Claude Code — Referensi AI Engineer

Ringkasan konsultasi: perintah penting, fondasi proyek, alur kerja harian, kustomisasi, integrasi, keamanan & biaya. Disusun November 2026.

> **`/init` — titik awal setiap proyek**
> **Apa:** Claude memindai codebase (struktur folder, package manager, script build/test, konvensi) lalu membuat draf `CLAUDE.md` di root repo. File ini otomatis dimuat ke konteks di setiap sesi.
> **Kapan:** (1) pertama kali memakai Claude Code di sebuah repo; (2) setelah perubahan besar (stack, struktur folder, perintah build) untuk memperbarui CLAUDE.md yang sudah ada.
> **Setelahnya (wajib):** sunting hasilnya. Buang kalimat generik, tambahkan perintah test satu-file, konvensi yang tidak terbaca dari kode, dan jebakan (gotchas). Lalu commit ke Git agar dipakai seluruh tim.
> **Yang tidak dilakukan /init:** tidak mengatur permission, hook, MCP, atau subagent. Itu dikonfigurasi terpisah di `.claude/settings.json`, `.mcp.json`, dan folder `.claude/`.

A Slash command bawaan

Ketik `/` di dalam sesi untuk melihat daftar lengkap. Perintah bisa bertambah atau berubah antar versi; selalu cek `/help` di versi Anda.

Perintah	Fungsi	Kapan dipakai
PROYEK & MEMORI		
`/init`	Membuat/memperbarui draf CLAUDE.md dari analisis codebase	Awal proyek, setelah perubahan besar
`/memory`	Membuka dan menyunting file memori (CLAUDE.md proyek/pribadi)	Menambah aturan yang dilanggar Claude berulang kali
`/add-dir`	Menambah direktori kerja lain ke sesi	Monorepo, repo frontend + backend terpisah
SESI & KONTEKS		
`/clear`	Menghapus riwayat percakapan, mulai bersih	Setiap berganti tugas yang tidak berhubungan
`/compact [arahan]`	Meringkas riwayat agar konteks lega, tugas berlanjut	Sesi panjang; contoh: `/compact fokus desain API`
`/context`	Visualisasi apa saja yang memakan jendela konteks	Sesi terasa lambat/menurun, cek beban MCP
`/resume`	Memilih dan melanjutkan sesi sebelumnya	Melanjutkan pekerjaan kemarin
`/rewind`	Kembali ke titik sebelumnya; bisa membatalkan perubahan kode	Claude salah arah (sama dengan Esc Esc)
`/export`	Mengekspor percakapan ke file/clipboard	Dokumentasi, berbagi ke rekan
MODEL & BIAYA		
`/model`	Mengganti model (keluarga Opus / Sonnet / Haiku)	Opus: rencana & debug sulit; Sonnet: harian; Haiku: tugas ringan
`/cost`	Biaya token sesi saat ini	Pengguna API: pantau pengeluaran
`/usage`	Pemakaian terhadap batas paket langganan	Pengguna Pro/Max/Team
`/status`	Versi, model aktif, akun, status koneksi	Diagnosis cepat
KONFIGURASI & KUSTOMISASI		
`/config`	Membuka pengaturan (tema, model default, dll.)	Menyesuaikan preferensi
`/permissions`	Mengelola aturan allow / ask / deny	Mengurangi prompt izin, memblokir file sensitif
`/hooks`	Melihat dan mengelola hook	Auto-format, proteksi file, notifikasi
`/agents`	Membuat dan mengelola subagent	Reviewer, penulis test, auditor
`/mcp`	Status server MCP dan login OAuth	Setelah menambah server; autentikasi
`/plugin`	Memasang/mengelola plugin dan marketplace	Memakai paket alat tim lain
`/sandbox`	Mengaktifkan isolasi filesystem & jaringan (tergantung OS)	Ingin otonomi lebih tinggi dengan aman
`/statusline`	Mengatur baris status di terminal	Menampilkan model, branch, konteks
`/terminal-setup`	Mengatur keybinding (mis. Shift+Enter baris baru)	Setup awal terminal
`/vim`	Mode edit input ala Vim	Pengguna Vim
INTEGRASI & REVIEW		
`/ide`	Menghubungkan sesi ke VS Code / JetBrains	Lihat diff di editor, berbagi seleksi
`/install-github-app`	Memasang GitHub App + secret untuk Actions	Mengaktifkan `@claude` di issue/PR
`/review`	Meminta review kode / PR	Sebelum merge
`/security-review`	Review keamanan atas perubahan yang tertunda	Sebelum commit kode sensitif
`/pr-comments`	Mengambil komentar dari PR GitHub	Menindaklanjuti review
BANTUAN & AKUN		
`/help`	Daftar perintah dan bantuan	Kapan saja
`/doctor`	Memeriksa kesehatan instalasi	Ada yang tidak berfungsi
`/bug`	Mengirim laporan/umpan balik ke Anthropic	Menemukan bug
`/login` · `/logout`	Masuk/keluar akun	Ganti akun atau metode tagihan
`/exit`	Keluar dari sesi	Selesai bekerja

Slash command kustom Anda sendiri (dari `.claude/commands/`, skill, atau plugin) muncul di daftar yang sama.

B Pintasan keyboard & awalan input

Tombol / awalan	Fungsi
`Esc`	Hentikan Claude di tengah jalan; konteks tetap ada, langsung beri arahan baru
`Esc Esc`	Buka rewind: kembali ke pesan sebelumnya / batalkan perubahan kode
`Shift+Tab`	Ganti mode: normal → auto-accept edit → plan mode
`@path/file`	Merujuk file/folder langsung di prompt (dengan autocomplete)
`!perintah`	Jalankan perintah shell langsung; output masuk ke konteks
`↑` / `↓`	Riwayat prompt sebelumnya
`Ctrl+V` / seret file	Tempel atau seret screenshot/gambar ke prompt
`Ctrl+C`	Batalkan input; tekan lagi untuk keluar

C Perintah CLI & flag penting

Perintah	Fungsi
`claude`	Mulai sesi interaktif di direktori saat ini
`claude "prompt"`	Mulai sesi dengan prompt awal
`claude -c` / `--continue`	Lanjutkan sesi terakhir
`claude -r` / `--resume`	Pilih sesi lama untuk dilanjutkan
`claude -p "prompt"`	Mode headless: kerjakan, cetak hasil, selesai (untuk script & CI)
`--output-format json|stream-json`	Output terstruktur untuk diproses program lain
`--model <nama>`	Pilih model saat memulai
`--permission-mode plan|acceptEdits`	Mulai langsung di mode tertentu
`--allowedTools` / `--disallowedTools`	Batasi tool, mis. `"Read,Edit,Bash(npm test:*)"`
`--max-turns <n>`	Batasi jumlah putaran di headless (cegah loop & biaya liar)
`--add-dir <path>`	Tambah direktori kerja
`--dangerously-skip-permissions`	Tanpa konfirmasi sama sekali. **Hanya di container/VM terisolasi**
`claude mcp add | list | remove`	Kelola server MCP (`--transport http`, `--scope project`)
`claude update` · `claude doctor`	Perbarui versi · cek instalasi

1 Fondasi Proyek

Prinsip: setiap sesi dimulai **tanpa ingatan**. Fondasi = orientasi karyawan baru yang otomatis. Tiga lapisan: **CLAUDE.md** (pengetahuan) · **folder .claude/** (konfigurasi) · **kesiapan repo** (umpan balik).

Hierarki memori

Lokasi CLAUDE.md	Cakupan	Git?
`~/.claude/CLAUDE.md`	Semua proyek (preferensi pribadi)	Tidak
`./CLAUDE.md`	Seluruh tim di proyek ini	Ya
`./modul/CLAUDE.md`	Dimuat saat Claude bekerja di folder itu	Ya
`@docs/file.md`	Impor file lain agar CLAUDE.md tetap ramping	Ya

Prinsip menulis

- **Ringkas**: dimuat di setiap sesi, setiap baris dibayar berulang
- **Spesifik & imperatif**: "pakai 2 spasi", bukan "rapikan"
- **Tekankan** aturan kritis (PENTING/JANGAN) secukupnya
- **Dokumen hidup**: salah dua kali → tambahkan aturan
- **Review di PR** seperti kode

Kesiapan repo (umpan balik)

- Test suite satu perintah dan cepat
- Linter + type checker
- Script jelas di `package.json` / `Makefile`

Isi CLAUDE.md yang bernilai tinggi

- **Perintah:** build, test (termasuk satu file), lint, typecheck, dev server
- **Peta arsitektur** singkat: folder apa berisi apa
- **Konvensi** yang tidak terbaca dari kode
- **Gotchas:** "JANGAN edit `generated/`"
- **Alur kerja:** "jalankan test sebelum menyatakan selesai"

Jangan dimasukkan

- Dokumentasi panjang, aturan yang sudah dipaksa linter, secret/token, hal yang jelas dari kode

Struktur folder .claude/

```
proyek/
├── CLAUDE.md
├── .mcp.json             # server MCP tim
└── .claude/
    ├── settings.json       # permission, hooks (tim)
    ├── settings.local.json # pribadi, jangan commit
    ├── commands/           # slash command kustom
    ├── skills/<nama>/SKILL.md
    ├── agents/             # subagent
    └── hooks/              # script hook
```

2 Alur Kerja Harian

> **Pola inti:** Eksplorasi (baca, jangan ubah) → Rencana (plan mode, koreksi dulu) → Kode → Verifikasi (test, lint, typecheck) → Commit. Jangan lompat langsung ke kode untuk tugas non-sepele.

Instruksi yang efektif

- Sebut **file/pola rujukan**: "ikuti pola `SettingsPage.vue`"
- Definisikan **kriteria selesai**: "test lulus, tanpa error TS"
- Beri **bahan konkret**: error lengkap, log, screenshot
- Sebut **batasan**: apa yang tidak boleh diubah
- Bug: "tulis test yang mereproduksi dulu, baru perbaiki"
- Masalah sulit: minta Claude **berpikir lebih dalam** dan membandingkan alternatif

Memecah tugas besar

- Irisan yang bisa diverifikasi: skema → service+test → API → UI
- Checklist di `PLAN.md`, dicentang bertahap, bisa dilanjut di sesi baru
- Commit per irisan = titik aman

Mengelola konteks

- **Satu sesi, satu tujuan**
- `/clear` saat ganti tugas (kebiasaan paling berdampak)
- `/compact` di sesi panjang, `/context` untuk cek beban
- `claude -c` / `/resume` untuk melanjutkan

Mengawasi & membatalkan

- `Esc` sela lebih awal; jangan tunggu 300 baris salah
- `Esc Esc` / `/rewind` kembali ke checkpoint
- **Git tetap jaring pengaman utama**: mulai dari working tree bersih

Review hasil

- Baca diff sebelum commit
- Minta review skeptis di sesi baru (`/clear` dulu)
- Waspadai: test diubah agar lulus, error ditelan, duplikasi

Situasi	Tingkat otonomi
Area baru, kode kritis, produksi	Mode normal: setujui tiap edit & perintah
Tugas jelas + test kuat	Auto-accept edit (`Shift+Tab`), review diff di akhir
Tugas mekanis berulang	Otonomi tinggi, permission dibatasi, idealnya di lingkungan terisolasi

3 Kustomisasi

Alat	Dipicu oleh	Lokasi	Gunakan saat
Slash command	Anda (`/nama`)	`.claude/commands/*.md`	Mengetik instruksi yang sama berulang
Skill	Claude, otomatis	`.claude/skills/<nama>/SKILL.md`	Prosedur khusus yang dibutuhkan saat mengerjakan X
Hook	Sistem, pada event	`settings.json` → `hooks`	Aturan yang WAJIB, deterministik
Subagent	Claude atau Anda	`.claude/agents/*.md`	Spesialis, konteks terpisah agar sesi utama bersih
Plugin	Dipasang sekali	`/plugin`, marketplace	Membagikan semua alat di atas lintas proyek/tim
CLAUDE.md	Selalu dimuat	root repo	Fakta dasar proyek yang perlu di setiap sesi

Slash command: .claude/commands/fix-issue.md

```
---
description: Analisis dan perbaiki issue GitHub
argument-hint: [nomor-issue]
---
Perbaiki issue #$ARGUMENTS:
1. gh issue view $ARGUMENTS
2. Tulis test yang mereproduksi (harus GAGAL)
3. Perbaiki sampai test lulus, jalankan lint
4. Commit: "fix: ... (#$ARGUMENTS)"
```

Panggil: `/fix-issue 142`. `$ARGUMENTS` = teks setelah nama perintah.

Skill: SKILL.md

```
---
name: migrasi-database
description: Gunakan saat membuat/mengubah skema,
  tabel, kolom, atau index.
---
- Buat migrasi baru, jangan edit yang lama
- down() wajib membalik perubahan
```

Kunci skill: **description** yang jelas kapan dipakai. Isi hanya dimuat saat relevan.

Subagent: .claude/agents/code-reviewer.md

```
---
name: code-reviewer
description: Reviewer kritis. Pakai setelah
  perubahan signifikan, sebelum commit.
tools: Read, Grep, Glob, Bash
model: sonnet
---
Review dengan skeptis: bug, edge case,
keamanan, duplikasi. Jangan ubah kode.
Laporkan: Kritis / Penting / Saran.
```

Tanpa `Edit`/`Write` di `tools` = reviewer secara teknis tidak bisa mengubah kode. `model` per subagent mengatur biaya.

Hook: auto-format setelah edit

```
"hooks": {
  "PostToolUse": [{
    "matcher": "Edit|Write",
    "hooks": [{ "type": "command",
      "command": "jq -r '.tool_input.file_path'
        | xargs npx prettier --write" }]
  }]
}
```

Event hook	Kapan	Contoh
`PreToolUse`	Sebelum tool dipakai	Blokir akses `.env` (exit code **2** = blokir, stderr dikirim ke Claude)
`PostToolUse`	Setelah tool selesai	Auto-format, auto-lint
`UserPromptSubmit`	Saat Anda mengirim prompt	Tambah konteks otomatis
`Stop`	Claude selesai merespons	Jalankan test, notifikasi
`SessionStart`	Sesi dimulai	Muat info branch/status

Data aksi dikirim ke hook sebagai **JSON via stdin**. Prinsip: "sebaiknya" → CLAUDE.md; "harus selalu" → hook. **Adopsi bertahap:** minggu 1 hook format/lint → minggu 2 slash command → minggu 3 subagent reviewer → skill → plugin.

4 Integrasi

MCP (Model Context Protocol)

- Standar terbuka: server MCP menambah **tool baru** untuk Claude
- **stdio** = server lokal (DB lokal, browser) · **http** = remote (SaaS, OAuth)
- `claude mcp add --transport http sentry <url>`
- `claude mcp add playwright -- npx -y <paket>`
- `/mcp` di sesi: status & login OAuth

Scope konfigurasi MCP

Scope	Disimpan di	Untuk
`local`	config pribadi, proyek ini	Eksperimen
`project`	`.mcp.json` (Git)	Seluruh tim
`user`	config pribadi, semua proyek	Alat pribadi

Kredensial di `.mcp.json` selalu lewat `${ENV_VAR}`.

Headless (-p)

```
cat error.log | claude -p "kelompokkan error"
claude -p "daftar endpoint tanpa test" \
  --output-format json
claude -p "perbaiki lint di src/" \
  --allowedTools "Read,Edit,Bash(npm run lint:*)" \
  --max-turns 10
```

Migrasi massal: satu `claude -p` per file dalam loop, lebih andal daripada satu sesi raksasa. Agent kustom: **Claude Agent SDK** (Python/TypeScript).

Integrasi MCP berdampak tinggi

- **GitHub/GitLab**: baca review PR, perbaiki
- **Jira/Linear**: kerjakan tiket sesuai acceptance criteria
- **Sentry**: analisis error teratas
- **Database**: user **read-only**, ke **staging**
- **Browser (Playwright)**: Claude melihat hasil UI & memperbaiki sendiri
- **Figma**: implementasi sesuai desain

IDE

- VS Code (+ Cursor/Windsurf) & JetBrains: ekstensi/plugin resmi
- Diff visual, seleksi jadi konteks, diagnostik editor
- `/ide` untuk menghubungkan sesi terminal

GitHub Actions

- Setup: `/install-github-app` (App + secret API key)
- Mode mention: `@claude` di issue/PR → Claude mengerjakan & buka PR
- Mode otomatis: action `anthropics/claude-code-action` pada event PR
- CLAUDE.md, skill, subagent **ikut terpakai** di CI
- Batasi `permissions` workflow, filter path/label untuk hemat biaya
- Hati-hati PR dari fork (prompt injection)
- Claude **bukan pengganti** approval manusia

Urutan adopsi: IDE → 1 MCP (issue tracker/GitHub) → browser MCP (frontend) → GitHub Actions mention → review otomatis → DB MCP read-only.

5 Keamanan & Biaya

> **Pertahanan berlapis:** permission → sandbox/isolasi → hook → kebiasaan review. **Risiko:** kesalahan tak disengaja · kebocoran secret · prompt injection · rantai pasokan (MCP/hook/plugin pihak ketiga).

Permission: .claude/settings.json

```
{
  "permissions": {
    "allow": ["Bash(npm run test:*)",
              "Bash(npm run lint)",
              "Bash(git status)", "Bash(git diff:*)"],
    "ask":   ["Bash(git push:*)",
              "Bash(npm install:*)"],
    "deny":  ["Read(./.env)", "Read(./.env.*)",
              "Read(./secrets/**)", "Read(**/*.pem)",
              "Read(**/*.key)"]
  }
}
```

- **deny** menang atas segalanya; **ask** selalu konfirmasi
- **allow** perintah aman agar tidak "kelelahan menekan ya"
- Aturan `Bash(...)` = pencocokan teks, **pagar pengaman, bukan batas keamanan**. Batas sungguhan = sandbox/container

Hierarki pengaturan (kuat → lemah)

1. **Managed settings** (kebijakan admin perusahaan)
2. Argumen command line
3. `.claude/settings.local.json` (pribadi)
4. `.claude/settings.json` (tim)
5. `~/.claude/settings.json` (pribadi global)

Mode permission

Mode	Gunakan saat
Default	Area baru, kode kritis
Accept edits	Tugas jelas, test kuat
Plan	Eksplorasi & desain
Bypass	**Hanya** container/VM terisolasi

Isolasi

- `/sandbox`: batasi filesystem & domain jaringan di level OS
- Dev container/Docker tanpa kredensial pribadi + firewall
- **Otonomi besar → lingkungan kecil**

Prompt injection: putus salah satu dari tiga

1. Membaca **konten tak tepercaya** (issue publik, web, data DB)
2. Akses ke **data sensitif**
3. Kemampuan **mengirim data keluar** (jaringan, push, komentar)

Checklist rahasia

- ■ deny `.env*`, `*.pem`, `*.key`, `secrets/`
- ■ Hook `PreToolUse` sebagai lapisan kedua
- ■ `${VAR}` di `.mcp.json`, token scope minimum
- ■ DB read-only ke staging
- ■ Secret scanning di CI (mis. gitleaks)
- ■ Review hook/MCP/plugin pihak ketiga seperti dependensi produksi

Memilih model (/model)

Keluarga	Cocok untuk
Opus	Arsitektur, debugging sulit, refactor besar
Sonnet	Mayoritas coding harian
Haiku	Tugas ringan, subagent pencarian, otomatisasi volume tinggi

Opus merencanakan, Sonnet mengeksekusi. Subagent mekanis & CI: model hemat.

Kebiasaan hemat token

- Setiap giliran **mengirim ulang seluruh konteks** (prompt caching membantu)
- `/clear` ganti tugas · `/compact` sesi panjang
- CLAUDE.md ringkas · MCP seperlunya
- Prompt spesifik dengan path file
- Subagent untuk eksplorasi berat
- Minta output ringkas ("hanya test gagal")
- `--max-turns` di headless
- **Biaya termahal = pekerjaan yang diulang** → pakai plan mode

G Glosarium

Istilah	Arti singkat
Agentic loop	Siklus Claude: berpikir → pakai tool → lihat hasil → ulangi sampai tugas selesai
Context window	Memori kerja satu sesi: system prompt, tool, CLAUDE.md, riwayat, isi file
Token	Satuan teks yang dihitung untuk kapasitas konteks dan biaya
Prompt caching	Bagian konteks yang tidak berubah dikirim ulang dengan biaya jauh lebih murah
Plan mode	Mode baca-saja untuk menyusun rencana sebelum mengubah apa pun
Checkpoint / rewind	Titik simpan otomatis yang bisa dikembalikan (`Esc Esc`)
MCP	Protokol standar untuk menghubungkan Claude ke alat & data eksternal
Skill	Folder instruksi + file pendukung yang dimuat otomatis saat relevan
Subagent	Agent spesialis dengan konteks, tool, dan model sendiri
Hook	Perintah shell yang dijalankan sistem pada event tertentu, selalu
Plugin / marketplace	Paket command, skill, hook, subagent, MCP untuk dibagikan
Headless	Menjalankan Claude tanpa sesi interaktif (`-p`) untuk script & CI
Worktree	Salinan kerja Git terpisah agar beberapa sesi Claude bisa paralel tanpa bentrok
Managed settings	Kebijakan organisasi yang tidak bisa ditimpa developer
Prompt injection	Instruksi tersembunyi di konten eksternal yang mencoba membelokkan Claude
Agent SDK	Mesin Claude Code yang bisa diprogram (Python/TypeScript) untuk agent kustom

Checklist harian AI engineer

- ■ Working tree bersih sebelum mulai · ☐ satu sesi, satu tujuan
- ■ Tugas non-sepele: eksplorasi → plan mode → koreksi rencana → eksekusi
- ■ Prompt: file rujukan + kriteria selesai + batasan
- ■ Claude memverifikasi sendiri (test, lint, typecheck) sebelum menyatakan selesai
- ■ Baca diff, minta review skeptis, commit per irisan
- ■ Claude salah dua kali → perbarui CLAUDE.md, atau jadikan hook bila wajib
- ■ `/clear` saat ganti tugas · pantau `/context` dan `/cost` / `/usage`