OPENCODE CLI

Setup Reference Card

OpenCode's open-source agentic command-line coding tool • Gateway: **American Science Cloud Model Access Gateway (MAG)**

OpenCode is a terminal-based coding agent that reads and edits files, runs commands, and works directly inside your repository. This card covers connecting OpenCode to AmSC MAG and validating both the OpenAI-compatible and Anthropic-compatible gateway paths.

1 · Install

Install OpenCode with the official installer or a package manager. Pick one method per machine and avoid mixing installers.

Platform	Command	Updates
macOS / Linux / WSL	curl -fsSL https://opencode.ai/install | bash	opencode upgrade
Windows (PowerShell)	irm https://opencode.ai/install.ps1 | iex	opencode upgrade
npm	npm install -g opencode-ai	npm update -g opencode-ai
macOS (Homebrew)	brew install sst/tap/opencode	brew upgrade opencode
Arch / paru	paru -S opencode-bin	paru -Syu opencode-bin

Verify the install — open a new terminal in your project, then:

```
opencode --version        # confirm the binary is on your PATH
opencode                  # start an interactive session
```

Useful diagnostics:

```
opencode auth list        # list configured provider credentials
opencode models           # list configured providers and models
opencode mcp list         # list configured MCP servers
opencode debug            # troubleshooting/debug tools
opencode upgrade          # update OpenCode
```

2 · Connect to the Model Access Gateway (MAG)

Step 1 — Get an American Science Cloud Account

Follow the instructions listed here: https://amsc-docs-d762d2.gitlab.io/GM-getting-started#access--login to create an AmSC account. Once the account is approved, proceed to the next step.

Step 2 — Request MAG access and generate a Personal Access Token (PAT)

Follow the instructions listed here: https://amsc-docs-d762d2.gitlab.io/model-access-gateway#quick-start

Step 3 — Set environment variables for the OpenAI-compatible protocol

Add these to your shell profile (~/.zshrc, ~/.bashrc, ~/.bash_profile, or PowerShell $PROFILE).

For bash/zsh:

```
# Your MAG project key
export AMSC_I2_API_KEY=<your-MAG-PAT>

# OpenAI-compatible MAG protocol variables
export OPENAI_API_KEY="$AMSC_I2_API_KEY"
export OPENAI_BASE_URL="https://i2-api.genesis.american-science-cloud.org"
```

For PowerShell:

```
# Your MAG project key
$env:AMSC_I2_API_KEY="<your-MAG-PAT>"

# OpenAI-compatible MAG protocol variables
$env:OPENAI_API_KEY=$env:AMSC_I2_API_KEY
$env:OPENAI_BASE_URL="https://i2-api.genesis.american-science-cloud.org"
```

Restart your shell, then confirm that the variables are loaded:

```
env | grep AMSC
env | grep OPENAI
```

Optional — Set environment variables for the Anthropic-compatible protocol

MAG also supports an Anthropic-compatible Messages endpoint for Claude-family models. These variables are useful for Claude Code or other tools that expect Anthropic-style configuration.

For bash/zsh:

```
export ANTHROPIC_AUTH_TOKEN="$AMSC_I2_API_KEY"
export ANTHROPIC_BASE_URL="https://i2-api.genesis.american-science-cloud.org"
```

For PowerShell:

```
$env:ANTHROPIC_AUTH_TOKEN=$env:AMSC_I2_API_KEY
$env:ANTHROPIC_BASE_URL="https://i2-api.genesis.american-science-cloud.org"
```

Confirm the optional Anthropic variables:

```
env | grep ANTHROPIC
```

Secrets: Store keys in environment variables or other local secret stores. Project files such as AGENTS.md and shared configs should contain references to environment variables, not raw keys.

3 · Configure OpenCode for AmSC MAG

OpenCode configuration lives in JSON or JSONC files. Use global config for user-wide settings and project config when a repository needs overrides.

Scope	Location
Global user config	~/.config/opencode/opencode.jsonc
Project config	./opencode.jsonc

Project commands/agents/plugins	./.opencode/
Custom config file	path set by OPENCODE_CONFIG

For AmSC use, configure the MAG provider in OpenCode. Users typically select a model interactively using/models. Projects that need reproducible agent behavior can pin a model in the project config.

Create or modify ~/.config/opencode/opencode.jsonc:

```
{
  "$schema": "https://opencode.ai/config.json",
  "provider": {
    "amsc": {
      "npm": "@ai-sdk/openai-compatible",
      "name": "AmSC Model Access Gateway",
      "options": {
        "baseURL": "{env:OPENAI_BASE_URL}/v1",
        "apiKey": "{env:OPENAI_API_KEY}"
      },
      "models": {
        "gpt-5.3-codex": {"name": "GPT-5.3 Codex"},
        "gpt-5.3-codex-high": {"name": "GPT-5.3 Codex High"},
        "gpt-5.3-codex-xhigh": {"name": "GPT-5.3 Codex XHigh"},
        "gpt-5.1-codex-mini": {"name": "GPT-5.1 Codex Mini"},
        "claude-sonnet-4-6": {"name": "Claude Sonnet 4.6"},
        "claude-opus-4-8": {"name": "Claude Opus 4.8"},
        "mistral-large-3": {"name": "Mistral Large 3"},
        "nova-pro-1": {"name": "Nova Pro 1"},
        "llama-4-maverick": {"name": "Llama 4 Maverick"}
      }
    },
    "amsc-anthropic": {
      "npm": "@ai-sdk/anthropic",
      "name": "AmSC MAG Anthropic-Compatible",
      "options": {
        "baseURL": "{env:ANTHROPIC_BASE_URL}/v1",
        "apiKey": "{env:ANTHROPIC_AUTH_TOKEN}"
      },
      "models": {
        "claude-sonnet-4-6": {"name": "Claude Sonnet 4.6"},
        "claude-opus-4-8": {"name": "Claude Opus 4.8"}
      }
    }
  }
}
```

This registers the OpenAI-compatible MAG provider as amsc and the optional Anthropic-compatible provider as amsc-anthropic.

Verify that OpenCode sees the AmSC models:

```
opencode models | grep '^amsc'
```

4 · Test the MAG Connection

Before relying on OpenCode for a long coding session, verify that MAG answers directly.

Confirm environment variables

```
env | grep AMSC
env | grep OPENAI
env | grep ANTHROPIC
```

List available models

```
curl -s "$OPENAI_BASE_URL/v1/models" -H "Authorization: Bearer $OPENAI_API_KEY"
```

For a readable list of model IDs:

```
curl -s "$OPENAI_BASE_URL/v1/models" -H "Authorization: Bearer $OPENAI_API_KEY" | python3 -c 'import
sys,json; print("\n".join(m["id"] for m in json.load(sys.stdin)["data"]))'
```

Test a chat completion

```
curl -s "$OPENAI_BASE_URL/v1/chat/completions" -H "Authorization: Bearer $OPENAI_API_KEY" -H "Content-Type:
application/json" -d '{"model":"gpt-5.3-codex","messages":[{"role":"user","content":"Reply with OK."}]}'
```

Test the Anthropic-compatible Messages endpoint

```
curl -i "$ANTHROPIC_BASE_URL/v1/messages" -H "Authorization: Bearer $ANTHROPIC_AUTH_TOKEN" -H "anthropic-
version: 2023-06-01" -H "content-type: application/json" -d '{"model":"claude-sonnet-4-
6","max_tokens":32,"messages":[{"role":"user","content":"Reply with OK."}]}'
```

Expected result: HTTP 200 with a JSON response containing assistant text such as OK.

5 · Start OpenCode and Select a Model

Start OpenCode from the project directory:

```
cd my-project
opencode
```

Inside OpenCode, select a model:

```
/models
```

The default MAG provider appears as amsc/..., and the optional Anthropic-compatible provider appears as amsc-anthropic/... .

Examples:

```
amsc/gpt-5.3-codex
amsc/gpt-5.3-codex-high
amsc/claude-sonnet-4-6
amsc-anthropic/claude-sonnet-4-6
amsc/mistral-large-3
```

Run a one-off prompt:

```
opencode run "Explain the authentication flow in this repository."
```

6 · Project Context: AGENTS.md

OpenCode uses AGENTS.md for project-specific context. Put it in the repository root and commit it with the project.

Keep it short: include only the information OpenCode needs to work safely and effectively in the repository.

```
# AGENTS.md

## Project Notes
- Briefly describe what this repository does.
- Mention important directories or generated files.

## Commands
- Test: `pytest`
- Lint: `ruff check .`
- Format: `ruff format .`

## Working Rules
- Keep changes focused.
- Run relevant tests before finishing.
- Store secrets in local environment variables or secret stores.
```

Use /init if you want OpenCode to generate a starter AGENTS.md, then edit it down to the project essentials.

7 · Everyday Commands

Command	What it does
opencode	Start an interactive session in the current directory
opencode --version	Print installed version
opencode models	List configured providers and models
opencode mcp list	List configured MCP servers
opencode debug	Debug or troubleshoot OpenCode
opencode upgrade	Upgrade OpenCode
/models	Select or inspect available models inside OpenCode
/init	Create or update AGENTS.md inside OpenCode
/help	Show available slash commands inside OpenCode