DEVOPS INTERMEDIATE 3 DAYS

AI Driven CI/CD Pipeline using GitLab

Build a strong foundation in enterprise CI/CD using GitLab, from pipeline design to secure deployments and merge-request quality gates. Learn how to optimize builds using caching and artifacts, apply governance controls, and implement DevSecOps scanning with AI-assisted pipeline creation and troubleshooting.

Why AI Driven GitLab CI/CD?

Faster pipeline development

AI accelerates .gitlab-ci.yml authoring and pipeline design without starting from scratch.

Standardized pipelines across teams

Reusable templates and include patterns enforce consistent practices and compliance.

Reduced pipeline failures

AI-assisted troubleshooting improves diagnosis of YAML, rules, runner, and build errors.

Improved developer experience

Faster CI feedback with caching and better pipeline structure reduces friction.

Secure delivery workflows by default

Security scans and guarded deployments become standard gates in every pipeline.

What you'll be able to do

- Understand GitLab CI/CD fundamentals and enterprise pipeline requirements.
- Build CI pipelines using .gitlab-ci.yml for real applications including build, unit tests, lint checks, and artifact creation.
- Build secure CD pipelines with environment-based deployments, manual approvals for production, and controlled release workflows.
- Use AI to accelerate pipeline creation: generate templates, create reusable CI templates, and auto-suggest improvements.
- Use AI for troubleshooting: interpret pipeline logs, debug YAML errors, and fix rules and runner misconfigurations.
- Integrate DevSecOps gates: secret scanning patterns, dependency scanning, container scanning, and optional IaC scanning.
- Optimize pipelines using caching strategies, artifacts and dependencies, and parallel execution.
- Implement enterprise governance: protected branches, protected environments, approvals, and auditability.
- Deliver a complete AI-assisted GitLab CI/CD workflow as a capstone.

Who this is for

- DevOps Engineers
- Platform Engineers
- SRE Teams
- Cloud Engineers

- Developers maintaining pipelines
- Security teams enforcing CI/CD governance

Curriculum

01 GitLab CI/CD Fundamentals (Enterprise Requirements)

- What GitLab CI/CD solves in modern delivery
- Enterprise requirements (reliability, speed, compliance, traceability)
- Pipeline concepts (stages, jobs, runners, artifacts)
- GitLab variables, environments, and permissions overview
- **Hands-on:** Activity: Map a commit-to-production workflow using GitLab stages and environments

02 Building CI Pipelines with .gitlab-ci.yml (Build, Test, Lint, Artifacts)

- .gitlab-ci.yml structure (stages, jobs, scripts)
- Build and unit testing automation
- Lint checks and quality gates
- Artifacts creation and passing outputs between stages
- **Hands-on:** Lab: Create a CI pipeline that builds, tests, lints, and publishes artifacts

03 Secure CD with Environments and Controlled Releases

- Environment-based deployments (dev, staging, production)
- Manual approvals for production deployments
- Release workflows (tag-based releases, release notes concept)
- Rollback strategy basics (re-deploy last stable)
- **Hands-on:** Lab: Build a CD pipeline with environment deployments and manual approval for production

04 AI-Assisted Pipeline Creation (Templates and Reusable CI Components)

- Using AI to generate .gitlab-ci.yml quickly
- Creating reusable CI templates (include, extends, YAML anchors concepts)
- Standardizing pipelines across projects using templates
- AI-driven suggestions for best practices and improvements
- **Hands-on:** Lab: Generate a pipeline with AI and refactor it into reusable templates

05 AI-Assisted Troubleshooting (Logs, YAML Errors, Rules, Runners)

- Reading GitLab pipeline logs and common failure patterns
- Debugging YAML syntax, rules, and workflow conditions
- Runner troubleshooting (tags, executors, permissions, network access)
- Using AI to propose fixes and validate corrections
- **Hands-on:** Lab: Break a pipeline intentionally and use AI to troubleshoot runner/rules/YAML issues

06 DevSecOps Gates in GitLab CI/CD (Scan and Enforce)

- Secret scanning patterns (preventing leaks, blocking merges)
- Dependency scanning concepts (SCA) and severity thresholds
- Container scanning in pipelines (image scanning workflow)
- Optional IaC scanning concepts (Terraform checks)
- **Hands-on:** Lab: Integrate security scans and enforce policy thresholds in CI

07 Pipeline Optimization (Caching, Artifacts, Parallel Execution)

- Caching strategies (dependencies, build cache)
- Artifacts and dependencies (passing outputs safely between jobs)
- Parallel execution patterns (split tests, matrix-like strategies)
- Reducing pipeline time without reducing confidence
- **Hands-on:** Lab: Add caching + parallel jobs and compare pipeline duration improvements

08 Enterprise Governance (Protected Branches, Environments, Approvals, Auditability)

- Protected branches and merge request approvals
- Protected environments and deployment approvals
- Audit trails (who changed pipeline, who deployed, what version)
- Least privilege permissions for variables and runners
- **Hands-on:** Workshop: Design an enterprise governance checklist for GitLab pipelines

09 Capstone Project (AI-Assisted GitLab CI/CD Workflow)

- Capstone goal: Deliver a production-style GitLab CI/CD pipeline with AI assistance
- CI: build, unit tests, lint, artifacts
- CD: environment-based deploys, manual production approval, controlled release
- DevSecOps: secret scanning patterns, dependency scanning, container scanning, optional IaC scanning
- **Hands-on:** Capstone Lab: Build and present a complete AI-assisted GitLab CI/CD workflow with evidence