

Agentic Coding with Gemini CLI

Google's AI Agent in Your Terminal

Press Space for next page →

Contact Info

Ken Kousen Kousen IT, Inc.

- ken.kousen@kousenit.com
- <http://www.kousenit.com>
- <http://kousenit.org> (blog)
- Social Media:
 - [@kenkousen](https://twitter.com/kenkousen) (twitter)
 - [@kenkousen@foojay.social](https://mastodon.foojay.social/@kenkousen) (mastodon)
 - [@kousenit.com](https://bluesky.kousenit.com/@kousenit) (bluesky)
- *Tales from the jar side* (free newsletter)
 - <https://kenkousen.substack.com>
 - <https://youtube.com/@talesfromthejarside>

Course Overview

- **Duration:** 5 hours of hands-on learning
- **Format:** Instructor-led with multiple labs
- **Hands-on Labs:** Real codebases in Python, JavaScript, Java
- **Prerequisites:** Command-line experience, development background

Topics Covered

- **Foundation:** Installation, CLI basics, authentication
- **Core Skills:** File operations, shell integration, context management
- **Customization:** GEMINI.md, custom commands, settings.json
- **Safety:** Sandbox mode, approval policies, checkpointing
- **Advanced:** MCP integration, extensions, session management

What is Gemini CLI?

- Open-source AI agent from Google
- **Gemini 3 Pro** (latest) or Gemini 2.5 Pro/Flash
- Built-in tools: Google Search, file ops, shell, web fetch
- Model Context Protocol (MCP) support
- Designed for developers who live in the terminal

Key Differentiators

- **Massive Context:** 1 million token context window
- **Google Search Grounding:** Real-time web access
- **Free Tier Available:** Generous API limits
- **Open Source:** Fully open, community-driven
- **MCP Native:** Built-in Model Context Protocol support

Models Available

- **Gemini 3 Pro:** Most intelligent, best for complex coding
- **Gemini 2.5 Pro:** Strong performance, 1M token context
- **Gemini 2.5 Flash:** Faster, lower cost option
- **Auto routing:** CLI picks best model for each task

API Access

- **Free Tier:** Generous daily limits
- **Google AI Ultra:** Full Gemini 3 Pro access
- **Paid API Key:** Pay-as-you-go pricing
- Enable via `/settings` → "Preview features"

 **Get API Key:** [Google AI Studio](#)

Installation

- **npm** (recommended): `npm install -g @google/gemini-cli`
- **npx** (no install): `npx @google/gemini-cli`
- Verify: `gemini --version`
- Current version: 0.19.x

```
1 # Install globally
2 npm install -g @google/gemini-cli
3
4 # Verify installation
5 gemini --version
```

Authentication

- **Environment Variable:** `export GEMINI_API_KEY="your-key"`
- **Global .env file:** `~/.gemini/.env`
- **Project .env file:** `./.gemini/.env`
- **Google Cloud:** `GOOGLE_CLOUD_PROJECT` for Vertex AI

```
1 # Option 1: Environment variable
2 export GEMINI_API_KEY="your-api-key"
3
4 # Option 2: Global .env file
5 echo 'GEMINI_API_KEY=your-api-key' >> ~/.gemini/.env
6
7 # Option 3: Project .env file
8 mkdir -p .gemini && echo 'GEMINI_API_KEY=your-api-key' >> .gemini/.env
```

Basic Usage Modes

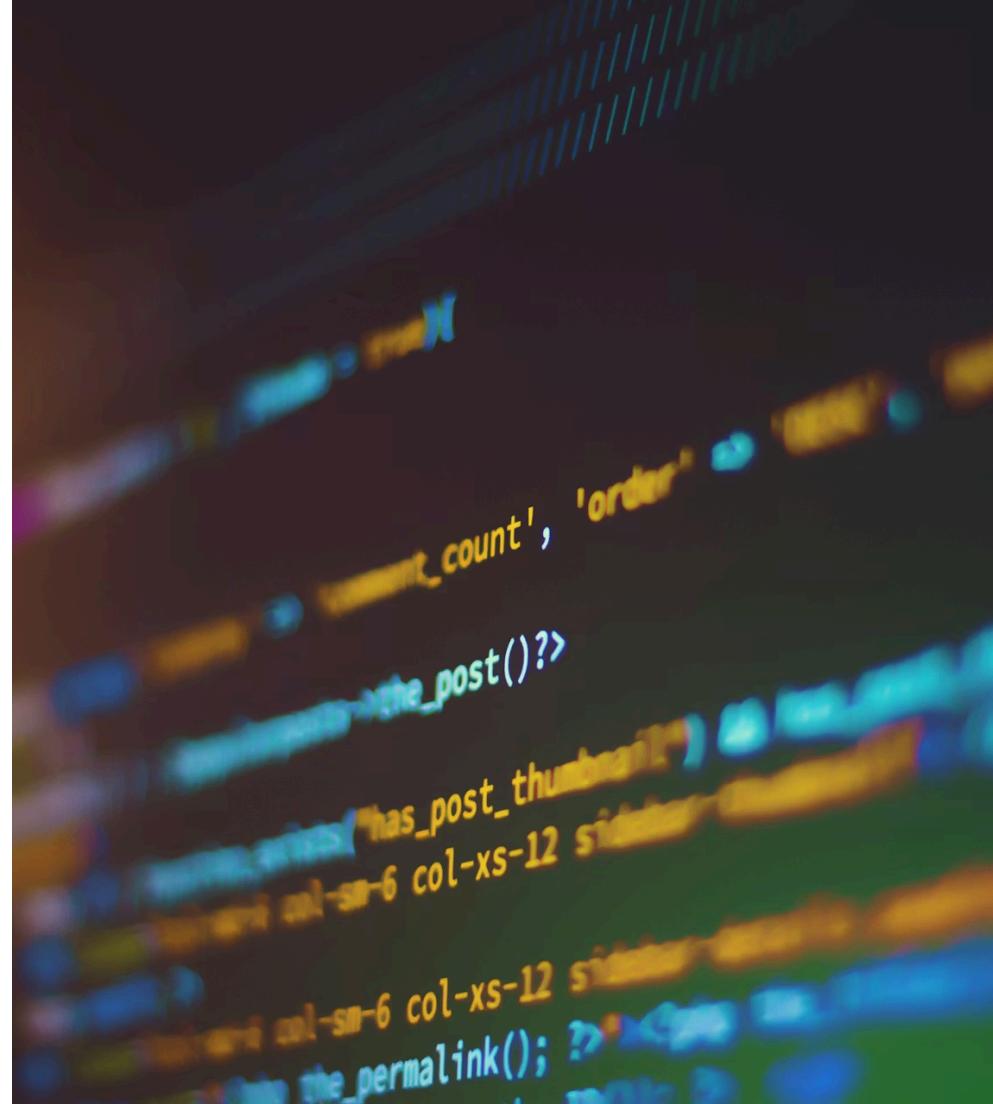
- **Interactive REPL:** `gemini` - Start a conversation
- **One-shot:** `gemini "prompt"` - Single response
- **Piped input:** `echo "task" | gemini`
- **Interactive with context:** `gemini -i "initial context"`

```
1 # Interactive mode
2 gemini
3
4 # One-shot mode
5 gemini "Explain what this codebase does"
6
7 # With initial context
8 gemini -i "You are a Python expert"
```

Core Features

Essential Capabilities

Master the fundamentals



File References with @

- Reference files directly: `@./src/main.js`
- Reference directories: `@./src/` (recursive)
- Reference images: `@./screenshot.png`
- Multiple references in one prompt

```
1 # Reference a specific file
2 gemini "Explain @./src/app.py"
3
4 # Reference multiple files
5 gemini "Compare @./old.js and @./new.js"
6
7 # Reference a directory
8 gemini "Analyze the architecture in @./src/"
```

Shell Integration with !

- Execute shell commands: `!git status`
- Toggle persistent shell mode: `!`
- Gemini can observe and analyze output
- Combine with AI analysis

```
1 # In interactive mode:  
2 > !npm test  
3 # Gemini sees the test output  
4  
5 > !git diff  
6 # Ask Gemini to analyze the changes  
7  
8 # Toggle persistent shell mode  
9 > !
```

Slash Commands: Navigation

- `/help` - Show available commands
- `/clear` - Clear conversation history
- `/memory show` - View loaded context
- `/memory refresh` - Reload GEMINI.md files

Slash Commands: Sessions

- `/init` - Generate project GEMINI.md
- `/chat save <tag>` - Save conversation
- `/compress` - Summarize conversation
- `/restore` - Recover from checkpoint

Slash Commands in Action

```
1 # Show what context is loaded
2 /memory show
3
4 # Reload all GEMINI.md files
5 /memory refresh
6
7 # Generate a GEMINI.md for current project
8 /init
9
10 # Save current conversation
11 /chat save feature-implementation
12
13 # Compress long conversation
14 /compress
```

Keyboard Shortcuts: Editing

- `Ctrl+L` - Clear screen
- `Ctrl+V` - Paste text/images
- `Ctrl+X` - Open external editor

Keyboard Shortcuts: Control

- `Ctrl+Y` - Toggle auto-approval (YOLO mode)
- `Ctrl+C` - Cancel current operation
- `Ctrl+D` - Exit Gemini CLI

Built-in Tools

- **File System:** `read_file()`, `write_file()`, `replace()`, `glob()`
- **Shell:** Execute terminal commands
- **Web:** `google_web_search()`, `web_fetch()`
- **Memory:** `save_memory()` for cross-session recall

```
1 # Gemini automatically uses appropriate tools
2 "Search the web for React 19 new features"
3 # Uses google_web_search()
4
5 "Read all Python files in src/"
6 # Uses glob() and read_file()
7
8 "Update the README with the changes we made"
9 # Uses write_file()
```

Safety & Control

Work Safely

Approval modes and sandboxing

```
link https://devel...  
package _s  
(! function_exists('incode_starter_setup')):  
 * Sets up theme defaults and registers support for vari...  
 * Note that this function is hooked into the after_setu...  
 * runs before the init hook. The init hook is too late f...  
 */  
function incode_starter_setup():  
 /* Make theme available for translation.  
 * Translations can be filed in the /languages/ directory.  
 * If you're building a theme based on incode, use  
 * the incode_theme_setup() hook instead of this function.  
 */  
add_action('init', 'incode_starter_setup');
```

Approval Modes

- **default:** Prompt for approval on tool calls
- **auto_edit:** Auto-approve file edit tools only
- **yolo:** Auto-approve ALL tool calls

```
1 # Default - asks for approval
2 gemini
3
4 # Auto-approve edits only
5 gemini --approval-mode auto_edit
6
7 # Auto-approve everything (YOL0 mode)
8 gemini --yolo
9 # Or use Ctrl+Y in interactive mode
```

Sandbox Mode

- Isolate file operations in a container
- Requires Docker or Podman
- Prevents accidental system changes
- Perfect for exploring unfamiliar code

```
1 # Run in sandbox mode
2 gemini --sandbox
3
4 # Or with -s flag
5 gemini -s "Refactor this entire codebase"
```

Checkpointing

- **Automatic:** Snapshots created before each file modification
- **Shadow Git:** Stored in `~/.gemini/history/` (not your repo)
- **Includes:** Files + conversation + tool call
- **Disabled by default:** Must enable in settings

```
1 // ~/.gemini/settings.json
2 { "checkpointing": { "enabled": true } }
```

Restoring Checkpoints

```
1 # List and select a checkpoint to restore
2 /restore
3
4 # Shows timestamps + filename + tool name
5 # e.g., 2025-06-22T10-00-00Z-app.py-write_file
```

Restores files AND resets conversation to that point



Context Management

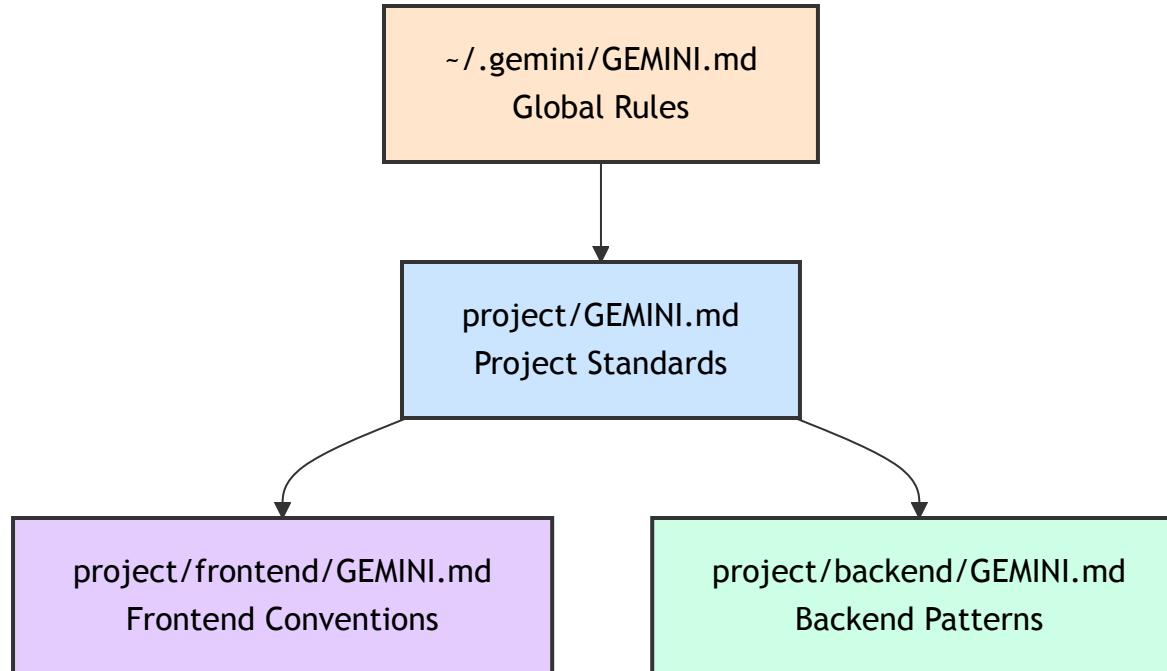
GEMINI.md Files

Project memory and instructions

What is GEMINI.md?

- **Project memory** loaded automatically
- **Coding standards** and conventions
- **Architecture context** for the AI
- **Persistent instructions** across sessions
- Similar to CLAUDE.md or AGENTS.md

Hierarchical Loading



More specific files override general ones

Example GEMINI.md

```
1 # Project: Weather API
2
3 ## Tech Stack
4 - Backend: Python Flask
5 - Database: PostgreSQL
6 - Testing: pytest
7
8 ## Coding Standards
9 - Use type hints for all functions
10 - Follow PEP 8 style guide
11 - Write docstrings for public APIs
12
13 ## Current Focus
14 Implementing caching layer for API responses
```

Memory Commands

- `/memory show` - View combined context
- `/memory refresh` - Reload all GEMINI.md files
- `/memory add <text>` - Append to global GEMINI.md
- `/init` - Generate starter GEMINI.md

```
1 # See what context is loaded
2 /memory show
3
4 # Add a quick note to global memory
5 /memory add "Always use async/await for database calls"
6
7 # Generate a project-specific GEMINI.md
8 /init
```

Modular Imports

- Import other files with `@file.md` syntax
- Break large context into components
- Supports relative and absolute paths

```
1 # GEMINI.md
2
3 ## Project Overview
4 ./docs/architecture.md
5
6 ## Coding Standards
7 ./docs/style-guide.md
8
9 ## API Documentation
10 ./docs/api-reference.md
```

Configuration

Customize Your Setup

settings.json and environment



Configuration Layers

1. **Default values** - Built-in defaults
2. **User settings** - `~/.gemini/settings.json`
3. **Project settings** - `.gemini/settings.json`
4. **Environment variables** - Including `.env` files
5. **Command-line arguments** - Highest priority

settings.json Options

```
1  {
2    "theme": "Default",
3    "vimMode": false,
4    "hideTips": false,
5    "hideBanner": false,
6    "autoAccept": false,
7    "sandbox": false,
8    "checkpointing": true,
9    "preferredEditor": "vscode"
10 }
```

Tool Configuration

- **coreTools**: Whitelist available tools
- **excludeTools**: Blacklist specific tools
- Restrict dangerous operations for safety

```
1  {
2    "coreTools": ["read_file", "write_file", "glob"],
3    "excludeTools": ["shell"]
4 }
```

File Filtering

- **respectGitIgnore**: Honor .gitignore patterns
- **enableRecursiveFileSearch**: Recursive completion
- **.geminiignore**: Custom ignore patterns

```
1  {
2    "fileFiltering": {
3      "respectGitIgnore": true,
4      "enableRecursiveFileSearch": true
5    }
6  }
```

Environment Variables

Variable	Purpose
GEMINI_API_KEY	API authentication (required)
GEMINI_MODEL	Override default model
GOOGLE_CLOUD_PROJECT	GCP project for Vertex AI
GOOGLE_CLOUD_LOCATION	GCP region
HTTP_PROXY	Network proxy

Custom Context Filename

- Change from `GEMINI.md` to custom name
- Support multiple filenames
- Include additional directories

```
1  {
2    "context": {
3      "fileName": ["CONTEXT.md", "GEMINI.md"],
4      "includeDirectories": ["~/shared-context"],
5      "loadFromIncludeDirectories": true
6    }
7 }
```

Advanced Features

Power User Tools

MCP, Extensions, Sessions



Model Context Protocol (MCP)

- Standard protocol for AI-to-system connections
- Connect to external tools and services
- Supports local commands, HTTP, and SSE
- OAuth 2.0 for remote authentication

MCP Configuration: GitHub

```
1  {
2    "mcpServers": {
3      "github": {
4        "command": "npx",
5        "args": ["@modelcontextprotocol/server-github"],
6        "env": {
7          "GITHUB_TOKEN": "${GITHUB_TOKEN}"
8        }
9      }
10    }
11  }
```

MCP Configuration: Database

```
1  {
2    "mcpServers": {
3      "postgres": {
4        "command": "npx",
5        "args": ["@modelcontextprotocol/server-postgres"],
6        "env": {
7          "CONNECTION_STRING": "${DATABASE_URL}"
8        }
9      }
10    }
11  }
```

Managing MCP Servers

```
1 # List configured MCP servers
2 gemini mcp list
3
4 # Add a new MCP server
5 gemini mcp add github
6
7 # Remove an MCP server
8 gemini mcp remove github
9
10 # Test MCP server connection
11 gemini mcp test github
```

MCP Server Options

- **command:** Shell command to start server
- **args:** Command arguments
- **env:** Environment variables
- **cwd:** Working directory
- **timeout:** Startup timeout in ms
- **includeTools/excludeTools:** Filter available tools

Popular MCP Servers

- **GitHub** - Repository operations, issues, PRs
- **PostgreSQL** - Database queries and schema
- **Filesystem** - Extended file operations
- **Slack** - Team communication
- **Playwright** - Browser automation

 **Registry:** modelcontextprotocol.io/registry

Extensions System

- Extend Gemini CLI capabilities
- Place in `~/.gemini/extensions/`
- Configure with `gemini-extension.json`
- List with `gemini --list-extensions`

```
1 # List available extensions
2 gemini --list-extensions
3
4 # Use specific extensions only
5 gemini -e extension1 -e extension2
6
7 # Manage extensions
8 gemini extensions list
9 gemini extensions enable my-extension
```

Custom Commands

- Create reusable prompt templates
- Place TOML files in `~/.gemini/commands/`
- Access via slash command syntax

```
1 # ~/.gemini/commands/review.toml
2 [command]
3 name = "review"
4 description = "Review code for issues"
5
6 [prompt]
7 template = """
8 Review the following code for:
9 - Security vulnerabilities
10 - Performance issues
11 - Best practices violations
12
13 {{content}}
14 """
```

Session Management

- **Resume sessions:** Continue previous conversations
- **List sessions:** View available sessions
- **Delete sessions:** Clean up old conversations

```
1 # Resume the latest session
2 gemini --resume latest
3
4 # Resume by index
5 gemini --resume 3
6
7 # List available sessions
8 gemini --list-sessions
9
10 # Delete a session
11 gemini --delete-session 5
```

Output Formats

- **text**: Default human-readable output
- **json**: Structured JSON for scripting
- **stream-json**: Streaming JSON for real-time

```
1 # JSON output for automation
2 gemini -o json "List all TODO comments"
3
4 # Stream JSON for real-time processing
5 gemini -o stream-json "Analyze this codebase"
```

IDE Integration

- **VS Code Integration:** Connect to workspace
- **Native diff viewing:** Review changes in editor
- **Context sharing:** IDE context available to Gemini

```
1  {
2    "preferredEditor": "vscode"
3 }
```



Practical Applications

Real-World Workflows

Common use cases

Code Exploration

- Understand unfamiliar codebases
- Trace dependencies and data flow
- Find patterns and conventions
- Generate architecture documentation

```
1  gemini "Analyze the architecture of @./src/ and explain  
2  how the components interact"  
3  
4  gemini "Trace the flow from the API endpoint to the database  
5  in @./src/controllers/ and @./src/services/"
```

Test Generation

- Generate unit tests for existing code
- Identify edge cases automatically
- Create integration test scaffolding
- Mock setup and fixtures

```
1  gemini "Create comprehensive unit tests for @./src/utils.py
2  with pytest, including edge cases"
3
4  gemini "Generate integration tests for @./src/api/users.py
5  with proper mocking"
```

Documentation Generation

- README files for projects
- API documentation
- Architecture diagrams (Mermaid)
- Code comments and docstrings

```
1  gemini "Generate a comprehensive README.md for this project"
2
3  gemini "Add detailed docstrings to all public functions
4  in ./src/services/"
5
6  gemini "Create a Mermaid diagram showing the system architecture"
```

Refactoring & Modernization

- Upgrade legacy code patterns
- Apply modern language features
- Improve code organization
- Fix anti-patterns

```
1  gemini "Refactor @./src/legacy.py to use modern Python 3.12
2  features like type hints and match statements"
3
4  gemini "Convert this callback-based code to async/await
5  @./src/api.js"
```

Debugging Workflows

- Analyze error messages and stack traces
- Identify root causes
- Suggest fixes with context
- Test and verify solutions

```
1  gemini "This test is failing with @./tests/output.log.  
2  Analyze the error and fix the issue in @./src/app.py"  
3  
4  gemini "Debug why the API returns 500 errors.  
5  Check @./src/routes.py and @./src/middleware.py"
```

Git Workflows

- Generate commit messages
- Create pull request descriptions
- Analyze diffs and changes
- Resolve merge conflicts

```
1 # Analyze staged changes
2 !git diff --staged
3 "Generate a conventional commit message for these changes"
4
5 # Create PR description
6 "Create a pull request description summarizing the changes
7 from the last 5 commits"
```

CI/CD Integration

- Non-interactive mode for pipelines
- JSON output for parsing
- Exit codes for success/failure
- Automated code reviews

```
1 # In CI/CD pipeline
2 gemini "Review ./src/ for security issues" -o json > review.json
3
4 # Check exit code
5 if gemini "Verify all tests pass" -o json; then
6   echo "All checks passed"
7 fi
```

Best Practices

Professional Workflows

Tips for success



Effective Prompting

- **Be specific** about what you want to achieve
- **Provide context** about goals and constraints
- **Use file references** to ground the conversation
- **Iterate** for complex tasks
- **Include examples** when possible

Safety Guidelines

- **Start in default mode** - Get comfortable first
- **Enable checkpointing** - Safety net for mistakes
- **Review generated code** - Don't blindly accept
- **Use sandbox** for exploratory work
- **Commit often** - Git is your safety net

Project Setup

1. Create comprehensive GEMINI.md
2. Set up project-specific settings.json
3. Configure relevant MCP servers
4. Create custom commands for common tasks
5. Establish team conventions

GEMINI.md Best Practices

- **Keep it focused** - Relevant project info only
- **Update regularly** - Reflect current state
- **Use imports** - Modularize large contexts
- **Include examples** - Show expected patterns
- **Document conventions** - Style guides, patterns

Team Collaboration

- Share GEMINI.md in version control
- Standardize settings.json across team
- Create shared custom commands
- Document AI-assisted workflows
- Review AI-generated code together

Common Pitfalls

- **Overly broad prompts** → Be specific
- **Missing context** → Use GEMINI.md
- **Skipping review** → Always verify output
- **YOLO mode too early** → Build trust first
- **Ignoring checkpoints** → Enable early

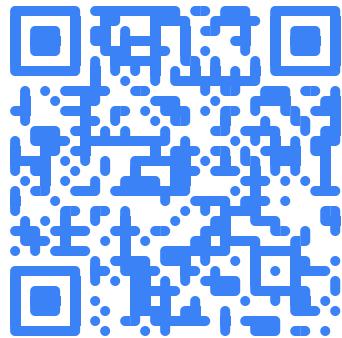
Troubleshooting

- **Authentication issues:** Check `GEMINI_API_KEY`
- **Rate limits:** Use appropriate tier
- **Tool failures:** Check MCP server status
- **Context not loading:** Run `/memory refresh`
- **Debug mode:** Use `gemini -d` for details

```
1 # Debug mode for troubleshooting
2 gemini -d
3
4 # Check memory/context
5 /memory show
6
7 # Refresh context files
8 /memory refresh
```

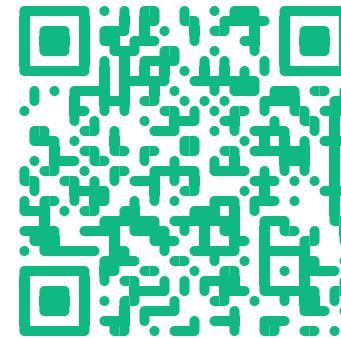
Quick Access

Gemini CLI Docs



github.com/google-gemini/gemini-cli

Course Repository



github.com/kousen/gemini-training

Important Links

Official Documentation

<https://github.com/google-gemini/gemini-cli>

Get API Key

<https://aistudio.google.com/apikey>

MCP Server Registry

<https://modelcontextprotocol.io/registry>

Course Materials

<https://github.com/kousen/gemini-training>

Command Reference: Basic Usage

```
1 # Interactive mode
2 gemini
3
4 # One-shot mode
5 gemini "prompt"
6
7 # Interactive with initial context
8 gemini -i "context"
```

Command Reference: Safety

```
1 # Run in sandbox mode
2 gemini --sandbox
3
4 # Auto-approve all tool calls
5 gemini --yolo
6
7 # Auto-approve file edits only
8 gemini --approval-mode auto_edit
```

Command Reference: Sessions & Output

```
1 # Resume last session
2 gemini --resume latest
3
4 # List available sessions
5 gemini --list-sessions
6
7 # JSON output for scripting
8 gemini -o json "prompt"
9
10 # Debug mode
11 gemini -d
```

Thank You!

Questions?



Kenneth Kousen*Author, Speaker, Java & AI Expert*

kousenit.com | [@kenkousen](https://twitter.com/kenkousen)