# Gemini CLI for Developers – Beginner

- 1 Day Course
- Hands-on and Lecture

## Course Overview

This course provides developers with hands-on experience using Google's open-source Gemini CLI terminal agent. Participants learn how to set up native command-line AI pairing, structure repository-level project rules using GEMINI.md context files, execute precise local file edits, and leverage direct shell execution with approval safety controls.

## Who Should Attend

- Software Engineers
- DevOps Engineers
- Technical Leads

## What You'll Learn

- Install, configure, and authenticate the native Gemini CLI developer environment.
- Leverage the interactive REPL and direct shell integration for rapid pairing.
- Structure repository-level project rules using GEMINI.md context files.
- Execute precise local file reads, edits, diffs, and project restructuring.
- Utilize system symbols (@ and !) to inject file context and trigger shell commands safely.

## Outline

1. Introduction to Gemini CLI Architecture

- Understanding terminal-native AI agents vs. IDE extensions
- Installing Gemini CLI via Node.js (@google/gemini-cli)
- Authentication models: Google Cloud Vertex AI vs. Gemini API Keys
- REPL mode, command-line flags, and non-interactive scripted invocations
- Hands-on: Environment Setup and Initial API Authentication
- Hands-on: Executing First Terminal Conversations and Workspace Invocations

2. Steering Behavior with GEMINI.md

- Anatomy of a GEMINI.md file: Formatting conventions, build steps, and coding style rules
- Project context scoping: Global vs. Directory-level rules
- Automatic rule discovery and workspace initialization with /init
- Preventing unwanted file modifications and enforcing architectural boundaries
- Hands-on: Initializing Custom GEMINI.md Project Context Files
- Hands-on: Testing Rule Enforcement Across Multi-Directory Codebases

3. Precision Context Injection & Shell Guardrails

- Referencing workspace files using @file, @dir, and glob patterns
- Executing terminal commands inline with !shell syntax
- Interactive confirmation prompts, approval policies, and safety controls
- Understanding approval states: Manual prompt, trusted folders, and yolo execution mode
- Hands-on: Targeted Context Injection for Bug Diagnostics
- Hands-on: Safely Executing Test Suites and Build Scripts via Terminal Invocations

4. Code Modification & Iterative Refactoring

- File editing mechanics: Search-and-replace blocks vs. direct file re-writing
- Managing local Git status and inspecting inline AI diffs
- Handling syntax failures, automatic retry loops, and code formatting
- Iterative test-driven refactoring with Gemini CLI pairing
- Hands-on: Automated Code Refactoring and Unit Test Generation
- Hands-on: End-to-End Bug Remediation and Git Commit Preparation

## Prerequisites

- Proficiency with basic command-line navigation and shell commands (Bash/Zsh).
- Experience writing code in at least one major programming language (Python, JavaScript/TypeScript, Go, etc.).
- Node.js (v18+) and npm installed locally.

## Next Courses

- ai-gem-2d1