CLOUD BEGINNER 3 DAYS

Cloud-Native Services and Serverless Starter Bootcamp

Build a strong foundation in cloud-native serverless application development, from managed services to event-driven architecture patterns. Learn how to build secure, scalable systems with production-ready reliability, monitoring, and operational best practices for real enterprise workloads.

Why Cloud-Native Serverless?

Faster product delivery

Ship features without waiting on server provisioning and infrastructure cycles.

Higher reliability with managed services

Built-in scaling, retries, and availability patterns reduce downtime risk.

Lower operational burden

No server patching, scaling management, or manual capacity planning.

Strong foundation for modernization

Enables event-driven automation and async workflows across teams.

Cost-efficient for bursty workloads

Pay per execution and scale to zero when idle.

What you'll be able to do

- ✓ Understand cloud-native principles and how they differ from traditional hosting models.
- ✓ Explain serverless computing fundamentals, benefits, and trade-offs.
- ✓ Identify best-fit serverless use cases for APIs, automation, background processing, and scheduled jobs.
- ✓ Build and deploy serverless functions and integrate managed storage services.
- ✓ Design event-driven systems using queues, pub/sub, triggers, retries, and DLQ handling.
- ✓ Implement secure access patterns using IAM roles and least privilege practices.
- ✓ Apply secure secrets handling concepts and avoid hardcoding sensitive values.
- ✓ Implement serverless reliability practices including idempotency, timeouts, and backoff retries.
- ✓ Build basic observability for serverless workloads using logs, metrics, and alerting.
- ✓ Deliver a complete mini-project demonstrating cloud-native and serverless services working together.

Who this is for

- Developers and Backend Engineers
- DevOps and Cloud Engineers
- Platform and SRE Teams (starter level)
- Solution Architects (foundation learning)
- Engineering Leads planning modernization
- Teams migrating away from monolithic hosting

Curriculum

01 Cloud-Native Principles (Modern Hosting Mindset)

- Cloud-native principles (elasticity, managed services, automation-first)
- Traditional hosting vs cloud-native (pets vs cattle, scaling, resilience)
- Shared responsibility model (what you manage vs what cloud manages)
- Cloud-native architecture patterns (stateless services, managed dependencies)
- **Hands-on:** Activity: Classify workloads as VM-based, container-based, or cloud-native managed services

02 Serverless Fundamentals (Benefits and Trade-Offs)

- What serverless is (functions, managed runtimes, event triggers)
- Benefits (cost, scaling, faster delivery, reduced ops)
- Trade-offs (cold starts, timeouts, debugging complexity)
- Serverless vs containers vs Kubernetes decision guide
- **Hands-on:** Activity: Pick 5 real business use cases and decide if serverless is a fit

03 Serverless Use Cases (APIs, Automation, Background Jobs)

- Best-fit use cases for serverless HTTP APIs
- Automation workflows (file processing, notifications, integrations)
- Background processing with queues and workers
- Scheduled jobs and cron triggers for batch tasks
- **Hands-on:** Lab: Design a serverless workload blueprint for API + worker + scheduler

04 Build and Deploy Serverless Functions with Managed Storage

- Function project structure and deployment workflow
- Integrating object storage (uploads, event triggers, metadata patterns)
- Storage integration patterns (API writes + storage event processing)
- Handling errors and designing idempotent writes
- **Hands-on:** Lab: Build a serverless function that stores files in object storage and writes metadata to a managed store

05 Event-Driven Systems with Queues, Pub/Sub, Retries, and DLQ

- Event-driven architecture fundamentals (decoupling, asynchronous processing)
- Queues vs pub/sub (fan-out vs single consumer patterns)
- Retries, visibility timeouts, and poison message handling
- DLQ design and operational workflow
- **Hands-on:** Lab: Implement API → queue → worker pipeline with retry and DLQ handling

06 Secure Access Patterns with IAM Least Privilege

- IAM roles and service identities for serverless workloads
- Least privilege policy design and common pitfalls
- Secure access between functions and services (storage, queue, database)
- Permission boundaries and environment separation concepts
- **Hands-on:** Lab: Apply IAM roles to functions and validate restricted access behavior

07 Secure Secrets Handling (No Hardcoding)

- Why secrets must never be stored in code or container images
- Secrets manager concepts (versioning, rotation awareness)
- Safe configuration patterns (env vars, secure parameter injection)
- Secret leak response workflow (rotate, revoke, audit)
- **Hands-on:** Lab: Inject secrets securely into functions and validate they are not exposed in logs

08 Serverless Reliability Practices (Idempotency, Timeouts, Backoff)

- Idempotency patterns for APIs and workers
- Timeout design and why it prevents cascading failures
- Exponential backoff retries and what not to retry
- Designing safe consumers for at-least-once delivery
- **Hands-on:** Lab: Add idempotency key + backoff retry logic and test duplicate event handling

09 Observability for Serverless (Logs, Metrics, Alerting)

- Observability basics (logs vs metrics vs traces overview)
- Structured logging and correlation id patterns
- Key metrics for serverless (invocations, errors, duration, throttles)
- Alerting on failures and latency (symptom-based monitoring)
- **Hands-on:** Lab: Build dashboards/alerts for API errors, worker failures, and DLQ growth

10 Capstone: (End-to-End Cloud-Native + Serverless System)

- Capstone goal: Build a complete serverless workflow with API + async processing
- HTTP API validates request and writes to managed storage
- Queue/pubsub triggers background worker function
- Implement IAM least privilege and secure secrets handling
- Add retries, DLQ, and observability for production-style readiness
- **Hands-on:** Mini Capstone: Deliver working workflow with logs, metrics, alerts, and documentation