DEVOPS ADVANCED 3 DAYS

Service Mesh and Microservices Security

Build a strong foundation in securing and operating microservices communication using a Kubernetes service mesh, from mTLS fundamentals to zero trust policies. Learn how to apply service identity, encrypted traffic, authorization controls, safe rollout strategies, and resilience patterns with observability-driven troubleshooting in real-world environments.

Why Service Mesh?

Stronger internal security (Zero Trust)

Encrypt and authenticate every service call using mTLS.

Better reliability and controlled rollouts

Traffic rules enable safer releases and faster rollback.

Reduced blast radius

Authorization policies stop lateral movement across services.

Improved observability

Service-level visibility accelerates troubleshooting and reduces downtime.

Consistent security enforcement

Shift security controls from application code to platform policies.

What you'll be able to do

- Understand what a service mesh is and why it is needed in microservices environments.
- Explain the problems a service mesh solves: secure service-to-service communication, traffic control, observability, and policy enforcement.
- Understand service mesh architecture including data plane vs control plane, sidecar proxy model, service identity, and mTLS.
- Implement mTLS for encrypted and authenticated microservices communication.
- Apply microservices security fundamentals: zero trust inside clusters, least privilege access, and strong authentication between services.
- Implement traffic management policies including routing rules, retries, timeouts, and circuit breaker concepts.
- Apply resilience patterns using the mesh including fault injection and rate limiting concepts.
- Integrate observability using service-to-service metrics, tracing concepts, and dependency visualization.
- Apply best practices for operating a mesh in production including onboarding, upgrades, failure modes, and overhead awareness.
- Deliver a capstone service mesh deployment with security policies enabled.

Who this is for

- Platform Engineers
- DevOps Engineers

- SRE Teams
- Kubernetes Engineers
- Cloud Engineers running microservices
- Security engineers working with Kubernetes environments
- Architects designing microservices platforms

Curriculum

01 Service Mesh Fundamentals (Why It Exists)

- What a service mesh is and where it fits in cloud-native architecture
- Why microservices need a mesh (security, traffic control, observability, policy)
- What a mesh is not (avoid confusion with API gateways and ingress)
- Common use cases in enterprises (compliance, multi-team reliability, zero trust)
- **Hands-on:** Activity: Identify which problems in a microservices system require a mesh vs native Kubernetes features

02 Problems a Service Mesh Solves (Security, Traffic, Observability, Policy)

- Secure service-to-service communication as a default requirement
- Traffic control without changing application code
- Observability for service interactions (latency, errors, dependencies)
- Policy enforcement and governance across teams
- **Hands-on:** Workshop: Map a sample microservices architecture to mesh capabilities and outcomes

03 Service Mesh Architecture (Data Plane vs Control Plane)

- Data plane vs control plane responsibilities
- Sidecar proxy model and traffic interception concepts
- Service identity and workload authentication concepts
- mTLS fundamentals (cert issuance, rotation concepts)
- **Hands-on:** Lab: Deploy a mesh and inspect sidecars, listeners, and workload identity signals

04 mTLS Implementation (Encryption and Authentication Between Services)

- mTLS modes (permissive vs strict concepts)
- Certificate management and rotation awareness
- Enforcing encrypted in-cluster traffic
- Validating mTLS status and troubleshooting common issues
- **Hands-on:** Lab: Enable mTLS for a microservices app and verify encrypted and authenticated traffic

05 Microservices Security Fundamentals with the Mesh (Zero Trust and Least Privilege)

- Zero trust inside clusters and why it matters
- Least privilege access patterns (service-to-service authorization concepts)
- Strong authentication between services using service identity
- Policy design patterns (allowlists, deny-by-default concepts)
- **Hands-on:** Lab: Apply authorization policies to restrict which services can call each other

06 Traffic Management Policies (Routing, Retries, Timeouts, Circuit Breakers)

- Traffic routing rules (version routing, canary concepts)
- Retries and timeouts and their impact on reliability
- Circuit breaker concepts and outlier detection patterns
- Safe rollout patterns using mesh traffic control
- **Hands-on:** Lab: Implement routing + retries + timeouts and validate behavior under failure

07 Resilience Patterns in the Mesh (Fault Injection and Rate Limiting)

- Fault injection concepts (delay, abort) for testing resilience
- Rate limiting concepts (protect upstream dependencies)
- Load shedding and backpressure awareness
- Chaos testing practices for microservices with the mesh
- **Hands-on:** Lab: Inject faults and validate that resilience controls reduce impact and improve stability

08 Observability with Service Mesh (Metrics, Tracing, Dependency Views)

- Service-to-service metrics (latency, error rate, traffic volume)
- Distributed tracing concepts and correlation across services
- Dependency visualization and service graph interpretation
- Golden signals and alerting concepts for service interactions
- **Hands-on:** Lab: Visualize dependencies and troubleshoot an incident using mesh telemetry signals

09 Operating a Mesh in Production (Onboarding, Upgrades, Failure Modes)

- Workload onboarding patterns (namespace labeling, gradual adoption)
- Overhead awareness (latency, CPU/memory, scaling impacts)
- Upgrade strategies and compatibility considerations
- Common failure modes (cert issues, sidecar injection problems, policy breakages)
- **Hands-on:** Lab: Perform a safe onboarding and simulate a failure mode to practice recovery

10 Capstone Project (Service Mesh Deployment with Security Policies)

- Capstone goal: Deliver a secured microservices deployment using a service mesh
- Deploy a multi-service application and onboard it to the mesh
- Enable and enforce mTLS across services
- Apply least privilege authorization policies and validate access boundaries
- Implement traffic controls (retries/timeouts) and add an observability view
- **Hands-on:** Capstone Lab: Demo secure communication, controlled routing, and t