DEVOPS INTERMEDIATE 5 DAYS

Microservices on Kubernetes with CI/CD and GitOps (Argo CD)

Build a strong foundation in running microservices on Kubernetes, from architecture basics to CI/CD automation and GitOps delivery. Learn how to design scalable cloud-native services, implement modern release practices, and operate workloads reliably using production-grade Kubernetes patterns.

Why Microservices on Kubernetes?

Faster feature delivery

Microservices enable independent releases and CI/CD makes shipping continuous and safer.

GitOps security and governance

Argo CD improves auditability and reduces direct production access by enforcing Git as source of truth.

Stable and scalable systems

Kubernetes provides self-healing, scaling, and controlled rollout strategies.

Standard enterprise platform foundation

This stack enables scalable SaaS platforms, internal products, and customer-facing services.

Higher engineering productivity

Automated build, test, and deploy reduces manual effort and deployment risk.

What you'll be able to do

- Understand microservices architecture, why enterprises migrate from monoliths, and the trade-offs involved.
- Design microservices using bounded contexts, API contracts, and resilience principles.
- Deploy microservices workloads on Kubernetes using production-aligned objects and best practices.
- Configure Deployments, Services, ConfigMaps, Secrets, Storage, and Ingress for real multi-service systems.
- Implement reliability mechanisms including probes, rolling updates, rollbacks, scaling, and resource limits.
- Troubleshoot real Kubernetes failure modes including CrashLoopBackOff, ImagePullBackOff, and OOMKilled.
- Build CI pipelines using GitHub Actions to build, test, containerize, and push images to a registry.
- Implement GitOps pull-based continuous delivery using Argo CD with drift detection and controlled rollbacks.
- Deliver a complete end-to-end deployment workflow from code commit to production sync on Kubernetes.

Who this is for

- Backend and Full Stack Developers building distributed systems
- DevOps and SRE Engineers operating Kubernetes deployments
- Cloud and Platform Engineers supporting product teams

- Architects and Tech Leads planning modernization initiatives
- Engineering Managers driving DevOps culture and delivery maturity

Curriculum

01 Microservices Architecture Fundamentals (Enterprise View)

- Monolith vs microservices (why enterprises migrate)
- Benefits and trade-offs (complexity, ops overhead, scaling)
- Microservices characteristics (independent deployability, ownership)
- Common enterprise patterns (API gateway, service boundaries, observability needs)
- **Hands-on:** Activity: Break a monolith into microservices and define ownership boundaries

02 Microservices Design (Bounded Contexts + API Contracts)

- Bounded contexts and domain-driven boundaries
- Service-to-service communication patterns (sync vs async)
- API contracts and versioning fundamentals
- Resilience principles (timeouts, retries, circuit breakers concept)
- **Hands-on:** Lab: Define APIs for 3 services and validate contracts with sample payloads

03 Kubernetes Foundations for Microservices Deployments

- Kubernetes architecture recap (control plane, nodes, scheduler)
- Core objects for microservices (Pods, Deployments, Services, Namespaces)
- Labeling strategy for multi-service systems
- kubectl workflow for microservices operations
- **Hands-on:** Lab: Deploy first microservice into Kubernetes with Deployment + Service

04 Production Kubernetes Objects for Multi-Service Systems

- Deployments and rollout strategy basics
- Services for internal discovery (ClusterIP) and external access
- ConfigMaps for application configuration
- Secrets handling for sensitive configuration
- **Hands-on:** Lab: Deploy multiple services with ConfigMaps + Secrets and validate connectivity

05 Storage and Ingress for Real Microservices Workloads

- Storage requirements in microservices (stateful dependencies)
- PersistentVolumeClaims (PVC) and StorageClass concepts
- Ingress fundamentals for routing (host/path based routing)
- Service exposure best practices for enterprise apps
- **Hands-on:** Lab: Add Ingress routing for multiple microservices + attach PVC to stateful component

06 Reliability Patterns in Kubernetes (Probes + Updates + Rollbacks)

- Readiness probes vs liveness probes
- Rolling updates strategy (maxSurge, maxUnavailable)
- Rollback and deployment history practices
- Zero-downtime delivery patterns for services
- **Hands-on:** Lab: Configure probes and simulate a failed rollout followed by rollback

07 Scaling and Resource Controls (Stability in Production)

- Horizontal scaling basics and when to scale services
- Resource requests and limits (CPU/memory)
- Avoiding noisy neighbor and OOMKilled risk
- Performance vs cost awareness for multi-service clusters
- **Hands-on:** Lab: Add requests/limits to services and validate stability under load

08 Troubleshooting Kubernetes Failure Modes (Real Incidents)

- CrashLoopBackOff root causes and debugging approach
- ImagePullBackOff and registry/auth troubleshooting
- OOMKilled causes and resource remediation
- Debug toolkit (kubectl logs, describe, events, exec)
- **Hands-on:** Lab: Fix broken workloads with CrashLoopBackOff, ImagePullBackOff, and OOMKilled scenarios

09 CI Pipelines with GitHub Actions for Microservices

- CI workflow design for microservices (build, test, package)
- Docker build and multi-service containerization strategy
- Tagging strategy (versioning, git sha) and pushing to registry
- Pipeline best practices (caching, parallel jobs, security basics)
- **Hands-on:** Lab: Build GitHub Actions pipeline to build, test, containerize, and push images

10 Kubernetes Deployment Automation from CI (Release Strategy)

- Deployment manifest strategy (kustomize/values concept)
- Promotion workflow (dev → staging → production)
- Environment-specific configuration patterns
- Release traceability (image tags → deployments)
- **Hands-on:** Lab: Update Kubernetes manifests automatically with new image tags in GitHub Actions

11 GitOps Fundamentals with Argo CD (Pull-Based CD)

- Why GitOps for enterprises (auditability, consistency, safety)
- Argo CD architecture and core concepts (apps, sync, health)
- Git as the source of truth
- Manual vs auto sync strategies
- **Hands-on:** Lab: Install Argo CD and deploy a microservices app from a Git repo

12 Drift Detection, Rollbacks, and Controlled Deployments in Argo CD

- Drift detection and reconciliation behavior
- Controlled rollbacks using Git revert strategy
- Sync options and safe deployment controls
- App-of-apps concept (intro for scale)
- **Hands-on:** Lab: Simulate drift, detect it in Argo CD, and restore desired state with rollback

13 End-to-End Workflow (Code Commit → CI → GitOps Sync → Production)

- Full pipeline flow from developer commit to cluster sync
- PR checks, build artifacts, registry push, manifest update
- Argo CD sync and health validation gates
- Release verification and rollback process
- **Hands-on:** Capstone Lab: Deliver an end-to-end workflow from code commit to production sync using GitHub Actions + Argo CD