Back to Professional Cloud Developer Certification > Current (/certification/cloud-developer)

Professional Cloud Developer

Certification exam guide

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

*Note: The exam does not directly assess coding skill. It focuses on your ability to leverage Google Cloud services and recommended practices in order to build, test, deploy, and manage scalable and highly available applications. If you are proficient in at least one general purpose coding language, you should be able to interpret any questions with code snippets.

Register now (https://webassessor.com/googlecloud)

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

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

- Microservices
- Scaling velocity characteristics/tradeoffs of IaaS (infrastructure as a service), CaaS (container as a service), PaaS (platform as a service), and FaaS (function as a service)
- 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)
- Designing API services with API Gateway and Cloud Endpoints
- Loosely coupled asynchronous applications (e.g., Apache Kafka, Pub/Sub, Eventarc)
- Instrumenting code to produce metrics, logs, and traces
- Graceful shutdown of applications on platform termination
- Writing fault-tolerant code

1.2 Designing secure applications. Considerations include:

- Implementing data lifecycle and residency requirements relevant for applicable regulations
- Security mechanisms that protect services and resources
- Security mechanisms that secure/scan application binaries and manifests
- Storing, accessing, and rotating application secrets and 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 using Identity Platform
- IAM roles for users, groups, and service accounts
- Securing service-to-service communications (e.g., service mesh, Kubernetes Network Policies, and Kubernetes namespaces)
- Running services with least privileged access (e.g., Workload Identity)
- Certificate-based authentication (e.g., SSL, mTLS)

1.3 Managing application data. Considerations include:

- Defining database schemas for Google-managed databases (e.g., Firestore, Cloud Spanner, Bigtable, Cloud SQL)
- Defining a data storage key structure for high-write applications
- Choosing data storage options based on use case considerations, such as:
  - Time-limited access to objects
  - Data retention requirements
  - Structured versus unstructured data
  - Strong versus eventual consistency
  - Data volume
  - Data access patterns
  - Online transaction processing (OLTP) versus data warehousing

Section 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, and Cloud Shell tools
- Using developer tooling (e.g., Cloud Code, Skaffold)

2.2 Building. Considerations include:

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

2.3 Testing. Considerations include:

- Unit testing (e.g., emulators)
- Integration testing
- Performance testing
- Load testing
- Failure testing/chaos engineering

Section 3: Deploying applications

3.1 Adopting appropriate feature rollout strategies. Considerations include:

- A/B testing
- Feature flags
- Backward compatibility

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

- Sizing and scaling serverless environments
- Deploying from source code
- Invocation via triggers
- Configuring event receivers
- Exposing and securing application APIs (e.g., API Gateway, Cloud Endpoints)

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

- Deploying a containerized application to GKE
- Integrating Kubernetes RBAC with Identity and Access Management (IAM)
- Configuring Kubernetes namespaces
- Defining workload specifications (e.g., resource requirements)
- Building a container image using Cloud Build
- Configuring application accessibility to user traffic and other services
- Managing container lifecycle

Section 4: Integrating Google Cloud services

4.1 Integrating an application with data and storage services. Considerations include:

- Managing connections to data stores (e.g., Cloud SQL, Cloud Spanner, Firestore, Bigtable, Cloud Storage)
- Reading/writing data to/from various data stores
- Writing an application that publishes/consumes data asynchronously (e.g., from Pub/Sub)

4.2 Integrating an application with compute services. Considerations include:

- Using service discovery (e.g., Service Directory)
- Reading instance metadata to obtain application configuration
- Graceful application startup and shutdown

4.3 Integrating Cloud APIs with applications. Considerations include:

- Enabling a Cloud API
- Making API calls 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

Section 5: Managing deployed applications

5.1 Managing cloud compute services (e.g., Google Kubernetes Engine, serverless). Considerations include:

- Analyzing lifecycle events
- Using external metrics and corresponding alerts
- Configuring workload autoscaling

5.2 Troubleshooting applications. Considerations include:

- Using Debugger

- Using Cloud Logging
- Using Cloud Monitoring
- Using Cloud Profiler
- Using Cloud Trace
- Using Error Reporting
- Using documentation, forums, and Google Cloud support

Take the next step

Tell us what you're solving for. A Google Cloud expert will help you find the best solution.

Contact sales (/contact)

Work with a trusted partner

Find a partner (https://cloud.google.com/find-a-partner)

Start using Google Cloud

Try it free (https://console.cloud.google.com/freetrial)

Continue browsing

See all products (/products)