PassQueen
a.k.a dumpsqueen.com

# Google Cloud Certified - Professional Cloud DevOps Engineer Exam

## Google Google-Professional-Cloud-DevOps-Engineer

**Version Demo**

**Total Demo Questions: 10**

**Total Premium Questions: 101**

**Buy Premium PDF**

https://passqueen.com

support@passqueen.com

# Topic Break Down

| Topic | No. of Questions |
|---|---|
| Topic 1, Applying site reliability engineering principles to a service | 19 |
| Topic 2, Building and implementing CI/CD pipelines for a service | 29 |
| Topic 3, Implementing service monitoring strategies | 27 |
| Topic 4, Optimizing service performance | 10 |
| Topic 5, Managing service incidents | 12 |
| Topic 6, Mix Questions | 4 |
| Total | 101 |

QUESTION NO: 1

You are running an experiment to see whether your users like a new feature of a web application. Shortly after deploying the feature as a canary release, you receive a spike in the number of 500 errors sent to users, and your monitoring reports show increased latency. You want to quickly minimize the negative impact on users. What should you do first?

**A.** Roll back the experimental canary release.

**B.** Start monitoring latency, traffic, errors, and saturation.

**C.** Record data for the postmortem document of the incident.

**D.** Trace the origin of 500 errors and the root cause of increased latency.

**ANSWER: A**

**Explanation:**

**Roll back the experimental canary release.** is the appropriate first action because the canary is the most recent, deliberately limited production change and the service is showing clear user-facing harm: elevated HTTP 500 responses and increased latency. The immediate incident-management priority is mitigation: stop exposing users to the suspected unhealthy version and restore the last known-good application behavior. Rolling back the canary rapidly reduces the blast radius and protects both the current canary audience and any additional users who might otherwise receive the new release as rollout continues.

Canary deployments are intended to make this decision safe and fast. They expose a new version to a small portion of traffic, compare key health signals such as error rate and latency against expected behavior, and enable an automated or manual rollback when those signals breach acceptable thresholds. After service impact has been contained, the team can preserve relevant telemetry and investigate the failure with less pressure and less ongoing customer impact. Google Cloud's canary analysis guidance describes using metrics to determine whether a deployment should be promoted or rolled back, while Google SRE guidance emphasizes mitigating user impact as the first priority during an incident. See Automated canary analysis on Google Cloud and Google SRE: Managing Incidents.

**QUESTION NO: 2**

Your application services run in Google Kubernetes Engine (GKE). You want to make sure that only images from your centrally-managed Google Container Registry (GCR) image registry in the altostratimages project can be deployed to the cluster while minimizing development time. What should you do?

**A.** Create a custom builder for Cloud Build that will only push images to gcr.io/altostrat-images.

**B.** Use a Binary Authorization policy that includes the whitelist name pattern gcr.io/altostratimages/*.

**C.** Add logic to the deployment pipeline to check that all manifests contain only images from gcr.io/altostrat-images.

**D.** Add a tag to each image in gcr.io/altostrat-images and check that this tag is present when the image is deployed.

**ANSWER: B**

**Explanation:**

Use a Binary Authorization policy that includes the whitelist name pattern `gcr.io/altostratimages/*`. Binary Authorization is a GKE-integrated admission control service that evaluates container images when a workload is submitted to the cluster. An admission policy can define an image allowlist through `admissionWhitelistPatterns`, allowing images whose registry path matches the specified pattern to bypass attestation requirements. Restricting the pattern to the centrally managed registry project enforces the required source boundary at deployment time, regardless of the developer tool or delivery pipeline that submits the manifest.

This is the lowest-development-effort solution because the protection is centrally enforced by the cluster admission path rather than relying on each application pipeline to implement and maintain custom validation logic. The wildcard is important because workload manifests reference individual image repositories and tags or digests below the project-level registry path. Configure Binary Authorization enforcement for the GKE cluster and use the allowlist pattern in the policy so only matching

images are admitted under this exception. See the Binary Authorization policy YAML reference and Configure a Binary Authorization policy for GKE.

## QUESTION NO: 3

You support a web application that runs on App Engine and uses CloudSQL and Cloud Storage for data storage. After a short spike in website traffic, you notice a big increase in latency for all user requests, increase in CPU use, and the number of processes running the application. Initial troubleshooting reveals:

- After the initial spike in traffic, load levels returned to normal but users still experience high latency.
- Requests for content from the CloudSQL database and images from Cloud Storage show the same high latency.
- No changes were made to the website around the time the latency increased.
- There is no increase in the number of errors to the users.

You expect another spike in website traffic in the coming days and want to make sure users don't experience latency. What should you do?

**A.** Upgrade the GCS buckets to Multi-Regional.

**B.** Enable high availability on the CloudSQL instances.

**C.** Move the application from App Engine to Compute Engine.

**D.** Modify the App Engine configuration to have additional idle instances.

## ANSWER: D

**Explanation:**

Modify the App Engine configuration to have additional idle instances. The simultaneous latency seen for both Cloud SQL-backed content and Cloud Storage images indicates that the delay is occurring in the application request path rather than being isolated to one storage service. During a traffic burst, App Engine must create and initialize additional serving instances when existing capacity is insufficient. Requests can queue while new instances start, increasing observed latency and causing autoscaling activity and elevated application CPU usage. Keeping a configured pool of idle instances means warmed, initialized capacity is available immediately when the next traffic spike arrives, reducing startup-related request queuing and improving responsiveness.

For automatically scaled App Engine services, the `min_idle_instances` setting controls how many idle instances App Engine should keep ready. This is specifically useful for workloads where rapid traffic increases make instance startup time visible to users. The setting should be sized using observed traffic patterns, startup time, concurrency, and latency objectives, while balancing the additional cost of maintaining idle capacity. Google documents idle-instance behavior and the relevant automatic-scaling configuration in How instances are managed and the app.yaml automatic scaling reference.

## QUESTION NO: 4

You support an application that stores product information in cached memory. For every cache miss, an entry is logged in Stackdriver Logging. You want to visualize how often a cache miss happens over time. What should you do?

**A.** Link Stackdriver Logging as a source in Google Data Studio. Filter the logs on the cache misses.

**B.** Configure Stackdriver Profiler to identify and visualize when the cache misses occur based on the logs.

**C.** Create a logs-based metric in Stackdriver Logging and a dashboard for that metric in Stackdriver Monitoring.

**D.** Configure BigQuery as a sink for Stackdriver Logging. Create a scheduled query to filter the cache miss logs and write them to a separate table.

## ANSWER: C

**Explanation:**

Create a logs-based metric in Stackdriver Logging and a dashboard for that metric in Stackdriver Monitoring. A logs-based metric extracts numerical time-series data from matching log entries. In this case, configure a counter metric with a Logs Explorer query that identifies the cache-miss log entries. Each matching entry increments the metric, creating a time series that represents the number of cache misses during each reporting interval. Cloud Monitoring can then chart that metric on a dashboard, allowing the team to visualize the cache-miss rate and trend over time. The metric can also be used as the basis for alerting if the cache-miss count exceeds an expected threshold. This approach is purpose-built for converting operational events recorded in Cloud Logging into monitored, queryable, and visualizable telemetry, without requiring data exports or scheduled processing. Google documents that logs-based metrics derive metric data from log entries and that user-defined metrics are available for charts and alerting in Cloud Monitoring. See Logs-based metrics documentation and Cloud Monitoring charts and Metrics Explorer documentation.

## QUESTION NO: 5

You support a multi-region web service running on Google Kubernetes Engine (GKE) behind a Global HTTP/S Cloud Load Balancer (CLB). For legacy reasons, user requests first go through a third-party Content Delivery Network (CDN), which then routes traffic to the CLB. You have already implemented an availability Service Level Indicator (SLI) at the CLB level. However, you want to increase coverage in case of a potential load balancer misconfiguration, CDN failure, or other global networking catastrophe. Where should you measure this new SLI? (Choose two.)

**A.** Your application servers' logs.

**B.** Instrumentation coded directly in the client.

**C.** Metrics exported from the application servers.

**D.** GKE health checks for your application servers.

**E.** A synthetic client that periodically sends simulated user requests.

## ANSWER: B E

**Explanation:**

Instrumentation coded directly in the client and a synthetic client that periodically sends simulated user requests provide availability measurements from outside the existing load-balancer measurement boundary. Client-side instrumentation captures the experience of real users, including whether a request can traverse the third-party CDN, reach the global load balancer, and receive a usable response. This makes it an especially strong user-journey SLI when the application can safely and consistently report client telemetry.

A synthetic client complements real-user instrumentation by executing controlled, periodic requests from an external vantage point. It can detect broad reachability or routing failures even during periods with little or no real traffic, and it tests the complete externally visible request path through the CDN and load-balancing layer. The synthetic check should validate a meaningful response, not merely TCP connectivity, and should run from suitable geographic locations to reflect the service's global audience.

Using both measurement sources gives broader coverage: real clients represent observed user outcomes, while synthetic clients continuously test the end-to-end path. This follows the SRE guidance to define service-level indicators around user-visible behavior and to use black-box monitoring to identify externally observable failures. See Google SRE Workbook: Monitoring and Google Cloud documentation on service monitoring and SLOs.

## QUESTION NO: 6

You are ready to deploy a new feature of a web-based application to production. You want to use Google Kubernetes Engine (GKE) to perform a phased rollout to half of the web server pods.

What should you do?

**A.** Use a partitioned rolling update.

**B.** Use Node taints with NoExecute.

**C.** Use a replica set in the deployment specification.

**D.** Use a stateful set with parallel pod management policy.

**ANSWER: A**

**Explanation:**

Use a partitioned rolling update. A partitioned rolling update provides a controlled, incremental way to update only a defined subset of pods rather than immediately applying the new version to every replica. With a StatefulSet configured for a rolling update, the `partition` value identifies the ordinal at which updates begin. For example, for ten replicas, setting the partition to five updates the pods with ordinals five through nine, while pods zero through four continue running the prior revision. This allows the new web-server feature to be evaluated on approximately half of the serving pods in production before reducing the partition and completing the rollout.

This approach supports phased delivery because the operator can pause after the initial subset is updated, validate application behavior and operational signals, and then progressively update the remaining pods. If issues are detected, the partition can remain in place so that the unchanged pods continue serving the earlier version while remediation occurs. Kubernetes documents that StatefulSet rolling updates support partitions to stage updates by pod ordinal, and GKE supports these standard Kubernetes workload-update mechanisms. See the Kubernetes StatefulSet partitioned rolling update documentation and the GKE guide to updating deployments.

**QUESTION NO: 7**

You support a high-traffic web application that runs on Google Cloud Platform (GCP). You need to measure application reliability from a user perspective without making any engineering changes to it.

What should you do? (Choose two.)

**A.** Review current application metrics and add new ones as needed.

**B.** Modify the code to capture additional information for user interaction.

**C.** Analyze the web proxy logs only and capture response time of each request.

**D.** Create new synthetic clients to simulate a user journey using the application.

**E.** Use current and historic Request Logs to trace customer interaction with the application.

**ANSWER: D E**

**Explanation:**

Creating new synthetic clients to simulate a user journey using the application is appropriate because synthetic monitoring evaluates externally observable behavior, such as whether critical pages and workflows are reachable, complete successfully, and meet latency expectations. It does not require modifying the application code because the synthetic client acts like an independent user outside the application. These checks can run continuously from selected locations and provide a consistent, controlled measure of availability and performance for important user journeys.

Using current and historic Request Logs to trace customer interaction with the application provides a complementary real-user perspective without application changes. Existing request logs can reveal the requests real customers made, their timing, response status, latency, and traffic patterns. Examining both live and historical data helps establish normal behavior, identify reliability regressions, and assess the practical experience of actual users over time. Together, synthetic journeys and request-log analysis provide black-box and real-user evidence of reliability while avoiding code instrumentation. Google Cloud documents externally executed availability checks through Cloud Monitoring uptime checks and describes how to analyze request traffic and latency through Cloud Logging.

**QUESTION NO: 8**

You are designing a Cloud Build pipeline that deploys an application to a GKE production cluster. You want to follow the principle of least privilege and avoid using service account keys. What should you do? (Choose two.)

**A.** Create a dedicated service account for the production deployment pipeline.

**B.** Grant the dedicated service account only the IAM and Kubernetes RBAC permissions needed for deployment.

**C.** Download a key for the service account and store it as a Cloud Build substitution.

**D.** Grant the default Compute Engine service account the Owner role on the production project.

**E.** Use a developer's personal credentials in the deployment step.

**ANSWER: A B**

**Explanation:**

Create and use a dedicated service account for the production deployment pipeline. A dedicated identity separates deployment permissions from other build activities and makes permissions, audit logs, and lifecycle management specific to the production deployment workload.

Grant this service account only the required IAM and Kubernetes RBAC permissions for the target cluster and resources. Cloud Build can run builds using a user-specified service account, eliminating the need to create or distribute a service account key. The identity also requires appropriate Kubernetes RBAC authorization after it authenticates to the cluster. See Use user-specified service accounts with Cloud Build and GKE role-based access control.

## QUESTION NO: 9

Your Cloud Build pipeline deploys an application to GKE. A security review finds that the Cloud Build service account has broad project-level permissions, including permissions that are unrelated to builds and deployments. You want to apply least privilege without interrupting the pipeline. What should you do? (Choose two.)

**A.** Configure the Cloud Build trigger to use a dedicated service account.

**B.** Grant the dedicated service account only the required permissions at the narrowest applicable resource scope, then remove unrelated broad roles.

**C.** Grant the Cloud Build service account the Owner role so that future pipeline changes do not fail.

**D.** Store a project Owner service account key in Secret Manager and use it during deployment.

**E.** Grant all developers permission to impersonate the deployment service account.

**ANSWER: A B**

**Explanation:**

Create or use a dedicated service account for the Cloud Build trigger and grant it only the permissions required by the build and deployment workflow. Cloud Build supports executing builds with a user-specified service account. A dedicated identity limits the scope of permissions and makes audit records easier to attribute to the pipeline.

Grant required permissions at the narrowest applicable resource scope and remove unrelated broad roles after validating the pipeline. For example, grant Artifact Registry write access only where images are pushed, and grant the deployment identity the minimum required GKE and Kubernetes RBAC permissions. Testing the revised permissions in a nonproduction environment before removal helps avoid deployment disruption. See Use user-specified service accounts with Cloud Build and Google Cloud IAM security best practices.

## QUESTION NO: 10

You are preparing a disaster recovery plan for a critical service that uses Cloud SQL for PostgreSQL. The service has a defined recovery time objective (RTO) and recovery point objective (RPO). You want to validate that the documented recovery process can meet these objectives. What should you do? (Choose two.)

**A.** Perform scheduled recovery exercises by restoring to an isolated environment.

**B.** Measure recovery duration and recovered-data age against the RTO and RPO.

**C.** Assume that successful automated backups prove that the service can meet its RTO.

**D.** Run the recovery procedure for the first time during an actual production outage.

**E.** Reduce the RPO by deleting backups older than one day.

**ANSWER: A B**

**Explanation:**

Regularly perform a controlled recovery exercise that restores the database to an isolated environment. A recovery exercise validates that backups, required permissions, connection configuration, and operational runbooks work in practice. It also identifies gaps before an actual disaster occurs.

Measure the elapsed recovery time and the age of recovered data against the defined RTO and RPO. RTO measures how quickly service must be restored, while RPO measures the acceptable amount of data loss. Recording these results gives objective evidence of whether the recovery design and procedures meet the service requirements. See Cloud SQL backup and recovery and Google Cloud disaster recovery guidance.