multisoft
Systems

Professional Cloud Developer Training

COURSE CONTENT

GET IN TOUCH

Multisoft Systems
B - 125, Sector - 2, Noida

(+91) 9810-306-956

info@multisoftsystems.com

www.multisoftsystems.com

About Multisoft

Train yourself with the best and develop valuable in-demand skills with Multisoft Systems. A leading certification training provider, Multisoft collaborates with top technologies to bring world-class one-on-one and certification trainings. With the goal to empower professionals and business across the globe, we offer more than 1500 training courses, which are delivered by Multisoft's global subject matter experts. We offer tailored corporate training; project Based Training, comprehensive learning solution with lifetime e-learning access, after training support and globally recognized training certificates.

About Course

The Professional Cloud Developer training by Multisoft Systems is designed for developers aiming to excel in building, deploying, and optimizing applications on the Google Cloud Platform (GCP). This course provides a robust foundation in cloud-native application development, covering key concepts and implementing security best practices.

Module 1: Designing highly scalable, available, and reliable cloud-native applications

1.1 Designing high-performing applications and APIs. Considerations include:

- ✓ Microservices architecture
- ✓ Choosing the appropriate platform based on the use case and requirements (e.g., IaaS [infrastructure as a service], CaaS [container as a service], PaaS [platform as a service], FaaS [function as a service])
- ✓ Application modernization (e.g., containerization)
- ✓ Understanding how Google Cloud services are geographically distributed (e.g., latency, regional services, zonal services)
- ✓ User session management
- ✓ Caching solutions
- ✓ HTTP REST versus gRPC (Google Remote Procedure Call)
- ✓ Incorporating Service Control capabilities offered by API services (e.g. Apigee)
- ✓ Loosely coupled asynchronous applications (e.g., Apache Kafka, Pub/Sub, Eventarc)
- ✓ Instrumenting code to produce metrics, logs, and traces
- ✓ Cost optimization and resource optimization
- ✓ Graceful handling of errors, disasters, and scaling events

1.2 Designing secure applications. Considerations include:

- ✓ Implementing data lifecycle and residency for applicable regulatory requirements
- ✓ Security mechanisms that identify vulnerabilities and protect services and resources (e.g., Identity-Aware Proxy [IAP], Web Security Scanner)
- ✓ Security mechanisms that secure/scan application binaries, dependencies, and manifests (e.g., Container Analysis)
- ✓ Storing, accessing, and rotating application secrets and encryption keys (e.g., Secret Manager, Cloud Key Management Service)
- ✓ Authenticating to Google Cloud services (e.g., application default credentials, JSON Web Token [JWT], OAuth 2.0)

- ✓ End-user account management and authentication by using Identity Platform
- ✓ Identity and Access Management (IAM) roles for users, groups, and service accounts
- ✓ Securing service-to-service communications (e.g., service mesh, Kubernetes Network Policies, Kubernetes namespaces)
- ✓ Running services with keyless and least privileged access (e.g., Workload Identity, Workload identity federation)
- ✓ Certificate-based authentication (e.g., SSL, mTLS)
- ✓ Supply-chain Levels for Software Artifacts (SLSA)

1.3 Choosing storage options for application data. Considerations include:

- ✓ Time-limited access to objects
- ✓ Data retention requirements
- ✓ Structured versus unstructured data (e.g., SQL versus NoSQL)
- ✓ Strong versus eventual consistency
- ✓ Data volume
- ✓ Data access patterns
- ✓ Online transaction processing (OLTP) versus data warehousing

Module 2: Building and testing applications

2.1 Setting up your local development environment. Considerations include:

- ✓ Emulating Google Cloud services for local application development
- ✓ Using the Google Cloud console, Google Cloud SDK, Cloud Shell, and Cloud Workstations
- ✓ Using developer tooling (e.g., common IDEs, Cloud Code, Skaffold)
- ✓ Authenticating to Google Cloud services (e.g., Cloud SQL Auth proxy, AlloyDB Auth proxy)

2.2 Building. Considerations include:

- ✓ Source control management

- ✓ Creating secure container images from code
- ✓ Developing a continuous integration pipeline by using services (e.g., Cloud Build, Artifact Registry) that construct deployment artifacts
- ✓ Code and test build optimization

2.3 Testing. Considerations include:

- ✓ Unit testing
- ✓ Integration testing including the use of emulators
- ✓ Performance testing
- ✓ Load testing
- ✓ Failure testing/chaos engineering

Module 3: Deploying applications

3.1 Adopting appropriate feature rollout strategies. Considerations include:

- ✓ A/B testing
- ✓ Feature flags
- ✓ Backward compatibility
- ✓ Versioning APIs (e.g., Apigee)

3.2 Deploying applications to a serverless computing environment. Considerations include:

- ✓ Deploying applications from source code
- ✓ Using triggers to invoke functions
- ✓ Configuring event receivers (e.g., Eventarc, Pub/Sub)
- ✓ Exposing and securing application APIs (e.g., Apigee)

3.3 Deploying applications and services to Google Kubernetes Engine (GKE). Considerations include:

- ✓ Deploying a containerized application to GKE
- ✓ Integrating Kubernetes role-based access control (RBAC) with IAM

- ✓ Defining workload specifications (e.g., resource requirements)
- ✓ Building a container image by using Cloud Build

Module 4: Integrating applications with Google Cloud services

4.1 Integrating applications with data and storage services. Considerations include:

- ✓ Managing connections to datastores (e.g., Cloud SQL, Firestore, Bigtable, Cloud Storage)
- ✓ Reading/writing data to or from various datastores
- ✓ Writing an application that publishes or consumes data asynchronously (e.g., from Pub/Sub or streaming data sources)
- ✓ Orchestrate application services with Workflows, Eventarc, Cloud Tasks, and Cloud Scheduler

4.2 Integrating applications with Google Cloud APIs. Considerations include:

- ✓ Enabling Google Cloud services
- ✓ Making API calls by using supported options (e.g., Cloud Client Library, REST API, or gRPC, API Explorer) taking into consideration:
- o Batching requests
- o Restricting return data
- o Paginating results
- o Caching results
- o Error handling (e.g., exponential backoff)
- ✓ Using service accounts to make Cloud API calls
- ✓ Integrating with Google Cloud’s operations suite