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

— APRIL 8, 2026

Terraform MCP Server & Ally-Controlled AI

How we teach Claude to think like an Ally engineer — using MCP servers, Claude rules, and the PCE pipeline to produce compliant, production-ready infrastructure every time.

MCP Server

Ally's private Terraform module registry, wired directly into Claude

Claude Rules

Guardrails that enforce Ally standards on every AI-generated line

PCE Pipeline

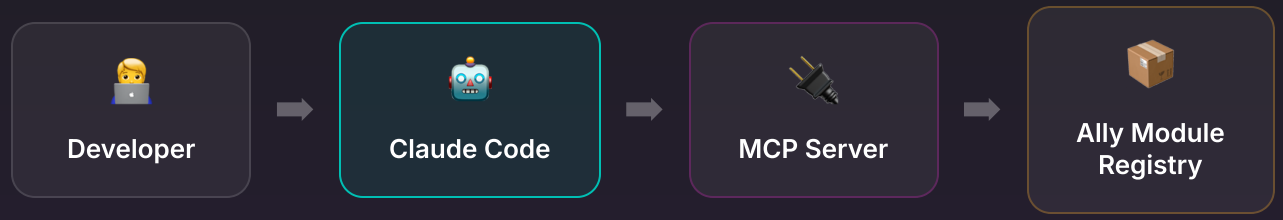
CI/CD that validates, scans, and deploys — the safety net

Ally Starter

Why our starter still beats raw Claude — even today

What is MCP?

Model Context Protocol — a standard that lets Claude call external tools in real time. Instead of guessing, Claude *asks* our systems for the right answer.



~/.mcp.json

```
{
  "mcpServers": {
    "terraform-docs": {
      "type": "http",
      "url": "https://mcp-pce-qa.7415.aws.ally.com/terraform/mcp"
    }
  }
}
```

3 Core Tools

1. **list_terraform_modules** — Discover all 65+ Ally modules and their versions
2. **list_terraform_submodules** — Drill into module structure (e.g., ECS service vs task)
3. **get_terraform_module_docs** — Full README with inputs, outputs, examples

Claude reads the **actual module docs** before writing a single line of Terraform. No hallucinated inputs. No wrong versions.

— MODULE REGISTRY

65+ Ally Terraform Modules

Every AWS service you need, pre-hardened with Ally security, tagging, and compliance baked in. Claude discovers these *at query time* via MCP.

`ecs-fargate` `lambda` `rds` `s3` `alb` `cloudfront` `dynamodb` `sqs` `sns` `ecr` `eks` `elasticache` `secretsmanager`
`parameter-store` `iam` `encryption` `kms` `api-gateway` `eventbridge` `step-functions` `glue` `msk` `mwaa` `batch` `apigee`

Private Registry Only

Source: **module-registry.services.ally.com/ally/**. Public registry modules are **FORBIDDEN**.

Latest Major Pin

Version: **~> 9.0** (latest major). Never pin minor versions like ~> 9.17.0.

No Raw Resources

If an Ally module exists for a service, Claude **must use it**. No raw `resource` blocks.

Teaching Claude the Ally Way

Rules files in `~/.claude/rules/` are injected into every conversation. They're not suggestions — they're hard constraints Claude cannot override.

~/.claude/rules/terraform.md

```
# Key Rules (abbreviated)

Source: module-registry.services.ally.com/ally/
Public registry FORBIDDEN

Module Discovery: MANDATORY before writing TF
  1. list_terraform_modules
  2. list_terraform_submodules
  3. get_terraform_module_docs

backend.tf: ALL 5 fields required (BLOCKING)
  bucket, key, region, dynamodb_table, encrypt

terraform.tfvars: ALL 9 vars required (BLOCKING)
  application_id, application_name, service_name
  owner, data_classification, issrcl_level
  scm_project, scm_repo, awsacct_cidr_code

Security Groups: Managed by modules.
  No aws_security_group resources.

Destroy: ALWAYS manual trigger, NEVER auto
```

What This Prevents

- ✕ Claude using `terraform-aws-modules/*` or `hashicorp/*` from the public registry
- ✕ Missing backend config that would create orphaned state files
- ✕ Skipping mandatory tagging variables that compliance needs
- ✕ Creating raw `aws_security_group` resources that bypass module controls
- ✕ Auto-destroy on pipeline failure — a production safety hazard

These rules fire on file globs. Any `*.tf` or `terraform.tfvars` file triggers the full ruleset. Claude literally cannot write Terraform without reading these first.

— SAFETY CONTROLS

Guardrails Beyond Terraform

CLAUDE.md is loaded into every single conversation. These are the global rules that keep Claude safe across all operations.

🚨

AWS CLI — Hard Rule

Read-only: always OK (describe, get, list). **Write commands require explicit approval in THAT message.** No carryover. Workflow: diagnose → present fix → wait for "do it" → execute.

🔐

Secrets — Never Display

NEVER output tokens, API keys, passwords, `.env` contents, `AKIA*` values. NEVER run `env` or `printenv`. Check existence only: **`[ -n "$VAR" ] && echo "set"`**

🧪

TDD Strategy

Always test-driven: **write failing tests first**, implement minimum code, refactor green. For bugs: reproduce in a test before fixing.

🐋

Docker in CI

NEVER pass tarballs as GitLab artifacts (>1GB). Use Kaniko `--tarPath` + `--no-push` for images known <1GB. Fix by slimming, not compressing.

💾

DB Migrations

Always in `central-core-db` repo. Always both up+down. FK with CASCADE, indexes, update triggers. **NEVER in service repos.**

🔗

Jira + GitLab

One ticket per repo. Branch: `feature/<PROJECT>-<NUMBER>`. MR format enforced: Summary → Changes → Test Plan with ## headings.

The PCE Pipeline Safety Net

Even if AI writes perfect Terraform, the pipeline validates it. Claude knows these rules too — it generates CI configs that comply from the start.

What Claude Knows About the Pipeline

- ✓ Dev and Prod are **separate pipelines** — prod NEVER rebuilds images
- ✓ 5 mandatory scans: SAST, dependency, secret detection, code quality, container
- ✓ OIDC auth — never combine `.aws-oidc-assume-role` with `.global-before-script`
- ✓ Runner tags are exact strings: `Innovation-federated-docker`, not approximations
- ✓ All images from `registry.services.ally.com/` — never Docker Hub
- ✓ Destroy is always `when: manual` — never auto-triggered

13 Common Mistakes It Avoids

1. Missing `.secret-analyzer` on secret_detection
2. OIDC + before_script collision
3. Prod pipeline including workflow rules
4. Rebuilding images in prod
5. Missing Kaniko `entrypoint: [""]`
6. `AWS_ENVIRONMENT` set globally
7. Docker-in-Docker instead of Kaniko

Three Layers of Control

MCP, Rules, and Pipeline work together. Each layer catches what the others miss.

🔌

Layer 1: MCP Server

What it does: Gives Claude real-time access to Ally's module registry. Claude reads actual docs, not cached training data.

What it catches: Wrong module names, deprecated versions, missing inputs, hallucinated parameters.

📋

Layer 2: Claude Rules

What it does: Injects Ally-specific constraints into every conversation. Fires automatically on file type.

What it catches: Public modules, missing backend/tfvars, write commands without approval, secrets exposure.

🛠️

Layer 3: PCE Pipeline

What it does: Validates everything at merge time. Security scans, Terraform plan, manual approval gates.

What it catches: Anything that slipped through — vulnerable deps, misconfigured resources, unapproved changes.

How Claude writes Terraform for Ally

```
# Step 1: Claude reads rules/terraform.md (automatic on *.tf glob)
# Step 2: Claude calls MCP to discover modules
list_terraform_modules()          # What modules exist?
list_terraform_submodules("ecs-fargate")  # What submodules?
get_terraform_module_docs("ecs-fargate", "9.18.0")  # Full README

# Step 3: Claude generates compliant TF using real inputs from docs
# Step 4: Pipeline validates on push — 5 scans + terraform plan
# Step 5: Manual approval for prod deploy
```

— THE CASE FOR THE STARTER

Why the Ally Starter Still Wins

Claude + MCP + Rules is powerful. But the Terraform Starter represents **years** of accumulated institutional knowledge that no AI can replicate from scratch.

CAPABILITY	RAW CLAUDE (NO RULES)	CLAUDE + MCP + RULES	ALLY TERRAFORM STARTER
Uses Ally private modules	✕ Uses public registry	✓ Via MCP	✓ Pre-configured
Correct backend.tf (5 fields)	✕ Generic or missing	✓ Rules enforce	✓ Template included
9 mandatory tfvars	✕ Doesn't know them	✓ Rules enforce	✓ Pre-filled template
Full CI/CD pipeline	✕ Generic YAML	~80% — knows rules	✓ Battle-tested YAML
Prod pipeline (separate)	✕ Doesn't know pattern	~70% — complex rules	✓ .prod-gitlab-ci.yml included
OIDC auth wiring	✕ No context	Knows rules, can miss edge cases	✓ Pre-wired correctly
Kaniko + Crane image flow	✕ Suggests docker build	✓ Rules guide it	✓ End-to-end working
Runner tag accuracy	✕ Guesses tag names	✓ Exact strings in rules	✓ Already correct
Time to first deploy	**Hours of debugging**	**~30 min with iteration**	**~5 min — clone & go**

— THE BOTTOM LINE

The Starter is Best in Class

No matter the effort we put into teaching Claude, a battle-tested scaffold still represents the fastest, safest path to production. Claude is the accelerant — the Starter is the launchpad.

65+	5 min	100%	0
ALLY MODULES	FIRST DEPLOY	COMPLIANT FROM DAY 1	PIPELINE DEBUGGING

Speed

Clone the starter, fill in your app details, push. Pipeline runs. Infrastructure deploys. **No iteration needed.**

Safety

Every decision — runner tags, scan configs, image flows, OIDC wiring — has been **validated in production** across dozens of teams.

Claude + Starter

The real power? **Use both.** Start from the Starter, then let Claude customize for your specific service. Best of both worlds.

The takeaway: Claude + MCP + Rules is how we ensure AI-generated Terraform is Ally-compliant. But the Starter is still faster than asking Claude to build it from scratch — even with perfect rules. **Use the Starter. Let Claude extend it.**

↑ ↓ to navigate

WHAT'S NEXT

Next Session

Your Topics Welcome

What do you want to see? A deep-dive on rules authoring? MCP server development? Agentic CI/CD? Claude Code tricks? **You decide.**

Try It Today

Add the Terraform MCP server to your `~/.mcp.json`. Copy the rules to your `~/.claude/rules/`. Watch Claude level up instantly.

The COE is yours. Bring questions, bring challenges, bring ideas. We build together. No question is bad. No topic is off limits.