AXIOMIC·AI ACADEMY

AI Pottery Class

Build with Claude Code — Fremont, Bay Area

COMPLETE BOOTCAMP SYLLABUS

24 Hours · 8 Sessions · 4 Weekends · $50/day

v2.0 — Now includes: Agent Harness Repo · MCP vs CLI Ecosystem · Personal Agent Declaration

Format	Saturday = Learn · Sunday = Build
Time	7:00 – 10:00 AM PST every weekend
Location	Fremont, Bay Area, California
Duration	4 weekends · 8 sessions · 24 hours total
Price	$50 per day · drop in · no lock-in
Stack	Next.js 14 · Supabase · Netlify · Claude Code · Google OAuth · Stripe
AI Engine	Claude Code CLI · claude-sonnet-4-20250514 · Axiomic Multi-Agent Framework
MCP Servers	GitHub MCP · Supabase MCP · Netlify MCP
Harness Repo	Personal agent harness repo (each student) — introduced Week 2, deployed Week 4
Deliverable	Live deployed web product with Google login + AI feature — yours to keep

FOUNDATIONAL CONCEPT

The Tools Ecosystem: MCP vs CLI

Before diving into the week-by-week curriculum, every student needs to understand the tools landscape. This is not just theory — knowing when to reach for MCP versus the CLI is a daily judgement call you will make as an AI builder. We cover this explicitly in Week 2, but the framing starts here.

What Are We Working With?

Claude Code CLI

The Claude Code CLI is your primary build tool. It is a terminal-based agentic coding assistant that understands your entire codebase, writes and edits files, runs commands, and iterates autonomously. You invoke it from your terminal. It has no concept of 'outside world' unless you give it tools.

```
# How you invoke Claude Code
$ claude                          # interactive session
$ claude 'build me a login page'  # single task
$ claude --continue               # resume last session
$ claude --model claude-sonnet-4-20250514 'refactor this'
```

MCP — Model Context Protocol

MCP is a standard protocol that lets Claude connect to external services through a structured server/client interface. Instead of Claude guessing how to call an API, an MCP server exposes tools that Claude can call natively — with type-safe inputs, real-time data, and two-way communication. Think of MCP servers as Claude's hands into external systems.

```
# MCP servers configured in .claude/settings.json
{
  "mcpServers": {
    "github":   { "command": "npx", "args":
["@modelcontextprotocol/server-github"] },
    "supabase": { "command": "npx", "args":
["@supabase/mcp-server-supabase"] },
    "netlify":  { "command": "npx", "args": ["@netlify/mcp-server"] }
  }
}
```

MCP vs CLI — Decision Framework

Dimension	Claude Code CLI	MCP Server Tool
What it is	Agentic coding assistant in terminal	Protocol bridge to an external service
Primary use	Write, edit, run, debug code	Read/write external systems (GitHub, Supabase, Netlify)
Invocation	$ claude 'do this'	Claude calls it automatically when relevant
Auth	Your terminal session / env vars	OAuth token or API key in MCP config
Speed	Fastest for local code changes	Adds network latency — use purposefully
Best for	Scaffolding, refactoring, debugging, building	Creating PRs, querying DB, triggering deploys
Examples	Build a login page, fix a bug, write tests	Push to GitHub, insert Supabase row, deploy to Netlify
When NOT to use	For actions outside your local codebase	For purely local code tasks — CLI is faster
Taught in	Week 1 (first contact)	Week 2 (MCP deep dive) + Week 3 (agent wiring)

The Mental Model

Think of it this way:

- CLI is your hands — it does the work on your machine
- MCP is your phone — it reaches out to the world
- Your agent harness repo is your brain — it remembers who you are, what tools you have, and how you like to work

Use Case	Right Tool	Why
Create a new React component	CLI	Pure local code task
Push your code to GitHub	MCP: GitHub	External system action
Create a Supabase table	MCP: Supabase	Reaches your hosted DB
Trigger a Netlify deploy	MCP: Netlify	External deploy system
Debug a failing API route	CLI	Local file + terminal
Check open PRs on your repo	MCP: GitHub	External data read
Run database migrations	MCP: Supabase	DB mutation
Write a multi-agent orchestrator	CLI + MCP	Code locally, wire externally

NEW IN V2.0

The Student Agent Harness Repo

Every student gets their own personal agent harness repository. This is not a tutorial repo — it is your operational AI environment. It defines who you are as an agent operator: your tools, your MCP servers, your personas, and your deployment targets. You build it across Weeks 2–4 and take it home when the bootcamp ends.

What Is the Agent Harness?

- A GitHub repository you own, named: axiomic-agent-harness-[yourname]
- Contains your Claude Code project settings, MCP server declarations, and agent system prompts
- Connected to YOUR GitHub, Supabase project, and Netlify account via MCP
- Includes a personal agent declaration — a system prompt that describes your build style, tech preferences, and product context
- Week 2: created and scaffolded · Week 3: MCP servers connected · Week 4: fully operational

Harness Repository Structure

```
axiomic-agent-harness-[yourname]/
├── .claude/
│   ├── settings.json          # MCP server declarations
│   └── CLAUDE.md              # Your personal agent context file
├── agents/
│   ├── orchestrator.md        # Your master orchestrator system prompt
│   ├── builder.md             # Code-writing subagent prompt
│   ├── reviewer.md            # Code review subagent prompt
│   └── deployer.md            # Deploy + smoke-test subagent prompt
├── mcp/
│   ├── github-config.json     # GitHub MCP server config
│   ├── supabase-config.json   # Supabase MCP server config
│   └── netlify-config.json    # Netlify MCP server config
├── projects/
│   └── [your-product]/
│       ├── spec.md            # Your product specification
│       ├── tasks.json         # Current task manifest
│       └── progress.md        # Build log
├── scripts/
│   ├── bootstrap.sh           # One-command environment setup
│   └── run-agent.sh           # Launch your agent swarm
└── README.md                  # Your agent identity card
```

The CLAUDE.md — Your Personal Agent Declaration

The most important file in your harness. Claude Code reads CLAUDE.md automatically at the start of every session. It is your agent's memory — your identity, your stack, your product context, your preferences. You write it in Week 2 and refine it every week.

```
# CLAUDE.md — Agent Declaration Template

## Who I Am
I am [Name], building [Product] — [one sentence description].
My product URL is [netlify-url] and my GitHub repo is [repo-url].

## My Stack
- Frontend: Next.js 14 (App Router), Tailwind CSS, TypeScript
- Auth: Supabase Auth with Google OAuth
- Database: Supabase (PostgreSQL)
- Deployment: Netlify (main branch auto-deploy)
- AI: Claude API, claude-sonnet-4-20250514

## My MCP Servers
- github:   push commits, create PRs, manage issues
- supabase: run SQL, manage tables, check RLS policies
- netlify:  trigger deploys, check build logs, manage env vars

## My Build Preferences
- Always write TypeScript, never plain JS
- Mobile-first responsive design
- Prefer server components unless interactivity needed
- Always add error boundaries and loading states

## Current Product Context
Product: [name] — [description]
Core AI feature: [describe the main thing Claude does in your app]
Current week focus: [what you're building this weekend]
```

WEEK 01 · FOUNDATION

AI First Principles + Claude Code First Contact

WEEK 1 SATURDAY — LEARN · 7:00 – 10:00 AM PST · 3 hrs

AI 101 + Claude Code First Contact

SESSION PLAN

- What is AI, what is an LLM — honest framing, no hype (30 min)
- What is Claude Code and how it differs from ChatGPT and Copilot (20 min)
- The tools landscape overview: CLI vs MCP vs API — mental model planted (20 min)
- Environment setup: Node.js, npm, VS Code, Claude Code CLI install, API key (40 min)
- Your first Claude Code command — live, everyone follows in terminal (20 min)
- Anatomy of a prompt: system context, user message, CLAUDE.md concept introduced (20 min)
- Q&A + troubleshooting corner — no one leaves with a broken setup (30 min)

TOOLS THIS SESSION

- Claude Code CLI — primary tool this session
- VS Code — editor setup
- Node.js / npm — runtime
- Terminal / bash — you will live here

✓ **SESSION DELIVERABLE:** Dev environment working · First Claude Code response in terminal · CLI understood

WEEK 1 SUNDAY — BUILD · 7:00 – 10:00 AM PST · 3 hrs

Build Your First AI-Powered Web Page

SESSION PLAN

- Project kick-off: every student pitches their product idea — 90 seconds each (20 min)
- Scaffold a Next.js 14 app using Claude Code slash commands (30 min)
- Add a simple AI feature: text generation triggered by a button (40 min)
- Connect Supabase: create project, env vars, first table via Supabase dashboard (40 min)
- Deploy to Netlify — live URL generated from scratch, auto-deploy on push (20 min)
- Share your URL in the group → peer feedback round (30 min)

TOOLS THIS SESSION

- Claude Code CLI — scaffold, code, debug
- Netlify UI — first deploy
- Supabase dashboard — project creation (CLI/MCP introduced Week 2)
- GitHub — repo init and push (manual this week, MCP Week 2+)

✓ **SESSION DELIVERABLE:** Live URL · AI feature running · Supabase connected · GitHub repo created

WEEK 02 · ECOSYSTEM

Claude Code Deep Dive · Skills · MCP · Agent Harness

WEEK 2 SATURDAY — LEARN · 7:00 – 10:00 AM PST · 3 hrs

The Claude Ecosystem + MCP Deep Dive + Harness Setup

SESSION PLAN

- Claude Code ecosystem: projects, memory, context windows, CLAUDE.md in depth (25 min)
- What are Skills — reusable prompt modules, when and how to write them (20 min)
- MCP explained in full: protocol, server/client model, why it beats raw API calls (25 min)
- Live: connect GitHub MCP server — push a commit from inside Claude Code (25 min)
- Live: connect Supabase MCP server — query your DB via Claude in terminal (25 min)
- CLI vs MCP decision framework — class discussion with real examples (20 min)

TOOLS THIS SESSION

- Claude Code CLI — primary session tool
- MCP: GitHub server — first MCP experience
- MCP: Supabase server — DB queries from Claude
- npx @modelcontextprotocol/server-github — MCP install pattern

✓ **SESSION DELIVERABLE:** GitHub MCP + Supabase MCP connected · First DB query via Claude · CLI vs MCP understood

WEEK 2 SUNDAY — BUILD · 7:00 – 10:00 AM PST · 3 hrs

Build: Google Auth + Agent Harness Repo Created

SESSION PLAN

- Implement Google OAuth via Supabase Auth — end-to-end working login (45 min)
- Row-level security in Supabase: your data vs everyone's data (25 min)
- Build a CRUD feature using Claude Code — data scoped to logged-in user (30 min)
- HARNESS REPO: fork template, personalise CLAUDE.md, configure .claude/settings.json (30 min)
- Connect Netlify MCP — trigger a deploy from Claude terminal (20 min)
- Re-deploy with env variables managed via Netlify MCP (10 min)

TOOLS THIS SESSION

- Claude Code CLI — auth build + CRUD
- MCP: GitHub — push harness repo

- MCP: Supabase — RLS verification
- MCP: Netlify — first MCP-triggered deploy
- Supabase Auth — Google OAuth provider

✓ **SESSION DELIVERABLE:** Google login live · User data persisting · Agent harness repo created · CLAUDE.md written

WEEK 03 · AGENTS

Multi-Agent Systems + Advanced Claude Code + MCP Wiring

WEEK 3 SATURDAY — LEARN · 7:00 – 10:00 AM PST · 3 hrs

Multi-Agent Architecture + Wiring Your Agent Harness

SESSION PLAN

- What is a multi-agent system: orchestrator + subagents, mental model with diagrams (25 min)
- Agent patterns: sequential pipeline, parallel fan-out, recursive (20 min)
- Build live: 2-agent pipeline — researcher subagent feeds writer subagent (45 min)
- Structured outputs: forcing reliable JSON from Claude (typed schemas) (25 min)
- Error handling and retry logic — what happens when an agent fails (20 min)
- Wire MCP servers into your harness agents — GitHub + Supabase + Netlify all talking (25 min)

TOOLS THIS SESSION

- Claude Code CLI — agent build
- MCP: GitHub — agent pushes code to your repo
- MCP: Supabase — agent reads/writes your DB
- MCP: Netlify — agent triggers deploys
- Claude API direct — for programmatic agent calls from Next.js

✓ **SESSION DELIVERABLE:** 2-agent pipeline running · All 3 MCP servers wired in harness · Structured JSON outputs

WEEK 3 SUNDAY — BUILD · 7:00 – 10:00 AM PST · 3 hrs

Build: Core AI Feature + Multi-Agent Integration

SESSION PLAN

- Spec your product's core AI feature in your harness projects/ folder (20 min)
- Build the AI pipeline with multi-step logic using Claude Code (55 min)
- Store AI outputs in Supabase, display in UI — real data, real users (35 min)
- Add streaming responses and loading states to your AI feature (25 min)
- Test with agents: use your harness agents to write tests, push via GitHub MCP (20 min)
- Peer code review — constructive critique on AI feature design (25 min)

TOOLS THIS SESSION

- Claude Code CLI — feature build

- MCP: Supabase — AI output storage
- MCP: GitHub — test commit and PR creation
- MCP: Netlify — deploy after feature merge
- Claude API (from Next.js) — in-app AI calls

✓ **SESSION DELIVERABLE:** Core AI feature complete · Data persisted · Streaming responses · Tests pushed via MCP

WEEK 04 · SHIP

Production · Full Agent Harness Deploy · Demo Day

WEEK 4 SATURDAY — LEARN · 7:00 – 10:00 AM PST · 3 hrs

Production Readiness + Full Agent Harness Activation

SESSION PLAN

- Production checklist: env vars, error boundaries, rate limits, input validation (25 min)
- Security: prompt injection defense, Supabase RLS audit via MCP, CORS (25 min)
- Performance: lazy loading, Supabase indexes (created via MCP), API caching (20 min)
- HARNESS FINAL: add deployer agent — runs smoke test after every Netlify deploy (30 min)
- HARNESS FINAL: activate full swarm — orchestrator coordinates builder + reviewer + deployer (30 min)
- UI polish sprint: run your harness builder agent on your product UI (20 min)
- Demo prep: write your 4-minute pitch, generate landing copy with Claude (30 min)

TOOLS THIS SESSION

- Claude Code CLI — polish and hardening
- MCP: Supabase — RLS audit, index creation
- MCP: GitHub — final PR review and merge
- MCP: Netlify — deploy + build log check
- Full agent swarm: orchestrator → builder → reviewer → deployer

✓ **SESSION DELIVERABLE:** Production-ready · Secure · Full agent harness operational (4 agents + 3 MCP servers)

WEEK 4 SUNDAY — BUILD · 7:00 – 10:00 AM PST · 3 hrs

Demo Day — Ship It. Present It. Own It.

SESSION PLAN

- Final deploy — custom domain setup on Netlify (if desired) via MCP (20 min)
- Harness handover: confirm repo is clean, CLAUDE.md final, MCP servers all green (15 min)
- Product demos: every builder gets 4 min demo + 2 min Q&A from cohort (90 min)
- Group vote: Most Useful · Most Creative · Most Technically Ambitious (15 min)
- Retrospective: what worked, what broke, what you'd do differently (20 min)
- What's next: launch your product, join Axiomic Labs, return as a mentor (20 min)

TOOLS THIS SESSION

- Full agent harness — operational and yours to keep
- MCP: all 3 servers live in your own accounts
- Your product URL — deployed and shareable
- CLAUDE.md — your personal AI build identity

✓ **SESSION DELIVERABLE:** Live product · Google login · AI-powered · Deployed · Agent harness repo — yours forever

SUMMARY

Cumulative Skills by Week

Skill Area	Week 1	Week 2	Week 3	Week 4
Claude Code CLI	First commands	Full workflows	Agent builds	Swarm activation
MCP: GitHub	—	Connect + push	Agent pushes code	PR review via agent
MCP: Supabase	—	Connect + query	Agent reads/writes	RLS audit + indexes
MCP: Netlify	—	First deploy	Agent triggers deploy	Smoke test agent
Agent Harness	—	Created + CLAUDE.md	MCP wired in	Full swarm live
Multi-agent	—	—	2-agent pipeline	4-agent orchestrator
Google Auth	—	Live	RLS verified	Production hardened
Product	Scaffolded	Auth + CRUD live	AI feature live	Deployed + demoed

What Every Student Walks Away With

Your Product
A live, deployed web app at a real URL. Google login enabled. Core AI feature working. Built from your own idea.

Your Agent Harness Repo
Personal GitHub repo with CLAUDE.md, 4 agent prompts, 3 MCP servers wired. Operational on your own accounts.

MCP Proficiency
GitHub, Supabase, and Netlify MCP servers configured and used in anger. You know when to reach for MCP vs CLI.

Claude Code Fluency
CLI-native workflow. Multi-agent orchestration. Structured outputs. Streaming. Production patterns. All muscle memory.

Stop talking about AI. Start building with it.

academy.axiomic-ai.com · $50/day · Fremont, Bay Area