CLAUDE CODE CENTER OF EXCELLENCE

Force Multiplying Our People

Welcome to the Agentic Age. No question is bad. No topic is off limits. We are here for **you** — to help every engineer ship faster, think bigger, and build with confidence.

OPEN DOOR

Every question welcome, every skill level valued

ACCELERATE

10x your impact with agentic workflows

LEARN TOGETHER

Share patterns, avoid pitfalls, level up as a team

SHIP IT

Real results, real code, real production impact

— V2.1.140 — V2.1.144

What's New with Claude Code

✨

New Features

/model is now session-only; press d to set a new default

Plugin dependency enforcement: disable warns, enable chains deps

Fast mode upgraded to Opus 4.7 by default

MCP_TOOL_TIMEOUT now raises HTTP/SSE timeout (was capped at 60s)

🔌

MCP Fixes

Paginated tools/list responses fully consumed — no more silently dropped tools

POSIX shell params in MCP configs no longer falsely flagged as missing env vars

SVG/unsupported MIME images no longer break the conversation

`claude mcp list` now shows parse errors instead of empty list

🔧

Bedrock Fixes

awsCredentialExport now always runs — fixes cross-account auth

Bedrock/Vertex background side-queries now fall back to correct Haiku model

Opus 1M context fixed in /model picker for Bedrock users

Startup no longer hangs 75s when `api.anthropic.com` is unreachable

— SECURITY CORNER

Vulnerability Remediation 101

Fix pipeline vulnerabilities in 3 steps with Claude Code

No more ignoring those red 🔴 badges — let Claude do the heavy lifting

🔧

Step 1 — Gear Up

Make sure **glab** is installed and authenticated (`glab auth status`), **or** the **GitLab MCP server** is connected in Claude Code.

Either path gives Claude the access it needs to read your pipeline findings.

📋

Step 2 — Drop the URL

Paste your GitLab pipeline vulnerability URL directly into Claude Code and say:

"Remediate the vulnerabilities in my pipeline"

Claude fetches the findings via `glab` or GitLab MCP and starts patching deps.

📄

Step 3 — Ship the Fix

Ask Claude to **commit and show you the diff**. Review the changes, confirm, and push.

Claude creates a clean commit with traceability refs — no manual package-hunting required.

— ANNOUNCEMENT

New Claude Code Teams Hub

As Claude Code moves to GA, we're consolidating support into a dedicated **Claude Code** team on Microsoft Teams. The POC chat will be shutting down — please join the new channels below. Search **"Claude Code"** in Teams to find the team.

🏠

Claude Code CoE

One stop shop for all things Ally Claude Code. Links to resources, process guides, and support channels.

Main channel

🛒

Marketplace

For MCP integrations, skills, and agents. Plugin questions, installation help, and contribution guidance.

Recommended

🔌

MCP

Information and discussion specific to MCP servers — building, hosting, and connecting them.

🛠️

Platform Support

Installation support, troubleshooting questions, and incident tickets. Your first stop for issues.

🎓

Training Tips

Techniques, prompts, and patterns to get the most out of Claude Code. Learn from each other.

Recommended

☕

Watercooler

General chat — migrating the Claude Code Lobby from Slack to our official community. All topics welcome.

Recommended

🏆

Champions Channel

Private sub-channel reserved for Claude Code Champions program members.

Private • By invitation

POC chat is shutting down. Please migrate to the new Teams channels above. All future announcements, support, and community discussion will happen here.

OBSERVABILITY TERRACE

Observability Terrace

📅 SAVE THE DATE

Dynatrace Center of Practice

June 4

Join the Dynatrace community for best practices, demos, and a look at what's coming next.

— MARKETPLACE BAZAAR

Marketplace Bazaar

🤖

allyai v0.3.0

/allyai:scaffold now converts Claude Code skills to LangGraph agents

`get_model()` pattern + Claude `sonnet-4-6` defaults

ECS task def & IAM templates added to **/allyai:infra**

Security audit fixes in model_factory — *Adric Saxon*

🔧

terraform & forge

terraform v0.2.7: renamed, MCP call rewrites, advisory warnings

forge v6.1.1: macOS date compat fix, SAST suppressions

Both: username header added to MCP server requests

— *Nick Allison • Brady Mahar • Hemanth Kalathil*

🚧

Coming Soon

acu-battery: ACU usage in status bar — *Zack Wilson*

ally-slides v0.7.5: setup redesign, uv tool — *Chris Irving*

aws: 75-tool read-only AWS context server — *Austin Ahlijian*

forge /spec: formal spec generation — *Brady Mahar*

```
claude plugin marketplace add ally-claude-plugins-official
```

— THE EPIC METHOD — STEP 1 OF 4

E — Explore

Before writing a single line of code, ground Claude in context. The more it knows about *where* it is, the less it has to guess — and the fewer corrections you'll make later.

Map the Codebase

Ask Claude to read key files, trace a data flow, or summarize what a module does. Use **subagents** to explore in parallel — one for the frontend, one for the API, one for the DB layer.

Read the Context

Point Claude at your **CLAUDE.md**, relevant docs, and recent git history. Tell it what team conventions exist. The system prompt is not enough — give it the *why* behind the code.

Ask Before Assuming

Start with *"What do you know about this area of the codebase?"* or *"What would you need to understand before touching this?"* Catch gaps in context early, not mid-implementation.

📖 **Read the output** | 🔄 **Course-correct at every turn** | 💬 **Don't like the choices? Have a conversation with Claude.**

Prompt pattern: "Before we start, read [files/docs] and summarize what you understand about [problem area]. Tell me what you're uncertain about."

— THE EPIC METHOD — STEP 2 OF 4

P — Plan

Never skip straight to implementation. A plan written takes minutes. Discovering a wrong approach takes hours. Use Claude's plan mode to get alignment before a single file changes.

Enter Plan Mode

Use **EnterPlanMode** (or just ask Claude to plan first). It will explore, propose an approach, and surface tradeoffs — all before touching your code.

Review & Approve

Read the plan. Push back on anything wrong. The plan is your contract with Claude — what it will do and, critically, **what it won't touch**. Approve it explicitly before proceeding.

Small Plans, Small Surprises

Scope each plan to one logical change. Big plans drift. If the plan covers more than 3–4 files or 2 distinct behaviors, break it apart — each piece gets its own explore → plan cycle.

📖 **Read the output** | 🔄 **Course-correct at every turn** | 💬 **Don't like the choices? Have a conversation with Claude.**

Prompt pattern: "Don't write any code yet. First, tell me your plan for [task]. What files will you touch, what will you change, and what are the risks?"

— THE EPIC METHOD — STEP 3 OF 4

I — Implement

Execute the approved plan in small, testable increments. This is where Claude shines — but only if you stay in the loop. Don't let it run for 20 minutes unchecked.

TDD First

Write the **failing test first**, then ask Claude to make it pass. Tests define the contract — Claude fills in the code. You end up with proof the feature works, not just code that looks right.

🔄

Iterate in Chunks

Implement one piece at a time. After each chunk: run tests, review the diff, confirm it looks right. **Don't batch** 10 changes and review them all at the end — you'll miss things.

🛑

Course-Correct Early

If the implementation drifts from the plan, **stop and re-plan**. Don't let Claude continue building on a bad foundation. A mid-stream correction is cheaper than a revert.

📖 **Read the output** | 🔄 **Course-correct at every turn** | 💬 **Don't like the choices? Have a conversation with Claude.**

Prompt pattern: “Implement only [specific piece] from the plan. Stop after that and show me the diff before continuing.”

— THE EPIC METHOD — STEP 4 OF 4

C — Commit

The last mile matters. A clean commit tells the story of the change — for your reviewers, your future self, and your audit trail. Don't let Claude phone this part in.

Review the Diff

Before committing, ask Claude: **"Show me everything you changed."** Walk the diff yourself. You are accountable for what goes in — not Claude. Own the output.

🔗

Traceability Refs

Every commit message should reference the **Jira ticket and/or GitLab issue**. Claude knows your conventions from CLAUDE.md — remind it if needed. `OC-123: fix auth token expiry`

🚀

Small & Frequent

Commit after each logical piece, not at the end of a multi-hour session. **Small commits = easy reviews, easy reverts, easy bisects.** CI catches regressions sooner when diffs are tight.

📖 **Read the output** | 🔄 **Course-correct at every turn** | 💬 **Don't like the choices? Have a conversation with Claude.**

Full cycle: Explore → Plan → Implement → **Commit** → repeat for the next piece. EPIC isn't one-and-done — it's a loop.

↑ ↓ to navigate

— MARKETPLACE SPOTLIGHT

forge — EPIC on Autopilot

forge is the Ally Claude Code plugin that automates the full EPIC cycle for you. Describe a feature, and it plans, implements, then throws **14 independent reviewers** at your diff — none of whom have seen your code before. They find the stuff you stopped noticing three commits ago.

/explore

Maps the codebase, surfaces context

/plan

Structured plan, reviewed before any code

/implement

Executes the plan, tests included

/review → /ship

14-agent adversarial review, then commit

The 14-Reviewer Panel

Security, performance, testing, observability, style, correctness — each reviewer looks at your diff from a fresh perspective. Adversarial by design. Catches what a single review pass misses.

Get Started

```
claude plugin marketplace add ally-claude-plugins-official
claude plugin install forge
```

forge v6.1.1 — Nick Allison • Brady Mahar • Hemanth Kalathil

WHAT'S NEXT

Next Session

Upcoming Topic

Brief description of what the next COE session will cover.

Requests Welcome

Have a topic you'd like covered? A tool you want demonstrated? Let us know.

The COE is yours. Bring questions, bring challenges, bring ideas. We build together.