Google Cloud Professional Cloud DevOps Engineer

Introduction

A Professional Cloud DevOps Engineer is responsible for efficient development operations that can balance service reliability and delivery speed.

They are skilled at using Google Cloud Platform to build software delivery pipelines, deploy and monitor services, and manage and learn from incidents.

The Professional Cloud DevOps Engineer exam assesses your ability to:

- Apply site reliability engineering principles to a service
- Optimize service performance
- Implement service monitoring strategies

Course Objective

- Applying site reliability engineering principles to a service
- Building and implementing CI/CD pipelines for a service
- Implementing service monitoring strategies
- Optimizing service performance
- Managing service incidents

Course Outline

1. Applying site reliability engineering principles to a service

1.1 Balance change, velocity, and reliability of the service:

- Discover SLIs (availability, latency, etc.)
- Define SLOs and understand SLAs
- Agree to consequences of not meeting the error budget
- Construct feedback loops to decide what to build next
- Toil automation

1.2 Manage service life cycle:

- Manage a service (e.g., introduce a new service, deploy it, maintain and retire it)
- Plan for capacity (e.g., quotas and limits management)

1.3 Ensure healthy communication and collaboration for operations:

- Prevent burnout (e.g., set up automation processes to prevent burnout)
- Foster a learning culture
- Foster a culture of blamelessness

2. Building and implementing CI/CD pipelines for a service

2.1 Design CI/CD pipelines:

- Immutable artifacts with Container Registry
- Artifact repositories with Container Registry

- Deployment strategies with Cloud Build, Spinnaker
- Deployment to hybrid and multi-cloud environments with Anthos, Spinnaker, Kubernetes
- Artifact versioning strategy with Cloud Build, Container Registry
- CI/CD pipeline triggers with Cloud Source Repositories, Cloud Build GitHub App, Cloud Pub/Sub
- Testing a new version with Spinnaker
- Configure deployment processes (e.g., approval flows)

2.2 Implement CI/CD pipelines:

- CI with Cloud Build
- CD with Cloud Build
- Open source tooling (e.g. Jenkins, Spinnaker, GitLab, Concourse)
- Auditing and tracing of deployments (e.g., CSR, Cloud Build, Cloud Audit Logs)

2.3 Manage configuration and secrets:

- Secure storage methods
- Secret rotation and config changes

2.4 Manage infrastructure as code:

- Terraform / Cloud Deployment Manager
- Infrastructure code versioning
- Make infrastructure changes safer
- Immutable architecture

2.5 Deploy CI/CD tooling:

- Centralized tools vs. multiple tools (single vs multi-tenant)
- Security of CI/CD tooling

2.6 Manage different development environments (e.g., staging, production, etc.):

- Decide on the number of environments and their purpose
- Create environments dynamically per feature branch with GKE, Cloud Deployment Manager
- Local development environments with Docker, Cloud Code, Skaffold

2.7 Secure the deployment pipeline:

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

3. Implementing service monitoring strategies

3.1 Manage application logs:

- Collecting logs from Compute Engine, GKE with Stackdriver Logging, Fluentd
- Collecting third-party and structured logs with Stackdriver Logging, Fluentd
- Sending application logs directly to Stackdriver API with Stackdriver Logging

3.2 Manage application metrics with Stackdriver Monitoring:

- Collecting metrics from Compute Engine
- Collecting GKE/Kubernetes metrics
- Use metric explorer for ad hoc metric analysis

3.3 Manage Stackdriver Monitoring platform:

- Creating a monitoring dashboard
- Filtering and sharing dashboards
- Configure third-party alerting in Stackdriver Monitoring (i.e., PagerDuty, Slack, etc.)
- Define alerting policies based on SLIs with Stackdriver Monitoring
- Automate alerting policy definition with Cloud DM or Terraform
- Implementing SLO monitoring and alerting with Stackdriver Monitoring
- Understand Stackdriver Monitoring integrations (e.g., Grafana, BigQuery)

- Using SIEM tools to analyze audit/flow logs (e.g., Splunk, Datadog)
- Design Stackdriver Workspace strategy

3.4 Manage Stackdriver Logging platform:

- Enabling data access logs (e.g., Cloud Audit Logs)
- Enabling VPC flow logs
- Viewing logs in the GCP Console
- Using basic vs. advanced logging filters
- Implementing logs-based metrics
- Understanding the logging exclusion vs. logging export
- Selecting the options for logging export
- Implementing a project-level / org-level export
- Viewing export logs in Cloud Storage and BigQuery
- Sending logs to an external logging platform

3.5 Implement logging and monitoring access controls:

- Set ACL to restrict access to audit logs with IAM, Stackdriver Logging
- Set ACL to restrict export configuration with IAM, Stackdriver Logging
- Set ACL to allow metric writing for custom metrics with IAM, Stackdriver Monitoring

4. Optimizing service performance

4.1 Identify service performance issues:

- Evaluate and understand user impact (Stackdriver Service Monitoring for App Engine, Istio)
- Utilize Stackdriver to identify cloud resource utilization
- Utilize Stackdriver Trace/Profiler to profile performance characteristics
- Interpret service mesh telemetry
- Troubleshoot issues with the image/OS
- Troubleshoot network issues (e.g., VPC flow logs, firewall logs, latency, view network details)

4.2 Debug application code:

- Application instrumentation
- Stackdriver Debugger
- Stackdriver Logging
- Stackdriver Trace
- Debugging distributed applications
- App Engine local development server
- Stackdriver Error Reporting
- Stackdriver Profiler

4.3 Optimize resource utilization:

- Identify resource costs
- Identify resource utilization levels
- Develop a plan to optimize areas of greatest cost or lowest utilization
- Manage preemptible VMs
- Work with committed-use discounts
- TCO considerations
- Consider network pricing

5. Managing service incidents

5.1 Coordinate roles and implement communication channels during a service incident:

- Define roles (incident commander, communication lead, operations lead)
- Handle requests for impact assessment
- Provide regular status updates, internal and external
- Record major changes in incident state (When mitigated? When all clear? etc.)
- Establish communications channels (email, IRC, Hangouts, Slack, phone, etc.)
- Scaling response team and delegation
- Avoid exhaustion/burnout
- Rotate / hand over roles
- Manage stakeholder relationships

5.2 Investigate incident symptoms impacting users:

- Identify probable causes of service failure
- Evaluate symptoms against probable causes; rank probability of cause based on observed behavior
- Perform investigation to isolate the most likely actual cause
- Identify alternatives to mitigate the issue

5.3 Mitigate incident impact on users:

- Rollback release
- Drain/redirect traffic
- Turn off experiment
- Add capacity

5.4 Resolve issues (e.g., Cloud Build, Jenkins):

- Code change / fix bug
- Verify fix
- Declare all-clear

5.5 Document issue in a postmortem:

- Document root causes
- Create and prioritize action items
- Communicate postmortem to stakeholders

Prerequisites

- Basic understanding of cloud concepts (virtual machines, containers, networking, etc)
- General knowledge of IT architecture
- Familiarity with DevOps practices and principles

Recommended Experience

Three+ years of industry experience including one+ years managing solutions on GCP.

Target Audience

- DevOps Engineers who want to build/maintain infrastructure on Google Cloud Platform
- People preparing for the Google Professional Cloud DevOps Engineer exam
- Developer
- Software Engineer
- Automation Engineer
- Cloud Developer
- Site Reliability Engineer (SRE)
- Test Development Engineer
- Security Automation Engineer

Duration

3 days training course