Claude Code Masterclass - Course Notes

Claude Code Masterclass - Course Notes

Everything from the 11 sessions on one page. Commands verified against the official Claude Code docs.

INSTALL

Pick one. macOS or Linux, official installer:

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

Homebrew:

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

npm, if you already have Node:

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

Windows PowerShell:

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

Check it worked:

```
claude --version
```

SIGN IN

Run claude to start it

/login - sign in to your Anthropic account

claude setup-token - long-lived token, only needed for scripts and CI

Free, Pro and Max differ in usage limits, not in the commands you get

FIRST SESSION

cd into a real project folder, then run claude

First question to ask - What is this codebase?

@src/auth.ts - point at one exact file

@src/ - point at a whole folder

Using @ means Claude reads what you pointed at instead of guessing

CORE COMMANDS

/help - list every command

/init - generate CLAUDE.md from your codebase

/clear - new conversation, empty context

/compact "focus on X" - summarize instead of wiping

/context - see what is filling the context window right now

/rewind - roll code and conversation back to a checkpoint

/memory - edit CLAUDE.md and auto memory

/usage - how much of your plan you have used

/plan - plan mode, explore without editing. Shift+Tab does the same

/model - switch the model

/resume - return to an earlier conversation

/btw - ask a side question that stays out of history

/effort - set how hard Claude thinks

/agents - manage subagents

/mcp - manage MCP servers

/doctor - diagnose a broken setup

Esc - interrupt Claude mid-action

CLI FLAGS

claude --continue - resume the last session

claude --resume - pick which session to resume

claude -p "prompt" - non-interactive, for scripts

--permission-mode - set the permission mode at launch

--output-format json - structured output you can parse

--allowedTools "Read,Grep" - restrict what it is allowed to do

PERMISSION MODES

Default - asks before every tool use. Use on a new project.

Auto - safe actions approved automatically. Use day to day.

Bypass - no prompts at all. Scripts and CI only.

You can allowlist specific commands so they never prompt again.

Bypass mode is a footgun. Convenient until it runs something destructive with nothing to stop it.

CLAUDE.MD AND MEMORY

/init - create it

./CLAUDE.md - rules for this project

~/.claude/CLAUDE.md - your rules, every project

CLAUDE.local.md - private, not committed

@docs/conventions.md - pull another file into it

Put in - commands, conventions, constraints, anything Claude keeps getting wrong

Leave out - anything Claude can already read from the code

Every line costs context on every request, so keep it short

THE CORE WORKFLOW

Explore, Plan, Code, Verify, Commit

Plan mode first for anything large. Shift+Tab.

Verify means a real signal - a passing test, a clean build, a working endpoint. Not Claude saying it is done.

Commit before starting the next task

CONTEXT MANAGEMENT

Run /context. Almost nobody does, which is why they think the model got dumb.

A long session going bad is context noise, not a worse model.

New task - /clear

Same task, running long - /compact "focus on X"

Send research to a subagent so findings come back instead of raw files

/rewind to recover when it goes wrong

SKILLS

Live in .claude/skills/your-skill-name/SKILL.md

Frontmatter needs name and description

The description is what decides when Claude reaches for it, so write it precisely

Invoke by hand with /your-skill-name

Body loads only when used, so long reference material costs almost nothing until needed

HOOKS

Configured in .claude/settings.json

PreToolUse - before a tool runs, can block it

PostToolUse - after it runs, good for auto-format and lint

UserPromptSubmit, SessionStart, SessionEnd, Stop

A hook always runs. A skill runs when Claude decides it is relevant. That is the whole difference.

SUBAGENTS

Defined in .claude/agents/your-agent.md

Own context window. Findings come back, raw files do not.

Use for research and wide searches across a big codebase

Do not use when you need the actual details in your main thread

Every subagent starts cold, so it re-reads context you already have

MCP

Add an HTTP server:

```
claude mcp add --transport http github https://api.github.com/mcp
```

Add a local one:

```
claude mcp add --transport stdio postgres -- postgres-mcp-server
```

/mcp - manage connections

Every connected server spends context before you type anything. Remove what you do not use.

A CLI tool is often cheaper than an MCP server. gh beats a GitHub server for most tasks.

MODELS, API AND SCRIPTING

/model - switch

claude -p "Explain this codebase" - non-interactive

cat error.log | claude -p "What caused this error?" - pipe into it

claude -p "List all API endpoints" --output-format json - parseable output

claude -p "Check for security issues" --allowedTools "Read,Grep" - restricted run

anthropic/claude-code-action - run it in GitHub Actions

CAPSTONE - THE JOB APPLICATION AGENT

Start with the project CLAUDE.md

Fetch jobs, then parse each into title, company, requirements, link

Resume tailoring as a skill - .claude/skills/tailor-resume/SKILL.md

A hook to keep the output tidy

A subagent for company research

Spreadsheet output through MCP

Break it on purpose, then recover, so you have seen recovery work

Host it headless on Modal so it runs every morning

PLUGINS

/plugin marketplace add <source>

/plugin install <name>

YOUR FIRST WEEK

Day 1 - install, /login, run claude in a real project, ask what is this codebase

Day 2 - /init, write your CLAUDE.md, then cut it in half

Day 3 - one real task through the full loop, plan mode first

Day 4 - watch /context, practise /clear against /compact

Day 5 - build one skill you will genuinely reuse

Day 6 - one hook and one subagent

Day 7 - connect one MCP server, then run something with claude -p

Watch the full course: https://www.youtube.com/@ssktechy

Want to implement AI and save hours in your workflow?

YouTube (Telugu): https://www.youtube.com/@ssktechy
Instagram (Telugu): https://www.instagram.com/ssktechy
YouTube (English): https://www.youtube.com/@saisantoshkumar-ssk
Instagram (English): https://www.instagram.com/ssktechy.ai