DEVOPS ADVANCED 3 DAYS

GitOps using Argo CD and Advanced Kubernetes

Build a strong foundation in pull-based GitOps delivery using Argo CD, from repository structure to production-grade Kubernetes deployments. Learn how to enable self-healing, safe promotions, and rollback strategies while troubleshooting real failures like CrashLoopBackOff, ImagePullBackOff, probe failures, and broken ingress routing in production environments.

Why GitOps with Argo CD?

Faster and safer deployments

Git history becomes deployment history with controlled changes.

Consistency across teams

Standard repo structures, templates, and environment promotion patterns.

Reduced production mistakes

Fewer manual kubectl changes and consistent workflows.

Improved reliability

Drift detection and self-heal keep clusters aligned with desired state.

Better auditability and compliance

Every change is traceable via commits and PRs.

What you'll be able to do

- Understand GitOps principles and why pull-based deployment is the modern standard.
- Install, configure, and operate Argo CD in Kubernetes.
- Design GitOps repository structures for single environment, multi-environment (dev, stage, prod), and multi-team setups.
- Apply advanced Kubernetes production practices: probes, requests and limits, autoscaling concepts, Ingress, ConfigMaps, and Secrets.
- Troubleshoot real-world failures including CrashLoopBackOff, ImagePullBackOff, OOMKilled, and incorrect Services or Ingress.
- Implement Argo CD best practices: auto-sync, self-heal, pruning, and drift detection.
- Perform release strategies using GitOps including safe rollout, version promotion, and rollback via Git revert.
- Implement end-to-end GitOps readiness for application delivery workflows.
- Deliver a working multi-service GitOps deployment as a capstone.

Who this is for

- DevOps Engineers
- SRE Engineers
- Platform Engineers
- Kubernetes Administrators
- Cloud Engineers deploying workloads to Kubernetes
- Developers working in microservices and Kubernetes setups

Curriculum

01 GitOps Foundations (Pull-Based Deployment Principles)

- What GitOps is and why pull-based CD is the modern standard
- Git as the single source of truth for infrastructure and apps
- Declarative delivery and reconciliation loop concepts
- GitOps vs push-based CD (security, auditability, rollback)
- **Hands-on:** Activity: Map a traditional deployment flow into a GitOps pull-based workflow

02 Installing and Operating Argo CD on Kubernetes

- Argo CD core components and architecture overview
- Installation patterns (manifests vs Helm concept)
- Access setup (UI, CLI) and basic authentication concepts
- Creating first Argo CD Application and syncing workloads
- **Hands-on:** Lab: Install Argo CD and deploy a sample app using Git-based sync

03 GitOps Repository Design (Single Env, Multi-Env, Multi-Team)

- Repo structure patterns (apps repo vs infra repo)
- Single environment setup (simple GitOps repo layout)
- Multi-environment layout (dev, stage, prod promotion strategy)
- Multi-team governance (ownership boundaries, folder strategy)
- **Hands-on:** Workshop: Design a GitOps repository structure for 3 teams and 3 environments

04 Kubernetes Production Practices for GitOps Deployments

- Deployments, Services, and stable rollout fundamentals
- Probes (liveness/readiness) and safe rollout behavior
- Requests and limits, autoscaling concepts (HPA intro)
- Ingress basics and traffic routing patterns
- **Hands-on:** Lab: Deploy a workload with probes, requests/limits, ConfigMaps, and Secrets via GitOps

05 Config and Secrets Management (ConfigMaps and Secrets)

- ConfigMaps for application configuration patterns
- Secrets management fundamentals (do not store secrets in Git)
- Safe runtime configuration and environment variables
- Config validation and rollout safety patterns
- **Hands-on:** Lab: Externalize configuration using ConfigMaps and inject secrets safely at runtime

06 Troubleshooting Kubernetes Failures (Real-World Incidents)

- CrashLoopBackOff troubleshooting (logs, config errors, missing deps)
- ImagePullBackOff troubleshooting (image tags, registry auth, policies)
- OOMKilled troubleshooting (memory limits, leaks, right-sizing)
- Service/Ingress misconfigurations (selectors, ports, routing issues)
- **Hands-on:** Lab: Fix broken workloads and restore health using kubectl and Argo CD sync status

07 Argo CD Best Practices (Auto-Sync, Self-Heal, Pruning, Drift Detection)

- Auto-sync configuration and when to enable it
- Self-heal and drift detection workflow
- Pruning and safe deletion behavior
- Sync waves and ordering concepts for multi-service apps
- **Hands-on:** Lab: Enable auto-sync, simulate drift, validate self-heal and pruning behavior

08 Release Strategies in GitOps (Promotion, Rollback, Safe Rollouts)

- Safe rollout practices (progressive changes and verification)
- Version promotion from dev → stage → prod using Git commits
- Rollback strategies (Git revert, pinning last known good commit)
- Change control and approvals for production releases
- **Hands-on:** Lab: Perform release promotion and rollback using Git commits and Argo CD sync

09 End-to-End GitOps Readiness (Delivery Workflow Integration)

- Connecting CI outputs to GitOps (image tagging and manifest updates)
- Required checks and approvals before deployment changes
- Auditability and traceability (who changed what and when)
- Operational readiness (alerts, runbooks, ownership)
- **Hands-on:** Workshop: Design an end-to-end GitOps delivery flow from PR merge to Argo CD sync

10 Capstone Project (Multi-Service GitOps Deployment)

- Capstone goal: Deliver a complete multi-service deployment using GitOps
- Deploy services with ConfigMaps, Secrets, probes, and requests/limits
- Enable Argo CD best practices (auto-sync, self-heal, prune, drift detection)
- Release strategy implemented (promotion + rollback via Git revert)
- **Hands-on:** Capstone Lab: Build and demo a working GitOps repo with Argo CD managing multi-service workloads