Google Cloud

The exam guide for this exam will be updated on March 12, 2025.

- Refer to this exam guide if you are taking the exam before March 12.
- Refer to this updated exam guide if you are taking the exam on or after March 12. This version includes Google's AI tools (Gemini Code Assist and Cloud Assist), and includes a new objective, 4.3, that represents troubleshooting and observability tasks.

Professional Cloud Developer

Certification exam guide (effective through March 11, 2025)

A Professional Cloud Developer builds and deploys scalable, secure, and highly available applications by using Google-recommended tools and best practices. This individual has experience with cloud-native applications, containerized applications, APIs, developer tools, orchestration tools, managed services, test strategies, serverless platforms, and next-generation databases. This individual also has proficiency with at least one general-purpose programming language and instruments their code to produce metrics, logs, and traces.

Section 1: Designing highly scalable, available, and reliable cloud-native applications (~33% of the exam)

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

Section 2: Building and testing applications (~26% of the exam)

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

Section 3: Deploying applications (~19% of the exam)

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

Section 4: Integrating applications with Google Cloud services (~22% of the exam)

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:
    - Batching requests
    - Restricting return data
    - Paginating results
    - Caching results
    - Error handling (e.g., exponential backoff)
- Using service accounts to make Cloud API calls
- Integrating with Google Cloud's operations suite

Google Cloud

Professional Cloud Developer

Certification exam guide (effective March 12, 2025)

A Professional Cloud Developer builds and deploys scalable, secure, and highly available applications by using Google-recommended tools and best practices. This individual has experience with cloud-native applications, Google Cloud APIs, developer and AI tools, managed services, orchestration tools, serverless platforms, containerized applications, test and deployment strategies, problem determination and resolution, and datastores. This individual also has proficiency with at least one general-purpose programming language and can instrument their code to produce metrics, logs, and traces.

Section 1: Designing highly scalable, available, and reliable cloud-native applications (~36% 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, GKE, 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)
- Configuring load balancers and applications for session affinity and performant content delivery
- Implementing caching solutions (e.g., Memorystore)
- Creating and deploying APIs (e.g., HTTP REST, gRPC [Google 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)
- 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)
- Managing and authenticating end-user accounts (e.g., Identity Platform)
- Securing cloud resources using Identity and Access Management (IAM) roles for service accounts
- Securing service-to-service communications (e.g., Cloud Service Mesh, Kubernetes Network Policies)
- 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, Datastore)
- 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, Gemini Code Assist, Cloud Shell, and Cloud Workstations

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 Gemini Code Assist
- Executing automated integration tests in Cloud Build

Section 3: Deploying applications (~20% 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)
- Exposing and securing APIs in applications (e.g., Apigee)
- Deploying a new API version in Cloud Endpoints considering backward compatibility

3.2 Deploying containers to GKE. Considerations include:
- Deploying containerized applications
- Defining resource requirements for container workloads
- Implementing Kubernetes health checks to increase application availability
- Configuring the Horizontal Pod Autoscaler for cost optimization

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 datastores
- Writing applications that publish and consume data using Pub/Sub

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 Gemini Cloud Assist