

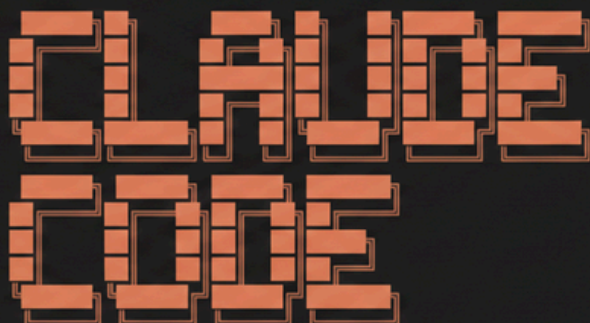


Claude Code · Cowork

Fra terminal til skrivebordsagent



* Welcome to the **Claude Code** research preview!



✦ Login successful. Press **Enter** to continue

Roadmap

Del 1 Claude Code



Innledning



Kapittel 1:

- [Abonnement og modeller](#)
- [Opus, Sonnet og Haiku](#)

Kapittel 2

- [Installasjon](#)
- [Claude.MD](#)
- [Editor](#)



Kapittel 3

- [Kommandoer](#)
- [Snarveier](#)
- [Prefiks](#)

Kapittel 4

- [Kontekst](#)
- [Fokus](#)



Kapittel 5

- [Planleggingsmodus](#)
- [Godkjenning](#)

Kapittel 6

- [Skills](#)
- [Ferdige skills](#)



Kapittel 7

- [Plugins](#)
- [Roller](#)

Kapittel 8

- [Hva er Cowork?](#)
- [Typiske oppgaver](#)



Del 2 Claude Cowork



Kapittel 9

- [Cowork i praksis](#)
- [Skills og plugins](#)

Kapittel 10

- [Sikkerhet](#)
- [Gode vaner](#)



Avslutning

Hurtigreferanse

Kilder

Innledning

Fra chat til agent. Hva er egentlig forskjellen?

Du har sikkert brukt Claude i nettleseren. Du skriver, Claude svarer. Det fungerer fint for spørsmål, tekst og ideer. Men hvis du vil ha Claude til å faktisk gjøre noe, som å lage en fil, endre kode eller gå gjennom hundre dokumenter da holder det ikke lenger å bare chatte.

Claude Code og Cowork er to verktøy der Claude ikke bare svarer, men handler. Claude kan lese filer på maskinen din, skrive og lagre nye dokumenter, og jobbe selvstendig med en oppgave fra start til slutt uten at du trenger å kopiere og lime inn ting manuelt hele veien.

Det er den store forskjellen. En vanlig chatbot hjelper deg å tenke. Disse verktøyene hjelper deg å gjøre.

Hva er en agentic loop?

Når Claude Code eller Cowork jobber, gjør den ikke alt på en gang. Den jobber i en sløyfe: **tenk-gjør-sjekk-tenk igjen**. Det er det som kalles en agentic loop.

Analogi: Tenk deg at du skal rydde skrivebordet ditt. Du ser seg rundt, fjerner en kaffekopp, sjekker om det er noe annet, tenker over noe mer du bør rydde samtidig og fortsetter til skrivebordet ser bra ut. Det er nøyaktig slik Claude Code og Cowork jobber.

I praksis betyr det at Claude:

- Leser oppgaven og lager en plan
- Utfører ett steg av gangen
- Sjekker resultatet av hvert steg
- Justerer planen om noe ikke fungerte som forventet
- Fortsetter til oppgaven er ferdig eller den trenger input fra deg

Du trenger ikke gjøre noe mens loopen kjører. Men du kan alltid avbryte med Escape og ta over manuelt.

To verktøy, to typer brukere

- **Claude Code** er for deg som vil lære å bruke terminalen og jobbe tettere med kode og filer. Det krever litt oppsett, men gir mye kontroll og fleksibilitet.
- **Cowork** er for deg som vil ha hjelp med hverdagsoppgaver sortere filer, skrive rapporter, håndtere dokumenter uten terminal og uten kode.

Mange bruker begge. Start gjerne med det som virker minst skummelt.

Slik bruker du denne boka

Denne boken er bygget opp som en innføring. Hvert kapittel inneholder forklaringer, praktiske, eksempler og øvelser du kan prøve ut med en gang. Øvelsene er markert med blå boks og forteller deg nøyaktig hva du skal skrive eller gjøre.

Tipsene er merket med fargede etiketter slik at du alltid vet hvilken plattform de gjelder:

WEB Gjelder `claude.ai` i nettleseren.

CLI Gjelder `Claude Code` i terminalen.

COWORK Gjelder `Cowork` i skrivebordsappen.

Slik starter du verktøyene

WEB `Claude.ai` bruker du i nettleseren på `claude.ai`. Logg inn og start en samtale. Ingen installasjon nødvendig.

COWORK er en skrivebordsapp du laster ned fra `anthropic.com` og installerer som et vanlig program. Når den er åpen, gir du den oppgaver direkte i grensesnittet. NB! Per nå, februar 2026: `Cowork` krever Pro-abonnement på Mac. På Windows kreves Max-abonnement.

CLI `Claude Code` krever litt mer oppsett. Det kjører i terminalen, som er et tekstbasert verktøy innebygd i datamaskinen din.

Slik åpner du terminalen:

CLI Mac: Trykk `Cmd + Mellomrom`, skriv «Terminal» og trykk `Enter`.

CLI Windows: Trykk `Windows-tasten`, skriv «PowerShell» og trykk `Enter`.

CLI Linux: Trykk `Ctrl + Alt + T`.

Se kapittel 2 om hvordan installerer `Claude` gjennom terminalen.

Claude Code

Kapittel 1: Abonnement, modeller og hva du faktisk betaler for

Før du installerer noe er det lurt å forstå hva du egentlig betaler for og hvilken modell som passer deg. Det sparer forvirring og unødvendige utgifter.

Abonnementstyper

Claude Code kan brukes på to måter:

- **Claude Pro/Max (abonnement):** Du betaler en fast månedspris og får et visst antall meldinger og Claude Code-bruk inkludert. Enklest å komme i gang med ingen overraskelser på fakturaen.
- **API-nøkkel (betaling per bruk):** Du betaler for det du bruker, målt i tokens (tekststykker). Kan bli billigere for lav bruk, men kan også bli dyrt hvis du setter i gang store operasjoner uten å passe på.
- **WEB** Priser for de ulike abonnementene finnes [her](#).

Tips: Anbefaling for nybegynnere Start med Pro-abonnementet. Du slipper å tenke på kostnad per melding og kan bare utforske. Gå over til API-nøkkel hvis du etterhvert vet hva du gjør og vil optimalisere kostnad.

Opus, Sonnet og Haiku hva er forskjellen?

Claude finnes i tre varianter med ulike styrker. Tenk på dem som tre ansatte med ulike profiler:

- **Haiku** den raske. Svarer umiddelbart, bruker lite ressurser, koster minst. Perfekt for enkle spørsmål, små oppgaver og rask skrivning. Ikke den smarteste, men langt fra dum.
- **Sonnet** er balansen. Det er denne du bruker mesteparten av tiden. Smart nok for de fleste oppgaver, rask nok til å ikke irritere deg, og rimelig nok til å ikke sprengre budsjettet. Standard i Claude Code.
- **Opus** den grundige. Bruker mer tid, koster mer, men tenker seg bedre gjennom komplekse problemer. Bruk den når Sonnet gir deg svar du ikke stoler på, eller når oppgaven er særlig viktig.

Analogi: Haiku er vikaren som fikser enkle ting raskt. Sonnet er den erfarne kollegaen du stoler på i hverdagen. Opus er spesialisten du kaller inn når noe er vanskelig.

Kapittel 2: Installasjon og første steg

Claude Code er et program du kjører i terminalen. Terminalen er det svarte eller hvite vinduet der du skriver kommandoer i stedet for å klikke med musa. Høres skummelt ut, men du trenger egentlig bare å lære deg noen få kommandoer for å komme langt.

Installasjon

For å installere Claude Code trenger du først **Node.js** på maskinen. Node.js er en plattform som lar datamaskinen kjøre JavaScript-programmer utenfor nettleseren. Du trenger ikke forstå hva det betyr i praksis, men mange utviklerværktøy, inkludert Claude Code, er bygget på toppen av det. Tenk på det som et fundament som må være på plass før du kan sette opp selve verktøyet. Last ned Node.js [her](#).

CLI Ikke sikker på om du har Node.js? Skriv dette i terminalen (NB! vær nøye med mellomrom og riktige tegn):

```
node --version
```

Får du et versjonsnummer tilbake, for eksempel «v20.11.0», er du klar. Får du en feilmelding, last ned Node.js fra nodejs.org. Velg versjonen merket «LTS», som betyr langtidstøtte og er den mest stabile.

Når Node.js er installert, installerer du Claude Code med én linje:

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

Her er «npm» en pakkebehandler som følger med Node.js og «-g» betyr at programmet installeres globalt, det vil si at du kan bruke det fra hvilken som helst mappe på maskinen. Uten «-g» ville programmet bare fungert i den mappen du sto i da du installerte det.

Øvelse 1: Sjekk hva du bruker

1. Start Claude Code ved å skrive **claude**.
2. Skriv: **!status** og du ser hvilken modell du bruker og annen sesjonsinformasjon.
3. Skriv: **!usage** og du ser tokenforbruk for gjeldende sesjon.
4. Skriv: **!stats** og du ser all informasjon om Claude **CLI**.
5. Skriv: **!model** og du ser tilgjengelige modeller og kan bytte om du vil.

Hva skjer egentlig når du starter Claude Code?

Når du kjører **CLI** Claude i en mappe, leser Claude Code filene og mappestrukturen rundt seg. Den danner seg et bilde av hva slags prosjekt det er i mappa den er åpnet i. Nesten som en ny kollega som bruker litt tid på å sette seg inn i situasjonen før de begynner å jobbe.

Det betyr at du ofte kan bare beskrive hva du vil ha, og Claude forstår konteksten uten at du trenger å forklare alt fra bunnen av.

I bildet under ser du hvordan arbeidsvinduet ser ut når du starter **CLI** Claude Code i terminalen.



Øvelse 2: Start terminalen

1. Åpne terminalen
2. Naviger til en mappe med noen filer. Eksempel: `cd Downloads`
3. Skriv: `claude`
4. Logg inn når du blir bedt om det.
5. Skriv: «Hva slags filer finnes i denne mappen?»
6. Claude leser mappen automatisk og forteller deg hva den ser.

CLAUDE.md er hjelpenotatet til Claude

Tenk deg at du har en assistent som begynner på null hver morgen. Ingen hukommelse. Vet ikke hvem du er eller hva prosjektet dreier seg om. Det er frustrerende og det er nøyaktig slik Claude Code fungerer som standard.

Analogi: CLAUDE.md er som en lapp du legger på pulen til assistenten hver morgen: Her er det du trenger å vite. Disse reglene gjelder alltid. Disse filene er viktige. Når du åpner Claude Code i en mappe med en CLAUDE.md-fil, leser Claude den automatisk og husker alt som står der

CLAUDE.md løser dette. Det er en vanlig tekstfil som ligger i prosjektmappen din, og som Claude leser automatisk hver gang du åpner Claude Code i den mappen.

Det kan stå hva som helst i filen: hva prosjektet handler om, hvilke kodefiler som er viktigst, hvilken skrivestil du foretrekker, hvilke verktøy du bruker og regler Claude skal følge. Jo mer relevant informasjon du legger inn, jo mindre må du forklare på nytt hver gang.

Du lager CLAUDE.md ved å skrive dette inne i **CLI** Claude Code:

```
/init
```

Da genererer Claude et utkast basert på innholdet i mappen. Åpne filen i en teksteditor og fyll inn det du vil at Claude alltid skal vite.

CLI Start enkelt. Du trenger ikke fylle inn alt på en gang. Legg til ting i CLAUDE.md etter hvert som du oppdager at du gjentar de samme forklaringene til Claude.

Øvelse 3: Lag en CLAUDE.md

1. Åpne Claude Code i en mappe du jobber med.
2. Skriv: `/init`
3. Åpne den nye CLAUDE.md-filen i en teksteditor.
4. Legg til en setning om hva mappen eller prosjektet handler om.
5. Lagre filen, avslutt Claude Code med `/exit`, og start den igjen.
6. Skriv: «Hva vet du om dette prosjektet?»

Legg merke til at Claude nå svarer med akkurat det du skrev inn. Du har gitt Claude hukommelse.

Editor-integrasjon

Mange foretrekker å skrive kode i en editor fremfor å jobbe rent i terminalen. Claude Code har offisielle integrasjoner for de vanligste editorene, slik at du kan ha Claude tilgjengelig direkte i arbeidsvinduet ditt.

Med integrasjonen kan du markere et stykke kode, sende det til Claude og få svar uten å bytte vindu. Du kan også be Claude forklare, forbedre eller feilsøke koden mens du jobber.

Slik installerer du:

CLI VS Code: Åpne Extensions-panelet, søk etter «Claude Code» og installer.

CLI JetBrains (IntelliJ, PyCharm og lignende): Åpne innstillinger, gå til Plugins, søk etter «Claude Code» i JetBrains Marketplace og installer.

CLI Andre editorer som Cursor og Antigravity: Se Anthropic's dokumentasjon for oppdatert liste over støttede editorer.

CLI Editor-integrasjonen krever at Claude Code er installert globalt (-g) på maskinen først.

Øvelse 4: Koble Claude Code til editoren din

1. Åpne VS Code eller JetBrains-editoren din.
2. Installer Claude Code-tillegget slik beskrevet over.
3. Åpne en fil med kode eller tekst.
4. Marker noen linjer og send dem til Claude med spørsmålet: «Hva gjør denne koden?»
5. Legg merke til at du får svar direkte i editoren uten å åpne terminalen.

Hvis du ikke bruker VS Code, JetBrains, kan du hoppe over denne øvelsen og fortsette med terminalen.

Kapittel 3: Kommandoene og snarveiene du trenger

Claude Code har innebygde kommandoer som starter med skråstrek. Du trenger ikke lære alle, men her er de du vil bruke mest.

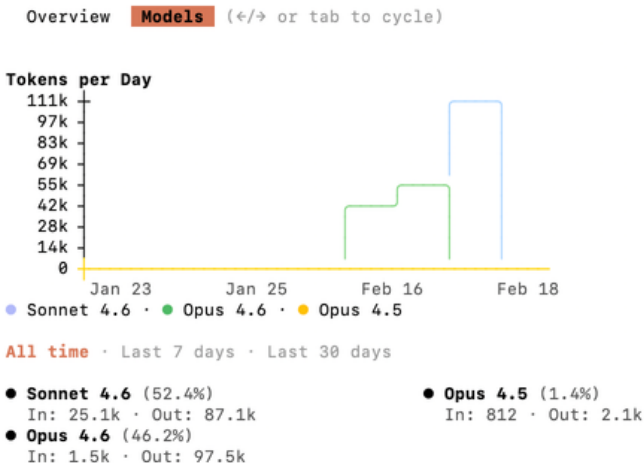
CLI Skråstrekskommandoer

- **/init** Lag CLAUDE.md for mappen din. Kjør denne når du starter et nytt prosjekt.
- **/clear** Tøm hukommelsen. Claude glemmer alt fra gjeldende samtale og begynner friskt. Nyttig når du bytter til en helt annen oppgave.
- **/compact** Komprimer en lang samtale. Claude beholder det viktigste, men kaster detaljene.
- **/context** Se hva som bruker plass i kontekstvinduet akkurat nå.
- **/usage** Se tokenforbruk for gjeldende økt.
- **/stats** Se detaljert statistikk over gjeldende økt, inkludert kostnader og modellbruk.
- **/model** Bytt mellom ulike Claude-modeller.
- **/rewind** Tilbakestill kode og samtale til et tidligere punkt. Nyttig hvis Claude tok feil retning eller gjorde en endring du angrer på.
- **/config** Juster innstillinger for Claude Code.
- **/help** Vis alle tilgjengelige kommandoer.
- **/exit** Avslutt Claude Code.

Husker du bare to kommandoer, la det være `/init` og `/help`. De løser de fleste situasjonene der noe går galt.

Under ser du hva som skjer når du taster inn kommandoen

`/stats`. Da viser den hvilke modeller du har brukt, hvor mye på hver og antall tokens per dag.



Hurtigtaster og prefiks

I tillegg til skråstrekkommandoer finnes det noen tastetrykk og prefiks som endrer hvordan **CLI** Claude tolker det du skriver.

Tastetrykk du kan bruke under en økt:

- **Shift+Tab** veksler mellom planleggingsmodus og redigeringsmodus. Mer om dette i kapittel 5.
- **Escape** stopper en pågående prosess umiddelbart.
- **Ctrl+R** åpner et søk i tidligere meldinger du har sendt i gjeldende økt. Nyttig når du vil finne og gjenbruke en instruksjon du har skrevet før.

Prefiks er tegn du skriver helt først i meldingen for å gi Claude en ekstra instruksjon om hvordan den skal tolkes.

- **!** kjører kommandoen direkte i terminalen uten å involvere Claude. Eksempel: `!!s` viser innholdet i mappen med en gang.
- **@** refererer til en konkret fil. Eksempel: `@rapport.txt` hva handler denne filen om? Da vet Claude nøyaktig hvilken fil du mener, uten at du trenger å forklare hvor den ligger.
- **/** sender meldingen som vanlig tekst selv om den begynner med **/**. Nyttig hvis meldingen din tilfeldigvis starter med en skråstrek og du ikke vil at Claude skal tolke det som en kommando.

Øvelse 5: Utforsk kommandoer og prefiks

1. Start Claude Code i en mappe med noen filer.
2. Skriv: `!!s` (eller `!dir` på Windows). Legg merke til at filene vises direkte uten at Claude er involvert.
3. Skriv: `@filnavn.txt` hva handler denne filen om? Bytt ut filnavn.txt med en fil som faktisk finnes i mappen.
4. Skriv: `/help` og se listen over tilgjengelige kommandoer.
5. Skriv noe til Claude, og bruk deretter `/rewind` for å se hva som skjer med samtalen.

Du har nå det du trenger for å navigere Claude Code. Kommandoer, hurtigtaster og prefiks er verktøyene du vil bruke om og om igjen. De fleste lærer dem raskt fordi de dukker opp naturlig i arbeidet.

Kapittel 4: Slik holder du Claude fokusert

Hva er kontekst og hvorfor betyr det noe?

Kontekst er det **CLI** Claude husker og jobber ut fra i en gitt samtale. Jo mer du legger til spørsmål, filer og svar, jo mer fullt blir det. Og akkurat som et menneske som har fått altfor mange ting å holde styr på, kan Claude begynne å gjøre merkelige valg når konteksten blir for overlastet. I praksis kan det bety at Claude glemmer instruksjoner du ga tidlig i samtalen, motsier seg selv eller slutter å følge stilen du ba om.

Analogi: Tenk deg at du skal forklare noe til en venn, men pakker inn forklaringen i tju andre historier og sidetema. Til slutt husker ikke vennen hva poenget var. Samme greie med Claude. Kortere og mer fokusert kontekst gir bedre svar.

Det er tre ting som fyller opp konteksten:

- Alt du har skrevet i samtalen
- Alt Claude har svart
- Filer Claude har lest underveis

Praktiske tips

Jobb med en oppgave om gangen og fullfør den før du starter på neste. Det gir mye bedre resultater enn å stable oppgave på oppgave i samme samtale.

/clear Start en ny samtale med **/clear** når du bytter til en ny oppgave. Ikke dra med deg gammel bagasje.

/compact Bruk **/compact** når samtalen har blitt veldig lang og du ikke er ferdig ennå.

NB! Hold **CLAUDE.md** oppdatert. Det er din permanente huskelapp og den eneste konteksten som overlever **/clear**.

Når noe går galt

Noen ganger gjør **CLI** Claude noe annet enn du mente. Kanskje var instruksjonene dine uklare. Her er en enkel løkke for å komme videre:

- Trykk **Escape** for å stoppe hvis noe pågår
- Bruk **/rewind** for å spole tilbake til forut for feilen
- Les hva Claude har gjort og tenk over hva som var uklart i instruksjonen din
- Omformuler med mer detalj og prøv igjen

Vær konkret. I stedet for å skrive «fiks filen». Prøv heller en spørring som dette: «gå gjennom **todo.txt**, finn alle punkter merket med **VIKTIG**, og lag en ny fil kalt **viktig.txt** med bare disse punktene.» Jo mer konkret, jo bedre.

Øvelse 6: Test konteksteffekten

1. Åpne Claude Code og still spørsmål om to helt ulike ting i samme samtale.
2. Mot slutten, spør Claude: «Hva har vi snakket om i dag?»
3. Kjør `/clear` og spør det samme igjen.
4. Kjør `/resume` etterpå for å gjenopprette den avsluttede samtalen og se hva som kommer tilbake.

På bildet under kjørte jeg først `/clear`, deretter stilte jeg spørsmål om hva vi har jobbet med i dag.

hva har vi jobbet med i dag?

Dette er en ny samtale, så jeg har ikke oversikt over hva som ble gjort i tidligere sesjoner i dag. La meg sjekke git-historikk for å se om det er gjort noen endringer nylig.

```
• Bash(cd /Users/kennethbareksten/Koding/neuro-planner && git log --oneline --since="2026-02-21" 2>/dev/null || echo "Ingen git repo eller ingen commits i dag")
└─ (No output)
```

```
• Bash(find /Users/kennethbareksten/Koding -name "*.ts" -o -name "*.tsx" -o -name "*.js" -o -name "*.jsx" | xargs ls -lt 2>/dev/null | head -20)
└─ -rw-r--r-- 1 kennethbareksten staff 30300 21 des. 11:36 /Users/kennethbareksten/Koding/https-github.com-barx10-treningssappen/App.tsx
  -rw-r--r-- 1 kennethbareksten staff 3479 17 des. 19:01 /Users/kennethbareksten/Koding/https-github.com-barx10
  ... +26 lines (ctrl+o to expand)
```

Det ser ikke ut til at det er gjort noen endringer i kode i dag (21. feb 2026). Den siste aktiviteten jeg kan se er fra desember 2025.

Siden dette er en ny samtale har jeg ikke minne om hva vi eventuelt snakket om i en tidligere sesjon i dag. Husker du hva dere jobbet med, eller vil du fortsette med noe spesifikt?

Basert på minnefilen har vi tidligere jobbet med **Neuro Planner** - en PWA for nevrodivergente. Neste steg på listen er:

1. Deploy til Vercel
2. Push-notification backend (Edge Functions + Web Push)

Kapittel 5: Få mer ut av Claude Code

Planleggingsmodus

En av de nyttigste tingene i Claude Code er planleggingsmodus. Når du trykker Shift+Tab, bytter Claude fra å bare gjøre det du sier, til å først tenke høyt og legge frem en plan.

Claude beskriver hva den har tenkt å gjøre, steg for steg, og venter på at du godkjenne planen før noe faktisk skjer. Du kan lese, stille spørsmål og be om justeringer. Når du er fornøyd, trykker du Shift+Tab igjen for å sette i gang. Du kan også bruke kommandoen **/plan** for å be Claude lage en plan uten å bruke Shift+Tab:

```
/plan reorganiser filene i denne mappen etter tema
```

/plan og Shift+Tab gjør i praksis det samme, det setter Claude i planleggingsmodus før den handler. Forskjellen er at /plan lar deg skrive oppgaven og be om plan i én og samme melding.

CLI Bruk planleggingsmodus hver gang du er usikker på hva Claude vil gjøre. Det koster ingenting, og det sparer deg for mange feil. Det er lett å glemme og bare trykke Enter. Planleggingsmodus er en vane du må bygge deg bevisst.

CLI For enkle og korte oppgaver der resultatet er åpenbart, trenger du ikke planleggingsmodus. Bruk det når oppgaven har flere steg eller du er usikker på hva Claude faktisk kommer til å gjøre.

Øvelse 7: Bruk planleggingsmodus

1. Tenk ut en oppgave du vil ha Claude til å hjelpe med. Eksempel: «Lag en oversikt over alle .txt-filene i denne mappen»
2. Skriv meldingen, men trykk Shift+Tab før du trykker Enter. Du vil se at Claude nå er i planleggingsmodus – den svarer med en plan i stedet for å handle med en gang.
3. Les planen Claude legger frem.
4. Prøv å be om en justering: «Kan du også ta med størrelsen på filene?»
5. Trykk Shift+Tab igjen for å utføre den justerte planen

I bildet under kjørte jeg først **Shift + Tab** og så stilte spørsmålet mitt.

```
hvilke filer har vi endret sist?
```

```
Jobbet mye med blokkert tid / ukentlig skjema i settings! Hva vil du jobbe med nå?
```

```
Exit plan mode?
```

```
Claude wants to exit plan mode
```

```
> 1. Yes  
2. No
```

Kapittel 6: Skills gir Claude spesialkunnskap

Etter hvert som du bruker **CLI** Claude Code, vil du oppdage at du gjentar de samme forklaringene om og om igjen. Hva prosjektet handler om, hvilken stil du vil ha eller hva du vil at Claude alltid skal sjekke. Skills er løsningen på det.

En skill er en liten tekstfil du lagrer i prosjektet ditt. Den inneholder instruksjoner som Claude automatisk henter inn når de er relevante uten at du trenger å gjenta dem.

Analogi: Tenk på en skill som et kurs du sender assistenten på. Etter kurset kan assistenten gjøre en konkret oppgave uten at du trenger å forklare prosedyren hver gang.

Slik lager du en skill

En skill er en instruksjonsfil du lager én gang, og som Claude bruker automatisk når situasjonen tilsier det. Hver skill består av en mappe og en fil. Mappen gir skillen et navn, og filen inni, som alltid heter SKILL.md, inneholder instruksjonene.

Slik oppretter du en skill som lager oppsummeringer. Lag først mappestrukturen:

```
.claude/  
├── skills/  
    ├── oppsummering/  
        └── SKILL.md
```

Du lager mapper med kommandoen `mkdir`. For å lage hele mappestrukturen til en skill i én operasjon, skriv:

```
mkdir -p .claude/skills/oppsummering
```

Kommandoen **-p** betyr at alle mappene i kjeden opprettes på én gang, selv om de ikke finnes fra før. Uten **-p** ville du fått en feilmelding hvis `.claude` eller `skills` ikke allerede eksisterte. Deretter oppretter du SKILL.md-filen:

```
touch .claude/skills/oppsummering/SKILL.md
```

Åpne filen i en teksteditor (f.eks Teksteditor i MacOS/ Notepad i Windows, lim inn innholdet og lagre filen.

Filen har to deler. Øverst, mellom de to strekene, ligger metadata som forteller Claude hva skillen heter og når den skal brukes. Nedenfor strekene kommer selve instruksjonene Claude følger når skillen aktiveres.

Neste gang du starter Claude Code i denne mappen, leser den SKILL.md automatisk. Hvis du ber Claude oppsummere noe, vet den nøyaktig hvordan du vil ha det gjort. Du kan også aktivere en skill direkte ved å skrive **/oppsummering**.

Øvelse 8: Lage SKILL

1. Lag mappen `.claude/skills/oppsummering/` i prosjektet ditt.
2. Lag filen `SKILL.md` inne i mappen (se kommando over) med dette innholdet:

```
---  
description: Lag korte oppsummeringer av tekst og kode.  
triggers:  
- oppsummer  
- lag sammendrag  
---
```

Når du lager en oppsummering, bruk alltid dette formatet:

- Hva handler dette om? (en setning)
- Hva er de viktigste punktene? (maks tre)
- Hva bør man gjøre videre? (en anbefaling)

Nå kan du prøve ut. Skriv i **CLI** Claude Code: «Lag en oppsummering av dette prosjektet.»
Se at Claude bruker formatet fra skillen din automatisk.

Få Claude til å lage skills

Ikke sikker på hva du skal skrive i `SKILL.md`? Du kan bruke Claude på **WEB** `claude.ai` til å hjelpe deg. Beskriv hva du vil at skillen skal gjøre, og be Claude lage innholdet til filen for deg.

Du kan for eksempel be Claude lage en skill som alltid formaterer Word-dokumenter på en bestemt måte, med fast skriftstørrelse, bestemte overskriftsnivåer og en fast struktur for innholdet. Claude lager et ferdig forslag du kan kopiere direkte inn i `SKILL.md` i prosjektet du jobber med eller globalt

Anthropic og fellesskapet har laget mange ferdige skills som både kan brukes i nettsversjon eller i Claude Code. Se referanseliste bakerst.

Kapittel 7: Gi Claude en rolle med plugins

En skill gir **CLI** Claude én spesifikk ferdighet. En plugin gir Claude en hel rolle med alt som hører til: fagkunnskap, egne kommandoer, tilgang til eksterne verktøy og evnen til å jobbe med flere deloppgaver samtidig.

Analogi: En skill er som å sende Claude på et dagskurs. En plugin er som å ansette Claude i en fast stilling med stillingstittel, arbeidsverktøy, kollegaer å samarbeide med og rutiner for hvordan jobben skal gjøres.

En plugin for kundeservice gjør Claude til en kundeserviceagent som vet hvordan henvendelser skal håndteres, hvilke systemer som brukes og hvilke svar som passer i ulike situasjoner. En plugin for salg gjør Claude til en selger som kan forberede møter, hente kundeinformasjon og lage presentasjoner.

Plugins ble lansert i januar 2026 og er tilgjengelige i både Claude Code og Cowork.

Hva inneholder en plugin?

En plugin pakker fire ting sammen: skills med faglig kunnskap og prosedyrer for rollen, egne kommandoer som for eksempel `/salg:forbered-møte`, koblinger til eksterne verktøy som Slack, Notion eller kalender og evnen til å koordinere parallelle deloppgaver.

Koblingene til eksterne verktøy skjer gjennom MCP (Model Context Protocol), et standardisert system som lar Claude sende og hente data fra andre programmer. Du trenger ikke forstå hvordan det fungerer teknisk, det holder å vite at det er derfor Claude kan hente en kalenderavtale eller oppdatere et regneark uten at du kopierer og limer inn noe for hånd.

Slik installerer du en plugin

Du installerer plugins direkte inne i **CLI** Claude Code med `/plugin`-kommandoen. Skriv `/plugin` for å åpne menyen, der du kan bla gjennom tilgjengelige plugins, installere nye og administrere de du allerede har.

I menyen kan du oppdage nye plugins, se hvilke du installert fra før og sjekke ut et "supermarked med ulike plugins med ulike roller (Marketplaces). Når pluginen er installert, er de tilhørende kommandoene tilgjengelige med en gang. Du trenger å starte Claude Code på nytt for at plugin skal virke.

Under ser du et skjermbilde av plugin-menyen og hvilke jeg har installert:

```
/plugin
Plugins Discover Installed Marketplaces (+/- or tab to cycle)
# Search...

User
> frontend-design Plugin · claude-plugins-official · ✓ enabled
superpowers Plugin · superpowers-marketplace · ✓ enabled
vercel Plugin · claude-plugins-official · ✓ enabled

claudeai
claude.ai Canva MCB · ✓ connected
claude.ai Hugging Face MCB · ✓ connected
claude.ai Learning Commons Knowledge Graph MCB · ✓ connected

type to search · Space to toggle · Enter to details · Esc to back
```

Plugin-Marketplaces

Marketplace-seksjonen i **/plugin**-menyen er der du finner plugins laget av andre. Her er plugins organisert etter kategori, slik at du raskt finner det du trenger uten å vite nøyaktig hva du leter etter.

Anthropic vedlikeholder sin egen offisielle samling, men markedsplassen inneholder også plugins fra fellesskapet. Kvaliteten varierer, så les beskrivelsen og sjekk hvem som har laget pluginen før du installerer. En plugin fra Anthropic selv er tryggere enn en ukjent bidragsyter.

Når du har funnet en plugin du vil prøve, installerer du den med noen tastetrykk direkte i menyen. Restart Claude Code.

Øvelse 9: Installer en offisiell plugin fra Anthropic

1. Skriv `/plugin` i terminalen for å åpne plugin-menyen.
2. Naviger til "Discover"-fanen med Tab-tasten.
3. Bla gjennom listen over tilgjengelige plugins fra Anthropic offisielle samling.
4. Velg commit-commands, som legger til nyttige kommandoer for å skrive gode git-meldinger.
5. Velg omfang: "User" gjør pluginen tilgjengelig i alle prosjekter dine.
6. Start Claude Code på nytt for at pluginen skal aktiveres.

commit-commands er et godt første valg fordi den er enkel, laget av Anthropic, og nyttig uansett hva du jobber med.

Claude Cowork

Kapittel 8: Hva er Claude Cowork?

COWORK er et verktøy i Claude-skrivebordsappen. Du åpner det, peker det mot en mappe på maskinen din, og ber Claude om hjelp med filene der. Claude leser, analyserer og lager nye dokumenter alt uten at du trenger å kopiere inn tekst i et chatvindu.

Cowork er laget for folk som ikke vil bruke terminal. Du trenger ikke installere noe ekstra. Åpne appen, velg en mappe og beskriv oppgaven du ønsker utført.

Hva skiller Cowork fra vanlig chat?

I vanlig chat limer du inn tekst og Claude svarer. I Cowork har Claude direkte tilgang til filene i mappen du velger. Claude kan lese ti filer på en gang, sammenstille informasjon fra alle, og levere et ferdig dokument uten at du løfter en finger underveis.

Viktig: Cowork har kun tilgang til mappen du bestemmer. Vær nøye på å ikke ha sensitiv informasjon som passord, private kontrakter eller personopplysninger i Cowork-mappen.

Analogi: I en vanlig chat er det du som er budbringer mellom Claude og filene dine. I Cowork sitter Claude direkte i arkivet og kan jobbe selvstendig.

Typiske oppgaver Cowork løser bra

Den kan:

- Gå gjennom alle Word-filene i denne mappen og lag ett samlet sammendrag
- Sorter bildene i denne mappen i undermapper etter dato
- Les regnearkene og skriv en rapport med de viktigste tallene
- Gi alle filene her et fornuftig navn basert på innholdet
- Finn alle PDF-ene som inneholder ordet faktura og flytt dem til en ny mappe

Nedlastning

Cowork er tilgjengelig for betalte abonnementer (Pro, Max, Team, Enterprise) på disse plattformene:

- Claude Desktop for macOS
- Claude Desktop for Windows (kun x64) Windows-brukere: Cowork krever den nyeste versjonen av Claude for Windows. Windows arm64 støttes ikke.

Lenke til nedlasting er [her](#)

Kapittel 9: Skills i Claude Cowork?

COWORK har allerede innebygde skills for de vanligste filformatene: Word-dokumenter (lese, skrive, formatere), Excel-regneark (lese data, analysere, opprette nye ark), PowerPoint-presentasjoner (lese og lage slides) og PDF-filer (lese, slå sammen, splitte, fylle inn skjemaer).

Når du ber Cowork om å lage et Word-dokument, aktiverer den Word-skillen automatisk. Du trenger ikke gjøre noe, det skjer av seg selv. Vil du styre det selv, kan du aktivere en skill manuelt ved å skrive for eksempel /word eller /excel i chatfeltet.

Øvelse 10: Skills i praksis

1. Legg noen tekstfiler eller Word-dokumenter i Cowork-mappen.
2. Skriv: «Les dokumentene i mappen min.»
3. Når Cowork bekrefter at den har lest dem, skriv: «Lag en PowerPoint-presentasjon med de viktigste punktene fra hvert dokument.»
4. Åpne den genererte filen og sjekk resultatet.

Plugins i Cowork

Plugins fungerer i **COWORK** akkurat som i Claude Code, men aktiveres fra chatfeltet i stedet for terminalen. En plugin kan gi Cowork tilgang til eksterne tjenester som Notion, Slack eller et CRM-system, og la Claude hente og sende data dit og tilbake, alt fra samme arbeidsøkt.

Du legger til plugins via innstillingene i Cowork, ikke via en terminal. Når en plugin er installert, kan du bruke den ved å nevne tjenesten direkte i meldingen din, for eksempel: «Hent de siste oppgavene fra Notion og lag et sammendrag i et Word-dokument.» Cowork skjønner at den må bruke Notion-plugininen og Word-skillen i kombinasjon.

Dette er der Cowork skiller seg fra vanlig chat. I stedet for å kopiere informasjon fra Notion, lime den inn i Claude og laste ned et dokument manuelt, gjør Cowork hele flyten for deg i én operasjon. Jo flere plugins du har installert, jo mer kan du overlate til Cowork uten å byte mellom programmer selv.

Kombiner lokale filer og nett med Claude i Chrome

Hvis du installerer Chrome-tillegget **WEB** Claude i Chrome, kan Cowork også bruke nettleseren. Det åpner for oppgaver som kombinerer lokale filer med nettinhold: hent priser fra en nettside og lagre dem i et regneark, sammenstill informasjon fra ti nettsider i ett dokument, eller fyll inn et nettskjema med data fra en lokal fil.

Det som gjør denne kombinasjonen kraftig er at Claude ikke bare leser nettsider, den kan også klikke, fylle inn felt og navigere videre, akkurat som du ville gjort selv. Det betyr at du kan sette i gang en oppgave som krever både lokale filer og flere steg på nettet, og overlate hele jobben til Cowork mens du gjør noe annet.

Eksempel: Du har en produktliste i Excel og vil ha priser fra en nettbutikk. Skriv til Cowork: «Les produktlisten og søk opp prisen på hvert produkt hos <navn på nettbutikk>. Fyll inn prisene i en ny kolonne.» Cowork gjør resten.

NB! Claude in Chrome er et eget tillegg du installerer i Chrome-nettleseren. Velg innstillinger, utvidelser og søk på Claude. Når det er installert, oppdager Cowork det automatisk og kan begynne å bruke det.

Øvelse 11: Kombiner to kilder

1. Lag en kort tekstfil med noen spørsmål du vil ha svar på. Kall den spørsmål.txt.
2. Åpne Cowork og skriv: «Les spørsmålene i spørsmål.txt og søk opp svarene på nettet. Lag deretter et dokument med spørsmål og svar.» (Krever Claude in Chrome-tillegget.)
3. Se at Cowork leser filen din og henter informasjon fra nettet.
4. Sjekk det ferdige dokumentet.

Kapittel 10: Prioriter sikkerhet og gode vaner

Claude Code og Cowork handler på vegne av deg. De kan lese, endre, opprette og slette filer. Det er kraftig og nyttig, men det betyr også at du bør tenke deg litt om, akkurat som du ville gjort før du ba noen andre om å rydde i filene dine.

Les alltid planen

Både Claude Code og Cowork presenterer alltid en plan før de gjør noe. Bruk den. Les hva Claude har tenkt å gjøre og tenk gjennom om det stemmer med det du mente. En rask gjennomlesning tar ti sekunder og kan spare deg for å miste viktige filer.

Tenk på det slik: du ville ikke sendt noen for å rydde kontoret ditt uten å si hva som er søppel og hva som er viktige papirer. Planen er din sjanse til å klargjøre nettopp det.

Jobbe trygt i Claude Code

Claude Code kan lese, endre, slette filer og kjøre kommandoer i terminalen. Det gir stor frihet, men krever litt mer oppmerksomhet enn Cowork.

Bruk planleggingsmodus (**Shift+Tab** eller **/plan**) for oppgaver der noe kan slettes eller overskrives. Hvis noe gikk galt, bruker du **/rewind** for å spole tilbake til før feilen skjedde. Start gjerne med å teste instruksjonene på filer du ikke er avhengig av, og vær ekstra forsiktig med instruksjoner som inneholder ordene "slett", "fjern" eller "rydd opp".

Jobbe trygt i Claude Cowork

Cowork er begrenset til mappen du velger, noe som gjør den litt mer forutsigbar enn Claude Code. Likevel gjelder de samme grunnreglene.

Lag en dedikert arbeidsmappe for Cowork i stedet for å peke den mot hjemmemappen din. Ikke legg filer med passord, sensitive personopplysninger eller private kontrakter i Cowork-mappen. Les alltid planen før du godkjenner, og test gjerne på ufølsomme filer først.

Tips: Vær ekstra forsiktig med filer lastet ned fra ukjente kilder. Et skadelig dokument kan i teorien inneholde tekst som forsøker å lure Claude til å gjøre noe du ikke har bedt om. Det kalles prompt-injeksjon, som betyr at en fil kan inneholde skjulte instruksjoner til Claude.

Øvelse 12: Sikkerhetssimulering

1. Lag en testmappe med noen kopier av ufølsomme filer.
2. Gi Claude Code eller Cowork denne instruksjonen: «Gå gjennom mappen og slett alle filer som virker overflødige.»
3. Les planen nøye og se hva som vil bli slettet?
4. Avbryt (ikke godkjenn) og omformuler instruksjonen mer presist. Eksempel: «Finn filer som begynner med kopi_ og flytt dem til mappen arkiv.»
5. Legg merke til hvordan en mer presis instruksjon gir en mer forutsigbar plan.

Avslutning

Du har nå gått gjennom det meste av det jeg skulle ønske jeg visste da jeg begynte. Fra de første skrittene i terminalen til plugins, markedsplasser og sikkerhetsvaner, handler alt i denne boka om én ting: å gi deg kontroll over verktøyene, ikke omvendt.

Claude Code og Cowork er kraftige, men de er fortsatt verktøy. De gjør det du ber dem om, og det er derfor det viktigste du kan ta med deg fra denne boka ikke er en bestemt kommando eller en smart snarvei. Det er vanen med å lese planen, starte smått og bygge tillit gradvis.

Sikkerhet handler ikke om å være redd for teknologien. Det handler om å forstå hva som skjer, slik at du kan bruke den trygt og effektivt. En AI-assistent som handler på vegne av deg fortjener like mye oppmerksomhet som en kollega du ber om hjelp.

Feltet utvikler seg raskt. Nye plugins, nye kommandoer og nye arbeidsflyter vil dukke opp. Men grunnprinsippene du har lært her holder seg: forstå hva verktøyet gjør, gi det tydelige instruksjoner og sjekk alltid resultatet.

Nå er det din tur. Åpne terminalen, skriv claude og begynn med noe lite.

Hurtigreferanse

Innebygde kommandoer

<code>/add-dir</code>	Legg til flere arbeidsmapper
<code>/agents</code>	Administrer underagenter
<code>/bashes</code>	Se og administrer bakgrunnsoppgaver
<code>/bug</code>	Rapporter feil til Anthropic
<code>/clear</code>	Tom hukommelsen og start friskt
<code>/compact</code>	Komprimer lang samtale
<code>/config</code>	Åpne innstillinger
<code>/context</code>	Vis kontekstbruk som farget rutenett
<code>/cost</code>	Vis tokenforbruk og estimert kostnad
<code>/doctor</code>	Sjekk at installasjonen fungerer
<code>/exit</code>	Avslutt Claude Code
<code>/export</code>	Eksporert samtalen til fil
<code>/help</code>	Vis hjelp og tilgjengelige kommandoer
<code>/hooks</code>	Administrer hooks for verktøyshendelser
<code>/ide</code>	Administrer IDE-integrasjoner
<code>/init</code>	Opprett CLAUDE.md for prosjektet
<code>/login</code>	Bytt Anthropic-konto
<code>/logout</code>	Logg ut
<code>/mcp</code>	Administrer MCP-servertilkoblinger
<code>/memory</code>	Rediger CLAUDE.md-filer
<code>/model</code>	Bytt modell
<code>/output-style</code>	Endre outputstil
<code>/permissions</code>	Se eller endre tillatelser
<code>/plugin</code>	Administrer plugins
<code>/pr-comments</code>	Vis kommentarer på pull request
<code>/resume</code>	Fortsett en tidligere samtale
<code>/review</code>	Be om kodegjennomgang
<code>/rewind</code>	Spol tilbake samtale og kode
<code>/sandbox</code>	Aktiver isolert kjøringsmiljø

<code>/security-review</code>	Sikkerhetsgjennomgang av ventende endringer
<code>/status</code>	Vis versjon, modell og kontostatus
<code>/terminal-setup</code>	Installer Shift+Enter-binding
<code>/todos</code>	Vis gjeldende gjoremsliste
<code>/usage</code>	Vis plangrenser og ratebegrensningsstatus
<code>/vim</code>	Aktiver vim-modus

Hurtigtaster og prefiks

<code>Ctrl+C</code>	Avbryt gjeldende input eller generering
<code>Ctrl+D</code>	Avslutt Claude Code
<code>Ctrl+L</code>	Tom terminalskjermen (beholder historikk)
<code>Ctrl+O</code>	Vis detaljert verktøybruk
<code>Ctrl+R</code>	Sok bakover i kommandohistorikk
<code>Ctrl+B</code>	Kjør kommando i bakgrunnen
<code>Ctrl+V / Alt+V</code>	Lim inn bilde fra utklippstavlen
<code>Pil opp/ned</code>	Naviger i kommandohistorikk
<code>Esc + Esc</code>	Spol tilbake kode og samtale
<code>Tab</code>	Skru utvidet tenkning av og på
<code>Shift+Tab</code>	Veksle Auto-Accept, Plan og normal modus
<code>Option+Enter</code>	Ny linje i melding (macOS)
<code>\ + Enter</code>	Ny linje i melding (alle terminaler)
<code># foran melding</code>	Legg til i CLAUDE.md direkte
<code>! foran melding</code>	Kjør terminalkommando direkte
<code>@ foran navn</code>	Referer til fil med autofullstyring

Plugin-kommandoer

<code>/plugin</code>	Åpne plugin-menyen
<code>/plugin marketplace add eier/repo</code>	Legg til en markeds plass
<code>/plugin install navn@markeds plass</code>	Installer en plugin
<code>/plugin marketplace update</code>	Oppdater markeds plassen

Kilder

Offisiell dokumentasjon

- Anthropic. Claude Code Docs. <https://code.claude.com/docs>
- Anthropic. Skills — offisielt GitHub-repositorium. <https://github.com/anthropics/skills>
- Anthropic. The Complete Guide to Building Skills for Claude (PDF, 29. januar 2026). <https://resources.anthropic.com/hubfs/The-Complete-Guide-to-Building-Skill-for-Claude.pdf?hsLang=en>
- Anthropic. A Complete Guide to Building Skills for Claude (bloggversjon). <https://claude.com/blog/complete-guide-to-building-skills-for-claude>
- Anthropic. Official Claude Code Plugins. <https://github.com/anthropics/claude-plugins-official>
- Anthropic. Claude Cookbook — praktiske guider og kodeeksempler. <https://platform.claude.com/cookbook/>

Artikler og guider

- AI Blew My Mind. Claude Skills — 36 eksempler. <https://aiblewmymind.substack.com/p/claude-skills-36-examples>
- AI Blew My Mind. Claude Cowork Plugins Guide. <https://aiblewmymind.substack.com/p/claude-cowork-plugins-guide>
- Product Talk. How to Use Claude Code Features. <https://www.producttalk.org/how-to-use-claude-code-features/>
- DataCamp. How to Build Claude Code Plugins: A Step-by-Step Guide. <https://www.datacamp.com/tutorial/how-to-build-claude-code-plugins>
- Composio. Improving Your Coding Workflow with Claude Code Plugins. <https://composio.dev/blog/claude-code-plugin>

Ressurser og samlinger

- AI Templates. Skills og plugin-komponenter. <https://www.aitmpl.com/component/skill/docx>
- Claude Marketplaces. Plugin-oversikt. <https://claudemarketplaces.com>
- Script by AI. The Ultimate Claude Code Resource List (2026). <https://www.scriptbyai.com/claude-code-resource-list/>
- hesreallyhim. Awesome Claude Code. <https://github.com/hesreallyhim/awesome-claude-code>
- ComposioHQ. Awesome Claude Skills. <https://github.com/ComposioHQ/awesome-claude-skills>
- Jeremy Longshore. Claude Code Plugins + Skills (270+ plugins). <https://github.com/jeremylongshore/claude-code-plugins-plus-skills>

© 2026 KENNETH BAREKSTEN

Denne boka er lisensiert under Creative Commons (CC BY 4.0). Det betyr at du kan dele, bruke og tilpasse innholdet, også kommersielt, så lenge du krediterer forfatteren.

Kontakt: kenneth@laererliv.no
Nettside: laererliv.no