Google Cloud

Professional Cloud Developer

Certification exam guide

A Professional Cloud Developer builds and configures scalable, secure applications by using Google-recommended tools and best practices. They are proficient in the full development lifecycle, from architecting cloud-native applications to integrating advanced machine learning capabilities. They are also responsible for utilizing generative AI APIs to create intelligent user experiences. Additionally, they leverage AI-powered development tools—such as AI coding assistants, context engineering, and automated debugging agents—to accelerate delivery and improve code quality.

Section 1: Designing highly scalable, secure, and reliable cloud-native applications (~32% of the exam)

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

- Choosing the appropriate platform based on the use case and requirements (e.g., Compute Engine, Google Kubernetes Engine, Cloud Run)
- Building, refactoring, and deploying application containers to Cloud Run and GKE
- Understanding how Google Cloud services are geographically distributed (e.g., latency, regional services, zonal services)
- Understanding the use cases for load balancers
- Enabling session affinity for performant content delivery
- Implementing caching solutions (e.g., Memorystore)
- Creating and deploying APIs (e.g., HTTP REST, gRPC [Remote Procedure Call])
- Using application rate limiting, authentication, and observability (e.g., Apigee, Cloud API Gateway)
- Integrating applications using asynchronous or event-driven approaches (e.g., Eventarc, Pub/Sub)
- Defining resource requirements for workloads
- Optimizing for cost and resource usage
- Understanding data replication to support zonal and regional failover models
- Using traffic splitting strategies (e.g., gradual rollouts, rollbacks, A/B testing) on a new service on Cloud Run or GKE
- Orchestrating application services with Workflows, Eventarc, Cloud Tasks, and Cloud Scheduler

1.2 Designing secure applications. Considerations include:

- Implementing data retention and organization policies (e.g., Cloud Storage Object Lifecycle Management, Cloud Storage use and lock retention policies)
- Using security mechanisms that identify vulnerabilities and protect services and resources (e.g., Identity-Aware Proxy [IAP], Web Security Scanner)
- Responding to and resolving vulnerabilities, including those identified by Artifact Analysis and Security Command Center
- Storing, accessing, and rotating application secrets, credentials, and encryption keys (e.g., Secret Manager, Cloud Key Management Service, Workload Identity Federation)
- Authenticating to Google Cloud services (e.g., Application Default Credentials, JSON Web Token [JWT], OAuth 2.0, Cloud SQL Auth Proxy, AlloyDB Auth Proxy, Identity Platform, WIF)
- Securing cloud resources using Identity and Access Management (IAM) roles for service accounts
- Incorporating secure service-to-service communications (e.g., Cloud Service Mesh, Kubernetes Network Policies, Direct VPC egress, private service connectivity)
- Running services with least privileged access
- Securing application artifacts using Binary Authorization

1.3 Storing and accessing data. Considerations include:

- Selecting the appropriate storage system based on the volume of data and performance requirements
- Designing appropriate schemas for structured databases (e.g., AlloyDB, Spanner) and unstructured databases (e.g., Bigtable, Firestore)
- Understanding the implications of eventual and strongly consistent replication of AlloyDB, Bigtable, Cloud SQL, Spanner, and Cloud Storage
- Creating signed URLs to grant access to Cloud Storage objects
- Writing data to BigQuery for analytics and AI/ML workloads

Section 2: Building and testing applications (~23% of the exam)

2.1 Setting up your development environment. Considerations include:

- Emulating Google Cloud services using the Google Cloud CLI for local application development and local unit testing
- Using the Google Cloud console, Cloud SDK, Cloud Code, Gemini Cloud Assist, Cloud Shell, and Cloud Workstations

- Configuring IDEs with the appropriate integrations (e.g., Cloud SDK, AI tooling [coding assistants, MCP servers])

2.2 Building. Considerations include:

- Using Cloud Build and Artifact Registry to build and store containers from source code
- Configuring provenance in Cloud Build (e.g., Binary Authorization)

2.3 Testing. Considerations include:

- Writing unit tests with the help of AI coding assistants
- Executing automated integration tests in Cloud Build

Section 3: Configuring cloud-native applications for deployment (~24% of the exam)

3.1 Deploying applications to Cloud Run. Considerations include:

- Deploying applications from source code
- Invoking Cloud Run services using triggers (e.g., Eventarc, Pub/Sub)
- Configuring event receivers (e.g., Eventarc, Pub/Sub)
- Versioning, exposing and securing APIs in applications (e.g., Apigee)

3.2 Deploying containers to GKE. Considerations include:

- Deploying containerized applications
- Implementing Kubernetes health checks to increase application availability
- Incorporating Horizontal Pod Autoscaler attributes (scaling, metrics)

Section 4: Integrating applications with Google Cloud services (~21% of the exam)

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

- Managing connections to various Google Cloud datastores (e.g., Cloud SQL, Firestore, Cloud Storage)
- Reading and writing data to and from various Google Cloud data sources
- Writing applications that publish and consume data using messaging services

4.2 Consuming Google Cloud APIs. Considerations include:

- Enabling Google Cloud services
- Making API calls by using supported options (e.g., Cloud Client Libraries, REST API, gRPC, API Explorer) taking into consideration:
    - Batching requests
    - Restricting return data
    - Paginating results
    - Caching results
    - Handling errors (e.g., exponential backoff)
- Using service accounts to make Cloud API calls

4.3 Troubleshooting and observability. Considerations include:

- Instrumenting code to facilitate troubleshooting using metrics, logs, and traces in Google Cloud Observability
- Identifying and resolving issues using Google Cloud Observability
- Managing application issues using Error Reporting
- Using trace IDs to correlate trace spans across services
- Using AI-assisted observability