkan Skills ke plugin Anda

Plugins dapat mencakup Agent Skills untuk memperluas kemampuan Claude. Skills diinvokasi oleh model: Claude secara otomatis menggunakannya berdasarkan konteks tugas.

Tambahkan direktori `skills/` di root plugin Anda dengan folder Skill yang berisi file `SKILL.md`:

```
my-plugin/
├── .claude-plugin/
│   └── plugin.json
└── skills/
    └── code-review/
        └── SKILL.md
```

Setiap `SKILL.md` memerlukan frontmatter dengan field `name` dan `description`, diikuti dengan instruksi:

```
---
name: code-review
description: Reviews code for best practices and potential issues. Use when
reviewing code, checking PRs, or analyzing code quality.
---

When reviewing code, check for:
1. Code organization and structure
2. Error handling
3. Security concerns
4. Test coverage
```

Setelah memasang plugin, jalankan `/reload-plugins` untuk memuat Skills. Untuk panduan authoring Skill lengkap termasuk progressive disclosure dan pembatasan tool, lihat Agent Skills.

Tambahkan LSP servers ke plugin Anda

> **Tip:**

> Untuk bahasa umum seperti TypeScript, Python, dan Rust, pasang plugin LSP yang sudah dibangun sebelumnya dari marketplace resmi. Buat plugin LSP kustom hanya ketika Anda memerlukan dukungan untuk bahasa yang belum tercakup.

Plugin LSP (Language Server Protocol) memberikan Claude code intelligence real-time. Jika Anda perlu mendukung bahasa yang tidak memiliki plugin LSP resmi, Anda dapat membuat plugin Anda sendiri dengan menambahkan file `.lsp.json` ke plugin Anda:

```
{
  "go": {
    "command": "gopls",
    "args": ["serve"],
    "extensionToLanguage": {
      ".go": "go"
    }
  }
}
```

Pengguna yang memasang plugin Anda harus memiliki binary language server yang diinstal di mesin mereka.

Untuk opsi konfigurasi LSP lengkap, lihat LSP servers.

Kirim settings default dengan plugin Anda

Plugins dapat menyertakan file `settings.json` di root plugin untuk menerapkan konfigurasi default ketika plugin diaktifkan. Saat ini, hanya key `agent` yang didukung.

Mengatur `agent` mengaktifkan salah satu custom agents plugin sebagai thread utama, menerapkan system prompt, pembatasan tool, dan modelnya. Ini memungkinkan plugin untuk mengubah perilaku Claude Code secara default ketika diaktifkan.

```
{
  "agent": "security-reviewer"
}
```

Contoh ini mengaktifkan agent `security-reviewer` yang didefinisikan di direktori `agents/` plugin. Settings dari `settings.json` mengambil prioritas atas `settings` yang dideklarasikan di `plugin.json`. Key yang tidak dikenal diabaikan secara diam-diam.

Organisir plugins kompleks

Untuk plugins dengan banyak komponen, organisir struktur direktori Anda berdasarkan fungsionalitas. Untuk layout direktori lengkap dan pola organisasi, lihat Struktur direktori plugin.

Uji plugins Anda secara lokal

Gunakan flag `--plugin-dir` untuk menguji plugins selama pengembangan. Ini memuat plugin Anda secara langsung tanpa memerlukan instalasi.

```
claude --plugin-dir ./my-plugin
```

Saat Anda membuat perubahan pada plugin Anda, jalankan `/reload-plugins` untuk mengambil pembaruan tanpa memulai ulang. Perubahan pada konfigurasi LSP server masih memerlukan restart penuh. Uji komponen plugin Anda:

- Coba skills Anda dengan `/plugin-name:skill-name`
- Periksa bahwa agents muncul di `/agents`
- Verifikasi hooks bekerja seperti yang diharapkan

> **Tip:**
>
> Anda dapat memuat beberapa plugins sekaligus dengan menentukan flag berkali-kali:
>
> ```
> claude --plugin-dir ./plugin-one --plugin-dir ./plugin-two
> ```

Debug masalah plugin

Jika plugin Anda tidak bekerja seperti yang diharapkan:

1. **Periksa struktur**: Pastikan direktori Anda berada di root plugin, bukan di dalam `.claude-plugin/`
2. **Uji komponen secara individual**: Periksa setiap command, agent, dan hook secara terpisah
3. **Gunakan alat validasi dan debugging**: Lihat Alat debugging dan pengembangan untuk perintah CLI dan teknik troubleshooting

Bagikan plugins Anda

Ketika plugin Anda siap untuk dibagikan:

1. **Tambahkan dokumentasi**: Sertakan `README.md` dengan instruksi instalasi dan penggunaan
2. **Versi plugin Anda**: Gunakan semantic versioning di `plugin.json` Anda
3. **Buat atau gunakan marketplace**: Distribusikan melalui plugin marketplaces untuk instalasi
4. **Uji dengan orang lain**: Minta anggota tim menguji plugin sebelum distribusi yang lebih luas

Setelah plugin Anda berada di marketplace, orang lain dapat memasangnya menggunakan instruksi di Temukan dan pasang plugins.

Kirimkan plugin Anda ke marketplace resmi

Untuk mengirimkan plugin ke marketplace Anthropic resmi, gunakan salah satu formulir pengajuan in-app:

- **Claude.ai**: claude.ai/settings/plugins/submit
- **Console**: platform.claude.com/plugins/submit

> **Note:**
>
> Untuk spesifikasi teknis lengkap, teknik debugging, dan strategi distribusi, lihat Referensi plugins.

Konversi konfigurasi yang ada ke plugins

Jika Anda sudah memiliki skills atau hooks di direktori `.claude/` Anda, Anda dapat mengonversinya menjadi plugin untuk berbagi dan distribusi yang lebih mudah.

Langkah migrasi

Step 1: Buat struktur plugin

Buat direktori plugin baru:

```
mkdir -p my-plugin/.claude-plugin
```

Buat file manifest di `my-plugin/.claude-plugin/plugin.json`:

```
{
  "name": "my-plugin",
  "description": "Migrated from standalone configuration",
  "version": "1.0.0"
}
```

Step 2: Salin file yang ada

Salin konfigurasi yang ada ke direktori plugin:

```
## Copy commands
cp -r .claude/commands my-plugin/

## Copy agents (if any)
cp -r .claude/agents my-plugin/

## Copy skills (if any)
cp -r .claude/skills my-plugin/
```

Step 3: Migrasi hooks

Jika Anda memiliki hooks di settings Anda, buat direktori hooks:

```
mkdir my-plugin/hooks
```

Buat `my-plugin/hooks/hooks.json` dengan konfigurasi hooks Anda. Salin objek `hooks` dari `.claude/settings.json` atau `settings.local.json` Anda, karena formatnya sama. Perintah menerima input hook sebagai JSON di stdin, jadi gunakan `jq` untuk mengekstrak path file:

```
{
  "hooks": {
    "PostToolUse": [
      {
        "matcher": "Write|Edit",
        "hooks": [{ "type": "command", "command":
"jq -r '.tool_input.file_path' | xargs npm run lint:fix" }]
      }
    ]
  }
}
```

Step 4: Uji plugin yang dimigrasikan

Muat plugin Anda untuk memverifikasi semuanya berfungsi:

```
claude --plugin-dir ./my-plugin
```

Uji setiap komponen: jalankan commands Anda, periksa agents muncul di `/agents`, dan verifikasi hooks dipicu dengan benar.

Apa yang berubah saat migrasi

Standalone (`.claude/`)	Plugin
Hanya tersedia di satu proyek	Dapat dibagikan melalui marketplaces
File di `.claude/commands/`	File di `plugin-name/commands/`
Hooks di `settings.json`	Hooks di `hooks/hooks.json`
Harus menyalin secara manual untuk berbagi	Pasang dengan `/plugin install`

> **Note:**
>
> Setelah migrasi, Anda dapat menghapus file asli dari `.claude/` untuk menghindari duplikat. Versi plugin akan mengambil prioritas saat dimuat.

Langkah berikutnya

Sekarang bahwa Anda memahami sistem plugin Claude Code, berikut adalah jalur yang disarankan untuk tujuan yang berbeda:

Untuk pengguna plugin

- Temukan dan pasang plugins: jelajahi marketplaces dan pasang plugins
- Konfigurasi marketplaces tim: atur plugins tingkat repository untuk tim Anda

Untuk pengembang plugin

- Buat dan distribusikan marketplace: paket dan bagikan plugins Anda
- Referensi plugins: spesifikasi teknis lengkap
- Selami lebih dalam komponen plugin spesifik:
  - Skills: detail pengembangan skill
  - Subagents: konfigurasi dan kemampuan agent
  - Hooks: penanganan event dan otomasi
  - MCP: integrasi tool eksternal

Referensi Plugins

> *Referensi teknis lengkap untuk sistem plugin Claude Code, termasuk skema, perintah CLI, dan spesifikasi komponen.*

> **Tip:**
>
> Mencari cara memasang plugins? Lihat Temukan dan pasang plugins. Untuk membuat plugins, lihat Plugins. Untuk mendistribusikan plugins, lihat Plugin marketplaces.

Referensi ini menyediakan spesifikasi teknis lengkap untuk sistem plugin Claude Code, termasuk skema komponen, perintah CLI, dan alat pengembangan.

Sebuah **plugin** adalah direktori yang mandiri berisi komponen yang memperluas Claude Code dengan fungsionalitas khusus. Komponen plugin mencakup skills, agents, hooks, MCP servers, dan LSP servers.

Referensi komponen plugin

Skills

Plugins menambahkan skills ke Claude Code, membuat pintasan `/name` yang dapat Anda atau Claude panggil.

Lokasi: Direktori `skills/` atau `commands/` di root plugin

Format file: Skills adalah direktori dengan `SKILL.md` ; commands adalah file markdown sederhana

Struktur skill:

```
skills/
├── pdf-processor/
│   ├── SKILL.md
│   ├── reference.md (opsional)
│   └── scripts/ (opsional)
└── code-reviewer/
    └── SKILL.md
```

Perilaku integrasi:

- Skills dan commands secara otomatis ditemukan saat plugin dipasang
- Claude dapat memanggilnya secara otomatis berdasarkan konteks tugas
- Skills dapat menyertakan file pendukung di samping SKILL.md

Untuk detail lengkap, lihat Skills.

Agents

Plugins dapat menyediakan subagents khusus untuk tugas-tugas tertentu yang dapat Claude panggil secara otomatis jika sesuai.

Lokasi: Direktori `agents/` di root plugin

Format file: File markdown yang menjelaskan kemampuan agent

Struktur agent:

```
---
name: agent-name
description: Apa yang agent ini spesialisasikan dan kapan Claude harus
memanggilnya
---

Prompt sistem terperinci untuk agent yang menjelaskan peran, keahlian, dan
perilakunya.
```

Titik integrasi:

- Agents muncul di antarmuka `/agents`
- Claude dapat memanggil agents secara otomatis berdasarkan konteks tugas
- Agents dapat dipanggil secara manual oleh pengguna
- Plugin agents bekerja bersama agents Claude bawaan

Untuk detail lengkap, lihat Subagents.

Hooks

Plugins dapat menyediakan event handlers yang merespons peristiwa Claude Code secara otomatis.

Lokasi: `hooks/hooks.json` di root plugin, atau inline di plugin.json

Format: Konfigurasi JSON dengan event matchers dan actions

Konfigurasi hook:

```
{
  "hooks": {
    "PostToolUse": [
      {
        "matcher": "Write|Edit",
        "hooks": [
          {
            "type": "command",
            "command": "${CLAUDE_PLUGIN_ROOT}/scripts/format-code.sh"
          }
        ]
      }
    ]
  }
}
```

Event yang tersedia:

- `PreToolUse` : Sebelum Claude menggunakan alat apa pun
- `PostToolUse` : Setelah Claude berhasil menggunakan alat apa pun
- `PostToolUseFailure` : Setelah eksekusi alat Claude gagal
- `PermissionRequest` : Saat dialog izin ditampilkan
- `UserPromptSubmit` : Saat pengguna mengirimkan prompt
- `Notification` : Saat Claude Code mengirim notifikasi
- `Stop` : Saat Claude mencoba berhenti
- `SubagentStart` : Saat subagent dimulai
- `SubagentStop` : Saat subagent mencoba berhenti
- `SessionStart` : Di awal sesi
- `SessionEnd` : Di akhir sesi
- `TeammateIdle` : Saat rekan tim tim agent akan menjadi idle
- `TaskCompleted` : Saat tugas ditandai sebagai selesai
- `PreCompact` : Sebelum riwayat percakapan dikompres

Tipe hook:

- `command` : Jalankan perintah shell atau script
- `prompt` : Evaluasi prompt dengan LLM (menggunakan placeholder `$ARGUMENTS` untuk konteks)
- `agent` : Jalankan verifier agentic dengan alat untuk tugas verifikasi kompleks

MCP servers

Plugins dapat menggabungkan Model Context Protocol (MCP) servers untuk menghubungkan Claude Code dengan alat dan layanan eksternal.

Lokasi: `.mcp.json` di root plugin, atau inline di plugin.json

Format: Konfigurasi MCP server standar

Konfigurasi MCP server:

```
{
  "mcpServers": {
    "plugin-database": {
      "command": "${CLAUDE_PLUGIN_ROOT}/servers/db-server",
      "args": ["--config", "${CLAUDE_PLUGIN_ROOT}/config.json"],
      "env": {
        "DB_PATH": "${CLAUDE_PLUGIN_ROOT}/data"
      }
    },
    "plugin-api-client": {
      "command": "npx",
      "args": ["@company/mcp-server", "--plugin-mode"],
      "cwd": "${CLAUDE_PLUGIN_ROOT}"
    }
  }
}
```

Perilaku integrasi:

- Plugin MCP servers dimulai secara otomatis saat plugin diaktifkan
- Servers muncul sebagai alat MCP standar di toolkit Claude
- Kemampuan server terintegrasi dengan mulus dengan alat Claude yang ada
- Plugin servers dapat dikonfigurasi secara independen dari MCP servers pengguna

LSP servers

> **Tip:**
>
> Mencari cara menggunakan LSP plugins? Pasang dari marketplace resmi: cari "lsp" di tab Discover `/plugin`. Bagian ini mendokumentasikan cara membuat LSP plugins untuk bahasa yang tidak tercakup oleh marketplace resmi.

Plugins dapat menyediakan server Language Server Protocol (LSP) untuk memberikan Claude intelijen kode real-time saat bekerja pada codebase Anda.

Integrasi LSP menyediakan:

- **Diagnostik instan**: Claude melihat kesalahan dan peringatan segera setelah setiap edit
- **Navigasi kode**: buka definisi, temukan referensi, dan informasi hover
- **Kesadaran bahasa**: informasi tipe dan dokumentasi untuk simbol kode

Lokasi: `.lsp.json` di root plugin, atau inline di `plugin.json`

Format: Konfigurasi JSON yang memetakan nama language server ke konfigurasinya

Format file `.lsp.json`:

```
{
  "go": {
    "command": "gopls",
    "args": ["serve"],
    "extensionToLanguage": {
      ".go": "go"
    }
  }
}
```

Inline di `plugin.json`:

```
{
  "name": "my-plugin",
  "lspServers": {
    "go": {
      "command": "gopls",
      "args": ["serve"],
      "extensionToLanguage": {
        ".go": "go"
      }
    }
  }
}
```

Field yang diperlukan:

Field	Deskripsi
`command`	Biner LSP yang akan dijalankan (harus ada di PATH)
`extensionToLanguage`	Memetakan ekstensi file ke pengenal bahasa

Field opsional:

Field	Deskripsi
`args`	Argumen baris perintah untuk LSP server
`transport`	Transport komunikasi: `stdio` (default) atau `socket`
`env`	Variabel lingkungan yang diatur saat memulai server
`initializationOptions`	Opsi yang diteruskan ke server selama inisialisasi
`settings`	Pengaturan yang diteruskan melalui `workspace/didChangeConfiguration`
`workspaceFolder`	Jalur folder workspace untuk server
`startupTimeout`	Waktu maksimal untuk menunggu startup server (milidetik)
`shutdownTimeout`	Waktu maksimal untuk menunggu shutdown yang elegan (milidetik)

Field	Deskripsi
`restartOnCrash`	Apakah secara otomatis memulai ulang server jika crash
`maxRestarts`	Jumlah maksimal upaya restart sebelum menyerah

> **Warning:**
>
> **Anda harus memasang biner language server secara terpisah.** LSP plugins mengonfigurasi cara Claude Code terhubung ke language server, tetapi mereka tidak menyertakan server itu sendiri. Jika Anda melihat `Executable not found in $PATH` di tab Errors `/plugin`, pasang biner yang diperlukan untuk bahasa Anda.

LSP plugins yang tersedia:

Plugin	Language server	Perintah instalasi
`pyright-lsp`	Pyright (Python)	`pip install pyright` atau `npm install -g pyright`
`typescript-lsp`	TypeScript Language Server	`npm install -g typescript-language-server typescript`
`rust-lsp`	rust-analyzer	Lihat instalasi rust-analyzer

Pasang language server terlebih dahulu, kemudian pasang plugin dari marketplace.

Cakupan instalasi plugin

Saat Anda memasang plugin, Anda memilih **cakupan** yang menentukan di mana plugin tersedia dan siapa lagi yang dapat menggunakannya:

Cakupan	File pengaturan	Kasus penggunaan
`user`	`~/.claude/settings.json`	Plugin pribadi tersedia di semua proyek (default)
`project`	`.claude/settings.json`	Plugin tim yang dibagikan melalui version control

Cakupan	File pengaturan	Kasus penggunaan
`local`	`.claude/settings.local.json`	Plugin khusus proyek, gitignored
`managed`	Pengaturan terkelola	Plugin terkelola (read-only, hanya update)

Plugins menggunakan sistem cakupan yang sama dengan konfigurasi Claude Code lainnya. Untuk instruksi instalasi dan flag cakupan, lihat Pasang plugins. Untuk penjelasan lengkap tentang cakupan, lihat Configuration scopes.

Skema manifest plugin

File `.claude-plugin/plugin.json` mendefinisikan metadata dan konfigurasi plugin Anda. Bagian ini mendokumentasikan semua field dan opsi yang didukung.

Manifest bersifat opsional. Jika dihilangkan, Claude Code secara otomatis menemukan komponen di lokasi default dan menurunkan nama plugin dari nama direktori. Gunakan manifest saat Anda perlu memberikan metadata atau jalur komponen khusus.

Skema lengkap

```
{
  "name": "plugin-name",
  "version": "1.2.0",
  "description": "Brief plugin description",
  "author": {
    "name": "Author Name",
    "email": "author@example.com",
    "url": "https://github.com/author"
  },
  "homepage": "https://docs.example.com/plugin",
  "repository": "https://github.com/author/plugin",
  "license": "MIT",
  "keywords": ["keyword1", "keyword2"],
  "commands": ["./custom/commands/special.md"],
  "agents": "./custom/agents/",
  "skills": "./custom/skills/",
  "hooks": "./config/hooks.json",
  "mcpServers": "./mcp-config.json",
  "outputStyles": "./styles/",
  "lspServers": "./.lsp.json"
}
```

Field yang diperlukan

Jika Anda menyertakan manifest, `name` adalah satu-satunya field yang diperlukan.

Field	Tipe	Deskripsi	Contoh
`name`	string	Pengenal unik (ke-bab-case, tanpa spasi)	`"deployment-tools"`

Nama ini digunakan untuk namespacing komponen. Misalnya, di UI, agent `agent-creator` untuk plugin dengan nama `plugin-dev` akan muncul sebagai `plugin-dev:agent-creator`.

Field metadata

Field	Tipe	Deskripsi	Contoh
`version`	string	Versi semantik. Jika juga diatur di entri marketplace, `plugin.json` memiliki prioritas. Anda hanya perlu mengaturnya di satu tempat.	`"2.1.0"`
`description`	string	Penjelasan singkat tentang tujuan plugin	`"Deployment automation tools"`
`author`	object	Informasi penulis	`{"name": "Dev Team", "email": "dev@company.com"}`
`homepage`	string	URL dokumentasi	`"https://docs.example.com"`
`repository`	string	URL kode sumber	`"https://github.com/user/plugin"`
`license`	string	Pengenal lisensi	`"MIT"`, `"Apache-2.0"`
`keywords`	array	Tag penemuan	`["deployment", "ci-cd"]`

Field jalur komponen

Field	Tipe	Deskripsi	Contoh
`commands`	string|array	File/direktori command tambahan	`"./custom/cmd.md"` atau `["./cmd1.md"]`

Field	Tipe	Deskripsi	Contoh
`agents`	string|array	File agent tambahan	`"./custom/agents/reviewer.md"`
`skills`	string|array	Direktori skill tambahan	`"./custom/skills/"`
`hooks`	string|array|object	Jalur konfigurasi hook atau konfigurasi inline	`"./my-extra-hooks.json"`
`mcpServers`	string|array|object	Jalur konfigurasi MCP atau konfigurasi inline	`"./my-extra-mcp-config.json"`
`outputStyles`	string|array	File/direktori gaya output tambahan	`"./styles/"`
`lspServers`	string|array|object	Konfigurasi Language Server Protocol untuk intelijen kode (buka definisi, temukan referensi, dll.)	`"./.lsp.json"`

Aturan perilaku jalur

Penting: Jalur khusus melengkapi direktori default - mereka tidak menggantikannya.

- Jika `commands/` ada, itu dimuat selain jalur command khusus
- Semua jalur harus relatif terhadap root plugin dan dimulai dengan `./`
- Commands dari jalur khusus menggunakan aturan penamaan dan namespacing yang sama
- Beberapa jalur dapat ditentukan sebagai array untuk fleksibilitas

Contoh jalur:

```
{
  "commands": [
    "./specialized/deploy.md",
    "./utilities/batch-process.md"
  ],
  "agents": [
    "./custom-agents/reviewer.md",
    "./custom-agents/tester.md"
  ]
}
```

Variabel lingkungan

`${CLAUDE_PLUGIN_ROOT}` : Berisi jalur absolut ke direktori plugin Anda. Gunakan ini di hooks, MCP servers, dan scripts untuk memastikan jalur yang benar terlepas dari lokasi instalasi.

```
{
  "hooks": {
    "PostToolUse": [
      {
        "hooks": [
          {
            "type": "command",
            "command": "${CLAUDE_PLUGIN_ROOT}/scripts/process.sh"
          }
        ]
      }
    ]
  }
}
```

Caching plugin dan resolusi file

Plugins ditentukan dalam salah satu dari dua cara:

- Melalui `claude --plugin-dir`, untuk durasi sesi.
- Melalui marketplace, dipasang untuk sesi mendatang.

Untuk tujuan keamanan dan verifikasi, Claude Code menyalin plugin *marketplace* ke **plugin cache** lokal pengguna (`~/.claude/plugins/cache`) daripada menggunakannya di tempat. Memahami perilaku ini penting saat mengembangkan plugins yang mereferensikan file eksternal.

Batasan path traversal

Plugin yang dipasang tidak dapat mereferensikan file di luar direktorinya. Jalur yang melintasi di luar root plugin (seperti `../shared-utils`) tidak akan berfungsi setelah instalasi karena file eksternal tersebut tidak disalin ke cache.

Bekerja dengan dependensi eksternal

Jika plugin Anda perlu mengakses file di luar direktorinya, Anda dapat membuat symbolic links ke file eksternal dalam direktori plugin Anda. Symlinks dihormati selama proses penyalinan:

```
## Di dalam direktori plugin Anda
ln -s /path/to/shared-utils ./shared-utils
```

Konten yang di-symlink akan disalin ke plugin cache. Ini memberikan fleksibilitas sambil mempertahankan manfaat keamanan dari sistem caching.

Struktur direktori plugin

Tata letak plugin standar

Plugin lengkap mengikuti struktur ini:

```
enterprise-plugin/
├── .claude-plugin/           # Direktori metadata (opsional)
│   └── plugin.json             # plugin manifest
├── commands/                 # Lokasi command default
│   ├── status.md
│   └── logs.md
├── agents/                   # Lokasi agent default
│   ├── security-reviewer.md
│   ├── performance-tester.md
│   └── compliance-checker.md
├── skills/                   # Agent Skills
│   ├── code-reviewer/
│   │   └── SKILL.md
│   └── pdf-processor/
│       ├── SKILL.md
│       └── scripts/
├── hooks/                    # Konfigurasi hook
│   ├── hooks.json           # Konfigurasi hook utama
│   └── security-hooks.json  # Hook tambahan
├── settings.json             # Pengaturan default untuk plugin
├── .mcp.json                 # Definisi MCP server
├── .lsp.json                 # Konfigurasi LSP server
├── scripts/                  # Hook dan utility scripts
│   ├── security-scan.sh
│   ├── format-code.py
│   └── deploy.js
├── LICENSE                   # File lisensi
└── CHANGELOG.md              # Riwayat versi
```

Warning:

Direktori `.claude-plugin/` berisi file `plugin.json`. Semua direktori lainnya (commands/, agents/, skills/, hooks/) harus berada di root plugin, bukan di dalam `.claude-plugin/`.

Referensi lokasi file

Komponen	Lokasi Default	Tujuan
Manifest	`.claude-plugin/plugin.json`	Metadata dan konfigurasi plugin (opsional)
Commands	`commands/`	File Markdown Skill (legacy; gunakan `skills/` untuk skill baru)
Agents	`agents/`	File Markdown Subagent
Skills	`skills/`	Skills dengan struktur `<name>/SKILL.md`
Hooks	`hooks/hooks.json`	Konfigurasi hook
MCP servers	`.mcp.json`	Definisi MCP server
LSP servers	`.lsp.json`	Konfigurasi language server
Settings	`settings.json`	Konfigurasi default yang diterapkan saat plugin diaktifkan. Saat ini hanya pengaturan `agent` yang didukung

Referensi perintah CLI

Claude Code menyediakan perintah CLI untuk manajemen plugin non-interaktif, berguna untuk scripting dan otomasi.

plugin install

Pasang plugin dari marketplace yang tersedia.

```
claude plugin install <plugin> [options]
```

Argumen:

- `<plugin>` : Nama plugin atau `plugin-name@marketplace-name` untuk marketplace tertentu

Opsi:

Opsi	Deskripsi	Default
`-s, --scope <scope>`	Cakupan instalasi: `user`, `project`, atau `local`	`user`
`-h, --help`	Tampilkan bantuan untuk perintah	

Cakupan menentukan file pengaturan mana yang ditambahkan plugin yang dipasang. Misalnya, –scope project menulis ke `enabledPlugins` di .claude/settings.json, membuat plugin tersedia untuk semua orang yang mengkloning repositori proyek.

Contoh:

```
## Pasang ke cakupan user (default)
claude plugin install formatter@my-marketplace

## Pasang ke cakupan project (dibagikan dengan tim)
claude plugin install formatter@my-marketplace --scope project

## Pasang ke cakupan local (gitignored)
claude plugin install formatter@my-marketplace --scope local
```

plugin uninstall

Hapus plugin yang dipasang.

```
claude plugin uninstall <plugin> [options]
```

Argumen:

- `<plugin>`: Nama plugin atau `plugin-name@marketplace-name`

Opsi:

Opsi	Deskripsi	Default
`-s, --scope <scope>`	Hapus dari cakupan: `user`, `project`, atau `local`	`user`
`-h, --help`	Tampilkan bantuan untuk perintah	

Alias: `remove`, `rm`

plugin enable

Aktifkan plugin yang dinonaktifkan.

```
claude plugin enable <plugin> [options]
```

Argumen:

- `<plugin>`: Nama plugin atau `plugin-name@marketplace-name`

Opsi:

Opsi	Deskripsi	Default
`-s, --scope <scope>`	Cakupan untuk diaktifkan: `user`, `project`, atau `local`	`user`
`-h, --help`	Tampilkan bantuan untuk perintah	

plugin disable

Nonaktifkan plugin tanpa menghapusnya.

```
claude plugin disable <plugin> [options]
```

Argumen:

- `<plugin>`: Nama plugin atau `plugin-name@marketplace-name`

Opsi:

Opsi	Deskripsi	Default
`-s, --scope <scope>`	Cakupan untuk dinonaktifkan: `user`, `project`, atau `local`	`user`
`-h, --help`	Tampilkan bantuan untuk perintah	

plugin update

Perbarui plugin ke versi terbaru.

```
claude plugin update <plugin> [options]
```

Argumen:

- `<plugin>`: Nama plugin atau `plugin-name@marketplace-name`

Opsi:

Opsi	Deskripsi	Default
`-s, --scope <scope>`	Cakupan untuk diperbarui: `user`, `project`, `local`, atau `managed`	`user`
`-h, --help`	Tampilkan bantuan untuk perintah	

Alat debugging dan pengembangan

Perintah debugging

Gunakan `claude --debug` (atau `/debug` dalam TUI) untuk melihat detail loading plugin:

Ini menunjukkan:

- Plugin mana yang sedang dimuat
- Kesalahan apa pun dalam manifest plugin
- Registrasi command, agent, dan hook
- Inisialisasi MCP server

Masalah umum

Masalah	Penyebab	Solusi
Plugin tidak dimuat	`plugin.json` tidak valid	Validasi sintaks JSON dengan `claude plugin validate` atau `/plugin validate`
Commands tidak muncul	Struktur direktori salah	Pastikan `commands/` di root, bukan di `.claude-plugin/`
Hooks tidak aktif	Script tidak dapat dieksekusi	Jalankan `chmod +x script.sh`

Masalah	Penyebab	Solusi
MCP server gagal	`${CLAUDE_PLUGIN_ROOT}` hilang	Gunakan variabel untuk semua jalur plugin
Kesalahan jalur	Jalur absolut digunakan	Semua jalur harus relatif dan dimulai dengan `./`
LSP `Executable not found in $PATH`	Language server tidak dipasang	Pasang biner (misalnya, `npm install -g typescript-language-server typescript`)

Contoh pesan kesalahan

Kesalahan validasi manifest:

- `Invalid JSON syntax: Unexpected token } in JSON at position 142` : periksa koma yang hilang, koma ekstra, atau string yang tidak dikutip
- `Plugin has an invalid manifest file at .claude-plugin/plugin.json. Validation errors: name: Required` : field yang diperlukan hilang
- `Plugin has a corrupt manifest file at .claude-plugin/plugin.json. JSON parse error: ...` : kesalahan sintaks JSON

Kesalahan loading plugin:

- `Warning: No commands found in plugin my-plugin custom directory: ./cmds. Expected .md files or SKILL.md in subdirectories.` : jalur command ada tetapi tidak berisi file command yang valid
- `Plugin directory not found at path: ./plugins/my-plugin. Check that the marketplace entry has the correct path.` : jalur `source` di marketplace.json menunjuk ke direktori yang tidak ada
- `Plugin my-plugin has conflicting manifests: both plugin.json and marketplace entry specify components.` : hapus definisi komponen duplikat atau hapus `strict: false` di entri marketplace

Troubleshooting hook

Hook script tidak dieksekusi:

1. Periksa script dapat dieksekusi: `chmod +x ./scripts/your-script.sh`
2. Verifikasi baris shebang: Baris pertama harus `#!/bin/bash` atau `#!/usr/bin/env bash`

3. Periksa jalur menggunakan `${CLAUDE_PLUGIN_ROOT}`: `"command": "${CLAUDE_PLUGIN_ROOT}/scripts/your-script.sh"`
4. Uji script secara manual: `./scripts/your-script.sh`

Hook tidak dipicu pada event yang diharapkan:

1. Verifikasi nama event benar (case-sensitive): `PostToolUse`, bukan `postToolUse`
2. Periksa pola matcher cocok dengan alat Anda: `"matcher": "Write|Edit"` untuk operasi file
3. Konfirmkan tipe hook valid: `command`, `prompt`, atau `agent`

Troubleshooting MCP server

Server tidak dimulai:

1. Periksa command ada dan dapat dieksekusi
2. Verifikasi semua jalur menggunakan variabel `${CLAUDE_PLUGIN_ROOT}`
3. Periksa log MCP server: `claude --debug` menunjukkan kesalahan inisialisasi
4. Uji server secara manual di luar Claude Code

Alat server tidak muncul:

1. Pastikan server dikonfigurasi dengan benar di `.mcp.json` atau `plugin.json`
2. Verifikasi server mengimplementasikan protokol MCP dengan benar
3. Periksa timeout koneksi di output debug

Kesalahan struktur direktori

Gejala: Plugin dimuat tetapi komponen (commands, agents, hooks) hilang.

Struktur yang benar: Komponen harus berada di root plugin, bukan di dalam `.claude-plugin/`. Hanya `plugin.json` yang termasuk di `.claude-plugin/`.

```
my-plugin/
├── .claude-plugin/
│   └── plugin.json      ← Hanya manifest di sini
├── commands/            ← Di level root
├── agents/              ← Di level root
└── hooks/               ← Di level root
```

Jika komponen Anda berada di dalam `.claude-plugin/`, pindahkan ke root plugin.

Daftar periksa debug:

1. Jalankan `claude --debug` dan cari pesan "loading plugin"
2. Periksa bahwa setiap direktori komponen terdaftar di output debug
3. Verifikasi izin file memungkinkan membaca file plugin

Referensi distribusi dan versioning

Manajemen versi

Ikuti semantic versioning untuk rilis plugin:

```
{
  "name": "my-plugin",
  "version": "2.1.0"
}
```

Format versi: `MAJOR.MINOR.PATCH`

- **MAJOR**: Perubahan breaking (perubahan API yang tidak kompatibel)
- **MINOR**: Fitur baru (penambahan yang kompatibel ke belakang)
- **PATCH**: Perbaikan bug (perbaikan yang kompatibel ke belakang)

Best practices:

- Mulai dari `1.0.0` untuk rilis stabil pertama Anda
- Perbarui versi di `plugin.json` sebelum mendistribusikan perubahan
- Dokumentasikan perubahan dalam file `CHANGELOG.md`
- Gunakan versi pre-release seperti `2.0.0-beta.1` untuk pengujian

> **Warning:**
>
> Claude Code menggunakan versi untuk menentukan apakah akan memperbarui plugin Anda. Jika Anda mengubah kode plugin Anda tetapi tidak meningkatkan versi di `plugin.json`, pengguna plugin Anda yang ada tidak akan melihat perubahan Anda karena caching.
>
> Jika plugin Anda berada dalam direktori marketplace, Anda dapat mengelola versi melalui `marketplace.json` sebagai gantinya dan menghilangkan field `version` dari `plugin.json`.

Lihat juga

- Plugins - Tutorial dan penggunaan praktis
- Plugin marketplaces - Membuat dan mengelola marketplace
- Skills - Detail pengembangan skill
- Subagents - Konfigurasi dan kemampuan agent
- Hooks - Penanganan event dan otomasi
- MCP - Integrasi alat eksternal
- Settings - Opsi konfigurasi untuk plugins

Buat dan distribusikan marketplace plugin

> *Bangun dan hosting marketplace plugin untuk mendistribusikan ekstensi Claude Code di seluruh tim dan komunitas.*

Sebuah **marketplace plugin** adalah katalog yang memungkinkan Anda mendistribusikan plugin kepada orang lain. Marketplace menyediakan penemuan terpusat, pelacakan versi, pembaruan otomatis, dan dukungan untuk berbagai jenis sumber (repositori git, jalur lokal, dan lainnya). Panduan ini menunjukkan cara membuat marketplace Anda sendiri untuk berbagi plugin dengan tim atau komunitas Anda.

Mencari cara memasang plugin dari marketplace yang sudah ada? Lihat Temukan dan pasang plugin yang sudah dibuat.

Ikhtisar

Membuat dan mendistribusikan marketplace melibatkan:

1. **Membuat plugin**: bangun satu atau lebih plugin dengan commands, agents, hooks, MCP servers, atau LSP servers. Panduan ini mengasumsikan Anda sudah memiliki plugin untuk didistribusikan; lihat Buat plugin untuk detail tentang cara membuat plugin.
2. **Membuat file marketplace**: tentukan `marketplace.json` yang mencantumkan plugin Anda dan di mana menemukannya (lihat Buat file marketplace).
3. **Host marketplace**: dorong ke GitHub, GitLab, atau host git lainnya (lihat Host dan distribusikan marketplace).
4. **Bagikan dengan pengguna**: pengguna menambahkan marketplace Anda dengan `/plugin marketplace add` dan memasang plugin individual (lihat Temukan dan pasang plugin).

Setelah marketplace Anda aktif, Anda dapat memperbaruinya dengan mendorong perubahan ke repositori Anda. Pengguna menyegarkan salinan lokal mereka dengan `/plugin marketplace update`.

Panduan: buat marketplace lokal

Contoh ini membuat marketplace dengan satu plugin: skill `/quality-review` untuk ulasan kode. Anda akan membuat struktur direktori, menambahkan skill, membuat manifest plugin dan katalog marketplace, kemudian memasang dan mengujinya.

Step 1: Buat struktur direktori

```
mkdir -p my-marketplace/.claude-plugin
mkdir -p my-marketplace/plugins/quality-review-plugin/.claude-plugin
mkdir -p my-marketplace/plugins/quality-review-plugin/skills/quality-review
```

Step 2: Buat skill

Buat file `SKILL.md` yang mendefinisikan apa yang dilakukan skill `/quality-review`.

```
---
description: Review code for bugs, security, and performance
disable-model-invocation: true
---

Review the code I've selected or the recent changes for:
- Potential bugs or edge cases
- Security concerns
- Performance issues
- Readability improvements

Be concise and actionable.
```

Step 3: Buat manifest plugin

Buat file `plugin.json` yang mendeskripsikan plugin. Manifest berada di direktori `.claude-plugin/`.

```
{
  "name": "quality-review-plugin",
  "description": "Adds a /quality-review skill for quick code reviews",
  "version": "1.0.0"
}
```

Step 4: Buat file marketplace

Buat katalog marketplace yang mencantumkan plugin Anda.

```
{
  "name": "my-plugins",
  "owner": {
    "name": "Your Name"
  },
  "plugins": [
    {
      "name": "quality-review-plugin",
      "source": "./plugins/quality-review-plugin",
      "description": "Adds a /quality-review skill for quick code reviews"
    }
  ]
}
```

Step 5: Tambahkan dan pasang

Tambahkan marketplace dan pasang plugin.

```
/plugin marketplace add ./my-marketplace
/plugin install quality-review-plugin@my-plugins
```

Step 6: Coba

Pilih beberapa kode di editor Anda dan jalankan perintah baru Anda.

```
/review
```

Untuk mempelajari lebih lanjut tentang apa yang dapat dilakukan plugin, termasuk hooks, agents, MCP servers, dan LSP servers, lihat Plugins.

> **Note:**
>
> **Cara plugin dipasang**: Ketika pengguna memasang plugin, Claude Code menyalin direktori plugin ke lokasi cache. Ini berarti plugin tidak dapat mereferensikan file di luar direktorinya menggunakan jalur seperti `../shared-utils`, karena file tersebut tidak akan disalin.
>
> Jika Anda perlu berbagi file di seluruh plugin, gunakan symlink (yang diikuti selama penyalinan). Lihat Plugin caching and file resolution untuk detail.

Buat file marketplace

Buat `.claude-plugin/marketplace.json` di root repositori Anda. File ini mendefinisikan nama marketplace Anda, informasi pemilik, dan daftar plugin dengan sumbernya.

Setiap entri plugin memerlukan minimal `name` dan `source` (di mana mengambilnya). Lihat skema lengkap di bawah untuk semua field yang tersedia.

```
{
  "name": "company-tools",
  "owner": {
    "name": "DevTools Team",
    "email": "devtools@example.com"
  },
  "plugins": [
    {
      "name": "code-formatter",
      "source": "./plugins/formatter",
      "description": "Automatic code formatting on save",
      "version": "2.1.0",
      "author": {
        "name": "DevTools Team"
      }
    },
    {
      "name": "deployment-tools",
      "source": {
        "source": "github",
        "repo": "company/deploy-plugin"
      },
      "description": "Deployment automation tools"
    }
  ]
}
```

Skema marketplace

Field yang diperlukan

Field	Type	Deskripsi	Contoh
`name`	string	Identifier marketplace (kebab-case, tanpa spasi). Ini menghadap publik: pengguna melihatnya saat memasang plugin (misalnya, `/plugin install my-tool@your-marketplace`).	`"acme-tools"`
`owner`	object	Informasi pengelola marketplace (lihat field di bawah)	
`plugins`	array	Daftar plugin yang tersedia	Lihat di bawah

> **Note:**
>
> **Nama yang dicadangkan**: Nama marketplace berikut dicadangkan untuk penggunaan resmi Anthropic dan tidak dapat digunakan oleh marketplace pihak ketiga: `claude-code-marketplace`, `claude-code-plugins`, `claude-plugins-official`, `anthropic-marketplace`, `anthropic-plugins`, `agent-skills`, `life-sciences`. Nama yang meniru marketplace resmi (seperti `official-claude-plugins` atau `anthropic-tools-v2`) juga diblokir.

Field pemilik

Field	Type	Diperlukan	Deskripsi
`name`	string	Ya	Nama pengelola atau tim
`email`	string	Tidak	Email kontak untuk pengelola

Metadata opsional

Field	Type	Deskripsi
`metadata.description`	string	Deskripsi marketplace singkat
`metadata.version`	string	Versi marketplace
`metadata.pluginRoot`	string	Direktori dasar yang ditambahkan ke jalur sumber plugin relatif (misalnya, `"./plugins"` memungkinkan Anda menulis `"source": "formatter"` alih-alih `"source": "./plugins/formatter"`)

Entri plugin

Setiap entri plugin dalam array `plugins` mendeskripsikan plugin dan di mana menemukannya. Anda dapat menyertakan field apa pun dari skema manifest plugin (seperti `description`, `version`, `author`, `commands`, `hooks`, dll.), ditambah field khusus marketplace ini: `source`, `category`, `tags`, dan `strict`.

Field yang diperlukan

Field	Type	Deskripsi
`name`	string	Identifier plugin (kebab-case, tanpa spasi). Ini menghadap publik: pengguna melihatnya saat memasang (misalnya, `/plugin install my-plugin@marketplace`).
`source`	string|object	Di mana mengambil plugin (lihat Plugin sources di bawah)

Field plugin opsional

Field metadata standar:

Field	Type	Deskripsi
`description`	string	Deskripsi plugin singkat
`version`	string	Versi plugin
`author`	object	Informasi penulis plugin (`name` diperlukan, `email` opsional)
`homepage`	string	URL homepage atau dokumentasi plugin
`repository`	string	URL repositori kode sumber
`license`	string	Identifier lisensi SPDX (misalnya, MIT, Apache-2.0)
`keywords`	array	Tag untuk penemuan dan kategorisasi plugin
`category`	string	Kategori plugin untuk organisasi
`tags`	array	Tag untuk kemudahan pencarian
`strict`	boolean	Mengontrol apakah `plugin.json` adalah otoritas untuk definisi komponen (default: true). Lihat Strict mode di bawah.

Field konfigurasi komponen:

Field	Type	Deskripsi
`commands`	string|array	Jalur kustom ke file atau direktori command
`agents`	string|array	Jalur kustom ke file agent
`hooks`	string|object	Konfigurasi hooks kustom atau jalur ke file hooks

Field	Type	Deskripsi
`mcpServers`	string|object	Konfigurasi MCP server atau jalur ke config MCP
`lspServers`	string|object	Konfigurasi LSP server atau jalur ke config LSP

Plugin sources

Plugin sources memberitahu Claude Code di mana mengambil setiap plugin individual yang tercantum di marketplace Anda. Ini diatur dalam field `source` dari setiap entri plugin di `marketplace.json`.

Setelah plugin diklon atau disalin ke mesin lokal, plugin disalin ke cache plugin lokal yang diversi di `~/.claude/plugins/cache`.

Source	Type	Fields	Catatan
Relative path	`string` (misalnya `"./my-plugin"`)	—	Direktori lokal dalam repo marketplace. Harus dimulai dengan `./`
`github`	object	`repo`, `ref?`, `sha?`	
`url`	object	`url` (harus berakhir .git), `ref?`, `sha?`	Sumber URL Git
`git-subdir`	object	`url`, `path`, `ref?`, `sha?`	Subdirektori dalam repo git. Mengklon secara sparse untuk meminimalkan bandwidth untuk monorepo
`npm`	object	`package`, `version?`, `registry?`	Dipasang via `npm install`
`pip`	object	`package`, `version?`, `registry?`	Dipasang via pip

Note:

Marketplace sources vs plugin sources: Ini adalah konsep berbeda yang mengontrol hal berbeda.

- **Marketplace source** — di mana mengambil katalog `marketplace.json` itu sendiri. Diatur ketika pengguna menjalankan `/plugin marketplace add` atau dalam pengaturan `extraKnownMarketplaces`. Mendukung `ref` (branch/tag) tetapi bukan `sha`.
- **Plugin source** — di mana mengambil plugin individual yang tercantum di marketplace. Diatur dalam field `source` dari setiap entri plugin di dalam `marketplace.json`. Mendukung baik `ref` (branch/tag) maupun `sha` (commit yang tepat).

Misalnya, marketplace yang dihosting di `acme-corp/plugin-catalog` (marketplace source) dapat mencantumkan plugin yang diambil dari `acme-corp/code-formatter` (plugin source). Marketplace source dan plugin source menunjuk ke repositori berbeda dan disematkan secara independen.

Jalur relatif

Untuk plugin di repositori yang sama:

```
{
  "name": "my-plugin",
  "source": "./plugins/my-plugin"
}
```

Note:

Jalur relatif hanya berfungsi ketika pengguna menambahkan marketplace Anda melalui Git (GitHub, GitLab, atau URL git). Jika pengguna menambahkan marketplace Anda melalui URL langsung ke file `marketplace.json`, jalur relatif tidak akan terselesaikan dengan benar. Untuk distribusi berbasis URL, gunakan sumber GitHub, npm, atau URL git sebagai gantinya. Lihat Troubleshooting untuk detail.

Repositori GitHub

```
{
  "name": "github-plugin",
  "source": {
    "source": "github",
    "repo": "owner/plugin-repo"
  }
}
```

Anda dapat menyematkan ke branch, tag, atau commit tertentu:

```
{
  "name": "github-plugin",
  "source": {
    "source": "github",
    "repo": "owner/plugin-repo",
    "ref": "v2.0.0",
    "sha": "a1b2c3d4e5f6a7b8c9d0e1f2a3b4c5d6e7f8a9b0"
  }
}
```

Field	Type	Deskripsi
`repo`	string	Diperlukan. Repositori GitHub dalam format `owner/repo`
`ref`	string	Opsional. Branch atau tag Git (default ke branch default repositori)
`sha`	string	Opsional. SHA commit git 40-karakter penuh untuk menyematkan ke versi yang tepat

Repositori Git

```
{
  "name": "git-plugin",
  "source": {
    "source": "url",
    "url": "https://gitlab.com/team/plugin.git"
  }
}
```

Anda dapat menyematkan ke branch, tag, atau commit tertentu:

```
{
  "name": "git-plugin",
  "source": {
    "source": "url",
    "url": "https://gitlab.com/team/plugin.git",
    "ref": "main",
    "sha": "a1b2c3d4e5f6a7b8c9d0e1f2a3b4c5d6e7f8a9b0"
  }
}
```

Field	Type	Deskripsi
`url`	string	Diperlukan. URL repositori git lengkap (harus berakhir dengan `.git`)
`ref`	string	Opsional. Branch atau tag Git (default ke branch default repositori)
`sha`	string	Opsional. SHA commit git 40-karakter penuh untuk menyematkan ke versi yang tepat

Subdirektori Git

Gunakan `git-subdir` untuk menunjuk ke plugin yang berada di dalam subdirektori repositori git. Claude Code menggunakan klon parsial dan sparse untuk mengambil hanya subdirektori, meminimalkan bandwidth untuk monorepo besar.

```
{
  "name": "my-plugin",
  "source": {
    "source": "git-subdir",
    "url": "https://github.com/acme-corp/monorepo.git",
    "path": "tools/claude-plugin"
  }
}
```

Anda dapat menyematkan ke branch, tag, atau commit tertentu:

```
{
  "name": "my-plugin",
  "source": {
    "source": "git-subdir",
    "url": "https://github.com/acme-corp/monorepo.git",
    "path": "tools/claude-plugin",
    "ref": "v2.0.0",
    "sha": "a1b2c3d4e5f6a7b8c9d0e1f2a3b4c5d6e7f8a9b0"
  }
}
```

Field `url` juga menerima shorthand GitHub (`owner/repo`) atau URL SSH (`git@github.com:owner/repo.git`).

Field	Type	Deskripsi
`url`	string	Diperlukan. URL repositori Git, shorthand GitHub `owner/repo` , atau URL SSH

Field	Type	Deskripsi
`path`	string	Diperlukan. Jalur subdirektori dalam repo yang berisi plugin (misalnya, `"tools/claude-plugin"`)
`ref`	string	Opsional. Branch atau tag Git (default ke branch default repositori)
`sha`	string	Opsional. SHA commit git 40-karakter penuh untuk menyematkan ke versi yang tepat

Paket npm

Plugin yang didistribusikan sebagai paket npm dipasang menggunakan `npm install`. Ini berfungsi dengan paket apa pun di registry npm publik atau registry pribadi yang dihosting tim Anda.

```
{
  "name": "my-npm-plugin",
  "source": {
    "source": "npm",
    "package": "@acme/claude-plugin"
  }
}
```

Untuk menyematkan ke versi tertentu, tambahkan field `version`:

```
{
  "name": "my-npm-plugin",
  "source": {
    "source": "npm",
    "package": "@acme/claude-plugin",
    "version": "2.1.0"
  }
}
```

Untuk memasang dari registry pribadi atau internal, tambahkan field `registry`:

```
{
  "name": "my-npm-plugin",
  "source": {
    "source": "npm",
    "package": "@acme/claude-plugin",
    "version": "^2.0.0",
    "registry": "https://npm.example.com"
  }
}
```

Field	Type	Deskripsi
`package`	string	Diperlukan. Nama paket atau paket scoped (misalnya, `@org/plugin`)
`version`	string	Opsional. Versi atau rentang versi (misalnya, `2.1.0`, `^2.0.0`, `~1.5.0`)
`registry`	string	Opsional. URL registry npm kustom. Default ke registry npm sistem (biasanya npmjs.org)

Entri plugin lanjutan

Contoh ini menunjukkan entri plugin menggunakan banyak field opsional, termasuk jalur kustom untuk commands, agents, hooks, dan MCP servers:

```
{
  "name": "enterprise-tools",
  "source": {
    "source": "github",
    "repo": "company/enterprise-plugin"
  },
  "description": "Enterprise workflow automation tools",
  "version": "2.1.0",
  "author": {
    "name": "Enterprise Team",
    "email": "enterprise@example.com"
  },
  "homepage": "https://docs.example.com/plugins/enterprise-tools",
  "repository": "https://github.com/company/enterprise-plugin",
  "license": "MIT",
  "keywords": ["enterprise", "workflow", "automation"],
  "category": "productivity",
  "commands": [
    "./commands/core/",
    "./commands/enterprise/",
    "./commands/experimental/preview.md"
  ],
  "agents": ["./agents/security-reviewer.md", "./agents/compliance-checker.md"],
  "hooks": {
    "PostToolUse": [
      {
        "matcher": "Write|Edit",
        "hooks": [
          {
            "type": "command",
            "command": "${CLAUDE_PLUGIN_ROOT}/scripts/validate.sh"
          }
        ]
      }
    ]
  },
  "mcpServers": {
    "enterprise-db": {
```

```
      "command": "${CLAUDE_PLUGIN_ROOT}/servers/db-server",
      "args": ["--config", "${CLAUDE_PLUGIN_ROOT}/config.json"]
    }
  },
  "strict": false
}
```

Hal-hal penting untuk diperhatikan:

- `commands` dan `agents` : Anda dapat menentukan beberapa direktori atau file individual. Jalur relatif terhadap root plugin.
- `${CLAUDE_PLUGIN_ROOT}` : Gunakan variabel ini dalam hooks dan config MCP server untuk mereferensikan file dalam direktori instalasi plugin. Ini diperlukan karena plugin disalin ke lokasi cache saat dipasang.
- `strict: false` : Karena ini diatur ke false, plugin tidak memerlukan `plugin.json` sendiri. Entri marketplace mendefinisikan semuanya. Lihat Strict mode di bawah.

Strict mode

Field `strict` mengontrol apakah `plugin.json` adalah otoritas untuk definisi komponen (commands, agents, hooks, skills, MCP servers, output styles).

Value	Perilaku
`true` (default)	`plugin.json` adalah otoritas. Entri marketplace dapat melengkapinya dengan komponen tambahan, dan kedua sumber digabungkan.
`false`	Entri marketplace adalah definisi lengkap. Jika plugin juga memiliki `plugin.json` yang mendeklarasikan komponen, itu adalah konflik dan plugin gagal dimuat.

Kapan menggunakan setiap mode:

- `strict: true` : plugin memiliki `plugin.json` sendiri dan mengelola komponennya sendiri. Entri marketplace dapat menambahkan commands atau hooks tambahan di atas. Ini adalah default dan berfungsi untuk sebagian besar plugin.
- `strict: false` : operator marketplace menginginkan kontrol penuh. Repo plugin menyediakan file mentah, dan entri marketplace mendefinisikan file mana yang diekspos sebagai commands, agents, hooks, dll. Berguna ketika marketplace merestruktur atau mengkurasi komponen plugin secara berbeda dari yang dimaksudkan penulis plugin.

Host dan distribusikan marketplace

Host di GitHub (direkomendasikan)

GitHub menyediakan metode distribusi paling mudah:

1. **Buat repositori**: Siapkan repositori baru untuk marketplace Anda
2. **Tambahkan file marketplace**: Buat `.claude-plugin/marketplace.json` dengan definisi plugin Anda
3. **Bagikan dengan tim**: Pengguna menambahkan marketplace Anda dengan `/plugin marketplace add owner/repo`

Manfaat: Kontrol versi bawaan, pelacakan masalah, dan fitur kolaborasi tim.

Host di layanan git lainnya

Layanan hosting git apa pun berfungsi, seperti GitLab, Bitbucket, dan server yang dihosting sendiri. Pengguna menambahkan dengan URL repositori lengkap:

```
/plugin marketplace add https://gitlab.com/company/plugins.git
```

Repositori pribadi

Claude Code mendukung pemasangan plugin dari repositori pribadi. Untuk instalasi manual dan pembaruan, Claude Code menggunakan helper kredensial git yang ada. Jika `git clone` berfungsi untuk repositori pribadi di terminal Anda, itu berfungsi di Claude Code juga. Helper kredensial umum termasuk `gh auth login` untuk GitHub, Keychain macOS, dan `git-credential-store`.

Pembaruan otomatis latar belakang berjalan saat startup tanpa helper kredensial, karena prompt interaktif akan memblokir Claude Code dari startup. Untuk mengaktifkan pembaruan otomatis untuk marketplace pribadi, atur token autentikasi yang sesuai di lingkungan Anda:

Provider	Variabel lingkungan	Catatan
GitHub	`GITHUB_TOKEN` atau `GH_TOKEN`	Token akses pribadi atau token GitHub App
GitLab	`GITLAB_TOKEN` atau `GL_TOKEN`	Token akses pribadi atau token proyek
Bitbucket	`BITBUCKET_TOKEN`	Sandi aplikasi atau token akses repositori

Atur token dalam konfigurasi shell Anda (misalnya, `.bashrc`, `.zshrc`) atau teruskan saat menjalankan Claude Code:

```
export GITHUB_TOKEN=ghp_xxxxxxxxxxxxxxxxxxxx
```

> **Note:**
>
> Untuk lingkungan CI/CD, konfigurasikan token sebagai variabel lingkungan rahasia. GitHub Actions secara otomatis menyediakan `GITHUB_TOKEN` untuk repositori dalam organisasi yang sama.

Uji secara lokal sebelum distribusi

Uji marketplace Anda secara lokal sebelum berbagi:

```
/plugin marketplace add ./my-local-marketplace
/plugin install test-plugin@my-local-marketplace
```

Untuk rangkaian lengkap perintah add (GitHub, URL Git, jalur lokal, URL jarak jauh), lihat Tambahkan marketplace.

Wajibkan marketplace untuk tim Anda

Anda dapat mengonfigurasi repositori Anda sehingga anggota tim secara otomatis diminta untuk memasang marketplace Anda ketika mereka mempercayai folder proyek. Tambahkan marketplace Anda ke `.claude/settings.json`:

```
{
  "extraKnownMarketplaces": {
    "company-tools": {
      "source": {
        "source": "github",
        "repo": "your-org/claude-plugins"
      }
    }
  }
}
```

Anda juga dapat menentukan plugin mana yang harus diaktifkan secara default:

```
{
  "enabledPlugins": {
    "code-formatter@company-tools": true,
    "deployment-tools@company-tools": true
  }
}
```

Untuk opsi konfigurasi lengkap, lihat Plugin settings.

Pembatasan marketplace yang dikelola

Untuk organisasi yang memerlukan kontrol ketat atas sumber plugin, administrator dapat membatasi marketplace plugin mana yang diizinkan pengguna untuk tambahkan menggunakan pengaturan `strictKnownMarketplaces` dalam pengaturan yang dikelola.

Ketika `strictKnownMarketplaces` dikonfigurasi dalam pengaturan yang dikelola, perilaku pembatasan tergantung pada nilainya:

Value	Perilaku
Tidak terdefinisi (default)	Tidak ada pembatasan. Pengguna dapat menambahkan marketplace apa pun
Array kosong `[]`	Lockdown lengkap. Pengguna tidak dapat menambahkan marketplace baru apa pun
Daftar sumber	Pengguna hanya dapat menambahkan marketplace yang cocok dengan daftar izin secara tepat

Konfigurasi umum

Nonaktifkan semua penambahan marketplace:

```
{
  "strictKnownMarketplaces": []
}
```

Izinkan marketplace tertentu saja:

```
{
  "strictKnownMarketplaces": [
    {
      "source": "github",
      "repo": "acme-corp/approved-plugins"
    },
    {
      "source": "github",
      "repo": "acme-corp/security-tools",
      "ref": "v2.0"
    },
    {
      "source": "url",
      "url": "https://plugins.example.com/marketplace.json"
    }
  ]
}
```

Izinkan semua marketplace dari server git internal menggunakan pencocokan pola regex pada host:

```
{
  "strictKnownMarketplaces": [
    {
      "source": "hostPattern",
      "hostPattern": "^github\\.example\\.com$"
    }
  ]
}
```

Izinkan marketplace berbasis filesystem dari direktori tertentu menggunakan pencocokan pola regex pada jalur:

```
{
  "strictKnownMarketplaces": [
    {
      "source": "pathPattern",
      "pathPattern": "^/opt/approved/"
    }
  ]
}
```

Gunakan `".*"` sebagai `pathPattern` untuk mengizinkan jalur filesystem apa pun sambil tetap mengontrol sumber jaringan dengan `hostPattern`.

Cara pembatasan bekerja

Pembatasan divalidasi awal dalam proses instalasi plugin, sebelum permintaan jaringan atau operasi filesystem apa pun terjadi. Ini mencegah upaya akses marketplace yang tidak sah.

Daftar izin menggunakan pencocokan tepat untuk sebagian besar jenis sumber. Agar marketplace diizinkan, semua field yang ditentukan harus cocok secara tepat:

- Untuk sumber GitHub: `repo` diperlukan, dan `ref` atau `path` juga harus cocok jika ditentukan dalam daftar izin
- Untuk sumber URL: URL lengkap harus cocok secara tepat
- Untuk sumber `hostPattern`: host marketplace dicocokkan dengan pola regex
- Untuk sumber `pathPattern`: jalur filesystem marketplace dicocokkan dengan pola regex

Karena `strictKnownMarketplaces` diatur dalam pengaturan yang dikelola, konfigurasi pengguna individual dan proyek tidak dapat mengganti pembatasan ini.

Untuk detail konfigurasi lengkap termasuk semua jenis sumber yang didukung dan perbandingan dengan `extraKnownMarketplaces`, lihat referensi strictKnownMarketplaces.

Resolusi versi dan saluran rilis

Versi plugin menentukan jalur cache dan deteksi pembaruan. Anda dapat menentukan versi dalam manifest plugin (`plugin.json`) atau dalam entri marketplace (`marketplace.json`).

> **Warning:**

> Jika memungkinkan, hindari menetapkan versi di kedua tempat. Manifest plugin selalu menang secara diam-diam, yang dapat menyebabkan versi marketplace diabaikan. Untuk plugin jalur relatif, atur versi dalam entri marketplace. Untuk semua sumber plugin lainnya, atur dalam manifest plugin.

Siapkan saluran rilis

Untuk mendukung saluran rilis “stable” dan “latest” untuk plugin Anda, Anda dapat menyiapkan dua marketplace yang menunjuk ke refs atau SHA berbeda dari repo yang sama. Anda kemudian dapat menetapkan dua marketplace ke grup pengguna berbeda melalui pengaturan yang dikelola.

> **Warning:**
>
> `plugin.json` plugin harus mendeklarasikan `version` berbeda di setiap ref atau commit yang disematkan. Jika dua refs atau commits memiliki versi manifest yang sama, Claude Code memperlakukannya sebagai identik dan melewati pembaruan.

Contoh

```
{
  "name": "stable-tools",
  "plugins": [
    {
      "name": "code-formatter",
      "source": {
        "source": "github",
        "repo": "acme-corp/code-formatter",
        "ref": "stable"
      }
    }
  ]
}
```

```
{
  "name": "latest-tools",
  "plugins": [
    {
      "name": "code-formatter",
      "source": {
        "source": "github",
        "repo": "acme-corp/code-formatter",
        "ref": "latest"
      }
    }
  ]
}
```

Tetapkan saluran ke grup pengguna

Tetapkan setiap marketplace ke grup pengguna yang sesuai melalui pengaturan yang dikelola. Misalnya, grup stabil menerima:

```
{
  "extraKnownMarketplaces": {
    "stable-tools": {
      "source": {
        "source": "github",
        "repo": "acme-corp/stable-tools"
      }
    }
  }
}
```

Grup early-access menerima `latest-tools` sebagai gantinya:

```
{
  "extraKnownMarketplaces": {
    "latest-tools": {
      "source": {
        "source": "github",
        "repo": "acme-corp/latest-tools"
      }
    }
  }
}
```

Validasi dan pengujian

Uji marketplace Anda sebelum berbagi.

Validasi sintaks JSON marketplace Anda:

```
claude plugin validate .
```

Atau dari dalam Claude Code:

```
/plugin validate .
```

Tambahkan marketplace untuk pengujian:

```
/plugin marketplace add ./path/to/marketplace
```

Pasang plugin uji untuk memverifikasi semuanya berfungsi:

```
/plugin install test-plugin@marketplace-name
```

Untuk alur kerja pengujian plugin lengkap, lihat Uji plugin Anda secara lokal. Untuk troubleshooting teknis, lihat Plugins reference.

Troubleshooting

Marketplace tidak memuat

Gejala: Tidak dapat menambahkan marketplace atau melihat plugin darinya

Solusi:

- Verifikasi URL marketplace dapat diakses
- Periksa bahwa `.claude-plugin/marketplace.json` ada di jalur yang ditentukan
- Pastikan sintaks JSON valid menggunakan `claude plugin validate` atau `/plugin validate`
- Untuk repositori pribadi, konfirmasi Anda memiliki izin akses

Kesalahan validasi marketplace

Jalankan `claude plugin validate .` atau `/plugin validate .` dari direktori marketplace Anda untuk memeriksa masalah. Kesalahan umum:

Kesalahan	Penyebab	Solusi
`File not found: .claude-plugin/marketplace.json`	Manifest hilang	Buat `.claude-plugin/marketplace.json` dengan field yang diperlukan
`Invalid JSON syntax: Unexpected token...`	Kesalahan sintaks JSON	Periksa koma yang hilang, koma ekstra, atau string yang tidak dikutip
`Duplicate plugin name "x" found in marketplace`	Dua plugin berbagi nama yang sama	Berikan setiap plugin nilai `name` yang unik
`plugins[0].source: Path traversal not allowed`	Jalur sumber berisi `..`	Gunakan jalur relatif terhadap root marketplace tanpa `..`

Peringatan (non-blocking):

- `Marketplace has no plugins defined`: tambahkan setidaknya satu plugin ke array `plugins`
- `No marketplace description provided`: tambahkan `metadata.description` untuk membantu pengguna memahami marketplace Anda

Kegagalan instalasi plugin

Gejala: Marketplace muncul tetapi instalasi plugin gagal

Solusi:

- Verifikasi URL sumber plugin dapat diakses
- Periksa bahwa direktori plugin berisi file yang diperlukan
- Untuk sumber GitHub, pastikan repositori publik atau Anda memiliki akses
- Uji sumber plugin secara manual dengan mengklon/mengunduh

Autentikasi repositori pribadi gagal

Gejala: Kesalahan autentikasi saat memasang plugin dari repositori pribadi

Solusi:

Untuk instalasi manual dan pembaruan:

- Verifikasi Anda diautentikasi dengan penyedia git Anda (misalnya, jalankan `gh auth status` untuk GitHub)
- Periksa bahwa helper kredensial Anda dikonfigurasi dengan benar: `git config --global credential.helper`
- Coba klon repositori secara manual untuk memverifikasi kredensial Anda berfungsi

Untuk pembaruan otomatis latar belakang:

- Atur token yang sesuai di lingkungan Anda: `echo $GITHUB_TOKEN`
- Periksa bahwa token memiliki izin yang diperlukan (akses baca ke repositori)
- Untuk GitHub, pastikan token memiliki scope `repo` untuk repositori pribadi
- Untuk GitLab, pastikan token memiliki setidaknya scope `read_repository`
- Verifikasi token belum kedaluwarsa

Operasi Git habis waktu

Gejala: Instalasi plugin atau pembaruan marketplace gagal dengan kesalahan timeout seperti “Git clone timed out after 120s” atau “Git pull timed out after 120s”.

Penyebab: Claude Code menggunakan timeout 120 detik untuk semua operasi git, termasuk mengklon repositori plugin dan menarik pembaruan marketplace. Repositori besar atau koneksi jaringan lambat mungkin melebihi batas ini.

Solusi: Tingkatkan timeout menggunakan variabel lingkungan `CLAUDE_CODE_PLUGIN_GIT_TIMEOUT_MS`. Nilainya dalam milidetik:

```
export CLAUDE_CODE_PLUGIN_GIT_TIMEOUT_MS=300000  # 5 minutes
```

Plugin dengan jalur relatif gagal di marketplace berbasis URL

Gejala: Menambahkan marketplace melalui URL (seperti `https://example.com/marketplace.json`), tetapi plugin dengan sumber jalur relatif seperti `"./plugins/my-plugin"` gagal dipasang dengan kesalahan "path not found".

Penyebab: Marketplace berbasis URL hanya mengunduh file `marketplace.json` itu sendiri. Mereka tidak mengunduh file plugin dari server. Jalur relatif dalam entri marketplace mereferensikan file di server jarak jauh yang tidak diunduh.

Solusi:

- **Gunakan sumber eksternal**: Ubah entri plugin untuk menggunakan sumber GitHub, npm, atau URL git alih-alih jalur relatif:

```
{ "name": "my-plugin", "source": { "source": "github", "repo": "owner/repo" } }
```

- **Gunakan marketplace berbasis Git**: Host marketplace Anda di repositori Git dan tambahkan dengan URL git. Marketplace berbasis Git mengklon seluruh repositori, membuat jalur relatif berfungsi dengan benar.

File tidak ditemukan setelah instalasi

Gejala: Plugin dipasang tetapi referensi ke file gagal, terutama file di luar direktori plugin

Penyebab: Plugin disalin ke direktori cache daripada digunakan di tempat. Jalur yang mereferensikan file di luar direktori plugin (seperti `../shared-utils`) tidak akan berfungsi karena file tersebut tidak disalin.

Solusi: Lihat Plugin caching and file resolution untuk solusi termasuk symlink dan restruktur direktori.

Untuk alat debugging tambahan dan masalah umum, lihat Debugging and development tools.

Lihat juga

- Temukan dan pasang plugin yang sudah dibuat - Memasang plugin dari marketplace yang ada
- Plugins - Membuat plugin Anda sendiri

- Plugins reference - Spesifikasi teknis lengkap dan skema
- Plugin settings - Opsi konfigurasi plugin
- strictKnownMarketplaces reference - Pembatasan marketplace yang dikelola

Temukan dan instal plugin yang sudah dibuat melalui marketplace

> *Temukan dan instal plugin dari marketplace untuk memperluas Claude Code dengan perintah, agen, dan kemampuan baru.*

Plugin memperluas Claude Code dengan skills, agen, hooks, dan MCP servers. Plugin marketplace adalah katalog yang membantu Anda menemukan dan menginstal ekstensi ini tanpa membuatnya sendiri.

Mencari cara membuat dan mendistribusikan marketplace Anda sendiri? Lihat Buat dan distribusikan plugin marketplace.

Cara kerja marketplace

Marketplace adalah katalog plugin yang telah dibuat dan dibagikan oleh orang lain. Menggunakan marketplace adalah proses dua langkah:

Step 1: Tambahkan marketplace

Ini mendaftarkan katalog dengan Claude Code sehingga Anda dapat menjelajahi apa yang tersedia. Tidak ada plugin yang diinstal lagi.

Step 2: Instal plugin individual

Jelajahi katalog dan instal plugin yang Anda inginkan.

Anggap saja seperti menambahkan app store: menambahkan toko memberi Anda akses untuk menjelajahi koleksinya, tetapi Anda masih memilih aplikasi mana yang akan diunduh secara individual.

Official Anthropic marketplace

Official Anthropic marketplace (`claude-plugins-official`) secara otomatis tersedia saat Anda memulai Claude Code. Jalankan `/plugin` dan buka tab **Discover** untuk menjelajahi apa yang tersedia.

Untuk menginstal plugin dari official marketplace:

```
/plugin install plugin-name@claude-plugins-official
```

> **Note:**
>
> Official marketplace dikelola oleh Anthropic. Untuk mengirimkan plugin ke official marketplace, gunakan salah satu formulir pengajuan dalam aplikasi:
>
> - **Claude.ai**: claude.ai/settings/plugins/submit
> - **Console**: platform.claude.com/plugins/submit
>
> Untuk mendistribusikan plugin secara independen, buat marketplace Anda sendiri dan bagikan dengan pengguna.

Official marketplace mencakup beberapa kategori plugin:

Code intelligence

Plugin code intelligence mengaktifkan alat LSP bawaan Claude Code, memberikan Claude kemampuan untuk melompat ke definisi, menemukan referensi, dan melihat kesalahan tipe segera setelah edit. Plugin ini mengonfigurasi koneksi Language Server Protocol, teknologi yang sama yang mendukung code intelligence VS Code.

Plugin ini memerlukan binary language server untuk diinstal di sistem Anda. Jika Anda sudah memiliki language server yang diinstal, Claude mungkin akan meminta Anda untuk menginstal plugin yang sesuai saat Anda membuka proyek.

Language	Plugin	Binary required
C/C++	`clangd-lsp`	`clangd`
C#	`csharp-lsp`	`csharp-ls`
Go	`gopls-lsp`	`gopls`
Java	`jdtls-lsp`	`jdtls`
Kotlin	`kotlin-lsp`	`kotlin-language-server`
Lua	`lua-lsp`	`lua-language-server`
PHP	`php-lsp`	`intelephense`
Python	`pyright-lsp`	`pyright-langserver`
Rust	`rust-analyzer-lsp`	`rust-analyzer`

Language	Plugin	Binary required
Swift	`swift-lsp`	`sourcekit-lsp`
TypeScript	`typescript-lsp`	`typescript-language-server`

Anda juga dapat membuat plugin LSP Anda sendiri untuk bahasa lain.

> **Note:**
>
> Jika Anda melihat `Executable not found in $PATH` di tab `/plugin` Errors setelah menginstal plugin, instal binary yang diperlukan dari tabel di atas.

Apa yang Claude dapatkan dari plugin code intelligence

Setelah plugin code intelligence diinstal dan binary language server-nya tersedia, Claude mendapatkan dua kemampuan:

- **Automatic diagnostics**: setelah setiap edit file yang dilakukan Claude, language server menganalisis perubahan dan melaporkan kesalahan dan peringatan secara otomatis. Claude melihat kesalahan tipe, impor yang hilang, dan masalah sintaks tanpa perlu menjalankan compiler atau linter. Jika Claude memperkenalkan kesalahan, itu akan menyadari dan memperbaiki masalah dalam giliran yang sama. Ini tidak memerlukan konfigurasi apa pun selain menginstal plugin. Anda dapat melihat diagnostik secara inline dengan menekan **Ctrl+O** saat indikator "diagnostics found" muncul.
- **Code navigation**: Claude dapat menggunakan language server untuk melompat ke definisi, menemukan referensi, mendapatkan informasi tipe saat hover, membuat daftar simbol, menemukan implementasi, dan melacak hierarki panggilan. Operasi ini memberikan Claude navigasi yang lebih presisi daripada pencarian berbasis grep, meskipun ketersediaan mungkin berbeda menurut bahasa dan lingkungan.

Jika Anda mengalami masalah, lihat Code intelligence troubleshooting.

External integrations

Plugin ini menggabungkan MCP servers yang sudah dikonfigurasi sebelumnya sehingga Anda dapat menghubungkan Claude ke layanan eksternal tanpa setup manual:

- **Source control**: `github`, `gitlab`
- **Project management**: `atlassian` (Jira/Confluence), `asana`, `linear`, `notion`
- **Design**: `figma`
- **Infrastructure**: `vercel`, `firebase`, `supabase`

- **Communication**: `slack`
- **Monitoring**: `sentry`

Development workflows

Plugin yang menambahkan perintah dan agen untuk tugas pengembangan umum:

- **commit-commands**: Git commit workflows termasuk commit, push, dan pembuatan PR
- **pr-review-toolkit**: Agen khusus untuk meninjau pull request
- **agent-sdk-dev**: Tools untuk membangun dengan Claude Agent SDK
- **plugin-dev**: Toolkit untuk membuat plugin Anda sendiri

Output styles

Sesuaikan cara Claude merespons:

- **explanatory-output-style**: Wawasan edukatif tentang pilihan implementasi
- **learning-output-style**: Mode pembelajaran interaktif untuk membangun skill

Coba: tambahkan demo marketplace

Anthropic juga memelihara demo plugins marketplace (`claude-code-plugins`) dengan plugin contoh yang menunjukkan apa yang mungkin dengan sistem plugin. Tidak seperti official marketplace, Anda perlu menambahkan ini secara manual.

Step 1: Tambahkan marketplace

Dari dalam Claude Code, jalankan perintah `plugin marketplace add` untuk marketplace `anthropics/claude-code` :

```
/plugin marketplace add anthropics/claude-code
```

Ini mengunduh katalog marketplace dan membuat plugin-nya tersedia untuk Anda.

Step 2: Jelajahi plugin yang tersedia

Jalankan `/plugin` untuk membuka plugin manager. Ini membuka antarmuka bertab dengan empat tab yang dapat Anda siklus menggunakan **Tab** (atau **Shift+Tab** untuk mundur):

- **Discover**: jelajahi plugin yang tersedia dari semua marketplace Anda
- **Installed**: lihat dan kelola plugin yang diinstal

- **Marketplaces**: tambah, hapus, atau perbarui marketplace yang ditambahkan
- **Errors**: lihat kesalahan pemuatan plugin apa pun

Buka tab **Discover** untuk melihat plugin dari marketplace yang baru saja Anda tambahkan.

Step 3: Instal plugin

Pilih plugin untuk melihat detailnya, kemudian pilih cakupan instalasi:

- **User scope**: instal untuk diri sendiri di semua proyek
- **Project scope**: instal untuk semua kolaborator di repositori ini
- **Local scope**: instal untuk diri sendiri di repositori ini saja

Misalnya, pilih **commit-commands** (plugin yang menambahkan perintah alur kerja git) dan instal ke cakupan pengguna Anda.

Anda juga dapat menginstal langsung dari baris perintah:

```
/plugin install commit-commands@anthropics-claude-code
```

Lihat Configuration scopes untuk mempelajari lebih lanjut tentang cakupan.

Step 4: Gunakan plugin baru Anda

Setelah menginstal, perintah plugin segera tersedia. Perintah plugin diberi namespace oleh nama plugin, jadi **commit-commands** menyediakan perintah seperti `/commit-commands:commit`.

Coba dengan membuat perubahan pada file dan menjalankan:

```
/commit-commands:commit
```

Ini menampilkan perubahan Anda, menghasilkan pesan commit, dan membuat commit.

Setiap plugin bekerja berbeda. Periksa deskripsi plugin di tab **Discover** atau homepagenya untuk mempelajari perintah dan kemampuan apa yang disediakan.

Sisa panduan ini mencakup semua cara Anda dapat menambahkan marketplace, menginstal plugin, dan mengelola konfigurasi Anda.

Tambahkan marketplace

Gunakan perintah `/plugin marketplace add` untuk menambahkan marketplace dari sumber yang berbeda.

> **Tip:**
>
> **Shortcuts**: Anda dapat menggunakan `/plugin market` sebagai ganti `/plugin marketplace`, dan `rm` sebagai ganti `remove`.

- **GitHub repositories**: format `owner/repo` (misalnya, `anthropics/claude-code`)
- **Git URLs**: URL repositori git apa pun (GitLab, Bitbucket, self-hosted)
- **Local paths**: direktori atau jalur langsung ke file `marketplace.json`
- **Remote URLs**: URL langsung ke file `marketplace.json` yang dihosting

Tambahkan dari GitHub

Tambahkan repositori GitHub yang berisi file `.claude-plugin/marketplace.json` menggunakan format `owner/repo` —di mana `owner` adalah nama pengguna atau organisasi GitHub dan `repo` adalah nama repositori.

Misalnya, `anthropics/claude-code` merujuk ke repositori `claude-code` yang dimiliki oleh `anthropics`:

```
/plugin marketplace add anthropics/claude-code
```

Tambahkan dari host Git lainnya

Tambahkan repositori git apa pun dengan memberikan URL lengkap. Ini bekerja dengan host Git apa pun, termasuk GitLab, Bitbucket, dan server self-hosted:

Menggunakan HTTPS:

```
/plugin marketplace add https://gitlab.com/company/plugins.git
```

Menggunakan SSH:

```
/plugin marketplace add git@gitlab.com:company/plugins.git
```

Untuk menambahkan cabang atau tag tertentu, tambahkan `#` diikuti oleh ref:

```
/plugin marketplace add https://gitlab.com/company/plugins.git#v1.0.0
```

Tambahkan dari jalur lokal

Tambahkan direktori lokal yang berisi file `.claude-plugin/marketplace.json`:

```
/plugin marketplace add ./my-marketplace
```

Anda juga dapat menambahkan jalur langsung ke file `marketplace.json`:

```
/plugin marketplace add ./path/to/marketplace.json
```

Tambahkan dari URL jarak jauh

Tambahkan file `marketplace.json` jarak jauh melalui URL:

```
/plugin marketplace add https://example.com/marketplace.json
```

> **Note:**
>
> Marketplace berbasis URL memiliki beberapa keterbatasan dibandingkan dengan marketplace berbasis Git. Jika Anda mengalami kesalahan "path not found" saat menginstal plugin, lihat Troubleshooting.

Instal plugin

Setelah Anda menambahkan marketplace, Anda dapat menginstal plugin secara langsung (menginstal ke cakupan pengguna secara default):

```
/plugin install plugin-name@marketplace-name
```

Untuk memilih cakupan instalasi yang berbeda, gunakan UI interaktif: jalankan `/plugin`, buka tab **Discover**, dan tekan **Enter** pada plugin. Anda akan melihat opsi untuk:

- **User scope** (default): instal untuk diri sendiri di semua proyek
- **Project scope**: instal untuk semua kolaborator di repositori ini (menambahkan ke `.claude/settings.json`)

- **Local scope**: instal untuk diri sendiri di repositori ini saja (tidak dibagikan dengan kolaborator)

Anda juga dapat melihat plugin dengan cakupan **managed**—ini diinstal oleh administrator melalui managed settings dan tidak dapat dimodifikasi.

Jalankan `/plugin` dan buka tab **Installed** untuk melihat plugin Anda dikelompokkan menurut cakupan.

> **Warning:**
>
> Pastikan Anda mempercayai plugin sebelum menginstalnya. Anthropic tidak mengontrol MCP servers, file, atau perangkat lunak lain apa yang disertakan dalam plugin dan tidak dapat memverifikasi bahwa mereka bekerja seperti yang dimaksudkan. Periksa homepage setiap plugin untuk informasi lebih lanjut.

Kelola plugin yang diinstal

Jalankan `/plugin` dan buka tab **Installed** untuk melihat, mengaktifkan, menonaktifkan, atau menghapus plugin Anda. Ketik untuk memfilter daftar berdasarkan nama atau deskripsi plugin.

Anda juga dapat mengelola plugin dengan perintah langsung.

Nonaktifkan plugin tanpa menghapusnya:

```
/plugin disable plugin-name@marketplace-name
```

Aktifkan kembali plugin yang dinonaktifkan:

```
/plugin enable plugin-name@marketplace-name
```

Hapus plugin sepenuhnya:

```
/plugin uninstall plugin-name@marketplace-name
```

Opsi `--scope` memungkinkan Anda menargetkan cakupan tertentu dengan perintah CLI:

```
claude plugin install formatter@your-org --scope project
claude plugin uninstall formatter@your-org --scope project
```

Terapkan perubahan plugin tanpa restart

Saat Anda menginstal, mengaktifkan, atau menonaktifkan plugin selama sesi, beberapa perubahan (seperti perintah dan hook baru) berlaku segera. Yang lain, termasuk pembaruan server LSP, memerlukan restart.

Untuk mengaktifkan semua perubahan plugin yang tertunda tanpa restart, jalankan:

```
/reload-plugins
```

Claude Code memuat ulang semua plugin aktif dan melaporkan apa yang dimuat. Jika ada server LSP yang ditambahkan atau diperbarui, itu akan memberi tahu Anda bahwa server tersebut memerlukan restart untuk berlaku.

Kelola marketplace

Anda dapat mengelola marketplace melalui antarmuka `/plugin` interaktif atau dengan perintah CLI.

Gunakan antarmuka interaktif

Jalankan `/plugin` dan buka tab **Marketplaces** untuk:

- Lihat semua marketplace yang ditambahkan dengan sumber dan statusnya
- Tambahkan marketplace baru
- Perbarui daftar marketplace untuk mengambil plugin terbaru
- Hapus marketplace yang tidak lagi Anda butuhkan

Gunakan perintah CLI

Anda juga dapat mengelola marketplace dengan perintah langsung.

Daftar semua marketplace yang dikonfigurasi:

```
/plugin marketplace list
```

Segarkan daftar plugin dari marketplace:

```
/plugin marketplace update marketplace-name
```

Hapus marketplace:

```
/plugin marketplace remove marketplace-name
```

> **Warning:**
>
> Menghapus marketplace akan menghapus instalasi plugin apa pun yang Anda instal darinya.

Konfigurasi auto-updates

Claude Code dapat secara otomatis memperbarui marketplace dan plugin yang diinstal saat startup. Saat auto-update diaktifkan untuk marketplace, Claude Code menyegarkan data marketplace dan memperbarui plugin yang diinstal ke versi terbaru mereka. Jika ada plugin yang diperbarui, Anda akan melihat notifikasi yang meminta Anda untuk menjalankan `/reload-plugins`.

Alihkan auto-update untuk marketplace individual melalui UI:

1. Jalankan `/plugin` untuk membuka plugin manager
2. Pilih **Marketplaces**
3. Pilih marketplace dari daftar
4. Pilih **Enable auto-update** atau **Disable auto-update**

Official Anthropic marketplace memiliki auto-update diaktifkan secara default. Marketplace pihak ketiga dan pengembangan lokal memiliki auto-update dinonaktifkan secara default.

Untuk menonaktifkan semua pembaruan otomatis sepenuhnya untuk Claude Code dan semua plugin, atur variabel lingkungan `DISABLE_AUTOUPDATER`. Lihat Auto updates untuk detail.

Untuk menjaga plugin auto-updates tetap diaktifkan sambil menonaktifkan Claude Code auto-updates, atur `FORCE_AUTOUPDATE_PLUGINS=true` bersama dengan `DISABLE_AUTOUPDATER`:

```
export DISABLE_AUTOUPDATER=true
export FORCE_AUTOUPDATE_PLUGINS=true
```

Ini berguna saat Anda ingin mengelola pembaruan Claude Code secara manual tetapi masih menerima pembaruan plugin otomatis.

Konfigurasi team marketplace

Admin tim dapat menyiapkan instalasi marketplace otomatis untuk proyek dengan menambahkan konfigurasi marketplace ke `.claude/settings.json`. Saat anggota tim mempercayai folder repositori, Claude Code meminta mereka untuk menginstal marketplace dan plugin ini.

Tambahkan `extraKnownMarketplaces` ke `.claude/settings.json` proyek Anda:

```
{
  "extraKnownMarketplaces": {
    "my-team-tools": {
      "source": {
        "source": "github",
        "repo": "your-org/claude-plugins"
      }
    }
  }
}
```

Untuk opsi konfigurasi lengkap termasuk `extraKnownMarketplaces` dan `enabledPlugins`, lihat Plugin settings.

Security

Plugin dan marketplace adalah komponen yang sangat dipercaya yang dapat menjalankan kode arbitrer di mesin Anda dengan hak istimewa pengguna Anda. Hanya instal plugin dan tambahkan marketplace dari sumber yang Anda percayai. Organisasi dapat membatasi marketplace mana yang diizinkan pengguna untuk ditambahkan menggunakan managed marketplace restrictions.

Troubleshooting

/plugin command not recognized

Jika Anda melihat "unknown command" atau perintah `/plugin` tidak muncul:

1. **Periksa versi Anda**: Jalankan `claude --version`. Plugin memerlukan versi 1.0.33 atau lebih baru.
2. **Perbarui Claude Code**:
   - **Homebrew**: `brew upgrade claude-code`
   - **npm**: `npm update -g @anthropic-ai/claude-code`
   - **Native installer**: Jalankan kembali perintah install dari Setup
3. **Restart Claude Code**: Setelah memperbarui, restart terminal Anda dan jalankan `claude` lagi.

Common issues

- **Marketplace not loading**: Verifikasi URL dapat diakses dan bahwa `.claude-plugin/marketplace.json` ada di jalur
- **Plugin installation failures**: Periksa bahwa URL sumber plugin dapat diakses dan repositori bersifat publik (atau Anda memiliki akses)
- **Files not found after installation**: Plugin disalin ke cache, jadi jalur yang merefe-rensikan file di luar direktori plugin tidak akan berfungsi
- **Plugin skills not appearing**: Hapus cache dengan `rm -rf ~/.claude/plugins/cache`, restart Claude Code, dan instal ulang plugin.

Untuk troubleshooting terperinci dengan solusi, lihat Troubleshooting dalam panduan marketplace. Untuk tools debugging, lihat Debugging and development tools.

Code intelligence issues

- **Language server not starting**: verifikasi binary diinstal dan tersedia di `$PATH` Anda. Periksa tab `/plugin` Errors untuk detail.
- **High memory usage**: language server seperti `rust-analyzer` dan `pyright` dapat mengonsumsi memori signifikan pada proyek besar. Jika Anda mengalami masalah memori, nonaktifkan plugin dengan `/plugin disable <plugin-name>` dan andalkan tools pencarian bawaan Claude sebagai gantinya.
- **False positive diagnostics in monorepos**: language server mungkin melaporkan kesalahan impor yang tidak terselesaikan untuk paket internal jika workspace tidak dikonfigurasi dengan benar. Ini tidak mempengaruhi kemampuan Claude untuk mengedit kode.

Next steps

- **Build your own plugins**: Lihat Plugins untuk membuat skills, agen, dan hooks
- **Create a marketplace**: Lihat Create a plugin marketplace untuk mendistribusikan plugin ke tim atau komunitas Anda
- **Technical reference**: Lihat Plugins reference untuk spesifikasi lengkap

Part 6: Advanced & Automation

Buat subagent khusus

> *Buat dan gunakan subagent AI khusus di Claude Code untuk alur kerja spesifik tugas dan manajemen konteks yang lebih baik.*

Subagent adalah asisten AI khusus yang menangani jenis tugas tertentu. Setiap subagent berjalan di jendela konteksnya sendiri dengan prompt sistem khusus, akses alat tertentu, dan izin independen. Ketika Claude menemukan tugas yang sesuai dengan deskripsi subagent, Claude mendelegasikan ke subagent tersebut, yang bekerja secara independen dan mengembalikan hasil.

> **Note:**
>
> Jika Anda memerlukan beberapa agen yang bekerja secara paralel dan berkomunikasi satu sama lain, lihat tim agen sebagai gantinya. Subagent bekerja dalam satu sesi; tim agen mengoordinasikan di seluruh sesi terpisah.

Subagent membantu Anda:

- **Menjaga konteks** dengan menjaga eksplorasi dan implementasi di luar percakapan utama Anda
- **Menerapkan batasan** dengan membatasi alat mana yang dapat digunakan subagent
- **Menggunakan kembali konfigurasi** di seluruh proyek dengan subagent tingkat pengguna
- **Mengkhususkan perilaku** dengan prompt sistem yang terfokus untuk domain tertentu
- **Mengontrol biaya** dengan merutekan tugas ke model yang lebih cepat dan lebih murah seperti Haiku

Claude menggunakan deskripsi setiap subagent untuk memutuskan kapan mendelegasikan tugas. Ketika Anda membuat subagent, tulis deskripsi yang jelas sehingga Claude tahu kapan menggunakannya.

Claude Code mencakup beberapa subagent bawaan seperti **Explore**, **Plan**, dan **general-purpose**. Anda juga dapat membuat subagent khusus untuk menangani tugas tertentu. Halaman ini mencakup subagent bawaan, cara membuat subagent Anda sendiri, opsi konfigurasi lengkap, pola untuk bekerja dengan subagent, dan contoh subagent.

Subagent bawaan

Claude Code mencakup subagent bawaan yang Claude gunakan secara otomatis jika sesuai. Masing-masing mewarisi izin percakapan induk dengan pembatasan alat tambahan.

Explore

Agen cepat yang dioptimalkan hanya-baca untuk pencarian dan analisis basis kode.

- **Model**: Haiku (cepat, latensi rendah)
- **Alat**: Alat hanya-baca (akses ditolak ke alat Write dan Edit)
- **Tujuan**: Penemuan file, pencarian kode, eksplorasi basis kode

Claude mendelegasikan ke Explore ketika perlu mencari atau memahami basis kode tanpa membuat perubahan. Ini menjaga hasil eksplorasi di luar konteks percakapan utama Anda.

Saat memanggil Explore, Claude menentukan tingkat ketelitian: **quick** untuk pencarian tertarget, **medium** untuk eksplorasi seimbang, atau **very thorough** untuk analisis komprehensif.

Plan

Agen penelitian yang digunakan selama plan mode untuk mengumpulkan konteks sebelum menyajikan rencana.

- **Model**: Mewarisi dari percakapan utama
- **Alat**: Alat hanya-baca (akses ditolak ke alat Write dan Edit)
- **Tujuan**: Penelitian basis kode untuk perencanaan

Ketika Anda dalam plan mode dan Claude perlu memahami basis kode Anda, Claude mendelegasikan penelitian ke subagent Plan. Ini mencegah nesting tak terbatas (subagent tidak dapat menelurkan subagent lain) sambil tetap mengumpulkan konteks yang diperlukan.

General-purpose

Agen yang mampu untuk tugas kompleks multi-langkah yang memerlukan eksplorasi dan tindakan.

- **Model**: Mewarisi dari percakapan utama
- **Alat**: Semua alat
- **Tujuan**: Penelitian kompleks, operasi multi-langkah, modifikasi kode

Claude mendelegasikan ke general-purpose ketika tugas memerlukan eksplorasi dan modifikasi, penalaran kompleks untuk menginterpretasi hasil, atau beberapa langkah yang saling bergantung.

Other

Claude Code mencakup agen pembantu tambahan untuk tugas tertentu. Ini biasanya dipanggil secara otomatis, jadi Anda tidak perlu menggunakannya secara langsung.

Agen	Model	Kapan Claude menggunakannya
Bash	Mewarisi	Menjalankan perintah terminal dalam konteks terpisah
statusline-setup	Sonnet	Ketika Anda menjalankan `/statusline` untuk mengonfigurasi baris status Anda
Claude Code Guide	Haiku	Ketika Anda mengajukan pertanyaan tentang fitur Claude Code

Selain subagent bawaan ini, Anda dapat membuat subagent Anda sendiri dengan prompt khusus, pembatasan alat, mode izin, hooks, dan skills. Bagian berikut menunjukkan cara memulai dan menyesuaikan subagent.

Quickstart: buat subagent pertama Anda

Subagent didefinisikan dalam file Markdown dengan frontmatter YAML. Anda dapat membuatnya secara manual atau menggunakan perintah `/agents`.

Panduan ini memandu Anda melalui pembuatan subagent tingkat pengguna dengan perintah `/agent`. Subagent meninjau kode dan menyarankan perbaikan untuk basis kode.

Step 1: Buka antarmuka subagent

Di Claude Code, jalankan:

```
/agents
```

Step 2: Buat agen tingkat pengguna baru

Pilih **Create new agent**, kemudian pilih **User-level**. Ini menyimpan subagent ke `~/.claude/agents/` sehingga tersedia di semua proyek Anda.

Step 3: Hasilkan dengan Claude

Pilih **Generate with Claude**. Ketika diminta, jelaskan subagent:

```
A code improvement agent that scans files and suggests improvements
for readability, performance, and best practices. It should explain
each issue, show the current code, and provide an improved version.
```

Claude menghasilkan prompt sistem dan konfigurasi. Tekan `e` untuk membukanya di editor Anda jika ingin menyesuaikannya.

Step 4: Pilih alat

Untuk reviewer hanya-baca, batalkan pilihan semuanya kecuali **Read-only tools**. Jika Anda membiarkan semua alat dipilih, subagent mewarisi semua alat yang tersedia untuk percakapan utama.

Step 5: Pilih model

Pilih model mana yang digunakan subagent. Untuk agen contoh ini, pilih **Sonnet**, yang menyeimbangkan kemampuan dan kecepatan untuk menganalisis pola kode.

Step 6: Pilih warna

Pilih warna latar belakang untuk subagent. Ini membantu Anda mengidentifikasi subagent mana yang berjalan di UI.

Step 7: Simpan dan coba

Simpan subagent. Tersedia segera (tidak perlu restart). Coba:

```
Use the code-improver agent to suggest improvements in this project
```

Claude mendelegasikan ke subagent baru Anda, yang memindai basis kode dan mengembalikan saran perbaikan.

Anda sekarang memiliki subagent yang dapat Anda gunakan di proyek apa pun di mesin Anda untuk menganalisis basis kode dan menyarankan perbaikan.

Anda juga dapat membuat subagent secara manual sebagai file Markdown, mendefinisikannya melalui flag CLI, atau mendistribusikannya melalui plugin. Bagian berikut mencakup semua opsi konfigurasi.

Konfigurasi subagent

Gunakan perintah /agents

Perintah `/agents` menyediakan antarmuka interaktif untuk mengelola subagent. Jalankan `/agents` untuk:

- Melihat semua subagent yang tersedia (bawaan, pengguna, proyek, dan plugin)
- Membuat subagent baru dengan setup terpandu atau generasi Claude
- Mengedit konfigurasi subagent yang ada dan akses alat
- Menghapus subagent khusus
- Melihat subagent mana yang aktif ketika duplikat ada

Ini adalah cara yang direkomendasikan untuk membuat dan mengelola subagent. Untuk pembuatan manual atau otomasi, Anda juga dapat menambahkan file subagent secara langsung.

Untuk membuat daftar semua subagent yang dikonfigurasi dari baris perintah tanpa memulai sesi interaktif, jalankan `claude agents`. Ini menunjukkan agen yang dikelompokkan berdasarkan sumber dan menunjukkan mana yang ditimpa oleh definisi prioritas lebih tinggi.

Pilih cakupan subagent

Subagent adalah file Markdown dengan frontmatter YAML. Simpan mereka di lokasi berbeda tergantung cakupan. Ketika beberapa subagent berbagi nama yang sama, lokasi prioritas lebih tinggi menang.

Lokasi	Cakupan	Prioritas	Cara membuat
Flag CLI `--agents`	Sesi saat ini	1 (tertinggi)	Lewatkan JSON saat meluncurkan Claude Code

Lokasi	Cakupan	Prioritas	Cara membuat
`.claude/agents/`	Proyek saat ini	2	Interaktif atau manual
`~/.claude/agents/`	Semua proyek Anda	3	Interaktif atau manual
Direktori `agents/` plugin	Tempat plugin diaktifkan	4 (terendah)	Diinstal dengan plugins

Subagent proyek (`.claude/agents/`) ideal untuk subagent spesifik untuk basis kode. Periksa mereka ke kontrol versi sehingga tim Anda dapat menggunakannya dan meningkatkannya secara kolaboratif.

Subagent pengguna (`~/.claude/agents/`) adalah subagent pribadi yang tersedia di semua proyek Anda.

Subagent yang ditentukan CLI dilewatkan sebagai JSON saat meluncurkan Claude Code. Mereka hanya ada untuk sesi itu dan tidak disimpan ke disk, menjadikannya berguna untuk pengujian cepat atau skrip otomasi:

```
claude --agents '{
  "code-reviewer": {
    "description": "Expert code reviewer. Use proactively after code changes.",
    "prompt": "You are a senior code reviewer. Focus on code quality, security,
and best practices.",
    "tools": ["Read", "Grep", "Glob", "Bash"],
    "model": "sonnet"
  }
}'
```

Flag `--agents` menerima JSON dengan frontmatter yang sama bidang file-based subagent: `description` , `prompt` , `tools` , `disallowedTools` , `model` , `permissionMode` , `mcpServers` , `hooks` , `maxTurns` , `skills` , dan `memory` . Gunakan `prompt` untuk prompt sistem, setara dengan badan markdown dalam subagent berbasis file. Lihat referensi CLI untuk format JSON lengkap.

Subagent plugin berasal dari plugin yang telah Anda instal. Mereka muncul di `/agents` bersama subagent khusus Anda. Lihat referensi komponen plugin untuk detail tentang pembuatan subagent plugin.

Tulis file subagent

File subagent menggunakan frontmatter YAML untuk konfigurasi, diikuti oleh prompt sistem dalam Markdown:

> **Note:**
>
> Subagent dimuat saat awal sesi. Jika Anda membuat subagent dengan menambahkan file secara manual, mulai ulang sesi Anda atau gunakan `/agents` untuk memuatnya segera.

```
---
name: code-reviewer
description: Reviews code for quality and best practices
tools: Read, Glob, Grep
model: sonnet
---

You are a code reviewer. When invoked, analyze the code and provide
specific, actionable feedback on quality, security, and best practices.
```

Frontmatter mendefinisikan metadata dan konfigurasi subagent. Badan menjadi prompt sistem yang memandu perilaku subagent. Subagent menerima hanya prompt sistem ini (ditambah detail lingkungan dasar seperti direktori kerja), bukan prompt sistem Claude Code lengkap.

Bidang frontmatter yang didukung

Bidang berikut dapat digunakan dalam frontmatter YAML. Hanya `name` dan `description` yang diperlukan.

Bidang	Diperlukan	Deskripsi
`name`	Ya	Pengenal unik menggunakan huruf kecil dan tanda hubung
`description`	Ya	Kapan Claude harus mendelegasikan ke subagent ini

Bidang	Diperlukan	Deskripsi
`tools`	Tidak	Alat yang dapat digunakan subagent. Mewarisi semua alat jika dihilangkan
`disallowedTools`	Tidak	Alat untuk ditolak, dihapus dari daftar yang diwarisi atau ditentukan
`model`	Tidak	Model untuk digunakan: `sonnet`, `opus`, `haiku`, atau `inherit`. Default ke `inherit`
`permissionMode`	Tidak	Mode izin: `default`, `acceptEdits`, `dontAsk`, `bypassPermissions`, atau `plan`
`maxTurns`	Tidak	Jumlah maksimum putaran agentic sebelum subagent berhenti
`skills`	Tidak	Skills untuk dimuat ke dalam konteks subagent saat startup. Konten skill lengkap disuntikkan, bukan hanya tersedia untuk invokasi. Subagent tidak mewarisi skills dari percakapan induk
`mcpServers`	Tidak	MCP servers tersedia untuk subagent ini. Setiap entri adalah nama server yang mereferensikan server yang sudah dikonfigurasi (misalnya, `"slack"`) atau definisi inline dengan nama server sebagai kunci dan konfigurasi MCP server lengkap sebagai nilai

Bidang	Diperlukan	Deskripsi
`hooks`	Tidak	Lifecycle hooks yang dibatasi pada subagent ini
`memory`	Tidak	Cakupan memori persisten: `user`, `project`, atau `local`. Memungkinkan pembelajaran lintas sesi
`background`	Tidak	Atur ke `true` untuk selalu menjalankan subagent ini sebagai tugas latar belakang. Default: `false`
`isolation`	Tidak	Atur ke `worktree` untuk menjalankan subagent dalam git worktree sementara, memberikannya salinan terisolasi dari repositori. Worktree secara otomatis dibersihkan jika subagent tidak membuat perubahan

Pilih model

Bidang `model` mengontrol model AI mana yang digunakan subagent:

- **Alias model**: Gunakan salah satu alias yang tersedia: `sonnet`, `opus`, atau `haiku`
- **inherit**: Gunakan model yang sama dengan percakapan utama
- **Dihilangkan**: Jika tidak ditentukan, default ke `inherit` (menggunakan model yang sama dengan percakapan utama)

Kontrol kemampuan subagent

Anda dapat mengontrol apa yang dapat dilakukan subagent melalui akses alat, mode izin, dan aturan bersyarat.

Alat yang tersedia

Subagent dapat menggunakan salah satu alat internal Claude Code. Secara default, subagent mewarisi semua alat dari percakapan utama, termasuk alat MCP.

Untuk membatasi alat, gunakan bidang `tools` (allowlist) atau bidang `disallowedTools` (denylist):

```
---
name: safe-researcher
description: Research agent with restricted capabilities
tools: Read, Grep, Glob, Bash
disallowedTools: Write, Edit
---
```

Batasi subagent mana yang dapat dihasilkan

Ketika agen berjalan sebagai thread utama dengan `claude --agent`, agen dapat menelurkan subagent menggunakan alat Agent. Untuk membatasi jenis subagent mana yang dapat dihasilkan, gunakan sintaks `Agent(agent_type)` dalam bidang `tools`.

> **Note:**
>
> Dalam versi 2.1.63, alat Task diganti nama menjadi Agent. Referensi `Task(...)` yang ada dalam pengaturan dan definisi agen masih berfungsi sebagai alias.

```
---
name: coordinator
description: Coordinates work across specialized agents
tools: Agent(worker, researcher), Read, Bash
---
```

Ini adalah allowlist: hanya subagent `worker` dan `researcher` yang dapat dihasilkan. Jika agen mencoba menelurkan jenis lain, permintaan gagal dan agen hanya melihat jenis yang diizinkan dalam promptnya. Untuk memblokir agen tertentu sambil mengizinkan semua yang lain, gunakan `permissions.deny` sebagai gantinya.

Untuk mengizinkan peneluruan subagent apa pun tanpa pembatasan, gunakan `Agent` tanpa tanda kurung:

```
tools: Agent, Read, Bash
```

Jika `Agent` dihilangkan dari daftar `tools` sepenuhnya, agen tidak dapat menelurkan subagent apa pun. Pembatasan ini hanya berlaku untuk agen yang berjalan sebagai thread utama dengan `claude --agent`. Subagent tidak dapat menelurkan subagent lain, jadi `Agent(agent_type)` tidak berpengaruh dalam definisi subagent.

Mode izin

Bidang `permissionMode` mengontrol bagaimana subagent menangani prompt izin. Subagent mewarisi konteks izin dari percakapan utama tetapi dapat menimpa mode.

Mode	Perilaku
`default`	Pemeriksaan izin standar dengan prompt
`acceptEdits`	Auto-terima pengeditan file
`dontAsk`	Auto-tolak prompt izin (alat yang secara eksplisit diizinkan masih berfungsi)
`bypassPermissions`	Lewati semua pemeriksaan izin
`plan`	Plan mode (eksplorasi hanya-baca)

> **Warning:**
>
> Gunakan `bypassPermissions` dengan hati-hati. Ini melewati semua pemeriksaan izin, memungkinkan subagent untuk menjalankan operasi apa pun tanpa persetujuan.

Jika induk menggunakan `bypassPermissions`, ini mengambil alih dan tidak dapat ditimpa.

Preload skills ke dalam subagent

Gunakan bidang `skills` untuk menyuntikkan konten skill ke dalam konteks subagent saat startup. Ini memberikan subagent pengetahuan domain tanpa memerlukan penemuan dan pemuatan skills selama eksekusi.

```
---
name: api-developer
description: Implement API endpoints following team conventions
skills:
  - api-conventions
  - error-handling-patterns
---

Implement API endpoints. Follow the conventions and patterns from the preloaded
skills.
```

Konten lengkap setiap skill disuntikkan ke dalam konteks subagent, bukan hanya tersedia untuk invokasi. Subagent tidak mewarisi skills dari percakapan induk; Anda harus mencantumkannya secara eksplisit.

> **Note:**
>
> Ini adalah kebalikan dari menjalankan skill dalam subagent. Dengan `skills` dalam subagent, subagent mengontrol prompt sistem dan memuat konten skill. Dengan `context: fork` dalam skill, konten skill disuntikkan ke dalam agen yang Anda tentukan. Keduanya menggunakan sistem yang mendasar sama.

Aktifkan memori persisten

Bidang `memory` memberikan subagent direktori persisten yang bertahan di seluruh percakapan. Subagent menggunakan direktori ini untuk membangun pengetahuan seiring waktu, seperti pola basis kode, wawasan debugging, dan keputusan arsitektur.

```
---
name: code-reviewer
description: Reviews code for quality and best practices
memory: user
---

You are a code reviewer. As you review code, update your agent memory with
patterns, conventions, and recurring issues you discover.
```

Pilih cakupan berdasarkan seberapa luas memori harus diterapkan:

Cakupan	Lokasi	Gunakan ketika
`user`	`~/.claude/agent-memory/<name-of-agent>/`	subagent harus mengingat pembelajaran di semua proyek
`project`	`.claude/agent-memory/<name-of-agent>/`	pengetahuan subagent spesifik proyek dan dapat dibagikan melalui kontrol versi
`local`	`.claude/agent-memory-local/<name-of-agent>/`	pengetahuan subagent spesifik proyek tetapi tidak boleh diperiksa ke dalam kontrol versi

Ketika memori diaktifkan:

- Prompt sistem subagent mencakup instruksi untuk membaca dan menulis ke direktori memori.
- Prompt sistem subagent juga mencakup 200 baris pertama `MEMORY.md` dalam direktori memori, dengan instruksi untuk mengkurasi `MEMORY.md` jika melebihi 200 baris.
- Alat Read, Write, dan Edit secara otomatis diaktifkan sehingga subagent dapat mengelola file memorinya.

Tips memori persisten

- `user` adalah cakupan default yang direkomendasikan. Gunakan `project` atau `local` ketika pengetahuan subagent hanya relevan untuk basis kode tertentu.
- Minta subagent untuk berkonsultasi dengan memorinya sebelum memulai pekerjaan: "Review PR ini, dan periksa memori Anda untuk pola yang telah Anda lihat sebelumnya."
- Minta subagent untuk memperbarui memorinya setelah menyelesaikan tugas: "Sekarang setelah Anda selesai, simpan apa yang Anda pelajari ke memori Anda." Seiring waktu, ini membangun basis pengetahuan yang membuat subagent lebih efektif.
- Sertakan instruksi memori langsung dalam file markdown subagent sehingga secara proaktif mempertahankan basis pengetahuannya sendiri:

```
Update your agent memory as you discover codepaths, patterns, library
locations, and key architectural decisions. This builds up institutional
knowledge across conversations. Write concise notes about what you found
and where.
```

Aturan bersyarat dengan hooks

Untuk kontrol yang lebih dinamis atas penggunaan alat, gunakan hooks `PreToolUse` untuk memvalidasi operasi sebelum dieksekusi. Ini berguna ketika Anda perlu mengizinkan beberapa operasi alat sambil memblokir yang lain.

Contoh ini membuat subagent yang hanya mengizinkan kueri database hanya-baca. Hook `PreToolUse` menjalankan skrip yang ditentukan dalam `command` sebelum setiap perintah Bash dieksekusi:

```
---
name: db-reader
description: Execute read-only database queries
tools: Bash
hooks:
  PreToolUse:
    - matcher: "Bash"
      hooks:
        - type: command
          command: "./scripts/validate-readonly-query.sh"
---
```

Claude Code melewatkan input hook sebagai JSON melalui stdin ke perintah hook. Skrip validasi membaca JSON ini, mengekstrak perintah Bash, dan keluar dengan kode 2 untuk memblokir operasi penulisan:

```
#!/bin/bash
## ./scripts/validate-readonly-query.sh

INPUT=$(cat)
COMMAND=$(echo "$INPUT" | jq -r '.tool_input.command // empty')

## Block SQL write operations (case-insensitive)
if echo "$COMMAND" | grep -iE '\b(INSERT|UPDATE|DELETE|DROP|CREATE|ALTER|TRUNCATE)
\b' > /dev/null; then
  echo "Blocked: Only SELECT queries are allowed" >&2
  exit 2
fi

exit 0
```

Lihat Hook input untuk skema input lengkap dan exit codes untuk bagaimana kode keluar mempengaruhi perilaku.

Nonaktifkan subagent tertentu

Anda dapat mencegah Claude menggunakan subagent tertentu dengan menambahkannya ke array `deny` dalam pengaturan Anda. Gunakan format `Agent(subagent-name)` di mana `subagent-name` cocok dengan bidang nama subagent.

```
{
  "permissions": {
    "deny": ["Agent(Explore)", "Agent(my-custom-agent)"]
  }
}
```

Ini berfungsi untuk subagent bawaan dan khusus. Anda juga dapat menggunakan flag CLI `--disallowedTools`:

```
claude --disallowedTools "Agent(Explore)"
```

Lihat dokumentasi Permissions untuk detail lebih lanjut tentang aturan izin.

Tentukan hooks untuk subagent

Subagent dapat mendefinisikan hooks yang berjalan selama siklus hidup subagent. Ada dua cara untuk mengonfigurasi hooks:

1. **Dalam frontmatter subagent**: Tentukan hooks yang hanya berjalan saat subagent itu aktif
2. **Dalam** `settings.json` : Tentukan hooks yang berjalan dalam sesi utama ketika subagent dimulai atau berhenti

Hooks dalam frontmatter subagent

Tentukan hooks langsung dalam file markdown subagent. Hooks ini hanya berjalan saat subagent spesifik itu aktif dan dibersihkan saat selesai.

Semua hook events didukung. Peristiwa paling umum untuk subagent adalah:

Peristiwa	Input Matcher	Kapan terjadi
`PreToolUse`	Nama alat	Sebelum subagent menggunakan alat
`PostToolUse`	Nama alat	Setelah subagent menggunakan alat
`Stop`	(tidak ada)	Ketika subagent selesai (dikonversi ke `SubagentStop` saat runtime)

Contoh ini memvalidasi perintah Bash dengan hook `PreToolUse` dan menjalankan linter setelah pengeditan file dengan `PostToolUse` :

```
---
name: code-reviewer
description: Review code changes with automatic linting
hooks:
  PreToolUse:
    - matcher: "Bash"
      hooks:
        - type: command
          command: "./scripts/validate-command.sh $TOOL_INPUT"
  PostToolUse:
    - matcher: "Edit|Write"
      hooks:
        - type: command
          command: "./scripts/run-linter.sh"
---
```

Hook `Stop` dalam frontmatter secara otomatis dikonversi ke peristiwa `SubagentStop`.

Hook tingkat proyek untuk peristiwa subagent

Konfigurasi hooks dalam `settings.json` yang merespons peristiwa siklus hidup subagent dalam sesi utama.

Peristiwa	Input Matcher	Kapan terjadi
`SubagentStart`	Nama jenis agen	Ketika subagent mulai dieksekusi
`SubagentStop`	Nama jenis agen	Ketika subagent selesai

Kedua peristiwa mendukung matcher untuk menargetkan jenis agen tertentu berdasarkan nama. Contoh ini menjalankan skrip setup hanya ketika subagent `db-agent` dimulai, dan skrip cleanup ketika subagent apa pun berhenti:

```
{
  "hooks": {
    "SubagentStart": [
      {
        "matcher": "db-agent",
        "hooks": [
          { "type": "command", "command": "./scripts/setup-db-connection.sh" }
        ]
      }
    ],
    "SubagentStop": [
      {
        "hooks": [
          { "type": "command", "command": "./scripts/cleanup-db-connection.sh" }
        ]
      }
    ]
  }
}
```

Lihat Hooks untuk format konfigurasi hook lengkap.

Bekerja dengan subagent

Pahami delegasi otomatis

Claude secara otomatis mendelegasikan tugas berdasarkan deskripsi tugas dalam permintaan Anda, bidang `description` dalam konfigurasi subagent, dan konteks saat ini. Untuk mendorong delegasi proaktif, sertakan frasa seperti “use proactively” dalam bidang deskripsi subagent Anda.

Anda juga dapat meminta subagent tertentu secara eksplisit:

```
Use the test-runner subagent to fix failing tests
Have the code-reviewer subagent look at my recent changes
```

Jalankan subagent di foreground atau background

Subagent dapat berjalan di foreground (blocking) atau background (concurrent):

- **Subagent foreground** memblokir percakapan utama sampai selesai. Prompt izin dan pertanyaan klarifikasi (seperti `AskUserQuestion`) dilewatkan kepada Anda.
- **Subagent background** berjalan secara bersamaan sementara Anda terus bekerja. Sebelum diluncurkan, Claude Code meminta izin alat apa pun yang akan dibutuhkan subagent, memastikan memiliki persetujuan yang diperlukan di muka. Setelah berjalan, subagent mewarisi izin ini dan auto-tolak apa pun yang tidak disetujui sebelumnya. Jika subagent background perlu mengajukan pertanyaan klarifikasi, panggilan alat itu gagal tetapi subagent terus.

Jika subagent background gagal karena izin yang hilang, Anda dapat melanjutkannya di foreground untuk mencoba lagi dengan prompt interaktif.

Claude memutuskan apakah akan menjalankan subagent di foreground atau background berdasarkan tugas. Anda juga dapat:

- Minta Claude untuk "run this in the background"
- Tekan **Ctrl+B** untuk menempatkan tugas yang sedang berjalan di background

Untuk menonaktifkan semua fungsionalitas tugas background, atur variabel lingkungan `CLAUDE_CODE_DISABLE_BACKGROUND_TASKS` ke `1`. Lihat Environment variables.

Pola umum

Isolasi operasi volume tinggi

Salah satu penggunaan paling efektif untuk subagent adalah mengisolasi operasi yang menghasilkan jumlah output besar. Menjalankan tes, mengambil dokumentasi, atau memproses file log dapat mengonsumsi konteks yang signifikan. Dengan mendelegasikan ini ke subagent, output verbose tetap dalam konteks subagent sementara hanya ringkasan relevan yang kembali ke percakapan utama Anda.

```
Use a subagent to run the test suite and report only the failing tests with their
error messages
```

Jalankan penelitian paralel

Untuk investigasi independen, hasilkan beberapa subagent untuk bekerja secara bersamaan:

```
Research the authentication, database, and API modules in parallel using separate
subagents
```

Setiap subagent mengeksplorasi areanya secara independen, kemudian Claude mensintesis temuan. Ini berfungsi terbaik ketika jalur penelitian tidak saling bergantung.

> **Warning:**
>
> Ketika subagent selesai, hasil mereka kembali ke percakapan utama Anda. Menjalankan banyak subagent yang masing-masing mengembalikan hasil terperinci dapat mengonsumsi konteks yang signifikan.

Untuk tugas yang memerlukan paralelisme berkelanjutan atau melebihi jendela konteks Anda, tim agen memberikan setiap pekerja konteksnya sendiri yang independen.

Rantai subagent

Untuk alur kerja multi-langkah, minta Claude untuk menggunakan subagent secara berurutan. Setiap subagent menyelesaikan tugasnya dan mengembalikan hasil ke Claude, yang kemudian melewatkan konteks relevan ke subagent berikutnya.

```
Use the code-reviewer subagent to find performance issues, then use the optimizer
subagent to fix them
```

Pilih antara subagent dan percakapan utama

Gunakan **percakapan utama** ketika:

- Tugas memerlukan bolak-balik atau penyempurnaan iteratif yang sering
- Beberapa fase berbagi konteks signifikan (perencanaan → implementasi → pengujian)
- Anda membuat perubahan cepat dan tertarget
- Latensi penting. Subagent dimulai segar dan mungkin memerlukan waktu untuk mengumpulkan konteks

Gunakan **subagent** ketika:

- Tugas menghasilkan output verbose yang Anda tidak butuhkan dalam konteks utama Anda
- Anda ingin menerapkan pembatasan alat atau izin tertentu
- Pekerjaan mandiri dan dapat mengembalikan ringkasan

Pertimbangkan Skills sebagai gantinya ketika Anda menginginkan prompt atau alur kerja yang dapat digunakan kembali yang berjalan dalam konteks percakapan utama daripada konteks subagent terisolasi.

Untuk pertanyaan cepat tentang sesuatu yang sudah ada dalam percakapan Anda, gunakan `/btw` sebagai gantinya dari subagent. Ini melihat konteks penuh Anda tetapi tidak memiliki akses alat, dan jawabannya dibuang daripada ditambahkan ke riwayat.

> **Note:**
>
> Subagent tidak dapat menelurkan subagent lain. Jika alur kerja Anda memerlukan delegasi bersarang, gunakan Skills atau rantai subagent dari percakapan utama.

Kelola konteks subagent

Lanjutkan subagent

Setiap invokasi subagent membuat instance baru dengan konteks segar. Untuk melanjutkan pekerjaan subagent yang ada daripada memulai dari awal, minta Claude untuk melanjutkannya.

Subagent yang dilanjutkan mempertahankan riwayat percakapan lengkap mereka, termasuk semua panggilan alat sebelumnya, hasil, dan penalaran. Subagent melanjutkan tepat di mana berhenti daripada memulai segar.

Ketika subagent selesai, Claude menerima ID agennya. Untuk melanjutkan subagent, minta Claude untuk melanjutkan pekerjaan sebelumnya:

```
Use the code-reviewer subagent to review the authentication module
[Agent completes]

Continue that code review and now analyze the authorization logic
[Claude resumes the subagent with full context from previous conversation]
```

Anda juga dapat meminta Claude untuk ID agen jika ingin mereferensikannya secara eksplisit, atau temukan ID dalam file transkrip di `~/.claude/projects/{project}/{sessionId}/subagents/`. Setiap transkrip disimpan sebagai `agent-{agentId}.jsonl`.

Transkrip subagent bertahan secara independen dari percakapan utama:

- **Pemadatan percakapan utama**: Ketika percakapan utama dipadatkan, transkrip subagent tidak terpengaruh. Mereka disimpan dalam file terpisah.

- **Persistensi sesi**: Transkrip subagent bertahan dalam sesi mereka. Anda dapat melanjutkan subagent setelah memulai ulang Claude Code dengan melanjutkan sesi yang sama.
- **Pembersihan otomatis**: Transkrip dibersihkan berdasarkan pengaturan `cleanupPeriodDays` (default: 30 hari).

Auto-compaction

Subagent mendukung pemadatan otomatis menggunakan logika yang sama dengan percakapan utama. Secara default, auto-compaction dipicu pada sekitar 95% kapasitas. Untuk memicu pemadatan lebih awal, atur `CLAUDE_AUTOCOMPACT_PCT_OVERRIDE` ke persentase lebih rendah (misalnya, `50`). Lihat environment variables untuk detail.

Peristiwa pemadatan dicatat dalam file transkrip subagent:

```
{
  "type": "system",
  "subtype": "compact_boundary",
  "compactMetadata": {
    "trigger": "auto",
    "preTokens": 167189
  }
}
```

Nilai `preTokens` menunjukkan berapa banyak token yang digunakan sebelum pemadatan terjadi.

Contoh subagent

Contoh-contoh ini mendemonstrasikan pola efektif untuk membangun subagent. Gunakan mereka sebagai titik awal, atau hasilkan versi yang disesuaikan dengan Claude.

> **Tip:**
>
> **Best practices:**
>
> - **Desain subagent yang terfokus:** setiap subagent harus unggul dalam satu tugas spesifik
> - **Tulis deskripsi terperinci:** Claude menggunakan deskripsi untuk memutuskan kapan mendelegasikan
> - **Batasi akses alat:** berikan hanya izin yang diperlukan untuk keamanan dan fokus

- **Periksa ke dalam kontrol versi:** bagikan subagent proyek dengan tim Anda

Peninjau kode

Subagent hanya-baca yang meninjau kode tanpa memodifikasinya. Contoh ini menunjukkan cara merancang subagent yang terfokus dengan akses alat terbatas (tidak ada Edit atau Write) dan prompt terperinci yang menentukan dengan tepat apa yang harus dicari dan cara memformat output.

```
---
name: code-reviewer
description: Expert code review specialist. Proactively reviews code for quality,
security, and maintainability. Use immediately after writing or modifying code.
tools: Read, Grep, Glob, Bash
model: inherit
---

You are a senior code reviewer ensuring high standards of code quality and
security.

When invoked:
1. Run git diff to see recent changes
2. Focus on modified files
3. Begin review immediately

Review checklist:
- Code is clear and readable
- Functions and variables are well-named
- No duplicated code
- Proper error handling
- No exposed secrets or API keys
- Input validation implemented
- Good test coverage
- Performance considerations addressed

Provide feedback organized by priority:
- Critical issues (must fix)
- Warnings (should fix)
- Suggestions (consider improving)

Include specific examples of how to fix issues.
```

Debugger

Subagent yang dapat menganalisis dan memperbaiki masalah. Tidak seperti peninjau kode, yang ini mencakup Edit karena memperbaiki bug memerlukan memodifikasi kode. Prompt menyediakan alur kerja yang jelas dari diagnosis hingga verifikasi.

```
---
name: debugger
description: Debugging specialist for errors, test failures, and unexpected
behavior. Use proactively when encountering any issues.
tools: Read, Edit, Bash, Grep, Glob
---

You are an expert debugger specializing in root cause analysis.

When invoked:
1. Capture error message and stack trace
2. Identify reproduction steps
3. Isolate the failure location
4. Implement minimal fix
5. Verify solution works

Debugging process:
- Analyze error messages and logs
- Check recent code changes
- Form and test hypotheses
- Add strategic debug logging
- Inspect variable states

For each issue, provide:
- Root cause explanation
- Evidence supporting the diagnosis
- Specific code fix
- Testing approach
- Prevention recommendations

Focus on fixing the underlying issue, not the symptoms.
```

Data scientist

Subagent khusus domain untuk pekerjaan analisis data. Contoh ini menunjukkan cara membuat subagent untuk alur kerja khusus di luar tugas pengkodean khas. Ini secara eksplisit menetapkan `model: sonnet` untuk analisis yang lebih mampu.

```
---
name: data-scientist
description: Data analysis expert for SQL queries, BigQuery operations, and data
insights. Use proactively for data analysis tasks and queries.
tools: Bash, Read, Write
model: sonnet
---

You are a data scientist specializing in SQL and BigQuery analysis.

When invoked:
1. Understand the data analysis requirement
2. Write efficient SQL queries
3. Use BigQuery command line tools (bq) when appropriate
4. Analyze and summarize results
5. Present findings clearly

Key practices:
- Write optimized SQL queries with proper filters
- Use appropriate aggregations and joins
- Include comments explaining complex logic
- Format results for readability
- Provide data-driven recommendations

For each analysis:
- Explain the query approach
- Document any assumptions
- Highlight key findings
- Suggest next steps based on data

Always ensure queries are efficient and cost-effective.
```

Validator kueri database

Subagent yang memungkinkan akses Bash tetapi memvalidasi perintah untuk mengizinkan hanya kueri SQL hanya-baca. Contoh ini menunjukkan cara menggunakan hooks `PreToolUse` untuk validasi bersyarat ketika Anda memerlukan kontrol lebih halus daripada bidang `tools` yang disediakan.

```
---
name: db-reader
description: Execute read-only database queries. Use when analyzing data or
generating reports.
tools: Bash
hooks:
  PreToolUse:
    - matcher: "Bash"
      hooks:
        - type: command
          command: "./scripts/validate-readonly-query.sh"
---

You are a database analyst with read-only access. Execute SELECT queries to answer
questions about the data.

When asked to analyze data:
1. Identify which tables contain the relevant data
2. Write efficient SELECT queries with appropriate filters
3. Present results clearly with context

You cannot modify data. If asked to INSERT, UPDATE, DELETE, or modify schema,
explain that you only have read access.
```

Claude Code melewatkan input hook sebagai JSON melalui stdin ke perintah hook. Skrip validasi membaca JSON ini, mengekstrak perintah yang sedang dieksekusi, dan memeriksanya terhadap daftar operasi penulisan SQL. Jika operasi penulisan terdeteksi, skrip keluar dengan kode 2 untuk memblokir eksekusi dan mengembalikan pesan kesalahan ke Claude melalui stderr.

Buat skrip validasi di mana saja dalam proyek Anda. Jalur harus cocok dengan bidang `command` dalam konfigurasi hook Anda:

```
#!/bin/bash
## Blocks SQL write operations, allows SELECT queries

## Read JSON input from stdin
INPUT=$(cat)

## Extract the command field from tool_input using jq
COMMAND=$(echo "$INPUT" | jq -r '.tool_input.command // empty')

if [ -z "$COMMAND" ]; then
  exit 0
fi

## Block write operations (case-insensitive)
if echo "$COMMAND" | grep -iE '\b(INSERT|UPDATE|DELETE|DROP|CREATE|ALTER|TRUNCATE|
REPLACE|MERGE)\b' > /dev/null; then
  echo "Blocked: Write operations not allowed. Use SELECT queries only." >&2
  exit 2
fi

exit 0
```

Buat skrip dapat dieksekusi:

```
chmod +x ./scripts/validate-readonly-query.sh
```

Hook menerima JSON melalui stdin dengan perintah Bash dalam `tool_input.command`. Kode keluar 2 memblokir operasi dan mengirimkan pesan kesalahan kembali ke Claude. Lihat Hooks untuk detail tentang kode keluar dan Hook input untuk skema input lengkap.

Langkah berikutnya

Sekarang setelah Anda memahami subagent, jelajahi fitur terkait ini:

- Distribusikan subagent dengan plugin untuk berbagi subagent di seluruh tim atau proyek
- Jalankan Claude Code secara terprogram dengan Agent SDK untuk CI/CD dan otomasi

- Gunakan MCP servers untuk memberikan subagent akses ke alat dan data eksternal

Koordinasikan tim Claude Code sessions

> *Koordinasikan beberapa instance Claude Code yang bekerja bersama sebagai tim, dengan tugas bersama, pesan antar-agent, dan manajemen terpusat.*

Warning:

Tim agent bersifat eksperimental dan dinonaktifkan secara default. Aktifkan dengan menambahkan `CLAUDE_CODE_EXPERIMENTAL_AGENT_TEAMS` ke settings.json atau environment Anda. Tim agent memiliki keterbatasan yang diketahui seputar resumption session, koordinasi tugas, dan perilaku shutdown.

Tim agent memungkinkan Anda mengoordinasikan beberapa instance Claude Code yang bekerja bersama. Satu session bertindak sebagai team lead, mengoordinasikan pekerjaan, menugaskan tugas, dan mensintesis hasil. Rekan tim bekerja secara independen, masing-masing dalam context window-nya sendiri, dan berkomunikasi langsung satu sama lain.

Tidak seperti subagents, yang berjalan dalam satu session dan hanya dapat melaporkan kembali ke agent utama, Anda juga dapat berinteraksi dengan rekan tim individual secara langsung tanpa melalui lead.

Note:

Tim agent memerlukan Claude Code v2.1.32 atau lebih baru. Periksa versi Anda dengan `claude --version`.

Halaman ini mencakup:

- Kapan menggunakan tim agent, termasuk use case terbaik dan bagaimana perbandingannya dengan subagents
- Memulai tim
- Mengendalikan rekan tim, termasuk mode tampilan, penugasan tugas, dan delegasi
- Best practices untuk pekerjaan paralel

Kapan menggunakan tim agent

Tim agent paling efektif untuk tugas di mana eksplorasi paralel menambah nilai nyata. Lihat contoh use case untuk skenario lengkap. Use case terkuat adalah:

- **Penelitian dan review**: beberapa rekan tim dapat menyelidiki aspek berbeda dari masalah secara bersamaan, kemudian berbagi dan menantang temuan satu sama lain
- **Modul atau fitur baru**: rekan tim dapat masing-masing memiliki bagian terpisah tanpa saling mengganggu
- **Debugging dengan hipotesis bersaing**: rekan tim menguji teori berbeda secara paralel dan berkumpul pada jawaban lebih cepat
- **Koordinasi lintas-layer**: perubahan yang mencakup frontend, backend, dan test, masing-masing dimiliki oleh rekan tim berbeda

Tim agent menambah overhead koordinasi dan menggunakan token secara signifikan lebih banyak daripada satu session. Mereka bekerja paling baik ketika rekan tim dapat beroperasi secara independen. Untuk tugas sekuensial, edit file yang sama, atau pekerjaan dengan banyak dependensi, satu session atau subagents lebih efektif.

Bandingkan dengan subagents

Baik tim agent maupun subagents memungkinkan Anda memparalelkan pekerjaan, tetapi mereka beroperasi berbeda. Pilih berdasarkan apakah pekerja Anda perlu berkomunikasi satu sama lain:

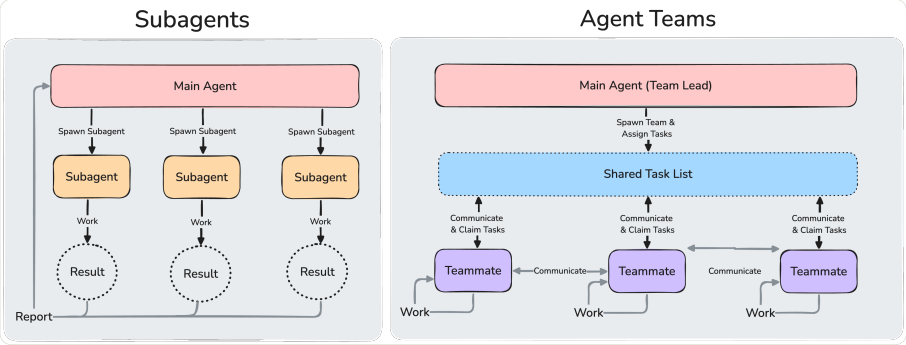


Diagram membandingkan arsitektur subagent dan tim agent. Subagents dihasilkan oleh agent utama, melakukan pekerjaan, dan melaporkan hasil kembali. Tim agent berkoordinasi melalui daftar tugas bersama, dengan rekan tim berkomunikasi langsung satu sama lain.

	Subagents	Tim agent
Context	Context window sendiri; hasil kembali ke pemanggil	Context window sendiri; sepenuhnya independen
Komunikasi	Melaporkan hasil kembali ke agent utama saja	Rekan tim saling mengirim pesan secara langsung
Koordinasi	Agent utama mengelola semua pekerjaan	Daftar tugas bersama dengan self-coordination
Terbaik untuk	Tugas terfokus di mana hanya hasil yang penting	Pekerjaan kompleks yang memerlukan diskusi dan kolaborasi
Biaya token	Lebih rendah: hasil diringkas kembali ke context utama	Lebih tinggi: setiap rekan tim adalah instance Claude terpisah

Gunakan subagents ketika Anda membutuhkan pekerja cepat dan terfokus yang melaporkan kembali. Gunakan tim agent ketika rekan tim perlu berbagi temuan, menantang satu sama lain, dan berkoordinasi sendiri.

Aktifkan tim agent

Tim agent dinonaktifkan secara default. Aktifkan dengan mengatur variabel environment `CLAUDE_CODE_EXPERIMENTAL_AGENT_TEAMS` ke `1`, baik di environment shell Anda atau melalui settings.json:

```
{
  "env": {
    "CLAUDE_CODE_EXPERIMENTAL_AGENT_TEAMS": "1"
  }
}
```

Mulai tim agent pertama Anda

Setelah mengaktifkan tim agent, beri tahu Claude untuk membuat tim agent dan jelaskan tugas dan struktur tim yang Anda inginkan dalam bahasa alami. Claude membuat tim, menelurkan rekan tim, dan mengoordinasikan pekerjaan berdasarkan prompt Anda.

Contoh ini bekerja dengan baik karena tiga peran independen dan dapat mengeksplorasi masalah tanpa menunggu satu sama lain:

```
Saya merancang alat CLI yang membantu developer melacak komentar TODO di seluruh
codebase mereka. Buat tim agent untuk mengeksplorasi ini dari sudut berbeda: satu
rekan tim pada UX, satu pada arsitektur teknis, satu memainkan devil's advocate.
```

Dari sana, Claude membuat tim dengan daftar tugas bersama, menelurkan rekan tim untuk setiap perspektif, membuat mereka mengeksplorasi masalah, mensintesis temuan, dan mencoba membersihkan tim ketika selesai.

Terminal lead mencantumkan semua rekan tim dan apa yang mereka kerjakan. Gunakan Shift+Down untuk bersiklus melalui rekan tim dan kirim pesan kepada mereka secara langsung. Setelah rekan tim terakhir, Shift+Down membungkus kembali ke lead.

Jika Anda ingin setiap rekan tim di split pane-nya sendiri, lihat Pilih mode tampilan.

Kontrol tim agent Anda

Beri tahu lead apa yang Anda inginkan dalam bahasa alami. Ini menangani koordinasi tim, penugasan tugas, dan delegasi berdasarkan instruksi Anda.

Pilih mode tampilan

Tim agent mendukung dua mode tampilan:

- **In-process**: semua rekan tim berjalan di dalam terminal utama Anda. Gunakan Shift+Down untuk bersiklus melalui rekan tim dan ketik untuk mengirim pesan kepada mereka secara langsung. Bekerja di terminal apa pun, tidak ada setup tambahan yang diperlukan.
- **Split panes**: setiap rekan tim mendapat pane-nya sendiri. Anda dapat melihat output semua orang sekaligus dan klik ke dalam pane untuk berinteraksi secara langsung. Memerlukan tmux, atau iTerm2.

> **Note:**
>
> `tmux` memiliki keterbatasan yang diketahui pada sistem operasi tertentu dan secara tradisional bekerja paling baik di macOS. Menggunakan `tmux -CC` di iTerm2 adalah entry-point yang disarankan ke `tmux`.

Default adalah `"auto"`, yang menggunakan split panes jika Anda sudah berjalan di dalam session tmux, dan in-process sebaliknya. Pengaturan `"tmux"` mengaktifkan mode split-pane dan auto-detects apakah akan menggunakan tmux atau iTerm2 berdasarkan terminal Anda. Untuk mengganti, atur `teammateMode` di settings.json Anda:

```
{
  "teammateMode": "in-process"
}
```

Untuk memaksa mode in-process untuk satu session, teruskan sebagai flag:

```
claude --teammate-mode in-process
```

Mode split-pane memerlukan baik tmux atau iTerm2 dengan `it2` CLI. Untuk menginstal secara manual:

- **tmux**: instal melalui package manager sistem Anda. Lihat tmux wiki untuk instruksi spesifik platform.
- **iTerm2**: instal `it2` CLI, kemudian aktifkan Python API di **iTerm2 → Settings → General → Magic → Enable Python API**.

Tentukan rekan tim dan model

Claude memutuskan jumlah rekan tim untuk dihasilkan berdasarkan tugas Anda, atau Anda dapat menentukan dengan tepat apa yang Anda inginkan:

```
Buat tim dengan 4 rekan tim untuk refactor modul-modul ini secara paralel.
Gunakan Sonnet untuk setiap rekan tim.
```

Perlukan persetujuan rencana untuk rekan tim

Untuk tugas kompleks atau berisiko, Anda dapat memerlukan rekan tim untuk merencanakan sebelum mengimplementasikan. Rekan tim bekerja dalam mode rencana read-only sampai lead menyetujui pendekatan mereka:

```
Hasilkan rekan tim architect untuk refactor modul autentikasi.
Perlukan persetujuan rencana sebelum mereka membuat perubahan apa pun.
```

Ketika rekan tim selesai merencanakan, mereka mengirim permintaan persetujuan rencana ke lead. Lead meninjau rencana dan baik menyetujuinya atau menolaknya dengan umpan balik. Jika ditolak, rekan tim tetap dalam mode rencana, merevisi berdasarkan umpan balik, dan mengirimkan kembali. Setelah disetujui, rekan tim keluar dari mode rencana dan mulai implementasi.

Lead membuat keputusan persetujuan secara otonom. Untuk mempengaruhi penilaian lead, berikan kriteria dalam prompt Anda, seperti "hanya setujui rencana yang mencakup test coverage" atau "tolak rencana yang memodifikasi skema database."

Berbicara dengan rekan tim secara langsung

Setiap rekan tim adalah session Claude Code penuh dan independen. Anda dapat mengirim pesan ke rekan tim mana pun secara langsung untuk memberikan instruksi tambahan, mengajukan pertanyaan lanjutan, atau mengalihkan pendekatan mereka.

- **Mode in-process**: gunakan Shift+Down untuk bersiklus melalui rekan tim, kemudian ketik untuk mengirim pesan kepada mereka. Tekan Enter untuk melihat session rekan tim, kemudian Escape untuk mengganggu giliran mereka saat ini. Tekan Ctrl+T untuk toggle daftar tugas.
- **Mode split-pane**: klik ke dalam pane rekan tim untuk berinteraksi dengan session mereka secara langsung. Setiap rekan tim memiliki tampilan penuh dari terminal mereka sendiri.

Tetapkan dan klaim tugas

Daftar tugas bersama mengoordinasikan pekerjaan di seluruh tim. Lead membuat tugas dan rekan tim mengerjakannya. Tugas memiliki tiga status: pending, in progress, dan completed. Tugas juga dapat bergantung pada tugas lain: tugas pending dengan dependensi yang tidak terselesaikan tidak dapat diklaim sampai dependensi tersebut selesai.

Lead dapat menugaskan tugas secara eksplisit, atau rekan tim dapat self-claim:

- **Lead menugaskan**: beri tahu lead tugas mana yang diberikan kepada rekan tim mana
- **Self-claim**: setelah menyelesaikan tugas, rekan tim mengambil tugas unassigned, unblocked berikutnya sendiri

Klaim tugas menggunakan file locking untuk mencegah race conditions ketika beberapa rekan tim mencoba mengklaim tugas yang sama secara bersamaan.

Matikan rekan tim

Untuk mengakhiri session rekan tim dengan baik:

```
Minta rekan tim peneliti untuk shutdown
```

Lead mengirim permintaan shutdown. Rekan tim dapat menyetujui, keluar dengan baik, atau menolak dengan penjelasan.

Bersihkan tim

Ketika Anda selesai, minta lead untuk membersihkan:

```
Bersihkan tim
```

Ini menghapus sumber daya tim bersama. Ketika lead menjalankan cleanup, ia memeriksa rekan tim aktif dan gagal jika ada yang masih berjalan, jadi matikan mereka terlebih dahulu.

> **Warning:**
>
> Selalu gunakan lead untuk membersihkan. Rekan tim tidak boleh menjalankan cleanup karena konteks tim mereka mungkin tidak terselesaikan dengan benar, berpotensi meninggalkan sumber daya dalam keadaan tidak konsisten.

Terapkan quality gates dengan hooks

Gunakan hooks untuk menerapkan aturan ketika rekan tim menyelesaikan pekerjaan atau tugas selesai:

- `TeammateIdle` : berjalan ketika rekan tim akan idle. Keluar dengan kode 2 untuk mengirim umpan balik dan membuat rekan tim tetap bekerja.
- `TaskCompleted` : berjalan ketika tugas ditandai selesai. Keluar dengan kode 2 untuk mencegah penyelesaian dan mengirim umpan balik.

Bagaimana tim agent bekerja

Bagian ini mencakup arsitektur dan mekanik di balik tim agent. Jika Anda ingin mulai menggunakannya, lihat Kontrol tim agent Anda di atas.

Bagaimana Claude memulai tim agent

Ada dua cara tim agent dimulai:

- **Anda meminta tim**: berikan Claude tugas yang menguntungkan dari pekerjaan paralel dan secara eksplisit minta tim agent. Claude membuat satu berdasarkan instruksi Anda.
- **Claude mengusulkan tim**: jika Claude menentukan tugas Anda akan menguntungkan dari pekerjaan paralel, mungkin menyarankan membuat tim. Anda mengonfirmasi sebelum melanjutkan.

Dalam kedua kasus, Anda tetap mengendalikan. Claude tidak akan membuat tim tanpa persetujuan Anda.

Arsitektur

Tim agent terdiri dari:

Komponen	Peran
Team lead	Session Claude Code utama yang membuat tim, menelurkan rekan tim, dan mengoordinasikan pekerjaan
Rekan tim	Instance Claude Code terpisah yang masing-masing bekerja pada tugas yang ditugaskan
Daftar tugas	Daftar item pekerjaan bersama yang diklaim dan diselesaikan rekan tim
Mailbox	Sistem pesan untuk komunikasi antar agent

Lihat Pilih mode tampilan untuk opsi konfigurasi tampilan. Pesan rekan tim tiba di lead secara otomatis.

Sistem mengelola dependensi tugas secara otomatis. Ketika rekan tim menyelesaikan tugas yang tugas lain bergantung padanya, tugas yang diblokir membuka tanpa intervensi manual.

Tim dan tugas disimpan secara lokal:

- **Konfigurasi tim**: `~/.claude/teams/{team-name}/config.json`
- **Daftar tugas**: `~/.claude/tasks/{team-name}/`

Konfigurasi tim berisi array `members` dengan nama setiap rekan tim, agent ID, dan tipe agent. Rekan tim dapat membaca file ini untuk menemukan anggota tim lainnya.

Izin

Rekan tim dimulai dengan pengaturan izin lead. Jika lead berjalan dengan `--dangerously-skip-permissions`, semua rekan tim juga demikian. Setelah dihasilkan, Anda dapat mengubah mode rekan tim individual, tetapi Anda tidak dapat mengatur mode per-rekan tim pada waktu spawn.

Context dan komunikasi

Setiap rekan tim memiliki context window-nya sendiri. Ketika dihasilkan, rekan tim memuat konteks proyek yang sama seperti session reguler: CLAUDE.md, MCP servers, dan skills. Mereka juga menerima spawn prompt dari lead. Riwayat percakapan lead tidak terbawa.

Bagaimana rekan tim berbagi informasi:

- **Pengiriman pesan otomatis**: ketika rekan tim mengirim pesan, mereka dikirim-kan secara otomatis ke penerima. Lead tidak perlu polling untuk update.
- **Notifikasi idle**: ketika rekan tim selesai dan berhenti, mereka secara otomatis memberi tahu lead.
- **Daftar tugas bersama**: semua agent dapat melihat status tugas dan mengklaim pekerjaan yang tersedia.

Pesan rekan tim:

- **message**: kirim pesan ke satu rekan tim spesifik
- **broadcast**: kirim ke semua rekan tim secara bersamaan. Gunakan dengan hemat, karena biaya skala dengan ukuran tim.

Penggunaan token

Tim agent menggunakan token secara signifikan lebih banyak daripada satu session. Setiap rekan tim memiliki context window-nya sendiri, dan penggunaan token skala dengan jumlah rekan tim aktif. Untuk penelitian, review, dan pekerjaan fitur baru, token tambahan biasanya berharga. Untuk tugas rutin, satu session lebih cost-effective. Lihat biaya token tim agent untuk panduan penggunaan.

Contoh use case

Contoh-contoh ini menunjukkan bagaimana tim agent menangani tugas di mana eksplorasi paralel menambah nilai.

Jalankan code review paralel

Seorang reviewer tunggal cenderung tertarik pada satu jenis masalah pada satu waktu. Membagi kriteria review menjadi domain independen berarti keamanan, kinerja, dan test coverage semuanya mendapat perhatian menyeluruh secara bersamaan. Prompt menugaskan setiap rekan tim lensa yang berbeda sehingga mereka tidak tumpang tindih:

```
Buat tim agent untuk review PR #142. Hasilkan tiga reviewer:
- Satu fokus pada implikasi keamanan
- Satu memeriksa dampak kinerja
- Satu memvalidasi test coverage
Buat mereka masing-masing review dan laporkan temuan.
```

Setiap reviewer bekerja dari PR yang sama tetapi menerapkan filter berbeda. Lead mensintesis temuan di ketiga setelah mereka selesai.

Investigasi dengan hipotesis bersaing

Ketika akar penyebab tidak jelas, satu agent cenderung menemukan satu penjelasan yang masuk akal dan berhenti mencari. Prompt melawan ini dengan membuat rekan tim secara eksplisit adversarial: pekerjaan setiap orang bukan hanya menyelidiki teori mereka sendiri tetapi menantang yang lain.

```
Pengguna melaporkan aplikasi keluar setelah satu pesan alih-alih tetap terhubung.
Hasilkan 5 rekan tim agent untuk menyelidiki hipotesis berbeda. Buat mereka
berbicara
satu sama lain untuk mencoba membantah teori satu sama lain, seperti debat
ilmiah. Perbarui dokumen temuan dengan konsensus apa pun yang muncul.
```

Struktur debat adalah mekanisme kunci di sini. Investigasi sekuensial menderita dari anchoring: setelah satu teori dieksplorasi, investigasi berikutnya bias terhadapnya.

Dengan beberapa investigator independen secara aktif mencoba membantah satu sama lain, teori yang bertahan jauh lebih mungkin menjadi akar penyebab sebenarnya.

Best practices

Berikan rekan tim konteks yang cukup

Rekan tim memuat konteks proyek secara otomatis, termasuk CLAUDE.md, MCP servers, dan skills, tetapi mereka tidak mewarisi riwayat percakapan lead. Lihat Context dan komunikasi untuk detail. Sertakan detail spesifik tugas dalam spawn prompt:

```
Hasilkan rekan tim security reviewer dengan prompt: "Review modul autentikasi
di src/auth/ untuk kerentanan keamanan. Fokus pada penanganan token, manajemen
session, dan validasi input. Aplikasi menggunakan token JWT yang disimpan di
httpOnly cookies. Laporkan masalah apa pun dengan rating severity."
```

Pilih ukuran tim yang sesuai

Tidak ada batas keras pada jumlah rekan tim, tetapi batasan praktis berlaku:

- **Biaya token skala linear**: setiap rekan tim memiliki context window-nya sendiri dan mengkonsumsi token secara independen. Lihat biaya token tim agent untuk detail.
- **Overhead koordinasi meningkat**: lebih banyak rekan tim berarti lebih banyak komunikasi, koordinasi tugas, dan potensi konflik
- **Diminishing returns**: di luar titik tertentu, rekan tim tambahan tidak mempercepat pekerjaan secara proporsional

Mulai dengan 3-5 rekan tim untuk sebagian besar workflow. Ini menyeimbangkan pekerjaan paralel dengan koordinasi yang dapat dikelola. Contoh-contoh dalam panduan ini menggunakan 3-5 rekan tim karena rentang itu bekerja dengan baik di berbagai jenis tugas.

Memiliki 5-6 tasks per rekan tim membuat semua orang produktif tanpa context switching yang berlebihan. Jika Anda memiliki 15 tugas independen, 3 rekan tim adalah titik awal yang baik.

Skala naik hanya ketika pekerjaan benar-benar menguntungkan dari rekan tim bekerja secara bersamaan. Tiga rekan tim terfokus sering mengungguli lima yang tersebar.

Ukuran tugas dengan tepat

- **Terlalu kecil**: overhead koordinasi melebihi manfaat
- **Terlalu besar**: rekan tim bekerja terlalu lama tanpa check-in, meningkatkan risiko usaha yang terbuang

- **Tepat**: unit self-contained yang menghasilkan deliverable yang jelas, seperti fungsi, file test, atau review

> **Tip:**
>
> Lead memecah pekerjaan menjadi tugas dan menugaskan mereka ke rekan tim secara otomatis. Jika tidak membuat cukup tugas, minta untuk membagi pekerjaan menjadi potongan yang lebih kecil. Memiliki 5-6 tugas per rekan tim membuat semua orang produktif dan memungkinkan lead untuk menugaskan kembali pekerjaan jika seseorang terjebak.

Tunggu rekan tim selesai

Kadang-kadang lead mulai mengimplementasikan tugas sendiri alih-alih menunggu rekan tim. Jika Anda memperhatikan ini:

```
Tunggu rekan tim Anda menyelesaikan tugas mereka sebelum melanjutkan
```

Mulai dengan penelitian dan review

Jika Anda baru mengenal tim agent, mulai dengan tugas yang memiliki batas yang jelas dan tidak memerlukan penulisan kode: review PR, penelitian library, atau investigasi bug. Tugas-tugas ini menunjukkan nilai eksplorasi paralel tanpa tantangan koordinasi yang datang dengan implementasi paralel.

Hindari konflik file

Dua rekan tim mengedit file yang sama menyebabkan overwrites. Pecah pekerjaan sehingga setiap rekan tim memiliki set file berbeda.

Monitor dan kemudi

Periksa kemajuan rekan tim, alihkan pendekatan yang tidak berfungsi, dan sintesis temuan saat tiba. Membiarkan tim berjalan tanpa diawasi terlalu lama meningkatkan risiko usaha yang terbuang.

Troubleshooting

Rekan tim tidak muncul

Jika rekan tim tidak muncul setelah Anda meminta Claude untuk membuat tim:

- Dalam mode in-process, rekan tim mungkin sudah berjalan tetapi tidak terlihat. Tekan Shift+Down untuk bersiklus melalui rekan tim aktif.

- Periksa bahwa tugas yang Anda berikan Claude cukup kompleks untuk menjamin tim. Claude memutuskan apakah akan menelurkan rekan tim berdasarkan tugas.
- Jika Anda secara eksplisit meminta split panes, pastikan tmux diinstal dan tersedia di PATH Anda:

```
which tmux
```

- Untuk iTerm2, verifikasi `it2` CLI diinstal dan Python API diaktifkan di preferensi iTerm2.

Terlalu banyak permission prompts

Permintaan izin rekan tim naik ke lead, yang dapat menciptakan gesekan. Pre-approve operasi umum di pengaturan izin Anda sebelum menelurkan rekan tim untuk mengurangi gangguan.

Rekan tim berhenti pada error

Rekan tim dapat berhenti setelah mengalami error alih-alih pulih. Periksa output mereka menggunakan Shift+Down dalam mode in-process atau dengan mengklik pane dalam mode split, kemudian baik:

- Berikan instruksi tambahan kepada mereka secara langsung
- Hasilkan rekan tim pengganti untuk melanjutkan pekerjaan

Lead shutdown sebelum pekerjaan selesai

Lead dapat memutuskan tim selesai sebelum semua tugas benar-benar selesai. Jika ini terjadi, beri tahu untuk terus. Anda juga dapat memberi tahu lead untuk menunggu rekan tim selesai sebelum melanjutkan jika mulai melakukan pekerjaan alih-alih mendelegasikan.

Orphaned tmux sessions

Jika session tmux bertahan setelah tim berakhir, mungkin tidak sepenuhnya dibersihkan. Daftar session dan bunuh yang dibuat oleh tim:

```
tmux ls
tmux kill-session -t <session-name>
```

Keterbatasan

Tim agent bersifat eksperimental. Keterbatasan saat ini untuk diketahui:

- **Tidak ada session resumption dengan rekan tim in-process**: `/resume` dan `/rewind` tidak mengembalikan rekan tim in-process. Setelah melanjutkan session, lead dapat mencoba mengirim pesan ke rekan tim yang tidak lagi ada. Jika ini terjadi, beri tahu lead untuk menelurkan rekan tim baru.
- **Status tugas dapat tertinggal**: rekan tim kadang-kadang gagal menandai tugas sebagai selesai, yang memblokir tugas dependen. Jika tugas tampak terjebak, periksa apakah pekerjaan benar-benar selesai dan perbarui status tugas secara manual atau beri tahu lead untuk mendorong rekan tim.
- **Shutdown dapat lambat**: rekan tim menyelesaikan permintaan atau tool call saat ini sebelum shutdown, yang dapat memakan waktu.
- **Satu tim per session**: lead hanya dapat mengelola satu tim pada satu waktu. Bersihkan tim saat ini sebelum memulai yang baru.
- **Tidak ada tim bersarang**: rekan tim tidak dapat menelurkan tim atau rekan tim mereka sendiri. Hanya lead yang dapat mengelola tim.
- **Lead tetap**: session yang membuat tim adalah lead seumur hidupnya. Anda tidak dapat mempromosikan rekan tim ke lead atau mentransfer kepemimpinan.
- **Izin ditetapkan pada spawn**: semua rekan tim dimulai dengan mode izin lead. Anda dapat mengubah mode rekan tim individual setelah spawn, tetapi Anda tidak dapat mengatur mode per-rekan tim pada waktu spawn.
- **Split panes memerlukan tmux atau iTerm2**: mode in-process default bekerja di terminal apa pun. Mode split-pane tidak didukung di integrated terminal VS Code, Windows Terminal, atau Ghostty.

> **Tip:**
>
> `CLAUDE.md` **bekerja secara normal**: rekan tim membaca file `CLAUDE.md` dari direktori kerja mereka. Gunakan ini untuk memberikan panduan spesifik proyek ke semua rekan tim.

Langkah berikutnya

Jelajahi pendekatan terkait untuk pekerjaan paralel dan delegasi:

- **Delegasi ringan**: subagents menelurkan agent pembantu untuk penelitian atau verifikasi dalam session Anda, lebih baik untuk tugas yang tidak memerlukan koordinasi inter-agent

- **Session paralel manual**: Git worktrees memungkinkan Anda menjalankan beberapa session Claude Code sendiri tanpa koordinasi tim otomatis
- **Bandingkan pendekatan**: lihat perbandingan subagent vs tim agent untuk rincian side-by-side

Jalankan Claude Code secara programatis

> *Gunakan Agent SDK untuk menjalankan Claude Code secara programatis dari CLI, Python, atau TypeScript.*

Agent SDK memberikan Anda alat yang sama, loop agen, dan manajemen konteks yang mendukung Claude Code. Tersedia sebagai CLI untuk skrip dan CI/CD, atau sebagai paket Python dan TypeScript untuk kontrol programatis penuh.

> **Note:**
>
> CLI sebelumnya disebut "headless mode." Bendera `-p` dan semua opsi CLI bekerja dengan cara yang sama.

Untuk menjalankan Claude Code secara programatis dari CLI, berikan `-p` dengan prompt Anda dan opsi CLI apa pun:

```
claude -p "Find and fix the bug in auth.py" --allowedTools "Read,Edit,Bash"
```

Halaman ini mencakup penggunaan Agent SDK melalui CLI (`claude -p`). Untuk paket SDK Python dan TypeScript dengan output terstruktur, callback persetujuan alat, dan objek pesan asli, lihat dokumentasi Agent SDK lengkap.

Penggunaan dasar

Tambahkan bendera `-p` (atau `--print`) ke perintah `claude` apa pun untuk menjalankannya secara non-interaktif. Semua opsi CLI bekerja dengan `-p` , termasuk:

- `--continue` untuk melanjutkan percakapan
- `--allowedTools` untuk persetujuan otomatis alat
- `--output-format` untuk output terstruktur

Contoh ini menanyakan Claude tentang basis kode Anda dan mencetak respons:

```
claude -p "What does the auth module do?"
```

Contoh

Contoh-contoh ini menyoroti pola CLI umum.

Dapatkan output terstruktur

Gunakan `--output-format` untuk mengontrol bagaimana respons dikembalikan:

- `text` (default): output teks biasa
- `json` : JSON terstruktur dengan hasil, ID sesi, dan metadata
- `stream-json` : JSON yang dibatasi baris baru untuk streaming real-time

Contoh ini mengembalikan ringkasan proyek sebagai JSON dengan metadata sesi, dengan hasil teks di bidang `result` :

```
claude -p "Summarize this project" --output-format json
```

Untuk mendapatkan output yang sesuai dengan skema tertentu, gunakan `--output-format json` dengan `--json-schema` dan definisi JSON Schema. Respons mencakup metadata tentang permintaan (ID sesi, penggunaan, dll.) dengan output terstruktur di bidang `structured_output` .

Contoh ini mengekstrak nama fungsi dan mengembalikannya sebagai array string:

```
claude -p "Extract the main function names from auth.py" \
  --output-format json \
  --json-schema '{"type":"object","properties":{"functions":
{"type":"array","items":{"type":"string"}}},"required":["functions"]}'
```

> **Tip:**
>
> Gunakan alat seperti jq untuk mengurai respons dan mengekstrak bidang tertentu:

```
## Extract the text result
claude -p "Summarize this project" --output-format json | jq -r '.result'

## Extract structured output
claude -p "Extract function names from auth.py" \
  --output-format json \
  --json-schema '{"type":"object","properties":{"functions":
{"type":"array","items":{"type":"string"}}},"required":["functions"]}' \
  | jq '.structured_output'
```

Streaming respons

Gunakan `--output-format stream-json` dengan `--verbose` dan `--include-partial-messages` untuk menerima token saat dihasilkan. Setiap baris adalah objek JSON yang mewakili acara:

```
claude -p "Explain recursion" --output-format stream-json --verbose --include-
partial-messages
```

Contoh berikut menggunakan jq untuk memfilter delta teks dan menampilkan hanya teks streaming. Bendera `-r` menampilkan string mentah (tanpa tanda kutip) dan `-j` bergabung tanpa baris baru sehingga token streaming terus menerus:

```
claude -p "Write a poem" --output-format stream-json --verbose --include-partial-
messages | \
  jq -rj 'select(.type == "stream_event" and .event.delta.type? == "text_delta")
| .event.delta.text'
```

Untuk streaming programatis dengan callback dan objek pesan, lihat Stream responses in real-time dalam dokumentasi Agent SDK.

Persetujuan otomatis alat

Gunakan `--allowedTools` untuk membiarkan Claude menggunakan alat tertentu tanpa meminta. Contoh ini menjalankan suite pengujian dan memperbaiki kegagalan, memungkinkan Claude untuk menjalankan perintah Bash dan membaca/mengedit file tanpa meminta izin:

```
claude -p "Run the test suite and fix any failures" \
  --allowedTools "Bash,Read,Edit"
```

Buat komit

Contoh ini meninjau perubahan yang dipentaskan dan membuat komit dengan pesan yang sesuai:

```
claude -p "Look at my staged changes and create an appropriate commit" \
  --allowedTools "Bash(git diff *),Bash(git log *),Bash(git status *),Bash(git
commit *)"
```

Bendera `--allowedTools` menggunakan sintaks aturan izin. Spasi di akhir `*` memungkinkan pencocokan awalan, jadi `Bash(git diff *)` memungkinkan perintah apa pun yang dimulai dengan `git diff`. Spasi sebelum `*` penting: tanpanya, `Bash(git diff*)` juga akan cocok dengan `git diff-index`.

> **Note:**
>
> skills yang dipanggil pengguna seperti `/commit` dan perintah bawaan hanya tersedia dalam mode interaktif. Dalam mode `-p`, jelaskan tugas yang ingin Anda capai.

Sesuaikan prompt sistem

Gunakan `--append-system-prompt` untuk menambahkan instruksi sambil mempertahankan perilaku default Claude Code. Contoh ini menyalurkan diff PR ke Claude dan menginstruksikannya untuk meninjau kerentanan keamanan:

```
gh pr diff "$1" | claude -p \
  --append-system-prompt "You are a security engineer. Review for
vulnerabilities." \
  --output-format json
```

Lihat system prompt flags untuk opsi lainnya termasuk `--system-prompt` untuk sepenuhnya mengganti prompt default.

Lanjutkan percakapan

Gunakan `--continue` untuk melanjutkan percakapan terbaru, atau `--resume` dengan ID sesi untuk melanjutkan percakapan tertentu. Contoh ini menjalankan tinjauan, kemudian mengirim prompt tindak lanjut:

```
## First request
claude -p "Review this codebase for performance issues"

## Continue the most recent conversation
claude -p "Now focus on the database queries" --continue
claude -p "Generate a summary of all issues found" --continue
```

Jika Anda menjalankan beberapa percakapan, tangkap ID sesi untuk melanjutkan percakapan tertentu:

```
session_id=$(claude -p "Start a review" --output-format json | jq -r
'.session_id')
claude -p "Continue that review" --resume "$session_id"
```

Langkah berikutnya

- Agent SDK quickstart: bangun agen pertama Anda dengan Python atau TypeScript
- CLI reference: semua bendera dan opsi CLI
- GitHub Actions: gunakan Agent SDK dalam alur kerja GitHub
- GitLab CI/CD: gunakan Agent SDK dalam pipeline GitLab

Lanjutkan sesi lokal dari perangkat apa pun dengan Remote Control

> *Lanjutkan sesi Claude Code lokal dari ponsel, tablet, atau browser apa pun menggunakan Remote Control. Bekerja dengan claude.ai/code dan aplikasi Claude mobile.*

Note:

Remote Control tersedia di semua paket. Admin Tim dan Enterprise harus terlebih dahulu mengaktifkan Claude Code di pengaturan admin.

Remote Control menghubungkan claude.ai/code atau aplikasi Claude untuk iOS dan Android ke sesi Claude Code yang berjalan di mesin Anda. Mulai tugas di meja Anda, kemudian lanjutkan dari ponsel Anda di sofa atau browser di komputer lain.

Ketika Anda memulai sesi Remote Control di mesin Anda, Claude terus berjalan secara lokal sepanjang waktu, jadi tidak ada yang pindah ke cloud. Dengan Remote Control Anda dapat:

- **Gunakan lingkungan lokal penuh Anda dari jarak jauh**: sistem file, MCP servers, alat, dan konfigurasi proyek Anda tetap tersedia
- **Bekerja dari kedua permukaan sekaligus**: percakapan tetap tersinkronisasi di semua perangkat yang terhubung, sehingga Anda dapat mengirim pesan dari terminal, browser, dan ponsel Anda secara bergantian
- **Bertahan dari gangguan**: jika laptop Anda tidur atau jaringan Anda terputus, sesi akan terhubung kembali secara otomatis ketika mesin Anda kembali online

Tidak seperti Claude Code di web, yang berjalan di infrastruktur cloud, sesi Remote Control berjalan langsung di mesin Anda dan berinteraksi dengan sistem file lokal Anda. Antarmuka web dan mobile hanyalah jendela ke sesi lokal tersebut.

Note:

Remote Control memerlukan Claude Code v2.1.51 atau lebih baru. Periksa versi Anda dengan `claude --version`.

Halaman ini mencakup pengaturan, cara memulai dan terhubung ke sesi, dan bagaimana Remote Control dibandingkan dengan Claude Code di web.

Persyaratan

Sebelum menggunakan Remote Control, konfirmasi bahwa lingkungan Anda memenuhi kondisi berikut:

- **Langganan**: tersedia di paket Pro, Max, Tim, dan Enterprise. Admin Tim dan Enterprise harus terlebih dahulu mengaktifkan Claude Code di pengaturan admin. Kunci API tidak didukung.
- **Autentikasi**: jalankan `claude` dan gunakan `/login` untuk masuk melalui claude.ai jika Anda belum melakukannya.
- **Kepercayaan ruang kerja**: jalankan `claude` di direktori proyek Anda setidaknya sekali untuk menerima dialog kepercayaan ruang kerja.

Mulai sesi Remote Control

Anda dapat memulai sesi baru langsung di Remote Control, atau menghubungkan sesi yang sudah berjalan.

Sesi baru

Navigasikan ke direktori proyek Anda dan jalankan:

```
claude remote-control
```

Proses tetap berjalan di terminal Anda, menunggu koneksi jarak jauh. Ini menampilkan URL sesi yang dapat Anda gunakan untuk terhubung dari perangkat lain, dan Anda dapat menekan spacebar untuk menampilkan kode QR untuk akses cepat dari ponsel Anda. Saat sesi jarak jauh aktif, terminal menampilkan status koneksi dan aktivitas alat.

Perintah ini mendukung flag berikut:

- `--name "My Project"` : atur judul sesi khusus yang terlihat dalam daftar sesi di claude.ai/code. Anda juga dapat meneruskan nama sebagai argumen posisional: `claude remote-control "My Project"`
- `--verbose` : tampilkan log koneksi dan sesi terperinci
- `--sandbox` / `--no-sandbox` : aktifkan atau nonaktifkan sandboxing untuk isolasi sistem file dan jaringan selama sesi. Sandboxing dimatikan secara default.

Dari sesi yang ada

Jika Anda sudah dalam sesi Claude Code dan ingin melanjutkannya dari jarak jauh, gunakan perintah `/remote-control` (atau `/rc`):

```
/remote-control
```

Teruskan nama sebagai argumen untuk menetapkan judul sesi khusus:

```
/remote-control My Project
```

Ini memulai sesi Remote Control yang membawa riwayat percakapan Anda saat ini dan menampilkan URL sesi dan kode QR yang dapat Anda gunakan untuk terhubung dari perangkat lain. Flag `--verbose`, `--sandbox`, dan `--no-sandbox` tidak tersedia dengan perintah ini.

Terhubung dari perangkat lain

Setelah sesi Remote Control aktif, Anda memiliki beberapa cara untuk terhubung dari perangkat lain:

- **Buka URL sesi** di browser apa pun untuk langsung ke sesi di claude.ai/code. Baik `claude remote-control` maupun `/remote-control` menampilkan URL ini di terminal.
- **Pindai kode QR** yang ditampilkan di samping URL sesi untuk membukanya langsung di aplikasi Claude. Dengan `claude remote-control`, tekan spacebar untuk mengalihkan tampilan kode QR.
- **Buka claude.ai/code atau aplikasi Claude** dan temukan sesi berdasarkan nama dalam daftar sesi. Sesi Remote Control menampilkan ikon komputer dengan titik status hijau saat online.

Sesi jarak jauh mengambil namanya dari argumen `--name` (atau nama yang diteruskan ke `/remote-control`), pesan terakhir Anda, nilai `/rename` Anda, atau “Remote Control session” jika tidak ada riwayat percakapan. Jika lingkungan sudah memiliki sesi aktif, Anda akan ditanya apakah akan melanjutkannya atau memulai yang baru.

Jika Anda belum memiliki aplikasi Claude, gunakan perintah `/mobile` di dalam Claude Code untuk menampilkan kode QR unduhan untuk iOS atau Android.

Aktifkan Remote Control untuk semua sesi

Secara default, Remote Control hanya diaktifkan ketika Anda secara eksplisit menjalankan `claude remote-control` atau `/remote-control`. Untuk mengaktifkannya secara otomatis untuk setiap sesi, jalankan `/config` di dalam Claude Code dan atur **Enable Remote Control for all sessions** ke `true`. Atur kembali ke `false` untuk menonaktifkan.

Setiap instans Claude Code mendukung satu sesi jarak jauh pada satu waktu. Jika Anda menjalankan beberapa instans, masing-masing mendapatkan lingkungan dan sesinya sendiri.

Koneksi dan keamanan

Sesi Claude Code lokal Anda hanya membuat permintaan HTTPS keluar dan tidak pernah membuka port masuk di mesin Anda. Ketika Anda memulai Remote Control, sesi tersebut mendaftar dengan API Anthropic dan melakukan polling untuk pekerjaan. Ketika Anda terhubung dari perangkat lain, server merutekan pesan antara klien web atau mobile dan sesi lokal Anda melalui koneksi streaming.

Semua lalu lintas berjalan melalui API Anthropic melalui TLS, keamanan transportasi yang sama seperti sesi Claude Code apa pun. Koneksi menggunakan beberapa kredensial berumur pendek, masing-masing dibatasi untuk satu tujuan dan kedaluwarsa secara independen.

Remote Control vs Claude Code di web

Remote Control dan Claude Code di web keduanya menggunakan antarmuka claude.ai/code. Perbedaan utamanya adalah di mana sesi berjalan: Remote Control dieksekusi di mesin Anda, jadi MCP servers lokal, alat, dan konfigurasi proyek Anda tetap tersedia. Claude Code di web dieksekusi di infrastruktur cloud yang dikelola Anthropic.

Gunakan Remote Control ketika Anda sedang dalam pekerjaan lokal dan ingin terus melanjutkan dari perangkat lain. Gunakan Claude Code di web ketika Anda ingin memulai tugas tanpa pengaturan lokal apa pun, bekerja pada repo yang tidak Anda miliki, atau menjalankan beberapa tugas secara paralel.

Batasan

- **Satu sesi jarak jauh pada satu waktu**: setiap sesi Claude Code mendukung satu koneksi jarak jauh.

- **Terminal harus tetap terbuka**: Remote Control berjalan sebagai proses lokal. Jika Anda menutup terminal atau menghentikan proses `claude`, sesi berakhir. Jalankan `claude remote-control` lagi untuk memulai yang baru.
- **Pemadaman jaringan yang diperpanjang**: jika mesin Anda aktif tetapi tidak dapat menjangkau jaringan selama lebih dari kira-kira 10 menit, sesi habis waktu dan proses keluar. Jalankan `claude remote-control` lagi untuk memulai sesi baru.

Sumber daya terkait

- Claude Code di web: jalankan sesi di lingkungan cloud yang dikelola Anthropic alih-alih di mesin Anda
- Autentikasi: atur `/login` dan kelola kredensial untuk claude.ai
- Referensi CLI: daftar lengkap flag dan perintah termasuk `claude remote-control`
- Keamanan: bagaimana sesi Remote Control sesuai dengan model keamanan Claude Code
- Penggunaan data: data apa yang mengalir melalui API Anthropic selama sesi lokal dan jarak jauh

Jalankan prompt sesuai jadwal

> *Gunakan /loop dan alat penjadwalan cron untuk menjalankan prompt berulang kali, polling status, atau mengatur pengingat sekali jalan dalam sesi Claude Code.*

Note:

Tugas terjadwal memerlukan Claude Code v2.1.72 atau lebih baru. Periksa versi Anda dengan `claude --version`.

Tugas terjadwal memungkinkan Claude menjalankan kembali prompt secara otomatis pada interval tertentu. Gunakan untuk polling deployment, mengawasi PR, memeriksa build yang berjalan lama, atau mengingatkan diri sendiri untuk melakukan sesuatu nanti dalam sesi.

Tugas bersifat session-scoped: mereka hidup dalam proses Claude Code saat ini dan hilang saat Anda keluar. Untuk penjadwalan yang tahan lama yang bertahan setelah restart dan berjalan tanpa sesi terminal aktif, lihat Desktop scheduled tasks atau GitHub Actions.

Jadwalkan prompt berulang dengan /loop

Skill `/loop` bundled skill adalah cara tercepat untuk menjadwalkan prompt berulang. Berikan interval opsional dan prompt, dan Claude menyiapkan pekerjaan cron yang berjalan di latar belakang sementara sesi tetap terbuka.

```
/loop 5m check if the deployment finished and tell me what happened
```

Claude mengurai interval, mengonversinya ke ekspresi cron, menjadwalkan pekerjaan, dan mengonfirmasi cadence dan ID pekerjaan.

Sintaks interval

Interval bersifat opsional. Anda dapat memimpinnya, mengikutinya, atau meninggalkannya sepenuhnya.

Form	Contoh	Interval yang diurai
Token terdepan	`/loop 30m check the build`	setiap 30 menit
Klausa `every` di belakang	`/loop check the build every 2 hours`	setiap 2 jam
Tidak ada interval	`/loop check the build`	default setiap 10 menit

Unit yang didukung adalah `s` untuk detik, `m` untuk menit, `h` untuk jam, dan `d` untuk hari. Detik dibulatkan ke menit terdekat karena cron memiliki granularitas satu menit. Interval yang tidak terbagi rata ke dalam unit mereka, seperti `7m` atau `90m`, dibulatkan ke interval yang bersih terdekat dan Claude memberi tahu Anda apa yang dipilihnya.

Loop di atas perintah lain

Prompt terjadwal itu sendiri dapat berupa invokasi perintah atau skill. Ini berguna untuk menjalankan kembali alur kerja yang telah Anda paket.

```
/loop 20m /review-pr 1234
```

Setiap kali pekerjaan berjalan, Claude menjalankan `/review-pr 1234` seolah-olah Anda telah mengetiknya.

Atur pengingat sekali jalan

Untuk pengingat sekali jalan, jelaskan apa yang Anda inginkan dalam bahasa alami daripada menggunakan `/loop`. Claude menjadwalkan tugas single-fire yang menghapus dirinya sendiri setelah berjalan.

```
remind me at 3pm to push the release branch
```

```
in 45 minutes, check whether the integration tests passed
```

Claude menyematkan waktu api ke menit dan jam tertentu menggunakan ekspresi cron dan mengonfirmasi kapan akan berjalan.

Kelola tugas terjadwal

Minta Claude dalam bahasa alami untuk membuat daftar atau membatalkan tugas, atau referensikan alat yang mendasarinya secara langsung.

```
what scheduled tasks do I have?
```

```
cancel the deploy check job
```

Di balik layar, Claude menggunakan alat-alat ini:

Alat	Tujuan
`CronCreate`	Jadwalkan tugas baru. Menerima ekspresi cron 5-field, prompt untuk dijalankan, dan apakah itu berulang atau berjalan sekali.
`CronList`	Buat daftar semua tugas terjadwal dengan ID, jadwal, dan prompt mereka.
`CronDelete`	Batalkan tugas berdasarkan ID.

Setiap tugas terjadwal memiliki ID 8-karakter yang dapat Anda berikan ke `CronDelete`. Sesi dapat menampung hingga 50 tugas terjadwal sekaligus.

Bagaimana tugas terjadwal berjalan

Penjadwal memeriksa setiap detik untuk tugas yang jatuh tempo dan memasukkannya ke antrian dengan prioritas rendah. Prompt terjadwal berjalan di antara giliran Anda, bukan saat Claude sedang merespons. Jika Claude sibuk saat tugas jatuh tempo, prompt menunggu sampai giliran saat ini berakhir.

Semua waktu ditafsirkan dalam zona waktu lokal Anda. Ekspresi cron seperti `0 9 * * *` berarti 9am di mana pun Anda menjalankan Claude Code, bukan UTC.

Jitter

Untuk menghindari setiap sesi mengenai API pada momen dinding jam yang sama, penjadwal menambahkan offset deterministik kecil untuk waktu api:

- Tugas berulang berjalan hingga 10% dari periode mereka terlambat, dibatasi pada 15 menit. Pekerjaan per jam mungkin berjalan di mana saja dari `:00` hingga `:06`.
- Tugas sekali jalan yang dijadwalkan untuk bagian atas atau bawah jam berjalan hingga 90 detik lebih awal.

Offset berasal dari ID tugas, jadi tugas yang sama selalu mendapatkan offset yang sama. Jika waktu yang tepat penting, pilih menit yang bukan `:00` atau `:30`, misalnya `3 9 * * *` daripada `0 9 * * *`, dan jitter sekali jalan tidak akan berlaku.

Kedaluwarsa tiga hari

Tugas berulang secara otomatis kedaluwarsa 3 hari setelah pembuatan. Tugas berjalan satu kali terakhir, kemudian menghapus dirinya sendiri. Ini membatasi berapa lama loop yang terlupakan dapat berjalan. Jika Anda memerlukan tugas berulang untuk bertahan lebih lama, batalkan dan buat ulang sebelum kedaluwarsa, atau gunakan Desktop scheduled tasks untuk penjadwalan yang tahan lama.

Referensi ekspresi cron

`CronCreate` menerima ekspresi cron 5-field standar: `minute hour day-of-month month day-of-week`. Semua field mendukung wildcard (`*`), nilai tunggal (`5`), langkah (`*/15`), rentang (`1-5`), dan daftar yang dipisahkan koma (`1,15,30`).

Contoh	Arti
`*/5 * * * *`	Setiap 5 menit
`0 * * * *`	Setiap jam pada jam
`7 * * * *`	Setiap jam pada 7 menit lewat
`0 9 * * *`	Setiap hari pada jam 9 pagi lokal
`0 9 * * 1-5`	Hari kerja pada jam 9 pagi lokal
`30 14 15 3 *`	15 Maret pada jam 2:30 sore lokal

Day-of-week menggunakan `0` atau `7` untuk Minggu hingga `6` untuk Sabtu. Sintaks yang diperluas seperti `L`, `W`, `?`, dan alias nama seperti `MON` atau `JAN` tidak didukung.

Ketika hari-bulan dan hari-minggu keduanya dibatasi, tanggal cocok jika salah satu field cocok. Ini mengikuti semantik vixie-cron standar.

Nonaktifkan tugas terjadwal

Atur `CLAUDE_CODE_DISABLE_CRON=1` di lingkungan Anda untuk menonaktifkan penjadwal sepenuhnya. Alat cron dan `/loop` menjadi tidak tersedia, dan tugas yang sudah terjadwal berhenti berjalan. Lihat Environment variables untuk daftar lengkap flag disable.

Keterbatasan

Penjadwalan session-scoped memiliki batasan yang melekat:

- Tugas hanya berjalan saat Claude Code berjalan dan idle. Menutup terminal atau membiarkan sesi keluar membatalkan semuanya.
- Tidak ada catch-up untuk api yang terlewat. Jika waktu terjadwal tugas berlalu saat Claude sibuk dengan permintaan yang berjalan lama, itu berjalan sekali saat Claude menjadi idle, bukan sekali per interval yang terlewat.
- Tidak ada persistensi di seluruh restart. Memulai ulang Claude Code menghapus semua tugas session-scoped.

Untuk otomasi yang didorong cron yang perlu berjalan tanpa pengawasan, gunakan GitHub Actions workflow dengan pemicu `schedule`, atau Desktop scheduled tasks jika Anda menginginkan alur pengaturan grafis.

Code Review

> *Siapkan ulasan PR otomatis yang menangkap kesalahan logika, kerentanan keamanan, dan regresi menggunakan analisis multi-agen dari seluruh basis kode Anda*

Note:

Code Review sedang dalam pratinjau penelitian, tersedia untuk langganan Teams dan Enterprise. Tidak tersedia untuk organisasi dengan Zero Data Retention yang diaktifkan.

Code Review menganalisis permintaan tarik GitHub Anda dan memposting temuan sebagai komentar sebaris pada baris kode tempat ditemukannya masalah. Armada agen khusus memeriksa perubahan kode dalam konteks basis kode lengkap Anda, mencari kesalahan logika, kerentanan keamanan, kasus tepi yang rusak, dan regresi halus.

Temuan diberi tag berdasarkan tingkat keparahan dan tidak menyetujui atau memblokir PR Anda, sehingga alur kerja ulasan yang ada tetap utuh. Anda dapat menyesuaikan apa yang Claude tandai dengan menambahkan file `CLAUDE.md` atau `REVIEW.md` ke repositori Anda.

Untuk menjalankan Claude di infrastruktur CI Anda sendiri alih-alih layanan terkelola ini, lihat GitHub Actions atau GitLab CI/CD.

Halaman ini mencakup:

- Cara kerja ulasan
- Penyiapan
- Menyesuaikan ulasan dengan `CLAUDE.md` dan `REVIEW.md`
- Harga

Cara kerja ulasan

Setelah admin mengaktifkan Code Review untuk organisasi Anda, ulasan berjalan secara otomatis ketika permintaan tarik dibuka atau diperbarui. Beberapa agen menganalisis diff dan kode sekitarnya secara paralel pada infrastruktur Anthropic. Setiap agen mencari kelas masalah yang berbeda, kemudian langkah verifikasi memeriksa kandidat terhadap perilaku kode aktual untuk menyaring positif palsu. Hasilnya dideduplikasi, diurutkan

berdasarkan tingkat keparahan, dan diposting sebagai komentar sebaris pada baris spesifik tempat masalah ditemukan. Jika tidak ada masalah yang ditemukan, Claude memposting komentar konfirmasi singkat pada PR.

Ulasan diskalakan dalam biaya dengan ukuran dan kompleksitas PR, selesai rata-rata dalam 20 menit. Admin dapat memantau aktivitas ulasan dan pengeluaran melalui dasbor analitik.

Tingkat keparahan

Setiap temuan diberi tag dengan tingkat keparahan:

Penanda	Keparahan	Arti
🔴	Normal	Bug yang harus diperbaiki sebelum penggabungan
🟡	Nit	Masalah kecil, layak diperbaiki tetapi tidak memblokir
🟣	Sudah ada sebelumnya	Bug yang ada di basis kode tetapi tidak diperkenalkan oleh PR ini

Temuan mencakup bagian penalaran yang dapat diperluas yang dapat Anda perluas untuk memahami mengapa Claude menandai masalah dan bagaimana Claude memverifikasi masalah.

Apa yang diperiksa Code Review

Secara default, Code Review berfokus pada kebenaran: bug yang akan merusak produksi, bukan preferensi pemformatan atau cakupan pengujian yang hilang. Anda dapat memperluas apa yang diperiksa dengan menambahkan file panduan ke repositori Anda.

Siapkan Code Review

Admin mengaktifkan Code Review sekali untuk organisasi dan memilih repositori mana yang akan disertakan.

Step 1: Buka pengaturan admin Claude Code

Buka claude.ai/admin-settings/claude-code dan temukan bagian Code Review. Anda memerlukan akses admin ke organisasi Claude Anda dan izin untuk memasang GitHub Apps di organisasi GitHub Anda.

Step 2: Mulai penyiapan

Klik **Setup**. Ini memulai alur instalasi GitHub App.

Step 3: Pasang Claude GitHub App

Ikuti petunjuk untuk memasang Claude GitHub App ke organisasi GitHub Anda. Aplikasi meminta izin repositori ini:

- **Contents**: baca dan tulis
- **Issues**: baca dan tulis
- **Pull requests**: baca dan tulis

Code Review menggunakan akses baca ke konten dan akses tulis ke permintaan tarik. Kumpulan izin yang lebih luas juga mendukung GitHub Actions jika Anda mengaktifkannya nanti.

Step 4: Pilih repositori

Pilih repositori mana yang akan diaktifkan untuk Code Review. Jika Anda tidak melihat repositori, pastikan Anda memberikan akses Claude GitHub App ke repositori tersebut selama instalasi. Anda dapat menambahkan lebih banyak repositori nanti.

Step 5: Atur pemicu ulasan per repo

Setelah penyiapan selesai, bagian Code Review menampilkan repositori Anda dalam tabel. Untuk setiap repositori, gunakan dropdown untuk memilih kapan ulasan berjalan:

- **Setelah pembuatan PR saja**: ulasan berjalan sekali ketika PR dibuka atau ditandai siap untuk ditinjau
- **Setelah setiap push ke cabang PR**: ulasan berjalan pada setiap push, menangkap masalah baru saat PR berkembang dan secara otomatis menyelesaikan thread ketika Anda memperbaiki masalah yang ditandai

Meninjau pada setiap push menjalankan lebih banyak ulasan dan biaya lebih banyak. Mulai dengan pembuatan PR saja dan beralih ke on-push untuk repo tempat Anda menginginkan cakupan berkelanjutan dan pembersihan thread otomatis.

Tabel repositori juga menampilkan biaya rata-rata per ulasan untuk setiap repo berdasarkan aktivitas terbaru. Gunakan menu tindakan baris untuk mengaktifkan atau menonaktifkan Code Review per repositori, atau untuk menghapus repositori sepenuhnya.

Untuk memverifikasi penyiapan, buka PR pengujian. Jalankan pemeriksaan bernama **Claude Code Review** muncul dalam beberapa menit. Jika tidak, konfirmasi bahwa repositori terdaftar di pengaturan admin Anda dan Claude GitHub App memiliki akses ke repositori tersebut.

Sesuaikan ulasan

Code Review membaca dua file dari repositori Anda untuk memandu apa yang ditandai. Keduanya bersifat aditif di atas pemeriksaan kebenaran default:

- `CLAUDE.md` : instruksi proyek bersama yang digunakan Claude Code untuk semua tugas, bukan hanya ulasan. Gunakan ketika panduan juga berlaku untuk sesi Claude Code interaktif.
- `REVIEW.md` : panduan khusus ulasan, dibaca secara eksklusif selama ulasan kode. Gunakan untuk aturan yang ketat tentang apa yang harus ditandai atau dilewati selama ulasan dan akan mengacaukan `CLAUDE.md` umum Anda.

CLAUDE.md

Code Review membaca file `CLAUDE.md` repositori Anda dan memperlakukan pelanggaran yang baru diperkenalkan sebagai temuan tingkat nit. Ini bekerja secara dua arah: jika PR Anda mengubah kode dengan cara yang membuat pernyataan `CLAUDE.md` ketinggalan zaman, Claude menandai bahwa dokumen perlu diperbarui juga.

Claude membaca file `CLAUDE.md` di setiap tingkat hierarki direktori Anda, sehingga aturan dalam `CLAUDE.md` subdirektori hanya berlaku untuk file di bawah jalur tersebut. Lihat dokumentasi memori untuk lebih lanjut tentang cara kerja `CLAUDE.md` .

Untuk panduan khusus ulasan yang tidak ingin Anda terapkan pada sesi Claude Code umum, gunakan `REVIEW.md` sebagai gantinya.

REVIEW.md

Tambahkan file `REVIEW.md` ke akar repositori Anda untuk aturan khusus ulasan. Gunakan untuk mengkodekan:

- Panduan gaya perusahaan atau tim: “lebih suka pengembalian awal daripada kondisional bersarang”
- Konvensi khusus bahasa atau kerangka kerja yang tidak dicakup oleh linter
- Hal-hal yang Claude harus selalu tandai: “rute API baru harus memiliki tes integrasi”
- Hal-hal yang Claude harus lewati: “jangan berkomentar tentang pemformatan dalam kode yang dihasilkan di bawah `/gen/` ”

Contoh `REVIEW.md` :

```
## Panduan Ulasan Kode

### Selalu periksa
- Titik akhir API baru memiliki tes integrasi yang sesuai
- Migrasi basis data kompatibel ke belakang
- Pesan kesalahan tidak membocorkan detail internal kepada pengguna

### Gaya
- Lebih suka pernyataan `match` daripada pemeriksaan `isinstance` berantai
- Gunakan logging terstruktur, bukan interpolasi f-string dalam panggilan log

### Lewati
- File yang dihasilkan di bawah `src/gen/`
- Perubahan hanya pemformatan dalam file `*.lock`
```

Claude secara otomatis menemukan `REVIEW.md` di akar repositori. Tidak ada konfigurasi yang diperlukan.

Lihat penggunaan

Buka claude.ai/analytics/code-review untuk melihat aktivitas Code Review di seluruh organisasi Anda. Dasbor menampilkan:

Bagian	Apa yang ditampilkan
PRs ditinjau	Hitungan harian permintaan tarik yang ditinjau selama rentang waktu yang dipi-lih
Biaya ming-guan	Pengeluaran mingguan pada Code Review
Umpan balik	Hitungan komentar ulasan yang secara otomatis diselesaikan karena pengembang mengatasi masalah
Rincian re-positori	Hitungan per-repo dari PR yang ditinjau dan komentar yang diselesaikan

Tabel repositori di pengaturan admin juga menampilkan biaya rata-rata per ulasan untuk setiap repo.

Harga

Code Review ditagih berdasarkan penggunaan token. Ulasan rata-rata $15-25, diskalakan dengan ukuran PR, kompleksitas basis kode, dan berapa banyak masalah yang memerlukan verifikasi. Penggunaan Code Review ditagih secara terpisah melalui penggunaan ekstra dan tidak dihitung terhadap penggunaan yang disertakan dalam paket Anda.

Pemicu ulasan yang Anda pilih mempengaruhi total biaya:

- **Setelah pembuatan PR saja**: berjalan sekali per PR
- **Setelah setiap push**: berjalan pada setiap komit, mengalikan biaya dengan jumlah push

Biaya muncul di tagihan Anthropic Anda terlepas dari apakah organisasi Anda menggunakan AWS Bedrock atau Google Vertex AI untuk fitur Claude Code lainnya. Untuk menetapkan batas pengeluaran bulanan untuk Code Review, buka claude.ai/admin-settings/usage dan konfigurasikan batas untuk layanan Claude Code Review.

Pantau pengeluaran melalui bagan biaya mingguan di analitik atau kolom biaya rata-rata per-repo di pengaturan admin.

Sumber daya terkait

Code Review dirancang untuk bekerja bersama dengan sisa Claude Code. Jika Anda ingin menjalankan ulasan secara lokal sebelum membuka PR, memerlukan penyiapan yang di-host sendiri, atau ingin mendalami bagaimana `CLAUDE.md` membentuk perilaku Claude di seluruh alat, halaman-halaman ini adalah perhentian berikutnya yang baik:

- Plugins: telusuri pasar plugin, termasuk plugin `code-review` untuk menjalankan ulasan on-demand secara lokal sebelum push
- GitHub Actions: jalankan Claude dalam alur kerja GitHub Actions Anda sendiri untuk otomasi khusus di luar ulasan kode
- GitLab CI/CD: integrasi Claude yang di-host sendiri untuk pipeline GitLab
- Memory: cara kerja file `CLAUDE.md` di seluruh Claude Code
- Analytics: lacak penggunaan Claude Code di luar ulasan kode

Part 7: CI/CD & Integrations

Claude Code GitHub Actions

> *Pelajari tentang integrasi Claude Code ke dalam alur kerja pengembangan Anda dengan Claude Code GitHub Actions*

Claude Code GitHub Actions membawa otomasi bertenaga AI ke alur kerja GitHub Anda. Dengan penyebutan `@claude` sederhana di PR atau issue apa pun, Claude dapat menganalisis kode Anda, membuat pull request, mengimplementasikan fitur, dan memperbaiki bug - semuanya sambil mengikuti standar proyek Anda. Untuk ulasan otomatis yang diposting di setiap PR tanpa pemicu, lihat GitHub Code Review.

Note:

Claude Code GitHub Actions dibangun di atas Claude Agent SDK, yang memungkinkan integrasi programatik Claude Code ke dalam aplikasi Anda. Anda dapat menggunakan SDK untuk membangun alur kerja otomasi kustom di luar GitHub Actions.

Info:

Claude Opus 4.6 sekarang tersedia. Claude Code GitHub Actions default ke Sonnet. Untuk menggunakan Opus 4.6, konfigurasikan parameter model untuk menggunakan `claude-opus-4-6`.

Mengapa menggunakan Claude Code GitHub Actions?

- **Pembuatan PR instan**: Jelaskan apa yang Anda butuhkan, dan Claude membuat PR lengkap dengan semua perubahan yang diperlukan
- **Implementasi kode otomatis**: Ubah issue menjadi kode yang berfungsi dengan satu perintah
- **Mengikuti standar Anda**: Claude menghormati panduan `CLAUDE.md` Anda dan pola kode yang ada
- **Penyiapan sederhana**: Mulai dalam hitungan menit dengan installer dan kunci API kami
- **Aman secara default**: Kode Anda tetap berada di runner Github

Apa yang dapat dilakukan Claude?

Claude Code menyediakan GitHub Action yang kuat yang mengubah cara Anda bekerja dengan kode:

Claude Code Action

GitHub Action ini memungkinkan Anda menjalankan Claude Code dalam alur kerja GitHub Actions Anda. Anda dapat menggunakannya untuk membangun alur kerja kustom apa pun di atas Claude Code.

Lihat repositori →

Penyiapan

Penyiapan cepat

Cara termudah untuk menyiapkan action ini adalah melalui Claude Code di terminal. Cukup buka claude dan jalankan `/install-github-app`.

Perintah ini akan memandu Anda melalui penyiapan aplikasi GitHub dan rahasia yang diperlukan.

> **Note:**
>
> - Anda harus menjadi admin repositori untuk menginstal aplikasi GitHub dan menambahkan rahasia
> - Aplikasi GitHub akan meminta izin baca & tulis untuk Contents, Issues, dan Pull requests
> - Metode quickstart ini hanya tersedia untuk pengguna Claude API langsung. Jika Anda menggunakan AWS Bedrock atau Google Vertex AI, silakan lihat bagian Using with AWS Bedrock & Google Vertex AI.

Penyiapan manual

Jika perintah `/install-github-app` gagal atau Anda lebih suka penyiapan manual, silakan ikuti instruksi penyiapan manual ini:

1. **Instal aplikasi Claude GitHub** ke repositori Anda: https://github.com/apps/claude

   Aplikasi Claude GitHub memerlukan izin repositori berikut:

   - **Contents**: Baca & tulis (untuk memodifikasi file repositori)
   - **Issues**: Baca & tulis (untuk merespons issue)

    - **Pull requests**: Baca & tulis (untuk membuat PR dan push perubahan)

  Untuk detail lebih lanjut tentang keamanan dan izin, lihat dokumentasi keamanan.

2. **Tambahkan ANTHROPIC_API_KEY** ke rahasia repositori Anda (Pelajari cara menggunakan rahasia di GitHub Actions)
3. **Salin file workflow** dari examples/claude.yml ke dalam direktori `.github/workflows/` repositori Anda

> **Tip:**
>
> Setelah menyelesaikan penyiapan cepat atau manual, uji action dengan menandai `@claude` dalam komentar issue atau PR.

Upgrade dari Beta

> **Warning:**
>
> Claude Code GitHub Actions v1.0 memperkenalkan perubahan breaking yang memerlukan pembaruan file workflow Anda untuk upgrade ke v1.0 dari versi beta.

Jika Anda saat ini menggunakan versi beta Claude Code GitHub Actions, kami merekomendasikan untuk memperbarui workflow Anda agar menggunakan versi GA. Versi baru menyederhanakan konfigurasi sambil menambahkan fitur baru yang kuat seperti deteksi mode otomatis.

Perubahan penting

Semua pengguna beta harus membuat perubahan ini pada file workflow mereka untuk upgrade:

1. **Perbarui versi action**: Ubah `@beta` menjadi `@v1`
2. **Hapus konfigurasi mode**: Hapus `mode: "tag"` atau `mode: "agent"` (sekarang terdeteksi otomatis)
3. **Perbarui input prompt**: Ganti `direct_prompt` dengan `prompt`
4. **Pindahkan opsi CLI**: Konversi `max_turns`, `model`, `custom_instructions`, dll. ke `claude_args`

Breaking Changes Reference

Old Beta Input	New v1.0 Input
`mode`	*(Removed - auto-detected)*
`direct_prompt`	`prompt`
`override_prompt`	`prompt` with GitHub variables
`custom_instructions`	`claude_args: --append-system-prompt`
`max_turns`	`claude_args: --max-turns`
`model`	`claude_args: --model`
`allowed_tools`	`claude_args: --allowedTools`
`disallowed_tools`	`claude_args: --disallowedTools`
`claude_env`	`settings` JSON format

Contoh Sebelum dan Sesudah

Versi beta:

```yaml
- uses: anthropics/claude-code-action@beta
  with:
    mode: "tag"
    direct_prompt: "Review this PR for security issues"
    anthropic_api_key: ${{ secrets.ANTHROPIC_API_KEY }}
    custom_instructions: "Follow our coding standards"
    max_turns: "10"
    model: "claude-sonnet-4-6"
```

Versi GA (v1.0):

```
- uses: anthropics/claude-code-action@v1
  with:
    prompt: "Review this PR for security issues"
    anthropic_api_key: ${{ secrets.ANTHROPIC_API_KEY }}
    claude_args: |
      --append-system-prompt "Follow our coding standards"
      --max-turns 10
      --model claude-sonnet-4-6
```

> **Tip:**
>
> Action sekarang secara otomatis mendeteksi apakah akan dijalankan dalam mode interaktif (merespons penyebutan `@claude`) atau mode otomasi (berjalan segera dengan prompt) berdasarkan konfigurasi Anda.

Contoh kasus penggunaan

Claude Code GitHub Actions dapat membantu Anda dengan berbagai tugas. Direktori examples berisi workflow siap pakai untuk skenario berbeda.

Workflow dasar

```
name: Claude Code
on:
  issue_comment:
    types: [created]
  pull_request_review_comment:
    types: [created]
jobs:
  claude:
    runs-on: ubuntu-latest
    steps:
      - uses: anthropics/claude-code-action@v1
        with:
          anthropic_api_key: ${{ secrets.ANTHROPIC_API_KEY }}
          # Responds to @claude mentions in comments
```

Menggunakan skills

```yaml
name: Code Review
on:
  pull_request:
    types: [opened, synchronize]
jobs:
  review:
    runs-on: ubuntu-latest
    steps:
      - uses: anthropics/claude-code-action@v1
        with:
          anthropic_api_key: ${{ secrets.ANTHROPIC_API_KEY }}
          prompt: "Review this pull request for code quality, correctness, and
security. Analyze the diff, then post your findings as review comments."
          claude_args: "--max-turns 5"
```

Otomasi kustom dengan prompt

```yaml
name: Daily Report
on:
  schedule:
    - cron: "0 9 * * *"
jobs:
  report:
    runs-on: ubuntu-latest
    steps:
      - uses: anthropics/claude-code-action@v1
        with:
          anthropic_api_key: ${{ secrets.ANTHROPIC_API_KEY }}
          prompt: "Generate a summary of yesterday's commits and open issues"
          claude_args: "--model opus"
```

Kasus penggunaan umum

Dalam komentar issue atau PR:

```
@claude implement this feature based on the issue description
@claude how should I implement user authentication for this endpoint?
@claude fix the TypeError in the user dashboard component
```

Claude akan secara otomatis menganalisis konteks dan merespons dengan tepat.

Praktik terbaik

Konfigurasi CLAUDE.md

Buat file `CLAUDE.md` di root repositori Anda untuk mendefinisikan panduan gaya kode, kriteria ulasan, aturan khusus proyek, dan pola yang disukai. File ini memandu pemahaman Claude tentang standar proyek Anda.

Pertimbangan keamanan

> **Warning:**
>
> Jangan pernah commit kunci API langsung ke repositori Anda.

Untuk panduan keamanan komprehensif termasuk izin, autentikasi, dan praktik terbaik, lihat dokumentasi keamanan Claude Code Action.

Selalu gunakan GitHub Secrets untuk kunci API:

- Tambahkan kunci API Anda sebagai rahasia repositori bernama `ANTHROPIC_API_KEY`
- Referensikan dalam workflow: `anthropic_api_key: ${{ secrets.ANTHROPIC_API_KEY }}`
- Batasi izin action hanya untuk apa yang diperlukan
- Tinjau saran Claude sebelum merge

Selalu gunakan GitHub Secrets (misalnya, `${{ secrets.ANTHROPIC_API_KEY }}`) daripada hardcoding kunci API langsung dalam file workflow Anda.

Mengoptimalkan kinerja

Gunakan template issue untuk memberikan konteks, jaga `CLAUDE.md` Anda ringkas dan terfokus, dan konfigurasikan timeout yang sesuai untuk workflow Anda.

Biaya CI

Saat menggunakan Claude Code GitHub Actions, waspadai biaya terkait:

Biaya GitHub Actions:

- Claude Code berjalan di runner yang dihosting GitHub, yang mengonsumsi menit GitHub Actions Anda
- Lihat dokumentasi penagihan GitHub untuk harga terperinci dan batas menit

Biaya API:

- Setiap interaksi Claude mengonsumsi token API berdasarkan panjang prompt dan respons
- Penggunaan token bervariasi menurut kompleksitas tugas dan ukuran codebase
- Lihat halaman harga Claude untuk tarif token saat ini

Tips optimasi biaya:

- Gunakan perintah `@claude` spesifik untuk mengurangi panggilan API yang tidak perlu
- Konfigurasikan `--max-turns` yang sesuai dalam `claude_args` untuk mencegah iterasi berlebihan
- Atur timeout tingkat workflow untuk menghindari pekerjaan yang tidak terkontrol
- Pertimbangkan menggunakan kontrol concurrency GitHub untuk membatasi run paralel

Contoh konfigurasi

Claude Code Action v1 menyederhanakan konfigurasi dengan parameter terpadu:

```yaml
- uses: anthropics/claude-code-action@v1
  with:
    anthropic_api_key: ${{ secrets.ANTHROPIC_API_KEY }}
    prompt: "Your instructions here" # Optional
    claude_args: "--max-turns 5" # Optional CLI arguments
```

Fitur utama:

- **Antarmuka prompt terpadu** - Gunakan `prompt` untuk semua instruksi
- **Skills** - Panggil skills yang terinstal langsung dari prompt
- **Passthrough CLI** - Argumen Claude Code CLI apa pun melalui `claude_args`
- **Pemicu fleksibel** - Bekerja dengan event GitHub apa pun

Kunjungi direktori examples untuk file workflow lengkap.

> **Tip:**
>
> Saat merespons komentar issue atau PR, Claude secara otomatis merespons penyebutan @claude. Untuk event lainnya, gunakan parameter `prompt` untuk memberikan instruksi.

Menggunakan dengan AWS Bedrock & Google Vertex AI

Untuk lingkungan enterprise, Anda dapat menggunakan Claude Code GitHub Actions dengan infrastruktur cloud Anda sendiri. Pendekatan ini memberi Anda kontrol atas residensi data dan penagihan sambil mempertahankan fungsionalitas yang sama.

Prasyarat

Sebelum menyiapkan Claude Code GitHub Actions dengan penyedia cloud, Anda memerlukan:

Untuk Google Cloud Vertex AI:

1. Proyek Google Cloud dengan Vertex AI diaktifkan
2. Workload Identity Federation dikonfigurasi untuk GitHub Actions
3. Akun layanan dengan izin yang diperlukan
4. Aplikasi GitHub (direkomendasikan) atau gunakan GITHUB_TOKEN default

Untuk AWS Bedrock:

1. Akun AWS dengan Amazon Bedrock diaktifkan
2. GitHub OIDC Identity Provider dikonfigurasi di AWS
3. Peran IAM dengan izin Bedrock
4. Aplikasi GitHub (direkomendasikan) atau gunakan GITHUB_TOKEN default

Step 1: Buat aplikasi GitHub kustom (Direkomendasikan untuk Penyedia Pihak Ketiga)

Untuk kontrol dan keamanan terbaik saat menggunakan penyedia pihak ketiga seperti Vertex AI atau Bedrock, kami merekomendasikan membuat aplikasi GitHub Anda sendiri:

1. Buka https://github.com/settings/apps/new
2. Isi informasi dasar:

- **GitHub App name**: Pilih nama unik (misalnya, “YourOrg Claude Assistant”)

- **Homepage URL**: Website organisasi Anda atau URL repositori

1. Konfigurasikan pengaturan aplikasi:

- **Webhooks**: Hapus centang "Active" (tidak diperlukan untuk integrasi ini)

1. Atur izin yang diperlukan:

- **Repository permissions**:
- Contents: Read & Write
- Issues: Read & Write
- Pull requests: Read & Write

1. Klik "Create GitHub App"
2. Setelah pembuatan, klik "Generate a private key" dan simpan file `.pem` yang diunduh
3. Catat App ID Anda dari halaman pengaturan aplikasi
4. Instal aplikasi ke repositori Anda:

- Dari halaman pengaturan aplikasi Anda, klik "Install App" di sidebar kiri
- Pilih akun atau organisasi Anda
- Pilih "Only select repositories" dan pilih repositori spesifik
- Klik "Install"

1. Tambahkan kunci pribadi sebagai rahasia ke repositori Anda:

- Buka Settings → Secrets and variables → Actions repositori Anda
- Buat rahasia baru bernama `APP_PRIVATE_KEY` dengan isi file `.pem`

1. Tambahkan App ID sebagai rahasia:

- Buat rahasia baru bernama `APP_ID` dengan ID aplikasi GitHub Anda

> **Note:**
>
> Aplikasi ini akan digunakan dengan action actions/create-github-app-token untuk menghasilkan token autentikasi dalam workflow Anda.

Alternatif untuk Claude API atau jika Anda tidak ingin menyiapkan aplikasi Github Anda sendiri: Gunakan aplikasi Anthropic resmi:

1. Instal dari: https://github.com/apps/claude
2. Tidak ada konfigurasi tambahan yang diperlukan untuk autentikasi

Step 2: Konfigurasikan autentikasi penyedia cloud

Pilih penyedia cloud Anda dan siapkan autentikasi aman:

Konfigurasikan AWS untuk memungkinkan GitHub Actions mengautentikasi dengan aman tanpa menyimpan kredensial.

> ***Security Note**: Gunakan konfigurasi khusus repositori dan berikan hanya izin minimum yang diperlukan.*

Required Setup:

1. **Enable Amazon Bedrock**:

- Minta akses ke model Claude di Amazon Bedrock
- Untuk model lintas region, minta akses di semua region yang diperlukan

1. **Set up GitHub OIDC Identity Provider**:

- Provider URL: `https://token.actions.githubusercontent.com`
- Audience: `sts.amazonaws.com`

1. **Create IAM Role for GitHub Actions**:

- Trusted entity type: Web identity
- Identity provider: `token.actions.githubusercontent.com`
- Permissions: `AmazonBedrockFullAccess` policy
- Configure trust policy for your specific repository

Required Values:

Setelah penyiapan, Anda akan memerlukan:

- **AWS_ROLE_TO_ASSUME**: ARN dari peran IAM yang Anda buat

> **Tip:**
>
> OIDC lebih aman daripada menggunakan kunci akses AWS statis karena kredensial bersifat sementara dan secara otomatis dirotasi.

Lihat dokumentasi AWS untuk instruksi penyiapan OIDC terperinci.

Konfigurasikan Google Cloud untuk memungkinkan GitHub Actions mengautentikasi dengan aman tanpa menyimpan kredensial.

> ***Security Note**: Gunakan konfigurasi khusus repositori dan berikan hanya izin minimum yang diperlukan.*

Required Setup:

1. **Enable APIs** di proyek Google Cloud Anda:

- IAM Credentials API
- Security Token Service (STS) API
- Vertex AI API

1. **Create Workload Identity Federation resources**:

- Buat Workload Identity Pool
- Tambahkan penyedia OIDC GitHub dengan:
- Issuer: `https://token.actions.githubusercontent.com`
- Attribute mappings untuk repositori dan pemilik
- **Security recommendation**: Gunakan kondisi atribut khusus repositori

1. **Create a Service Account**:

- Berikan hanya peran `Vertex AI User`
- **Security recommendation**: Buat akun layanan khusus per repositori

1. **Configure IAM bindings**:

- Izinkan Workload Identity Pool untuk menyamar sebagai akun layanan
- **Security recommendation**: Gunakan set principal khusus repositori

Required Values:

Setelah penyiapan, Anda akan memerlukan:

- **GCP_WORKLOAD_IDENTITY_PROVIDER**: Nama sumber daya penyedia lengkap
- **GCP_SERVICE_ACCOUNT**: Alamat email akun layanan

> **Tip:**
>
> Workload Identity Federation menghilangkan kebutuhan untuk kunci akun layanan yang dapat diunduh, meningkatkan keamanan.

Untuk instruksi penyiapan terperinci, konsultasikan dokumentasi Google Cloud Workload Identity Federation.

Step 3: Tambahkan Rahasia yang Diperlukan

Tambahkan rahasia berikut ke repositori Anda (Settings → Secrets and variables → Actions):

Untuk Claude API (Langsung):

1. **Untuk Autentikasi API**:

- `ANTHROPIC_API_KEY` : Kunci Claude API Anda dari console.anthropic.com

1. **Untuk Aplikasi GitHub (jika menggunakan aplikasi Anda sendiri)**:

- `APP_ID` : ID Aplikasi GitHub Anda
- `APP_PRIVATE_KEY` : Konten kunci pribadi (.pem)

Untuk Google Cloud Vertex AI

1. **Untuk Autentikasi GCP**:

- `GCP_WORKLOAD_IDENTITY_PROVIDER`
- `GCP_SERVICE_ACCOUNT`

1. **Untuk Aplikasi GitHub (jika menggunakan aplikasi Anda sendiri)**:

- `APP_ID` : ID Aplikasi GitHub Anda
- `APP_PRIVATE_KEY` : Konten kunci pribadi (.pem)

Untuk AWS Bedrock

1. **Untuk Autentikasi AWS**:

- `AWS_ROLE_TO_ASSUME`

1. **Untuk Aplikasi GitHub (jika menggunakan aplikasi Anda sendiri)**:

- `APP_ID` : ID Aplikasi GitHub Anda
- `APP_PRIVATE_KEY` : Konten kunci pribadi (.pem)

Step 4: Buat file workflow

Buat file workflow GitHub Actions yang terintegrasi dengan penyedia cloud Anda. Contoh di bawah menunjukkan konfigurasi lengkap untuk AWS Bedrock dan Google Vertex AI:

Prasyarat:

- Akses AWS Bedrock diaktifkan dengan izin model Claude
- GitHub dikonfigurasi sebagai penyedia identitas OIDC di AWS
- Peran IAM dengan izin Bedrock yang mempercayai GitHub Actions

Rahasia GitHub yang diperlukan:

Secret Name	Description
`AWS_ROLE_TO_ASSUME`	ARN dari peran IAM untuk akses Bedrock
`APP_ID`	ID Aplikasi GitHub Anda (dari pengaturan aplikasi)
`APP_PRIVATE_KEY`	Kunci pribadi yang Anda hasilkan untuk Aplikasi GitHub Anda

```yaml
name: Claude PR Action

permissions:
  contents: write
  pull-requests: write
  issues: write
  id-token: write

on:
  issue_comment:
    types: [created]
  pull_request_review_comment:
    types: [created]
  issues:
    types: [opened, assigned]

jobs:
  claude-pr:
    if: |
      (github.event_name == 'issue_comment' && contains(github.event.comment.body,
'@claude')) ||
      (github.event_name == 'pull_request_review_comment' &&
contains(github.event.comment.body, '@claude')) ||
      (github.event_name == 'issues' && contains(github.event.issue.body,
'@claude'))
    runs-on: ubuntu-latest
    env:
      AWS_REGION: us-west-2
    steps:
      - name: Checkout repository
        uses: actions/checkout@v4

      - name: Generate GitHub App token
        id: app-token
        uses: actions/create-github-app-token@v2
        with:
          app-id: ${{ secrets.APP_ID }}
          private-key: ${{ secrets.APP_PRIVATE_KEY }}
```

```yaml
      - name: Configure AWS Credentials (OIDC)
        uses: aws-actions/configure-aws-credentials@v4
        with:
          role-to-assume: ${{ secrets.AWS_ROLE_TO_ASSUME }}
          aws-region: us-west-2

      - uses: anthropics/claude-code-action@v1
        with:
          github_token: ${{ steps.app-token.outputs.token }}
          use_bedrock: "true"
          claude_args: '--model us.anthropic.claude-sonnet-4-6 --max-turns 10'
```

> **Tip:**
>
> Format ID model untuk Bedrock mencakup awalan region (misalnya, `us.anthropic.claude-sonnet-4-6`).

Prasyarat:

- Vertex AI API diaktifkan di proyek GCP Anda
- Workload Identity Federation dikonfigurasi untuk GitHub
- Akun layanan dengan izin Vertex AI

Rahasia GitHub yang diperlukan:

Secret Name	Description
`GCP_WORKLOAD_IDENTITY_PROVIDER`	Nama sumber daya penyedia identitas workload
`GCP_SERVICE_ACCOUNT`	Email akun layanan dengan akses Vertex AI
`APP_ID`	ID Aplikasi GitHub Anda (dari pengaturan aplikasi)
`APP_PRIVATE_KEY`	Kunci pribadi yang Anda hasilkan untuk Aplikasi GitHub Anda

```yaml
name: Claude PR Action

permissions:
  contents: write
  pull-requests: write
  issues: write
  id-token: write

on:
  issue_comment:
    types: [created]
  pull_request_review_comment:
    types: [created]
  issues:
    types: [opened, assigned]

jobs:
  claude-pr:
    if: |
      (github.event_name == 'issue_comment' && contains(github.event.comment.body,
'@claude')) ||
      (github.event_name == 'pull_request_review_comment' &&
contains(github.event.comment.body, '@claude')) ||
      (github.event_name == 'issues' && contains(github.event.issue.body,
'@claude'))
    runs-on: ubuntu-latest
    steps:
      - name: Checkout repository
        uses: actions/checkout@v4

      - name: Generate GitHub App token
        id: app-token
        uses: actions/create-github-app-token@v2
        with:
          app-id: ${{ secrets.APP_ID }}
          private-key: ${{ secrets.APP_PRIVATE_KEY }}

      - name: Authenticate to Google Cloud
```

```
        id: auth
        uses: google-github-actions/auth@v2
        with:
          workload_identity_provider: $
{{ secrets.GCP_WORKLOAD_IDENTITY_PROVIDER }}
          service_account: ${{ secrets.GCP_SERVICE_ACCOUNT }}

      - uses: anthropics/claude-code-action@v1
        with:
          github_token: ${{ steps.app-token.outputs.token }}
          trigger_phrase: "@claude"
          use_vertex: "true"
          claude_args: '--model claude-sonnet-4@20250514 --max-turns 10'
        env:
          ANTHROPIC_VERTEX_PROJECT_ID: ${{ steps.auth.outputs.project_id }}
          CLOUD_ML_REGION: us-east5
          VERTEX_REGION_CLAUDE_3_7_SONNET: us-east5
```

> **Tip:**
>
> ID proyek secara otomatis diambil dari langkah autentikasi Google Cloud, jadi Anda tidak perlu hardcode.

Troubleshooting

Claude tidak merespons perintah @claude

Verifikasi aplikasi GitHub terinstal dengan benar, periksa bahwa workflow diaktifkan, pastikan kunci API diatur dalam rahasia repositori, dan konfirmkan komentar berisi `@claude` (bukan `/claude`).

CI tidak berjalan pada commit Claude

Pastikan Anda menggunakan aplikasi GitHub atau aplikasi kustom (bukan pengguna Actions), periksa pemicu workflow mencakup event yang diperlukan, dan verifikasi izin aplikasi mencakup pemicu CI.

Kesalahan autentikasi

Konfirmkan kunci API valid dan memiliki izin yang cukup. Untuk Bedrock/Vertex, periksa konfigurasi kredensial dan pastikan rahasia dinamai dengan benar dalam workflow.

Konfigurasi lanjutan

Parameter action

Claude Code Action v1 menggunakan konfigurasi yang disederhanakan:

Parameter	Description	Required
`prompt`	Instruksi untuk Claude (teks biasa atau nama skill)	No*
`claude_args`	Argumen CLI yang diteruskan ke Claude Code	No
`anthropic_api_key`	Kunci Claude API	Yes**
`github_token`	Token GitHub untuk akses API	No
`trigger_phrase`	Frasa pemicu kustom (default: “@claude”)	No
`use_bedrock`	Gunakan AWS Bedrock alih-alih Claude API	No
`use_vertex`	Gunakan Google Vertex AI alih-alih Claude API	No

*Prompt opsional - saat dihilangkan untuk komentar issue/PR, Claude merespons frasa pemicu
**Diperlukan untuk Claude API langsung, bukan untuk Bedrock/Vertex

Teruskan argumen CLI

Parameter `claude_args` menerima argumen Claude Code CLI apa pun:

```
claude_args: "--max-turns 5 --model claude-sonnet-4-6 --mcp-config /path/to/config.json"
```

Argumen umum:

- `--max-turns` : Maksimum conversation turns (default: 10)
- `--model` : Model yang digunakan (misalnya, `claude-sonnet-4-6`)
- `--mcp-config` : Path ke konfigurasi MCP
- `--allowed-tools` : Daftar tools yang diizinkan dipisahkan koma

- `--debug` : Aktifkan output debug

Metode integrasi alternatif

Meskipun perintah `/install-github-app` adalah pendekatan yang direkomendasikan, Anda juga dapat:

- **Custom GitHub App**: Untuk organisasi yang memerlukan nama pengguna bermerek atau alur autentikasi kustom. Buat aplikasi GitHub Anda sendiri dengan izin yang diperlukan (contents, issues, pull requests) dan gunakan action actions/create-github-app-token untuk menghasilkan token dalam workflow Anda.
- **Manual GitHub Actions**: Konfigurasi workflow langsung untuk fleksibilitas maksimal
- **MCP Configuration**: Pemuatan dinamis server Model Context Protocol

Lihat dokumentasi Claude Code Action untuk panduan terperinci tentang autentikasi, keamanan, dan konfigurasi lanjutan.

Menyesuaikan perilaku Claude

Anda dapat mengonfigurasi perilaku Claude dengan dua cara:

1. **CLAUDE.md**: Tentukan standar coding, kriteria ulasan, dan aturan khusus proyek dalam file `CLAUDE.md` di root repositori Anda. Claude akan mengikuti panduan ini saat membuat PR dan merespons permintaan. Lihat dokumentasi Memory kami untuk detail lebih lanjut.
2. **Custom prompts**: Gunakan parameter `prompt` dalam file workflow untuk memberikan instruksi khusus workflow. Ini memungkinkan Anda menyesuaikan perilaku Claude untuk workflow atau tugas berbeda.

Claude akan mengikuti panduan ini saat membuat PR dan merespons permintaan.

Claude Code GitLab CI/CD

> *Pelajari tentang mengintegrasikan Claude Code ke dalam alur kerja pengembangan Anda dengan GitLab CI/CD*

Info:

Claude Code untuk GitLab CI/CD saat ini dalam versi beta. Fitur dan fungsionalitas dapat berkembang saat kami menyempurnakan pengalaman.

Integrasi ini dikelola oleh GitLab. Untuk dukungan, lihat masalah GitLab berikut.

Note:

Integrasi ini dibangun di atas Claude Code CLI dan Agent SDK, memungkinkan penggunaan Claude secara terprogram dalam pekerjaan CI/CD dan alur kerja otomasi khusus Anda.

Mengapa menggunakan Claude Code dengan GitLab?

- **Pembuatan MR instan**: Jelaskan apa yang Anda butuhkan, dan Claude mengusulkan MR lengkap dengan perubahan dan penjelasan
- **Implementasi otomatis**: Ubah masalah menjadi kode yang berfungsi dengan satu perintah atau penyebutan
- **Menyadari proyek**: Claude mengikuti panduan `CLAUDE.md` Anda dan pola kode yang ada
- **Pengaturan sederhana**: Tambahkan satu pekerjaan ke `.gitlab-ci.yml` dan satu variabel CI/CD yang disembunyikan
- **Siap untuk perusahaan**: Pilih Claude API, AWS Bedrock, atau Google Vertex AI untuk memenuhi kebutuhan residensi data dan pengadaan
- **Aman secara default**: Berjalan di runner GitLab Anda dengan perlindungan cabang dan persetujuan Anda

Cara kerjanya

Claude Code menggunakan GitLab CI/CD untuk menjalankan tugas AI dalam pekerjaan terisolasi dan melakukan commit hasil kembali melalui MR:

1. **Orkestrasi berbasis peristiwa**: GitLab mendengarkan pemicu pilihan Anda (misalnya, komentar yang menyebutkan `@claude` dalam masalah, MR, atau utas ulasan). Pekerjaan mengumpulkan konteks dari utas dan repositori, membangun prompt dari input tersebut, dan menjalankan Claude Code.
2. **Abstraksi penyedia**: Gunakan penyedia yang sesuai dengan lingkungan Anda:
   - Claude API (SaaS)
   - AWS Bedrock (akses berbasis IAM, opsi lintas wilayah)
   - Google Vertex AI (asli GCP, Workload Identity Federation)
3. **Eksekusi bersandbox**: Setiap interaksi berjalan dalam kontainer dengan aturan jaringan dan sistem file yang ketat. Claude Code memberlakukan izin berskop ruang kerja untuk membatasi penulisan. Setiap perubahan mengalir melalui MR sehingga pengulas melihat diff dan persetujuan masih berlaku.

Pilih titik akhir regional untuk mengurangi latensi dan memenuhi persyaratan kedaulatan data sambil menggunakan perjanjian cloud yang ada.

Apa yang dapat dilakukan Claude?

Claude Code memungkinkan alur kerja CI/CD yang kuat yang mengubah cara Anda bekerja dengan kode:

- Buat dan perbarui MR dari deskripsi masalah atau komentar
- Analisis regresi kinerja dan usulkan optimisasi
- Implementasikan fitur langsung di cabang, kemudian buka MR
- Perbaiki bug dan regresi yang diidentifikasi oleh tes atau komentar
- Merespons komentar lanjutan untuk mengulangi perubahan yang diminta

Pengaturan

Pengaturan cepat

Cara tercepat untuk memulai adalah menambahkan pekerjaan minimal ke `.gitlab-ci.yml` Anda dan menetapkan kunci API Anda sebagai variabel yang disembunyikan.

1. **Tambahkan variabel CI/CD yang disembunyikan**
   - Buka **Settings → CI/CD → Variables**

- Tambahkan `ANTHROPIC_API_KEY` (disembunyikan, dilindungi sesuai kebutuhan)

2. **Tambahkan pekerjaan Claude ke `.gitlab-ci.yml`**

```
stages:
  - ai

claude:
  stage: ai
  image: node:24-alpine3.21
  # Sesuaikan aturan agar sesuai dengan cara Anda ingin memicu pekerjaan:
  # - menjalankan secara manual
  # - peristiwa permintaan penggabungan
  # - pemicu web/API ketika komentar berisi '@claude'
  rules:
    - if: '$CI_PIPELINE_SOURCE == "web"'
    - if: '$CI_PIPELINE_SOURCE == "merge_request_event"'
  variables:
    GIT_STRATEGY: fetch
  before_script:
    - apk update
    - apk add --no-cache git curl bash
    - curl -fsSL https://claude.ai/install.sh | bash
  script:
    # Opsional: mulai server GitLab MCP jika pengaturan Anda menyediakannya
    - /bin/gitlab-mcp-server || true
    # Gunakan variabel AI_FLOW_* saat memanggil melalui pemicu web/API dengan
muatan konteks
    - echo "$AI_FLOW_INPUT for $AI_FLOW_CONTEXT on $AI_FLOW_EVENT"
    - >
      claude
      -p "${AI_FLOW_INPUT:-'Review this MR and implement the requested changes'}"
      --permission-mode acceptEdits
      --allowedTools "Bash Read Edit Write mcp__gitlab"
      --debug
```

Setelah menambahkan pekerjaan dan variabel `ANTHROPIC_API_KEY` Anda, uji dengan menjalankan pekerjaan secara manual dari **CI/CD → Pipelines**, atau picu dari MR untuk membiarkan Claude mengusulkan pembaruan di cabang dan membuka MR jika diperlukan.

> **Note:**
>
> Untuk menjalankan di AWS Bedrock atau Google Vertex AI alih-alih Claude API, lihat bagian Menggunakan dengan AWS Bedrock & Google Vertex AI di bawah untuk pengaturan autentikasi dan lingkungan.

Pengaturan manual (direkomendasikan untuk produksi)

Jika Anda lebih suka pengaturan yang lebih terkontrol atau memerlukan penyedia perusahaan:

1. **Konfigurasi akses penyedia**:
   - **Claude API**: Buat dan simpan `ANTHROPIC_API_KEY` sebagai variabel CI/CD yang disembunyikan
   - **AWS Bedrock**: **Konfigurasi GitLab → AWS OIDC** dan buat peran IAM untuk Bedrock
   - **Google Vertex AI**: **Konfigurasi Workload Identity Federation untuk GitLab → GCP**
2. **Tambahkan kredensial proyek untuk operasi GitLab API**:
   - Gunakan `CI_JOB_TOKEN` secara default, atau buat Project Access Token dengan cakupan `api`
   - Simpan sebagai `GITLAB_ACCESS_TOKEN` (disembunyikan) jika menggunakan PAT
3. **Tambahkan pekerjaan Claude ke `.gitlab-ci.yml`** (lihat contoh di bawah)
4. **(Opsional) Aktifkan pemicu berbasis penyebutan**:
   - Tambahkan webhook proyek untuk “Comments (notes)” ke pendengar peristiwa Anda (jika Anda menggunakannya)
   - Buat pendengar memanggil API pemicu pipeline dengan variabel seperti `AI_FLOW_INPUT` dan `AI_FLOW_CONTEXT` ketika komentar berisi `@claude`

Contoh kasus penggunaan

Ubah masalah menjadi MR

Dalam komentar masalah:

```
@claude implement this feature based on the issue description
```

Claude menganalisis masalah dan basis kode, menulis perubahan di cabang, dan membuka MR untuk ditinjau.

Dapatkan bantuan implementasi

Dalam diskusi MR:

```
@claude suggest a concrete approach to cache the results of this API call
```

Claude mengusulkan perubahan, menambahkan kode dengan caching yang sesuai, dan memperbarui MR.

Perbaiki bug dengan cepat

Dalam komentar masalah atau MR:

```
@claude fix the TypeError in the user dashboard component
```

Claude menemukan bug, mengimplementasikan perbaikan, dan memperbarui cabang atau membuka MR baru.

Menggunakan dengan AWS Bedrock & Google Vertex AI

Untuk lingkungan perusahaan, Anda dapat menjalankan Claude Code sepenuhnya di infrastruktur cloud Anda dengan pengalaman pengembang yang sama.

AWS Bedrock

Prasyarat

Sebelum menyiapkan Claude Code dengan AWS Bedrock, Anda memerlukan:

1. Akun AWS dengan akses Amazon Bedrock ke model Claude yang diinginkan
2. GitLab dikonfigurasi sebagai penyedia identitas OIDC di AWS IAM
3. Peran IAM dengan izin Bedrock dan kebijakan kepercayaan yang dibatasi pada proyek/ref GitLab Anda
4. Variabel CI/CD GitLab untuk asumsi peran:

- `AWS_ROLE_TO_ASSUME` (ARN peran)

- `AWS_REGION` (wilayah Bedrock)

Instruksi pengaturan

Konfigurasi AWS untuk memungkinkan pekerjaan GitLab CI mengasumsikan peran IAM melalui OIDC (tanpa kunci statis).

Pengaturan yang diperlukan:

1. Aktifkan Amazon Bedrock dan minta akses ke model Claude target Anda
2. Buat penyedia OIDC IAM untuk GitLab jika belum ada
3. Buat peran IAM yang dipercaya oleh penyedia OIDC GitLab, dibatasi pada proyek dan ref yang dilindungi Anda
4. Lampirkan izin hak istimewa minimal untuk API invoke Bedrock

Nilai yang diperlukan untuk disimpan dalam variabel CI/CD:

- `AWS_ROLE_TO_ASSUME`
- `AWS_REGION`

Tambahkan variabel di Settings → CI/CD → Variables:

```
## Untuk AWS Bedrock:
- AWS_ROLE_TO_ASSUME
- AWS_REGION
```

Gunakan contoh pekerjaan AWS Bedrock di atas untuk menukar token pekerjaan GitLab dengan kredensial AWS sementara saat runtime.

Google Vertex AI

Prasyarat

Sebelum menyiapkan Claude Code dengan Google Vertex AI, Anda memerlukan:

1. Proyek Google Cloud dengan:
   - Vertex AI API diaktifkan
   - Workload Identity Federation dikonfigurasi untuk mempercayai GitLab OIDC
1. Akun layanan khusus dengan hanya peran Vertex AI yang diperlukan
2. Variabel CI/CD GitLab untuk WIF:
   - `GCP_WORKLOAD_IDENTITY_PROVIDER` (nama sumber daya lengkap)
   - `GCP_SERVICE_ACCOUNT` (email akun layanan)

Instruksi pengaturan

Konfigurasi Google Cloud untuk memungkinkan pekerjaan GitLab CI menyamar sebagai akun layanan melalui Workload Identity Federation.

Pengaturan yang diperlukan:

1. Aktifkan IAM Credentials API, STS API, dan Vertex AI API
2. Buat Workload Identity Pool dan penyedia untuk GitLab OIDC
3. Buat akun layanan khusus dengan peran Vertex AI
4. Berikan izin principal WIF untuk menyamar sebagai akun layanan

Nilai yang diperlukan untuk disimpan dalam variabel CI/CD:

- `GCP_WORKLOAD_IDENTITY_PROVIDER`
- `GCP_SERVICE_ACCOUNT`

Tambahkan variabel di Settings → CI/CD → Variables:

```
## Untuk Google Vertex AI:
- GCP_WORKLOAD_IDENTITY_PROVIDER
- GCP_SERVICE_ACCOUNT
- CLOUD_ML_REGION (misalnya, us-east5)
```

Gunakan contoh pekerjaan Google Vertex AI di atas untuk autentikasi tanpa menyimpan kunci.

Contoh konfigurasi

Di bawah ini adalah cuplikan siap pakai yang dapat Anda sesuaikan dengan pipeline Anda.

.gitlab-ci.yml dasar (Claude API)

```
stages:
  - ai

claude:
  stage: ai
  image: node:24-alpine3.21
  rules:
    - if: '$CI_PIPELINE_SOURCE == "web"'
    - if: '$CI_PIPELINE_SOURCE == "merge_request_event"'
  variables:
    GIT_STRATEGY: fetch
  before_script:
    - apk update
    - apk add --no-cache git curl bash
    - curl -fsSL https://claude.ai/install.sh | bash
  script:
    - /bin/gitlab-mcp-server || true
    - >
      claude
      -p "${AI_FLOW_INPUT:-'Summarize recent changes and suggest improvements'}"
      --permission-mode acceptEdits
      --allowedTools "Bash Read Edit Write mcp__gitlab"
      --debug
  # Claude Code akan menggunakan ANTHROPIC_API_KEY dari variabel CI/CD
```

Contoh pekerjaan AWS Bedrock (OIDC)

Prasyarat:

- Amazon Bedrock diaktifkan dengan akses ke model Claude pilihan Anda
- GitLab OIDC dikonfigurasi di AWS dengan peran yang mempercayai proyek dan ref GitLab Anda
- Peran IAM dengan izin Bedrock (hak istimewa minimal direkomendasikan)

Variabel CI/CD yang diperlukan:

- `AWS_ROLE_TO_ASSUME` : ARN peran IAM untuk akses Bedrock
- `AWS_REGION` : Wilayah Bedrock (misalnya, `us-west-2`)

```
claude-bedrock:
  stage: ai
  image: node:24-alpine3.21
  rules:
    - if: '$CI_PIPELINE_SOURCE == "web"'
  before_script:
    - apk add --no-cache bash curl jq git python3 py3-pip
    - pip install --no-cache-dir awscli
    - curl -fsSL https://claude.ai/install.sh | bash
    # Tukar token OIDC GitLab dengan kredensial AWS
    - export AWS_WEB_IDENTITY_TOKEN_FILE="${CI_JOB_JWT_FILE:-/tmp/oidc_token}"
    - if [ -n "${CI_JOB_JWT_V2}" ]; then printf "%s" "$CI_JOB_JWT_V2" >
"$AWS_WEB_IDENTITY_TOKEN_FILE"; fi
    - >
      aws sts assume-role-with-web-identity
      --role-arn "$AWS_ROLE_TO_ASSUME"
      --role-session-name "gitlab-claude-$(date +%s)"
      --web-identity-token "file://$AWS_WEB_IDENTITY_TOKEN_FILE"
      --duration-seconds 3600 > /tmp/aws_creds.json
    - export AWS_ACCESS_KEY_ID="$(jq -r .Credentials.AccessKeyId /tmp/
aws_creds.json)"
    - export AWS_SECRET_ACCESS_KEY="$(jq -r .Credentials.SecretAccessKey /tmp/
aws_creds.json)"
    - export AWS_SESSION_TOKEN="$(jq -r .Credentials.SessionToken /tmp/
aws_creds.json)"
  script:
    - /bin/gitlab-mcp-server || true
    - >
      claude
      -p "${AI_FLOW_INPUT:-'Implement the requested changes and open an MR'}"
      --permission-mode acceptEdits
      --allowedTools "Bash Read Edit Write mcp__gitlab"
      --debug
  variables:
    AWS_REGION: "us-west-2"
```

Note:

> ID model untuk Bedrock mencakup awalan khusus wilayah (misalnya, `us.anthropic.claude-sonnet-4-6`). Teruskan model yang diinginkan melalui konfigurasi pekerjaan atau prompt Anda jika alur kerja Anda mendukungnya.

Contoh pekerjaan Google Vertex AI (Workload Identity Federation)

Prasyarat:

- Vertex AI API diaktifkan di proyek GCP Anda
- Workload Identity Federation dikonfigurasi untuk mempercayai GitLab OIDC
- Akun layanan dengan izin Vertex AI

Variabel CI/CD yang diperlukan:

- `GCP_WORKLOAD_IDENTITY_PROVIDER`: Nama sumber daya penyedia lengkap
- `GCP_SERVICE_ACCOUNT`: Email akun layanan
- `CLOUD_ML_REGION`: Wilayah Vertex (misalnya, `us-east5`)

```
claude-vertex:
  stage: ai
  image: gcr.io/google.com/cloudsdktool/google-cloud-cli:slim
  rules:
    - if: '$CI_PIPELINE_SOURCE == "web"'
  before_script:
    - apt-get update && apt-get install -y git && apt-get clean
    - curl -fsSL https://claude.ai/install.sh | bash
    # Autentikasi ke Google Cloud melalui WIF (tanpa kunci yang diunduh)
    - >
      gcloud auth login --cred-file=<(cat <<EOF
      {
        "type": "external_account",
        "audience": "${GCP_WORKLOAD_IDENTITY_PROVIDER}",
        "subject_token_type": "urn:ietf:params:oauth:token-type:jwt",
        "service_account_impersonation_url": "https://
iamcredentials.googleapis.com/v1/projects/-/serviceAccounts/$
{GCP_SERVICE_ACCOUNT}:generateAccessToken",
        "token_url": "https://sts.googleapis.com/v1/token"
      }
      EOF
      )
    - gcloud config set project "$(gcloud projects list --
format='value(projectId)' --filter="name:${CI_PROJECT_NAMESPACE}" | head -n1)" ||
true
  script:
    - /bin/gitlab-mcp-server || true
    - >
      CLOUD_ML_REGION="${CLOUD_ML_REGION:-us-east5}"
      claude
      -p "${AI_FLOW_INPUT:-'Review and update code as requested'}"
      --permission-mode acceptEdits
      --allowedTools "Bash Read Edit Write mcp__gitlab"
      --debug
  variables:
    CLOUD_ML_REGION: "us-east5"
```

> **Note:**
>
> Dengan Workload Identity Federation, Anda tidak perlu menyimpan kunci akun layanan. Gunakan kondisi kepercayaan khusus repositori dan akun layanan dengan hak istimewa minimal.

Praktik terbaik

Konfigurasi CLAUDE.md

Buat file `CLAUDE.md` di akar repositori untuk menentukan standar pengkodean, kriteria ulasan, dan aturan khusus proyek. Claude membaca file ini selama berjalan dan mengikuti konvensi Anda saat mengusulkan perubahan.

Pertimbangan keamanan

Jangan pernah melakukan commit kunci API atau kredensial cloud ke repositori Anda. Selalu gunakan variabel GitLab CI/CD:

- Tambahkan `ANTHROPIC_API_KEY` sebagai variabel yang disembunyikan (dan lindungi jika diperlukan)
- Gunakan OIDC khusus penyedia jika memungkinkan (tanpa kunci jangka panjang)
- Batasi izin pekerjaan dan egress jaringan
- Tinjau MR Claude seperti kontributor lainnya

Mengoptimalkan kinerja

- Jaga `CLAUDE.md` tetap fokus dan ringkas
- Berikan deskripsi masalah/MR yang jelas untuk mengurangi iterasi
- Konfigurasi timeout pekerjaan yang masuk akal untuk menghindari lari liar
- Cache npm dan instalasi paket di runner jika memungkinkan

Biaya CI

Saat menggunakan Claude Code dengan GitLab CI/CD, waspadai biaya terkait:

- **Waktu GitLab Runner**:
  - Claude berjalan di runner GitLab Anda dan mengonsumsi menit komputasi
  - Lihat penagihan runner rencana GitLab Anda untuk detail
- **Biaya API**:
  - Setiap interaksi Claude mengonsumsi token berdasarkan ukuran prompt dan respons

    - Penggunaan token bervariasi menurut kompleksitas tugas dan ukuran basis kode
    - Lihat harga Anthropic untuk detail

- **Tips optimisasi biaya**:
    - Gunakan perintah `@claude` spesifik untuk mengurangi putaran yang tidak perlu
    - Tetapkan nilai `max_turns` dan timeout pekerjaan yang sesuai
    - Batasi keselarasan untuk mengontrol lari paralel

Keamanan dan tata kelola

- Setiap pekerjaan berjalan dalam kontainer terisolasi dengan akses jaringan terbatas
- Perubahan Claude mengalir melalui MR sehingga pengulas melihat setiap diff
- Perlindungan cabang dan aturan persetujuan berlaku untuk kode yang dihasilkan AI
- Claude Code menggunakan izin berskop ruang kerja untuk membatasi penulisan
- Biaya tetap di bawah kontrol Anda karena Anda membawa kredensial penyedia Anda sendiri

Pemecahan masalah

Claude tidak merespons perintah @claude

- Verifikasi pipeline Anda dipicu (secara manual, peristiwa MR, atau melalui pendengar peristiwa catatan/webhook)
- Pastikan variabel CI/CD (`ANTHROPIC_API_KEY` atau pengaturan penyedia cloud) ada dan tidak disembunyikan
- Periksa bahwa komentar berisi `@claude` (bukan `/claude`) dan pemicu penyebutan Anda dikonfigurasi

Pekerjaan tidak dapat menulis komentar atau membuka MR

- Pastikan `CI_JOB_TOKEN` memiliki izin yang cukup untuk proyek, atau gunakan Project Access Token dengan cakupan `api`
- Periksa alat `mcp__gitlab` diaktifkan dalam `--allowedTools`
- Konfirmasi pekerjaan berjalan dalam konteks MR atau memiliki konteks yang cukup melalui variabel `AI_FLOW_*`

Kesalahan autentikasi

- **Untuk Claude API**: Konfirmasi `ANTHROPIC_API_KEY` valid dan tidak kedaluwarsa

- **Untuk Bedrock/Vertex**: Verifikasi konfigurasi OIDC/WIF, impersonasi peran, dan nama rahasia; konfirmasi ketersediaan wilayah dan model

Konfigurasi lanjutan

Parameter dan variabel umum

Claude Code mendukung input yang umum digunakan ini:

- `prompt` / `prompt_file`: Berikan instruksi inline (`-p`) atau melalui file
- `max_turns`: Batasi jumlah iterasi bolak-balik
- `timeout_minutes`: Batasi waktu eksekusi total
- `ANTHROPIC_API_KEY`: Diperlukan untuk Claude API (tidak digunakan untuk Bedrock/Vertex)
- Lingkungan khusus penyedia: `AWS_REGION`, variabel proyek/wilayah untuk Vertex

> **Note:**
>
> Bendera dan parameter yang tepat dapat bervariasi menurut versi `@anthropic-ai/claude-code`. Jalankan `claude --help` dalam pekerjaan Anda untuk melihat opsi yang didukung.

Menyesuaikan perilaku Claude

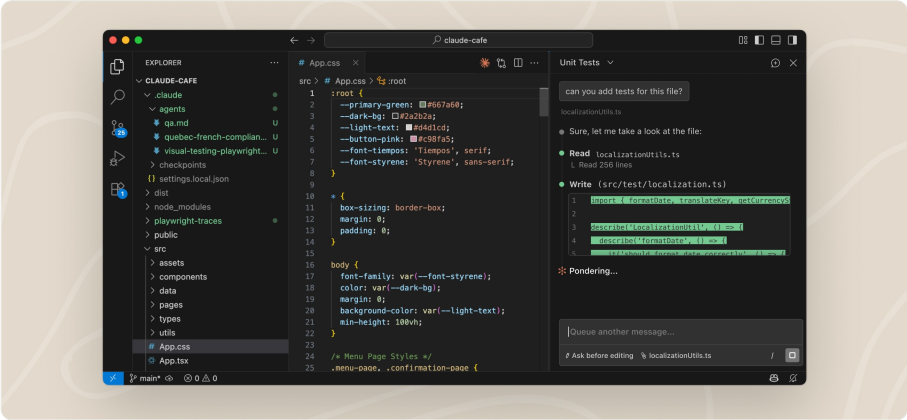
Anda dapat memandu Claude dengan dua cara utama:

1. **CLAUDE.md**: Tentukan standar pengkodean, persyaratan keamanan, dan konvensi proyek. Claude membaca ini selama berjalan dan mengikuti aturan Anda.
2. **Prompt khusus**: Teruskan instruksi khusus tugas melalui `prompt` / `prompt_file` dalam pekerjaan. Gunakan prompt berbeda untuk pekerjaan berbeda (misalnya, ulasan, implementasi, refaktor).

Part 8: IDE & Platform Integration

Gunakan Claude Code di VS Code

> *Instal dan konfigurasi ekstensi Claude Code untuk VS Code. Dapatkan bantuan pengkodean AI dengan diff inline, @-mentions, review rencana, dan pintasan keyboard.*



Editor VS Code dengan panel ekstensi Claude Code terbuka di sisi kanan, menampilkan percakapan dengan Claude

Ekstensi VS Code menyediakan antarmuka grafis asli untuk Claude Code, terintegrasi langsung ke dalam IDE Anda. Ini adalah cara yang direkomendasikan untuk menggunakan Claude Code di VS Code.

Dengan ekstensi, Anda dapat meninjau dan mengedit rencana Claude sebelum menerimanya, auto-accept edits saat dibuat, @-mention file dengan rentang baris tertentu dari pilihan Anda, akses riwayat percakapan, dan buka beberapa percakapan di tab atau jendela terpisah.

Prasyarat

Sebelum menginstal, pastikan Anda memiliki:

- VS Code 1.98.0 atau lebih tinggi

- Akun Anthropic (Anda akan masuk saat pertama kali membuka ekstensi). Jika Anda menggunakan penyedia pihak ketiga seperti Amazon Bedrock atau Google Vertex AI, lihat Gunakan penyedia pihak ketiga sebagai gantinya.

> **Tip:**
>
> Ekstensi mencakup CLI (command-line interface), yang dapat Anda akses dari terminal terintegrasi VS Code untuk fitur lanjutan. Lihat Ekstensi VS Code vs. Claude Code CLI untuk detail.

Instal ekstensi

Klik tautan untuk IDE Anda untuk menginstal secara langsung:

- Instal untuk VS Code
- Instal untuk Cursor

Atau di VS Code, tekan `Cmd+Shift+X` (Mac) atau `Ctrl+Shift+X` (Windows/Linux) untuk membuka tampilan Ekstensi, cari "Claude Code", dan klik **Instal**.

> **Note:**
>
> Jika ekstensi tidak muncul setelah instalasi, restart VS Code atau jalankan "Developer: Reload Window" dari Command Palette.

Memulai

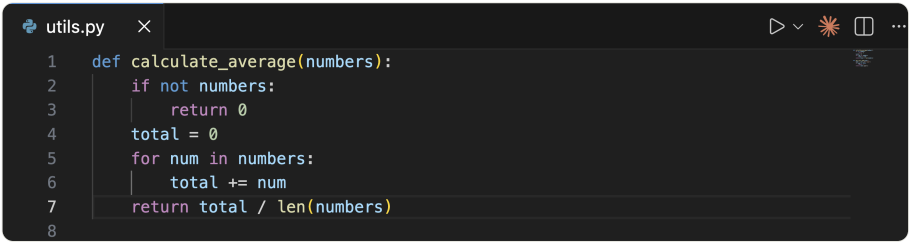
Setelah diinstal, Anda dapat mulai menggunakan Claude Code melalui antarmuka VS Code:

Step 1: Buka panel Claude Code

Di seluruh VS Code, ikon Spark menunjukkan Claude Code:

Spark icon

Cara tercepat untuk membuka Claude adalah dengan mengklik ikon Spark di **Editor Toolbar** (sudut kanan atas editor). Ikon hanya muncul saat Anda memiliki file terbuka.



Editor VS Code menampilkan ikon Spark di Editor Toolbar

Cara lain untuk membuka Claude Code:

- **Activity Bar**: klik ikon Spark di sidebar kiri untuk membuka daftar sesi. Klik sesi apa pun untuk membukanya sebagai tab editor penuh, atau mulai yang baru. Ikon ini selalu terlihat di Activity Bar.
- **Command Palette**: `Cmd+Shift+P` (Mac) atau `Ctrl+Shift+P` (Windows/Linux), ketik "Claude Code", dan pilih opsi seperti "Open in New Tab"
- **Status Bar**: klik **Claude Code** di sudut kanan bawah jendela. Ini berfungsi bahkan saat tidak ada file yang terbuka.

Saat Anda pertama kali membuka panel, daftar periksa **Learn Claude Code** muncul. Kerjakan setiap item dengan mengklik **Show me**, atau tutup dengan X. Untuk membukanya kembali nanti, hapus centang **Hide Onboarding** di pengaturan VS Code di bawah Extensions → Claude Code.

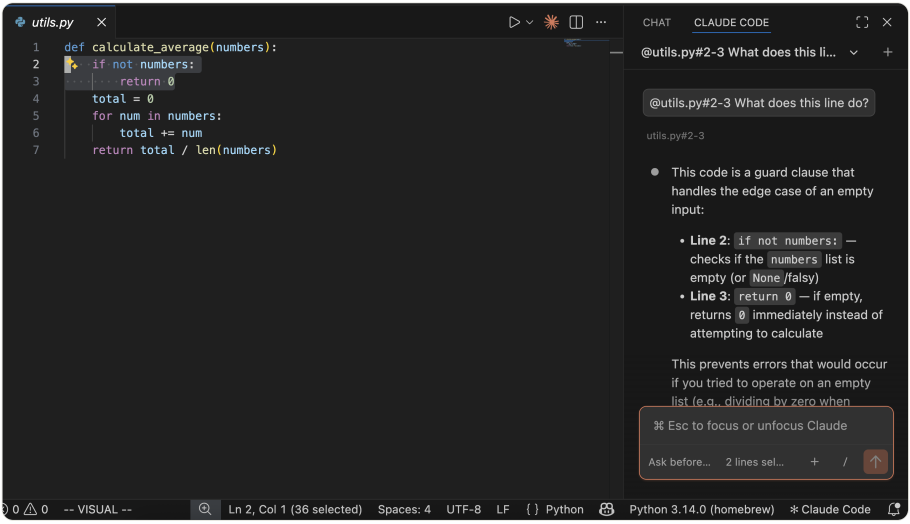
Anda dapat menyeret panel Claude untuk memposisikan ulang di mana saja di VS Code. Lihat Sesuaikan alur kerja Anda untuk detail.

Step 2: Kirim prompt

Minta Claude untuk membantu dengan kode atau file Anda, baik itu menjelaskan cara kerja sesuatu, men-debug masalah, atau membuat perubahan.

> **Tip:**
>
> Claude secara otomatis melihat teks pilihan Anda. Tekan `Option+K` (Mac) / `Alt+K` (Windows/Linux) untuk juga menyisipkan referensi @-mention (seperti `@file.ts#5-10`) ke dalam prompt Anda.

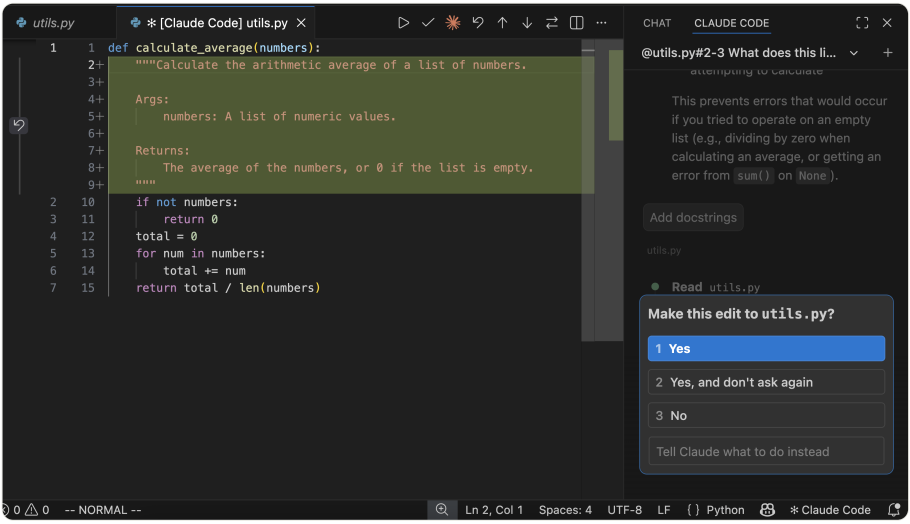
Berikut adalah contoh menanyakan tentang baris tertentu dalam file:



Editor VS Code dengan baris 2-3 dipilih dalam file Python, dan panel Claude Code menampilkan pertanyaan tentang baris tersebut dengan referensi @-mention

Step 3: Tinjau perubahan

Saat Claude ingin mengedit file, ia menampilkan perbandingan berdampingan dari perubahan asli dan yang diusulkan, kemudian meminta izin. Anda dapat menerima, menolak, atau memberi tahu Claude apa yang harus dilakukan sebagai gantinya.



VS Code menampilkan diff dari perubahan yang diusulkan Claude dengan prompt izin menanyakan apakah akan membuat edit

Untuk lebih banyak ide tentang apa yang dapat Anda lakukan dengan Claude Code, lihat Alur kerja umum.

> **Tip:**
>
> Jalankan "Claude Code: Open Walkthrough" dari Command Palette untuk tur terpandu tentang dasar-dasarnya.

Gunakan kotak prompt

Kotak prompt mendukung beberapa fitur:

- **Mode izin**: klik indikator mode di bagian bawah kotak prompt untuk beralih mode. Dalam mode normal, Claude meminta izin sebelum setiap tindakan. Dalam Plan mode, Claude menjelaskan apa yang akan dilakukan dan menunggu persetujuan sebelum membuat perubahan. VS Code secara otomatis membuka rencana sebagai dokumen markdown penuh di mana Anda dapat menambahkan komentar inline untuk memberikan umpan balik sebelum Claude mulai. Dalam mode auto-accept, Claude membuat edit tanpa bertanya. Atur default di pengaturan VS Code di bawah `claudeCode.initialPermissionMode`.
- **Menu perintah**: klik `/` atau ketik `/` untuk membuka menu perintah. Opsi termasuk melampirkan file, beralih model, mengalihkan extended thinking, dan melihat penggunaan rencana (`/usage`). Bagian Customize menyediakan akses ke MCP servers, hooks, memory, permissions, dan plugins. Item dengan ikon terminal terbuka di terminal terintegrasi.
- **Indikator konteks**: kotak prompt menunjukkan berapa banyak context window Claude yang Anda gunakan. Claude secara otomatis melakukan compact saat diperlukan, atau Anda dapat menjalankan `/compact` secara manual.
- **Extended thinking**: memungkinkan Claude menghabiskan lebih banyak waktu untuk bernalar melalui masalah kompleks. Alihkan melalui menu perintah (`/`). Lihat Extended thinking untuk detail.
- **Input multi-baris**: tekan `Shift+Enter` untuk menambahkan baris baru tanpa mengirim. Ini juga berfungsi di input teks bebas "Other" dari dialog pertanyaan.

Referensikan file dan folder

Gunakan @-mentions untuk memberikan Claude konteks tentang file atau folder tertentu. Saat Anda mengetik `@` diikuti dengan nama file atau folder, Claude membaca konten tersebut dan dapat menjawab pertanyaan tentangnya atau membuat perubahan padanya. Claude Code mendukung fuzzy matching, jadi Anda dapat mengetik nama parsial untuk menemukan apa yang Anda butuhkan:

```
> Explain the logic in @auth (fuzzy matches auth.js, AuthService.ts, etc.)
> What's in @src/components/ (include a trailing slash for folders)
```

Untuk PDF besar, Anda dapat meminta Claude membaca halaman tertentu alih-alih seluruh file: satu halaman, rentang seperti halaman 1-10, atau rentang terbuka seperti halaman 3 ke depan.

Saat Anda memilih teks di editor, Claude dapat melihat kode yang disorot secara otomatis. Footer kotak prompt menunjukkan berapa banyak baris yang dipilih. Tekan `Option+K` (Mac) / `Alt+K` (Windows/Linux) untuk menyisipkan @-mention dengan jalur file dan nomor baris (misalnya, `@app.ts#5-10`). Klik indikator pilihan untuk mengalihkan apakah Claude dapat melihat teks yang disorot Anda - ikon eye-slash berarti pilihan tersembunyi dari Claude.

Anda juga dapat menahan `Shift` sambil menyeret file ke kotak prompt untuk menambahkannya sebagai lampiran. Klik X pada lampiran apa pun untuk menghapusnya dari konteks.

Lanjutkan percakapan masa lalu

Klik dropdown di bagian atas panel Claude Code untuk mengakses riwayat percakapan Anda. Anda dapat mencari berdasarkan kata kunci atau menelusuri berdasarkan waktu (Today, Yesterday, Last 7 days, dll.). Klik percakapan apa pun untuk melanjutkannya dengan riwayat pesan lengkap. Arahkan kursor ke sesi untuk mengungkapkan tindakan rename dan remove: rename untuk memberikan judul deskriptif, atau remove untuk menghapusnya dari daftar. Untuk lebih lanjut tentang melanjutkan sesi, lihat Alur kerja umum.

Lanjutkan sesi jarak jauh dari Claude.ai

Jika Anda menggunakan Claude Code di web, Anda dapat melanjutkan sesi jarak jauh tersebut langsung di VS Code. Ini memerlukan masuk dengan **Claude.ai Subscription**, bukan Anthropic Console.

Step 1: Buka Percakapan Masa Lalu

Klik dropdown **Past Conversations** di bagian atas panel Claude Code.

Step 2: Pilih tab Remote

Dialog menampilkan dua tab: Local dan Remote. Klik **Remote** untuk melihat sesi dari claude.ai.

Step 3: Pilih sesi untuk dilanjutkan

Telusuri atau cari sesi jarak jauh Anda. Klik sesi apa pun untuk mengunduhnya dan melanjutkan percakapan secara lokal.

> **Note:**
>
> Hanya sesi web yang dimulai dengan repositori GitHub yang muncul di tab Remote. Melanjutkan memuat riwayat percakapan secara lokal; perubahan tidak disinkronkan kembali ke claude.ai.

Sesuaikan alur kerja Anda

Setelah Anda siap dan berjalan, Anda dapat memposisikan ulang panel Claude, menjalankan beberapa sesi, atau beralih ke mode terminal.

Pilih di mana Claude berada

Anda dapat menyeret panel Claude untuk memposisikan ulang di mana saja di VS Code. Ambil tab atau title bar panel dan seret ke:

- **Secondary sidebar**: sisi kanan jendela. Membuat Claude tetap terlihat saat Anda coding.
- **Primary sidebar**: sidebar kiri dengan ikon untuk Explorer, Search, dll.
- **Editor area**: membuka Claude sebagai tab bersama file Anda. Berguna untuk tugas sampingan.

> **Tip:**
>
> Gunakan sidebar untuk sesi Claude utama Anda dan buka tab tambahan untuk tugas sampingan. Claude mengingat lokasi pilihan Anda. Ikon daftar sesi Activity Bar terpisah dari panel Claude: daftar sesi selalu terlihat di Activity Bar, sementara ikon panel Claude hanya muncul di sana saat panel ditambatkan ke sidebar kiri.

Jalankan beberapa percakapan

Gunakan **Open in New Tab** atau **Open in New Window** dari Command Palette untuk memulai percakapan tambahan. Setiap percakapan mempertahankan riwayat dan konteksnya sendiri, memungkinkan Anda bekerja pada tugas berbeda secara paralel.

Saat menggunakan tab, titik berwarna kecil pada ikon spark menunjukkan status: biru berarti permintaan izin tertunda, oranye berarti Claude selesai saat tab tersembunyi.

Beralih ke mode terminal

Secara default, ekstensi membuka panel chat grafis. Jika Anda lebih suka antarmuka gaya CLI, buka pengaturan Use Terminal dan centang kotak.

Anda juga dapat membuka pengaturan VS Code (`Cmd+,` di Mac atau `Ctrl+,` di Windows/Linux), buka Extensions → Claude Code, dan centang **Use Terminal**.

Kelola plugins

Ekstensi VS Code mencakup antarmuka grafis untuk menginstal dan mengelola plugins. Ketik `/plugins` di kotak prompt untuk membuka antarmuka **Manage plugins**.

Instal plugins

Dialog plugin menampilkan dua tab: **Plugins** dan **Marketplaces**.

Di tab Plugins:

- **Installed plugins** muncul di bagian atas dengan tombol toggle untuk mengaktifkan atau menonaktifkannya
- **Available plugins** dari marketplace yang dikonfigurasi muncul di bawah
- Cari untuk memfilter plugins berdasarkan nama atau deskripsi
- Klik **Install** pada plugin yang tersedia apa pun

Saat Anda menginstal plugin, pilih cakupan instalasi:

- **Install for you**: tersedia di semua proyek Anda (user scope)
- **Install for this project**: dibagikan dengan kolaborator proyek (project scope)
- **Install locally**: hanya untuk Anda, hanya di repositori ini (local scope)

Kelola marketplaces

Beralih ke tab **Marketplaces** untuk menambah atau menghapus sumber plugin:

- Masukkan repo GitHub, URL, atau jalur lokal untuk menambahkan marketplace baru
- Klik ikon refresh untuk memperbarui daftar plugin marketplace

- Klik ikon trash untuk menghapus marketplace

Setelah membuat perubahan, banner meminta Anda untuk restart Claude Code untuk menerapkan pembaruan.

> **Note:**
>
> Manajemen plugin di VS Code menggunakan perintah CLI yang sama di balik layar. Plugins dan marketplaces yang Anda konfigurasi di ekstensi juga tersedia di CLI, dan sebaliknya.

Untuk lebih lanjut tentang sistem plugin, lihat Plugins dan Plugin marketplaces.

Otomatisasi tugas browser dengan Chrome

Hubungkan Claude ke browser Chrome Anda untuk menguji aplikasi web, debug dengan console logs, dan otomatisasi alur kerja browser tanpa meninggalkan VS Code. Ini memerlukan ekstensi Claude in Chrome versi 1.0.36 atau lebih tinggi.

Ketik `@browser` di kotak prompt diikuti dengan apa yang ingin Anda lakukan Claude:

```
@browser go to localhost:3000 and check the console for errors
```

Anda juga dapat membuka menu lampiran untuk memilih alat browser tertentu seperti membuka tab baru atau membaca konten halaman.

Claude membuka tab baru untuk tugas browser dan berbagi status login browser Anda, sehingga dapat mengakses situs apa pun yang sudah Anda masuki.

Untuk instruksi setup, daftar lengkap kemampuan, dan troubleshooting, lihat Gunakan Claude Code dengan Chrome.

Perintah dan pintasan VS Code

Buka Command Palette (`Cmd+Shift+P` di Mac atau `Ctrl+Shift+P` di Windows/Linux) dan ketik “Claude Code” untuk melihat semua perintah VS Code yang tersedia untuk ekstensi Claude Code.

Beberapa pintasan tergantung pada panel mana yang “focused” (menerima input keyboard). Saat kursor Anda berada di file kode, editor difokuskan. Saat kursor Anda berada di kotak prompt Claude, Claude difokuskan. Gunakan `Cmd+Esc` / `Ctrl+Esc` untuk beralih di antara keduanya.

> **Note:**
>
> Ini adalah perintah VS Code untuk mengontrol ekstensi. Tidak semua perintah Claude Code bawaan tersedia di ekstensi. Lihat Ekstensi VS Code vs. Claude Code CLI untuk detail.

Perintah	Pintasan	Deskripsi
Focus Input	`Cmd+Esc` (Mac) / `Ctrl+Esc` (Windows/Linux)	Alihkan fokus antara editor dan Claude
Open in Side Bar	-	Buka Claude di sidebar kiri
Open in Terminal	-	Buka Claude dalam mode terminal
Open in New Tab	`Cmd+Shift+Esc` (Mac) / `Ctrl+Shift+Esc` (Windows/Linux)	Buka percakapan baru sebagai tab editor
Open in New Window	-	Buka percakapan baru di jendela terpisah
New Conversation	`Cmd+N` (Mac) / `Ctrl+N` (Windows/Linux)	Mulai percakapan baru (memerlukan Claude difokuskan)
Insert @-Mention Reference	`Option+K` (Mac) / `Alt+K` (Windows/Linux)	Sisipkan referensi ke file saat ini dan pilihan (memerlukan editor difokuskan)
Show Logs	-	Lihat log debug ekstensi
Logout	-	Keluar dari akun Anthropic Anda

Konfigurasi pengaturan

Ekstensi memiliki dua jenis pengaturan:

- **Pengaturan ekstensi** di VS Code: mengontrol perilaku ekstensi dalam VS Code. Buka dengan `Cmd+,` (Mac) atau `Ctrl+,` (Windows/Linux), kemudian buka Extensions → Claude Code. Anda juga dapat mengetik `/` dan memilih **General Config** untuk membuka pengaturan.

- **Pengaturan Claude Code** di `~/.claude/settings.json` : dibagikan antara ekstensi dan CLI. Gunakan untuk perintah yang diizinkan, variabel lingkungan, hooks, dan MCP servers. Lihat Settings untuk detail.

> **Tip:**
>
> Tambahkan `"$schema": "https://json.schemastore.org/claude-code-settings.json"` ke `settings.json` Anda untuk mendapatkan autocomplete dan validasi inline untuk semua pengaturan yang tersedia langsung di VS Code.

Pengaturan ekstensi

Pengaturan	Default	Deskripsi
`selectedModel`	`default`	Model untuk percakapan baru. Ubah per-sesi dengan `/model`.
`useTerminal`	`false`	Luncurkan Claude dalam mode terminal alih-alih panel grafis
`initialPermissionMode`	`default`	Mengontrol prompt persetujuan: `default` (tanya setiap kali), `plan`, `acceptEdits`, atau `bypassPermissions`
`preferredLocation`	`panel`	Di mana Claude terbuka: `sidebar` (kanan) atau `panel` (tab baru)
`autosave`	`true`	Auto-save file sebelum Claude membaca atau menulisnya
`useCtrlEnterToSend`	`false`	Gunakan Ctrl/Cmd+Enter alih-alih Enter untuk mengirim prompt
`enableNewConversationShortcut`	`true`	Aktifkan Cmd/Ctrl+N untuk memulai percakapan baru

Pengaturan	Default	Deskripsi
`hideOnboarding`	`false`	Sembunyikan daftar periksa onboarding (ikon topi kelulusan)
`respectGitIgnore`	`true`	Kecualikan pola .gitignore dari pencarian file
`environmentVariables`	`[]`	Atur variabel lingkungan untuk proses Claude. Gunakan pengaturan Claude Code sebagai gantinya untuk konfigurasi bersama.
`disableLoginPrompt`	`false`	Lewati prompt autentikasi (untuk setup penyedia pihak ketiga)
`allowDangerouslySkipPermissions`	`false`	Lewati semua prompt izin. **Gunakan dengan sangat hati-hati.**
`claudeProcessWrapper`	-	Jalur executable yang digunakan untuk meluncurkan proses Claude

Ekstensi VS Code vs. Claude Code CLI

Claude Code tersedia sebagai ekstensi VS Code (panel grafis) dan CLI (command-line interface di terminal). Beberapa fitur hanya tersedia di CLI. Jika Anda memerlukan fitur khusus CLI, jalankan `claude` di terminal terintegrasi VS Code.

Fitur	CLI	Ekstensi VS Code
Perintah dan skills	Semua	Subset (ketik `/` untuk melihat yang tersedia)
Konfigurasi MCP server	Ya	Parsial (tambahkan server melalui CLI; kelola server yang ada dengan `/mcp` di panel chat)
Checkpoints	Ya	Ya

Fitur	CLI	Ekstensi VS Code
Pintasan bash `!`	Ya	Tidak
Tab completion	Ya	Tidak

Rewind dengan checkpoints

Ekstensi VS Code mendukung checkpoints, yang melacak edit file Claude dan memungkinkan Anda untuk rewind ke status sebelumnya. Arahkan kursor ke pesan apa pun untuk mengungkapkan tombol rewind, kemudian pilih dari tiga opsi:

- **Fork conversation from here**: mulai cabang percakapan baru dari pesan ini sambil menjaga semua perubahan kode tetap utuh
- **Rewind code to here**: kembalikan perubahan file ke titik ini dalam percakapan sambil menjaga riwayat percakapan lengkap
- **Fork conversation and rewind code**: mulai cabang percakapan baru dan kembalikan perubahan file ke titik ini

Untuk detail lengkap tentang cara kerja checkpoints dan keterbatasannya, lihat Checkpointing.

Jalankan CLI di VS Code

Untuk menggunakan CLI sambil tetap berada di VS Code, buka terminal terintegrasi (`Ctrl+`` di Windows/Linux atau `Cmd+`` di Mac) dan jalankan `claude`. CLI secara otomatis terintegrasi dengan IDE Anda untuk fitur seperti tampilan diff dan berbagi diagnostik.

Jika menggunakan terminal eksternal, jalankan `/ide` di dalam Claude Code untuk menghubungkannya ke VS Code.

Beralih antara ekstensi dan CLI

Ekstensi dan CLI berbagi riwayat percakapan yang sama. Untuk melanjutkan percakapan ekstensi di CLI, jalankan `claude --resume` di terminal. Ini membuka picker interaktif di mana Anda dapat mencari dan memilih percakapan Anda.

Sertakan output terminal dalam prompt

Referensikan output terminal dalam prompt Anda menggunakan `@terminal:name` di mana `name` adalah judul terminal. Ini memungkinkan Claude melihat output perintah, pesan kesalahan, atau log tanpa copy-paste.

Pantau proses latar belakang

Saat Claude menjalankan perintah yang berjalan lama, ekstensi menampilkan kemajuan di status bar. Namun, visibilitas untuk tugas latar belakang terbatas dibandingkan dengan CLI. Untuk visibilitas yang lebih baik, minta Claude menampilkan perintah sehingga Anda dapat menjalankannya di terminal terintegrasi VS Code.

Hubungkan ke alat eksternal dengan MCP

MCP (Model Context Protocol) servers memberikan Claude akses ke alat eksternal, database, dan API.

Untuk menambahkan MCP server, buka terminal terintegrasi (`Ctrl+`` ` atau `Cmd+``) dan jalankan:

```
claude mcp add --transport http github https://api.githubcopilot.com/mcp/
```

Setelah dikonfigurasi, minta Claude untuk menggunakan alat (misalnya, “Review PR #456”).

Untuk mengelola MCP servers tanpa meninggalkan VS Code, ketik `/mcp` di panel chat. Dialog manajemen MCP memungkinkan Anda mengaktifkan atau menonaktifkan server, reconnect ke server, dan mengelola autentikasi OAuth. Lihat dokumentasi MCP untuk server yang tersedia.

Bekerja dengan git

Claude Code terintegrasi dengan git untuk membantu dengan alur kerja kontrol versi langsung di VS Code. Minta Claude untuk commit perubahan, membuat pull request, atau bekerja di seluruh branch.

Buat commits dan pull requests

Claude dapat stage perubahan, menulis pesan commit, dan membuat pull request berdasarkan pekerjaan Anda:

```
> commit my changes with a descriptive message
> create a pr for this feature
> summarize the changes I've made to the auth module
```

Saat membuat pull request, Claude menghasilkan deskripsi berdasarkan perubahan kode aktual dan dapat menambahkan konteks tentang pengujian atau keputusan implementasi.

Gunakan git worktrees untuk tugas paralel

Gunakan flag `--worktree` (`-w`) untuk memulai Claude di worktree terisolasi dengan file dan branch-nya sendiri:

```
claude --worktree feature-auth
```

Setiap worktree mempertahankan status file independen sambil berbagi riwayat git. Ini mencegah instance Claude saling mengganggu saat bekerja pada tugas berbeda. Untuk detail lebih lanjut, lihat Jalankan sesi Claude Code paralel dengan Git worktrees.

Gunakan penyedia pihak ketiga

Secara default, Claude Code terhubung langsung ke API Anthropic. Jika organisasi Anda menggunakan Amazon Bedrock, Google Vertex AI, atau Microsoft Foundry untuk mengakses Claude, konfigurasi ekstensi untuk menggunakan penyedia Anda sebagai gantinya:

Step 1: Nonaktifkan prompt login

Buka pengaturan Disable Login Prompt dan centang kotak.

Anda juga dapat membuka pengaturan VS Code (`Cmd+,` di Mac atau `Ctrl+,` di Windows/Linux), cari “Claude Code login”, dan centang **Disable Login Prompt**.

Step 2: Konfigurasi penyedia Anda

Ikuti panduan setup untuk penyedia Anda:

- Claude Code di Amazon Bedrock
- Claude Code di Google Vertex AI
- Claude Code di Microsoft Foundry

Panduan ini mencakup konfigurasi penyedia Anda di `~/.claude/settings.json` , yang memastikan pengaturan Anda dibagikan antara ekstensi VS Code dan CLI.

Keamanan dan privasi

Kode Anda tetap pribadi. Claude Code memproses kode Anda untuk memberikan bantuan tetapi tidak menggunakannya untuk melatih model. Untuk detail tentang penanganan data dan cara opt out dari logging, lihat Data and privacy.

Dengan izin auto-edit diaktifkan, Claude Code dapat memodifikasi file konfigurasi VS Code (seperti `settings.json` atau `tasks.json`) yang mungkin dijalankan VS Code secara otomatis. Untuk mengurangi risiko saat bekerja dengan kode yang tidak dipercaya:

- Aktifkan VS Code Restricted Mode untuk workspace yang tidak dipercaya
- Gunakan mode persetujuan manual alih-alih auto-accept untuk edit
- Tinjau perubahan dengan hati-hati sebelum menerimanya

Perbaiki masalah umum

Ekstensi tidak akan diinstal

- Pastikan Anda memiliki versi VS Code yang kompatibel (1.98.0 atau lebih baru)
- Periksa bahwa VS Code memiliki izin untuk menginstal ekstensi
- Coba instal langsung dari VS Code Marketplace

Ikon Spark tidak terlihat

Ikon Spark muncul di **Editor Toolbar** (kanan atas editor) saat Anda memiliki file terbuka. Jika Anda tidak melihatnya:

1. **Buka file**: Ikon memerlukan file untuk dibuka. Hanya membuka folder tidak cukup.
2. **Periksa versi VS Code**: Memerlukan 1.98.0 atau lebih tinggi (Help → About)
3. **Restart VS Code**: Jalankan "Developer: Reload Window" dari Command Palette
4. **Nonaktifkan ekstensi yang bertentangan**: Sementara nonaktifkan ekstensi AI lainnya (Cline, Continue, dll.)
5. **Periksa kepercayaan workspace**: Ekstensi tidak berfungsi dalam Restricted Mode

Alternatifnya, klik " Claude Code" di **Status Bar** (sudut kanan bawah). Ini berfungsi bahkan tanpa file terbuka. Anda juga dapat menggunakan **Command Palette** (`Cmd+Shift+P` / `Ctrl+Shift+P`) dan ketik "Claude Code".

Claude Code tidak pernah merespons

Jika Claude Code tidak merespons prompt Anda:

1. **Periksa koneksi internet Anda**: Pastikan Anda memiliki koneksi internet yang stabil
2. **Mulai percakapan baru**: Coba mulai percakapan segar untuk melihat apakah masalah berlanjut
3. **Coba CLI**: Jalankan `claude` dari terminal untuk melihat apakah Anda mendapatkan pesan kesalahan yang lebih detail

Jika masalah berlanjut, file an issue on GitHub dengan detail tentang kesalahan.

Uninstal ekstensi

Untuk menguninstal ekstensi Claude Code:

1. Buka tampilan Extensions (`Cmd+Shift+X` di Mac atau `Ctrl+Shift+X` di Windows/Linux)
2. Cari “Claude Code”
3. Klik **Uninstall**

Untuk juga menghapus data ekstensi dan reset semua pengaturan:

```
rm -rf ~/.vscode/globalStorage/anthropic.claude-code
```

Untuk bantuan tambahan, lihat panduan troubleshooting.

Langkah berikutnya

Sekarang Anda telah menyiapkan Claude Code di VS Code:

- Jelajahi alur kerja umum untuk mendapatkan hasil maksimal dari Claude Code
- Siapkan MCP servers untuk memperluas kemampuan Claude dengan alat eksternal. Tambahkan server menggunakan CLI, kemudian kelola dengan `/mcp` di panel chat.
- Konfigurasi pengaturan Claude Code untuk menyesuaikan perintah yang diizinkan, hooks, dan lainnya. Pengaturan ini dibagikan antara ekstensi dan CLI.

JetBrains IDEs

> *Gunakan Claude Code dengan JetBrains IDEs termasuk IntelliJ, PyCharm, WebStorm, dan lainnya*

Claude Code terintegrasi dengan JetBrains IDEs melalui plugin khusus, menyediakan fitur seperti tampilan diff interaktif, berbagi konteks seleksi, dan lainnya.

IDE yang Didukung

Plugin Claude Code bekerja dengan sebagian besar JetBrains IDEs, termasuk:

- IntelliJ IDEA
- PyCharm
- Android Studio
- WebStorm
- PhpStorm
- GoLand

Fitur

- **Peluncuran cepat**: Gunakan `Cmd+Esc` (Mac) atau `Ctrl+Esc` (Windows/Linux) untuk membuka Claude Code langsung dari editor Anda, atau klik tombol Claude Code di UI
- **Tampilan diff**: Perubahan kode dapat ditampilkan langsung di penampil diff IDE alih-alih terminal
- **Konteks seleksi**: Seleksi/tab saat ini di IDE secara otomatis dibagikan dengan Claude Code
- **Pintasan referensi file**: Gunakan `Cmd+Option+K` (Mac) atau `Alt+Ctrl+K` (Linux/Windows) untuk menyisipkan referensi file (misalnya, @File#L1-99)
- **Berbagi diagnostik**: Kesalahan diagnostik (lint, sintaks, dll.) dari IDE secara otomatis dibagikan dengan Claude saat Anda bekerja

Instalasi

Instalasi Marketplace

Temukan dan instal plugin Claude Code dari marketplace JetBrains dan mulai ulang IDE Anda.

Jika Anda belum menginstal Claude Code, lihat panduan quickstart kami untuk instruksi instalasi.

> **Note:**
>
> Setelah menginstal plugin, Anda mungkin perlu memulai ulang IDE Anda sepenuhnya agar dapat diterapkan.

Penggunaan

Dari IDE Anda

Jalankan `claude` dari terminal terintegrasi IDE Anda, dan semua fitur integrasi akan aktif.

Dari Terminal Eksternal

Gunakan perintah `/ide` di terminal eksternal apa pun untuk menghubungkan Claude Code ke JetBrains IDE Anda dan mengaktifkan semua fitur:

```
claude
```

```
/ide
```

Jika Anda ingin Claude memiliki akses ke file yang sama dengan IDE Anda, mulai Claude Code dari direktori yang sama dengan root proyek IDE Anda.

Konfigurasi

Pengaturan Claude Code

Konfigurasikan integrasi IDE melalui pengaturan Claude Code:

1. Jalankan `claude`
2. Masukkan perintah `/config`

3. Atur alat diff ke `auto` untuk deteksi IDE otomatis

Pengaturan Plugin

Konfigurasikan plugin Claude Code dengan membuka **Settings → Tools → Claude Code [Beta]**:

Pengaturan Umum

- **Perintah Claude**: Tentukan perintah khusus untuk menjalankan Claude (misalnya, `claude`, `/usr/local/bin/claude`, atau `npx @anthropic/claude`)
- **Tekan notifikasi untuk perintah Claude tidak ditemukan**: Lewati notifikasi tentang tidak menemukan perintah Claude
- **Aktifkan penggunaan Option+Enter untuk prompt multi-baris** (hanya macOS): Ketika diaktifkan, Option+Enter menyisipkan baris baru dalam prompt Claude Code. Nonaktifkan jika mengalami masalah dengan tombol Option yang ditangkap secara tidak terduga (memerlukan restart terminal)
- **Aktifkan pembaruan otomatis**: Secara otomatis periksa dan instal pembaruan plugin (diterapkan saat restart)

> **Tip:**
>
> Untuk pengguna WSL: Atur `wsl -d Ubuntu -- bash -lic "claude"` sebagai perintah Claude Anda (ganti `Ubuntu` dengan nama distribusi WSL Anda)

Konfigurasi Tombol ESC

Jika tombol ESC tidak menghentikan operasi Claude Code di terminal JetBrains:

1. Buka **Settings → Tools → Terminal**
2. Salah satu dari:
   - Batalkan centang "Pindahkan fokus ke editor dengan Escape", atau
   - Klik "Konfigurasikan pintasan keyboard terminal" dan hapus pintasan "Alihkan fokus ke Editor"
3. Terapkan perubahan

Ini memungkinkan tombol ESC untuk dengan benar menghentikan operasi Claude Code.

Konfigurasi Khusus

Pengembangan Jarak Jauh

> **Warning:**
>
> Saat menggunakan JetBrains Remote Development, Anda harus menginstal plugin di host jarak jauh melalui **Settings → Plugin (Host)**.

Plugin harus diinstal di host jarak jauh, bukan di mesin klien lokal Anda.

Konfigurasi WSL

> **Warning:**
>
> Pengguna WSL mungkin memerlukan konfigurasi tambahan agar deteksi IDE berfungsi dengan baik. Lihat panduan troubleshooting WSL kami untuk instruksi setup terperinci.

Konfigurasi WSL mungkin memerlukan:

- Konfigurasi terminal yang tepat
- Penyesuaian mode jaringan
- Pembaruan pengaturan firewall

Troubleshooting

Plugin Tidak Berfungsi

- Pastikan Anda menjalankan Claude Code dari direktori root proyek
- Periksa bahwa plugin JetBrains diaktifkan dalam pengaturan IDE
- Mulai ulang IDE sepenuhnya (Anda mungkin perlu melakukan ini beberapa kali)
- Untuk Remote Development, pastikan plugin diinstal di host jarak jauh

IDE Tidak Terdeteksi

- Verifikasi plugin diinstal dan diaktifkan
- Mulai ulang IDE sepenuhnya
- Periksa bahwa Anda menjalankan Claude Code dari terminal terintegrasi
- Untuk pengguna WSL, lihat panduan troubleshooting WSL

Perintah Tidak Ditemukan

Jika mengklik ikon Claude menunjukkan “command not found”:

1. Verifikasi Claude Code diinstal: `npm list -g @anthropic-ai/claude-code`
2. Konfigurasikan jalur perintah Claude dalam pengaturan plugin
3. Untuk pengguna WSL, gunakan format perintah WSL yang disebutkan di bagian konfigurasi

Pertimbangan Keamanan

Ketika Claude Code berjalan di JetBrains IDE dengan izin auto-edit diaktifkan, Claude Code mungkin dapat memodifikasi file konfigurasi IDE yang dapat dijalankan secara otomatis oleh IDE Anda. Ini dapat meningkatkan risiko menjalankan Claude Code dalam mode auto-edit dan memungkinkan melewati prompt izin Claude Code untuk eksekusi bash.

Saat berjalan di JetBrains IDEs, pertimbangkan:

- Menggunakan mode persetujuan manual untuk edit
- Berhati-hati ekstra untuk memastikan Claude hanya digunakan dengan prompt terpercaya
- Menyadari file mana yang Claude Code memiliki akses untuk memodifikasi

Untuk bantuan tambahan, lihat panduan troubleshooting kami.

Gunakan Claude Code Desktop

> *Dapatkan lebih banyak dari Claude Code Desktop: sesi paralel dengan isolasi Git, tinjauan diff visual, pratinjau aplikasi, pemantauan PR, mode izin, konektor, dan konfigurasi enterprise.*

Tab Code dalam aplikasi Claude Desktop memungkinkan Anda menggunakan Claude Code melalui antarmuka grafis alih-alih terminal.

Desktop menambahkan kemampuan ini di atas pengalaman Claude Code standar:

- Tinjauan diff visual dengan komentar inline
- Pratinjau aplikasi langsung dengan server dev
- Pemantauan PR GitHub dengan perbaikan otomatis dan penggabungan otomatis
- Sesi paralel dengan isolasi Git worktree otomatis
- Tugas terjadwal yang menjalankan Claude sesuai jadwal berulang
- Konektor untuk GitHub, Slack, Linear, dan lainnya
- Lingkungan lokal, SSH, dan cloud

> **Tip:**
>
> Baru di Desktop? Mulai dengan Memulai untuk memasang aplikasi dan membuat edit pertama Anda.

Halaman ini mencakup bekerja dengan kode, mengelola sesi, memperluas Claude Code, tugas terjadwal, dan konfigurasi. Halaman ini juga mencakup perbandingan CLI dan pemecahan masalah.

Mulai sesi

Sebelum Anda mengirim pesan pertama, konfigurasikan empat hal di area prompt:

- **Lingkungan**: pilih di mana Claude berjalan. Pilih **Lokal** untuk mesin Anda, **Jarak Jauh** untuk sesi cloud yang dihosting Anthropic, atau **koneksi SSH** untuk mesin jarak jauh yang Anda kelola. Lihat konfigurasi lingkungan.
- **Folder proyek**: pilih folder atau repositori tempat Claude bekerja. Untuk sesi jarak jauh, Anda dapat menambahkan beberapa repositori.

- **Model**: pilih model dari dropdown di sebelah tombol kirim. Model dikunci setelah sesi dimulai.
- **Mode izin**: pilih berapa banyak otonomi yang dimiliki Claude dari pemilih mode. Anda dapat mengubah ini selama sesi.

Ketik tugas Anda dan tekan **Enter** untuk memulai. Setiap sesi melacak konteksnya sendiri dan perubahan secara independen.

Bekerja dengan kode

Berikan Claude konteks yang tepat, kontrol berapa banyak yang dilakukannya sendiri, dan tinjau apa yang diubahnya.

Gunakan kotak prompt

Ketik apa yang ingin Anda lakukan Claude dan tekan **Enter** untuk mengirim. Claude membaca file proyek Anda, membuat perubahan, dan menjalankan perintah berdasarkan mode izin Anda. Anda dapat menghentikan Claude kapan saja: klik tombol berhenti atau ketik koreksi Anda dan tekan **Enter**. Claude berhenti melakukan apa yang sedang dilakukannya dan menyesuaikan berdasarkan input Anda.

Tombol + di sebelah kotak prompt memberi Anda akses ke lampiran file, skills, konektor, dan plugins.

Tambahkan file dan konteks ke prompt

Kotak prompt mendukung dua cara untuk membawa konteks eksternal:

- **File @mention**: ketik `@` diikuti nama file untuk menambahkan file ke konteks percakapan. Claude kemudian dapat membaca dan mereferensikan file tersebut.
- **Lampirkan file**: lampirkan gambar, PDF, dan file lainnya ke prompt Anda menggunakan tombol lampiran, atau seret dan lepas file langsung ke prompt. Ini berguna untuk berbagi tangkapan layar bug, mockup desain, atau dokumen referensi.

Pilih mode izin

Mode izin mengontrol berapa banyak otonomi yang dimiliki Claude selama sesi: apakah itu meminta izin sebelum mengedit file, menjalankan perintah, atau keduanya. Anda dapat beralih mode kapan saja menggunakan pemilih mode di sebelah tombol kirim. Mulai dengan Minta izin untuk melihat dengan tepat apa yang dilakukan Claude, kemudian pindah ke Terima edit otomatis atau Mode Rencana saat Anda merasa nyaman.

Mode	Kunci Pengaturan	Perilaku
Minta izin	`default`	Claude meminta izin sebelum mengedit file atau menjalankan perintah. Anda melihat diff dan dapat menerima atau menolak setiap perubahan. Direkomendasikan untuk pengguna baru.
Terima edit otomatis	`acceptEdits`	Claude secara otomatis menerima edit file tetapi masih meminta izin sebelum menjalankan perintah terminal. Gunakan ini ketika Anda mempercayai perubahan file dan menginginkan iterasi yang lebih cepat.
Mode Rencana	`plan`	Claude menganalisis kode Anda dan membuat rencana tanpa memodifikasi file atau menjalankan perintah. Bagus untuk tugas kompleks di mana Anda ingin meninjau pendekatan terlebih dahulu.
Lewati izin	`bypassPermissions`	Claude berjalan tanpa prompt izin apa pun, setara dengan `--dangerously-skip-permissions` di CLI. Aktifkan di Pengaturan Anda → Claude Code di bawah "Izinkan mode lewati izin". Hanya gunakan ini di kontainer atau VM yang disandbox. Admin enterprise dapat menonaktifkan opsi ini.

Mode izin `dontAsk` hanya tersedia di CLI.

> **Tip:**
>
> Mulai tugas kompleks di Mode Rencana sehingga Claude memetakan pendekatan sebelum membuat perubahan. Setelah Anda menyetujui rencana, beralih ke Terima edit otomatis atau Minta izin untuk menjalankannya. Lihat jelajahi terlebih dahulu, kemudian rencanakan, kemudian kode untuk informasi lebih lanjut tentang alur kerja ini.

Sesi jarak jauh mendukung Terima edit otomatis dan Mode Rencana. Minta izin tidak tersedia karena sesi jarak jauh secara otomatis menerima edit file secara default, dan Lewati izin tidak tersedia karena lingkungan jarak jauh sudah disandbox.

Admin enterprise dapat membatasi mode izin mana yang tersedia. Lihat konfigurasi enterprise untuk detail.

Pratinjau aplikasi Anda

Claude dapat memulai server dev dan membuka browser tertanam untuk memverifikasi perubahannya. Ini berfungsi untuk aplikasi web frontend serta server backend: Claude dapat menguji endpoint API, melihat log server, dan mengulangi masalah yang ditemukannya. Dalam kebanyakan kasus, Claude memulai server secara otomatis setelah mengedit file proyek. Anda juga dapat meminta Claude untuk pratinjau kapan saja. Secara default, Claude memverifikasi otomatis perubahan setelah setiap edit.

Dari panel pratinjau, Anda dapat:

- Berinteraksi dengan aplikasi yang sedang berjalan langsung di browser tertanam
- Tonton Claude memverifikasi perubahannya sendiri secara otomatis: mengambil tangkapan layar, memeriksa DOM, mengklik elemen, mengisi formulir, dan memperbaiki masalah yang ditemukannya
- Mulai atau hentikan server dari dropdown **Pratinjau** di toolbar sesi
- Pertahankan cookie dan penyimpanan lokal di seluruh restart server dengan memilih **Pertahankan sesi** di dropdown, sehingga Anda tidak perlu masuk kembali selama pengembangan
- Edit konfigurasi server atau hentikan semua server sekaligus

Claude membuat konfigurasi server awal berdasarkan proyek Anda. Jika aplikasi Anda menggunakan perintah dev khusus, edit `.claude/launch.json` agar sesuai dengan setup Anda. Lihat Konfigurasi server pratinjau untuk referensi lengkap.

Untuk menghapus data sesi yang disimpan, aktifkan **Pertahankan sesi pratinjau** di Pengaturan → Claude Code. Untuk menonaktifkan pratinjau sepenuhnya, aktifkan **Pratinjau** di Pengaturan → Claude Code.

Tinjau perubahan dengan tampilan diff

Setelah Claude membuat perubahan pada kode Anda, tampilan diff memungkinkan Anda meninjau modifikasi file demi file sebelum membuat pull request.

Ketika Claude mengubah file, indikator statistik diff muncul menunjukkan jumlah baris yang ditambahkan dan dihapus, seperti `+12 -1`. Klik indikator ini untuk membuka penampil diff, yang menampilkan daftar file di sebelah kiri dan perubahan untuk setiap file di sebelah kanan.

Untuk mengomentari baris tertentu, klik baris apa pun di diff untuk membuka kotak komentar. Ketik umpan balik Anda dan tekan **Enter** untuk menambahkan komentar. Setelah menambahkan komentar ke beberapa baris, kirimkan semua komentar sekaligus:

- **macOS**: tekan **Cmd+Enter**
- **Windows**: tekan **Ctrl+Enter**

Claude membaca komentar Anda dan membuat perubahan yang diminta, yang muncul sebagai diff baru yang dapat Anda tinjau.

Tinjau kode Anda

Di tampilan diff, klik **Tinjau kode** di toolbar kanan atas untuk meminta Claude mengevaluasi perubahan sebelum Anda melakukan commit. Claude memeriksa diff saat ini dan meninggalkan komentar langsung di tampilan diff. Anda dapat merespons komentar apa pun atau meminta Claude untuk merevisi.

Tinjauan berfokus pada masalah sinyal tinggi: kesalahan kompilasi, kesalahan logika pasti, kerentanan keamanan, dan bug yang jelas. Ini tidak menandai gaya, pemformatan, masalah yang sudah ada sebelumnya, atau apa pun yang akan ditangkap linter.

Pantau status pull request

Setelah Anda membuka pull request, bilah status CI muncul di sesi. Claude Code menggunakan GitHub CLI untuk menanyakan hasil pemeriksaan dan menampilkan kegagalan.

- **Perbaikan otomatis**: ketika diaktifkan, Claude secara otomatis mencoba memperbaiki pemeriksaan CI yang gagal dengan membaca output kegagalan dan mengulangi.
- **Penggabungan otomatis**: ketika diaktifkan, Claude menggabungkan PR setelah semua pemeriksaan lulus. Metode penggabungan adalah squash. Penggabungan otomatis harus diaktifkan di pengaturan repositori GitHub Anda agar ini berfungsi.

Gunakan toggle **Perbaikan otomatis** dan **Penggabungan otomatis** di bilah status CI untuk mengaktifkan salah satu opsi. Claude Code juga mengirim notifikasi desktop ketika CI selesai.

> **Note:**
>
> Pemantauan PR memerlukan GitHub CLI (`gh`) untuk diinstal dan diautentikasi di mesin Anda. Jika `gh` tidak diinstal, Desktop akan meminta Anda untuk memasangnya saat pertama kali Anda mencoba membuat PR.

Kelola sesi

Setiap sesi adalah percakapan independen dengan konteks dan perubahannya sendiri. Anda dapat menjalankan beberapa sesi secara paralel atau mengirim pekerjaan ke cloud.

Bekerja secara paralel dengan sesi

Klik **+ Sesi Baru** di sidebar untuk bekerja pada beberapa tugas secara paralel. Untuk repositori Git, setiap sesi mendapatkan salinan proyek Anda yang terisolasi menggunakan Git worktrees, sehingga perubahan dalam satu sesi tidak mempengaruhi sesi lain sampai Anda melakukan commit.

Worktrees disimpan di `<project-root>/.claude/worktrees/` secara default. Anda dapat mengubah ini ke direktori khusus di Pengaturan → Claude Code di bawah "Lokasi Worktree". Anda juga dapat mengatur awalan cabang yang ditambahkan ke setiap nama cabang worktree, yang berguna untuk menjaga cabang yang dibuat Claude tetap terorganisir. Untuk menghapus worktree setelah selesai, arahkan ke sesi di sidebar dan klik ikon arsip.

> **Note:**
>
> Isolasi sesi memerlukan Git. Sebagian besar Mac menyertakan Git secara default. Jalankan `git --version` di Terminal untuk memeriksa. Di Windows, Git diperlukan agar tab Code berfungsi: unduh Git untuk Windows, pasang, dan mulai ulang aplikasi. Jika Anda mengalami kesalahan Git, coba sesi Cowork untuk membantu memecahkan masalah setup Anda.

Gunakan ikon filter di bagian atas sidebar untuk memfilter sesi berdasarkan status (Aktif, Diarsipkan) dan lingkungan (Lokal, Cloud). Untuk mengganti nama sesi atau memeriksa penggunaan konteks, klik judul sesi di toolbar di bagian atas sesi aktif. Ketika konteks penuh, Claude secara otomatis merangkum percakapan dan terus bekerja. Anda juga dapat mengetik `/compact` untuk memicu perangkuman lebih awal dan membebaskan ruang konteks. Lihat jendela konteks untuk detail tentang cara pemadatan bekerja.

Jalankan tugas jangka panjang dari jarak jauh

Untuk refaktor besar, suite pengujian, migrasi, atau tugas jangka panjang lainnya, pilih **Jarak Jauh** alih-alih **Lokal** saat memulai sesi. Sesi jarak jauh berjalan di infrastruktur cloud Anthropic dan terus berjalan bahkan jika Anda menutup aplikasi atau mematikan komputer. Periksa kembali kapan saja untuk melihat kemajuan atau mengarahkan Claude ke arah yang berbeda. Anda juga dapat memantau sesi jarak jauh dari claude.ai/code atau aplikasi Claude iOS.

Sesi jarak jauh juga mendukung beberapa repositori. Setelah memilih lingkungan cloud, klik tombol + di sebelah pil repo untuk menambahkan repositori tambahan ke sesi. Setiap repo mendapatkan pemilih cabang sendiri. Ini berguna untuk tugas yang mencakup beberapa basis kode, seperti memperbarui perpustakaan bersama dan konsumennya.

Lihat Claude Code di web untuk informasi lebih lanjut tentang cara kerja sesi jarak jauh.

Lanjutkan di permukaan lain

Menu **Lanjutkan di**, dapat diakses dari ikon VS Code di kanan bawah toolbar sesi, memungkinkan Anda memindahkan sesi ke permukaan lain:

- **Claude Code di Web**: mengirim sesi lokal Anda untuk terus berjalan dari jarak jauh. Desktop mendorong cabang Anda, menghasilkan ringkasan percakapan, dan membuat sesi jarak jauh baru dengan konteks lengkap. Anda kemudian dapat memilih untuk mengarsipkan sesi lokal atau menyimpannya. Ini memerlukan pohon kerja yang bersih, dan tidak tersedia untuk sesi SSH.
- **IDE Anda**: membuka proyek Anda di IDE yang didukung di direktori kerja saat ini.

Perluas Claude Code

Hubungkan layanan eksternal, tambahkan alur kerja yang dapat digunakan kembali, sesuaikan perilaku Claude, dan konfigurasikan server pratinjau.

Hubungkan alat eksternal

Untuk sesi lokal dan SSH, klik tombol + di sebelah kotak prompt dan pilih **Konektor** untuk menambahkan integrasi seperti Google Calendar, Slack, GitHub, Linear, Notion, dan lainnya. Anda dapat menambahkan konektor sebelum atau selama sesi. Konektor tidak tersedia untuk sesi jarak jauh.

Untuk mengelola atau memutuskan konektor, buka Pengaturan → Konektor di aplikasi desktop, atau pilih **Kelola konektor** dari menu Konektor di kotak prompt.

Setelah terhubung, Claude dapat membaca kalender Anda, mengirim pesan, membuat masalah, dan berinteraksi dengan alat Anda secara langsung. Anda dapat meminta Claude konektor apa yang dikonfigurasi di sesi Anda.

Konektor adalah MCP servers dengan alur pengaturan grafis. Gunakan untuk integrasi cepat dengan layanan yang didukung. Untuk integrasi yang tidak tercantum di Konektor, tambahkan MCP servers secara manual melalui file pengaturan. Anda juga dapat membuat konektor khusus.

Gunakan skills

Skills memperluas apa yang dapat dilakukan Claude. Claude memuatnya secara otomatis ketika relevan, atau Anda dapat menginvokan satu secara langsung: ketik `/` di kotak prompt atau klik tombol + dan pilih **Slash commands** untuk melihat apa yang tersedia. Ini mencakup perintah bawaan, skills khusus Anda, skills proyek dari basis kode Anda, dan skills dari plugins yang diinstal apa pun. Pilih satu dan itu muncul disorot di bidang input. Ketik tugas Anda setelahnya dan kirim seperti biasa.

Instal plugins

Plugins adalah paket yang dapat digunakan kembali yang menambahkan skills, agents, hooks, MCP servers, dan konfigurasi LSP ke Claude Code. Anda dapat memasang plugins dari aplikasi desktop tanpa menggunakan terminal.

Untuk sesi lokal dan SSH, klik tombol + di sebelah kotak prompt dan pilih **Plugins** untuk melihat plugins yang diinstal dan perintahnya. Untuk menambahkan plugin, pilih **Tambah plugin** dari submenu untuk membuka browser plugin, yang menampilkan plugins yang tersedia dari marketplaces yang dikonfigurasi termasuk marketplace Anthropic resmi. Pilih **Kelola plugins** untuk mengaktifkan, menonaktifkan, atau mencopot plugins.

Plugins dapat dibatasi pada akun pengguna Anda, proyek tertentu, atau lokal saja. Plugins tidak tersedia untuk sesi jarak jauh. Untuk referensi plugin lengkap termasuk membuat plugins Anda sendiri, lihat plugins.

Konfigurasikan server pratinjau

Claude secara otomatis mendeteksi setup server dev Anda dan menyimpan konfigurasi di `.claude/launch.json` di root folder yang Anda pilih saat memulai sesi. Pratinjau menggunakan folder ini sebagai direktori kerjanya, jadi jika Anda memilih folder induk, subfolder dengan server dev mereka sendiri tidak akan terdeteksi secara otomatis. Untuk bekerja dengan server subfolder, mulai sesi di folder itu secara langsung atau tambahkan konfigurasi secara manual.

Untuk menyesuaikan cara server Anda dimulai, misalnya menggunakan `yarn dev` alih-alih `npm run dev` atau mengubah port, edit file secara manual atau klik **Edit konfigurasi** di dropdown Pratinjau untuk membukanya di editor kode Anda. File mendukung JSON dengan komentar.

```
{
  "version": "0.0.1",
  "configurations": [
    {
      "name": "my-app",
      "runtimeExecutable": "npm",
      "runtimeArgs": ["run", "dev"],
      "port": 3000
    }
  ]
}
```

Anda dapat menentukan beberapa konfigurasi untuk menjalankan server berbeda dari proyek yang sama, seperti frontend dan API. Lihat contoh di bawah.

Verifikasi otomatis perubahan

Ketika `autoVerify` diaktifkan, Claude secara otomatis memverifikasi perubahan kode setelah mengedit file. Mengambil tangkapan layar, memeriksa kesalahan, dan mengkonfirmasi perubahan berfungsi sebelum menyelesaikan responsnya.

Verifikasi otomatis aktif secara default. Nonaktifkan per-proyek dengan menambahkan `"autoVerify": false` ke `.claude/launch.json`, atau aktifkan dari menu dropdown **Pratinjau**.

```
{
  "version": "0.0.1",
  "autoVerify": false,
  "configurations": [...]
}
```

Ketika dinonaktifkan, alat pratinjau masih tersedia dan Anda dapat meminta Claude untuk memverifikasi kapan saja. Verifikasi otomatis membuatnya otomatis setelah setiap edit.

Bidang konfigurasi

Setiap entri dalam array `configurations` menerima bidang berikut:

Bidang	Tipe	Deskripsi
`name`	string	Pengidentifikasi unik untuk server ini
`runtimeExecutable`	string	Perintah untuk dijalankan, seperti `npm`, `yarn`, atau `node`
`runtimeArgs`	string[]	Argumen yang dilewatkan ke `runtimeExecutable`, seperti `["run", "dev"]`
`port`	number	Port yang didengarkan server Anda. Default ke 3000
`cwd`	string	Direktori kerja relatif terhadap root proyek Anda. Default ke root proyek. Gunakan `${workspaceFolder}` untuk mereferensikan root proyek secara eksplisit
`env`	object	Variabel lingkungan tambahan sebagai pasangan kunci-nilai, seperti `{ "NODE_ENV": "development" }`. Jangan letakkan rahasia di sini karena file ini dilakukan commit ke repo Anda. Rahasia yang ditetapkan di profil shell Anda diwariskan secara otomatis.
`autoPort`	boolean	Cara menangani konflik port. Lihat di bawah

Bidang	Tipe	Deskripsi
`program`	string	Skrip untuk dijalankan dengan `node`. Lihat kapan menggunakan `program` vs `runtimeExecutable`
`args`	string[]	Argumen yang dilewatkan ke `program`. Hanya digunakan ketika `program` diatur

Kapan menggunakan `program` vs `runtimeExecutable`

Gunakan `runtimeExecutable` dengan `runtimeArgs` untuk memulai server dev melalui pengelola paket. Misalnya, `"runtimeExecutable": "npm"` dengan `"runtimeArgs": ["run", "dev"]` menjalankan `npm run dev`.

Gunakan `program` ketika Anda memiliki skrip mandiri yang ingin Anda jalankan dengan `node` secara langsung. Misalnya, `"program": "server.js"` menjalankan `node server.js`. Lewatkan flag tambahan dengan `args`.

Konflik port

Bidang `autoPort` mengontrol apa yang terjadi ketika port pilihan Anda sudah digunakan:

- `true`: Claude menemukan dan menggunakan port gratis secara otomatis. Cocok untuk sebagian besar server dev.
- `false`: Claude gagal dengan kesalahan. Gunakan ini ketika server Anda harus menggunakan port tertentu, seperti untuk callback OAuth atau allowlist CORS.
- **Tidak diatur (default)**: Claude menanyakan apakah server memerlukan port itu, kemudian menyimpan jawaban Anda.

Ketika Claude memilih port yang berbeda, itu melewatkan port yang ditugaskan ke server Anda melalui variabel lingkungan `PORT`.

Contoh

Konfigurasi ini menunjukkan setup umum untuk tipe proyek berbeda:

Next.js

Konfigurasi ini menjalankan aplikasi Next.js menggunakan Yarn di port 3000:

```
{
  "version": "0.0.1",
  "configurations": [
    {
      "name": "web",
      "runtimeExecutable": "yarn",
      "runtimeArgs": ["dev"],
      "port": 3000
    }
  ]
}
```

Multiple servers

Untuk monorepo dengan server frontend dan API, tentukan beberapa konfigurasi. Frontend menggunakan `autoPort: true` sehingga memilih port gratis jika 3000 diambil, sementara server API memerlukan port 8080 dengan tepat:

```
{
  "version": "0.0.1",
  "configurations": [
    {
      "name": "frontend",
      "runtimeExecutable": "npm",
      "runtimeArgs": ["run", "dev"],
      "cwd": "apps/web",
      "port": 3000,
      "autoPort": true
    },
    {
      "name": "api",
      "runtimeExecutable": "npm",
      "runtimeArgs": ["run", "start"],
      "cwd": "server",
      "port": 8080,
      "env": { "NODE_ENV": "development" },
      "autoPort": false
    }
  ]
}
```

Node.js script

Untuk menjalankan skrip Node.js secara langsung alih-alih menggunakan perintah pengelola paket, gunakan bidang `program`:

```
{
  "version": "0.0.1",
  "configurations": [
    {
      "name": "server",
      "program": "server.js",
      "args": ["--verbose"],
      "port": 4000
    }
  ]
}
```

Jadwalkan tugas berulang

Tugas terjadwal memulai sesi lokal baru secara otomatis pada waktu dan frekuensi yang Anda pilih. Gunakan untuk pekerjaan berulang seperti tinjauan kode harian, pemeriksaan pembaruan dependensi, atau briefing pagi yang menarik dari kalender dan inbox Anda.

Tugas berjalan di mesin Anda, jadi aplikasi desktop harus terbuka dan komputer Anda terjaga agar mereka dapat dijalankan. Lihat Cara tugas terjadwal berjalan untuk detail tentang run yang terlewat dan perilaku catch-up.

> **Note:**
>
> Secara default, tugas terjadwal berjalan terhadap status apa pun yang ada di direktori kerja Anda, termasuk perubahan yang tidak dilakukan commit. Aktifkan toggle worktree di input prompt untuk memberikan setiap run worktree Git terisolasi sendiri, dengan cara yang sama sesi paralel bekerja.

Untuk membuat tugas terjadwal, klik **Jadwal** di sidebar, kemudian **+ Tugas Baru**. Konfigurasikan bidang ini:

Bidang	Deskripsi
Nama	Pengidentifikasi untuk tugas. Dikonversi ke kebab-case huruf kecil dan digunakan sebagai nama folder di disk. Harus unik di seluruh tugas Anda.
Deskripsi	Ringkasan singkat yang ditampilkan di daftar tugas.

Bi-dang	Deskripsi
Prom-pt	Instruksi yang dikirim ke Claude ketika tugas berjalan. Tulis ini dengan cara yang sama seperti Anda menulis pesan apa pun di kotak prompt. Input prompt juga mencakup kontrol untuk model, mode izin, folder kerja, dan worktree.
Fre-kuen-si	Seberapa sering tugas berjalan. Lihat opsi frekuensi di bawah.

Anda juga dapat membuat tugas dengan mendeskripsikan apa yang Anda inginkan di sesi apa pun. Misalnya, “atur tinjauan kode harian yang berjalan setiap pagi jam 9 pagi.”

Opsi frekuensi

- **Manual**: tidak ada jadwal, hanya berjalan ketika Anda klik **Jalankan sekarang**. Berguna untuk menyimpan prompt yang Anda picu sesuai permintaan
- **Setiap jam**: berjalan setiap jam. Setiap tugas mendapatkan offset tetap hingga 10 menit dari atas jam untuk menjarangkan lalu lintas API
- **Harian**: menampilkan pemilih waktu, default ke 9:00 AM waktu lokal
- **Hari kerja**: sama dengan Harian tetapi melewati Sabtu dan Minggu
- **Mingguan**: menampilkan pemilih waktu dan pemilih hari

Untuk interval yang tidak ditawarkan pemilih (setiap 15 menit, hari pertama setiap bulan, dll.), minta Claude di sesi Desktop apa pun untuk mengatur jadwal. Gunakan bahasa biasa; misalnya, “jadwalkan tugas untuk menjalankan semua tes setiap 6 jam.”

Cara tugas terjadwal berjalan

Tugas terjadwal berjalan secara lokal di mesin Anda. Desktop memeriksa jadwal setiap menit saat aplikasi terbuka dan memulai sesi segar ketika tugas jatuh tempo, independen dari sesi manual apa pun yang Anda buka. Setiap tugas mendapatkan penundaan tetap hingga 10 menit setelah waktu terjadwal untuk menjarangkan lalu lintas API. Penundaan bersifat deterministik: tugas yang sama selalu dimulai pada offset yang sama.

Ketika tugas dijalankan, Anda mendapatkan notifikasi desktop dan sesi baru muncul di bawah bagian **Terjadwal** di sidebar. Buka untuk melihat apa yang dilakukan Claude, tinjau perubahan, atau respons ke prompt izin. Sesi bekerja seperti yang lain: Claude dapat mengedit file, menjalankan perintah, membuat commit, dan membuka pull request.

Tugas hanya berjalan saat aplikasi desktop berjalan dan komputer Anda terjaga. Jika komputer Anda tidur melalui waktu terjadwal, run dilewati. Untuk mencegah idle-sleep, aktifkan **Jaga komputer tetap terjaga** di Pengaturan di bawah **Aplikasi desktop → Umum**. Menutup tutup laptop masih membuatnya tidur.

Run yang terlewat

Ketika aplikasi dimulai atau komputer Anda bangun, Desktop memeriksa apakah setiap tugas melewatkan run apa pun dalam tujuh hari terakhir. Jika ya, Desktop memulai tepat satu run catch-up untuk waktu yang paling baru terlewat dan membuang apa pun yang lebih lama. Tugas harian yang melewatkan enam hari berjalan sekali saat bangun. Desktop menampilkan notifikasi ketika run catch-up dimulai.

Ingat ini saat menulis prompt. Tugas yang dijadwalkan untuk 9 pagi mungkin berjalan pada 11 malam jika komputer Anda tidur sepanjang hari. Jika waktu penting, tambahkan penjaga ke prompt itu sendiri, misalnya: “Hanya tinjau commit hari ini. Jika sudah setelah jam 5 sore, lewati tinjauan dan hanya posting ringkasan apa yang terlewat.”

Izin untuk tugas terjadwal

Setiap tugas memiliki mode izin sendiri, yang Anda atur saat membuat atau mengedit tugas. Aturan izin dari `~/.claude/settings.json` juga berlaku untuk sesi tugas terjadwal. Jika tugas berjalan dalam mode Tanya dan perlu menjalankan alat yang tidak memiliki izin, run macet sampai Anda menyetujuinya. Sesi tetap terbuka di sidebar sehingga Anda dapat menjawab nanti.

Untuk menghindari macet, klik **Jalankan sekarang** setelah membuat tugas, tonton prompt izin, dan pilih “selalu izinkan” untuk setiap satu. Run tugas masa depan secara otomatis menyetujui alat yang sama tanpa meminta. Anda dapat meninjau dan mencabut persetujuan ini dari halaman detail tugas.

Kelola tugas terjadwal

Klik tugas di daftar **Jadwal** untuk membuka halaman detailnya. Dari sini Anda dapat:

- **Jalankan sekarang**: mulai tugas segera tanpa menunggu waktu terjadwal berikutnya
- **Toggle berulang**: jeda atau lanjutkan run terjadwal tanpa menghapus tugas
- **Edit**: ubah prompt, frekuensi, folder, atau pengaturan lainnya
- **Tinjau riwayat**: lihat setiap run masa lalu, termasuk yang dilewati karena komputer Anda tidur
- **Tinjau izin yang diizinkan**: lihat dan cabut persetujuan alat yang disimpan untuk tugas ini dari panel **Selalu diizinkan**

- **Hapus**: hapus tugas dan arsipkan semua sesi yang dibuatnya

Anda juga dapat mengelola tugas dengan meminta Claude di sesi Desktop apa pun. Misalnya, “jeda tugas dependency-audit saya”, “hapus tugas standup-prep”, atau “tunjukkan tugas terjadwal saya.”

Untuk mengedit prompt tugas di disk, buka `~/.claude/scheduled-tasks/<task-name>/SKILL.md` (atau di bawah `CLAUDE_CONFIG_DIR` jika diatur). File menggunakan frontmatter YAML untuk `name` dan `description`, dengan prompt sebagai body. Perubahan berlaku pada run berikutnya. Jadwal, folder, model, dan status yang diaktifkan tidak ada di file ini: ubah melalui formulir Edit atau minta Claude.

Konfigurasi lingkungan

Lingkungan yang Anda pilih saat memulai sesi menentukan di mana Claude mengeksekusi dan cara Anda terhubung:

- **Lokal**: berjalan di mesin Anda dengan akses langsung ke file Anda
- **Jarak Jauh**: berjalan di infrastruktur cloud Anthropic. Sesi terus berlanjut bahkan jika Anda menutup aplikasi.
- **SSH**: berjalan di mesin jarak jauh yang Anda hubungkan melalui SSH, seperti server Anda sendiri, cloud VM, atau dev containers

Sesi lokal

Sesi lokal mewarisi variabel lingkungan dari shell Anda. Jika Anda memerlukan variabel tambahan, atur di profil shell Anda, seperti `~/.zshrc` atau `~/.bashrc`, dan mulai ulang aplikasi desktop. Lihat variabel lingkungan untuk daftar lengkap variabel yang didukung.

Extended thinking diaktifkan secara default, yang meningkatkan kinerja pada tugas penalaran kompleks tetapi menggunakan token tambahan. Untuk menonaktifkan pemikiran sepenuhnya, atur `MAX_THINKING_TOKENS=0` di profil shell Anda. Di Opus, `MAX_THINKING_TOKENS` diabaikan kecuali untuk `0` karena penalaran adaptif mengontrol kedalaman pemikiran sebagai gantinya.

Sesi jarak jauh

Sesi jarak jauh terus berlanjut di latar belakang bahkan jika Anda menutup aplikasi. Penggunaan dihitung terhadap batas rencana langganan Anda tanpa biaya komputasi terpisah.

Anda dapat membuat lingkungan cloud khusus dengan tingkat akses jaringan dan variabel lingkungan yang berbeda. Pilih dropdown lingkungan saat memulai sesi jarak jauh dan pilih **Tambah lingkungan**. Lihat lingkungan cloud untuk detail tentang mengonfigurasi akses jaringan dan variabel lingkungan.

Sesi SSH

Sesi SSH memungkinkan Anda menjalankan Claude Code di mesin jarak jauh sambil menggunakan aplikasi desktop sebagai antarmuka Anda. Ini berguna untuk bekerja dengan basis kode yang tinggal di cloud VM, dev containers, atau server dengan perangkat keras atau dependensi tertentu.

Untuk menambahkan koneksi SSH, klik dropdown lingkungan sebelum memulai sesi dan pilih **+ Tambah koneksi SSH**. Dialog menanyakan:

- **Nama**: label ramah untuk koneksi ini
- **SSH Host**: `user@hostname` atau host yang ditentukan di `~/.ssh/config`
- **SSH Port**: default ke 22 jika dibiarkan kosong, atau menggunakan port dari konfigurasi SSH Anda
- **Identity File**: path ke kunci pribadi Anda, seperti `~/.ssh/id_rsa`. Biarkan kosong untuk menggunakan kunci default atau konfigurasi SSH Anda.

Setelah ditambahkan, koneksi muncul di dropdown lingkungan. Pilih untuk memulai sesi di mesin itu. Claude berjalan di mesin jarak jauh dengan akses ke file dan alatnya.

Claude Code harus diinstal di mesin jarak jauh. Setelah terhubung, sesi SSH mendukung mode izin, konektor, plugins, dan MCP servers.

Konfigurasi enterprise

Organisasi pada rencana Teams atau Enterprise dapat mengelola perilaku aplikasi desktop melalui kontrol konsol admin, file pengaturan yang dikelola, dan kebijakan manajemen perangkat.

Kontrol konsol admin

Pengaturan ini dikonfigurasi melalui konsol pengaturan admin:

- **Aktifkan atau nonaktifkan tab Code**: kontrol apakah pengguna di organisasi Anda dapat mengakses Claude Code di aplikasi desktop
- **Nonaktifkan mode Lewati izin**: cegah pengguna di organisasi Anda dari mengaktifkan mode lewati izin

- **Nonaktifkan Claude Code di web**: aktifkan atau nonaktifkan sesi jarak jauh untuk organisasi Anda

Pengaturan yang dikelola

Pengaturan yang dikelola menimpa pengaturan proyek dan pengguna dan berlaku ketika Desktop menjalankan sesi CLI. Anda dapat mengatur kunci ini di file pengaturan yang dikelola organisasi Anda atau mendorongnya dari jarak jauh melalui konsol admin.

Kunci	Deskripsi
`disableBypassPermissionsMode`	atur ke `"disable"` untuk mencegah pengguna dari mengaktifkan mode lewati izin. Lihat pengaturan yang dikelola.

Untuk daftar lengkap pengaturan khusus yang dikelola termasuk `allowManagedPermissionRulesOnly` dan `allowManagedHooksOnly`, lihat pengaturan khusus yang dikelola.

Pengaturan yang dikelola jarak jauh yang diunggah melalui konsol admin saat ini berlaku untuk sesi CLI dan IDE saja. Untuk pembatasan khusus Desktop, gunakan kontrol konsol admin di atas.

Kebijakan manajemen perangkat

Tim IT dapat mengelola aplikasi desktop melalui MDM di macOS atau kebijakan grup di Windows. Kebijakan yang tersedia termasuk mengaktifkan atau menonaktifkan fitur Claude Code, mengontrol pembaruan otomatis, dan menetapkan URL penyebaran khusus.

- **macOS**: konfigurasikan melalui domain preferensi `com.anthropic.Claude` menggunakan alat seperti Jamf atau Kandji
- **Windows**: konfigurasikan melalui registri di `SOFTWARE\Policies\Claude`

Autentikasi dan SSO

Organisasi enterprise dapat memerlukan SSO untuk semua pengguna. Lihat autentikasi untuk detail tingkat rencana dan Menyiapkan SSO untuk konfigurasi SAML dan OIDC.

Penanganan data

Claude Code memproses kode Anda secara lokal di sesi lokal atau di infrastruktur cloud Anthropic di sesi jarak jauh. Percakapan dan konteks kode dikirim ke API Anthropic untuk diproses. Lihat penanganan data untuk detail tentang retensi data, privasi, dan kepatuhan.

Penyebaran

Desktop dapat didistribusikan melalui alat penyebaran enterprise:

- **macOS**: distribusikan melalui MDM seperti Jamf atau Kandji menggunakan installer `.dmg`
- **Windows**: sebarkan melalui paket MSIX atau installer `.exe`. Lihat Sebarkan Claude Desktop untuk Windows untuk opsi penyebaran enterprise termasuk instalasi senyap

Untuk konfigurasi jaringan seperti pengaturan proxy, allowlisting firewall, dan gateway LLM, lihat konfigurasi jaringan.

Untuk referensi konfigurasi enterprise lengkap, lihat panduan konfigurasi enterprise.

Datang dari CLI?

Jika Anda sudah menggunakan CLI Claude Code, Desktop menjalankan mesin yang sama dengan antarmuka grafis. Anda dapat menjalankan keduanya secara bersamaan di mesin yang sama, bahkan di proyek yang sama. Masing-masing mempertahankan riwayat sesi terpisah, tetapi mereka berbagi konfigurasi dan memori proyek melalui file CLAUDE.md.

Untuk memindahkan sesi CLI ke Desktop, jalankan `/desktop` di terminal. Claude menyimpan sesi Anda dan membukanya di aplikasi desktop, kemudian keluar dari CLI. Perintah ini hanya tersedia di macOS dan Windows.

> **Tip:**
>
> Kapan menggunakan Desktop vs CLI: gunakan Desktop ketika Anda menginginkan tinjauan diff visual, lampiran file, atau manajemen sesi di sidebar. Gunakan CLI ketika Anda memerlukan scripting, otomasi, penyedia pihak ketiga, atau lebih suka alur kerja terminal.

Setara flag CLI

Tabel ini menunjukkan setara aplikasi desktop untuk flag CLI umum. Flag yang tidak tercantum tidak memiliki setara desktop karena dirancang untuk scripting atau otomasi.

CLI	Setara desktop
`--model sonnet`	dropdown model di sebelah tombol kirim, sebelum memulai sesi
`--resume`, `--continue`	klik sesi di sidebar
`--permission-mode`	pemilih mode di sebelah tombol kirim

CLI	Setara desktop
`--dangerously-skip-permissions`	Mode Lewati izin. Aktifkan di Pengaturan → Claude Code → "Izinkan mode lewati izin". Admin enterprise dapat menonaktifkan pengaturan ini.
`--add-dir`	tambahkan beberapa repo dengan tombol + di sesi jarak jauh
`--allowedTools`, `--disallowedTools`	tidak tersedia di Desktop
`--verbose`	tidak tersedia. Periksa log sistem: Console.app di macOS, Event Viewer → Windows Logs → Application di Windows
`--print`, `--output-format`	tidak tersedia. Desktop hanya interaktif.
`ANTHROPIC_MODEL` env var	dropdown model di sebelah tombol kirim
`MAX_THINKING_TOKENS` env var	atur di profil shell; berlaku untuk sesi lokal. Lihat konfigurasi lingkungan.

Konfigurasi bersama

Desktop dan CLI membaca file konfigurasi yang sama, jadi setup Anda terbawa:

- File **CLAUDE.md** di proyek Anda digunakan oleh keduanya
- **MCP servers** yang dikonfigurasi di `~/.claude.json` atau `.mcp.json` bekerja di keduanya
- **Hooks** dan **skills** yang ditentukan dalam pengaturan berlaku untuk keduanya
- **Pengaturan** di `~/.claude.json` dan `~/.claude/settings.json` dibagikan. Aturan izin, alat yang diizinkan, dan pengaturan lainnya di `settings.json` berlaku untuk sesi Desktop.
- **Model**: Sonnet, Opus, dan Haiku tersedia di keduanya. Di Desktop, pilih model dari dropdown di sebelah tombol kirim sebelum memulai sesi. Anda tidak dapat mengubah model selama sesi aktif.

> **Note:**

> **MCP servers: aplikasi chat desktop vs Claude Code**: MCP servers yang dikonfigurasi untuk aplikasi chat Claude Desktop di `claude_desktop_config.json` terpisah dari Claude Code dan tidak akan muncul di tab Code. Untuk menggunakan MCP servers di Claude Code, konfigurasikan di `~/.claude.json` atau file `.mcp.json` proyek Anda. Lihat konfigurasi MCP untuk detail.

Perbandingan fitur

Tabel ini membandingkan kemampuan inti antara CLI dan Desktop. Untuk daftar lengkap flag CLI, lihat referensi CLI.

Fitur	CLI	Desktop
Mode izin	semua mode termasuk `dontAsk`	Minta izin, Terima edit otomatis, Mode Rencana, dan Lewati izin melalui Pengaturan
`--dangerously-skip-permissions`	Flag CLI	Mode Lewati izin. Aktifkan di Pengaturan → Claude Code → "Izinkan mode lewati izin"
Penyedia pihak ketiga	Bedrock, Vertex, Foundry	tidak tersedia. Desktop terhubung langsung ke API Anthropic.
MCP servers	konfigurasikan di file pengaturan	UI Konektor untuk sesi lokal dan SSH, atau file pengaturan
Plugins	perintah `/plugin`	UI pengelola plugin
File @mention	berbasis teks	dengan autocomplete
Lampiran file	tidak tersedia	gambar, PDF
Isolasi sesi	flag `--worktree`	worktrees otomatis
Beberapa sesi	terminal terpisah	tab sidebar
Tugas berulang	cron jobs, pipeline CI	tugas terjadwal
Scripting dan otomasi	`--print`, Agent SDK	tidak tersedia

Apa yang tidak tersedia di Desktop

Fitur berikut hanya tersedia di CLI atau ekstensi VS Code:

- **Penyedia pihak ketiga**: Desktop terhubung langsung ke API Anthropic. Gunakan CLI dengan Bedrock, Vertex, atau Foundry sebagai gantinya.
- **Linux**: aplikasi desktop hanya tersedia di macOS dan Windows.
- **Saran kode inline**: Desktop tidak menyediakan saran gaya autocomplete. Ini bekerja melalui prompt percakapan dan perubahan kode eksplisit.
- **Tim agent**: orkestrasi multi-agent tersedia melalui CLI dan Agent SDK, bukan di Desktop.

Pemecahan masalah

Periksa versi Anda

Untuk melihat versi aplikasi desktop yang Anda jalankan:

- **macOS**: klik **Claude** di menu bar, kemudian **Tentang Claude**
- **Windows**: klik **Bantuan**, kemudian **Tentang**

Klik nomor versi untuk menyalinnya ke clipboard Anda.

Kesalahan 403 atau autentikasi di tab Code

Jika Anda melihat `Error 403: Forbidden` atau kegagalan autentikasi lainnya saat menggunakan tab Code:

1. Keluar dan masuk kembali dari menu aplikasi. Ini adalah perbaikan paling umum.
2. Verifikasi Anda memiliki langganan berbayar aktif: Pro, Max, Teams, atau Enterprise.
3. Jika CLI berfungsi tetapi Desktop tidak, keluar dari aplikasi desktop sepenuhnya, bukan hanya tutup jendela, kemudian buka kembali dan masuk.
4. Periksa koneksi internet dan pengaturan proxy Anda.

Layar kosong atau macet saat peluncuran

Jika aplikasi terbuka tetapi menampilkan layar kosong atau tidak responsif:

1. Mulai ulang aplikasi.
2. Periksa pembaruan yang tertunda. Aplikasi secara otomatis memperbarui saat peluncuran.
3. Di Windows, periksa Event Viewer untuk log crash di bawah **Windows Logs → Application**.

"Gagal memuat sesi"

Jika Anda melihat `Failed to load session`, folder yang dipilih mungkin tidak lagi ada, repositori Git mungkin memerlukan Git LFS yang tidak diinstal, atau izin file mungkin mencegah akses. Coba pilih folder berbeda atau mulai ulang aplikasi.

Sesi tidak menemukan alat yang diinstal

Jika Claude tidak dapat menemukan alat seperti `npm`, `node`, atau perintah CLI lainnya, verifikasi alat bekerja di terminal biasa Anda, periksa bahwa profil shell Anda dengan benar menyiapkan PATH, dan mulai ulang aplikasi desktop untuk memuat ulang variabel lingkungan.

Kesalahan Git dan Git LFS

Di Windows, Git diperlukan untuk tab Code memulai sesi lokal. Jika Anda melihat "Git diperlukan," instal Git untuk Windows dan mulai ulang aplikasi.

Jika Anda melihat "Git LFS diperlukan oleh repositori ini tetapi tidak diinstal," instal Git LFS dari git-lfs.com, jalankan `git lfs install`, dan mulai ulang aplikasi.

MCP servers tidak bekerja di Windows

Jika toggle MCP server tidak merespons atau server gagal terhubung di Windows, periksa bahwa server dikonfigurasi dengan benar di pengaturan Anda, mulai ulang aplikasi, verifikasi proses server berjalan di Task Manager, dan tinjau log server untuk kesalahan koneksi.

Aplikasi tidak akan keluar

- **macOS**: tekan Cmd+Q. Jika aplikasi tidak merespons, gunakan Force Quit dengan Cmd+Option+Esc, pilih Claude, dan klik Force Quit.
- **Windows**: gunakan Task Manager dengan Ctrl+Shift+Esc untuk mengakhiri proses Claude.

Masalah khusus Windows

- **PATH tidak diperbarui setelah instalasi**: buka jendela terminal baru. Pembaruan PATH hanya berlaku untuk sesi terminal baru.
- **Kesalahan instalasi bersamaan**: jika Anda melihat kesalahan tentang instalasi lain sedang berlangsung tetapi tidak ada, coba jalankan installer sebagai Administrator.
- **ARM64**: perangkat Windows ARM64 sepenuhnya didukung.

Tab Cowork tidak tersedia di Mac Intel

Tab Cowork memerlukan Apple Silicon (M1 atau lebih baru) di macOS. Di Windows, Cowork tersedia di semua perangkat keras yang didukung. Tab Chat dan Code berfungsi normal di Mac Intel.

"Cabang belum ada" saat membuka di CLI

Sesi jarak jauh dapat membuat cabang yang tidak ada di mesin lokal Anda. Klik nama cabang di toolbar sesi untuk menyalinnya, kemudian ambil secara lokal:

```
git fetch origin <branch-name>
git checkout <branch-name>
```

Masih terjebak?

- Cari atau laporkan bug di GitHub Issues
- Kunjungi pusat dukungan Claude

Saat melaporkan bug, sertakan versi aplikasi desktop Anda, sistem operasi Anda, pesan kesalahan yang tepat, dan log yang relevan. Di macOS, periksa Console.app. Di Windows, periksa Event Viewer → Windows Logs → Application.

Gunakan Claude Code dengan Chrome (beta)

> *Hubungkan Claude Code ke browser Chrome Anda untuk menguji aplikasi web, debug dengan console logs, otomatisasi pengisian formulir, dan ekstrak data dari halaman web.*

Claude Code terintegrasi dengan ekstensi Claude in Chrome browser untuk memberikan Anda kemampuan otomasi browser dari CLI atau ekstensi VS Code. Bangun kode Anda, kemudian uji dan debug di browser tanpa beralih konteks.

Claude membuka tab baru untuk tugas browser dan berbagi status login browser Anda, sehingga dapat mengakses situs apa pun yang sudah Anda masuki. Tindakan browser berjalan di jendela Chrome yang terlihat secara real-time. Ketika Claude menemukan halaman login atau CAPTCHA, ia berhenti dan meminta Anda menanganinya secara manual.

Note:

Integrasi Chrome sedang dalam beta dan saat ini hanya bekerja dengan Google Chrome. Belum didukung di Brave, Arc, atau browser berbasis Chromium lainnya. WSL (Windows Subsystem for Linux) juga tidak didukung.

Kemampuan

Dengan Chrome terhubung, Anda dapat menggabungkan tindakan browser dengan tugas coding dalam satu alur kerja:

- **Live debugging**: baca kesalahan console dan status DOM secara langsung, kemudian perbaiki kode yang menyebabkannya
- **Verifikasi desain**: bangun UI dari mock Figma, kemudian buka di browser untuk memverifikasi kecocokannya
- **Pengujian aplikasi web**: uji validasi formulir, periksa regresi visual, atau verifikasi alur pengguna
- **Aplikasi web terautentikasi**: berinteraksi dengan Google Docs, Gmail, Notion, atau aplikasi apa pun yang Anda masuki tanpa konektor API
- **Ekstraksi data**: tarik informasi terstruktur dari halaman web dan simpan secara lokal

- **Otomasi tugas**: otomatisasi tugas browser berulang seperti entri data, pengisian formulir, atau alur kerja multi-situs
- **Perekaman sesi**: rekam interaksi browser sebagai GIF untuk mendokumentasikan atau berbagi apa yang terjadi

Prasyarat

Sebelum menggunakan Claude Code dengan Chrome, Anda memerlukan:

- Browser Google Chrome
- Ekstensi Claude in Chrome versi 1.0.36 atau lebih tinggi
- Claude Code versi 2.0.73 atau lebih tinggi
- Paket Anthropic langsung (Pro, Max, Team, atau Enterprise)

> **Note:**
>
> Integrasi Chrome tidak tersedia melalui penyedia pihak ketiga seperti Amazon Bedrock, Google Cloud Vertex AI, atau Microsoft Foundry. Jika Anda mengakses Claude secara eksklusif melalui penyedia pihak ketiga, Anda memerlukan akun claude.ai terpisah untuk menggunakan fitur ini.

Mulai di CLI

Step 1: Luncurkan Claude Code dengan Chrome

Mulai Claude Code dengan flag `--chrome`:

```
claude --chrome
```

Anda juga dapat mengaktifkan Chrome dari dalam sesi yang ada dengan menjalankan `/chrome`.

Step 2: Minta Claude menggunakan browser

Contoh ini menavigasi ke halaman, berinteraksi dengannya, dan melaporkan apa yang ditemukannya, semuanya dari terminal atau editor Anda:

```
Go to code.claude.com/docs, click on the search box,
type "hooks", and tell me what results appear
```

Jalankan `/chrome` kapan saja untuk memeriksa status koneksi, mengelola izin, atau menghubungkan kembali ekstensi.

Untuk VS Code, lihat otomasi browser di VS Code.

Aktifkan Chrome secara default

Untuk menghindari melewatkan `--chrome` setiap sesi, jalankan `/chrome` dan pilih "Enabled by default".

Di ekstensi VS Code, Chrome tersedia kapan pun ekstensi Chrome diinstal. Tidak ada flag tambahan yang diperlukan.

> **Note:**
>
> Mengaktifkan Chrome secara default di CLI meningkatkan penggunaan konteks karena alat browser selalu dimuat. Jika Anda melihat peningkatan konsumsi konteks, nonaktifkan pengaturan ini dan gunakan `--chrome` hanya saat diperlukan.

Kelola izin situs

Izin tingkat situs diwarisi dari ekstensi Chrome. Kelola izin di pengaturan ekstensi Chrome untuk mengontrol situs mana yang dapat dijelajahi, diklik, dan diketik oleh Claude.

Contoh alur kerja

Contoh-contoh ini menunjukkan cara umum untuk menggabungkan tindakan browser dengan tugas coding. Jalankan `/mcp` dan pilih `claude-in-chrome` untuk melihat daftar lengkap alat browser yang tersedia.

Uji aplikasi web lokal

Saat mengembangkan aplikasi web, minta Claude untuk memverifikasi perubahan Anda berfungsi dengan benar:

```
I just updated the login form validation. Can you open localhost:3000,
try submitting the form with invalid data, and check if the error
messages appear correctly?
```

Claude menavigasi ke server lokal Anda, berinteraksi dengan formulir, dan melaporkan apa yang diamatinya.

Debug dengan console logs

Claude dapat membaca output console untuk membantu mendiagnosis masalah. Beri tahu Claude pola apa yang harus dicari daripada meminta semua output console, karena log dapat sangat panjang:

```
Open the dashboard page and check the console for any errors when
the page loads.
```

Claude membaca pesan console dan dapat memfilter pola atau jenis kesalahan tertentu.

Otomatisasi pengisian formulir

Percepat tugas entri data berulang:

```
I have a spreadsheet of customer contacts in contacts.csv. For each row,
go to the CRM at crm.example.com, click "Add Contact", and fill in the
name, email, and phone fields.
```

Claude membaca file lokal Anda, menavigasi antarmuka web, dan memasukkan data untuk setiap catatan.

Buat draf konten di Google Docs

Gunakan Claude untuk menulis langsung di dokumen Anda tanpa penyiapan API:

```
Draft a project update based on the recent commits and add it to my
Google Doc at docs.google.com/document/d/abc123
```

Claude membuka dokumen, mengklik ke editor, dan mengetik konten. Ini bekerja dengan aplikasi web apa pun yang Anda masuki: Gmail, Notion, Sheets, dan lainnya.

Ekstrak data dari halaman web

Tarik informasi terstruktur dari situs web:

```
Go to the product listings page and extract the name, price, and
availability for each item. Save the results as a CSV file.
```

Claude menavigasi ke halaman, membaca konten, dan mengompilasi data ke dalam format terstruktur.

Jalankan alur kerja multi-situs

Koordinasikan tugas di berbagai situs web:

```
Check my calendar for meetings tomorrow, then for each meeting with
an external attendee, look up their company website and add a note
about what they do.
```

Claude bekerja di seluruh tab untuk mengumpulkan informasi dan menyelesaikan alur kerja.

Rekam GIF demo

Buat rekaman alur interaksi browser yang dapat dibagikan:

```
Record a GIF showing how to complete the checkout flow, from adding
an item to the cart through to the confirmation page.
```

Claude merekam urutan interaksi dan menyimpannya sebagai file GIF.

Troubleshooting

Ekstensi tidak terdeteksi

Jika Claude Code menampilkan “Chrome extension not detected”:

1. Verifikasi ekstensi Chrome diinstal dan diaktifkan di `chrome://extensions`
2. Verifikasi Claude Code terbaru dengan menjalankan `claude --version`
3. Periksa bahwa Chrome sedang berjalan
4. Jalankan `/chrome` dan pilih “Reconnect extension” untuk membangun kembali koneksi
5. Jika masalah berlanjut, restart Claude Code dan Chrome

Pertama kali Anda mengaktifkan integrasi Chrome, Claude Code menginstal file konfigurasi host messaging asli. Chrome membaca file ini saat startup, jadi jika ekstensi tidak terdeteksi pada upaya pertama Anda, restart Chrome untuk mengambil konfigurasi baru.

Jika koneksi masih gagal, verifikasi file konfigurasi host ada di:

- **macOS**: `~/Library/Application Support/Google/Chrome/NativeMessagingHosts/com.anthropic.claude_code_browser_extension.json`

- **Linux**: `~/.config/google-chrome/NativeMessagingHosts/com.anthropic.claude_code_browser_extension.json`
- **Windows**: periksa `HKCU\Software\Google\Chrome\NativeMessagingHosts\` di Windows Registry

Browser tidak merespons

Jika perintah browser Claude berhenti bekerja:

1. Periksa apakah dialog modal (alert, confirm, prompt) memblokir halaman. Dialog JavaScript memblokir peristiwa browser dan mencegah Claude menerima perintah. Tutup dialog secara manual, kemudian beri tahu Claude untuk melanjutkan.
2. Minta Claude membuat tab baru dan coba lagi
3. Restart ekstensi Chrome dengan menonaktifkan dan mengaktifkannya kembali di `chrome://extensions`

Koneksi terputus selama sesi panjang

Service worker ekstensi Chrome dapat menjadi idle selama sesi yang diperpanjang, yang memutus koneksi. Jika alat browser berhenti bekerja setelah periode inaktivitas, jalankan `/chrome` dan pilih "Reconnect extension".

Masalah khusus Windows

Di Windows, Anda mungkin mengalami:

- **Konflik named pipe (EADDRINUSE)**: jika proses lain menggunakan named pipe yang sama, restart Claude Code. Tutup sesi Claude Code lain apa pun yang mungkin menggunakan Chrome.
- **Kesalahan host messaging asli**: jika host messaging asli mogok saat startup, coba instal ulang Claude Code untuk membuat ulang konfigurasi host.

Pesan kesalahan umum

Ini adalah kesalahan yang paling sering dihadapi dan cara menyelesaikannya:

Kesalahan	Penyebab	Perbaikan
"Browser extension is not connected"	Host messaging asli tidak dapat menjangkau ekstensi	Restart Chrome dan Claude Code, kemudian jalankan `/chrome` untuk menghubungkan kembali

Kesalahan	Penyebab	Perbaikan
“Extension not detected”	Ekstensi Chrome tidak diinstal atau dinonaktifkan	Instal atau aktifkan ekstensi di `chrome://extensions`
“No tab available”	Claude mencoba bertindak sebelum tab siap	Minta Claude membuat tab baru dan coba lagi
“Receiving end does not exist”	Service worker ekstensi menjadi idle	Jalankan `/chrome` dan pilih “Reconnect extension”

Lihat juga

- Gunakan Claude Code di VS Code: otomasi browser di ekstensi VS Code
- Referensi CLI: flag baris perintah termasuk `--chrome`
- Alur kerja umum: lebih banyak cara untuk menggunakan Claude Code
- Data dan privasi: bagaimana Claude Code menangani data Anda
- Memulai dengan Claude in Chrome: dokumentasi lengkap untuk ekstensi Chrome, termasuk pintasan keyboard, penjadwalan, dan izin

Claude Code di Slack

> *Delegasikan tugas coding langsung dari workspace Slack Anda*

Claude Code di Slack membawa kekuatan Claude Code langsung ke workspace Slack Anda. Ketika Anda menyebutkan `@Claude` dengan tugas coding, Claude secara otomatis mendeteksi niat dan membuat sesi Claude Code di web, memungkinkan Anda untuk mendelegasikan pekerjaan pengembangan tanpa meninggalkan percakapan tim Anda.

Integrasi ini dibangun di atas aplikasi Claude untuk Slack yang sudah ada tetapi menambahkan perutean cerdas ke Claude Code di web untuk permintaan yang terkait dengan coding.

Kasus penggunaan

- **Investigasi dan perbaikan bug**: Minta Claude untuk menyelidiki dan memperbaiki bug segera setelah dilaporkan di saluran Slack.
- **Review kode cepat dan modifikasi**: Biarkan Claude mengimplementasikan fitur kecil atau refactor kode berdasarkan umpan balik tim.
- **Debugging kolaboratif**: Ketika diskusi tim memberikan konteks penting (misalnya, reproduksi error atau laporan pengguna), Claude dapat menggunakan informasi tersebut untuk menginformasikan pendekatan debugging-nya.
- **Eksekusi tugas paralel**: Mulai tugas coding di Slack sambil melanjutkan pekerjaan lain, menerima notifikasi saat selesai.

Prasyarat

Sebelum menggunakan Claude Code di Slack, pastikan Anda memiliki hal berikut:

Persyaratan	Detail
Claude Plan	Pro, Max, Team, atau Enterprise dengan akses Claude Code (kursi premium)
Claude Code di web	Akses ke Claude Code di web harus diaktifkan
Akun GitHub	Terhubung ke Claude Code di web dengan setidaknya satu repositori yang terautentikasi

Persyaratan	Detail
Autentikasi Slack	Akun Slack Anda tertaut ke akun Claude Anda melalui aplikasi Claude

Menyiapkan Claude Code di Slack

Step 1: Instal Aplikasi Claude di Slack

Administrator workspace harus menginstal aplikasi Claude dari Slack App Marketplace. Kunjungi Slack App Marketplace dan klik “Add to Slack” untuk memulai proses instalasi.

Step 2: Hubungkan akun Claude Anda

Setelah aplikasi diinstal, autentikasi akun Claude individual Anda:

1. Buka aplikasi Claude di Slack dengan mengklik “Claude” di bagian Apps Anda
2. Navigasi ke tab App Home
3. Klik “Connect” untuk menghubungkan akun Slack Anda dengan akun Claude Anda
4. Selesaikan alur autentikasi di browser Anda

Step 3: Konfigurasi Claude Code di web

Pastikan Claude Code di web Anda dikonfigurasi dengan benar:

- Kunjungi claude.ai/code dan masuk dengan akun yang sama yang Anda hubungkan ke Slack
- Hubungkan akun GitHub Anda jika belum terhubung
- Autentikasi setidaknya satu repositori yang ingin Anda gunakan Claude untuk bekerja

Step 4: Pilih mode perutean Anda

Setelah menghubungkan akun Anda, konfigurasi bagaimana Claude menangani pesan Anda di Slack. Navigasi ke Claude App Home di Slack untuk menemukan pengaturan **Routing Mode**.

Mode	Perilaku
Code only	Claude merutkan semua @mentions ke sesi Claude Code. Terbaik untuk tim yang menggunakan Claude di Slack secara eksklusif untuk tugas pengembangan.

Mode	Perilaku
Code + Chat	Claude menganalisis setiap pesan dan secara cerdas merutkan antara Claude Code (untuk tugas coding) dan Claude Chat (untuk penulisan, analisis, dan pertanyaan umum). Terbaik untuk tim yang menginginkan satu titik masuk @Claude untuk semua jenis pekerjaan.

> **Note:**
>
> Dalam mode Code + Chat, jika Claude merutkan pesan ke Chat tetapi Anda menginginkan sesi coding, Anda dapat mengklik “Retry as Code” untuk membuat sesi Claude Code sebagai gantinya. Demikian pula, jika itu dirutkan ke Code tetapi Anda menginginkan sesi Chat, Anda dapat memilih opsi itu di thread tersebut.

Cara kerjanya

Deteksi otomatis

Ketika Anda menyebutkan @Claude di saluran atau thread Slack, Claude secara otomatis menganalisis pesan Anda untuk menentukan apakah itu tugas coding. Jika Claude mendeteksi niat coding, itu akan merutkan permintaan Anda ke Claude Code di web alih-alih merespons sebagai asisten chat biasa.

Anda juga dapat secara eksplisit memberi tahu Claude untuk menangani permintaan sebagai tugas coding, bahkan jika itu tidak secara otomatis mendeteksinya.

> **Note:**
>
> Claude Code di Slack hanya berfungsi di saluran (publik atau pribadi). Itu tidak berfungsi di pesan langsung (DM).

Pengumpulan konteks

Dari thread: Ketika Anda @mention Claude di thread, itu mengumpulkan konteks dari semua pesan di thread tersebut untuk memahami percakapan lengkap.

Dari saluran: Ketika disebutkan langsung di saluran, Claude melihat pesan saluran terbaru untuk konteks yang relevan.

Konteks ini membantu Claude memahami masalah, memilih repositori yang sesuai, dan menginformasikan pendekatan terhadap tugas.

> **Warning:**
>
> Ketika @Claude dipanggil di Slack, Claude diberi akses ke konteks percakapan untuk lebih memahami permintaan Anda. Claude dapat mengikuti arahan dari pesan lain dalam konteks, jadi pengguna harus memastikan untuk hanya menggunakan Claude dalam percakapan Slack yang terpercaya.

Alur sesi

1. **Inisiasi**: Anda @mention Claude dengan permintaan coding
2. **Deteksi**: Claude menganalisis pesan Anda dan mendeteksi niat coding
3. **Pembuatan sesi**: Sesi Claude Code baru dibuat di claude.ai/code
4. **Pembaruan kemajuan**: Claude memposting pembaruan status ke thread Slack Anda saat pekerjaan berlangsung
5. **Penyelesaian**: Saat selesai, Claude @mention Anda dengan ringkasan dan tombol tindakan
6. **Tinjauan**: Klik “View Session” untuk melihat transkrip lengkap, atau “Create PR” untuk membuka pull request

Elemen antarmuka pengguna

App Home

Tab App Home menampilkan status koneksi Anda dan memungkinkan Anda untuk menghubungkan atau memutuskan akun Claude Anda dari Slack.

Tindakan pesan

- **View Session**: Membuka sesi Claude Code lengkap di browser Anda di mana Anda dapat melihat semua pekerjaan yang dilakukan, melanjutkan sesi, atau membuat permintaan tambahan.
- **Create PR**: Membuat pull request langsung dari perubahan sesi.
- **Retry as Code**: Jika Claude awalnya merespons sebagai asisten chat tetapi Anda menginginkan sesi coding, klik tombol ini untuk mencoba ulang permintaan sebagai tugas Claude Code.
- **Change Repo**: Memungkinkan Anda untuk memilih repositori yang berbeda jika Claude memilih dengan tidak benar.

Pemilihan repositori

Claude secara otomatis memilih repositori berdasarkan konteks dari percakapan Slack Anda. Jika beberapa repositori dapat berlaku, Claude dapat menampilkan dropdown yang memungkinkan Anda memilih yang benar.

Akses dan izin

Akses tingkat pengguna

Jenis Akses	Persyaratan
Sesi Claude Code	Setiap pengguna menjalankan sesi di bawah akun Claude mereka sendiri
Penggunaan & Batas Laju	Sesi dihitung terhadap batas rencana pengguna individual
Akses Repositori	Pengguna hanya dapat mengakses repositori yang telah mereka hubungkan secara pribadi
Riwayat Sesi	Sesi muncul di riwayat Claude Code Anda di claude.ai/code

Izin admin workspace

Administrator workspace Slack mengontrol apakah aplikasi Claude dapat diinstal di workspace. Pengguna individual kemudian mengautentikasi dengan akun Claude mereka sendiri untuk menggunakan integrasi.

Apa yang dapat diakses di mana

Di Slack: Anda akan melihat pembaruan status, ringkasan penyelesaian, dan tombol tindakan. Transkrip lengkap disimpan dan selalu dapat diakses.

Di web: Sesi Claude Code lengkap dengan riwayat percakapan lengkap, semua perubahan kode, operasi file, dan kemampuan untuk melanjutkan sesi atau membuat pull request.

Praktik terbaik

Menulis permintaan yang efektif

- **Jadilah spesifik**: Sertakan nama file, nama fungsi, atau pesan error ketika relevan.
- **Berikan konteks**: Sebutkan repositori atau proyek jika tidak jelas dari percakapan.
- **Tentukan kesuksesan**: Jelaskan seperti apa “selesai”—haruskah Claude menulis tes? Memperbarui dokumentasi? Membuat PR?

- **Gunakan thread**: Balas di thread saat membahas bug atau fitur sehingga Claude dapat mengumpulkan konteks lengkap.

Kapan menggunakan Slack vs. web

Gunakan Slack ketika: Konteks sudah ada dalam diskusi Slack, Anda ingin memulai tugas secara asinkron, atau Anda berkolaborasi dengan rekan tim yang membutuhkan visibilitas.

Gunakan web secara langsung ketika: Anda perlu mengunggah file, menginginkan interaksi real-time selama pengembangan, atau bekerja pada tugas yang lebih panjang dan kompleks.

Pemecahan masalah

Sesi tidak dimulai

1. Verifikasi akun Claude Anda terhubung di Claude App Home
2. Periksa bahwa Anda memiliki akses Claude Code di web yang diaktifkan
3. Pastikan Anda memiliki setidaknya satu repositori GitHub yang terhubung ke Claude Code

Repositori tidak ditampilkan

1. Hubungkan repositori di Claude Code di web di claude.ai/code
2. Verifikasi izin GitHub Anda untuk repositori tersebut
3. Coba putuskan dan hubungkan kembali akun GitHub Anda

Repositori yang salah dipilih

1. Klik tombol “Change Repo” untuk memilih repositori yang berbeda
2. Sertakan nama repositori dalam permintaan Anda untuk pemilihan yang lebih akurat

Kesalahan autentikasi

1. Putuskan dan hubungkan kembali akun Claude Anda di App Home
2. Pastikan Anda masuk ke akun Claude yang benar di browser Anda
3. Periksa bahwa rencana Claude Anda mencakup akses Claude Code

Kedaluwarsa sesi

1. Sesi tetap dapat diakses di riwayat Claude Code Anda di web
2. Anda dapat melanjutkan atau mereferensikan sesi masa lalu dari claude.ai/code

Keterbatasan saat ini

- **GitHub saja**: Saat ini mendukung repositori di GitHub.
- **Satu PR sekaligus**: Setiap sesi dapat membuat satu pull request.
- **Batas laju berlaku**: Sesi menggunakan batas laju rencana Claude individual Anda.
- **Akses web diperlukan**: Pengguna harus memiliki akses Claude Code di web; mereka yang tidak memilikinya hanya akan mendapatkan respons chat Claude standar.

Sumber daya terkait

- Claude Code di web Pelajari lebih lanjut tentang Claude Code di web
- Claude untuk Slack Dokumentasi Claude untuk Slack umum
- Slack App Marketplace Instal aplikasi Claude dari Slack Marketplace
- Claude Help Center Dapatkan dukungan tambahan

Claude Code di web

> *Jalankan tugas Claude Code secara asinkron pada infrastruktur cloud yang aman*

> **Note:**
>
> Claude Code di web saat ini dalam pratinjau penelitian.

Apa itu Claude Code di web?

Claude Code di web memungkinkan pengembang untuk memulai Claude Code dari aplikasi Claude. Ini sempurna untuk:

- **Menjawab pertanyaan**: Tanyakan tentang arsitektur kode dan bagaimana fitur di-implementasikan
- **Perbaikan bug dan tugas rutin**: Tugas yang terdefinisi dengan baik yang tidak memerlukan pengarahan sering
- **Pekerjaan paralel**: Tangani beberapa perbaikan bug secara bersamaan
- **Repositori yang tidak ada di mesin lokal Anda**: Bekerja pada kode yang tidak Anda miliki checkout secara lokal
- **Perubahan backend**: Di mana Claude Code dapat menulis tes dan kemudian menulis kode untuk lulus tes tersebut

Claude Code juga tersedia di aplikasi Claude untuk iOS dan Android untuk memulai tugas saat bepergian dan memantau pekerjaan yang sedang berlangsung.

Anda dapat memulai tugas baru di web dari terminal Anda dengan `--remote`, atau teleport sesi web kembali ke terminal Anda untuk melanjutkan secara lokal. Untuk menggunakan antarmuka web sambil menjalankan Claude Code di mesin Anda sendiri alih-alih infrastruktur cloud, lihat Remote Control.

Siapa yang dapat menggunakan Claude Code di web?

Claude Code di web tersedia dalam pratinjau penelitian untuk:

- **Pengguna Pro**
- **Pengguna Max**
- **Pengguna Team**

- **Pengguna Enterprise** dengan kursi premium atau kursi Chat + Claude Code

Memulai

1. Kunjungi claude.ai/code
2. Hubungkan akun GitHub Anda
3. Instal aplikasi Claude GitHub di repositori Anda
4. Pilih lingkungan default Anda
5. Kirimkan tugas pengkodean Anda
6. Tinjau perubahan dalam tampilan diff, ulangi dengan komentar, kemudian buat pull request

Cara kerjanya

Ketika Anda memulai tugas di Claude Code di web:

1. **Kloning repositori**: Repositori Anda dikloning ke mesin virtual yang dikelola Anthropic
2. **Penyiapan lingkungan**: Claude menyiapkan lingkungan cloud yang aman dengan kode Anda, kemudian menjalankan skrip setup Anda jika dikonfigurasi
3. **Konfigurasi jaringan**: Akses internet dikonfigurasi berdasarkan pengaturan Anda
4. **Eksekusi tugas**: Claude menganalisis kode, membuat perubahan, menjalankan tes, dan memeriksa pekerjaannya
5. **Penyelesaian**: Anda diberitahu ketika selesai dan dapat membuat PR dengan perubahan
6. **Hasil**: Perubahan didorong ke cabang, siap untuk pembuatan pull request

Tinjau perubahan dengan tampilan diff

Tampilan diff memungkinkan Anda melihat dengan tepat apa yang diubah Claude sebelum membuat pull request. Alih-alih mengklik “Create PR” untuk meninjau perubahan di GitHub, lihat diff langsung di aplikasi dan ulangi dengan Claude sampai perubahan siap.

Ketika Claude membuat perubahan pada file, indikator statistik diff muncul menunjukkan jumlah baris yang ditambahkan dan dihapus (misalnya, `+12 -1`). Pilih indikator ini untuk membuka penampil diff, yang menampilkan daftar file di sebelah kiri dan perubahan untuk setiap file di sebelah kanan.

Dari tampilan diff, Anda dapat:

- Meninjau perubahan file demi file

- Berkomentar pada perubahan spesifik untuk meminta modifikasi
- Terus mengulangi dengan Claude berdasarkan apa yang Anda lihat

Ini memungkinkan Anda menyempurnakan perubahan melalui beberapa putaran umpan balik tanpa membuat PR draft atau beralih ke GitHub.

Memindahkan tugas antara web dan terminal

Anda dapat memulai tugas baru di web dari terminal Anda, atau menarik sesi web ke terminal Anda untuk melanjutkan secara lokal. Sesi web bertahan bahkan jika Anda menutup laptop Anda, dan Anda dapat memantaunya dari mana saja termasuk aplikasi mobile Claude.

> **Note:**
>
> Handoff sesi adalah satu arah: Anda dapat menarik sesi web ke terminal Anda, tetapi Anda tidak dapat mendorong sesi terminal yang ada ke web. Bendera `--remote` membuat sesi web *baru* untuk repositori Anda saat ini.

Dari terminal ke web

Mulai sesi web dari baris perintah dengan bendera `--remote`:

```
claude --remote "Fix the authentication bug in src/auth/login.ts"
```

Ini membuat sesi web baru di claude.ai. Tugas berjalan di cloud sementara Anda terus bekerja secara lokal. Gunakan `/tasks` untuk memeriksa kemajuan, atau buka sesi di claude.ai atau aplikasi mobile Claude untuk berinteraksi langsung. Dari sana Anda dapat mengarahkan Claude, memberikan umpan balik, atau menjawab pertanyaan seperti percakapan lainnya.

Tips untuk tugas jarak jauh

Rencanakan secara lokal, jalankan dari jarak jauh: Untuk tugas kompleks, mulai Claude dalam plan mode untuk berkolaborasi pada pendekatan, kemudian kirim pekerjaan ke web:

```
claude --permission-mode plan
```

Dalam plan mode, Claude hanya dapat membaca file dan menjelajahi basis kode. Setelah Anda puas dengan rencana, mulai sesi jarak jauh untuk eksekusi otonom:

```
claude --remote "Execute the migration plan in docs/migration-plan.md"
```

Pola ini memberi Anda kontrol atas strategi sambil membiarkan Claude mengeksekusi secara otonom di cloud.

Jalankan tugas secara paralel: Setiap perintah `--remote` membuat sesi web sendiri yang berjalan secara independen. Anda dapat memulai beberapa tugas dan semuanya akan berjalan secara bersamaan dalam sesi terpisah:

```
claude --remote "Fix the flaky test in auth.spec.ts"
claude --remote "Update the API documentation"
claude --remote "Refactor the logger to use structured output"
```

Pantau semua sesi dengan `/tasks`. Ketika sesi selesai, Anda dapat membuat PR dari antarmuka web atau teleport sesi ke terminal Anda untuk melanjutkan bekerja.

Dari web ke terminal

Ada beberapa cara untuk menarik sesi web ke terminal Anda:

- **Menggunakan `/teleport`**: Dari dalam Claude Code, jalankan `/teleport` (atau `/tp`) untuk melihat pemilih interaktif sesi web Anda. Jika Anda memiliki perubahan yang tidak dikomit, Anda akan diminta untuk menyimpannya terlebih dahulu.
- **Menggunakan `--teleport`**: Dari baris perintah, jalankan `claude --teleport` untuk pemilih sesi interaktif, atau `claude --teleport <session-id>` untuk melanjutkan sesi spesifik secara langsung.
- **Dari `/tasks`**: Jalankan `/tasks` untuk melihat sesi latar belakang Anda, kemudian tekan `t` untuk teleport ke salah satunya
- **Dari antarmuka web**: Klik “Open in CLI” untuk menyalin perintah yang dapat Anda tempel ke terminal Anda

Ketika Anda teleport sesi, Claude memverifikasi Anda berada di repositori yang benar, mengambil dan checkout cabang dari sesi jarak jauh, dan memuat riwayat percakapan lengkap ke terminal Anda.

Persyaratan untuk teleporting

Teleport memeriksa persyaratan ini sebelum melanjutkan sesi. Jika ada persyaratan yang tidak terpenuhi, Anda akan melihat kesalahan atau diminta untuk menyelesaikan masalah.

Persya-ratan	Detail
Status git bersih	Direktori kerja Anda harus tidak memiliki perubahan yang tidak dikomit. Teleport meminta Anda untuk menyimpan perubahan jika diperlukan.
Reposito-ri yang benar	Anda harus menjalankan `--teleport` dari checkout repositori yang sama, bukan fork.
Cabang tersedia	Cabang dari sesi web harus telah didorong ke remote. Teleport secara otomatis mengambil dan checkout.
Akun yang sama	Anda harus diautentikasi ke akun Claude.ai yang sama yang digunakan dalam sesi web.

Berbagi sesi

Untuk berbagi sesi, alihkan visibilitasnya sesuai dengan jenis akun di bawah ini. Setelah itu, bagikan tautan sesi apa adanya. Penerima yang membuka sesi bersama Anda akan melihat status terbaru sesi saat dimuat, tetapi halaman penerima tidak akan diperbarui secara real-time.

Berbagi dari akun Enterprise atau Teams

Untuk akun Enterprise dan Teams, dua opsi visibilitas adalah **Private** dan **Team**. Visibilitas Team membuat sesi terlihat oleh anggota lain dari organisasi Claude.ai Anda. Verifikasi akses repositori diaktifkan secara default, berdasarkan akun GitHub yang terhubung ke akun penerima Anda. Nama tampilan akun Anda terlihat oleh semua penerima dengan akses. Sesi Claude in Slack secara otomatis dibagikan dengan visibilitas Team.

Berbagi dari akun Max atau Pro

Untuk akun Max dan Pro, dua opsi visibilitas adalah **Private** dan **Public**. Visibilitas Public membuat sesi terlihat oleh pengguna mana pun yang masuk ke claude.ai.

Periksa sesi Anda untuk konten sensitif sebelum berbagi. Sesi dapat berisi kode dan kredensial dari repositori GitHub pribadi. Verifikasi akses repositori tidak diaktifkan secara default.

Aktifkan verifikasi akses repositori dan/atau tahan nama Anda dari sesi bersama Anda dengan membuka Settings > Claude Code > Sharing settings.

Mengelola sesi

Mengarsipkan sesi

Anda dapat mengarsipkan sesi untuk menjaga daftar sesi Anda tetap terorganisir. Sesi yang diarsipkan disembunyikan dari daftar sesi default tetapi dapat dilihat dengan memfilter sesi yang diarsipkan.

Untuk mengarsipkan sesi, arahkan ke sesi di sidebar dan klik ikon arsip.

Menghapus sesi

Menghapus sesi secara permanen menghapus sesi dan datanya. Tindakan ini tidak dapat dibatalkan. Anda dapat menghapus sesi dengan dua cara:

- **Dari sidebar**: Filter untuk sesi yang diarsipkan, kemudian arahkan ke sesi yang ingin Anda hapus dan klik ikon hapus
- **Dari menu sesi**: Buka sesi, klik dropdown di sebelah judul sesi, dan pilih **Delete**

Anda akan diminta untuk mengonfirmasi sebelum sesi dihapus.

Lingkungan cloud

Gambar default

Kami membangun dan memelihara gambar universal dengan toolchain umum dan ekosistem bahasa yang sudah diinstal sebelumnya. Gambar ini mencakup:

- Bahasa pemrograman dan runtime populer
- Alat build umum dan pengelola paket
- Framework pengujian dan linter

Memeriksa alat yang tersedia

Untuk melihat apa yang sudah diinstal di lingkungan Anda, minta Claude Code untuk menjalankan:

```
check-tools
```

Perintah ini menampilkan:

- Bahasa pemrograman dan versinya
- Pengelola paket yang tersedia
- Alat pengembangan yang diinstal

Penyiapan khusus bahasa

Gambar universal mencakup lingkungan yang sudah dikonfigurasi untuk:

- **Python**: Python 3.x dengan pip, poetry, dan perpustakaan ilmiah umum
- **Node.js**: Versi LTS terbaru dengan npm, yarn, pnpm, dan bun
- **Ruby**: Versi 3.1.6, 3.2.6, 3.3.6 (default: 3.3.6) dengan gem, bundler, dan rbenv untuk manajemen versi
- **PHP**: Versi 8.4.14
- **Java**: OpenJDK dengan Maven dan Gradle
- **Go**: Versi stabil terbaru dengan dukungan modul
- **Rust**: Rust toolchain dengan cargo
- **C++**: Kompiler GCC dan Clang

Database

Gambar universal mencakup database berikut:

- **PostgreSQL**: Versi 16
- **Redis**: Versi 7.0

Konfigurasi lingkungan

Ketika Anda memulai sesi di Claude Code di web, inilah yang terjadi di balik layar:

1. **Persiapan lingkungan**: Kami mengkloning repositori Anda dan menjalankan skrip setup yang dikonfigurasi. Repo akan dikloning dengan cabang default di repo GitHub Anda. Jika Anda ingin checkout cabang spesifik, Anda dapat menentukan itu dalam prompt.
2. **Konfigurasi jaringan**: Kami mengonfigurasi akses internet untuk agen. Akses internet dibatasi secara default, tetapi Anda dapat mengonfigurasi lingkungan untuk tidak memiliki internet atau akses internet penuh berdasarkan kebutuhan Anda.
3. **Eksekusi Claude Code**: Claude Code berjalan untuk menyelesaikan tugas Anda, menulis kode, menjalankan tes, dan memeriksa pekerjaannya. Anda dapat memandu dan mengarahkan Claude sepanjang sesi melalui antarmuka web. Claude menghormati konteks yang telah Anda tentukan di `CLAUDE.md` Anda.
4. **Hasil**: Ketika Claude menyelesaikan pekerjaannya, itu akan mendorong cabang ke remote. Anda akan dapat membuat PR untuk cabang tersebut.

> **Note:**

> Claude beroperasi sepenuhnya melalui terminal dan alat CLI yang tersedia di lingkungan. Ini menggunakan alat yang sudah diinstal sebelumnya di gambar universal dan alat tambahan apa pun yang Anda instal melalui hooks atau manajemen dependensi.

Untuk menambahkan lingkungan baru: Pilih lingkungan saat ini untuk membuka pemilih lingkungan, kemudian pilih "Add environment". Ini akan membuka dialog di mana Anda dapat menentukan nama lingkungan, tingkat akses jaringan, variabel lingkungan, dan skrip setup.

Untuk memperbarui lingkungan yang ada: Pilih lingkungan saat ini, di sebelah kanan nama lingkungan, dan pilih tombol pengaturan. Ini akan membuka dialog di mana Anda dapat memperbarui nama lingkungan, akses jaringan, variabel lingkungan, dan skrip setup.

Untuk memilih lingkungan default Anda dari terminal: Jika Anda memiliki beberapa lingkungan yang dikonfigurasi, jalankan `/remote-env` untuk memilih mana yang akan digunakan saat memulai sesi web dari terminal Anda dengan `--remote`. Dengan satu lingkungan, perintah ini menunjukkan konfigurasi Anda saat ini.

> **Note:**
>
> Variabel lingkungan harus ditentukan sebagai pasangan kunci-nilai, dalam format `.env`. Misalnya:
>
> ```
> API_KEY=your_api_key
> DEBUG=true
> ```

Setup scripts

Setup script adalah skrip Bash yang berjalan ketika sesi cloud baru dimulai, sebelum Claude Code diluncurkan. Gunakan setup scripts untuk menginstal dependensi, mengonfigurasi alat, atau menyiapkan apa pun yang dibutuhkan lingkungan cloud yang tidak ada di gambar default.

Skrip berjalan sebagai root di Ubuntu 24.04, jadi `apt install` dan sebagian besar pengelola paket bahasa bekerja.

> **Tip:**

> Untuk memeriksa apa yang sudah diinstal sebelum menambahkannya ke skrip Anda, minta Claude untuk menjalankan `check-tools` dalam sesi cloud.

Untuk menambahkan setup script, buka dialog pengaturan lingkungan dan masukkan skrip Anda di bidang **Setup script**.

Contoh ini menginstal CLI `gh`, yang tidak ada di gambar default:

```
#!/bin/bash
apt update && apt install -y gh
```

Setup scripts berjalan hanya saat membuat sesi baru. Mereka dilewati saat melanjutkan sesi yang ada.

Jika skrip keluar dengan non-zero, sesi gagal dimulai. Tambahkan `|| true` ke perintah non-kritis untuk menghindari pemblokiran sesi pada instalasi yang tidak stabil.

> **Note:**
>
> Setup scripts yang menginstal paket memerlukan akses jaringan untuk menjangkau registri. Akses jaringan default memungkinkan koneksi ke registri paket umum termasuk npm, PyPI, RubyGems, dan crates.io. Skrip akan gagal menginstal paket jika lingkungan Anda memiliki akses jaringan yang dinonaktifkan.

Setup scripts vs. SessionStart hooks

Gunakan setup script untuk menginstal hal-hal yang dibutuhkan cloud tetapi laptop Anda sudah memiliki, seperti runtime bahasa atau alat CLI. Gunakan SessionStart hook untuk penyiapan proyek yang harus berjalan di mana-mana, cloud dan lokal, seperti `npm install`.

Keduanya berjalan di awal sesi, tetapi mereka termasuk di tempat yang berbeda:

	Setup scripts	SessionStart hooks
Terlampir pada	Lingkungan cloud	Repositori Anda
Dikonfigurasi di	UI lingkungan cloud	`.claude/settings.json` di repo Anda

	Setup scripts	SessionStart hooks
Berjalan	Sebelum Claude Code diluncurkan, hanya pada sesi baru	Setelah Claude Code diluncurkan, pada setiap sesi termasuk yang dilanjutkan
Cakupan	Hanya lingkungan cloud	Lokal dan cloud

SessionStart hooks juga dapat didefinisikan di `~/.claude/settings.json` tingkat pengguna Anda secara lokal, tetapi pengaturan tingkat pengguna tidak terbawa ke sesi cloud. Di cloud, hanya hooks yang dikomit ke repo yang berjalan.

Manajemen dependensi

Gambar lingkungan kustom dan snapshot belum didukung. Gunakan setup scripts untuk menginstal paket saat sesi dimulai, atau SessionStart hooks untuk instalasi dependensi yang juga harus berjalan di lingkungan lokal. SessionStart hooks memiliki batasan yang diketahui.

Untuk mengonfigurasi instalasi dependensi otomatis dengan setup script, buka pengaturan lingkungan Anda dan tambahkan skrip:

```
#!/bin/bash
npm install
pip install -r requirements.txt
```

Alternatifnya, Anda dapat menggunakan SessionStart hooks di file `.claude/settings.json` repositori Anda untuk instalasi dependensi yang juga harus berjalan di lingkungan lokal:

```
{
  "hooks": {
    "SessionStart": [
      {
        "matcher": "startup",
        "hooks": [
          {
            "type": "command",
            "command": "\"$CLAUDE_PROJECT_DIR\"/scripts/install_pkgs.sh"
          }
        ]
      }
    ]
  }
}
```

Buat skrip yang sesuai di `scripts/install_pkgs.sh`:

```
#!/bin/bash

## Only run in remote environments
if [ "$CLAUDE_CODE_REMOTE" != "true" ]; then
  exit 0
fi

npm install
pip install -r requirements.txt
exit 0
```

Buat dapat dieksekusi: `chmod +x scripts/install_pkgs.sh`

Pertahankan variabel lingkungan

SessionStart hooks dapat mempertahankan variabel lingkungan untuk perintah Bash berikutnya dengan menulis ke file yang ditentukan dalam variabel lingkungan `CLAUDE_ENV_FILE`. Untuk detail, lihat SessionStart hooks dalam referensi hooks.

Batasan manajemen dependensi

- **Hooks api untuk semua sesi**: SessionStart hooks berjalan di lingkungan lokal dan jarak jauh. Tidak ada konfigurasi hook untuk membatasi hook hanya ke sesi jarak jauh. Untuk melewati eksekusi lokal, periksa variabel lingkungan `CLAUDE_CODE_REMOTE` dalam skrip Anda seperti yang ditunjukkan di atas.
- **Memerlukan akses jaringan**: Perintah install memerlukan akses jaringan untuk menjangkau registri paket. Jika lingkungan Anda dikonfigurasi dengan akses "No internet", hooks ini akan gagal. Gunakan akses jaringan "Limited" (default) atau "Full". Daftar putih default mencakup registri umum seperti npm, PyPI, RubyGems, dan crates.io.
- **Kompatibilitas proxy**: Semua lalu lintas keluar di lingkungan jarak jauh melewati security proxy. Beberapa pengelola paket tidak bekerja dengan benar dengan proxy ini. Bun adalah contoh yang diketahui.
- **Berjalan pada setiap awal sesi**: Hooks berjalan setiap kali sesi dimulai atau dilanjutkan, menambah latensi startup. Jaga skrip install tetap cepat dengan memeriksa apakah dependensi sudah ada sebelum menginstal ulang.

Akses jaringan dan keamanan

Kebijakan jaringan

Proxy GitHub

Untuk keamanan, semua operasi GitHub melewati layanan proxy khusus yang secara transparan menangani semua interaksi git. Di dalam sandbox, klien git mengautentikasi menggunakan kredensial bersistem yang dibuat khusus. Proxy ini:

- Mengelola autentikasi GitHub dengan aman - klien git menggunakan kredensial bersistem di dalam sandbox, yang diverifikasi proxy dan diterjemahkan ke token autentikasi GitHub aktual Anda
- Membatasi operasi git push ke cabang kerja saat ini untuk keamanan
- Memungkinkan kloning, pengambilan, dan operasi PR yang mulus sambil mempertahankan batas keamanan

Security proxy

Lingkungan berjalan di belakang proxy jaringan HTTP/HTTPS untuk tujuan keamanan dan pencegahan penyalahgunaan. Semua lalu lintas internet keluar melewati proxy ini, yang menyediakan:

- Perlindungan terhadap permintaan berbahaya

- Pembatasan laju dan pencegahan penyalahgunaan
- Penyaringan konten untuk keamanan yang ditingkatkan

Tingkat akses

Secara default, akses jaringan dibatasi pada domain yang diizinkan.

Anda dapat mengonfigurasi akses jaringan kustom, termasuk menonaktifkan akses jaringan.

Domain yang diizinkan secara default

Saat menggunakan akses jaringan “Limited”, domain berikut diizinkan secara default:

Layanan Anthropic

- api.anthropic.com
- statsig.anthropic.com
- platform.claude.com
- code.claude.com
- claude.ai

Kontrol Versi

- github.com
- www.github.com
- api.github.com
- npm.pkg.github.com
- raw.githubusercontent.com
- pkg-npm.githubusercontent.com
- objects.githubusercontent.com
- codeload.github.com
- avatars.githubusercontent.com
- camo.githubusercontent.com
- gist.github.com
- gitlab.com
- www.gitlab.com
- registry.gitlab.com
- bitbucket.org
- www.bitbucket.org

- api.bitbucket.org

Registri Kontainer

- registry-1.docker.io
- auth.docker.io
- index.docker.io
- hub.docker.com
- www.docker.com
- production.cloudflare.docker.com
- download.docker.com
- gcr.io
- *.gcr.io
- ghcr.io
- mcr.microsoft.com
- *.data.mcr.microsoft.com
- public.ecr.aws

Platform Cloud

- cloud.google.com
- accounts.google.com
- gcloud.google.com
- *.googleapis.com
- storage.googleapis.com
- compute.googleapis.com
- container.googleapis.com
- azure.com
- portal.azure.com
- microsoft.com
- www.microsoft.com
- *.microsoftonline.com
- packages.microsoft.com
- dotnet.microsoft.com
- dot.net
- visualstudio.com

- dev.azure.com
- *.amazonaws.com
- *.api.aws
- oracle.com
- www.oracle.com
- java.com
- www.java.com
- java.net
- www.java.net
- download.oracle.com
- yum.oracle.com

Pengelola Paket - JavaScript/Node

- registry.npmjs.org
- www.npmjs.com
- www.npmjs.org
- npmjs.com
- npmjs.org
- yarnpkg.com
- registry.yarnpkg.com

Pengelola Paket - Python

- pypi.org
- www.pypi.org
- files.pythonhosted.org
- pythonhosted.org
- test.pypi.org
- pypi.python.org
- pypa.io
- www.pypa.io

Pengelola Paket - Ruby

- rubygems.org
- www.rubygems.org

- api.rubygems.org
- index.rubygems.org
- ruby-lang.org
- www.ruby-lang.org
- rubyforge.org
- www.rubyforge.org
- rubyonrails.org
- www.rubyonrails.org
- rvm.io
- get.rvm.io

Pengelola Paket - Rust

- crates.io
- www.crates.io
- index.crates.io
- static.crates.io
- rustup.rs
- static.rust-lang.org
- www.rust-lang.org

Pengelola Paket - Go

- proxy.golang.org
- sum.golang.org
- index.golang.org
- golang.org
- www.golang.org
- goproxy.io
- pkg.go.dev

Pengelola Paket - JVM

- maven.org
- repo.maven.org
- central.maven.org
- repo1.maven.org

- jcenter.bintray.com
- gradle.org
- www.gradle.org
- services.gradle.org
- plugins.gradle.org
- kotlin.org
- www.kotlin.org
- spring.io
- repo.spring.io

Pengelola Paket - Bahasa Lain

- packagist.org (PHP Composer)
- www.packagist.org
- repo.packagist.org
- nuget.org (.NET NuGet)
- www.nuget.org
- api.nuget.org
- pub.dev (Dart/Flutter)
- api.pub.dev
- hex.pm (Elixir/Erlang)
- www.hex.pm
- cpan.org (Perl CPAN)
- www.cpan.org
- metacpan.org
- www.metacpan.org
- api.metacpan.org
- cocoapods.org (iOS/macOS)
- www.cocoapods.org
- cdn.cocoapods.org
- haskell.org
- www.haskell.org
- hackage.haskell.org
- swift.org
- www.swift.org

Distribusi Linux

- archive.ubuntu.com
- security.ubuntu.com
- ubuntu.com
- www.ubuntu.com
- *.ubuntu.com
- ppa.launchpad.net
- launchpad.net
- www.launchpad.net

Alat Pengembangan & Platform

- dl.k8s.io (Kubernetes)
- pkgs.k8s.io
- k8s.io
- www.k8s.io
- releases.hashicorp.com (HashiCorp)
- apt.releases.hashicorp.com
- rpm.releases.hashicorp.com
- archive.releases.hashicorp.com
- hashicorp.com
- www.hashicorp.com
- repo.anaconda.com (Anaconda/Conda)
- conda.anaconda.org
- anaconda.org
- www.anaconda.com
- anaconda.com
- continuum.io
- apache.org (Apache)
- www.apache.org
- archive.apache.org
- downloads.apache.org
- eclipse.org (Eclipse)
- www.eclipse.org

- download.eclipse.org
- nodejs.org (Node.js)
- www.nodejs.org

Layanan Cloud & Monitoring

- statsig.com
- www.statsig.com
- api.statsig.com
- sentry.io
- *.sentry.io
- http-intake.logs.datadoghq.com
- *.datadoghq.com
- *.datadoghq.eu

Pengiriman Konten & Mirror

- sourceforge.net
- *.sourceforge.net
- packagecloud.io
- *.packagecloud.io

Skema & Konfigurasi

- json-schema.org
- www.json-schema.org
- json.schemastore.org
- www.schemastore.org

Model Context Protocol

- *.modelcontextprotocol.io

> **Note:**
>
> Domain yang ditandai dengan `*` menunjukkan pencocokan subdomain wildcard. Misalnya, `*.gcr.io` memungkinkan akses ke subdomain apa pun dari `gcr.io`.

Praktik terbaik keamanan untuk akses jaringan yang disesuaikan

1. **Prinsip privilege minimal**: Hanya aktifkan akses jaringan minimum yang diperlukan
2. **Audit secara teratur**: Tinjau domain yang diizinkan secara berkala
3. **Gunakan HTTPS**: Selalu lebih suka endpoint HTTPS daripada HTTP

Keamanan dan isolasi

Claude Code di web menyediakan jaminan keamanan yang kuat:

- **Mesin virtual terisolasi**: Setiap sesi berjalan di VM yang terisolasi dan dikelola Anthropic
- **Kontrol akses jaringan**: Akses jaringan dibatasi secara default, dan dapat dinonaktifkan

> **Note:**
>
> Saat berjalan dengan akses jaringan dinonaktifkan, Claude Code diizinkan untuk berkomunikasi dengan API Anthropic yang mungkin masih memungkinkan data keluar dari VM Claude Code yang terisolasi.

- **Perlindungan kredensial**: Kredensial sensitif (seperti kredensial git atau kunci penandatanganan) tidak pernah ada di dalam sandbox dengan Claude Code. Autentikasi ditangani melalui proxy aman menggunakan kredensial bersistem
- **Analisis aman**: Kode dianalisis dan dimodifikasi dalam VM terisolasi sebelum membuat PR

Harga dan batas laju

Claude Code di web berbagi batas laju dengan semua penggunaan Claude dan Claude Code lainnya dalam akun Anda. Menjalankan beberapa tugas secara paralel akan mengonsumsi lebih banyak batas laju secara proporsional.

Batasan

- **Autentikasi repositori**: Anda hanya dapat memindahkan sesi dari web ke lokal saat Anda diautentikasi ke akun yang sama
- **Pembatasan platform**: Claude Code di web hanya bekerja dengan kode yang dihosting di GitHub. Repositori GitLab dan non-GitHub lainnya tidak dapat digunakan dengan sesi cloud

Praktik terbaik

1. **Otomatisasi penyiapan lingkungan**: Gunakan setup scripts untuk menginstal dependensi dan mengonfigurasi alat sebelum Claude Code diluncurkan. Untuk skenario yang lebih canggih, konfigurasikan SessionStart hooks.
2. **Dokumentasikan persyaratan**: Tentukan dengan jelas dependensi dan perintah di file `CLAUDE.md` Anda. Jika Anda memiliki file `AGENTS.md`, Anda dapat bersumber di `CLAUDE.md` Anda menggunakan `@AGENTS.md` untuk mempertahankan sumber kebenaran tunggal.

Sumber daya terkait

- Konfigurasi Hooks
- Referensi Pengaturan
- Keamanan
- Penggunaan Data

Part 9: Security & Privacy

Keamanan

> *Pelajari tentang perlindungan keamanan Claude Code dan praktik terbaik untuk penggunaan yang aman.*

Bagaimana kami mendekati keamanan

Fondasi keamanan

Keamanan kode Anda adalah prioritas utama. Claude Code dibangun dengan keamanan sebagai inti, dikembangkan sesuai dengan program keamanan komprehensif Anthropic. Pelajari lebih lanjut dan akses sumber daya (laporan SOC 2 Type 2, sertifikat ISO 27001, dll.) di Anthropic Trust Center.

Arsitektur berbasis izin

Claude Code menggunakan izin baca-saja yang ketat secara default. Ketika tindakan tambahan diperlukan (mengedit file, menjalankan tes, mengeksekusi perintah), Claude Code meminta izin eksplisit. Pengguna mengontrol apakah akan menyetujui tindakan sekali atau mengizinkannya secara otomatis.

Kami merancang Claude Code agar transparan dan aman. Misalnya, kami memerlukan persetujuan untuk perintah bash sebelum mengeksekusinya, memberikan Anda kontrol langsung. Pendekatan ini memungkinkan pengguna dan organisasi untuk mengonfigurasi izin secara langsung.

Untuk konfigurasi izin terperinci, lihat Permissions.

Perlindungan bawaan

Untuk mengurangi risiko dalam sistem agentic:

- **Alat bash bersandbox**: Sandbox perintah bash dengan isolasi filesystem dan jaringan, mengurangi permintaan izin sambil mempertahankan keamanan. Aktifkan dengan `/sandbox` untuk menentukan batas tempat Claude Code dapat bekerja secara otonom
- **Pembatasan akses tulis**: Claude Code hanya dapat menulis ke folder tempat dimulai dan subfolder-nya—tidak dapat memodifikasi file di direktori induk tanpa izin eksplisit. Meskipun Claude Code dapat membaca file di luar direktori kerja (berguna

untuk mengakses perpustakaan sistem dan dependensi), operasi tulis dibatasi ketat pada cakupan proyek, menciptakan batas keamanan yang jelas

- **Mitigasi kelelahan permintaan**: Dukungan untuk allowlisting perintah aman yang sering digunakan per-pengguna, per-codebase, atau per-organisasi
- **Mode Accept Edits**: Batch menerima beberapa edit sambil mempertahankan permintaan izin untuk perintah dengan efek samping

Tanggung jawab pengguna

Claude Code hanya memiliki izin yang Anda berikan. Anda bertanggung jawab untuk meninjau kode dan perintah yang diusulkan untuk keamanan sebelum persetujuan.

Lindungi dari prompt injection

Prompt injection adalah teknik di mana penyerang mencoba mengganti atau memanipulasi instruksi asisten AI dengan menyisipkan teks berbahaya. Claude Code mencakup beberapa perlindungan terhadap serangan ini:

Perlindungan inti

- **Sistem izin**: Operasi sensitif memerlukan persetujuan eksplisit
- **Analisis yang menyadari konteks**: Mendeteksi instruksi yang berpotensi berbahaya dengan menganalisis permintaan lengkap
- **Sanitasi input**: Mencegah command injection dengan memproses input pengguna
- **Daftar blokir perintah**: Memblokir perintah berisiko yang mengambil konten arbitrer dari web seperti `curl` dan `wget` secara default. Ketika secara eksplisit diizinkan, waspadai batasan pola izin

Perlindungan privasi

Kami telah menerapkan beberapa perlindungan untuk melindungi data Anda, termasuk:

- Periode retensi terbatas untuk informasi sensitif (lihat Privacy Center untuk mempelajari lebih lanjut)
- Akses terbatas ke data sesi pengguna
- Kontrol pengguna atas preferensi pelatihan data. Pengguna konsumen dapat mengubah pengaturan privasi mereka kapan saja.

Untuk detail lengkap, silakan tinjau Commercial Terms of Service kami (untuk pengguna Team, Enterprise, dan API) atau Consumer Terms (untuk pengguna Free, Pro, dan Max) dan Privacy Policy.

Perlindungan tambahan

- **Persetujuan permintaan jaringan**: Alat yang membuat permintaan jaringan memerlukan persetujuan pengguna secara default
- **Jendela konteks terisolasi**: Web fetch menggunakan jendela konteks terpisah untuk menghindari injeksi prompt yang berpotensi berbahaya
- **Verifikasi kepercayaan**: Jalankan codebase pertama kali dan server MCP baru memerlukan verifikasi kepercayaan
  - Catatan: Verifikasi kepercayaan dinonaktifkan saat menjalankan secara non-interaktif dengan flag `-p`
- **Deteksi command injection**: Perintah bash yang mencurigakan memerlukan persetujuan manual bahkan jika sebelumnya allowlisted
- **Pencocokan fail-closed**: Perintah yang tidak cocok secara default memerlukan persetujuan manual
- **Deskripsi bahasa alami**: Perintah bash kompleks menyertakan penjelasan untuk pemahaman pengguna
- **Penyimpanan kredensial aman**: Kunci API dan token dienkripsi. Lihat Credential Management

> **Warning:**
>
> **Risiko keamanan Windows WebDAV**: Saat menjalankan Claude Code di Windows, kami merekomendasikan untuk tidak mengaktifkan WebDAV atau mengizinkan Claude Code mengakses path seperti `\\*` yang mungkin berisi subdirektori WebDAV. WebDAV telah dihentikan oleh Microsoft karena risiko keamanan. Mengaktifkan WebDAV dapat memungkinkan Claude Code memicu permintaan jaringan ke host jarak jauh, melewati sistem izin.

Praktik terbaik untuk bekerja dengan konten yang tidak dipercaya:

1. Tinjau perintah yang disarankan sebelum persetujuan
2. Hindari piping konten yang tidak dipercaya langsung ke Claude
3. Verifikasi perubahan yang diusulkan pada file kritis
4. Gunakan mesin virtual (VM) untuk menjalankan skrip dan membuat panggilan alat, terutama saat berinteraksi dengan layanan web eksternal
5. Laporkan perilaku mencurigakan dengan `/bug`

> **Warning:**

> Meskipun perlindungan ini secara signifikan mengurangi risiko, tidak ada sistem yang sepenuhnya kebal terhadap semua serangan. Selalu pertahankan praktik keamanan yang baik saat bekerja dengan alat AI apa pun.

Keamanan MCP

Claude Code memungkinkan pengguna untuk mengonfigurasi server Model Context Protocol (MCP). Daftar server MCP yang diizinkan dikonfigurasi dalam kode sumber Anda, sebagai bagian dari pengaturan Claude Code yang diperiksa insinyur ke dalam kontrol sumber.

Kami mendorong untuk menulis server MCP Anda sendiri atau menggunakan server MCP dari penyedia yang Anda percayai. Anda dapat mengonfigurasi izin Claude Code untuk server MCP. Anthropic tidak mengelola atau mengaudit server MCP apa pun.

Keamanan IDE

Lihat VS Code security and privacy untuk informasi lebih lanjut tentang menjalankan Claude Code di IDE.

Keamanan eksekusi cloud

Saat menggunakan Claude Code di web, kontrol keamanan tambahan tersedia:

- **Mesin virtual terisolasi**: Setiap sesi cloud berjalan di VM yang terisolasi dan dikelola Anthropic
- **Kontrol akses jaringan**: Akses jaringan dibatasi secara default dan dapat dikonfigurasi untuk dinonaktifkan atau hanya mengizinkan domain tertentu
- **Perlindungan kredensial**: Autentikasi ditangani melalui proxy aman yang menggunakan kredensial bersisir di dalam sandbox, yang kemudian diterjemahkan ke token autentikasi GitHub aktual Anda
- **Pembatasan cabang**: Operasi git push dibatasi pada cabang kerja saat ini
- **Pencatatan audit**: Semua operasi di lingkungan cloud dicatat untuk kepatuhan dan tujuan audit
- **Pembersihan otomatis**: Lingkungan cloud secara otomatis dihentikan setelah penyelesaian sesi

Untuk detail lebih lanjut tentang eksekusi cloud, lihat Claude Code di web.

Sesi Remote Control bekerja berbeda: antarmuka web terhubung ke proses Claude Code yang berjalan di mesin lokal Anda. Semua eksekusi kode dan akses file tetap lokal, dan data yang sama yang mengalir selama sesi Claude Code lokal apa pun berjalan melalui Anthropic API melalui TLS. Tidak ada VM cloud atau sandboxing yang terlibat. Koneksi menggunakan beberapa kredensial berumur pendek dengan cakupan sempit, masing-masing dibatasi untuk tujuan tertentu dan kedaluwarsa secara independen, untuk membatasi radius ledakan dari kredensial tunggal yang dikompromikan.

Praktik terbaik keamanan

Bekerja dengan kode sensitif

- Tinjau semua perubahan yang disarankan sebelum persetujuan
- Gunakan pengaturan izin khusus proyek untuk repositori sensitif
- Pertimbangkan menggunakan devcontainers untuk isolasi tambahan
- Audit secara teratur pengaturan izin Anda dengan `/permissions`

Keamanan tim

- Gunakan managed settings untuk menegakkan standar organisasi
- Bagikan konfigurasi izin yang disetujui melalui kontrol versi
- Latih anggota tim tentang praktik terbaik keamanan
- Pantau penggunaan Claude Code melalui OpenTelemetry metrics
- Audit atau blokir perubahan pengaturan selama sesi dengan `ConfigChange` hooks

Melaporkan masalah keamanan

Jika Anda menemukan kerentanan keamanan di Claude Code:

1. Jangan ungkapkan secara publik
2. Laporkan melalui program HackerOne kami
3. Sertakan langkah reproduksi terperinci
4. Berikan waktu bagi kami untuk mengatasi masalah sebelum pengungkapan publik

Sumber daya terkait

- Sandboxing - Isolasi filesystem dan jaringan untuk perintah bash
- Permissions - Konfigurasi izin dan kontrol akses
- Monitoring usage - Lacak dan audit aktivitas Claude Code
- Development containers - Lingkungan yang aman dan terisolasi

- Anthropic Trust Center - Sertifikasi keamanan dan kepatuhan

Sandboxing

> *Pelajari bagaimana alat bash sandboxed Claude Code menyediakan isolasi filesystem dan jaringan untuk eksekusi agen yang lebih aman dan mandiri.*

Ikhtisar

Claude Code menampilkan sandboxing asli untuk menyediakan lingkungan yang lebih aman untuk eksekusi agen sambil mengurangi kebutuhan akan prompt izin yang konstan. Alih-alih meminta izin untuk setiap perintah bash, sandboxing menciptakan batas yang ditentukan di awal di mana Claude Code dapat bekerja lebih bebas dengan risiko yang berkurang.

Alat bash sandboxed menggunakan primitif tingkat OS untuk memberlakukan isolasi filesystem dan jaringan.

Mengapa sandboxing penting

Keamanan berbasis izin tradisional memerlukan persetujuan pengguna yang konstan untuk perintah bash. Meskipun ini memberikan kontrol, hal ini dapat menyebabkan:

- **Kelelahan persetujuan**: Berulang kali mengklik "setujui" dapat menyebabkan pengguna kurang memperhatikan apa yang mereka setujui
- **Produktivitas berkurang**: Gangguan konstan memperlambat alur kerja pengembangan
- **Otonomi terbatas**: Claude Code tidak dapat bekerja seefisien mungkin saat menunggu persetujuan

Sandboxing mengatasi tantangan ini dengan:

1. **Mendefinisikan batas yang jelas**: Tentukan dengan tepat direktori dan host jaringan mana yang dapat diakses Claude Code
2. **Mengurangi prompt izin**: Perintah aman dalam sandbox tidak memerlukan persetujuan
3. **Mempertahankan keamanan**: Upaya untuk mengakses sumber daya di luar sandbox memicu notifikasi segera
4. **Memungkinkan otonomi**: Claude Code dapat berjalan lebih independen dalam batas yang ditentukan

> **Warning:**
>
> Sandboxing yang efektif memerlukan **baik** isolasi filesystem maupun jaringan. Tanpa isolasi jaringan, agen yang dikompromikan dapat mengeksfiltrasikan file sensitif seperti kunci SSH. Tanpa isolasi filesystem, agen yang dikompromikan dapat memasang pintu belakang pada sumber daya sistem untuk mendapatkan akses jaringan. Saat mengonfigurasi sandboxing, penting untuk memastikan bahwa pengaturan yang dikonfigurasi tidak menciptakan bypass dalam sistem ini.

Cara kerjanya

Isolasi filesystem

Alat bash sandboxed membatasi akses sistem file ke direktori tertentu:

- **Perilaku penulisan default**: Akses baca dan tulis ke direktori kerja saat ini dan subdirektorinya
- **Perilaku pembacaan default**: Akses baca ke seluruh komputer, kecuali direktori tertentu yang ditolak
- **Akses terblokir**: Tidak dapat memodifikasi file di luar direktori kerja saat ini tanpa izin eksplisit
- **Dapat dikonfigurasi**: Tentukan jalur yang diizinkan dan ditolak khusus melalui pengaturan

Anda dapat memberikan akses tulis ke jalur tambahan menggunakan `sandbox.filesystem.allowWrite` dalam pengaturan Anda. Pembatasan ini diberlakukan pada tingkat OS (Seatbelt di macOS, bubblewrap di Linux), sehingga berlaku untuk semua perintah subprocess, termasuk alat seperti `kubectl`, `terraform`, dan `npm`, bukan hanya alat file Claude.

Isolasi jaringan

Akses jaringan dikendalikan melalui server proxy yang berjalan di luar sandbox:

- **Pembatasan domain**: Hanya domain yang disetujui yang dapat diakses
- **Konfirmasi pengguna**: Permintaan domain baru memicu prompt izin (kecuali `allowManagedDomainsOnly` diaktifkan, yang secara otomatis memblokir domain yang tidak diizinkan)
- **Dukungan proxy khusus**: Pengguna tingkat lanjut dapat menerapkan aturan khusus pada lalu lintas keluar
- **Cakupan komprehensif**: Pembatasan berlaku untuk semua skrip, program, dan subprocess yang dihasilkan oleh perintah

Penegakan tingkat OS

Alat bash sandboxed memanfaatkan primitif keamanan sistem operasi:

- **macOS**: Menggunakan Seatbelt untuk penegakan sandbox
- **Linux**: Menggunakan bubblewrap untuk isolasi
- **WSL2**: Menggunakan bubblewrap, sama seperti Linux

WSL1 tidak didukung karena bubblewrap memerlukan fitur kernel yang hanya tersedia di WSL2.

Pembatasan tingkat OS ini memastikan bahwa semua proses anak yang dihasilkan oleh perintah Claude Code mewarisi batas keamanan yang sama.

Memulai

Prasyarat

Di **macOS**, sandboxing bekerja langsung menggunakan kerangka Seatbelt bawaan.

Di **Linux dan WSL2**, instal paket yang diperlukan terlebih dahulu:

Ubuntu/Debian

```
sudo apt-get install bubblewrap socat
```

Fedora

```
sudo dnf install bubblewrap socat
```

Aktifkan sandboxing

Anda dapat mengaktifkan sandboxing dengan menjalankan perintah `/sandbox`:

```
/sandbox
```

Ini membuka menu di mana Anda dapat memilih antara mode sandbox. Jika dependensi yang diperlukan hilang (seperti `bubblewrap` atau `socat` di Linux), menu menampilkan instruksi instalasi untuk platform Anda.

Mode sandbox

Claude Code menawarkan dua mode sandbox:

Mode izin otomatis: Perintah Bash akan mencoba berjalan di dalam sandbox dan secara otomatis diizinkan tanpa memerlukan izin. Perintah yang tidak dapat di-sandbox (seperti yang memerlukan akses jaringan ke host yang tidak diizinkan) kembali ke alur izin reguler. Aturan tanya/tolak eksplisit yang telah Anda konfigurasi selalu dihormati.

Mode izin reguler: Semua perintah bash melalui alur izin standar, bahkan saat di-sandbox. Ini memberikan lebih banyak kontrol tetapi memerlukan lebih banyak persetujuan.

Di kedua mode, sandbox memberlakukan pembatasan filesystem dan jaringan yang sama. Perbedaannya hanya dalam apakah perintah sandboxed disetujui secara otomatis atau memerlukan izin eksplisit.

> **Info:**
>
> Mode izin otomatis bekerja secara independen dari pengaturan mode izin Anda. Bahkan jika Anda tidak dalam mode "terima edit", perintah bash sandboxed akan berjalan secara otomatis saat izin otomatis diaktifkan. Ini berarti perintah bash yang memodifikasi file dalam batas sandbox akan dieksekusi tanpa meminta, bahkan ketika alat edit file biasanya memerlukan persetujuan.

Konfigurasi sandboxing

Sesuaikan perilaku sandbox melalui file `settings.json` Anda. Lihat Settings untuk referensi konfigurasi lengkap.

Memberikan akses tulis subprocess ke jalur tertentu

Secara default, perintah sandboxed hanya dapat menulis ke direktori kerja saat ini. Jika perintah subprocess seperti `kubectl`, `terraform`, atau `npm` perlu menulis di luar direktori proyek, gunakan `sandbox.filesystem.allowWrite` untuk memberikan akses ke jalur tertentu:

```
{
  "sandbox": {
    "enabled": true,
    "filesystem": {
      "allowWrite": ["~/.kube", "//tmp/build"]
    }
  }
}
```

Jalur ini diberlakukan pada tingkat OS, sehingga semua perintah yang berjalan di dalam sandbox, termasuk proses anak mereka, menghormatinya. Ini adalah pendekatan yang direkomendasikan ketika alat memerlukan akses tulis ke lokasi tertentu, daripada mengecualikan alat dari sandbox sepenuhnya dengan `excludedCommands`.

Ketika `allowWrite` (atau `denyWrite` / `denyRead`) didefinisikan dalam beberapa cakupan pengaturan, array **digabungkan**, artinya jalur dari setiap cakupan digabungkan, bukan diganti. Misalnya, jika pengaturan terkelola memungkinkan penulisan ke `//opt/company-tools` dan pengguna menambahkan `~/.kube` dalam pengaturan pribadi mereka, kedua jalur disertakan dalam konfigurasi sandbox akhir. Ini berarti pengguna dan proyek dapat memperluas daftar tanpa menduplikasi atau menimpa jalur yang ditetapkan oleh cakupan prioritas lebih tinggi.

Awalan jalur mengontrol bagaimana jalur diselesaikan:

Awalan	Arti	Contoh
`//`	Jalur absolut dari akar filesystem	`//tmp/build` menjadi `/tmp/build`
`~/`	Relatif terhadap direktori home	`~/.kube` menjadi `$HOME/.kube`
`/`	Relatif terhadap direktori file pengaturan	`/build` menjadi `$SETTINGS_DIR/build`
`./` atau tanpa awalan	Jalur relatif (diselesaikan oleh runtime sandbox)	`./output`

Anda juga dapat menolak akses tulis atau baca menggunakan `sandbox.filesystem.denyWrite` dan `sandbox.filesystem.denyRead`. Ini digabungkan dengan jalur apa pun dari aturan izin `Edit(...)` dan `Read(...)`.

> **Tip:**
>
> Tidak semua perintah kompatibel dengan sandboxing langsung. Beberapa catatan yang mungkin membantu Anda memanfaatkan sandbox sebaik-baiknya:
>
> - Banyak alat CLI memerlukan akses ke host tertentu. Saat Anda menggunakan alat ini, mereka akan meminta izin untuk mengakses host tertentu. Memberikan izin akan memungkinkan mereka mengakses host ini sekarang dan di masa depan, memungkinkan mereka untuk dieksekusi dengan aman di dalam sandbox.
> - `watchman` tidak kompatibel dengan berjalan di sandbox. Jika Anda menjalankan `jest`, pertimbangkan menggunakan `jest --no-watchman`

- `docker` tidak kompatibel dengan berjalan di sandbox. Pertimbangkan untuk menentukan `docker` dalam `excludedCommands` untuk memaksanya berjalan di luar sandbox.

Note:

Claude Code mencakup mekanisme pintu keluar yang disengaja yang memungkinkan perintah berjalan di luar sandbox saat diperlukan. Ketika perintah gagal karena pembatasan sandbox (seperti masalah konektivitas jaringan atau alat yang tidak kompatibel), Claude diminta untuk menganalisis kegagalan dan dapat mencoba kembali perintah dengan parameter `dangerouslyDisableSandbox`. Perintah yang menggunakan parameter ini melalui alur izin Claude Code normal yang memerlukan izin pengguna untuk dieksekusi. Ini memungkinkan Claude Code menangani kasus tepi di mana alat tertentu atau operasi jaringan tidak dapat berfungsi dalam batasan sandbox.

Anda dapat menonaktifkan pintu keluar ini dengan mengatur `"allowUnsandboxedCommands": false` dalam pengaturan sandbox Anda. Saat dinonaktifkan, parameter `dangerouslyDisableSandbox` sepenuhnya diabaikan dan semua perintah harus berjalan sandboxed atau secara eksplisit terdaftar dalam `excludedCommands`.

Manfaat keamanan

Perlindungan terhadap prompt injection

Bahkan jika penyerang berhasil memanipulasi perilaku Claude Code melalui prompt injection, sandbox memastikan sistem Anda tetap aman:

Perlindungan filesystem:

- Tidak dapat memodifikasi file konfigurasi kritis seperti `~/.bashrc`
- Tidak dapat memodifikasi file tingkat sistem di `/bin/`
- Tidak dapat membaca file yang ditolak dalam pengaturan izin Claude Anda

Perlindungan jaringan:

- Tidak dapat mengeksfiltrasikan data ke server yang dikendalikan penyerang
- Tidak dapat mengunduh skrip berbahaya dari domain yang tidak sah
- Tidak dapat melakukan panggilan API yang tidak terduga ke layanan yang tidak disetujui
- Tidak dapat menghubungi domain apa pun yang tidak secara eksplisit diizinkan

Pemantauan dan kontrol:

- Semua upaya akses di luar sandbox diblokir pada tingkat OS
- Anda menerima notifikasi segera ketika batas diuji

- Anda dapat memilih untuk menolak, mengizinkan sekali, atau secara permanen memperbarui konfigurasi Anda

Permukaan serangan berkurang

Sandboxing membatasi potensi kerusakan dari:

- **Dependensi berbahaya**: Paket NPM atau dependensi lain dengan kode berbahaya
- **Skrip yang dikompromikan**: Skrip build atau alat dengan kerentanan keamanan
- **Rekayasa sosial**: Serangan yang menipu pengguna untuk menjalankan perintah berbahaya
- **Prompt injection**: Serangan yang menipu Claude untuk menjalankan perintah berbahaya

Operasi transparan

Ketika Claude Code mencoba mengakses sumber daya jaringan di luar sandbox:

1. Operasi diblokir pada tingkat OS
2. Anda menerima notifikasi segera
3. Anda dapat memilih untuk:
    - Menolak permintaan
    - Mengizinkan sekali
    - Memperbarui konfigurasi sandbox Anda untuk secara permanen mengizinkannya

Keterbatasan Keamanan

- Keterbatasan Sandboxing Jaringan: Sistem penyaringan jaringan beroperasi dengan membatasi domain yang diizinkan untuk terhubung oleh proses. Ini tidak sebaliknya memeriksa lalu lintas yang melewati proxy dan pengguna bertanggung jawab untuk memastikan mereka hanya mengizinkan domain tepercaya dalam kebijakan mereka.

> **Warning:**
>
> Pengguna harus menyadari potensi risiko yang datang dari mengizinkan domain luas seperti `github.com` yang mungkin memungkinkan eksfiltrasi data. Juga, dalam beberapa kasus mungkin dapat membypass penyaringan jaringan melalui domain fronting.

- Eskalasi Privilege melalui Unix Sockets: Konfigurasi `allowUnixSockets` dapat secara tidak sengaja memberikan akses ke layanan sistem yang kuat yang dapat menyebabkan bypass sandbox. Misalnya, jika digunakan untuk memungkinkan akses ke `/var/`

`run/docker.sock` ini akan secara efektif memberikan akses ke sistem host melalui eksploitasi soket docker. Pengguna didorong untuk mempertimbangkan dengan hati-hati soket unix apa pun yang mereka izinkan melalui sandbox.

- Eskalasi Izin Filesystem: Izin penulisan filesystem yang terlalu luas dapat memungkinkan serangan eskalasi privilege. Mengizinkan penulisan ke direktori yang berisi executable dalam `$PATH`, direktori konfigurasi sistem, atau file konfigurasi shell pengguna (`.bashrc`, `.zshrc`) dapat menyebabkan eksekusi kode dalam konteks keamanan yang berbeda ketika pengguna lain atau proses sistem mengakses file ini.
- Kekuatan Sandbox Linux: Implementasi Linux menyediakan isolasi filesystem dan jaringan yang kuat tetapi mencakup mode `enableWeakerNestedSandbox` yang memungkinkannya bekerja di dalam lingkungan Docker tanpa namespace istimewa. Opsi ini secara konsiderabel melemahkan keamanan dan hanya boleh digunakan dalam kasus di mana isolasi tambahan sebaliknya diberlakukan.

Bagaimana sandboxing berhubungan dengan izin

Sandboxing dan izin adalah lapisan keamanan komplementer yang bekerja bersama:

- **Izin** mengontrol alat mana yang dapat digunakan Claude Code dan dievaluasi sebelum alat apa pun berjalan. Mereka berlaku untuk semua alat: Bash, Read, Edit, WebFetch, MCP, dan lainnya.
- **Sandboxing** menyediakan penegakan tingkat OS yang membatasi apa yang dapat diakses perintah Bash pada tingkat filesystem dan jaringan. Ini hanya berlaku untuk perintah Bash dan proses anak mereka.

Pembatasan filesystem dan jaringan dikonfigurasi melalui pengaturan sandbox dan aturan izin:

- Gunakan `sandbox.filesystem.allowWrite` untuk memberikan akses tulis subprocess ke jalur di luar direktori kerja
- Gunakan `sandbox.filesystem.denyWrite` dan `sandbox.filesystem.denyRead` untuk memblokir akses subprocess ke jalur tertentu
- Gunakan aturan tolak `Read` dan `Edit` untuk memblokir akses ke file atau direktori tertentu
- Gunakan aturan izin/tolak `WebFetch` untuk mengontrol akses domain
- Gunakan `allowedDomains` sandbox untuk mengontrol domain mana yang dapat dijangkau perintah Bash

Jalur dari pengaturan `sandbox.filesystem` dan aturan izin digabungkan bersama ke dalam konfigurasi sandbox akhir.

Repositori ini mencakup konfigurasi pengaturan pemula untuk skenario penyebaran umum, termasuk contoh khusus sandbox. Gunakan ini sebagai titik awal dan sesuaikan dengan kebutuhan Anda.

Penggunaan lanjutan

Konfigurasi proxy khusus

Untuk organisasi yang memerlukan keamanan jaringan lanjutan, Anda dapat menerapkan proxy khusus untuk:

- Mendekripsi dan memeriksa lalu lintas HTTPS
- Menerapkan aturan penyaringan khusus
- Mencatat semua permintaan jaringan
- Mengintegrasikan dengan infrastruktur keamanan yang ada

```
{
  "sandbox": {
    "network": {
      "httpProxyPort": 8080,
      "socksProxyPort": 8081
    }
  }
}
```

Integrasi dengan alat keamanan yang ada

Alat bash sandboxed bekerja bersama dengan:

- **Aturan izin**: Gabungkan dengan pengaturan izin untuk pertahanan berlapis
- **Kontainer pengembangan**: Gunakan dengan devcontainers untuk isolasi tambahan
- **Kebijakan perusahaan**: Terapkan konfigurasi sandbox melalui pengaturan terkelola

Praktik terbaik

1. **Mulai ketat**: Mulai dengan izin minimal dan perluas sesuai kebutuhan
2. **Pantau log**: Tinjau upaya pelanggaran sandbox untuk memahami kebutuhan Claude Code

3. **Gunakan konfigurasi khusus lingkungan**: Aturan sandbox berbeda untuk konteks pengembangan vs. produksi
4. **Gabungkan dengan izin**: Gunakan sandboxing bersama dengan kebijakan IAM untuk keamanan komprehensif
5. **Konfigurasi uji**: Verifikasi pengaturan sandbox Anda tidak memblokir alur kerja yang sah

Sumber terbuka

Runtime sandbox tersedia sebagai paket npm sumber terbuka untuk digunakan dalam proyek agen Anda sendiri. Ini memungkinkan komunitas agen AI yang lebih luas untuk membangun sistem otonom yang lebih aman dan lebih aman. Ini juga dapat digunakan untuk sandbox program lain yang mungkin ingin Anda jalankan. Misalnya, untuk sandbox server MCP Anda dapat menjalankan:

```
npx @anthropic-ai/sandbox-runtime <command-to-sandbox>
```

Untuk detail implementasi dan kode sumber, kunjungi repositori GitHub.

Keterbatasan

- **Overhead kinerja**: Minimal, tetapi beberapa operasi filesystem mungkin sedikit lebih lambat
- **Kompatibilitas**: Beberapa alat yang memerlukan pola akses sistem tertentu mungkin memerlukan penyesuaian konfigurasi, atau bahkan mungkin perlu dijalankan di luar sandbox
- **Dukungan platform**: Mendukung macOS, Linux, dan WSL2. WSL1 tidak didukung. Dukungan Windows asli sedang direncanakan.

Lihat juga

- Security - Fitur keamanan komprehensif dan praktik terbaik
- Permissions - Konfigurasi izin dan kontrol akses
- Settings - Referensi konfigurasi lengkap
- CLI reference - Opsi baris perintah

Checkpointing

> *Secara otomatis melacak dan membatalkan pengeditan Claude untuk pemulihan cepat dari perubahan yang tidak diinginkan.*

Claude Code secara otomatis melacak pengeditan file Claude saat Anda bekerja, memungkinkan Anda dengan cepat membatalkan perubahan dan kembali ke status sebelumnya jika ada yang tidak sesuai rencana.

Cara kerja checkpoint

Saat Anda bekerja dengan Claude, checkpointing secara otomatis menangkap status kode Anda sebelum setiap pengeditan. Jaring pengaman ini memungkinkan Anda mengejar tugas-tugas yang ambisius dan berskala besar dengan mengetahui Anda selalu dapat kembali ke status kode sebelumnya.

Pelacakan otomatis

Claude Code melacak semua perubahan yang dibuat oleh alat pengeditan filenya:

- Setiap prompt pengguna membuat checkpoint baru
- Checkpoint bertahan di seluruh sesi, sehingga Anda dapat mengaksesnya dalam percakapan yang dilanjutkan
- Secara otomatis dibersihkan bersama dengan sesi setelah 30 hari (dapat dikonfigurasi)

Membatalkan perubahan

Tekan `Esc` dua kali (`Esc` + `Esc`) atau gunakan perintah `/rewind` untuk membuka menu rewind. Anda dapat memilih untuk mengembalikan:

- **Percakapan saja**: Kembali ke pesan pengguna sambil mempertahankan perubahan kode
- **Kode saja**: Kembalikan perubahan file sambil mempertahankan percakapan
- **Kode dan percakapan**: Kembalikan keduanya ke titik sebelumnya dalam sesi

Kasus penggunaan umum

Checkpoint sangat berguna ketika:

- **Menjelajahi alternatif**: Coba pendekatan implementasi yang berbeda tanpa kehilangan titik awal Anda
- **Memulihkan dari kesalahan**: Dengan cepat batalkan perubahan yang memperkenalkan bug atau merusak fungsionalitas
- **Iterasi pada fitur**: Bereksperimen dengan variasi dengan mengetahui Anda dapat kembali ke status yang berfungsi

Keterbatasan

Perubahan perintah Bash tidak dilacak

Checkpointing tidak melacak file yang dimodifikasi oleh perintah bash. Misalnya, jika Claude Code menjalankan:

```
rm file.txt
mv old.txt new.txt
cp source.txt dest.txt
```

Modifikasi file ini tidak dapat dibatalkan melalui rewind. Hanya pengeditan file langsung yang dibuat melalui alat pengeditan file Claude yang dilacak.

Perubahan eksternal tidak dilacak

Checkpointing hanya melacak file yang telah diedit dalam sesi saat ini. Perubahan manual yang Anda buat pada file di luar Claude Code dan pengeditan dari sesi bersamaan lainnya biasanya tidak ditangkap, kecuali jika kebetulan mereka memodifikasi file yang sama dengan sesi saat ini.

Bukan pengganti kontrol versi

Checkpoint dirancang untuk pemulihan cepat tingkat sesi. Untuk riwayat versi permanen dan kolaborasi:

- Terus gunakan kontrol versi (mis. Git) untuk commit, branch, dan riwayat jangka panjang
- Checkpoint melengkapi tetapi tidak menggantikan kontrol versi yang tepat
- Pikirkan checkpoint sebagai “undo lokal” dan Git sebagai “riwayat permanen”

Lihat juga

- Mode interaktif - Pintasan keyboard dan kontrol sesi
- Perintah bawaan - Mengakses checkpoint menggunakan `/rewind`
- Referensi CLI - Opsi baris perintah

Penggunaan data

> *Pelajari kebijakan penggunaan data Anthropic untuk Claude*

Kebijakan data

Kebijakan pelatihan data

Pengguna konsumen (paket Free, Pro, dan Max): Kami memberi Anda pilihan untuk mengizinkan data Anda digunakan untuk meningkatkan model Claude di masa depan. Kami akan melatih model baru menggunakan data dari akun Free, Pro, dan Max ketika pengaturan ini aktif (termasuk ketika Anda menggunakan Claude Code dari akun-akun ini).

Pengguna komersial: (paket Team dan Enterprise, API, platform pihak ketiga, dan Claude Gov) mempertahankan kebijakan yang ada: Anthropic tidak melatih model generatif menggunakan kode atau prompt yang dikirim ke Claude Code berdasarkan syarat komersial, kecuali pelanggan telah memilih untuk memberikan data mereka kepada kami untuk peningkatan model (misalnya, Program Mitra Pengembang).

Program Mitra Pengembang

Jika Anda secara eksplisit memilih untuk memberikan materi kepada kami untuk dilatih, seperti melalui Program Mitra Pengembang, kami dapat menggunakan materi tersebut untuk melatih model kami. Admin organisasi dapat secara tegas memilih untuk bergabung dengan Program Mitra Pengembang untuk organisasi mereka. Perhatikan bahwa program ini hanya tersedia untuk API pihak pertama Anthropic, dan bukan untuk pengguna Bedrock atau Vertex.

Umpan balik menggunakan perintah `/bug`

Jika Anda memilih untuk mengirimkan umpan balik kepada kami tentang Claude Code menggunakan perintah `/bug`, kami dapat menggunakan umpan balik Anda untuk meningkatkan produk dan layanan kami. Transkrip yang dibagikan melalui `/bug` disimpan selama 5 tahun.

Survei kualitas sesi

Ketika Anda melihat prompt “Bagaimana Claude melakukan ini di sesi ini?” di Claude Code, merespons survei ini (termasuk memilih “Abaikan”), hanya peringkat numerik Anda (1, 2, 3, atau abaikan) yang dicatat. Kami tidak mengumpulkan atau menyimpan transkrip percakapan, input, output, atau data sesi lainnya sebagai bagian dari survei ini. Tidak seperti umpan balik jempol ke atas/ke bawah atau laporan `/bug`, survei kualitas sesi ini adalah metrik kepuasan produk sederhana. Respons Anda terhadap survei ini tidak mempengaruhi preferensi pelatihan data Anda dan tidak dapat digunakan untuk melatih model AI kami.

Untuk menonaktifkan survei ini, atur `CLAUDE_CODE_DISABLE_FEEDBACK_SURVEY=1`. Survei juga secara otomatis dinonaktifkan saat menggunakan penyedia pihak ketiga (Bedrock, Vertex, Foundry) atau ketika telemetri dinonaktifkan.

Retensi data

Anthropic menyimpan data Claude Code berdasarkan jenis akun dan preferensi Anda.

Pengguna konsumen (paket Free, Pro, dan Max):

- Pengguna yang mengizinkan penggunaan data untuk peningkatan model: periode retensi 5 tahun untuk mendukung pengembangan model dan peningkatan keamanan
- Pengguna yang tidak mengizinkan penggunaan data untuk peningkatan model: periode retensi 30 hari
- Pengaturan privasi dapat diubah kapan saja di claude.ai/settings/data-privacy-controls.

Pengguna komersial (Team, Enterprise, dan API):

- Standar: periode retensi 30 hari
- Retensi data nol: tersedia untuk Claude Code di Claude untuk Enterprise. ZDR diaktifkan berdasarkan per-organisasi; setiap organisasi baru harus memiliki ZDR diaktifkan secara terpisah oleh tim akun Anda
- Penyimpanan lokal: klien Claude Code dapat menyimpan sesi secara lokal hingga 30 hari untuk memungkinkan pemulihan sesi (dapat dikonfigurasi)

Anda dapat menghapus sesi Claude Code individual di web kapan saja. Menghapus sesi secara permanen menghapus data peristiwa sesi. Untuk instruksi tentang cara menghapus sesi, lihat Mengelola sesi.

Pelajari lebih lanjut tentang praktik retensi data di Pusat Privasi kami.

Untuk detail lengkap, silakan tinjau Syarat Layanan Komersial kami (untuk pengguna Team, Enterprise, dan API) atau Syarat Konsumen (untuk pengguna Free, Pro, dan Max) dan Kebijakan Privasi.

Akses data

Untuk semua pengguna pihak pertama, Anda dapat mempelajari lebih lanjut tentang data apa yang dicatat untuk Claude Code lokal dan Claude Code jarak jauh. Sesi Kontrol Jarak Jauh mengikuti alur data lokal karena semua eksekusi terjadi di mesin Anda. Perhatikan untuk Claude Code jarak jauh, Claude mengakses repositori tempat Anda memulai sesi Claude Code Anda. Claude tidak mengakses repositori yang telah Anda hubungkan tetapi belum memulai sesi di dalamnya.

Claude Code Lokal: Alur data dan dependensi

Diagram di bawah menunjukkan bagaimana Claude Code terhubung ke layanan eksternal selama instalasi dan operasi normal. Garis solid menunjukkan koneksi yang diperlukan, sementara garis putus-putus mewakili alur data opsional atau yang dimulai pengguna.

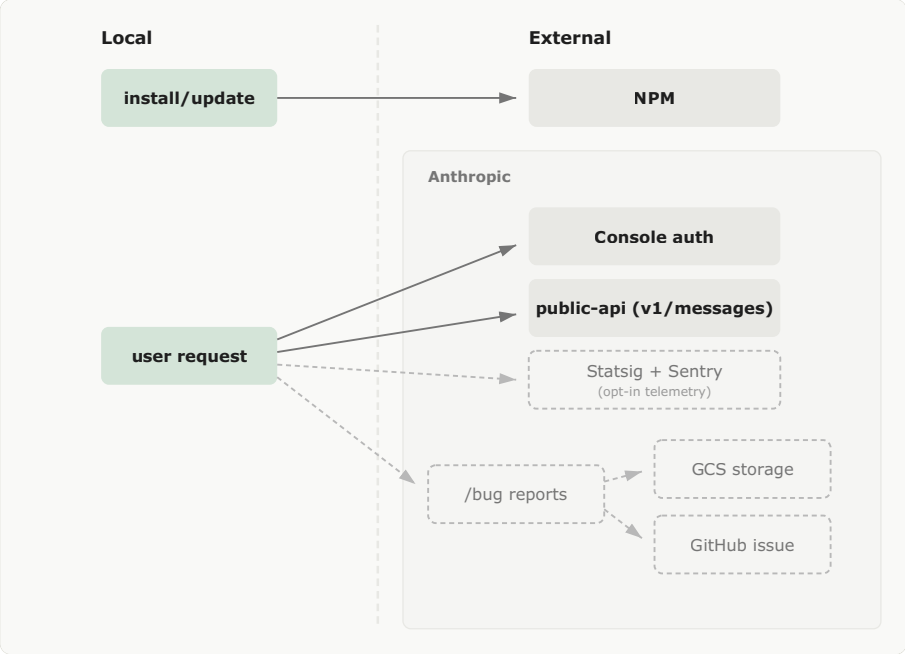


Diagram menunjukkan koneksi eksternal Claude Code: install/update terhubung ke NPM, dan permintaan pengguna terhubung ke layanan Anthropic termasuk auth Console, public-api, dan secara opsional Statsig, Sentry, dan pelaporan bug

Claude Code diinstal dari NPM. Claude Code berjalan secara lokal. Untuk berinteraksi dengan LLM, Claude Code mengirimkan data melalui jaringan. Data ini mencakup semua prompt pengguna dan output model. Data dienkripsi dalam transit melalui TLS dan tidak dienkripsi saat istirahat. Claude Code kompatibel dengan sebagian besar VPN dan proxy LLM populer.

Claude Code dibangun di atas API Anthropic. Untuk detail mengenai kontrol keamanan API kami, termasuk prosedur logging API kami, silakan lihat artefak kepatuhan yang ditawarkan di Pusat Kepercayaan Anthropic.

Eksekusi cloud: Alur data dan dependensi

Saat menggunakan Claude Code di web, sesi berjalan di mesin virtual yang dikelola Anthropic alih-alih secara lokal. Di lingkungan cloud:

- **Penyimpanan kode dan data:** Repositori Anda diklon ke VM terisolasi. Kode dan data sesi tunduk pada kebijakan retensi dan penggunaan untuk jenis akun Anda (lihat bagian Retensi data di atas)
- **Kredensial:** Autentikasi GitHub ditangani melalui proxy aman; kredensial GitHub Anda tidak pernah memasuki sandbox
- **Lalu lintas jaringan:** Semua lalu lintas keluar melewati proxy keamanan untuk logging audit dan pencegahan penyalahgunaan
- **Data sesi:** Prompt, perubahan kode, dan output mengikuti kebijakan data yang sama dengan penggunaan Claude Code lokal

Untuk detail keamanan tentang eksekusi cloud, lihat Keamanan.

Layanan telemetri

Claude Code terhubung dari mesin pengguna ke layanan Statsig untuk mencatat metrik operasional seperti latensi, keandalan, dan pola penggunaan. Logging ini tidak mencakup kode atau jalur file apa pun. Data dienkripsi dalam transit menggunakan TLS dan saat istirahat menggunakan enkripsi AES 256-bit. Baca lebih lanjut di dokumentasi keamanan Statsig. Untuk menolak telemetri Statsig, atur variabel lingkungan `DISABLE_TELEMETRY`.

Claude Code terhubung dari mesin pengguna ke Sentry untuk logging kesalahan operasional. Data dienkripsi dalam transit menggunakan TLS dan saat istirahat menggunakan enkripsi AES 256-bit. Baca lebih lanjut di dokumentasi keamanan Sentry. Untuk menolak logging kesalahan, atur variabel lingkungan `DISABLE_ERROR_REPORTING`.

Ketika pengguna menjalankan perintah `/bug`, salinan riwayat percakapan lengkap mereka termasuk kode dikirim ke Anthropic. Data dienkripsi dalam transit dan saat istirahat. Secara opsional, masalah Github dibuat di repositori publik kami. Untuk menolak pelaporan bug, atur variabel lingkungan `DISABLE_BUG_COMMAND`.

Perilaku default menurut penyedia API

Secara default, kami menonaktifkan semua lalu lintas non-esensial (termasuk pelaporan kesalahan, telemetri, fungsionalitas pelaporan bug, dan survei kualitas sesi) saat menggunakan Bedrock, Vertex, atau Foundry. Anda juga dapat menolak semua ini sekaligus dengan mengatur variabel lingkungan `CLAUDE_CODE_DISABLE_NONESSENTIAL_TRAFFIC`. Berikut adalah perilaku default lengkapnya:

Layanan	Claude API	Vertex API	Bedrock API	Foundry API
Statsig (Metrik)	Default aktif. `DISABLE_TELEMETRY=1` untuk menonaktifkan.	Default nonaktif. `CLAUDE_CODE_USE_VERTEX` harus 1.	Default nonaktif. `CLAUDE_CODE_USE_BEDROCK` harus 1.	Default nonaktif. `CLAUDE_CODE_USE_FOUNDRY` harus 1.
Sentry (Kesalahan)	Default aktif. `DISABLE_ERROR_REPORTING=1` untuk menonaktifkan.	Default nonaktif. `CLAUDE_CODE_USE_VERTEX` harus 1.	Default nonaktif. `CLAUDE_CODE_USE_BEDROCK` harus 1.	Default nonaktif. `CLAUDE_CODE_USE_FOUNDRY` harus 1.
Claude API (laporan `/bug`)	Default aktif. `DISABLE_BUG_COMMAND=1` untuk menonaktifkan.	Default nonaktif. `CLAUDE_CODE_USE_VERTEX` harus 1.	Default nonaktif. `CLAUDE_CODE_USE_BEDROCK` harus 1.	Default nonaktif. `CLAUDE_CODE_USE_FOUNDRY` harus 1.
Survei kualitas sesi	Default aktif. `CLAUDE_CODE_DISABLE_FEEDBACK_SURVEY=1` untuk menonaktifkan.	Default nonaktif. `CLAUDE_CODE_USE_VERTEX` harus 1.	Default nonaktif. `CLAUDE_CODE_USE_BEDROCK` harus 1.	Default nonaktif. `CLAUDE_CODE_USE_FOUNDRY` harus 1.

Semua variabel lingkungan dapat diperiksa ke dalam `settings.json` (baca lebih lanjut).

Hukum dan kepatuhan

> *Perjanjian hukum, sertifikasi kepatuhan, dan informasi keamanan untuk Claude Code.*

Perjanjian hukum

Lisensi

Penggunaan Anda terhadap Claude Code tunduk pada:

- Syarat Komersial - untuk pengguna Team, Enterprise, dan Claude API
- Syarat Konsumen - untuk pengguna Free, Pro, dan Max

Perjanjian komersial

Baik Anda menggunakan Claude API secara langsung (1P) atau mengaksesnya melalui AWS Bedrock atau Google Vertex (3P), perjanjian komersial yang ada akan berlaku untuk penggunaan Claude Code, kecuali kami telah menyetujui sebaliknya.

Kepatuhan

Kepatuhan kesehatan (BAA)

Jika pelanggan memiliki Business Associate Agreement (BAA) dengan kami, dan ingin menggunakan Claude Code, BAA akan secara otomatis diperluas untuk mencakup Claude Code jika pelanggan telah menjalankan BAA dan memiliki Zero Data Retention (ZDR) di-aktifkan. BAA akan berlaku untuk lalu lintas API pelanggan tersebut yang mengalir melalui Claude Code.

Keamanan dan kepercayaan

Kepercayaan dan keselamatan

Anda dapat menemukan informasi lebih lanjut di Pusat Kepercayaan Anthropic dan Hub Transparansi.

Pelaporan kerentanan keamanan

Anthropic mengelola program keamanan kami melalui HackerOne. Gunakan formulir ini untuk melaporkan kerentanan.

Retensi data nol

> *Pelajari tentang Zero Data Retention (ZDR) untuk Claude Code di Claude for Enterprise, termasuk cakupan, fitur yang dinonaktifkan, dan cara meminta pengaktifan.*

Zero Data Retention (ZDR) tersedia untuk Claude Code ketika digunakan melalui Claude for Enterprise. Ketika ZDR diaktifkan, prompt dan respons model yang dihasilkan selama sesi Claude Code diproses secara real-time dan tidak disimpan oleh Anthropic setelah respons dikembalikan, kecuali jika diperlukan untuk mematuhi hukum atau memerangi penyalahgunaan.

ZDR di Claude for Enterprise memberikan pelanggan enterprise kemampuan untuk menggunakan Claude Code dengan retensi data nol dan mengakses kemampuan administratif:

- Kontrol biaya per pengguna
- Dashboard Analytics
- Server-managed settings
- Audit logs

ZDR untuk Claude Code di Claude for Enterprise hanya berlaku untuk platform langsung Anthropic. Untuk penerapan Claude di AWS Bedrock, Google Vertex AI, atau Microsoft Foundry, lihat kebijakan retensi data platform tersebut.

Cakupan ZDR

ZDR mencakup inferensi Claude Code di Claude for Enterprise.

> **Warning:**
>
> ZDR diaktifkan berdasarkan per-organisasi. Setiap organisasi baru memerlukan ZDR untuk diaktifkan secara terpisah oleh tim akun Anthropic Anda. ZDR tidak secara otomatis berlaku untuk organisasi baru yang dibuat di bawah akun yang sama. Hubungi tim akun Anda untuk mengaktifkan ZDR untuk organisasi baru apa pun.

Apa yang dicakup ZDR

ZDR mencakup panggilan inferensi model yang dilakukan melalui Claude Code di Claude for Enterprise. Ketika Anda menggunakan Claude Code di terminal Anda, prompt yang Anda kirim dan respons yang dihasilkan Claude tidak disimpan oleh Anthropic. Ini berlaku terlepas dari model Claude mana yang digunakan.

Apa yang tidak dicakup ZDR

ZDR tidak berlaku untuk hal-hal berikut, bahkan untuk organisasi dengan ZDR diaktifkan. Fitur-fitur ini mengikuti kebijakan retensi data standar:

Fitur	Detail
Chat di claude.ai	Percakapan chat melalui antarmuka web Claude for Enterprise tidak dicakup oleh ZDR.
Cowork	Sesi Cowork tidak dicakup oleh ZDR.
Claude Code Analytics	Tidak menyimpan prompt atau respons model, tetapi mengumpulkan metadata produktivitas seperti email akun dan statistik penggunaan. Metrik kontribusi tidak tersedia untuk organisasi ZDR; dashboard analytics menampilkan metrik penggunaan saja.
Manajemen pengguna dan kursi	Data administratif seperti email akun dan penugasan kursi disimpan di bawah kebijakan standar.
Integrasi pihak ketiga	Data yang diproses oleh alat pihak ketiga, MCP servers, atau integrasi eksternal lainnya tidak dicakup oleh ZDR. Tinjau praktik penanganan data layanan tersebut secara independen.

Fitur yang dinonaktifkan di bawah ZDR

Ketika ZDR diaktifkan untuk organisasi Claude Code di Claude for Enterprise, fitur-fitur tertentu yang memerlukan penyimpanan prompt atau completion secara otomatis dinonaktifkan di tingkat backend:

Fitur	Alasan
Claude Code di Web	Memerlukan penyimpanan percakapan di sisi server.
Remote sessions dari aplikasi Desktop	Memerlukan data sesi persisten yang mencakup prompt dan completion.
Pengiriman umpan balik (`/feedback`)	Mengirimkan umpan balik mengirimkan data percakapan ke Anthropic.

Fitur-fitur ini diblokir di backend terlepas dari tampilan sisi klien. Jika Anda melihat fitur yang dinonaktifkan di terminal Claude Code selama startup, mencoba menggunakannya mengembalikan kesalahan yang menunjukkan kebijakan organisasi tidak memungkinkan tindakan tersebut.

Fitur-fitur di masa depan juga dapat dinonaktifkan jika memerlukan penyimpanan prompt atau completion.

Retensi data untuk pelanggaran kebijakan

Bahkan dengan ZDR diaktifkan, Anthropic dapat menyimpan data jika diperlukan oleh hukum atau untuk mengatasi pelanggaran Usage Policy. Jika sesi ditandai untuk pelanggaran kebijakan, Anthropic dapat menyimpan input dan output terkait selama hingga 2 tahun, konsisten dengan kebijakan ZDR standar Anthropic.

Minta ZDR

Untuk meminta ZDR untuk Claude Code di Claude for Enterprise, hubungi tim akun Anthropic Anda. Tim akun Anda akan mengirimkan permintaan secara internal, dan Anthropic akan meninjau dan mengaktifkan ZDR di organisasi Anda setelah mengkonfirmasi kelayakan. Semua tindakan pengaktifan dicatat dalam audit log.

Jika Anda saat ini menggunakan ZDR untuk Claude Code melalui kunci API pay-as-you-go, Anda dapat beralih ke Claude for Enterprise untuk mendapatkan akses ke fitur administratif sambil mempertahankan ZDR untuk Claude Code. Hubungi tim akun Anda untuk mengoordinasikan migrasi.

Part 10: Enterprise & Monitoring

Lacak penggunaan tim dengan analitik

> *Lihat metrik penggunaan Claude Code, lacak adopsi, dan ukur kecepatan teknik dalam dasbor analitik.*

Claude Code menyediakan dasbor analitik untuk membantu organisasi memahami pola penggunaan pengembang, melacak metrik kontribusi, dan mengukur bagaimana Claude Code mempengaruhi kecepatan teknik. Akses dasbor untuk paket Anda:

Paket	URL Dasbor	Mencakup	Baca selengkapnya
Claude for Teams / Enterprise	claude.ai/analytics/claude-code	Metrik penggunaan, metrik kontribusi dengan integrasi GitHub, papan peringkat, ekspor data	Detail
API (Claude Console)	platform.claude.com/claude-code	Metrik penggunaan, pelacakan pengeluaran, wawasan tim	Detail

Akses analitik untuk Teams dan Enterprise

Navigasikan ke claude.ai/analytics/claude-code. Admin dan Pemilik dapat melihat dasbor.

Dasbor Teams dan Enterprise mencakup:

- **Metrik penggunaan**: baris kode yang diterima, tingkat penerimaan saran, pengguna aktif harian dan sesi
- **Metrik kontribusi**: PR dan baris kode yang dikirim dengan bantuan Claude Code, dengan integrasi GitHub
- **Papan peringkat**: kontributor teratas yang diperingkat berdasarkan penggunaan Claude Code
- **Ekspor data**: unduh data kontribusi sebagai CSV untuk pelaporan khusus

Aktifkan metrik kontribusi

> **Note:**
>
> Metrik kontribusi berada dalam beta publik dan tersedia di paket Claude for Teams dan Claude for Enterprise. Metrik ini hanya mencakup pengguna dalam organisasi claude.ai Anda. Penggunaan melalui API Claude Console atau integrasi pihak ketiga tidak disertakan.

Data penggunaan dan adopsi tersedia untuk semua akun Claude for Teams dan Claude for Enterprise. Metrik kontribusi memerlukan pengaturan tambahan untuk menghubungkan organisasi GitHub Anda.

Anda memerlukan peran Pemilik untuk mengonfigurasi pengaturan analitik. Admin GitHub harus memasang aplikasi GitHub.

> **Warning:**
>
> Metrik kontribusi tidak tersedia untuk organisasi dengan Zero Data Retention diaktifkan. Dasbor analitik hanya akan menampilkan metrik penggunaan.

Step 1: Pasang aplikasi GitHub

Admin GitHub memasang aplikasi Claude GitHub di akun GitHub organisasi Anda di github.com/apps/claude.

Step 2: Aktifkan analitik Claude Code

Pemilik Claude menavigasi ke claude.ai/admin-settings/claude-code dan mengaktifkan fitur analitik Claude Code.

Step 3: Aktifkan analitik GitHub

Di halaman yang sama, aktifkan toggle “GitHub analytics”.

Step 4: Autentikasi dengan GitHub

Selesaikan alur autentikasi GitHub dan pilih organisasi GitHub mana yang akan disertakan dalam analisis.

Data biasanya muncul dalam 24 jam setelah diaktifkan, dengan pembaruan harian. Jika tidak ada data yang muncul, Anda mungkin melihat salah satu pesan ini:

- **“GitHub app required”**: pasang aplikasi GitHub untuk melihat metrik kontribusi
- **“Data processing in progress”**: periksa kembali dalam beberapa hari dan konfirmasi aplikasi GitHub terpasang jika data tidak muncul

Metrik kontribusi mendukung GitHub Cloud dan GitHub Enterprise Server.

Tinjau metrik ringkasan

> **Note:**
>
> Metrik ini sengaja konservatif dan mewakili perkiraan rendah dari dampak sebenarnya Claude Code. Hanya baris dan PR di mana ada kepercayaan tinggi pada keterlibatan Claude Code yang dihitung.

Dasbor menampilkan metrik ringkasan ini di bagian atas:

- **PRs with CC**: jumlah total permintaan tarik yang digabungkan yang berisi setidaknya satu baris kode yang ditulis dengan Claude Code
- **Lines of code with CC**: total baris kode di semua PR yang digabungkan yang ditulis dengan bantuan Claude Code. Hanya "baris efektif" yang dihitung: baris dengan lebih dari 3 karakter setelah normalisasi, tidak termasuk baris kosong dan baris dengan hanya tanda kurung atau tanda baca sepele.
- **PRs with Claude Code (%)**: persentase semua PR yang digabungkan yang berisi kode yang dibantu Claude Code
- **Suggestion accept rate**: persentase waktu pengguna menerima saran pengeditan kode Claude Code, termasuk penggunaan alat Edit, Write, dan NotebookEdit
- **Lines of code accepted**: total baris kode yang ditulis oleh Claude Code yang telah diterima pengguna dalam sesi mereka. Ini tidak termasuk saran yang ditolak dan tidak melacak penghapusan berikutnya.

Jelajahi grafik

Dasbor mencakup beberapa grafik untuk memvisualisasikan tren dari waktu ke waktu.

Lacak adopsi

Grafik Adopsi menunjukkan tren penggunaan harian:

- **users**: pengguna aktif harian
- **sessions**: jumlah sesi Claude Code aktif per hari

Ukur PR per pengguna

Grafik ini menampilkan aktivitas pengembang individu dari waktu ke waktu:

- **PRs per user**: jumlah total PR yang digabungkan per hari dibagi dengan pengguna aktif harian

- **users**: pengguna aktif harian

Gunakan ini untuk memahami bagaimana produktivitas individu berubah seiring dengan meningkatnya adopsi Claude Code.

Lihat rincian permintaan tarik

Grafik Pull requests menunjukkan rincian harian PR yang digabungkan:

- **PRs with CC**: permintaan tarik yang berisi kode yang dibantu Claude Code
- **PRs without CC**: permintaan tarik tanpa kode yang dibantu Claude Code

Alihkan ke tampilan **Lines of code** untuk melihat rincian yang sama berdasarkan baris kode daripada jumlah PR.

Temukan kontributor teratas

Papan Peringkat menampilkan 10 pengguna teratas yang diperingkat berdasarkan volume kontribusi. Alihkan antara:

- **Pull requests**: menampilkan PR dengan Claude Code vs Semua PR untuk setiap pengguna
- **Lines of code**: menampilkan baris dengan Claude Code vs Semua baris untuk setiap pengguna

Klik **Export all users** untuk mengunduh data kontribusi lengkap untuk semua pengguna sebagai file CSV. Ekspor mencakup semua pengguna, bukan hanya 10 teratas yang ditampilkan.

Atribusi PR

Ketika metrik kontribusi diaktifkan, Claude Code menganalisis permintaan tarik yang digabungkan untuk menentukan kode mana yang ditulis dengan bantuan Claude Code. Ini dilakukan dengan mencocokkan aktivitas sesi Claude Code terhadap kode di setiap PR.

Kriteria penandaan

PR ditandai sebagai “with Claude Code” jika berisi setidaknya satu baris kode yang ditulis selama sesi Claude Code. Sistem menggunakan pencocokan konservatif: hanya kode di mana ada kepercayaan tinggi pada keterlibatan Claude Code yang dihitung sebagai dibantu.

Proses atribusi

Ketika permintaan tarik digabungkan:

1. Baris yang ditambahkan diekstrak dari diff PR

2. Sesi Claude Code yang mengedit file yang cocok dalam jendela waktu diidentifikasi
3. Baris PR dicocokkan terhadap output Claude Code menggunakan beberapa strategi
4. Metrik dihitung untuk baris yang dibantu AI dan total baris

Sebelum perbandingan, baris dinormalisasi: spasi dipangkas, beberapa spasi diruntuhkan, tanda kutip distandarkan, dan teks dikonversi ke huruf kecil.

Permintaan tarik yang digabungkan yang berisi baris yang dibantu Claude Code diberi label `claude-code-assisted` di GitHub.

Jendela waktu

Sesi dari 21 hari sebelum hingga 2 hari setelah tanggal penggabungan PR dipertimbangkan untuk pencocokan atribusi.

File yang dikecualikan

File tertentu secara otomatis dikecualikan dari analisis karena mereka dibuat secara otomatis:

- File kunci: package-lock.json, yarn.lock, Cargo.lock, dan serupa
- Kode yang dibuat: output Protobuf, artefak build, file yang diminimalkan
- Direktori build: dist/, build/, node_modules/, target/
- Perlengkapan uji: snapshot, kaset, data mock
- Baris lebih dari 1.000 karakter, yang mungkin diminimalkan atau dibuat

Catatan atribusi

Ingat detail tambahan ini saat menafsirkan data atribusi:

- Kode yang ditulis ulang secara substansial oleh pengembang, dengan perbedaan lebih dari 20%, tidak dikaitkan dengan Claude Code
- Sesi di luar jendela 21 hari tidak dipertimbangkan
- Algoritma tidak mempertimbangkan sumber PR atau cabang tujuan saat melakukan atribusi

Dapatkan yang terbaik dari analitik

Gunakan metrik kontribusi untuk menunjukkan ROI, mengidentifikasi pola adopsi, dan menemukan anggota tim yang dapat membantu orang lain memulai.

Pantau adopsi

Lacak grafik Adopsi dan jumlah pengguna untuk mengidentifikasi:

- Pengguna aktif yang dapat berbagi praktik terbaik
- Tren adopsi keseluruhan di seluruh organisasi Anda
- Penurunan penggunaan yang mungkin menunjukkan gesekan atau masalah

Ukur ROI

Metrik kontribusi membantu menjawab "Apakah alat ini layak untuk investasi?" dengan data dari basis kode Anda sendiri:

- Lacak perubahan dalam PR per pengguna dari waktu ke waktu seiring dengan meningkatnya adopsi
- Bandingkan PR dan baris kode yang dikirim dengan vs. tanpa Claude Code
- Gunakan bersama metrik DORA, kecepatan sprint, atau KPI teknik lainnya untuk memahami perubahan dari adopsi Claude Code

Identifikasi pengguna power

Papan Peringkat membantu Anda menemukan anggota tim dengan adopsi Claude Code tinggi yang dapat:

- Berbagi teknik prompting dan alur kerja dengan tim
- Memberikan umpan balik tentang apa yang berfungsi dengan baik
- Membantu menginisialisasi pengguna baru

Akses data secara terprogram

Untuk menanyakan data ini melalui GitHub, cari PR yang diberi label dengan `claude-code-assisted`.

Akses analitik untuk pelanggan API

Pelanggan API yang menggunakan Claude Console dapat mengakses analitik di platform.claude.com/claude-code. Anda memerlukan izin UsageView untuk mengakses dasbor, yang diberikan kepada peran Developer, Billing, Admin, Owner, dan Primary Owner.

> **Note:**
>
> Metrik kontribusi dengan integrasi GitHub saat ini tidak tersedia untuk pelanggan API. Dasbor Console menampilkan metrik penggunaan dan pengeluaran saja.

Dasbor Console menampilkan:

- **Lines of code accepted**: total baris kode yang ditulis oleh Claude Code yang telah diterima pengguna dalam sesi mereka. Ini tidak termasuk saran yang ditolak dan tidak melacak penghapusan berikutnya.
- **Suggestion accept rate**: persentase waktu pengguna menerima penggunaan alat pengeditan kode, termasuk alat Edit, Write, dan NotebookEdit.
- **Activity**: pengguna aktif harian dan sesi ditampilkan pada grafik.
- **Spend**: biaya API harian dalam dolar bersama jumlah pengguna.

Lihat wawasan tim

Tabel wawasan tim menampilkan metrik per pengguna:

- **Members**: semua pengguna yang telah diautentikasi ke Claude Code. Pengguna kunci API ditampilkan berdasarkan pengidentifikasi kunci, pengguna OAuth ditampilkan berdasarkan alamat email.
- **Spend this month**: total biaya API per pengguna untuk bulan saat ini.
- **Lines this month**: total per pengguna dari baris kode yang diterima untuk bulan saat ini.

> **Note:**
>
> Angka pengeluaran di dasbor Console adalah perkiraan untuk tujuan analitik. Untuk biaya aktual, lihat halaman penagihan Anda.

Sumber daya terkait

- Monitoring with OpenTelemetry: ekspor metrik dan acara real-time ke tumpukan observabilitas Anda
- Manage costs effectively: tetapkan batas pengeluaran dan optimalkan penggunaan token
- Permissions: konfigurasi peran dan izin

Pemantauan

> *Pelajari cara mengaktifkan dan mengonfigurasi OpenTelemetry untuk Claude Code.*

Claude Code mendukung metrik dan acara OpenTelemetry (OTel) untuk pemantauan dan observabilitas.

Semua metrik adalah data deret waktu yang diekspor melalui protokol metrik standar OpenTelemetry, dan acara diekspor melalui protokol log/acara OpenTelemetry. Adalah tanggung jawab pengguna untuk memastikan backend metrik dan log mereka dikonfigurasi dengan benar dan bahwa granularitas agregasi memenuhi persyaratan pemantauan mereka.

Mulai cepat

Konfigurasikan OpenTelemetry menggunakan variabel lingkungan:

```
## 1. Aktifkan telemetri
export CLAUDE_CODE_ENABLE_TELEMETRY=1

## 2. Pilih pengekspor (keduanya bersifat opsional - konfigurasikan hanya yang
Anda butuhkan)
export OTEL_METRICS_EXPORTER=otlp       # Opsi: otlp, prometheus, console
export OTEL_LOGS_EXPORTER=otlp          # Opsi: otlp, console

## 3. Konfigurasikan titik akhir OTLP (untuk pengekspor OTLP)
export OTEL_EXPORTER_OTLP_PROTOCOL=grpc
export OTEL_EXPORTER_OTLP_ENDPOINT=http://localhost:4317

## 4. Atur autentikasi (jika diperlukan)
export OTEL_EXPORTER_OTLP_HEADERS="Authorization=Bearer your-token"

## 5. Untuk debugging: kurangi interval ekspor
export OTEL_METRIC_EXPORT_INTERVAL=10000  # 10 detik (default: 60000ms)
export OTEL_LOGS_EXPORT_INTERVAL=5000     # 5 detik (default: 5000ms)

## 6. Jalankan Claude Code
claude
```

> **Note:**
>
> Interval ekspor default adalah 60 detik untuk metrik dan 5 detik untuk log. Selama pengaturan, Anda mungkin ingin menggunakan interval yang lebih pendek untuk tujuan debugging. Ingat untuk mengatur ulang ini untuk penggunaan produksi.

Untuk opsi konfigurasi lengkap, lihat spesifikasi OpenTelemetry.

Konfigurasi administrator

Administrator dapat mengonfigurasi pengaturan OpenTelemetry untuk semua pengguna melalui file pengaturan terkelola. Ini memungkinkan kontrol terpusat pengaturan telemetri di seluruh organisasi. Lihat prioritas pengaturan untuk informasi lebih lanjut tentang bagaimana pengaturan diterapkan.

Contoh konfigurasi pengaturan terkelola:

```
{
  "env": {
    "CLAUDE_CODE_ENABLE_TELEMETRY": "1",
    "OTEL_METRICS_EXPORTER": "otlp",
    "OTEL_LOGS_EXPORTER": "otlp",
    "OTEL_EXPORTER_OTLP_PROTOCOL": "grpc",
    "OTEL_EXPORTER_OTLP_ENDPOINT": "http://collector.company.com:4317",
    "OTEL_EXPORTER_OTLP_HEADERS": "Authorization=Bearer company-token"
  }
}
```

Note:

Pengaturan terkelola dapat didistribusikan melalui MDM (Mobile Device Management) atau solusi manajemen perangkat lainnya. Variabel lingkungan yang ditentukan dalam file pengaturan terkelola memiliki prioritas tinggi dan tidak dapat ditimpa oleh pengguna.

Detail konfigurasi

Variabel konfigurasi umum

Variabel Lingkungan	Deskripsi	Nilai Contoh
`CLAUDE_CODE_ENABLE_TELEMETRY`	Mengaktifkan pengumpulan telemetri (diperlukan)	`1`
`OTEL_METRICS_EXPORTER`	Jenis pengekspor metrik (dipisahkan koma)	`console`, `otlp`, `prometheus`
`OTEL_LOGS_EXPORTER`	Jenis pengekspor log/acara (dipisahkan koma)	`console`, `otlp`
`OTEL_EXPORTER_OTLP_PROTOCOL`	Protokol untuk pengekspor OTLP (semua sinyal)	`grpc`, `http/json`, `http/protobuf`
`OTEL_EXPORTER_OTLP_ENDPOINT`	Titik akhir pengumpul OTLP (semua sinyal)	`http://localhost:4317`
`OTEL_EXPORTER_OTLP_METRICS_PROTOCOL`	Protokol untuk metrik (menimpa umum)	`grpc`, `http/json`, `http/protobuf`

Variabel Lingkungan	Deskripsi	Nilai Contoh
`OTEL_EXPORTER_OTLP_METRICS_ENDPOINT`	Titik akhir metrik OTLP (menimpa umum)	`http://localhost:4318/v1/metrics`
`OTEL_EXPORTER_OTLP_LOGS_PROTOCOL`	Protokol untuk log (menimpa umum)	`grpc`, `http/json`, `http/protobuf`
`OTEL_EXPORTER_OTLP_LOGS_ENDPOINT`	Titik akhir log OTLP (menimpa umum)	`http://localhost:4318/v1/logs`
`OTEL_EXPORTER_OTLP_HEADERS`	Header autentikasi untuk OTLP	`Authorization=Bearer token`
`OTEL_EXPORTER_OTLP_METRICS_CLIENT_KEY`	Kunci klien untuk autentikasi mTLS	Jalur ke file kunci klien
`OTEL_EXPORTER_OTLP_METRICS_CLIENT_CERTIFICATE`	Sertifikat klien untuk autentikasi mTLS	Jalur ke file sertifikat klien
`OTEL_METRIC_EXPORT_INTERVAL`	Interval ekspor dalam milidetik (default: 60000)	`5000`, `60000`
`OTEL_LOGS_EXPORT_INTERVAL`	Interval ekspor log dalam milidetik (default: 5000)	`1000`, `10000`
`OTEL_LOG_USER_PROMPTS`	Aktifkan pencatatan konten prompt pengguna (default: dinonaktifkan)	`1` untuk mengaktifkan
`CLAUDE_CODE_OTEL_HEADERS_HELPER_DEBOUNCE_MS`	Interval untuk menyegarkan header dinamis (default: 1740000ms / 29 menit)	`900000`

Kontrol kardinalitas metrik

Variabel lingkungan berikut mengontrol atribut mana yang disertakan dalam metrik untuk mengelola kardinalitas:

Variabel Lingkungan	Deskripsi	Nilai Default	Contoh untuk Menonaktifkan
`OTEL_METRICS_INCLUDE_SESSION_ID`	Sertakan atribut session.id dalam metrik	`true`	`false`
`OTEL_METRICS_INCLUDE_VERSION`	Sertakan atribut app.version dalam metrik	`false`	`true`
`OTEL_METRICS_INCLUDE_ACCOUNT_UUID`	Sertakan atribut user.account_uuid dalam metrik	`true`	`false`

Variabel-variabel ini membantu mengontrol kardinalitas metrik, yang mempengaruhi persyaratan penyimpanan dan kinerja kueri di backend metrik Anda. Kardinalitas yang lebih rendah umumnya berarti kinerja yang lebih baik dan biaya penyimpanan yang lebih rendah tetapi data yang kurang granular untuk analisis.

Header dinamis

Untuk lingkungan perusahaan yang memerlukan autentikasi dinamis, Anda dapat mengonfigurasi skrip untuk menghasilkan header secara dinamis:

Konfigurasi pengaturan

Tambahkan ke `.claude/settings.json` Anda:

```
{
  "otelHeadersHelper": "/bin/generate_opentelemetry_headers.sh"
}
```

Persyaratan skrip

Skrip harus menampilkan JSON yang valid dengan pasangan kunci-nilai string yang mewakili header HTTP:

```
#!/bin/bash
## Contoh: Header ganda
echo "{\"Authorization\": \"Bearer $(get-token.sh)\", \"X-API-Key\": \"$(get-api-key.sh)\"}"
```

Perilaku penyegaran

Skrip pembantu header berjalan saat startup dan secara berkala setelahnya untuk mendukung penyegaran token. Secara default, skrip berjalan setiap 29 menit. Sesuaikan interval dengan variabel lingkungan `CLAUDE_CODE_OTEL_HEADERS_HELPER_DEBOUNCE_MS`.

Dukungan organisasi multi-tim

Organisasi dengan beberapa tim atau departemen dapat menambahkan atribut khusus untuk membedakan antara kelompok yang berbeda menggunakan variabel lingkungan `OTEL_RESOURCE_ATTRIBUTES`:

```
## Tambahkan atribut khusus untuk identifikasi tim
export OTEL_RESOURCE_ATTRIBUTES="department=engineering,team.id=platform,cost_center=eng-123"
```

Atribut khusus ini akan disertakan dalam semua metrik dan acara, memungkinkan Anda untuk:

- Filter metrik berdasarkan tim atau departemen
- Lacak biaya per pusat biaya
- Buat dasbor khusus tim
- Atur peringatan untuk tim tertentu

> **Warning:**
>
> **Persyaratan pemformatan penting untuk OTEL_RESOURCE_ATTRIBUTES:**
>
> Variabel lingkungan `OTEL_RESOURCE_ATTRIBUTES` mengikuti spesifikasi W3C Baggage, yang memiliki persyaratan pemformatan yang ketat:
>
> - **Tidak ada spasi yang diizinkan**: Nilai tidak dapat berisi spasi. Misalnya, `user.organizationName=My Company` tidak valid
> - **Format**: Harus berupa pasangan kunci=nilai yang dipisahkan koma: `key1=value1,key2=value2`
> - **Karakter yang diizinkan**: Hanya karakter US-ASCII yang tidak termasuk karakter kontrol, spasi, tanda kutip ganda, koma, titik koma, dan garis miring terbalik

- **Karakter khusus**: Karakter di luar rentang yang diizinkan harus dikodekan persen

Contoh:

```
## ❌ Tidak valid - berisi spasi
export OTEL_RESOURCE_ATTRIBUTES="org.name=John's Organization"

## ✅ Valid - gunakan garis bawah atau camelCase sebagai gantinya
export OTEL_RESOURCE_ATTRIBUTES="org.name=Johns_Organization"
export OTEL_RESOURCE_ATTRIBUTES="org.name=JohnsOrganization"

## ✅ Valid - kodekan persen karakter khusus jika diperlukan
export OTEL_RESOURCE_ATTRIBUTES="org.name=John%27s%20Organization"
```

Catatan: membungkus nilai dalam tanda kutip tidak menghindari spasi. Misalnya, `org.name="My Company"` menghasilkan nilai literal `"My Company"` (dengan tanda kutip disertakan), bukan `My Company`.

Konfigurasi contoh

```
## Debugging konsol (interval 1 detik)
export CLAUDE_CODE_ENABLE_TELEMETRY=1
export OTEL_METRICS_EXPORTER=console
export OTEL_METRIC_EXPORT_INTERVAL=1000

## OTLP/gRPC
export CLAUDE_CODE_ENABLE_TELEMETRY=1
export OTEL_METRICS_EXPORTER=otlp
export OTEL_EXPORTER_OTLP_PROTOCOL=grpc
export OTEL_EXPORTER_OTLP_ENDPOINT=http://localhost:4317

## Prometheus
export CLAUDE_CODE_ENABLE_TELEMETRY=1
export OTEL_METRICS_EXPORTER=prometheus

## Pengekspor ganda
export CLAUDE_CODE_ENABLE_TELEMETRY=1
export OTEL_METRICS_EXPORTER=console,otlp
export OTEL_EXPORTER_OTLP_PROTOCOL=http/json

## Titik akhir/backend berbeda untuk metrik dan log
export CLAUDE_CODE_ENABLE_TELEMETRY=1
export OTEL_METRICS_EXPORTER=otlp
export OTEL_LOGS_EXPORTER=otlp
export OTEL_EXPORTER_OTLP_METRICS_PROTOCOL=http/protobuf
export OTEL_EXPORTER_OTLP_METRICS_ENDPOINT=http://metrics.company.com:4318
export OTEL_EXPORTER_OTLP_LOGS_PROTOCOL=grpc
export OTEL_EXPORTER_OTLP_LOGS_ENDPOINT=http://logs.company.com:4317

## Hanya metrik (tanpa acara/log)
export CLAUDE_CODE_ENABLE_TELEMETRY=1
export OTEL_METRICS_EXPORTER=otlp
export OTEL_EXPORTER_OTLP_PROTOCOL=grpc
export OTEL_EXPORTER_OTLP_ENDPOINT=http://localhost:4317

## Hanya acara/log (tanpa metrik)
export CLAUDE_CODE_ENABLE_TELEMETRY=1
```

```
export OTEL_LOGS_EXPORTER=otlp
export OTEL_EXPORTER_OTLP_PROTOCOL=grpc
export OTEL_EXPORTER_OTLP_ENDPOINT=http://localhost:4317
```

Metrik dan acara yang tersedia

Atribut standar

Semua metrik dan acara berbagi atribut standar ini:

Atribut	Deskripsi	Dikendalikan Oleh
`session.id`	Pengidentifikasi sesi unik	`OTEL_METRICS_INCLUDE_SESSION_ID` (default: true)
`app.version`	Versi Claude Code saat ini	`OTEL_METRICS_INCLUDE_VERSION` (default: false)
`organization.id`	UUID organisasi (saat diautentikasi)	Selalu disertakan saat tersedia
`user.account_uuid`	UUID akun (saat diautentikasi)	`OTEL_METRICS_INCLUDE_ACCOUNT_UUID` (default: true)
`terminal.type`	Jenis terminal (misalnya, `iTerm.app`, `vscode`, `cursor`, `tmux`)	Selalu disertakan saat terdeteksi

Metrik

Claude Code mengekspor metrik berikut:

Nama Metrik	Deskripsi	Unit
`claude_code.session.count`	Jumlah sesi CLI yang dimulai	count
`claude_code.lines_of_code.count`	Jumlah baris kode yang dimodifikasi	count
`claude_code.pull_request.count`	Jumlah permintaan tarik yang dibuat	count

Nama Metrik	Deskripsi	Unit
`claude_code.commit.count`	Jumlah komit git yang dibuat	count
`claude_code.cost.usage`	Biaya sesi Claude Code	USD
`claude_code.token.usage`	Jumlah token yang digunakan	tokens
`claude_code.code_edit_tool.decision`	Jumlah keputusan izin alat pengeditan kode	count
`claude_code.active_time.total`	Total waktu aktif dalam detik	s

Detail metrik

Penghitung sesi

Ditingkatkan pada awal setiap sesi.

Atribut:

- Semua atribut standar

Penghitung baris kode

Ditingkatkan saat kode ditambahkan atau dihapus.

Atribut:

- Semua atribut standar
- `type` : (`"added"` , `"removed"`)

Penghitung permintaan tarik

Ditingkatkan saat membuat permintaan tarik melalui Claude Code.

Atribut:

- Semua atribut standar

Penghitung komit

Ditingkatkan saat membuat komit git melalui Claude Code.

Atribut:

- Semua atribut standar

Penghitung biaya

Ditingkatkan setelah setiap permintaan API.

Atribut:

- Semua atribut standar
- `model` : Pengidentifikasi model (misalnya, “claude-sonnet-4-5-20250929”)

Penghitung token

Ditingkatkan setelah setiap permintaan API.

Atribut:

- Semua atribut standar
- `type` : (`"input"` , `"output"` , `"cacheRead"` , `"cacheCreation"`)
- `model` : Pengidentifikasi model (misalnya, “claude-sonnet-4-5-20250929”)

Penghitung keputusan alat pengeditan kode

Ditingkatkan saat pengguna menerima atau menolak penggunaan alat Edit, Write, atau NotebookEdit.

Atribut:

- Semua atribut standar
- `tool` : Nama alat (`"Edit"` , `"Write"` , `"NotebookEdit"`)
- `decision` : Keputusan pengguna (`"accept"` , `"reject"`)
- `language` : Bahasa pemrograman file yang diedit (misalnya, `"TypeScript"` , `"Python"` , `"JavaScript"` , `"Markdown"`). Mengembalikan `"unknown"` untuk ekstensi file yang tidak dikenali.

Penghitung waktu aktif

Melacak waktu aktual yang dihabiskan untuk menggunakan Claude Code secara aktif (bukan waktu idle). Metrik ini ditingkatkan selama interaksi pengguna seperti mengetik prompt atau menerima respons.

Atribut:

- Semua atribut standar

Acara

Claude Code mengekspor acara berikut melalui log/acara OpenTelemetry (saat `OTEL_LOGS_EXPORTER` dikonfigurasi):

Acara prompt pengguna

Dicatat saat pengguna mengirimkan prompt.

Nama Acara: `claude_code.user_prompt`

Atribut:

- Semua atribut standar
- `event.name` : `"user_prompt"`
- `event.timestamp` : Stempel waktu ISO 8601
- `prompt_length` : Panjang prompt
- `prompt` : Konten prompt (diredaksi secara default, aktifkan dengan `OTEL_LOG_USER_PROMPTS=1`)

Acara hasil alat

Dicatat saat alat menyelesaikan eksekusi.

Nama Acara: `claude_code.tool_result`

Atribut:

- Semua atribut standar
- `event.name` : `"tool_result"`
- `event.timestamp` : Stempel waktu ISO 8601
- `tool_name` : Nama alat
- `success` : `"true"` atau `"false"`
- `duration_ms` : Waktu eksekusi dalam milidetik
- `error` : Pesan kesalahan (jika gagal)
- `decision` : Baik `"accept"` atau `"reject"`
- `source` : Sumber keputusan - `"config"` , `"user_permanent"` , `"user_temporary"` , `"user_abort"` , atau `"user_reject"`
- `tool_parameters` : String JSON yang berisi parameter khusus alat (saat tersedia)
  - Untuk alat Bash: mencakup `bash_command` , `full_command` , `timeout` , `description` , `sandbox`

Acara permintaan API

Dicatat untuk setiap permintaan API ke Claude.

Nama Acara: `claude_code.api_request`

Atribut:

- Semua atribut standar
- `event.name` : `"api_request"`
- `event.timestamp` : Stempel waktu ISO 8601
- `model` : Model yang digunakan (misalnya, “claude-sonnet-4-5-20250929”)
- `cost_usd` : Biaya perkiraan dalam USD
- `duration_ms` : Durasi permintaan dalam milidetik
- `input_tokens` : Jumlah token input
- `output_tokens` : Jumlah token output
- `cache_read_tokens` : Jumlah token yang dibaca dari cache
- `cache_creation_tokens` : Jumlah token yang digunakan untuk pembuatan cache

Acara kesalahan API

Dicatat saat permintaan API ke Claude gagal.

Nama Acara: `claude_code.api_error`

Atribut:

- Semua atribut standar
- `event.name` : `"api_error"`
- `event.timestamp` : Stempel waktu ISO 8601
- `model` : Model yang digunakan (misalnya, “claude-sonnet-4-5-20250929”)
- `error` : Pesan kesalahan
- `status_code` : Kode status HTTP (jika berlaku)
- `duration_ms` : Durasi permintaan dalam milidetik
- `attempt` : Nomor upaya (untuk permintaan yang dicoba ulang)

Acara keputusan alat

Dicatat saat keputusan izin alat dibuat (terima/tolak).

Nama Acara: `claude_code.tool_decision`

Atribut:

- Semua atribut standar
- `event.name` : `"tool_decision"`
- `event.timestamp` : Stempel waktu ISO 8601
- `tool_name` : Nama alat (misalnya, “Read”, “Edit”, “Write”, “NotebookEdit”)
- `decision` : Baik `"accept"` atau `"reject"`
- `source` : Sumber keputusan - `"config"` , `"user_permanent"` , `"user_temporary"` , `"user_abort"` , atau `"user_reject"`

Menafsirkan data metrik dan acara

Metrik yang diekspor oleh Claude Code memberikan wawasan berharga tentang pola penggunaan dan produktivitas. Berikut adalah beberapa visualisasi dan analisis umum yang dapat Anda buat:

Pemantauan penggunaan

Metrik	Peluang Analisis
`claude_code.token.usage`	Pecahkan berdasarkan `type` (input/output), pengguna, tim, atau model
`claude_code.session.count`	Lacak adopsi dan keterlibatan dari waktu ke waktu
`claude_code.lines_of_code.count`	Ukur produktivitas dengan melacak penambahan/penghapusan kode
`claude_code.commit.count` & `claude_code.pull_request.count`	Pahami dampak pada alur kerja pengembangan

Pemantauan biaya

Metrik `claude_code.cost.usage` membantu dengan:

- Melacak tren penggunaan di seluruh tim atau individu
- Mengidentifikasi sesi penggunaan tinggi untuk optimasi

> **Note:**
>
> Metrik biaya adalah perkiraan. Untuk data penagihan resmi, lihat penyedia API Anda (Claude Console, AWS Bedrock, atau Google Cloud Vertex).

Peringatan dan segmentasi

Peringatan umum untuk dipertimbangkan:

- Lonjakan biaya
- Konsumsi token yang tidak biasa
- Volume sesi tinggi dari pengguna tertentu

Semua metrik dapat disegmentasikan berdasarkan `user.account_uuid`, `organization.id`, `session.id`, `model`, dan `app.version`.

Analisis acara

Data acara memberikan wawasan terperinci tentang interaksi Claude Code:

Pola Penggunaan Alat: analisis acara hasil alat untuk mengidentifikasi:

- Alat yang paling sering digunakan
- Tingkat keberhasilan alat
- Waktu eksekusi alat rata-rata
- Pola kesalahan berdasarkan jenis alat

Pemantauan Kinerja: lacak durasi permintaan API dan waktu eksekusi alat untuk mengidentifikasi hambatan kinerja.

Pertimbangan backend

Pilihan backend metrik dan log Anda menentukan jenis analisis yang dapat Anda lakukan:

Untuk metrik

- **Database deret waktu (misalnya, Prometheus)**: Perhitungan laju, metrik agregat
- **Toko kolumnar (misalnya, ClickHouse)**: Kueri kompleks, analisis pengguna unik
- **Platform observabilitas lengkap (misalnya, Honeycomb, Datadog)**: Kueri lanjutan, visualisasi, peringatan

Untuk acara/log

- **Sistem agregasi log (misalnya, Elasticsearch, Loki)**: Pencarian teks lengkap, analisis log
- **Toko kolumnar (misalnya, ClickHouse)**: Analisis acara terstruktur

- **Platform observabilitas lengkap (misalnya, Honeycomb, Datadog)**: Korelasi antara metrik dan acara

Untuk organisasi yang memerlukan metrik Pengguna Aktif Harian/Mingguan/Bulanan (DAU/WAU/MAU), pertimbangkan backend yang mendukung kueri nilai unik yang efisien.

Informasi layanan

Semua metrik dan acara diekspor dengan atribut sumber daya berikut:

- `service.name`: `claude-code`
- `service.version`: Versi Claude Code saat ini
- `os.type`: Jenis sistem operasi (misalnya, `linux`, `darwin`, `windows`)
- `os.version`: String versi sistem operasi
- `host.arch`: Arsitektur host (misalnya, `amd64`, `arm64`)
- `wsl.version`: Nomor versi WSL (hanya ada saat berjalan di Windows Subsystem for Linux)
- Nama Meter: `com.anthropic.claude_code`

Sumber daya pengukuran ROI

Untuk panduan komprehensif tentang mengukur pengembalian investasi untuk Claude Code, termasuk pengaturan telemetri, analisis biaya, metrik produktivitas, dan pelaporan otomatis, lihat Panduan Pengukuran ROI Claude Code. Repositori ini menyediakan konfigurasi Docker Compose siap pakai, pengaturan Prometheus dan OpenTelemetry, dan template untuk menghasilkan laporan produktivitas yang terintegrasi dengan alat seperti Linear.

Pertimbangan keamanan/privasi

- Telemetri adalah opt-in dan memerlukan konfigurasi eksplisit
- Informasi sensitif seperti kunci API atau konten file tidak pernah disertakan dalam metrik atau acara
- Konten prompt pengguna diredaksi secara default - hanya panjang prompt yang dicatat. Untuk mengaktifkan pencatatan prompt pengguna, atur `OTEL_LOG_USER_PROMPTS=1`

Memantau Claude Code di Amazon Bedrock

Untuk panduan pemantauan penggunaan Claude Code terperinci untuk Amazon Bedrock, lihat Implementasi Pemantauan Claude Code (Bedrock).

Ringkasan penyebaran enterprise

> *Pelajari bagaimana Claude Code dapat terintegrasi dengan berbagai layanan pihak ketiga dan infrastruktur untuk memenuhi persyaratan penyebaran enterprise.*

Halaman ini memberikan gambaran umum tentang opsi penyebaran yang tersedia dan membantu Anda memilih konfigurasi yang tepat untuk organisasi Anda.

Perbandingan penyedia

Fitur Anthropic Amazon Bedrock Google Vertex AI Microsoft Foundry

Wilayah Negara yang didukung countries Beberapa AWS regions Beberapa GCP regions Beberapa Azure regions

Prompt caching Diaktifkan secara default Diaktifkan secara default Diaktifkan secara default Diaktifkan secara default

Autentikasi Kunci API Kunci API atau kredensial AWS Kredensial GCP Kunci API atau Microsoft Entra ID

Pelacakan biaya Dashboard AWS Cost Explorer GCP Billing Azure Cost Management

Fitur enterprise Tim, pemantauan penggunaan Kebijakan IAM, CloudTrail Peran IAM, Cloud Audit Logs Kebijakan RBAC, Azure Monitor

Penyedia cloud

- Amazon Bedrock Gunakan model Claude melalui infrastruktur AWS dengan autentikasi berbasis kunci API atau IAM dan pemantauan asli AWS
- Google Vertex AI Akses model Claude melalui Google Cloud Platform dengan keamanan dan kepatuhan tingkat enterprise
- Microsoft Foundry Akses Claude melalui Azure dengan autentikasi kunci API atau Microsoft Entra ID dan penagihan Azure

Infrastruktur korporat

- Enterprise Network Konfigurasikan Claude Code untuk bekerja dengan server proxy organisasi Anda dan persyaratan SSL/TLS

- LLM Gateway Sebarkan akses model terpusat dengan pelacakan penggunaan, penganggaran, dan pencatatan audit

Ringkasan konfigurasi

Claude Code mendukung opsi konfigurasi fleksibel yang memungkinkan Anda menggabungkan penyedia dan infrastruktur yang berbeda:

> **Note:**
>
> Pahami perbedaan antara:
>
> - **Proxy korporat**: Proxy HTTP/HTTPS untuk merutekan lalu lintas (diatur melalui `HTTPS_PROXY` atau `HTTP_PROXY`)
> - **LLM Gateway**: Layanan yang menangani autentikasi dan menyediakan endpoint yang kompatibel dengan penyedia (diatur melalui `ANTHROPIC_BASE_URL`, `ANTHROPIC_BEDROCK_BASE_URL`, atau `ANTHROPIC_VERTEX_BASE_URL`)
>
> Kedua konfigurasi dapat digunakan secara bersamaan.

Menggunakan Bedrock dengan proxy korporat

Rutekan lalu lintas Bedrock melalui proxy HTTP/HTTPS korporat:

```
## Aktifkan Bedrock
export CLAUDE_CODE_USE_BEDROCK=1
export AWS_REGION=us-east-1

## Konfigurasikan proxy korporat
export HTTPS_PROXY='https://proxy.example.com:8080'
```

Menggunakan Bedrock dengan LLM Gateway

Gunakan layanan gateway yang menyediakan endpoint yang kompatibel dengan Bedrock:

```
## Aktifkan Bedrock
export CLAUDE_CODE_USE_BEDROCK=1

## Konfigurasikan gateway LLM
export ANTHROPIC_BEDROCK_BASE_URL='https://your-llm-gateway.com/bedrock'
export CLAUDE_CODE_SKIP_BEDROCK_AUTH=1  # Jika gateway menangani autentikasi AWS
```

Menggunakan Foundry dengan proxy korporat

Rutekan lalu lintas Azure melalui proxy HTTP/HTTPS korporat:

```
## Aktifkan Microsoft Foundry
export CLAUDE_CODE_USE_FOUNDRY=1
export ANTHROPIC_FOUNDRY_RESOURCE=your-resource
export ANTHROPIC_FOUNDRY_API_KEY=your-api-key  # Atau abaikan untuk autentikasi
Entra ID

## Konfigurasikan proxy korporat
export HTTPS_PROXY='https://proxy.example.com:8080'
```

Menggunakan Foundry dengan LLM Gateway

Gunakan layanan gateway yang menyediakan endpoint yang kompatibel dengan Azure:

```
## Aktifkan Microsoft Foundry
export CLAUDE_CODE_USE_FOUNDRY=1

## Konfigurasikan gateway LLM
export ANTHROPIC_FOUNDRY_BASE_URL='https://your-llm-gateway.com'
export CLAUDE_CODE_SKIP_FOUNDRY_AUTH=1  # Jika gateway menangani autentikasi Azure
```

Menggunakan Vertex AI dengan proxy korporat

Rutekan lalu lintas Vertex AI melalui proxy HTTP/HTTPS korporat:

```
## Aktifkan Vertex
export CLAUDE_CODE_USE_VERTEX=1
export CLOUD_ML_REGION=us-east5
export ANTHROPIC_VERTEX_PROJECT_ID=your-project-id

## Konfigurasikan proxy korporat
export HTTPS_PROXY='https://proxy.example.com:8080'
```

Menggunakan Vertex AI dengan LLM Gateway

Gabungkan model Google Vertex AI dengan gateway LLM untuk manajemen terpusat:

```
## Aktifkan Vertex
export CLAUDE_CODE_USE_VERTEX=1

## Konfigurasikan gateway LLM
export ANTHROPIC_VERTEX_BASE_URL='https://your-llm-gateway.com/vertex'
export CLAUDE_CODE_SKIP_VERTEX_AUTH=1  # Jika gateway menangani autentikasi GCP
```

Konfigurasi autentikasi

Claude Code menggunakan `ANTHROPIC_AUTH_TOKEN` untuk header `Authorization` ketika diperlukan. Bendera `SKIP_AUTH` (`CLAUDE_CODE_SKIP_BEDROCK_AUTH`, `CLAUDE_CODE_SKIP_VERTEX_AUTH`) digunakan dalam skenario gateway LLM di mana gateway menangani autentikasi penyedia.

Memilih konfigurasi penyebaran yang tepat

Pertimbangkan faktor-faktor ini saat memilih pendekatan penyebaran Anda:

Akses penyedia langsung

Terbaik untuk organisasi yang:

- Menginginkan pengaturan paling sederhana
- Memiliki infrastruktur AWS atau GCP yang sudah ada
- Membutuhkan pemantauan asli penyedia dan kepatuhan

Proxy korporat

Terbaik untuk organisasi yang:

- Memiliki persyaratan proxy korporat yang sudah ada
- Membutuhkan pemantauan lalu lintas dan kepatuhan
- Harus merutekan semua lalu lintas melalui jalur jaringan tertentu

LLM Gateway

Terbaik untuk organisasi yang:

- Membutuhkan pelacakan penggunaan di seluruh tim
- Ingin beralih secara dinamis antar model
- Memerlukan pembatasan kecepatan khusus atau anggaran
- Membutuhkan manajemen autentikasi terpusat

Debugging

Saat men-debug penyebaran Anda:

- Gunakan perintah slash `claude /status`. Perintah ini memberikan visibilitas ke dalam autentikasi, proxy, dan pengaturan URL apa pun yang diterapkan.
- Atur variabel lingkungan `export ANTHROPIC_LOG=debug` untuk mencatat permintaan.

Praktik terbaik untuk organisasi

1. Investasi dalam dokumentasi dan memori

Kami sangat merekomendasikan investasi dalam dokumentasi sehingga Claude Code memahami basis kode Anda. Organisasi dapat menyebarkan file CLAUDE.md di berbagai tingkat:

- **Di seluruh organisasi**: Sebarkan ke direktori sistem seperti `/Library/Application Support/ClaudeCode/CLAUDE.md` (macOS) untuk standar perusahaan
- **Tingkat repositori**: Buat file `CLAUDE.md` di akar repositori yang berisi arsitektur proyek, perintah build, dan pedoman kontribusi. Periksa ini ke dalam kontrol sumber sehingga semua pengguna mendapat manfaat

  Pelajari lebih lanjut.

2. Sederhanakan penyebaran

Jika Anda memiliki lingkungan pengembangan khusus, kami menemukan bahwa membuat cara “satu klik” untuk menginstal Claude Code adalah kunci untuk meningkatkan adopsi di seluruh organisasi.

3. Mulai dengan penggunaan terpandu

Dorong pengguna baru untuk mencoba Claude Code untuk Q&A basis kode, atau pada perbaikan bug yang lebih kecil atau permintaan fitur. Minta Claude Code untuk membuat rencana. Periksa saran Claude dan berikan umpan balik jika tidak sesuai jalur. Seiring waktu, ketika pengguna memahami paradigma baru ini dengan lebih baik, mereka akan lebih efektif dalam membiarkan Claude Code berjalan lebih agentic.

4. Konfigurasikan kebijakan keamanan

Tim keamanan dapat mengonfigurasi izin terkelola untuk apa yang Claude Code diizinkan dan tidak diizinkan untuk dilakukan, yang tidak dapat ditimpa oleh konfigurasi lokal. Pelajari lebih lanjut.

5. Manfaatkan MCP untuk integrasi

MCP adalah cara yang bagus untuk memberikan Claude Code lebih banyak informasi, seperti menghubungkan ke sistem manajemen tiket atau log kesalahan. Kami merekomendasikan bahwa satu tim pusat mengonfigurasi server MCP dan memeriksa konfigurasi `.mcp.json` ke dalam basis kode sehingga semua pengguna mendapat manfaat. Pelajari lebih lanjut.

Di Anthropic, kami mempercayai Claude Code untuk mendorong pengembangan di seluruh setiap basis kode Anthropic. Kami harap Anda menikmati menggunakan Claude Code sebanyak yang kami lakukan.

Langkah berikutnya

- Siapkan Amazon Bedrock untuk penyebaran asli AWS
- Konfigurasikan Google Vertex AI untuk penyebaran GCP
- Siapkan Microsoft Foundry untuk penyebaran Azure
- Konfigurasikan Enterprise Network untuk persyaratan jaringan
- Sebarkan LLM Gateway untuk manajemen enterprise
- Pengaturan untuk opsi konfigurasi dan variabel lingkungan

Konfigurasi pengaturan yang dikelola server (beta publik)

> *Konfigurasi Claude Code secara terpusat untuk organisasi Anda melalui pengaturan yang dikirimkan server, tanpa memerlukan infrastruktur manajemen perangkat.*

Pengaturan yang dikelola server memungkinkan administrator untuk mengonfigurasi Claude Code secara terpusat melalui antarmuka berbasis web di Claude.ai. Klien Claude Code secara otomatis menerima pengaturan ini ketika pengguna melakukan autentikasi dengan kredensial organisasi mereka.

Pendekatan ini dirancang untuk organisasi yang tidak memiliki infrastruktur manajemen perangkat, atau perlu mengelola pengaturan untuk pengguna pada perangkat yang tidak dikelola.

Note:

Pengaturan yang dikelola server berada dalam beta publik dan tersedia untuk pelanggan Claude for Teams dan Claude for Enterprise. Fitur dapat berkembang sebelum ketersediaan umum.

Persyaratan

Untuk menggunakan pengaturan yang dikelola server, Anda memerlukan:

- Paket Claude for Teams atau Claude for Enterprise
- Claude Code versi 2.1.38 atau lebih baru untuk Claude for Teams, atau versi 2.1.30 atau lebih baru untuk Claude for Enterprise
- Akses jaringan ke `api.anthropic.com`

Pilih antara pengaturan yang dikelola server dan endpoint

Claude Code mendukung dua pendekatan untuk konfigurasi terpusat. Pengaturan yang dikelola server mengirimkan konfigurasi dari server Anthropic. Pengaturan yang dikelola endpoint digunakan langsung ke perangkat melalui kebijakan OS asli (preferensi terkelola macOS, registri Windows) atau file pengaturan terkelola.

Pendekatan	Terbaik untuk	Model keamanan
Pengaturan yang dikelola server	Organisasi tanpa MDM, atau pengguna pada perangkat yang tidak dikelola	Pengaturan dikirimkan dari server Anthropic pada waktu autentikasi
Pengaturan yang dikelola endpoint	Organisasi dengan MDM atau manajemen endpoint	Pengaturan digunakan ke perangkat melalui profil konfigurasi MDM, kebijakan registri, atau file pengaturan terkelola

Jika perangkat Anda terdaftar dalam solusi MDM atau manajemen endpoint, pengaturan yang dikelola endpoint memberikan jaminan keamanan yang lebih kuat karena file pengaturan dapat dilindungi dari modifikasi pengguna di tingkat OS.

Konfigurasi pengaturan yang dikelola server

Step 1: Buka konsol admin

Di Claude.ai, navigasikan ke **Admin Settings > Claude Code > Managed settings**.

Step 2: Tentukan pengaturan Anda

Tambahkan konfigurasi Anda sebagai JSON. Semua pengaturan yang tersedia di `settings.json` didukung, termasuk pengaturan yang hanya dikelola seperti `disableBypassPermissionsMode`.

Contoh ini memberlakukan daftar penolakan izin dan mencegah pengguna dari melewati izin:

```
{
  "permissions": {
    "deny": [
      "Bash(curl *)",
      "Read(./.env)",
      "Read(./.env.*)",
      "Read(./secrets/**)"
    ]
  },
  "disableBypassPermissionsMode": "disable"
}
```

Step 3: Simpan dan terapkan

Simpan perubahan Anda. Klien Claude Code menerima pengaturan yang diperbarui pada startup berikutnya atau siklus polling per jam.

Verifikasi pengiriman pengaturan

Untuk mengonfirmasi bahwa pengaturan sedang diterapkan, minta pengguna untuk memulai ulang Claude Code. Jika konfigurasi mencakup pengaturan yang memicu dialog persetujuan keamanan, pengguna melihat prompt yang menjelaskan pengaturan yang dikelola pada startup. Anda juga dapat memverifikasi bahwa aturan izin terkelola aktif dengan meminta pengguna menjalankan `/permissions` untuk melihat aturan izin efektif mereka.

Kontrol akses

Peran berikut dapat mengelola pengaturan yang dikelola server:

- **Primary Owner**
- **Owner**

Batasi akses ke personel terpercaya, karena perubahan pengaturan berlaku untuk semua pengguna dalam organisasi.

Batasan saat ini

Pengaturan yang dikelola server memiliki batasan berikut selama periode beta:

- Pengaturan berlaku secara seragam untuk semua pengguna dalam organisasi. Konfigurasi per-grup belum didukung.
- Konfigurasi server MCP tidak dapat didistribusikan melalui pengaturan yang dikelola server.

Pengiriman pengaturan

Prioritas pengaturan

Pengaturan yang dikelola server dan pengaturan yang dikelola endpoint keduanya menempati tingkat tertinggi dalam hierarki pengaturan Claude Code. Tidak ada tingkat pengaturan lain yang dapat menimpanya, termasuk argumen baris perintah. Ketika keduanya ada, pengaturan yang dikelola server memiliki prioritas dan pengaturan yang dikelola endpoint tidak digunakan.

Perilaku pengambilan dan caching

Claude Code mengambil pengaturan dari server Anthropic pada startup dan melakukan polling untuk pembaruan setiap jam selama sesi aktif.

Peluncuran pertama tanpa pengaturan yang di-cache:

- Claude Code mengambil pengaturan secara asinkron
- Jika pengambilan gagal, Claude Code melanjutkan tanpa pengaturan terkelola
- Ada jendela singkat sebelum pengaturan dimuat di mana pembatasan belum diterapkan

Peluncuran berikutnya dengan pengaturan yang di-cache:

- Pengaturan yang di-cache berlaku segera pada startup
- Claude Code mengambil pengaturan segar di latar belakang
- Pengaturan yang di-cache bertahan melalui kegagalan jaringan

Claude Code menerapkan pembaruan pengaturan secara otomatis tanpa restart, kecuali untuk pengaturan lanjutan seperti konfigurasi OpenTelemetry, yang memerlukan restart penuh agar efektif.

Dialog persetujuan keamanan

Pengaturan tertentu yang dapat menimbulkan risiko keamanan memerlukan persetujuan pengguna eksplisit sebelum diterapkan:

- **Pengaturan perintah shell**: pengaturan yang menjalankan perintah shell
- **Variabel lingkungan kustom**: variabel yang tidak ada dalam daftar aman yang diketahui
- **Konfigurasi hook**: definisi hook apa pun

Ketika pengaturan ini ada, pengguna melihat dialog keamanan yang menjelaskan apa yang sedang dikonfigurasi. Pengguna harus menyetujui untuk melanjutkan. Jika pengguna menolak pengaturan, Claude Code keluar.

> **Note:**
>
> Dalam mode non-interaktif dengan flag `-p`, Claude Code melewati dialog keamanan dan menerapkan pengaturan tanpa persetujuan pengguna.

Ketersediaan platform

Pengaturan yang dikelola server memerlukan koneksi langsung ke `api.anthropic.com` dan tidak tersedia saat menggunakan penyedia model pihak ketiga:

- Amazon Bedrock
- Google Vertex AI

- Microsoft Foundry
- Endpoint API kustom melalui `ANTHROPIC_BASE_URL` atau gateway LLM

Audit logging

Acara log audit untuk perubahan pengaturan tersedia melalui API kepatuhan atau ekspor log audit. Hubungi tim akun Anthropic Anda untuk akses.

Acara audit mencakup jenis tindakan yang dilakukan, akun dan perangkat yang melakukan tindakan, dan referensi ke nilai sebelumnya dan baru.

Pertimbangan keamanan

Pengaturan yang dikelola server menyediakan penegakan kebijakan terpusat, tetapi mereka beroperasi sebagai kontrol sisi klien. Pada perangkat yang tidak dikelola, pengguna dengan akses admin atau sudo dapat memodifikasi biner Claude Code, sistem file, atau konfigurasi jaringan.

Skenario	Perilaku
Pengguna mengedit file pengaturan yang di-cache	File yang dirusak berlaku pada startup, tetapi pengaturan yang benar dipulihkan pada pengambilan server berikutnya
Pengguna menghapus file pengaturan yang di-cache	Perilaku peluncuran pertama terjadi: pengaturan mengambil secara asinkron dengan jendela unenforced singkat
API tidak tersedia	Pengaturan yang di-cache berlaku jika tersedia, jika tidak pengaturan terkelola tidak diterapkan sampai pengambilan yang berhasil berikutnya
Pengguna melakukan autentikasi dengan organisasi yang berbeda	Pengaturan tidak dikirimkan untuk akun di luar organisasi yang dikelola
Pengguna menetapkan `ANTHROPIC_BASE_URL` non-default	Pengaturan yang dikelola server dilewati saat menggunakan penyedia API pihak ketiga

Untuk mendeteksi perubahan konfigurasi runtime, gunakan hook `ConfigChange` untuk mencatat modifikasi atau memblokir perubahan yang tidak sah sebelum diterapkan.

Untuk jaminan penegakan yang lebih kuat, gunakan pengaturan yang dikelola endpoint pada perangkat yang terdaftar dalam solusi MDM.

Lihat juga

Halaman terkait untuk mengelola konfigurasi Claude Code:

- Settings: referensi konfigurasi lengkap termasuk semua pengaturan yang tersedia
- Pengaturan yang dikelola endpoint: pengaturan terkelola yang digunakan ke perangkat oleh IT
- Authentication: atur akses pengguna ke Claude Code
- Security: perlindungan keamanan dan praktik terbaik

Part 11: Cloud Providers

Claude Code di Amazon Bedrock

> *Pelajari tentang mengonfigurasi Claude Code melalui Amazon Bedrock, termasuk pengaturan, konfigurasi IAM, dan pemecahan masalah.*

Prasyarat

Sebelum mengonfigurasi Claude Code dengan Bedrock, pastikan Anda memiliki:

- Akun AWS dengan akses Bedrock yang diaktifkan
- Akses ke model Claude yang diinginkan (misalnya, Claude Sonnet 4.6) di Bedrock
- AWS CLI terinstal dan dikonfigurasi (opsional - hanya diperlukan jika Anda tidak memiliki mekanisme lain untuk mendapatkan kredensial)
- Izin IAM yang sesuai

> **Note:**
>
> Jika Anda menerapkan Claude Code ke beberapa pengguna, pin versi model Anda untuk mencegah kerusakan ketika Anthropic merilis model baru.

Pengaturan

1. Kirimkan detail kasus penggunaan

Pengguna pertama kali dari model Anthropic harus mengirimkan detail kasus penggunaan sebelum memanggil model. Ini dilakukan sekali per akun.

1. Pastikan Anda memiliki izin IAM yang tepat (lihat lebih lanjut di bawah)
2. Navigasikan ke konsol Amazon Bedrock
3. Pilih **Chat/Text playground**
4. Pilih model Anthropic apa pun dan Anda akan diminta untuk mengisi formulir kasus penggunaan

2. Konfigurasi kredensial AWS

Claude Code menggunakan rantai kredensial SDK AWS default. Atur kredensial Anda menggunakan salah satu metode berikut:

Opsi A: Konfigurasi AWS CLI

```
aws configure
```

Opsi B: Variabel lingkungan (kunci akses)

```
export AWS_ACCESS_KEY_ID=your-access-key-id
export AWS_SECRET_ACCESS_KEY=your-secret-access-key
export AWS_SESSION_TOKEN=your-session-token
```

Opsi C: Variabel lingkungan (profil SSO)

```
aws sso login --profile=<your-profile-name>

export AWS_PROFILE=your-profile-name
```

Opsi D: Kredensial AWS Management Console

```
aws login
```

Pelajari lebih lanjut tentang `aws login`.

Opsi E: Kunci API Bedrock

```
export AWS_BEARER_TOKEN_BEDROCK=your-bedrock-api-key
```

Kunci API Bedrock menyediakan metode autentikasi yang lebih sederhana tanpa memerlukan kredensial AWS lengkap. Pelajari lebih lanjut tentang kunci API Bedrock.

Konfigurasi kredensial lanjutan

Claude Code mendukung penyegaran kredensial otomatis untuk AWS SSO dan penyedia identitas perusahaan. Tambahkan pengaturan ini ke file pengaturan Claude Code Anda (lihat Settings untuk lokasi file).

Ketika Claude Code mendeteksi bahwa kredensial AWS Anda telah kedaluwarsa (baik secara lokal berdasarkan stempel waktu mereka atau ketika Bedrock mengembalikan kesalahan kredensial), Claude Code akan secara otomatis menjalankan perintah `awsAuthRefresh` dan/atau `awsCredentialExport` yang dikonfigurasi untuk mendapatkan kredensial baru sebelum mencoba ulang permintaan.

Contoh konfigurasi

```
{
  "awsAuthRefresh": "aws sso login --profile myprofile",
  "env": {
    "AWS_PROFILE": "myprofile"
  }
}
```

Pengaturan konfigurasi dijelaskan

`awsAuthRefresh`: Gunakan ini untuk perintah yang memodifikasi direktori `.aws`, seperti memperbarui kredensial, cache SSO, atau file konfigurasi. Output perintah ditampilkan kepada pengguna, tetapi input interaktif tidak didukung. Ini bekerja dengan baik untuk alur SSO berbasis browser di mana CLI menampilkan URL atau kode dan Anda menyelesaikan autentikasi di browser.

`awsCredentialExport`: Hanya gunakan ini jika Anda tidak dapat memodifikasi `.aws` dan harus secara langsung mengembalikan kredensial. Output ditangkap secara diam-diam dan tidak ditampilkan kepada pengguna. Perintah harus menampilkan JSON dalam format ini:

```
{
  "Credentials": {
    "AccessKeyId": "value",
    "SecretAccessKey": "value",
    "SessionToken": "value"
  }
}
```

3. Konfigurasi Claude Code

Atur variabel lingkungan berikut untuk mengaktifkan Bedrock:

```
## Aktifkan integrasi Bedrock
export CLAUDE_CODE_USE_BEDROCK=1
export AWS_REGION=us-east-1  # atau wilayah pilihan Anda

## Opsional: Ganti wilayah untuk model kecil/cepat (Haiku)
export ANTHROPIC_SMALL_FAST_MODEL_AWS_REGION=us-west-2
```

Saat mengaktifkan Bedrock untuk Claude Code, perhatikan hal berikut:

- `AWS_REGION` adalah variabel lingkungan yang diperlukan. Claude Code tidak membaca dari file konfigurasi `.aws` untuk pengaturan ini.
- Saat menggunakan Bedrock, perintah `/login` dan `/logout` dinonaktifkan karena autentikasi ditangani melalui kredensial AWS.
- Anda dapat menggunakan file pengaturan untuk variabel lingkungan seperti `AWS_PROFILE` yang tidak ingin Anda bocorkan ke proses lain. Lihat Settings untuk informasi lebih lanjut.

4. Pin versi model

> **Warning:**
>
> Pin versi model spesifik untuk setiap penerapan. Jika Anda menggunakan alias model (`sonnet`, `opus`, `haiku`) tanpa pinning, Claude Code mungkin mencoba menggunakan versi model yang lebih baru yang tidak tersedia di akun Bedrock Anda, merusak pengguna yang ada ketika Anthropic merilis pembaruan.

Atur variabel lingkungan ini ke ID model Bedrock spesifik:

```
export ANTHROPIC_DEFAULT_OPUS_MODEL='us.anthropic.claude-opus-4-6-v1'
export ANTHROPIC_DEFAULT_SONNET_MODEL='us.anthropic.claude-sonnet-4-6'
export ANTHROPIC_DEFAULT_HAIKU_MODEL='us.anthropic.claude-haiku-4-5-20251001-v1:0'
```

Variabel ini menggunakan ID profil inferensi lintas wilayah (dengan awalan `us.`). Jika Anda menggunakan awalan wilayah berbeda atau profil inferensi aplikasi, sesuaikan sesuai kebutuhan. Untuk ID model saat ini dan warisan, lihat Models overview. Lihat Model configuration untuk daftar lengkap variabel lingkungan.

Claude Code menggunakan model default ini ketika tidak ada variabel pinning yang diatur:

Jenis model	Nilai default
Model utama	`global.anthropic.claude-sonnet-4-6`
Model kecil/cepat	`us.anthropic.claude-haiku-4-5-20251001-v1:0`

Untuk menyesuaikan model lebih lanjut, gunakan salah satu metode berikut:

```
## Menggunakan ID profil inferensi
export ANTHROPIC_MODEL='global.anthropic.claude-sonnet-4-6'
export ANTHROPIC_SMALL_FAST_MODEL='us.anthropic.claude-haiku-4-5-20251001-v1:0'

## Menggunakan ARN profil inferensi aplikasi
export ANTHROPIC_MODEL='arn:aws:bedrock:us-east-2:your-account-id:application-
inference-profile/your-model-id'

## Opsional: Nonaktifkan prompt caching jika diperlukan
export DISABLE_PROMPT_CACHING=1
```

> **Note:**
>
> Prompt caching mungkin tidak tersedia di semua wilayah.

Petakan setiap versi model ke profil inferensi

Variabel lingkungan `ANTHROPIC_DEFAULT_*_MODEL` mengonfigurasi satu profil inferensi per keluarga model. Jika organisasi Anda perlu mengekspos beberapa versi dari keluarga yang sama di pemilih `/model`, masing-masing dirutekan ke ARN profil inferensi aplikasi sendiri, gunakan pengaturan `modelOverrides` di file pengaturan Anda sebagai gantinya.

Contoh ini memetakan tiga versi Opus ke ARN yang berbeda sehingga pengguna dapat beralih di antara mereka tanpa melewati profil inferensi organisasi Anda:

```
{
  "modelOverrides": {
    "claude-opus-4-6": "arn:aws:bedrock:us-east-2:123456789012:application-
inference-profile/opus-46-prod",
    "claude-opus-4-5-20251101": "arn:aws:bedrock:us-
east-2:123456789012:application-inference-profile/opus-45-prod",
    "claude-opus-4-1-20250805": "arn:aws:bedrock:us-
east-2:123456789012:application-inference-profile/opus-41-prod"
  }
}
```

Ketika pengguna memilih salah satu versi ini di `/model`, Claude Code memanggil Bedrock dengan ARN yang dipetakan. Versi tanpa override kembali ke ID model Bedrock bawaan atau profil inferensi yang cocok yang ditemukan saat startup. Lihat Override model IDs per version untuk detail tentang bagaimana override berinteraksi dengan `availableModels` dan pengaturan model lainnya.

Konfigurasi IAM

Buat kebijakan IAM dengan izin yang diperlukan untuk Claude Code:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowModelAndInferenceProfileAccess",
      "Effect": "Allow",
      "Action": [
        "bedrock:InvokeModel",
        "bedrock:InvokeModelWithResponseStream",
        "bedrock:ListInferenceProfiles"
      ],
      "Resource": [
        "arn:aws:bedrock:*:*:inference-profile/*",
        "arn:aws:bedrock:*:*:application-inference-profile/*",
        "arn:aws:bedrock:*:*:foundation-model/*"
      ]
    },
    {
      "Sid": "AllowMarketplaceSubscription",
      "Effect": "Allow",
      "Action": [
        "aws-marketplace:ViewSubscriptions",
        "aws-marketplace:Subscribe"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:CalledViaLast": "bedrock.amazonaws.com"
        }
      }
    }
  ]
}
```

Untuk izin yang lebih ketat, Anda dapat membatasi Resource ke ARN profil inferensi spesifik.

Untuk detail, lihat dokumentasi IAM Bedrock.

> **Note:**
>
> Buat akun AWS khusus untuk Claude Code untuk menyederhanakan pelacakan biaya dan kontrol akses.

AWS Guardrails

Amazon Bedrock Guardrails memungkinkan Anda menerapkan penyaringan konten untuk Claude Code. Buat Guardrail di konsol Amazon Bedrock, publikasikan versi, kemudian tambahkan header Guardrail ke file pengaturan Anda. Aktifkan inferensi Cross-Region pada Guardrail Anda jika Anda menggunakan profil inferensi lintas wilayah.

Contoh konfigurasi:

```
{
  "env": {
    "ANTHROPIC_CUSTOM_HEADERS": "X-Amzn-Bedrock-GuardrailIdentifier: your-guardrail-id\nX-Amzn-Bedrock-GuardrailVersion: 1"
  }
}
```

Pemecahan Masalah

Jika Anda mengalami masalah wilayah:

- Periksa ketersediaan model: `aws bedrock list-inference-profiles --region your-region`
- Beralih ke wilayah yang didukung: `export AWS_REGION=us-east-1`
- Pertimbangkan menggunakan profil inferensi untuk akses lintas wilayah

Jika Anda menerima kesalahan “on-demand throughput isn’t supported”:

- Tentukan model sebagai ID profil inferensi

Claude Code menggunakan Invoke API Bedrock dan tidak mendukung Converse API.

Sumber daya tambahan

- Dokumentasi Bedrock
- Harga Bedrock
- Profil inferensi Bedrock

- Claude Code di Amazon Bedrock: Panduan Pengaturan Cepat
- Implementasi Pemantauan Claude Code (Bedrock)

Claude Code pada Google Vertex AI

> *Pelajari tentang mengonfigurasi Claude Code melalui Google Vertex AI, termasuk pengaturan, konfigurasi IAM, dan pemecahan masalah.*

Prasyarat

Sebelum mengonfigurasi Claude Code dengan Vertex AI, pastikan Anda memiliki:

- Akun Google Cloud Platform (GCP) dengan penagihan diaktifkan
- Proyek GCP dengan Vertex AI API diaktifkan
- Akses ke model Claude yang diinginkan (misalnya, Claude Sonnet 4.5)
- Google Cloud SDK (`gcloud`) terinstal dan dikonfigurasi
- Kuota dialokasikan di wilayah GCP yang diinginkan

Konfigurasi Wilayah

Claude Code dapat digunakan dengan kedua titik akhir Vertex AI global dan regional.

> **Note:**
>
> Vertex AI mungkin tidak mendukung model default Claude Code di semua wilayah. Anda mungkin perlu beralih ke wilayah atau model yang didukung.

> **Note:**
>
> Vertex AI mungkin tidak mendukung model default Claude Code pada titik akhir global. Anda mungkin perlu beralih ke titik akhir regional atau model yang didukung.

Pengaturan

1. Aktifkan Vertex AI API

Aktifkan Vertex AI API di proyek GCP Anda:

```
## Atur ID proyek Anda
gcloud config set project YOUR-PROJECT-ID

## Aktifkan Vertex AI API
gcloud services enable aiplatform.googleapis.com
```

2. Minta akses model

Minta akses ke model Claude di Vertex AI:

1. Navigasikan ke Vertex AI Model Garden
2. Cari model "Claude"
3. Minta akses ke model Claude yang diinginkan (misalnya, Claude Sonnet 4.5)
4. Tunggu persetujuan (mungkin memakan waktu 24-48 jam)

3. Konfigurasi kredensial GCP

Claude Code menggunakan autentikasi Google Cloud standar.

Untuk informasi lebih lanjut, lihat dokumentasi autentikasi Google Cloud.

> **Note:**
>
> Saat melakukan autentikasi, Claude Code akan secara otomatis menggunakan ID proyek dari variabel lingkungan `ANTHROPIC_VERTEX_PROJECT_ID`. Untuk menimpanya, atur salah satu variabel lingkungan ini: `GCLOUD_PROJECT`, `GOOGLE_CLOUD_PROJECT`, atau `GOOGLE_APPLICATION_CREDENTIALS`.

4. Konfigurasi Claude Code

Atur variabel lingkungan berikut:

```
## Aktifkan integrasi Vertex AI
export CLAUDE_CODE_USE_VERTEX=1
export CLOUD_ML_REGION=global
export ANTHROPIC_VERTEX_PROJECT_ID=YOUR-PROJECT-ID

## Opsional: Nonaktifkan prompt caching jika diperlukan
export DISABLE_PROMPT_CACHING=1

## Ketika CLOUD_ML_REGION=global, timpa wilayah untuk model yang tidak didukung
export VERTEX_REGION_CLAUDE_3_5_HAIKU=us-east5

## Opsional: Timpa wilayah untuk model spesifik lainnya
export VERTEX_REGION_CLAUDE_3_5_SONNET=us-east5
export VERTEX_REGION_CLAUDE_3_7_SONNET=us-east5
export VERTEX_REGION_CLAUDE_4_0_OPUS=europe-west1
export VERTEX_REGION_CLAUDE_4_0_SONNET=us-east5
export VERTEX_REGION_CLAUDE_4_1_OPUS=europe-west1
```

Note:

Prompt caching didukung secara otomatis ketika Anda menentukan flag ephemeral `cache_control`. Untuk menonaktifkannya, atur `DISABLE_PROMPT_CACHING=1`. Untuk batas laju yang lebih tinggi, hubungi dukungan Google Cloud.

Note:

Saat menggunakan Vertex AI, perintah `/login` dan `/logout` dinonaktifkan karena autentikasi ditangani melalui kredensial Google Cloud.

5. Konfigurasi model

Claude Code menggunakan model default ini untuk Vertex AI:

Jenis model	Nilai default
Model utama	`claude-sonnet-4-5@20250929`
Model kecil/cepat	`claude-haiku-4-5@20251001`

> **Note:**
>
> Untuk pengguna Vertex AI, Claude Code tidak akan secara otomatis meningkatkan dari Haiku 3.5 ke Haiku 4.5. Untuk beralih secara manual ke model Haiku yang lebih baru, atur variabel lingkungan `ANTHROPIC_DEFAULT_HAIKU_MODEL` ke nama model lengkap (misalnya, `claude-haiku-4-5@20251001`).

Untuk menyesuaikan model:

```
export ANTHROPIC_MODEL='claude-opus-4-1@20250805'
export ANTHROPIC_SMALL_FAST_MODEL='claude-haiku-4-5@20251001'
```

Konfigurasi IAM

Tetapkan izin IAM yang diperlukan:

Peran `roles/aiplatform.user` mencakup izin yang diperlukan:

- `aiplatform.endpoints.predict` - Diperlukan untuk invokasi model dan penghitungan token

Untuk izin yang lebih ketat, buat peran kustom dengan hanya izin di atas.

Untuk detail, lihat dokumentasi Vertex IAM.

> **Note:**
>
> Kami merekomendasikan membuat proyek GCP khusus untuk Claude Code untuk menyederhanakan pelacakan biaya dan kontrol akses.

context window 1M token

Claude Sonnet 4 dan Sonnet 4.5 mendukung context window 1M token pada Vertex AI.

> **Note:**
>
> context window 1M token saat ini dalam beta. Untuk menggunakan context window yang diperluas, sertakan header beta `context-1m-2025-08-07` dalam permintaan Vertex AI Anda.

Pemecahan Masalah

Jika Anda mengalami masalah kuota:

- Periksa kuota saat ini atau minta peningkatan kuota melalui Cloud Console

Jika Anda mengalami kesalahan "model not found" 404:

- Konfirmasi model diaktifkan di Model Garden
- Verifikasi Anda memiliki akses ke wilayah yang ditentukan
- Jika menggunakan `CLOUD_ML_REGION=global`, periksa bahwa model Anda mendukung titik akhir global di Model Garden di bawah "Supported features". Untuk model yang tidak mendukung titik akhir global, baik:
  - Tentukan model yang didukung melalui `ANTHROPIC_MODEL` atau `ANTHROPIC_SMALL_FAST_MODEL`, atau
  - Atur titik akhir regional menggunakan variabel lingkungan `VERTEX_REGION_<MODEL_NAME>`

Jika Anda mengalami kesalahan 429:

- Untuk titik akhir regional, pastikan model utama dan model kecil/cepat didukung di wilayah pilihan Anda
- Pertimbangkan untuk beralih ke `CLOUD_ML_REGION=global` untuk ketersediaan yang lebih baik

Sumber daya tambahan

- Dokumentasi Vertex AI
- Harga Vertex AI
- Kuota dan batas Vertex AI

Claude Code di Microsoft Foundry

> *Pelajari tentang mengonfigurasi Claude Code melalui Microsoft Foundry, termasuk setup, konfigurasi, dan pemecahan masalah.*

Prasyarat

Sebelum mengonfigurasi Claude Code dengan Microsoft Foundry, pastikan Anda memiliki:

- Langganan Azure dengan akses ke Microsoft Foundry
- Izin RBAC untuk membuat sumber daya dan deployment Microsoft Foundry
- Azure CLI diinstal dan dikonfigurasi (opsional - hanya diperlukan jika Anda tidak memiliki mekanisme lain untuk mendapatkan kredensial)

Setup

1. Menyediakan sumber daya Microsoft Foundry

Pertama, buat sumber daya Claude di Azure:

1. Navigasikan ke portal Microsoft Foundry
2. Buat sumber daya baru, catat nama sumber daya Anda
3. Buat deployment untuk model Claude:
    - Claude Opus
    - Claude Sonnet
    - Claude Haiku

2. Konfigurasi kredensial Azure

Claude Code mendukung dua metode autentikasi untuk Microsoft Foundry. Pilih metode yang paling sesuai dengan persyaratan keamanan Anda.

Opsi A: Autentikasi kunci API

1. Navigasikan ke sumber daya Anda di portal Microsoft Foundry
2. Buka bagian **Endpoints and keys**
3. Salin **API Key**
4. Atur variabel lingkungan:

```
export ANTHROPIC_FOUNDRY_API_KEY=your-azure-api-key
```

Opsi B: Autentikasi Microsoft Entra ID

Ketika `ANTHROPIC_FOUNDRY_API_KEY` tidak diatur, Claude Code secara otomatis menggunakan Azure SDK rantai kredensial default. Ini mendukung berbagai metode untuk mengautentikasi beban kerja lokal dan jarak jauh.

Di lingkungan lokal, Anda biasanya dapat menggunakan Azure CLI:

```
az login
```

> **Note:**
>
> Saat menggunakan Microsoft Foundry, perintah `/login` dan `/logout` dinonaktifkan karena autentikasi ditangani melalui kredensial Azure.

3. Konfigurasi Claude Code

Atur variabel lingkungan berikut untuk mengaktifkan Microsoft Foundry. Perhatikan bahwa nama deployment Anda diatur sebagai pengenal model di Claude Code (mungkin opsional jika menggunakan nama deployment yang disarankan).

```
## Aktifkan integrasi Microsoft Foundry
export CLAUDE_CODE_USE_FOUNDRY=1

## Nama sumber daya Azure (ganti {resource} dengan nama sumber daya Anda)
export ANTHROPIC_FOUNDRY_RESOURCE={resource}
## Atau berikan URL dasar lengkap:
## export ANTHROPIC_FOUNDRY_BASE_URL=https://{resource}.services.ai.azure.com

## Atur model ke nama deployment sumber daya Anda
export ANTHROPIC_DEFAULT_SONNET_MODEL='claude-sonnet-4-5'
export ANTHROPIC_DEFAULT_HAIKU_MODEL='claude-haiku-4-5'
export ANTHROPIC_DEFAULT_OPUS_MODEL='claude-opus-4-1'
```

Untuk detail lebih lanjut tentang opsi konfigurasi model, lihat Konfigurasi model.

Konfigurasi Azure RBAC

Peran default `Azure AI User` dan `Cognitive Services User` mencakup semua izin yang diperlukan untuk memanggil model Claude.

Untuk izin yang lebih ketat, buat peran khusus dengan yang berikut:

```
{
  "permissions": [
    {
      "dataActions": [
        "Microsoft.CognitiveServices/accounts/providers/*"
      ]
    }
  ]
}
```

Untuk detail, lihat dokumentasi RBAC Microsoft Foundry.

Pemecahan Masalah

Jika Anda menerima kesalahan “Failed to get token from azureADTokenProvider: ChainedTokenCredential authentication failed”:

- Konfigurasi Entra ID di lingkungan, atau atur `ANTHROPIC_FOUNDRY_API_KEY`.

Sumber daya tambahan

- Dokumentasi Microsoft Foundry
- Model Microsoft Foundry
- Harga Microsoft Foundry

Konfigurasi LLM gateway

> *Pelajari cara mengonfigurasi Claude Code untuk bekerja dengan solusi LLM gateway. Mencakup persyaratan gateway, konfigurasi autentikasi, pemilihan model, dan pengaturan endpoint khusus penyedia.*

LLM gateway menyediakan lapisan proxy terpusat antara Claude Code dan penyedia model, sering kali menyediakan:

- **Autentikasi terpusat** - Titik tunggal untuk manajemen kunci API
- **Pelacakan penggunaan** - Pantau penggunaan di seluruh tim dan proyek
- **Kontrol biaya** - Terapkan anggaran dan batas laju
- **Pencatatan audit** - Lacak semua interaksi model untuk kepatuhan
- **Perutean model** - Beralih antar penyedia tanpa perubahan kode

Persyaratan gateway

Agar LLM gateway dapat bekerja dengan Claude Code, gateway harus memenuhi persyaratan berikut:

Format API

Gateway harus mengekspos ke klien setidaknya salah satu format API berikut:

1. **Anthropic Messages**: `/v1/messages`, `/v1/messages/count_tokens`
   - Harus meneruskan header permintaan: `anthropic-beta`, `anthropic-version`
2. **Bedrock InvokeModel**: `/invoke`, `/invoke-with-response-stream`
   - Harus mempertahankan bidang badan permintaan: `anthropic_beta`, `anthropic_version`
3. **Vertex rawPredict**: `:rawPredict`, `:streamRawPredict`, `/count-tokens:rawPredict`
   - Harus meneruskan header permintaan: `anthropic-beta`, `anthropic-version`

Kegagalan untuk meneruskan header atau mempertahankan bidang badan dapat mengakibatkan fungsionalitas berkurang atau ketidakmampuan menggunakan fitur Claude Code.

> **Note:**

> Claude Code menentukan fitur mana yang akan diaktifkan berdasarkan format API. Saat menggunakan format Anthropic Messages dengan Bedrock atau Vertex, Anda mungkin perlu mengatur variabel lingkungan `CLAUDE_CODE_DISABLE_EXPERIMENTAL_BETAS=1`.

Konfigurasi

Pemilihan model

Secara default, Claude Code akan menggunakan nama model standar untuk format API yang dipilih.

Jika Anda telah mengonfigurasi nama model khusus di gateway Anda, gunakan variabel lingkungan yang didokumentasikan dalam Konfigurasi Model untuk mencocokkan nama khusus Anda.

Konfigurasi LiteLLM

> **Note:**
>
> LiteLLM adalah layanan proxy pihak ketiga. Anthropic tidak mendukung, memelihara, atau mengaudit keamanan atau fungsionalitas LiteLLM. Panduan ini disediakan untuk tujuan informasi dan mungkin menjadi ketinggalan zaman. Gunakan atas kebijakan Anda sendiri.

Prasyarat

- Claude Code diperbarui ke versi terbaru
- LiteLLM Proxy Server diterapkan dan dapat diakses
- Akses ke model Claude melalui penyedia pilihan Anda

Pengaturan LiteLLM dasar

Konfigurasi Claude Code:

Metode autentikasi

Kunci API statis

Metode paling sederhana menggunakan kunci API tetap:

```
## Atur di lingkungan
export ANTHROPIC_AUTH_TOKEN=sk-litellm-static-key

## Atau di pengaturan Claude Code
{
  "env": {
    "ANTHROPIC_AUTH_TOKEN": "sk-litellm-static-key"
  }
}
```

Nilai ini akan dikirim sebagai header `Authorization`.

Kunci API dinamis dengan pembantu

Untuk kunci yang berputar atau autentikasi per pengguna:

1. Buat skrip pembantu kunci API:

```
#!/bin/bash
## ~/bin/get-litellm-key.sh

## Contoh: Ambil kunci dari vault
vault kv get -field=api_key secret/litellm/claude-code

## Contoh: Hasilkan token JWT
jwt encode \
  --secret="${JWT_SECRET}" \
  --exp="+1h" \
  '{"user":"'${USER}'","team":"engineering"}'
```

1. Konfigurasi pengaturan Claude Code untuk menggunakan pembantu:

```
{
  "apiKeyHelper": "~/bin/get-litellm-key.sh"
}
```

1. Atur interval penyegaran token:

```
## Segarkan setiap jam (3600000 ms)
export CLAUDE_CODE_API_KEY_HELPER_TTL_MS=3600000
```

Nilai ini akan dikirim sebagai header `Authorization` dan `X-Api-Key`. `apiKeyHelper` memiliki prioritas lebih rendah daripada `ANTHROPIC_AUTH_TOKEN` atau `ANTHROPIC_API_KEY`.

Endpoint terpadu (direkomendasikan)

Menggunakan endpoint format Anthropic LiteLLM:

```
export ANTHROPIC_BASE_URL=https://litellm-server:4000
```

Manfaat endpoint terpadu dibandingkan endpoint pass-through:

- Penyeimbangan beban
- Fallback
- Dukungan konsisten untuk pelacakan biaya dan pelacakan pengguna akhir

Endpoint pass-through khusus penyedia (alternatif)

Claude API melalui LiteLLM

Menggunakan endpoint pass-through:

```
export ANTHROPIC_BASE_URL=https://litellm-server:4000/anthropic
```

Amazon Bedrock melalui LiteLLM

Menggunakan endpoint pass-through:

```
export ANTHROPIC_BEDROCK_BASE_URL=https://litellm-server:4000/bedrock
export CLAUDE_CODE_SKIP_BEDROCK_AUTH=1
export CLAUDE_CODE_USE_BEDROCK=1
```

Google Vertex AI melalui LiteLLM

Menggunakan endpoint pass-through:

```
export ANTHROPIC_VERTEX_BASE_URL=https://litellm-server:4000/vertex_ai/v1
export ANTHROPIC_VERTEX_PROJECT_ID=your-gcp-project-id
export CLAUDE_CODE_SKIP_VERTEX_AUTH=1
export CLAUDE_CODE_USE_VERTEX=1
export CLOUD_ML_REGION=us-east5
```

Untuk informasi lebih terperinci, lihat dokumentasi LiteLLM.

Sumber daya tambahan

- Dokumentasi LiteLLM
- Pengaturan Claude Code
- Konfigurasi jaringan enterprise
- Ikhtisar integrasi pihak ketiga

Konfigurasi jaringan enterprise

> *Konfigurasikan Claude Code untuk lingkungan enterprise dengan server proxy, Certificate Authorities (CA) kustom, dan autentikasi mutual Transport Layer Security (mTLS).*

Claude Code mendukung berbagai konfigurasi jaringan dan keamanan enterprise melalui variabel lingkungan. Ini termasuk merutekan lalu lintas melalui server proxy perusahaan, mempercayai Certificate Authorities (CA) kustom, dan mengautentikasi dengan sertifikat mutual Transport Layer Security (mTLS) untuk keamanan yang ditingkatkan.

> **Note:**
>
> Semua variabel lingkungan yang ditampilkan di halaman ini juga dapat dikonfigurasi di `settings.json`.

Konfigurasi proxy

Variabel lingkungan

Claude Code menghormati variabel lingkungan proxy standar:

```
## HTTPS proxy (direkomendasikan)
export HTTPS_PROXY=https://proxy.example.com:8080

## HTTP proxy (jika HTTPS tidak tersedia)
export HTTP_PROXY=http://proxy.example.com:8080

## Lewati proxy untuk permintaan tertentu - format terpisah spasi
export NO_PROXY="localhost 192.168.1.1 example.com .example.com"
## Lewati proxy untuk permintaan tertentu - format terpisah koma
export NO_PROXY="localhost,192.168.1.1,example.com,.example.com"
## Lewati proxy untuk semua permintaan
export NO_PROXY="*"
```

> **Note:**

Claude Code tidak mendukung proxy SOCKS.

Autentikasi dasar

Jika proxy Anda memerlukan autentikasi dasar, sertakan kredensial dalam URL proxy:

```
export HTTPS_PROXY=http://username:password@proxy.example.com:8080
```

Warning:

Hindari hardcoding kata sandi dalam skrip. Gunakan variabel lingkungan atau penyimpanan kredensial aman sebagai gantinya.

Tip:

Untuk proxy yang memerlukan autentikasi lanjutan (NTLM, Kerberos, dll.), pertimbangkan menggunakan layanan LLM Gateway yang mendukung metode autentikasi Anda.

Sertifikat CA kustom

Jika lingkungan enterprise Anda menggunakan CA kustom untuk koneksi HTTPS (baik melalui proxy atau akses API langsung), konfigurasikan Claude Code untuk mempercayainya:

```
export NODE_EXTRA_CA_CERTS=/path/to/ca-cert.pem
```

Autentikasi mTLS

Untuk lingkungan enterprise yang memerlukan autentikasi sertifikat klien:

```
## Sertifikat klien untuk autentikasi
export CLAUDE_CODE_CLIENT_CERT=/path/to/client-cert.pem

## Kunci pribadi klien
export CLAUDE_CODE_CLIENT_KEY=/path/to/client-key.pem

## Opsional: Frasa sandi untuk kunci pribadi terenkripsi
export CLAUDE_CODE_CLIENT_KEY_PASSPHRASE="your-passphrase"
```

Persyaratan akses jaringan

Claude Code memerlukan akses ke URL berikut:

- `api.anthropic.com` : Titik akhir Claude API
- `claude.ai` : autentikasi untuk akun claude.ai
- `platform.claude.com` : autentikasi untuk akun Anthropic Console

Pastikan URL ini diizinkan dalam konfigurasi proxy dan aturan firewall Anda. Ini sangat penting ketika menggunakan Claude Code di lingkungan jaringan terkontainer atau terbatas.

Sumber daya tambahan

- Pengaturan Claude Code
- Referensi variabel lingkungan
- Panduan pemecahan masalah

Part 12: Environment Setup

Kontainer pengembangan

> *Pelajari tentang kontainer pengembangan Claude Code untuk tim yang membutuhkan lingkungan yang konsisten dan aman.*

Referensi pengaturan devcontainer dan Dockerfile terkait menawarkan kontainer pengembangan yang telah dikonfigurasi sebelumnya yang dapat Anda gunakan apa adanya, atau sesuaikan dengan kebutuhan Anda. Devcontainer ini bekerja dengan ekstensi Dev Containers Visual Studio Code dan alat serupa.

Langkah-langkah keamanan yang ditingkatkan dari kontainer (isolasi dan aturan firewall) memungkinkan Anda menjalankan `claude --dangerously-skip-permissions` untuk melewati permintaan izin untuk operasi tanpa pengawasan.

> **Warning:**
>
> Meskipun devcontainer menyediakan perlindungan yang substansial, tidak ada sistem yang sepenuhnya kebal terhadap semua serangan. Ketika dijalankan dengan `--dangerously-skip-permissions`, devcontainer tidak mencegah proyek berbahaya dari mengekstraksi apa pun yang dapat diakses di devcontainer termasuk kredensial Claude Code. Kami merekomendasikan hanya menggunakan devcontainer saat mengembangkan dengan repositori terpercaya. Selalu pertahankan praktik keamanan yang baik dan pantau aktivitas Claude.

Fitur utama

- **Node.js siap produksi**: Dibangun di atas Node.js 20 dengan dependensi pengembangan penting
- **Keamanan dengan desain**: Firewall khusus yang membatasi akses jaringan hanya ke layanan yang diperlukan
- **Alat ramah pengembang**: Mencakup git, ZSH dengan peningkatan produktivitas, fzf, dan lainnya
- **Integrasi VS Code yang mulus**: Ekstensi yang telah dikonfigurasi sebelumnya dan pengaturan yang dioptimalkan

- **Persistensi sesi**: Mempertahankan riwayat perintah dan konfigurasi antara restart kontainer
- **Bekerja di mana saja**: Kompatibel dengan lingkungan pengembangan macOS, Windows, dan Linux

Memulai dalam 4 langkah

1. Instal VS Code dan ekstensi Remote - Containers
2. Kloning repositori implementasi referensi Claude Code
3. Buka repositori di VS Code
4. Ketika diminta, klik “Reopen in Container” (atau gunakan Command Palette: Cmd+Shift+P → “Remote-Containers: Reopen in Container”)

Rincian konfigurasi

Pengaturan devcontainer terdiri dari tiga komponen utama:

- **devcontainer.json**: Mengontrol pengaturan kontainer, ekstensi, dan pemasangan volume
- **Dockerfile**: Mendefinisikan citra kontainer dan alat yang diinstal
- **init-firewall.sh**: Menetapkan aturan keamanan jaringan

Fitur keamanan

Kontainer mengimplementasikan pendekatan keamanan berlapis dengan konfigurasi firewallnya:

- **Kontrol akses yang tepat**: Membatasi koneksi keluar hanya ke domain yang diizinkan (registri npm, GitHub, Claude API, dll.)
- **Koneksi keluar yang diizinkan**: Firewall memungkinkan koneksi DNS dan SSH keluar
- **Kebijakan default-deny**: Memblokir semua akses jaringan eksternal lainnya
- **Verifikasi startup**: Memvalidasi aturan firewall ketika kontainer diinisialisasi
- **Isolasi**: Membuat lingkungan pengembangan yang aman terpisah dari sistem utama Anda

Opsi kustomisasi

Konfigurasi devcontainer dirancang untuk dapat disesuaikan dengan kebutuhan Anda:

- Tambahkan atau hapus ekstensi VS Code berdasarkan alur kerja Anda

- Modifikasi alokasi sumber daya untuk lingkungan perangkat keras yang berbeda
- Sesuaikan izin akses jaringan
- Sesuaikan konfigurasi shell dan alat pengembang

Contoh kasus penggunaan

Pekerjaan klien yang aman

Gunakan devcontainer untuk mengisolasi proyek klien yang berbeda, memastikan kode dan kredensial tidak pernah bercampur antar lingkungan.

Onboarding tim

Anggota tim baru dapat mendapatkan lingkungan pengembangan yang sepenuhnya di-konfigurasi dalam hitungan menit, dengan semua alat dan pengaturan yang diperlukan telah diinstal sebelumnya.

Lingkungan CI/CD yang konsisten

Cerminkan konfigurasi devcontainer Anda dalam pipeline CI/CD untuk memastikan lingkungan pengembangan dan produksi cocok.

Sumber daya terkait

- Dokumentasi devcontainer VS Code
- Praktik terbaik keamanan Claude Code
- Konfigurasi jaringan perusahaan

Optimalkan pengaturan terminal Anda

> *Claude Code bekerja paling baik ketika terminal Anda dikonfigurasi dengan benar. Ikuti panduan ini untuk mengoptimalkan pengalaman Anda.*

Tema dan tampilan

Claude tidak dapat mengontrol tema terminal Anda. Itu ditangani oleh aplikasi terminal Anda. Anda dapat mencocokkan tema Claude Code dengan terminal Anda kapan saja melalui perintah `/config`.

Untuk penyesuaian tambahan dari antarmuka Claude Code itu sendiri, Anda dapat mengonfigurasi baris status khusus untuk menampilkan informasi kontekstual seperti model saat ini, direktori kerja, atau cabang git di bagian bawah terminal Anda.

Jeda baris

Anda memiliki beberapa opsi untuk memasukkan jeda baris ke dalam Claude Code:

- **Escape cepat**: Ketik `\` diikuti Enter untuk membuat baris baru
- **Shift+Enter**: Bekerja langsung di iTerm2, WezTerm, Ghostty, dan Kitty
- **Pintasan keyboard**: Atur keybinding untuk menyisipkan baris baru di terminal lain

Atur Shift+Enter untuk terminal lain

Jalankan `/terminal-setup` dalam Claude Code untuk secara otomatis mengonfigurasi Shift+Enter untuk VS Code, Alacritty, Zed, dan Warp.

> **Note:**
>
> Perintah `/terminal-setup` hanya terlihat di terminal yang memerlukan konfigurasi manual. Jika Anda menggunakan iTerm2, WezTerm, Ghostty, atau Kitty, Anda tidak akan melihat perintah ini karena Shift+Enter sudah bekerja secara native.

Atur Option+Enter (VS Code, iTerm2 atau macOS Terminal.app)

Untuk Mac Terminal.app:

1. Buka Settings → Profiles → Keyboard
2. Centang “Use Option as Meta Key”

Untuk iTerm2 dan terminal VS Code:

1. Buka Settings → Profiles → Keys
2. Di bawah General, atur Left/Right Option key ke “Esc+”

Pengaturan notifikasi

Jangan lewatkan saat Claude menyelesaikan tugas dengan konfigurasi notifikasi yang tepat:

Notifikasi sistem iTerm 2

Untuk peringatan iTerm 2 saat tugas selesai:

1. Buka iTerm 2 Preferences
2. Navigasi ke Profiles → Terminal
3. Aktifkan “Silence bell” dan Filter Alerts → “Send escape sequence-generated alerts”
4. Atur penundaan notifikasi pilihan Anda

Perhatikan bahwa notifikasi ini khusus untuk iTerm 2 dan tidak tersedia di Terminal macOS default.

Hook notifikasi khusus

Untuk penanganan notifikasi tingkat lanjut, Anda dapat membuat hook notifikasi untuk menjalankan logika Anda sendiri.

Menangani input besar

Saat bekerja dengan kode ekstensif atau instruksi panjang:

- **Hindari penempelan langsung**: Claude Code mungkin kesulitan dengan konten yang ditempel sangat panjang
- **Gunakan alur kerja berbasis file**: Tulis konten ke file dan minta Claude untuk membacanya
- **Waspadai keterbatasan VS Code**: Terminal VS Code sangat rentan terhadap pemotongan penempelan panjang

Mode Vim

Claude Code mendukung subset keybinding Vim yang dapat diaktifkan dengan `/vim` atau dikonfigurasi melalui `/config`.

Subset yang didukung mencakup:

- Pengalihan mode: `Esc` (ke NORMAL), `i` / `I`, `a` / `A`, `o` / `O` (ke INSERT)

- Navigasi: `h` / `j` / `k` / `l`, `w` / `e` / `b`, `0` / `$` / `^`, `gg` / `G`, `f` / `F` / `t` / `T` dengan pengulangan `;` / `,`
- Pengeditan: `x`, `dw` / `de` / `db` / `dd` / `D`, `cw` / `ce` / `cb` / `cc` / `C`, `.` (ulangi)
- Yank/paste: `yy` / `Y`, `yw` / `ye` / `yb`, `p` / `P`
- Objek teks: `iw` / `aw`, `iW` / `aW`, `i"` / `a"`, `i'` / `a'`, `i(` / `a(`, `i[` / `a[`, `i{` / `a{`
- Indentasi: `>>` / `<<`
- Operasi baris: `J` (gabung baris)

Lihat Mode interaktif untuk referensi lengkap.

Sesuaikan baris status Anda

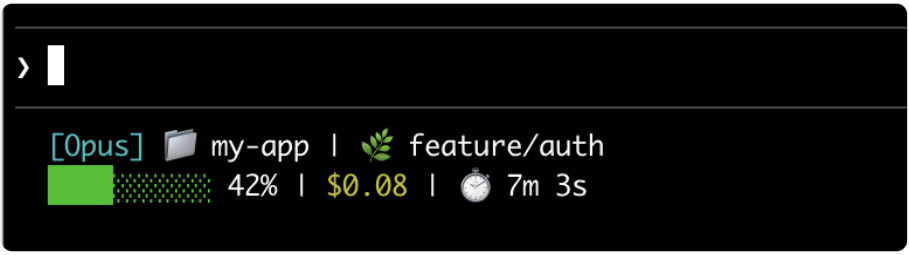
> *Konfigurasikan bilah status khusus untuk memantau penggunaan jendela konteks, biaya, dan status git di Claude Code*

Baris status adalah bilah yang dapat disesuaikan di bagian bawah Claude Code yang menjalankan skrip shell apa pun yang Anda konfigurasikan. Skrip menerima data sesi JSON di stdin dan menampilkan apa pun yang dicetak skrip Anda, memberikan Anda tampilan yang persisten dan sekilas dari penggunaan konteks, biaya, status git, atau apa pun yang ingin Anda lacak.

Baris status berguna ketika Anda:

- Ingin memantau penggunaan jendela konteks saat bekerja
- Perlu melacak biaya sesi
- Bekerja di berbagai sesi dan perlu membedakannya
- Ingin cabang git dan status selalu terlihat

Berikut adalah contoh baris status multi-baris yang menampilkan informasi git di baris pertama dan bilah konteks berkode warna di baris kedua.



Baris status multi-baris yang menampilkan nama model, direktori, cabang git di baris pertama, dan bilah kemajuan penggunaan konteks dengan biaya dan durasi di baris kedua

Halaman ini memandu Anda melalui pengaturan baris status dasar, menjelaskan bagaimana aliran data dari Claude Code ke skrip Anda, mencantumkan semua bidang yang dapat Anda tampilkan, dan menyediakan contoh siap pakai untuk pola umum seperti status git, pelacakan biaya, dan bilah kemajuan.

Atur baris status

Gunakan perintah `/statusline` untuk membuat Claude Code menghasilkan skrip untuk Anda, atau buat skrip secara manual dan tambahkan ke pengaturan Anda.

Gunakan perintah /statusline

Perintah `/statusline` menerima instruksi bahasa alami yang mendeskripsikan apa yang ingin Anda tampilkan. Claude Code menghasilkan file skrip di `~/.claude/` dan memperbarui pengaturan Anda secara otomatis:

```
/statusline show model name and context percentage with a progress bar
```

Konfigurasikan baris status secara manual

Tambahkan bidang `statusLine` ke pengaturan pengguna Anda (`~/.claude/settings.json`, di mana `~` adalah direktori home Anda) atau pengaturan proyek. Atur `type` ke `"command"` dan arahkan `command` ke jalur skrip atau perintah shell inline. Untuk panduan lengkap membuat skrip, lihat Bangun baris status langkah demi langkah.

```
{
  "statusLine": {
    "type": "command",
    "command": "~/.claude/statusline.sh",
    "padding": 2
  }
}
```

Bidang `command` berjalan di shell, jadi Anda juga dapat menggunakan perintah inline alih-alih file skrip. Contoh ini menggunakan `jq` untuk mengurai input JSON dan menampilkan nama model dan persentase konteks:

```
{
  "statusLine": {
    "type": "command",
    "command": "jq -r '\"[\\(.model.display_name)] \\(.context_window.used_percent
age // 0)% context\"'"
  }
}
```

Bidang `padding` opsional menambahkan spasi horizontal ekstra (dalam karakter) ke konten baris status. Default ke `0`. Padding ini selain spasi bawaan antarmuka, jadi mengontrol indentasi relatif daripada jarak absolut dari tepi terminal.

Nonaktifkan baris status

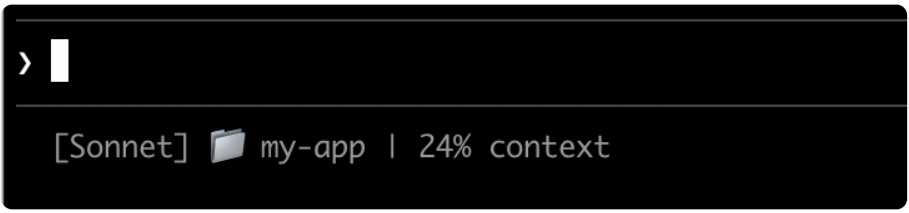
Jalankan `/statusline` dan minta untuk menghapus atau menghapus baris status Anda (misalnya, `/statusline delete`, `/statusline clear`, `/statusline remove it`). Anda juga dapat secara manual menghapus bidang `statusLine` dari settings.json Anda.

Bangun baris status langkah demi langkah

Panduan ini menunjukkan apa yang terjadi di balik layar dengan membuat baris status secara manual yang menampilkan model saat ini, direktori kerja, dan persentase penggunaan jendela konteks.

> **Note:**
>
> Menjalankan `/statusline` dengan deskripsi apa yang Anda inginkan mengonfigurasi semua ini untuk Anda secara otomatis.

Contoh-contoh ini menggunakan skrip Bash, yang berfungsi di macOS dan Linux. Di Windows, lihat Konfigurasi Windows untuk contoh PowerShell dan Git Bash.



Baris status yang menampilkan nama model, direktori, dan persentase konteks

Step 1: Buat skrip yang membaca JSON dan mencetak output

Claude Code mengirim data JSON ke skrip Anda melalui stdin. Skrip ini menggunakan `jq`, pengurai JSON baris perintah yang mungkin perlu Anda instal, untuk mengekstrak nama model, direktori, dan persentase konteks, kemudian mencetak baris yang diformat.

Simpan ini ke `~/.claude/statusline.sh` (di mana `~` adalah direktori home Anda, seperti `/Users/username` di macOS atau `/home/username` di Linux):

```
#!/bin/bash
## Read JSON data that Claude Code sends to stdin
input=$(cat)

## Extract fields using jq
MODEL=$(echo "$input" | jq -r '.model.display_name')
DIR=$(echo "$input" | jq -r '.workspace.current_dir')
## The "// 0" provides a fallback if the field is null
PCT=$(echo "$input" | jq -r '.context_window.used_percentage // 0' | cut -d. -f1)

## Output the status line - ${DIR##*/} extracts just the folder name
echo "[$MODEL] 📁 ${DIR##*/} | ${PCT}% context"
```

Step 2: Buat dapat dieksekusi

Tandai skrip sebagai dapat dieksekusi sehingga shell Anda dapat menjalankannya:

```
chmod +x ~/.claude/statusline.sh
```

Step 3: Tambahkan ke pengaturan

Beri tahu Claude Code untuk menjalankan skrip Anda sebagai baris status. Tambahkan konfigurasi ini ke `~/.claude/settings.json`, yang menetapkan `type` ke `"command"` (berarti “jalankan perintah shell ini”) dan menunjuk `command` ke skrip Anda:

```
{
  "statusLine": {
    "type": "command",
    "command": "~/.claude/statusline.sh"
  }
}
```

Baris status Anda muncul di bagian bawah antarmuka. Pengaturan dimuat ulang secara otomatis, tetapi perubahan tidak akan muncul sampai interaksi berikutnya Anda dengan Claude Code.

Bagaimana baris status bekerja

Claude Code menjalankan skrip Anda dan menyalurkan data sesi JSON ke dalamnya melalui stdin. Skrip Anda membaca JSON, mengekstrak apa yang dibutuhkan, dan mencetak teks ke stdout. Claude Code menampilkan apa pun yang dicetak skrip Anda.

Kapan itu diperbarui

Skrip Anda berjalan setelah setiap pesan asisten baru, ketika mode izin berubah, atau ketika vim mode beralih. Pembaruan dibatasi pada 300ms, berarti perubahan cepat dikumpulkan bersama dan skrip Anda berjalan sekali semuanya stabil. Jika pembaruan baru dipicu saat skrip Anda masih berjalan, eksekusi yang sedang berlangsung dibatalkan. Jika Anda mengedit skrip Anda, perubahan tidak akan muncul sampai interaksi berikutnya Anda dengan Claude Code memicu pembaruan.

Apa yang dapat dicetak skrip Anda

- **Beberapa baris**: setiap pernyataan `echo` atau `print` ditampilkan sebagai baris terpisah. Lihat contoh multi-baris.
- **Warna**: gunakan kode escape ANSI seperti `\033[32m` untuk hijau (terminal harus mendukungnya). Lihat contoh status git.
- **Tautan**: gunakan urutan escape OSC 8 untuk membuat teks dapat diklik (Cmd+klik di macOS, Ctrl+klik di Windows/Linux). Memerlukan terminal yang mendukung hyperlink seperti iTerm2, Kitty, atau WezTerm. Lihat contoh tautan yang dapat diklik.

> **Note:**

Baris status berjalan secara lokal dan tidak menggunakan token API. Itu sementara bersembunyi selama interaksi UI tertentu, termasuk saran pelengkapan otomatis, menu bantuan, dan prompt izin.

Data yang tersedia

Claude Code mengirim bidang JSON berikut ke skrip Anda melalui stdin:

Bidang	Deskripsi
`model.id`, `model.display_name`	Pengidentifikasi model saat ini dan nama tampilan
`cwd`, `workspace.current_dir`	Direktori kerja saat ini. Kedua bidang berisi nilai yang sama; `workspace.current_dir` lebih disukai untuk konsistensi dengan `workspace.project_dir`.
`workspace.project_dir`	Direktori tempat Claude Code diluncurkan, yang mungkin berbeda dari `cwd` jika direktori kerja berubah selama sesi
`cost.total_cost_usd`	Total biaya sesi dalam USD
`cost.total_duration_ms`	Total waktu dinding jam sejak sesi dimulai, dalam milidetik
`cost.total_api_duration_ms`	Total waktu yang dihabiskan menunggu respons API dalam milidetik
`cost.total_lines_added`, `cost.total_lines_removed`	Baris kode yang diubah
`context_window.total_input_tokens`, `context_window.total_output_tokens`	Jumlah token kumulatif di seluruh sesi
`context_window.context_window_size`	Ukuran jendela konteks maksimum dalam token. 200000 secara default, atau 1000000 untuk model dengan konteks diperpanjang.
`context_window.used_percentage`	Persentase jendela konteks yang digunakan yang telah dihitung sebelumnya
`context_window.remaining_percentage`	Persentase jendela konteks yang tersisa yang telah dihitung sebelumnya

Bidang	Deskripsi
`context_window.current_usage`	Jumlah token dari panggilan API terakhir, dijelaskan dalam bidang jendela konteks
`exceeds_200k_tokens`	Apakah jumlah token total (input, cache, dan output token digabungkan) dari respons API terbaru melebihi 200k. Ini adalah ambang batas tetap terlepas dari ukuran jendela konteks aktual.
`session_id`	Pengidentifikasi sesi unik
`transcript_path`	Jalur ke file transkrip percakapan
`version`	Versi Claude Code
`output_style.name`	Nama gaya output saat ini
`vim.mode`	Mode vim saat ini (`NORMAL` atau `INSERT`) ketika vim mode diaktifkan
`agent.name`	Nama agen saat menjalankan dengan flag `--agent` atau pengaturan agen dikonfigurasi
`worktree.name`	Nama worktree aktif. Hadir hanya selama sesi `--worktree`
`worktree.path`	Jalur absolut ke direktori worktree
`worktree.branch`	Nama cabang Git untuk worktree (misalnya, `"worktree-my-feature"`). Tidak ada untuk worktree berbasis hook
`worktree.original_cwd`	Direktori Claude berada sebelum memasuki worktree
`worktree.original_branch`	Cabang Git yang diperiksa sebelum memasuki worktree. Tidak ada untuk worktree berbasis hook

Perintah baris status Anda menerima struktur JSON ini melalui stdin:

```
{
  "cwd": "/current/working/directory",
  "session_id": "abc123...",
  "transcript_path": "/path/to/transcript.jsonl",
  "model": {
    "id": "claude-opus-4-6",
    "display_name": "Opus"
  },
  "workspace": {
    "current_dir": "/current/working/directory",
    "project_dir": "/original/project/directory"
  },
  "version": "1.0.80",
  "output_style": {
    "name": "default"
  },
  "cost": {
    "total_cost_usd": 0.01234,
    "total_duration_ms": 45000,
    "total_api_duration_ms": 2300,
    "total_lines_added": 156,
    "total_lines_removed": 23
  },
  "context_window": {
    "total_input_tokens": 15234,
    "total_output_tokens": 4521,
    "context_window_size": 200000,
    "used_percentage": 8,
    "remaining_percentage": 92,
    "current_usage": {
      "input_tokens": 8500,
      "output_tokens": 1200,
      "cache_creation_input_tokens": 5000,
      "cache_read_input_tokens": 2000
    }
  },
  "exceeds_200k_tokens": false,
  "vim": {
```

```
    "mode": "NORMAL"
  },
  "agent": {
    "name": "security-reviewer"
  },
  "worktree": {
    "name": "my-feature",
    "path": "/path/to/.claude/worktrees/my-feature",
    "branch": "worktree-my-feature",
    "original_cwd": "/path/to/project",
    "original_branch": "main"
  }
}
```

Bidang yang mungkin tidak ada (tidak ada dalam JSON):

- `vim` : muncul hanya ketika vim mode diaktifkan
- `agent` : muncul hanya saat menjalankan dengan flag `--agent` atau pengaturan agen dikonfigurasi
- `worktree` : muncul hanya selama sesi `--worktree` . Ketika ada, `branch` dan `original_branch` juga mungkin tidak ada untuk worktree berbasis hook

Bidang yang mungkin `null` :

- `context_window.current_usage` : `null` sebelum panggilan API pertama dalam sesi
- `context_window.used_percentage` , `context_window.remaining_percentage` : mungkin `null` awal dalam sesi

Tangani bidang yang hilang dengan akses bersyarat dan nilai null dengan default fallback dalam skrip Anda.

Bidang jendela konteks

Objek `context_window` menyediakan dua cara untuk melacak penggunaan konteks:

- **Total kumulatif** (`total_input_tokens` , `total_output_tokens`): jumlah semua token di seluruh sesi, berguna untuk melacak konsumsi total
- **Penggunaan saat ini** (`current_usage`): jumlah token dari panggilan API terbaru, gunakan ini untuk persentase konteks yang akurat karena mencerminkan keadaan konteks aktual

Objek `current_usage` berisi:

- `input_tokens` : token input dalam konteks saat ini
- `output_tokens` : token output yang dihasilkan
- `cache_creation_input_tokens` : token yang ditulis ke cache
- `cache_read_input_tokens` : token yang dibaca dari cache

Bidang `used_percentage` dihitung dari token input saja: `input_tokens + cache_creation_input_tokens + cache_read_input_tokens` . Itu tidak termasuk `output_tokens` .

Jika Anda menghitung persentase konteks secara manual dari `current_usage` , gunakan formula input-only yang sama untuk mencocokkan `used_percentage` .

Objek `current_usage` adalah `null` sebelum panggilan API pertama dalam sesi.

Contoh

Contoh-contoh ini menunjukkan pola baris status umum. Untuk menggunakan contoh apa pun:

1. Simpan skrip ke file seperti `~/.claude/statusline.sh` (atau `.py` / `.js`)
2. Buat dapat dieksekusi: `chmod +x ~/.claude/statusline.sh`
3. Tambahkan jalur ke pengaturan Anda

Contoh Bash menggunakan `jq` untuk mengurai JSON. Python dan Node.js memiliki penguraian JSON bawaan.

Penggunaan jendela konteks

Tampilkan model saat ini dan penggunaan jendela konteks dengan bilah kemajuan visual. Setiap skrip membaca JSON dari stdin, mengekstrak bidang `used_percentage` , dan membangun bilah 10 karakter di mana blok yang diisi (▓) mewakili penggunaan:

Baris status yang menampilkan nama model dan bilah kemajuan dengan persentase

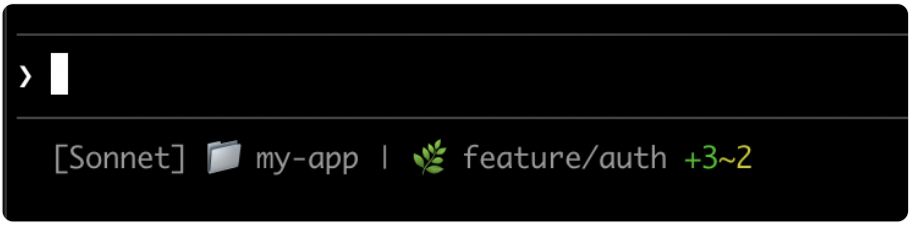
```bash
#!/bin/bash
## Read all of stdin into a variable
input=$(cat)

## Extract fields with jq, "// 0" provides fallback for null
MODEL=$(echo "$input" | jq -r '.model.display_name')
PCT=$(echo "$input" | jq -r '.context_window.used_percentage // 0' | cut -d. -f1)

## Build progress bar: printf creates spaces, tr replaces with blocks
BAR_WIDTH=10
FILLED=$((PCT * BAR_WIDTH / 100))
EMPTY=$((BAR_WIDTH - FILLED))
BAR=""
[ "$FILLED" -gt 0 ] && BAR=$(printf "%${FILLED}s" | tr ' ' '▓')
[ "$EMPTY" -gt 0 ] && BAR="${BAR}$(printf "%${EMPTY}s" | tr ' ' '░')"

echo "[$MODEL] $BAR $PCT%"
```

```python
#!/usr/bin/env python3
import json, sys

## json.load reads and parses stdin in one step
data = json.load(sys.stdin)
model = data['model']['display_name']
## "or 0" handles null values
pct = int(data.get('context_window', {}).get('used_percentage', 0) or 0)

## String multiplication builds the bar
filled = pct * 10 // 100
bar = '▓' * filled + '░' * (10 - filled)

print(f"[{model}] {bar} {pct}%")
```

```
#!/usr/bin/env node
// Node.js reads stdin asynchronously with events
let input = '';
process.stdin.on('data', chunk => input += chunk);
process.stdin.on('end', () => {
    const data = JSON.parse(input);
    const model = data.model.display_name;
    // Optional chaining (?.) safely handles null fields
    const pct = Math.floor(data.context_window?.used_percentage || 0);

    // String.repeat() builds the bar
    const filled = Math.floor(pct * 10 / 100);
    const bar = '▓'.repeat(filled) + '░'.repeat(10 - filled);

    console.log(`[${model}] ${bar} ${pct}%`);
});
```

Status git dengan warna

Tampilkan cabang git dengan indikator berkode warna untuk file yang dipentingkan dan dimodifikasi. Skrip ini menggunakan kode escape ANSI untuk warna terminal: `\033[32m` adalah hijau, `\033[33m` adalah kuning, dan `\033[0m` mengatur ulang ke default.



Baris status yang menampilkan model, direktori, cabang git, dan indikator berkode warna untuk file yang dipentingkan dan dimodifikasi

Setiap skrip memeriksa apakah direktori saat ini adalah repositori git, menghitung file yang dipentingkan dan dimodifikasi, dan menampilkan indikator berkode warna:

```
#!/bin/bash
input=$(cat)

MODEL=$(echo "$input" | jq -r '.model.display_name')
DIR=$(echo "$input" | jq -r '.workspace.current_dir')

GREEN='\033[32m'
YELLOW='\033[33m'
RESET='\033[0m'

if git rev-parse --git-dir > /dev/null 2>&1; then
    BRANCH=$(git branch --show-current 2>/dev/null)
    STAGED=$(git diff --cached --numstat 2>/dev/null | wc -l | tr -d ' ')
    MODIFIED=$(git diff --numstat 2>/dev/null | wc -l | tr -d ' ')

    GIT_STATUS=""
    [ "$STAGED" -gt 0 ] && GIT_STATUS="${GREEN}+${STAGED}${RESET}"
    [ "$MODIFIED" -gt 0 ] && GIT_STATUS="${GIT_STATUS}${YELLOW}~${MODIFIED}$
{RESET}"

    echo -e "[$MODEL] 📁 ${DIR##*/} | 🌿 $BRANCH $GIT_STATUS"
else
    echo "[$MODEL] 📁 ${DIR##*/}"
fi
```

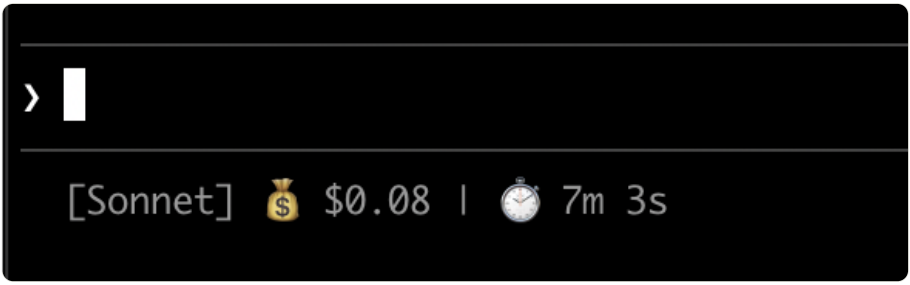
```
#!/usr/bin/env python3
import json, sys, subprocess, os

data = json.load(sys.stdin)
model = data['model']['display_name']
directory = os.path.basename(data['workspace']['current_dir'])

GREEN, YELLOW, RESET = '\033[32m', '\033[33m', '\033[0m'

try:
    subprocess.check_output(['git', 'rev-parse', '--git-dir'], stderr=subprocess.D
EVNULL)
    branch = subprocess.check_output(['git', 'branch', '--show-current'], text=Tru
e).strip()
    staged_output = subprocess.check_output(['git', 'diff', '--cached', '--
numstat'], text=True).strip()
    modified_output = subprocess.check_output(['git', 'diff', '--numstat'], text=T
rue).strip()
    staged = len(staged_output.split('\n')) if staged_output else 0
    modified = len(modified_output.split('\n')) if modified_output else 0

    git_status = f"{GREEN}+{staged}{RESET}" if staged else ""
    git_status += f"{YELLOW}~{modified}{RESET}" if modified else ""

    print(f"[{model}] 📁 {directory} | 🌿 {branch} {git_status}")
except:
    print(f"[{model}] 📁 {directory}")
```

```
#!/usr/bin/env node
const { execSync } = require('child_process');
const path = require('path');

let input = '';
process.stdin.on('data', chunk => input += chunk);
process.stdin.on('end', () => {
    const data = JSON.parse(input);
    const model = data.model.display_name;
    const dir = path.basename(data.workspace.current_dir);

    const GREEN = '\x1b[32m', YELLOW = '\x1b[33m', RESET = '\x1b[0m';

    try {
        execSync('git rev-parse --git-dir', { stdio: 'ignore' });
        const branch = execSync('git branch --show-current', { encoding:
'utf8' }).trim();
        const staged = execSync('git diff --cached --numstat', { encoding: 'utf8'
}).trim().split('\n').filter(Boolean).length;
        const modified = execSync('git diff --numstat', { encoding:
'utf8' }).trim().split('\n').filter(Boolean).length;

        let gitStatus = staged ? `${GREEN}+${staged}${RESET}` : '';
        gitStatus += modified ? `${YELLOW}~${modified}${RESET}` : '';

        console.log(`[${model}] 📁 ${dir} | 🌿 ${branch} ${gitStatus}`);
    } catch {
        console.log(`[${model}] 📁 ${dir}`);
    }
});
```

Pelacakan biaya dan durasi

Lacak biaya API sesi Anda dan waktu yang telah berlalu. Bidang `cost.total_cost_usd` mengakumulasi biaya semua panggilan API dalam sesi saat ini. Bidang `cost.total_duration_ms` mengukur total waktu yang telah berlalu sejak sesi dimulai, sementara `cost.total_api_duration_ms` melacak hanya waktu yang dihabiskan menunggu respons API.

Setiap skrip memformat biaya sebagai mata uang dan mengonversi milidetik menjadi menit dan detik:



Baris status yang menampilkan nama model, biaya sesi, dan durasi

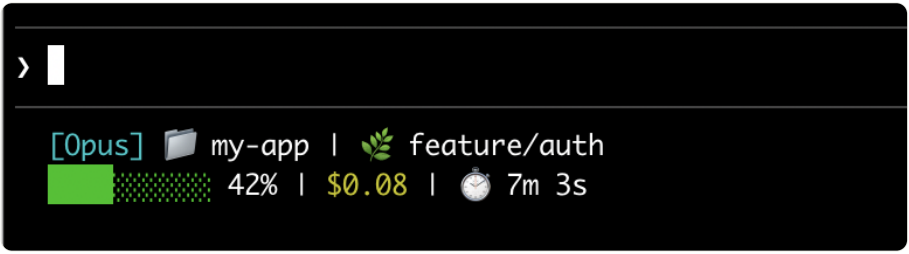
```
#!/bin/bash
input=$(cat)

MODEL=$(echo "$input" | jq -r '.model.display_name')
COST=$(echo "$input" | jq -r '.cost.total_cost_usd // 0')
DURATION_MS=$(echo "$input" | jq -r '.cost.total_duration_ms // 0')

COST_FMT=$(printf '$%.2f' "$COST")
DURATION_SEC=$((DURATION_MS / 1000))
MINS=$((DURATION_SEC / 60))
SECS=$((DURATION_SEC % 60))

echo "[$MODEL] 💰 $COST_FMT | ⏱️ ${MINS}m ${SECS}s"
```

```python
#!/usr/bin/env python3
import json, sys

data = json.load(sys.stdin)
model = data['model']['display_name']
cost = data.get('cost', {}).get('total_cost_usd', 0) or 0
duration_ms = data.get('cost', {}).get('total_duration_ms', 0) or 0

duration_sec = duration_ms // 1000
mins, secs = duration_sec // 60, duration_sec % 60

print(f"[{model}] 💰 ${cost:.2f} | ⏱️ {mins}m {secs}s")
```

```javascript
#!/usr/bin/env node
let input = '';
process.stdin.on('data', chunk => input += chunk);
process.stdin.on('end', () => {
    const data = JSON.parse(input);
    const model = data.model.display_name;
    const cost = data.cost?.total_cost_usd || 0;
    const durationMs = data.cost?.total_duration_ms || 0;

    const durationSec = Math.floor(durationMs / 1000);
    const mins = Math.floor(durationSec / 60);
    const secs = durationSec % 60;

    console.log(`[${model}] 💰 $${cost.toFixed(2)} | ⏱️ ${mins}m ${secs}s`);
});
```

Tampilkan beberapa baris

Skrip Anda dapat menampilkan beberapa baris untuk membuat tampilan yang lebih kaya. Setiap pernyataan `echo` menghasilkan baris terpisah di area status.



Baris status multi-baris yang menampilkan nama model, direktori, cabang git di baris pertama, dan bilah kemajuan penggunaan konteks dengan biaya dan durasi di baris kedua

Contoh ini menggabungkan beberapa teknik: warna berbasis ambang batas (hijau di bawah 70%, kuning 70-89%, merah 90%+), bilah kemajuan, dan informasi cabang git. Setiap pernyataan `print` atau `echo` membuat baris terpisah:

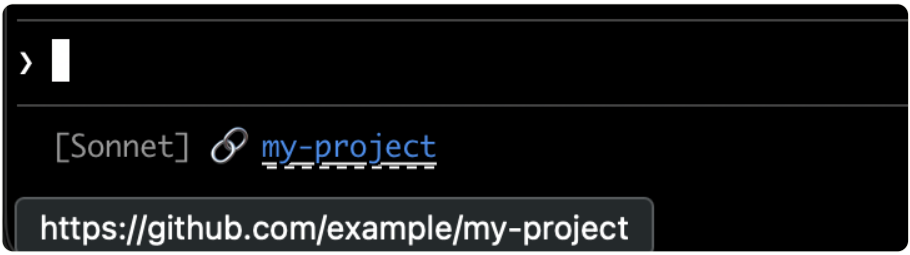
```
#!/bin/bash
input=$(cat)

MODEL=$(echo "$input" | jq -r '.model.display_name')
DIR=$(echo "$input" | jq -r '.workspace.current_dir')
COST=$(echo "$input" | jq -r '.cost.total_cost_usd // 0')
PCT=$(echo "$input" | jq -r '.context_window.used_percentage // 0' | cut -d. -f1)
DURATION_MS=$(echo "$input" | jq -r '.cost.total_duration_ms // 0')

CYAN='\033[36m'; GREEN='\033[32m'; YELLOW='\033[33m'; RED='\033[31m'; RESET='\033[
0m'

## Pick bar color based on context usage
if [ "$PCT" -ge 90 ]; then BAR_COLOR="$RED"
elif [ "$PCT" -ge 70 ]; then BAR_COLOR="$YELLOW"
else BAR_COLOR="$GREEN"; fi

FILLED=$((PCT / 10)); EMPTY=$((10 - FILLED))
BAR=$(printf "%${FILLED}s" | tr ' ' '▓')$(printf "%${EMPTY}s" | tr ' ' '░')

MINS=$((DURATION_MS / 60000)); SECS=$(((DURATION_MS % 60000) / 1000))

BRANCH=""
git rev-parse --git-dir > /dev/null 2>&1 && BRANCH=" | 🌿 $(git branch --show-
current 2>/dev/null)"

echo -e "${CYAN}[$MODEL]${RESET} 📁 ${DIR##*/}$BRANCH"
COST_FMT=$(printf '$%.2f' "$COST")
echo -e "${BAR_COLOR}${BAR}${RESET} ${PCT}% | ${YELLOW}${COST_FMT}${RESET} | ⏱️ $
{MINS}m ${SECS}s"
```

```
#!/usr/bin/env python3
import json, sys, subprocess, os

data = json.load(sys.stdin)
model = data['model']['display_name']
directory = os.path.basename(data['workspace']['current_dir'])
cost = data.get('cost', {}).get('total_cost_usd', 0) or 0
pct = int(data.get('context_window', {}).get('used_percentage', 0) or 0)
duration_ms = data.get('cost', {}).get('total_duration_ms', 0) or 0

CYAN, GREEN, YELLOW, RED, RESET = '\033[36m', '\033[32m', '\033[33m', '\033[31m',
'\033[0m'

bar_color = RED if pct ≥ 90 else YELLOW if pct ≥ 70 else GREEN
filled = pct // 10
bar = '█' * filled + '░' * (10 - filled)

mins, secs = duration_ms // 60000, (duration_ms % 60000) // 1000

try:
    branch = subprocess.check_output(['git', 'branch', '--show-current'], text=Tru
e, stderr=subprocess.DEVNULL).strip()
    branch = f" | 🌿 {branch}" if branch else ""
except:
    branch = ""

print(f"{CYAN}[{model}]{RESET} 📁 {directory}{branch}")
print(f"{bar_color}{bar}{RESET} {pct}% | {YELLOW}${cost:.2f}{RESET} | ⏱️ {mins}m
{secs}s")
```

```
#!/usr/bin/env node
const { execSync } = require('child_process');
const path = require('path');

let input = '';
process.stdin.on('data', chunk => input += chunk);
process.stdin.on('end', () => {
    const data = JSON.parse(input);
    const model = data.model.display_name;
    const dir = path.basename(data.workspace.current_dir);
    const cost = data.cost?.total_cost_usd || 0;
    const pct = Math.floor(data.context_window?.used_percentage || 0);
    const durationMs = data.cost?.total_duration_ms || 0;

    const CYAN = '\x1b[36m', GREEN = '\x1b[32m', YELLOW = '\x1b[33m', RED = '\x1b[
31m', RESET = '\x1b[0m';

    const barColor = pct >= 90 ? RED : pct >= 70 ? YELLOW : GREEN;
    const filled = Math.floor(pct / 10);
    const bar = '█'.repeat(filled) + '░'.repeat(10 - filled);

    const mins = Math.floor(durationMs / 60000);
    const secs = Math.floor((durationMs % 60000) / 1000);

    let branch = '';
    try {
        branch = execSync('git branch --show-current', { encoding: 'utf8', stdio:
['pipe', 'pipe', 'ignore'] }).trim();
        branch = branch ? ` | 🌿 ${branch}` : '';
    } catch {}

    console.log(`${CYAN}[${model}]${RESET} 📁 ${dir}${branch}`);
    console.log(`${barColor}${bar}${RESET} ${pct}% | ${YELLOW}$${cost.toFixed(2)}$
{RESET} | ⏱️ ${mins}m ${secs}s`);
});
```

Tautan yang dapat diklik

Contoh ini membuat tautan yang dapat diklik ke repositori GitHub Anda. Itu membaca URL remote git, mengonversi format SSH ke HTTPS dengan `sed`, dan membungkus nama repo dalam kode escape OSC 8. Tahan Cmd (macOS) atau Ctrl (Windows/Linux) dan klik untuk membuka tautan di browser Anda.



Baris status yang menampilkan tautan yang dapat diklik ke repositori GitHub

Setiap skrip mendapatkan URL remote git, mengonversi format SSH ke HTTPS, dan membungkus nama repo dalam kode escape OSC 8. Versi Bash menggunakan `printf '%b'` yang menginterpretasikan escape backslash lebih andal daripada `echo -e` di berbagai shell:

```
#!/bin/bash
input=$(cat)

MODEL=$(echo "$input" | jq -r '.model.display_name')

## Convert git SSH URL to HTTPS
REMOTE=$(git remote get-url origin 2>/dev/null | sed 's/git@github.com:/https:\/\/
github.com\//' | sed 's/\.git$//')

if [ -n "$REMOTE" ]; then
    REPO_NAME=$(basename "$REMOTE")
    # OSC 8 format: \e]8;;URL\a then TEXT then \e]8;;\a
    # printf %b interprets escape sequences reliably across shells
    printf '%b' "[$MODEL] 🔗 \e]8;;${REMOTE}\a${REPO_NAME}\e]8;;\a\n"
else
    echo "[$MODEL]"
fi
```

```
#!/usr/bin/env python3
import json, sys, subprocess, re, os

data = json.load(sys.stdin)
model = data['model']['display_name']

## Get git remote URL
try:
    remote = subprocess.check_output(
        ['git', 'remote', 'get-url', 'origin'],
        stderr=subprocess.DEVNULL, text=True
    ).strip()
    # Convert SSH to HTTPS format
    remote = re.sub(r'^git@github\.com:', 'https://github.com/', remote)
    remote = re.sub(r'\.git$', '', remote)
    repo_name = os.path.basename(remote)
    # OSC 8 escape sequences
    link = f"\033]8;;{remote}\a{repo_name}\033]8;;\a"
    print(f"[{model}] 🔗 {link}")
except:
    print(f"[{model}]")
```

```
#!/usr/bin/env node
const { execSync } = require('child_process');
const path = require('path');

let input = '';
process.stdin.on('data', chunk => input += chunk);
process.stdin.on('end', () => {
    const data = JSON.parse(input);
    const model = data.model.display_name;

    try {
        let remote = execSync('git remote get-url origin', { encoding: 'utf8', std
io: ['pipe', 'pipe', 'ignore'] }).trim();
        // Convert SSH to HTTPS format
        remote = remote.replace(/^git@github\.com:/, 'https://github.com/').replac
e(/\.git$/, '');
        const repoName = path.basename(remote);
        // OSC 8 escape sequences
        const link = `\x1b]8;;${remote}\x07${repoName}\x1b]8;;\x07`;
        console.log(`[${model}] 🔗 ${link}`);
    } catch {
        console.log(`[${model}]`);
    }
});
```

Cache operasi yang mahal

Skrip baris status Anda berjalan sering selama sesi aktif. Perintah seperti `git status` atau `git diff` dapat lambat, terutama di repositori besar. Contoh ini menyimpan informasi git ke file temp dan hanya menyegarkannya setiap 5 detik.

Gunakan nama file cache yang stabil dan tetap seperti `/tmp/statusline-git-cache`. Setiap invokasi baris status berjalan sebagai proses baru, jadi pengidentifikasi berbasis proses seperti `$$`, `os.getpid()`, atau `process.pid` menghasilkan nilai berbeda setiap kali dan cache tidak pernah digunakan kembali.

Setiap skrip memeriksa apakah file cache hilang atau lebih lama dari 5 detik sebelum menjalankan perintah git:

```
#!/bin/bash
input=$(cat)

MODEL=$(echo "$input" | jq -r '.model.display_name')
DIR=$(echo "$input" | jq -r '.workspace.current_dir')

CACHE_FILE="/tmp/statusline-git-cache"
CACHE_MAX_AGE=5  # seconds

cache_is_stale() {
    [ ! -f "$CACHE_FILE" ] || \
    # stat -f %m is macOS, stat -c %Y is Linux
    [ $(($(date +%s) - $(stat -f %m "$CACHE_FILE" 2>/dev/null || stat -c %Y "$CACH
E_FILE" 2>/dev/null || echo 0))) -gt $CACHE_MAX_AGE ]
}

if cache_is_stale; then
    if git rev-parse --git-dir > /dev/null 2>&1; then
        BRANCH=$(git branch --show-current 2>/dev/null)
        STAGED=$(git diff --cached --numstat 2>/dev/null | wc -l | tr -d ' ')
        MODIFIED=$(git diff --numstat 2>/dev/null | wc -l | tr -d ' ')
        echo "$BRANCH|$STAGED|$MODIFIED" > "$CACHE_FILE"
    else
        echo "||" > "$CACHE_FILE"
    fi
fi

IFS='|' read -r BRANCH STAGED MODIFIED < "$CACHE_FILE"

if [ -n "$BRANCH" ]; then
    echo "[$MODEL] 📁 ${DIR##*/} | 🌿 $BRANCH +$STAGED ~$MODIFIED"
else
    echo "[$MODEL] 📁 ${DIR##*/}"
fi
```

```
#!/usr/bin/env python3
import json, sys, subprocess, os, time

data = json.load(sys.stdin)
model = data['model']['display_name']
directory = os.path.basename(data['workspace']['current_dir'])

CACHE_FILE = "/tmp/statusline-git-cache"
CACHE_MAX_AGE = 5  # seconds

def cache_is_stale():
    if not os.path.exists(CACHE_FILE):
        return True
    return time.time() - os.path.getmtime(CACHE_FILE) > CACHE_MAX_AGE

if cache_is_stale():
    try:
        subprocess.check_output(['git', 'rev-parse', '--git-dir'], stderr=subproce
ss.DEVNULL)
        branch = subprocess.check_output(['git', 'branch', '--show-current'],
text=True).strip()
        staged = subprocess.check_output(['git', 'diff', '--cached', '--
numstat'], text=True).strip()
        modified = subprocess.check_output(['git', 'diff', '--numstat'],
text=True).strip()
        staged_count = len(staged.split('\n')) if staged else 0
        modified_count = len(modified.split('\n')) if modified else 0
        with open(CACHE_FILE, 'w') as f:
            f.write(f"{branch}|{staged_count}|{modified_count}")
    except:
        with open(CACHE_FILE, 'w') as f:
            f.write("||")

with open(CACHE_FILE) as f:
    branch, staged, modified = f.read().strip().split('|')

if branch:
```

```
    print(f"[{model}] 📁 {directory} | 🌿 {branch} +{staged} ~{modified}")
else:
    print(f"[{model}] 📁 {directory}")
```

```
#!/usr/bin/env node
const { execSync } = require('child_process');
const fs = require('fs');
const path = require('path');

let input = '';
process.stdin.on('data', chunk => input += chunk);
process.stdin.on('end', () => {
    const data = JSON.parse(input);
    const model = data.model.display_name;
    const dir = path.basename(data.workspace.current_dir);

    const CACHE_FILE = '/tmp/statusline-git-cache';
    const CACHE_MAX_AGE = 5; // seconds

    const cacheIsStale = () => {
        if (!fs.existsSync(CACHE_FILE)) return true;
        return (Date.now() / 1000) - fs.statSync(CACHE_FILE).mtimeMs / 1000 >
CACHE_MAX_AGE;
    };

    if (cacheIsStale()) {
        try {
            execSync('git rev-parse --git-dir', { stdio: 'ignore' });
            const branch = execSync('git branch --show-current', { encoding: 'utf8
' }).trim();
            const staged = execSync('git diff --cached --numstat', { encoding: 'ut
f8' }).trim().split('\n').filter(Boolean).length;
            const modified = execSync('git diff --numstat', { encoding:
'utf8' }).trim().split('\n').filter(Boolean).length;
            fs.writeFileSync(CACHE_FILE, `${branch}|${staged}|${modified}`);
        } catch {
            fs.writeFileSync(CACHE_FILE, '||');
        }
    }

    const [branch, staged, modified] = fs.readFileSync(CACHE_FILE,
'utf8').trim().split('|');
```

```
    if (branch) {
        console.log(`[${model}] 📁 ${dir} | 🌿 ${branch} +${staged} ~${modified}
`);
    } else {
        console.log(`[${model}] 📁 ${dir}`);
    }
});
```

Konfigurasi Windows

Di Windows, Claude Code menjalankan perintah baris status melalui Git Bash. Anda dapat memanggil PowerShell dari shell itu:

```
{
  "statusLine": {
    "type": "command",
    "command": "powershell -NoProfile -File C:/Users/username/.claude/
statusline.ps1"
  }
}
```

```
$input_json = $input | Out-String | ConvertFrom-Json
$cwd = $input_json.cwd
$model = $input_json.model.display_name
$used = $input_json.context_window.used_percentage
$dirname = Split-Path $cwd -Leaf

if ($used) {
    Write-Host "$dirname [$model] ctx: $used%"
} else {
    Write-Host "$dirname [$model]"
}
```

Atau jalankan skrip Bash secara langsung:

```
{
  "statusLine": {
    "type": "command",
    "command": "~/.claude/statusline.sh"
  }
}
```

```
#!/usr/bin/env bash
input=$(cat)
cwd=$(echo "$input" | grep -o '"cwd":"[^"]*"' | cut -d'"' -f4)
model=$(echo "$input" | grep -o '"display_name":"[^"]*"' | cut -d'"' -f4)
dirname="${cwd##*[/\\]}"
echo "$dirname [$model]"
```

Tips

- **Uji dengan input mock**: `echo '{"model":{"display_name":"Opus"},"context_window":{"used_percentage":25}}' | ./statusline.sh`
- **Jaga output tetap pendek**: bilah status memiliki lebar terbatas, jadi output panjang mungkin dipotong atau membungkus dengan canggung
- **Cache operasi lambat**: skrip Anda berjalan sering selama sesi aktif, jadi perintah seperti `git status` dapat menyebabkan lag. Lihat contoh caching untuk cara menangani ini.

Proyek komunitas seperti ccstatusline dan starship-claude menyediakan konfigurasi prabangun dengan tema dan fitur tambahan.

Troubleshooting

Baris status tidak muncul

- Verifikasi skrip Anda dapat dieksekusi: `chmod +x ~/.claude/statusline.sh`
- Periksa bahwa skrip Anda menampilkan ke stdout, bukan stderr
- Jalankan skrip Anda secara manual untuk memverifikasi itu menghasilkan output
- Jika `disableAllHooks` diatur ke `true` dalam pengaturan Anda, baris status juga dinonaktifkan. Hapus pengaturan ini atau atur ke `false` untuk mengaktifkan kembali.

- Jalankan `claude --debug` untuk mencatat kode keluar dan stderr dari invokasi baris status pertama dalam sesi
- Minta Claude untuk membaca file pengaturan Anda dan jalankan perintah `statusLine` secara langsung untuk mengungkap kesalahan

Baris status menampilkan `--` atau nilai kosong

- Bidang mungkin `null` sebelum respons API pertama selesai
- Tangani nilai null dalam skrip Anda dengan fallback seperti `// 0` dalam jq
- Mulai ulang Claude Code jika nilai tetap kosong setelah beberapa pesan

Persentase konteks menampilkan nilai yang tidak terduga

- Gunakan `used_percentage` untuk keadaan konteks yang akurat daripada total kumulatif
- `total_input_tokens` dan `total_output_tokens` adalah kumulatif di seluruh sesi dan mungkin melebihi ukuran jendela konteks
- Persentase konteks mungkin berbeda dari output `/context` karena kapan masing-masing dihitung

Tautan OSC 8 tidak dapat diklik

- Verifikasi terminal Anda mendukung hyperlink OSC 8 (iTerm2, Kitty, WezTerm)
- Terminal.app tidak mendukung tautan yang dapat diklik
- Sesi SSH dan tmux mungkin menghapus urutan OSC tergantung pada konfigurasi
- Jika urutan escape muncul sebagai teks literal seperti `\e]8;;`, gunakan `printf '%b'` alih-alih `echo -e` untuk penanganan escape yang lebih andal

Glitch tampilan dengan urutan escape

- Urutan escape kompleks (warna ANSI, tautan OSC 8) dapat sesekali menyebabkan output berantakan jika tumpang tindih dengan pembaruan UI lainnya
- Jika Anda melihat teks yang rusak, coba sederhanakan skrip Anda ke output teks biasa
- Baris status multi-baris dengan kode escape lebih rentan terhadap masalah rendering daripada teks biasa satu baris

Kesalahan skrip atau hang

- Skrip yang keluar dengan kode non-nol atau tidak menghasilkan output menyebabkan baris status menjadi kosong
- Skrip lambat memblokir baris status dari pembaruan sampai selesai. Jaga skrip tetap cepat untuk menghindari output basi.

- Jika pembaruan baru dipicu saat skrip lambat berjalan, skrip yang sedang berlangsung dibatalkan
- Uji skrip Anda secara independen dengan input mock sebelum mengonfigurasinya

Notifikasi berbagi baris status

- Notifikasi sistem seperti kesalahan server MCP, pembaruan otomatis, dan peringatan token ditampilkan di sisi kanan baris yang sama dengan baris status Anda
- Mengaktifkan mode verbose menambahkan penghitung token ke area ini
- Di terminal sempit, notifikasi ini mungkin memotong output baris status Anda

Part 13: Troubleshooting & Changelog

Troubleshooting

> *Temukan solusi untuk masalah umum dengan instalasi dan penggunaan Claude Code.*

Troubleshoot installation issues

> **Tip:**
>
> Jika Anda lebih suka melewati terminal sepenuhnya, aplikasi Claude Code Desktop memungkinkan Anda menginstal dan menggunakan Claude Code melalui antarmuka grafis. Unduh untuk macOS atau Windows dan mulai coding tanpa setup baris perintah apa pun.

Temukan pesan kesalahan atau gejala yang Anda lihat:

Apa yang Anda lihat	Solusi
`command not found: claude` atau `'claude' is not recognized`	Perbaiki PATH Anda
`syntax error near unexpected token '<'`	Install script returns HTML
`curl: (56) Failure writing output to destination`	Download script first, then run it
`Killed` selama install di Linux	Add swap space for low-memory servers
`TLS connect error` atau `SSL/TLS secure channel`	Update CA certificates
`Failed to fetch version` atau tidak dapat menjangkau server download	Check network and proxy settings

Apa yang Anda lihat	Solusi
`irm is not recognized` atau `&& is not valid`	Use the right command for your shell
`Claude Code on Windows requires git-bash`	Install or configure Git Bash
`Error loading shared library`	Wrong binary variant for your system
`Illegal instruction` di Linux	Architecture mismatch
`dyld: cannot load` atau `Abort trap` di macOS	Binary incompatibility
`Invoke-Expression: Missing argument in parameter list`	Install script returns HTML
`App unavailable in region`	Claude Code tidak tersedia di negara Anda. Lihat negara yang didukung.
`unable to get local issuer certificate`	Configure corporate CA certificates
`OAuth error` atau `403 Forbidden`	Fix authentication

Jika masalah Anda tidak terdaftar, kerjakan langkah-langkah diagnostik ini.

Debug installation problems

Check network connectivity

Installer mengunduh dari `storage.googleapis.com`. Verifikasi bahwa Anda dapat menjangkaunya:

```
curl -sI https://storage.googleapis.com
```

Jika ini gagal, jaringan Anda mungkin memblokir koneksi. Penyebab umum:

- Firewall perusahaan atau proxy yang memblokir Google Cloud Storage
- Pembatasan jaringan regional: coba VPN atau jaringan alternatif

- Masalah TLS/SSL: perbarui sertifikat CA sistem Anda, atau periksa apakah `HTTPS_PROXY` dikonfigurasi

Jika Anda berada di belakang proxy perusahaan, atur `HTTPS_PROXY` dan `HTTP_PROXY` ke alamat proxy Anda sebelum menginstal. Tanyakan kepada tim IT Anda untuk URL proxy jika Anda tidak mengetahuinya, atau periksa pengaturan proxy browser Anda.

Contoh ini menetapkan kedua variabel proxy, kemudian menjalankan installer melalui proxy Anda:

```
export HTTP_PROXY=http://proxy.example.com:8080
export HTTPS_PROXY=http://proxy.example.com:8080
curl -fsSL https://claude.ai/install.sh | bash
```

Verify your PATH

Jika instalasi berhasil tetapi Anda mendapatkan kesalahan `command not found` atau `not recognized` saat menjalankan `claude`, direktori instalasi tidak ada di PATH Anda. Shell Anda mencari program di direktori yang terdaftar di PATH, dan installer menempatkan `claude` di `~/.local/bin/claude` di macOS/Linux atau `%USERPROFILE%\.local\bin\claude.exe` di Windows.

Periksa apakah direktori instalasi ada di PATH Anda dengan membuat daftar entri PATH dan memfilter untuk `local/bin`:

macOS/Linux

```
echo $PATH | tr ':' '\n' | grep local/bin
```

Jika tidak ada output, direktori hilang. Tambahkan ke konfigurasi shell Anda:

```
## Zsh (macOS default)
echo 'export PATH="$HOME/.local/bin:$PATH"' >> ~/.zshrc
source ~/.zshrc

## Bash (Linux default)
echo 'export PATH="$HOME/.local/bin:$PATH"' >> ~/.bashrc
source ~/.bashrc
```

Atau, tutup dan buka kembali terminal Anda.

Verifikasi bahwa perbaikan berhasil:

```
claude --version
```

Windows PowerShell

```
$env:PATH -split ';' | Select-String 'local\\bin'
```

Jika tidak ada output, tambahkan direktori instalasi ke User PATH Anda:

```
$currentPath = [Environment]::GetEnvironmentVariable('PATH', 'User')
[Environment]::SetEnvironmentVariable('PATH', "$currentPath;$env:USERPROFILE\.local\bin", 'User')
```

Mulai ulang terminal Anda agar perubahan berlaku.

Verifikasi bahwa perbaikan berhasil:

```
claude --version
```

Windows CMD

```
echo %PATH% | findstr /i "local\bin"
```

Jika tidak ada output, buka System Settings, buka Environment Variables, dan tambahkan `%USERPROFILE%\.local\bin` ke variabel User PATH Anda. Mulai ulang terminal Anda.

Verifikasi bahwa perbaikan berhasil:

```
claude --version
```

Check for conflicting installations

Beberapa instalasi Claude Code dapat menyebabkan ketidakcocokan versi atau perilaku yang tidak terduga. Periksa apa yang diinstal:

macOS/Linux

Buat daftar semua biner `claude` yang ditemukan di PATH Anda:

```
which -a claude
```

Periksa apakah installer native dan versi npm ada:

```
ls -la ~/.local/bin/claude
```

```
ls -la ~/.claude/local/
```

```
npm -g ls @anthropic-ai/claude-code 2>/dev/null
```

Windows PowerShell

```
where.exe claude
Test-Path "$env:LOCALAPPDATA\Claude Code\claude.exe"
```

Jika Anda menemukan beberapa instalasi, pertahankan hanya satu. Instalasi native di `~/.local/bin/claude` direkomendasikan. Hapus instalasi tambahan apa pun:

Uninstall instalasi npm global:

```
npm uninstall -g @anthropic-ai/claude-code
```

Hapus instalasi Homebrew di macOS:

```
brew uninstall --cask claude-code
```

Check directory permissions

Installer memerlukan akses tulis ke `~/.local/bin/` dan `~/.claude/`. Jika instalasi gagal dengan kesalahan izin, periksa apakah direktori ini dapat ditulis:

```
test -w ~/.local/bin && echo "writable" || echo "not writable"
test -w ~/.claude && echo "writable" || echo "not writable"
```

Jika salah satu direktori tidak dapat ditulis, buat direktori instalasi dan atur pengguna Anda sebagai pemilik:

```
sudo mkdir -p ~/.local/bin
sudo chown -R $(whoami) ~/.local
```

Verify the binary works

Jika `claude` diinstal tetapi mogok atau hang saat startup, jalankan pemeriksaan ini untuk mempersempit penyebabnya.

Konfirmasi bahwa biner ada dan dapat dieksekusi:

```
ls -la $(which claude)
```

Di Linux, periksa perpustakaan bersama yang hilang. Jika `ldd` menunjukkan perpustakaan yang hilang, Anda mungkin perlu menginstal paket sistem. Di Alpine Linux dan distribusi berbasis musl lainnya, lihat Alpine Linux setup.

```
ldd $(which claude) | grep "not found"
```

Jalankan pemeriksaan kewarasan cepat bahwa biner dapat dieksekusi:

```
claude --version
```

Common installation issues

Ini adalah masalah instalasi yang paling sering dihadapi dan solusinya.

Install script returns HTML instead of a shell script

Saat menjalankan perintah install, Anda mungkin melihat salah satu kesalahan ini:

```
bash: line 1: syntax error near unexpected token `<'
bash: line 1: `<!DOCTYPE html>'
```

Di PowerShell, masalah yang sama muncul sebagai:

```
Invoke-Expression: Missing argument in parameter list.
```

Ini berarti URL instalasi mengembalikan halaman HTML alih-alih skrip instalasi. Jika halaman HTML mengatakan “App unavailable in region,” Claude Code tidak tersedia di negara Anda. Lihat negara yang didukung.

Jika tidak, ini dapat terjadi karena masalah jaringan, perutean regional, atau gangguan layanan sementara.

Solusi:

1. **Gunakan metode instalasi alternatif**:
   Di macOS atau Linux, instal melalui Homebrew:

```
brew install --cask claude-code
```

Di Windows, instal melalui WinGet:

```
winget install Anthropic.ClaudeCode
```

1. **Coba lagi setelah beberapa menit**: masalahnya sering bersifat sementara. Tunggu dan coba perintah asli lagi.

`command not found: claude` after installation

Instalasi selesai tetapi `claude` tidak berfungsi. Pesan kesalahan yang tepat bervariasi menurut platform:

Platform	Pesan kesalahan
macOS	`zsh: command not found: claude`
Linux	`bash: claude: command not found`
Windows CMD	`'claude' is not recognized as an internal or external command`
PowerShell	`claude : The term 'claude' is not recognized as the name of a cmdlet`

Ini berarti direktori instalasi tidak ada di jalur pencarian shell Anda. Lihat Verify your PATH untuk perbaikan di setiap platform.

`curl: (56) Failure writing output to destination`

Perintah `curl ... | bash` mengunduh skrip dan meneruskannya langsung ke Bash untuk dieksekusi menggunakan pipa (`|`). Kesalahan ini berarti koneksi putus sebelum skrip selesai diunduh. Penyebab umum termasuk gangguan jaringan, unduhan diblokir di tengah aliran, atau batas sumber daya sistem.

Solusi:

1. **Periksa stabilitas jaringan**: Biner Claude Code dihosting di Google Cloud Storage. Uji bahwa Anda dapat menjangkaunya:

```
curl -fsSL https://storage.googleapis.com -o /dev/null
```

Jika perintah selesai diam-diam, koneksi Anda baik-baik saja dan masalahnya mungkin intermiten. Coba ulang perintah install. Jika Anda melihat kesalahan, jaringan Anda mungkin memblokir unduhan.

1. **Coba metode instalasi alternatif**:
   Di macOS atau Linux:

```
brew install --cask claude-code
```

Di Windows:

```
winget install Anthropic.ClaudeCode
```

TLS or SSL connection errors

Kesalahan seperti `curl: (35) TLS connect error`, `schannel: next InitializeSecurityContext failed`, atau PowerShell's `Could not establish trust relationship for the SSL/TLS secure channel` menunjukkan kegagalan handshake TLS.

Solusi:

1. **Perbarui sertifikat CA sistem Anda**:
   Di Ubuntu/Debian:

```
sudo apt-get update && sudo apt-get install ca-certificates
```

Di macOS melalui Homebrew:

```
brew install ca-certificates
```

1. **Di Windows, aktifkan TLS 1.2** di PowerShell sebelum menjalankan installer:

```
[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12
irm https://claude.ai/install.ps1 | iex
```

1. **Periksa gangguan proxy atau firewall**: proxy perusahaan yang melakukan inspeksi TLS dapat menyebabkan kesalahan ini, termasuk `unable to get local issuer certificate`. Atur `NODE_EXTRA_CA_CERTS` ke bundel sertifikat CA perusahaan Anda:

```
export NODE_EXTRA_CA_CERTS=/path/to/corporate-ca.pem
```

Tanyakan kepada tim IT Anda untuk file sertifikat jika Anda tidak memilikinya. Anda juga dapat mencoba koneksi langsung untuk mengkonfirmasi proxy adalah penyebabnya.

`Failed to fetch version from storage.googleapis.com`

Installer tidak dapat menjangkau server download. Ini biasanya berarti `storage.googleapis.com` diblokir di jaringan Anda.

Solusi:

1. **Uji konektivitas secara langsung**:

```
curl -sI https://storage.googleapis.com
```

1. **Jika di belakang proxy**, atur `HTTPS_PROXY` sehingga installer dapat merutekan melaluinya. Lihat proxy configuration untuk detail.

```
export HTTPS_PROXY=http://proxy.example.com:8080
curl -fsSL https://claude.ai/install.sh | bash
```

1. **Jika di jaringan terbatas**, coba jaringan berbeda atau VPN, atau gunakan metode instalasi alternatif:

   Di macOS atau Linux:

```
brew install --cask claude-code
```

Di Windows:

```
winget install Anthropic.ClaudeCode
```

Windows: `irm` or `&&` not recognized

Jika Anda melihat `'irm' is not recognized` atau `The token '&&' is not valid`, Anda menjalankan perintah yang salah untuk shell Anda.

- `irm` **not recognized**: Anda berada di CMD, bukan PowerShell. Anda memiliki dua opsi:

  Buka PowerShell dengan mencari "PowerShell" di menu Start, kemudian jalankan perintah install asli:

```
irm https://claude.ai/install.ps1 | iex
```

Atau tetap di CMD dan gunakan installer CMD sebagai gantinya:

```
curl -fsSL https://claude.ai/install.cmd -o install.cmd && install.cmd && del
install.cmd
```

- `&&` **not valid**: Anda berada di PowerShell tetapi menjalankan perintah installer CMD. Gunakan installer PowerShell:

```
irm https://claude.ai/install.ps1 | iex
```

Install killed on low-memory Linux servers

Jika Anda melihat `Killed` selama instalasi di VPS atau instans cloud:

```
Setting up Claude Code...
Installing Claude Code native build latest...
bash: line 142: 34803 Killed    "$binary_path" install ${TARGET:+"$TARGET"}
```

Pembunuh OOM Linux menghentikan proses karena sistem kehabisan memori. Claude Code memerlukan setidaknya 4 GB RAM yang tersedia.

Solusi:

1. **Tambahkan ruang swap** jika server Anda memiliki RAM terbatas. Swap menggunakan ruang disk sebagai memori overflow, memungkinkan instalasi selesai bahkan dengan RAM fisik rendah.

   Buat file swap 2 GB dan aktifkan:

```
sudo fallocate -l 2G /swapfile
sudo chmod 600 /swapfile
sudo mkswap /swapfile
sudo swapon /swapfile
```

Kemudian coba ulang instalasi:

```
curl -fsSL https://claude.ai/install.sh | bash
```

1. **Tutup proses lain** untuk membebaskan memori sebelum menginstal.
2. **Gunakan instans yang lebih besar** jika memungkinkan. Claude Code memerlukan setidaknya 4 GB RAM.

Install hangs in Docker

Saat menginstal Claude Code di kontainer Docker, menginstal sebagai root ke `/` dapat menyebabkan hang.

Solusi:

1. **Atur direktori kerja** sebelum menjalankan installer. Saat dijalankan dari `/`, installer memindai seluruh sistem file, yang menyebabkan penggunaan memori berlebihan. Menetapkan `WORKDIR` membatasi pemindaian ke direktori kecil:

```
WORKDIR /tmp
RUN curl -fsSL https://claude.ai/install.sh | bash
```

1. **Tingkatkan batas memori Docker** jika menggunakan Docker Desktop:

```
docker build --memory=4g .
```

Windows: Claude Desktop overrides `claude` CLI command

Jika Anda menginstal versi lama Claude Desktop, itu mungkin mendaftarkan `Claude.exe` di direktori `WindowsApps` yang mengambil prioritas PATH atas Claude Code CLI. Menjalankan `claude` membuka aplikasi Desktop alih-alih CLI.

Perbarui Claude Desktop ke versi terbaru untuk memperbaiki masalah ini.

Windows: "Claude Code on Windows requires git-bash"

Claude Code di Windows native memerlukan Git for Windows, yang mencakup Git Bash.

Jika Git tidak diinstal, unduh dan instal dari git-scm.com/downloads/win. Selama setup, pilih "Add to PATH." Mulai ulang terminal Anda setelah menginstal.

Jika Git sudah diinstal tetapi Claude Code masih tidak dapat menemukannya, atur jalur di file settings.json Anda:

```
{
  "env": {
    "CLAUDE_CODE_GIT_BASH_PATH": "C:\\Program Files\\Git\\bin\\bash.exe"
  }
}
```

Jika Git Anda diinstal di tempat lain, temukan jalur dengan menjalankan `where.exe git` di PowerShell dan gunakan jalur `bin\bash.exe` dari direktori itu.

Linux: wrong binary variant installed (musl/glibc mismatch)

Jika Anda melihat kesalahan tentang perpustakaan bersama yang hilang seperti `libstdc++.so.6` atau `libgcc_s.so.1` setelah instalasi, installer mungkin telah mengunduh varian biner yang salah untuk sistem Anda.

```
Error loading shared library libstdc++.so.6: No such file or directory
```

Ini dapat terjadi pada sistem berbasis glibc yang memiliki paket cross-compilation musl terinstal, menyebabkan installer salah mendeteksi sistem sebagai musl.

Solusi:

1. **Periksa libc mana yang digunakan sistem Anda**:

```
ldd /bin/ls | head -1
```

Jika menunjukkan `linux-vdso.so` atau referensi ke `/lib/x86_64-linux-gnu/` , Anda berada di glibc. Jika menunjukkan `musl` , Anda berada di musl.

1. **Jika Anda berada di glibc tetapi mendapat biner musl**, hapus instalasi dan instal ulang. Anda juga dapat mengunduh biner yang benar secara manual dari bucket GCS di `https://storage.googleapis.com/claude-code-dist-86c565f3-f756-42ad-8dfa-d59b1c096819/claude-code-releases/{VERSION}/manifest.json` . Ajukan GitHub issue dengan output dari `ldd /bin/ls` dan `ls /lib/libc.musl*` .
2. **Jika Anda benar-benar di musl** (Alpine Linux), instal paket yang diperlukan:

```
apk add libgcc libstdc++ ripgrep
```

`Illegal instruction` on Linux

Jika installer mencetak `Illegal instruction` alih-alih pesan OOM `Killed` , biner yang diunduh tidak cocok dengan arsitektur CPU Anda. Ini biasanya terjadi pada server ARM yang menerima biner x86, atau pada CPU lama yang kekurangan set instruksi yang diperlukan.

```
bash: line 142: 2238232 Illegal instruction    "$binary_path" install ${TARGET:
+"$TARGET"}
```

Solusi:

1. **Verifikasi arsitektur Anda**:

```
uname -m
```

`x86_64` berarti 64-bit Intel/AMD, `aarch64` berarti ARM64. Jika biner tidak cocok, ajukan GitHub issue dengan output.

1. **Coba metode instalasi alternatif** sementara masalah arsitektur diselesaikan:

```
brew install --cask claude-code
```

`dyld: cannot load` on macOS

Jika Anda melihat `dyld: cannot load` atau `Abort trap: 6` selama instalasi, biner tidak kompatibel dengan versi macOS atau hardware Anda.

```
dyld: cannot load 'claude-2.1.42-darwin-x64' (load command 0x80000034 is unknown)
Abort trap: 6
```

Solusi:

1. **Periksa versi macOS Anda**: Claude Code memerlukan macOS 13.0 atau lebih baru. Buka menu Apple dan pilih About This Mac untuk memeriksa versi Anda.
2. **Perbarui macOS** jika Anda berada di versi lama. Biner menggunakan perintah load yang versi macOS lama tidak mendukung.
3. **Coba Homebrew** sebagai metode instalasi alternatif:

```
brew install --cask claude-code
```

Windows installation issues: errors in WSL

Anda mungkin mengalami masalah berikut di WSL:

OS/platform detection issues: jika Anda menerima kesalahan selama instalasi, WSL mungkin menggunakan `npm` Windows. Coba:

- Jalankan `npm config set os linux` sebelum instalasi
- Instal dengan `npm install -g @anthropic-ai/claude-code --force --no-os-check`. Jangan gunakan `sudo`.

Node not found errors: jika Anda melihat `exec: node: not found` saat menjalankan `claude`, lingkungan WSL Anda mungkin menggunakan instalasi Node.js Windows. Anda dapat mengkonfirmasi ini dengan `which npm` dan `which node`, yang harus menunjuk ke jalur Linux yang dimulai dengan `/usr/` daripada `/mnt/c/`. Untuk memperbaiki ini, coba menginstal Node melalui manajer paket distribusi Linux Anda atau melalui `nvm`.

nvm version conflicts: jika Anda memiliki nvm terinstal di WSL dan Windows, Anda mungkin mengalami konflik versi saat beralih versi Node di WSL. Ini terjadi karena WSL mengimpor PATH Windows secara default, menyebabkan Windows nvm/npm mengambil prioritas atas instalasi WSL.

Anda dapat mengidentifikasi masalah ini dengan:

- Menjalankan `which npm` dan `which node` - jika mereka menunjuk ke jalur Windows (dimulai dengan `/mnt/c/`), versi Windows sedang digunakan
- Mengalami fungsionalitas yang rusak setelah beralih versi Node dengan nvm di WSL

Untuk mengatasi masalah ini, perbaiki PATH Linux Anda untuk memastikan versi Linux node/npm mengambil prioritas:

Solusi utama: Pastikan nvm dimuat dengan benar di shell Anda

Penyebab paling umum adalah nvm tidak dimuat di shell non-interaktif. Tambahkan berikut ini ke file konfigurasi shell Anda (`~/.bashrc`, `~/.zshrc`, dll.):

```
## Load nvm if it exists
export NVM_DIR="$HOME/.nvm"
[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh"
[ -s "$NVM_DIR/bash_completion" ] && \. "$NVM_DIR/bash_completion"
```

Atau jalankan langsung di sesi saat ini:

```
source ~/.nvm/nvm.sh
```

Alternatif: Sesuaikan urutan PATH

Jika nvm dimuat dengan benar tetapi jalur Windows masih mengambil prioritas, Anda dapat secara eksplisit menambahkan jalur Linux Anda ke PATH di konfigurasi shell Anda:

```
export PATH="$HOME/.nvm/versions/node/$(node -v)/bin:$PATH"
```

> **Warning:**

> Hindari menonaktifkan impor PATH Windows melalui `appendWindowsPath = false` karena ini menghancurkan kemampuan untuk memanggil executable Windows dari WSL. Demikian pula, hindari mencopot Node.js dari Windows jika Anda menggunakannya untuk pengembangan Windows.

WSL2 sandbox setup

Sandboxing didukung di WSL2 tetapi memerlukan penginstalan paket tambahan. Jika Anda melihat kesalahan seperti "Sandbox requires socat and bubblewrap" saat menjalankan `/sandbox`, instal dependensi:

Ubuntu/Debian

```
sudo apt-get install bubblewrap socat
```

Fedora

```
sudo dnf install bubblewrap socat
```

WSL1 tidak mendukung sandboxing. Jika Anda melihat "Sandboxing requires WSL2", Anda perlu upgrade ke WSL2 atau menjalankan Claude Code tanpa sandboxing.

Permission errors during installation

Jika installer native gagal dengan kesalahan izin, direktori target mungkin tidak dapat ditulis. Lihat Check directory permissions.

Jika Anda sebelumnya menginstal dengan npm dan mengalami kesalahan izin khusus npm, beralih ke installer native:

```
curl -fsSL https://claude.ai/install.sh | bash
```

Permissions and authentication

Bagian-bagian ini mengatasi kegagalan login, masalah token, dan perilaku prompt izin.

Repeated permission prompts

Jika Anda menemukan diri Anda berulang kali menyetujui perintah yang sama, Anda dapat mengizinkan alat tertentu berjalan tanpa persetujuan menggunakan perintah `/permissions`. Lihat Permissions docs.

Authentication issues

Jika Anda mengalami masalah autentikasi:

1. Jalankan `/logout` untuk keluar sepenuhnya
2. Tutup Claude Code
3. Mulai ulang dengan `claude` dan selesaikan proses autentikasi lagi

Jika browser tidak terbuka secara otomatis selama login, tekan `c` untuk menyalin URL OAuth ke clipboard Anda, kemudian tempel ke browser Anda secara manual.

OAuth error: Invalid code

Jika Anda melihat `OAuth error: Invalid code. Please make sure the full code was copied`, kode login kedaluwarsa atau terpotong selama copy-paste.

Solusi:

- Tekan Enter untuk mencoba ulang dan selesaikan login dengan cepat setelah browser terbuka
- Ketik `c` untuk menyalin URL lengkap jika browser tidak terbuka secara otomatis
- Jika menggunakan sesi remote/SSH, browser mungkin terbuka di mesin yang salah. Salin URL yang ditampilkan di terminal dan buka di browser lokal Anda sebagai gantinya.

403 Forbidden after login

Jika Anda melihat `API Error: 403 {"error":{"type":"forbidden","message":"Request not allowed"}}` setelah login:

- **Claude Pro/Max users**: verifikasi langganan Anda aktif di claude.ai/settings
- **Console users**: konfirmkan akun Anda memiliki peran "Claude Code" atau "Developer" yang ditetapkan oleh admin Anda
- **Behind a proxy**: proxy perusahaan dapat mengganggu permintaan API. Lihat network configuration untuk setup proxy.

OAuth login fails in WSL2

Login berbasis browser di WSL2 mungkin gagal jika WSL tidak dapat membuka browser Windows Anda. Atur variabel lingkungan `BROWSER`:

```
export BROWSER="/mnt/c/Program Files/Google/Chrome/Application/chrome.exe"
claude
```

Atau salin URL secara manual: saat prompt login muncul, tekan `c` untuk menyalin URL OAuth, kemudian tempel ke browser Windows Anda.

"Not logged in" or token expired

Jika Claude Code meminta Anda untuk login lagi setelah sesi, token OAuth Anda mungkin telah kedaluwarsa.

Jalankan `/login` untuk re-authenticate. Jika ini terjadi sering, periksa bahwa jam sistem Anda akurat, karena validasi token bergantung pada timestamp yang benar.

Configuration file locations

Claude Code menyimpan konfigurasi di beberapa lokasi:

File	Tujuan
`~/.claude/settings.json`	Pengaturan pengguna (izin, hooks, override model)
`.claude/settings.json`	Pengaturan proyek (diperiksa ke kontrol sumber)
`.claude/settings.local.json`	Pengaturan proyek lokal (tidak dikomit)
`~/.claude.json`	Status global (tema, OAuth, MCP servers)
`.mcp.json`	MCP servers proyek (diperiksa ke kontrol sumber)
`managed-mcp.json`	Managed MCP servers
Managed settings	Managed settings (server-managed, MDM/OS-level policies, atau file-based)

Di Windows, `~` mengacu pada direktori home pengguna Anda, seperti `C:\Users\YourName`.

Untuk detail tentang mengonfigurasi file-file ini, lihat Settings dan MCP.

Resetting configuration

Untuk mengatur ulang Claude Code ke pengaturan default, Anda dapat menghapus file konfigurasi:

```
## Reset all user settings and state
rm ~/.claude.json
rm -rf ~/.claude/

## Reset project-specific settings
rm -rf .claude/
rm .mcp.json
```

> **Warning:**
>
> Ini akan menghapus semua pengaturan, konfigurasi server MCP, dan riwayat sesi Anda.

Performance and stability

Bagian-bagian ini mencakup masalah yang terkait dengan penggunaan sumber daya, responsivitas, dan perilaku pencarian.

High CPU or memory usage

Claude Code dirancang untuk bekerja dengan sebagian besar lingkungan pengembangan, tetapi dapat mengonsumsi sumber daya signifikan saat memproses basis kode besar. Jika Anda mengalami masalah kinerja:

1. Gunakan `/compact` secara teratur untuk mengurangi ukuran konteks
2. Tutup dan mulai ulang Claude Code di antara tugas-tugas besar
3. Pertimbangkan menambahkan direktori build besar ke file `.gitignore` Anda

Command hangs or freezes

Jika Claude Code tampak tidak responsif:

1. Tekan Ctrl+C untuk mencoba membatalkan operasi saat ini
2. Jika tidak responsif, Anda mungkin perlu menutup terminal dan memulai ulang

Search and discovery issues

Jika Search tool, `@file` mentions, custom agents, dan custom skills tidak berfungsi, instal sistem `ripgrep`:

```
## macOS (Homebrew)
brew install ripgrep

## Windows (winget)
winget install BurntSushi.ripgrep.MSVC

## Ubuntu/Debian
sudo apt install ripgrep

## Alpine Linux
apk add ripgrep

## Arch Linux
pacman -S ripgrep
```

Kemudian atur `USE_BUILTIN_RIPGREP=0` di environment Anda.

Slow or incomplete search results on WSL

Penalti kinerja pembacaan disk saat bekerja lintas sistem file di WSL dapat menghasilkan kecocokan yang lebih sedikit dari yang diharapkan saat menggunakan Claude Code di WSL. Pencarian masih berfungsi, tetapi mengembalikan hasil lebih sedikit daripada di sistem file native.

> **Note:**
>
> `/doctor` akan menunjukkan Search sebagai OK dalam kasus ini.

Solusi:

1. **Kirimkan pencarian yang lebih spesifik**: kurangi jumlah file yang dicari dengan menentukan direktori atau jenis file: “Search for JWT validation logic in the auth-service package” atau “Find use of md5 hash in JS files”.
2. **Pindahkan proyek ke sistem file Linux**: jika memungkinkan, pastikan proyek Anda berada di sistem file Linux (`/home/`) daripada sistem file Windows (`/mnt/c/`).
3. **Gunakan Windows native sebagai gantinya**: pertimbangkan menjalankan Claude Code secara native di Windows alih-alih melalui WSL, untuk kinerja sistem file yang lebih baik.

IDE integration issues

Jika Claude Code tidak terhubung ke IDE Anda atau berperilaku tidak terduga dalam terminal IDE, coba solusi di bawah ini.

JetBrains IDE not detected on WSL2

Jika Anda menggunakan Claude Code di WSL2 dengan IDE JetBrains dan mendapatkan kesalahan "No available IDEs detected", ini mungkin karena konfigurasi jaringan WSL2 atau Windows Firewall memblokir koneksi.

WSL2 networking modes

WSL2 menggunakan jaringan NAT secara default, yang dapat mencegah deteksi IDE. Anda memiliki dua opsi:

Option 1: Configure Windows Firewall (recommended)

1. Temukan alamat IP WSL2 Anda:

```
wsl hostname -I
## Example output: 172.21.123.45
```

1. Buka PowerShell sebagai Administrator dan buat aturan firewall:

```
New-NetFirewallRule -DisplayName "Allow WSL2 Internal Traffic" -Direction Inbound
-Protocol TCP -Action Allow -RemoteAddress 172.21.0.0/16 -LocalAddress 172.21.0.0/
16
```

Sesuaikan rentang IP berdasarkan subnet WSL2 Anda dari langkah 1.

1. Mulai ulang IDE dan Claude Code Anda

Option 2: Switch to mirrored networking

Tambahkan ke `.wslconfig` di direktori pengguna Windows Anda:

```
[wsl2]
networkingMode=mirrored
```

Kemudian mulai ulang WSL dengan `wsl --shutdown` dari PowerShell.

> **Note:**
>
> Masalah jaringan ini hanya mempengaruhi WSL2. WSL1 menggunakan jaringan host secara langsung dan tidak memerlukan konfigurasi ini.

Untuk tips konfigurasi JetBrains tambahan, lihat JetBrains IDE guide.

Report Windows IDE integration issues

Jika Anda mengalami masalah integrasi IDE di Windows, buat issue dengan informasi berikut:

- Tipe lingkungan: Windows native (Git Bash) atau WSL1/WSL2
- Mode jaringan WSL, jika berlaku: NAT atau mirrored
- Nama dan versi IDE
- Versi ekstensi/plugin Claude Code
- Tipe shell: Bash, Zsh, PowerShell, dll.

Escape key not working in JetBrains IDE terminals

Jika Anda menggunakan Claude Code di terminal JetBrains dan tombol `Esc` tidak mengganggu agen seperti yang diharapkan, ini mungkin karena benturan keybinding dengan pintasan default JetBrains.

Untuk memperbaiki masalah ini:

1. Buka Settings → Tools → Terminal
2. Baik:
   - Hapus centang "Move focus to the editor with Escape", atau
   - Klik "Configure terminal keybindings" dan hapus pintasan "Switch focus to Editor"
3. Terapkan perubahan

Ini memungkinkan tombol `Esc` untuk benar-benar mengganggu operasi Claude Code.

Markdown formatting issues

Claude Code kadang-kadang menghasilkan file markdown dengan tag bahasa yang hilang pada pagar kode, yang dapat mempengaruhi syntax highlighting dan keterbacaan di GitHub, editor, dan alat dokumentasi.

Missing language tags in code blocks

Jika Anda melihat blok kode seperti ini dalam markdown yang dihasilkan:

````
```
function example() {
  return "hello";
}
```text
````

Alih-alih blok yang diberi tag dengan benar seperti:

````
```javascript
function example() {
  return "hello";
}
```text
````

Solusi:

1. **Minta Claude untuk menambahkan tag bahasa**: minta "Add appropriate language tags to all code blocks in this markdown file."
2. **Gunakan post-processing hooks**: atur hooks pemformatan otomatis untuk mendeteksi dan menambahkan tag bahasa yang hilang. Lihat Auto-format code after edits untuk contoh hook PostToolUse pemformatan.
3. **Verifikasi manual**: setelah menghasilkan file markdown, tinjau untuk pemformatan blok kode yang tepat dan minta koreksi jika diperlukan.

Inconsistent spacing and formatting

Jika markdown yang dihasilkan memiliki baris kosong berlebihan atau spasi yang tidak konsisten:

Solusi:

1. **Minta koreksi pemformatan**: minta Claude untuk "Fix spacing and formatting issues in this markdown file."
2. **Gunakan alat pemformatan**: atur hooks untuk menjalankan pemformat markdown seperti `prettier` atau skrip pemformatan khusus pada file markdown yang dihasilkan.
3. **Tentukan preferensi pemformatan**: sertakan persyaratan pemformatan dalam prompt Anda atau file memory proyek.

Reduce markdown formatting issues

Untuk meminimalkan masalah pemformatan:

- **Jadilah eksplisit dalam permintaan**: minta "properly formatted markdown with language-tagged code blocks"
- **Gunakan konvensi proyek**: dokumentasikan gaya markdown pilihan Anda di `CLAUDE.md`
- **Atur validation hooks**: gunakan post-processing hooks untuk secara otomatis memverifikasi dan memperbaiki masalah pemformatan umum

Get more help

Jika Anda mengalami masalah yang tidak tercakup di sini:

1. Gunakan perintah `/bug` dalam Claude Code untuk melaporkan masalah langsung ke Anthropic
2. Periksa GitHub repository untuk masalah yang diketahui
3. Jalankan `/doctor` untuk mendiagnosis masalah. Ini memeriksa:
   - Tipe instalasi, versi, dan fungsionalitas pencarian
   - Status auto-update dan versi yang tersedia
   - File pengaturan yang tidak valid (JSON yang salah bentuk, tipe yang salah)
   - Kesalahan konfigurasi server MCP
   - Masalah konfigurasi keybinding
   - Peringatan penggunaan konteks (file CLAUDE.md besar, penggunaan token MCP tinggi, aturan izin yang tidak dapat dijangkau)
   - Plugin dan kesalahan pemuatan agen
4. Tanyakan Claude secara langsung tentang kemampuan dan fiturnya - Claude memiliki akses bawaan ke dokumentasinya