GCP Professional Cloud DevOps Engineer

Duration: 32 Hours (4 Days)

Overview

The GCP Professional Cloud DevOps Engineer certification validates an individual's expertise in efficiently developing and managing reliable, scalable, and available cloud solutions using Google Cloud Platform (GCP). It focuses on the ability to leverage cloud-native practices and tools to automate and enhance the software delivery pipeline. This includes proficiency in implementing service integration, configuration management, monitoring, logging, and security practices to enhance the performance of applications and systems. Industries use this certification to ensure their DevOps engineers have a strong grasp of Google Cloud's best practices, can optimize resource utilization, and can implement continuous integration and continuous delivery (CI/CD) pipelines for faster and safer deployments.

Audience Profile

- Experienced DevOps professionals seeking expertise in Google Cloud
- IT practitioners aiming to implement DevOps practices on GCP
- Cloud solutions architects aiming to design continuous delivery pipelines
- Systems administrators transitioning to Google Cloud DevOps roles
- Developers interested in automation and orchestration on Google Cloud

Course Syllabus

Prerequisites: GCP Associate Cloud Engineer course

Course Objective: Professional Cloud DevOps Engineers implement processes

- throughout the systems development lifecycle using Google-recommended
- methodologies and tools. They build and deploy software and infrastructure delivery
- pipelines, optimize and maintain production systems and services, and balance service reliability with delivery speed.

Lab Requirement: Participant Google Cloud Account Required

Section 1: Bootstrapping a Google Cloud organization for DevOps (~17% of the exam)

1.1 Designing the overall resource hierarchy for an organization. Considerations include:

- Projects and folders
- Shared networking
- Identity and Access Management (IAM) roles and organization-level policies
- Creating and managing service accounts

1.2 Managing infrastructure as code. Considerations include:

- Infrastructure as code tooling (e.g., Cloud Foundation Toolkit, Config
- Connector, Terraform, Helm)
- Making infrastructure changes using Google-recommended practices and

- infrastructure as code blueprints
- Immutable architecture

1.3 Designing a CI/CD architecture stack in Google Cloud, hybrid, and multi-cloud

- environments. Considerations include:
- CI with Cloud Build
- CD with Google Cloud Deploy
- Widely used third-party tooling (e.g., Jenkins, Git, ArgoCD, Packer)
- Security of CI/CD tooling

1.4 Managing multiple environments (e.g., staging, production). Considerations include:

- Determining the number of environments and their purpose
- Creating environments dynamically for each feature branch with Google
- Kubernetes Engine (GKE) and Terraform
- Config Management

Section 2: Building and implementing CI/CD pipelines for a service (~23% of the exam)

2.1 Designing and managing CI/CD pipelines. Considerations include:

- Artifact management with Artifact Registry
- Deployment to hybrid and multi-cloud environments (e.g., Anthos, GKE)
- CI/CD pipeline triggers
- Testing a new application version in the pipeline
- Configuring deployment processes (e.g., approval flows)
- CI/CD of serverless applications

2.2 Implementing CI/CD pipelines. Considerations include:

- Auditing and tracking deployments (e.g., Artifact Registry, Cloud Build, Google
- Cloud Deploy, Cloud Audit Logs)
- Deployment strategies (e.g., canary, blue/green, rolling, traffic splitting)
- Rollback strategies
- Troubleshooting deployment issues

2.3 Managing CI/CD configuration and secrets. Considerations include:

- Secure storage methods and key rotation services (e.g., Cloud Key
- Management Service, Secret Manager)
- Secret management
- Build versus runtime secret injection

2.4 Securing the CI/CD deployment pipeline. Considerations include:

- Vulnerability analysis with Artifact Registry
- Binary Authorization
- IAM policies per environment

Section 3: Applying site reliability engineering practices to a service (~23% of the exam)

3.1 Balancing change, velocity, and reliability of the service. Considerations include:

- Discovering SLIs (e.g., availability, latency)
- Defining SLOs and understanding SLAs
- Error budgets
- Toil automation
- Opportunity cost of risk and reliability (e.g., number of “nines”)

3.2 Managing service lifecycle. Considerations include:

- Service management (e.g., introduction of a new service by using a pre-service
- onboarding checklist, launch plan, or deployment plan, deployment, maintenance, and retirement)
- Capacity planning (e.g., quotas and limits management)
- Autoscaling using managed instance groups, Cloud Run, Cloud Functions, or GKE
- Implementing feedback loops to improve a service

3.3 Ensuring healthy communication and collaboration for operations.

- Considerations include:
- Preventing burnout (e.g., setting up automation processes to prevent burnout)
- Fostering a culture of learning and blamelessness
- Establishing joint ownership of services to eliminate team silos

3.4 Mitigating incident impact on users. Considerations include:

- Communicating during an incident
- Draining/redirecting traffic
- Adding capacity

3.5 Conducting a postmortem. Considerations include:

- Documenting root causes
- Creating and prioritizing action items
- Communicating the postmortem to stakeholders

Section 4: Implementing service monitoring strategies (~21% of the exam)

4.1 Managing logs. Considerations include:

- Collecting structured and unstructured logs from Compute Engine, GKE, and
- serverless platforms using Cloud Logging
- Configuring the Cloud Logging agent
- Collecting logs from outside Google Cloud
- Sending application logs directly to the Cloud Logging API
- Log levels (e.g., info, error, debug, fatal)
- Optimizing logs (e.g., multiline logging, exceptions, size, cost)

4.2 Managing metrics with Cloud Monitoring. Considerations include:

- Collecting and analyzing application and platform metrics
- Collecting networking and service mesh metrics
- Using Metrics Explorer for ad hoc metric analysis
- Creating custom metrics from logs

4.3 Managing dashboards and alerts in Cloud Monitoring. Considerations include:

- Creating a monitoring dashboard
- Filtering and sharing dashboards
- Configuring alerting
- Defining alerting policies based on SLOs and SLIs
- Automating alerting policy definition using Terraform
- Using Google Cloud Managed Service for Prometheus to collect metrics and
- set up monitoring and alerting

4.4 Managing Cloud Logging platform. Considerations include:

- Enabling data access logs (e.g., Cloud Audit Logs)
- Enabling VPC Flow Logs
- Viewing logs in the Google Cloud console
- Using basic versus advanced log filters
- Logs exclusion versus logs export
- Project-level versus organization-level export
- Managing and viewing log exports
- Sending logs to an external logging platform
- Filtering and redacting sensitive data (e.g., personally identifiable information
- [PII], protected health information [PHI])

4.5 Implementing logging and monitoring access controls. Considerations include:

- Restricting access to audit logs and VPC Flow Logs with Cloud Logging
- Restricting export configuration with Cloud Logging
- Allowing metric and log writing with Cloud Monitoring

Section 5: Optimizing the service performance (~16% of the exam)

5.1 Identifying service performance issues. Considerations include:

- Using Google Cloud's operations suite to identify cloud resource utilization
- Interpreting service mesh telemetry
- Troubleshooting issues with compute resources
- Troubleshooting deploy time and runtime issues with applications
- Troubleshooting network issues (e.g., VPC Flow Logs, firewall logs, latency, network details)

5.2 Implementing debugging tools in Google Cloud. Considerations include:

- Application instrumentation
- Cloud Logging
- Cloud Trace
- Error Reporting
- Cloud Profiler
- Cloud Monitoring

5.3 Optimizing resource utilization and costs. Considerations include:

- Preemptible/Spot virtual machines (VMs)
- Committed-use discounts (e.g., flexible, resource-based)
- Sustained-use discounts
- Network tiers
- Sizing recommendations