Google GCP-PCDE

GOOGLE PROFESSIONAL CLOUD DEVOPS ENGINEER CERTIFICATION QUESTIONS & ANSWERS

Exam Summary – Syllabus – Questions

GCP-PCDE

Google Cloud Platform - Professional Cloud DevOps Engineer (GCP-PCDE)

50-60 Questions Exam – 70% Cut Score – Duration of 120 minutes

www.VMExam.com

Table of Contents

Know Your GCP-PCDE Certification Well:

The GCP-PCDE is best suitable for candidates who want to gain knowledge in the Google Professional. Before you start your GCP-PCDE preparation you may struggle to get all the crucial Professional Cloud DevOps Engineer materials like GCP-PCDE syllabus, sample questions, study guide.

But don't worry the GCP-PCDE PDF is here to help you prepare in a stress free manner.

The PDF is a combination of all your queries like-

- What is in the GCP-PCDE syllabus?
- How many questions are there in the GCP-PCDE exam?
- Which Practice test would help me to pass the GCP-PCDE exam at the first attempt?

Passing the GCP-PCDE exam makes you Google Cloud Platform - Professional Cloud DevOps Engineer (GCP-PCDE). Having the Professional Cloud DevOps Engineer certification opens multiple opportunities for you. You can grab a new job, get a higher salary or simply get recognition within your current organization.

Google GCP-PCDE Professional Cloud DevOps Engineer Certification Details:

Exam Name	Google Professional Cloud DevOps Engineer (GCP-PCDE)
Exam Code	GCP-PCDE
Exam Price	$200 USD
Duration	120 minutes
Number of Questions	50-60
Passing Score	Pass / Fail (Approx 70%)
Recommended Training / Books	**Google Cloud documentation** **Google Cloud solutions**
Schedule Exam	**Google Cloud Webassessor**
Sample Questions	**Google GCP-PCDE Sample Questions**
Recommended Practice	**Google Cloud Platform - Professional Cloud DevOps Engineer (GCP-PCDE) Practice Test**

GCP-PCDE Syllabus:

Section	Objectives
Bootstrapping a Google Cloud organization for DevOps	
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
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
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
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 - Anthos Config Management
Building and implementing CI/CD pipelines for a service	
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
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

Section	Objectives
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
Securing the CI/CD deployment pipeline. Considerations include:	- Vulnerability analysis with Artifact Registry - Binary Authorization - IAM policies per environment
Applying site reliability engineering practices to a service	
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")
Managing service lifecycle. Considerations include:	- Service management (e.g., introduction of a new service by using pre-mortems [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
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
Mitigating incident impact on users. Considerations include:	- Communicating during an incident - Draining/redirecting traffic - Adding capacity
Conducting a postmortem. Considerations include:	- Documenting root causes - Creating and prioritizing action items - Communicating the postmortem to stakeholders

Section	Objectives
Implementing service monitoring strategies	
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)
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
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
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])
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
Optimizing service performance	
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

Section	Objectives
	- Troubleshooting deploy time and runtime issues with applications - Troubleshooting network issues (e.g., VPC Flow Logs, firewall logs, latency, network details)
Implementing debugging tools in Google Cloud. Considerations include:	- Application instrumentation - Cloud Logging - Cloud Trace - Error Reporting - Cloud Profiler - Cloud Monitoring
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

Google GCP-PCDE Sample Questions:

Question: 1

You have a Compute Engine instance that uses the default Debian image. The application hosted on this instance recently suffered a series of crashes that you weren't able to debug in real time: the application process died suddenly every time.

The application usually consumes 50% of the instance's memory, and normally never more than 70%, but you suspect that a memory leak was responsible for the crashes. You want to validate this hypothesis.

What should you do?

a) Go to Metrics Explorer and look for the "compute.googleapis.com/guest/system/problem_count" metric for that instance. Examine its value for when the application crashed in the past.
b) In Cloud Monitoring, create an uptime check for your application. Create an alert policy for that uptime check to be notified when your application crashes. When you receive an alert, use your usual debugging tools to investigate the behavior of the application in real time.
c) Install the Cloud Monitoring agent on the instance. Go to Metrics Explorer and look for the "agent.googleapis.com/memory/percent_used" metric for that instance. Examine its value for when the application crashed in the past.
d) Install the Cloud Monitoring agent on the instance. Create an alert policy on the "agent.googleapis.com/memory/percent_used" metric for that instance to be alerted when the memory used is higher than 75%. When you receive an alert, use your usual debugging tools to investigate the behavior of the application in real time.

Answer: d

Question: 2

You support a Python application running in production on Compute Engine. You want to debug some of the application code by inspecting the value of a specific variable. What should you do?

a) Create a Cloud Debugger logpoint with the variable at a specific line location in your application's source code, and view the value in the Logs Viewer.
b) Use your local development environment and code editor to set up a breakpoint in the source code, run the application locally, and then inspect the value of the variable.
c) Modify the source code of the application to log the value of the variable, deploy to the development environment, and then run the application to capture the value in Cloud Logging.
d) Create a Cloud Debugger snapshot at a specific line location in your application's source code, and view the value of the variable in the Google Cloud Console.

Answer: d

Question: 3

You are running a production application on Compute Engine. You want to monitor the key metrics of CPU, Memory, and Disk I/O time.

You want to ensure that the metrics are visible by the team and will be explorable if an issue occurs. What should you do? (Choose 2)

a) Set up logs-based metrics based on your application logs to identify errors.
b) Export key metrics to a Google Cloud Function and then analyze them for outliers.
c) Set up alerts in Cloud Monitoring for key metrics breaching defined thresholds.
d) Create a Dashboard with key metrics and indicators that can be viewed by the team.
e) Export key metrics to BigQuery and then run hourly queries on the metrics to identify outliers.

Answer: c, d

Question: 4

You have an application deployed on Google Kubernetes Engine (GKE). The application logs are captured by Cloud Logging. You need to remove sensitive data before it reaches the Cloud Logging API.

What should you do?

a) Customize the GKE clusters' Fluentd configuration with a filter rule. Update the Fluentd Config Map and Daemon Set in the GKE cluster.
b) Write the log information to the container file system. Execute a second process inside the container that will filter the sensitive information before writing to Standard Output.
c) Configure a filter in the Cloud Logging UI to exclude the logs with sensitive data.
d) Configure BigQuery as a sink for the logs from Cloud Logging, and then create a Data Loss Prevention job.

Answer: a

Question: 5

Several teams in your company want to use Cloud Build to deploy to their own Google Kubernetes Engine (GKE) clusters.

The clusters are in projects that are dedicated to each team. The teams only have access to their own projects. One team should not have access to the cluster of another team.

You are in charge of designing the Cloud Build setup, and want to follow Google-recommended practices. What should you do?

a) Limit each team member's access so that they only have access to their team's clusters. Ask each team member to install the gcloud CLI and to authenticate themselves by running "gcloud init". Ask each team member to execute Cloud Build builds by using "gcloud builds submit".
b) Create a single project for Cloud Build that all the teams will use. List the service accounts in this project and identify the one used by Cloud Build. Grant the Kubernetes Engine Developer IAM role to that service account in each team's project.
c) In each team's project, list the service accounts and identify the one used by Cloud Build for each project. In each project, grant the Kubernetes Engine Developer IAM role to the service account used by Cloud Build. Ask each team to execute Cloud Build builds in their own project.
d) In each team's project, create a service account, download a JSON key for that service account, and grant the Kubernetes Engine Developer IAM role to that service account in that project. Create a single project for Cloud Build that all the teams will use. In that project, encrypt all the service account keys by using Cloud KMS. Grant the Cloud KMS CryptoKey Decrypter IAM role to Cloud Build's service account. Ask each team to include in their "cloudbuild.yaml" files a step that decrypts the key of their service account, and use that key to connect to their cluster.

Answer: c

Question: 6

Your application runs in Google Kubernetes Engine (GKE). You want to use Spinnaker with the Kubernetes Provider to perform blue/green deployments and control which version of the application receives traffic. What should you do?

a) Use a Kubernetes Replica Set and use Spinnaker to create a new service for each new version of the application to be deployed.
b) Use a Kubernetes Replica Set and use Spinnaker to update the Replica Set for each new version of the application to be deployed.
c) Use a Kubernetes Deployment and use Spinnaker to update the deployment for each new version of the application to be deployed.
d) Use a Kubernetes Deployment and use Spinnaker to create a new deployment object for each new version of the application to be deployed.

Answer: b

Question: 7

You are deploying an application to a Kubernetes cluster that requires a username and password to connect to another service.

When you deploy the application, you want to ensure that the credentials are used securely in multiple environments with minimal code changes.

What should you do?

a) Bundle the credentials with the code inside the container and secure the container registry.
b) Leverage a CI/CD pipeline to update the variables at build time and inject them into a templated Kubernetes application manifest.
c) Store the credentials as a Kubernetes Secret and let the application access it via environment variables at runtime.
d) Store the credentials as a Kubernetes ConfigMap and let the application access it via environment variables at runtime.

Answer: c

Question: 8

You support a website with a global audience. The website has a frontend web service and a backend database service that runs on different clusters. All clusters are scaled to handle at least ⅓ of the total user traffic.

You use 4 different regions in Google Cloud and Cloud Load Balancing to direct traffic to a region closer to the user.

You are applying a critical security patch to the backend database. You successfully patch the database in the first 2 regions, but you make a configuration error while patching Region 3. The unsuccessful patching causes 50% of user requests to Region 3 to time out.

You want to mitigate the impact of unsuccessful patching on users. What should you do?

a) Add more capacity to the frontend of Region 3.
b) Revert the Region 3 backend database and run it without the patch.
c) Drain the requests to Region 3 and redirect new requests to other regions.
d) Back up the database in the backend of Region 3 and restart the database.

Answer: c

Question: 9

You work with a video rendering application that publishes small tasks as messages to a Cloud Pub/Sub topic. You need to deploy the application that will execute these tasks on multiple virtual machines (VMs).

Each task takes less than 1 hour to complete. The rendering is expected to be completed within a month. You need to minimize rendering costs.

What should you do?

a) Deploy the application as a managed instance group with Preemptible VMs.
b) Deploy the application as a managed instance group. Configure a Committed Use Discount for the amount of CPU and memory required.
c) Deploy the application as a managed instance group.
d) Deploy the application as a managed instance group with Preemptible VMs. Configure a Committed Use Discount for the amount of CPU and memory required.

Answer: a

Question: 10

Your Site Reliability Engineering team does toil work to archive unused data in tables within your application's relational database. This toil is required to ensure that your application has a low Latency Service Level Indicator (SLI) to meet your Service Level Objective (SLO).

Toil is preventing your team from focusing on a high-priority engineering project that will improve the Availability SLI of your application.

You want to: (1) reduce repetitive tasks to avoid burnout, (2) improve organizational efficiency, and (3) follow the Site Reliability Engineering recommended practices.

What should you do?

a) Identify repetitive tasks that contribute to toil and onboard additional team members for support.
b) Identify repetitive tasks that contribute to toil and automate them.
c) Change the SLO of your Latency SLI to accommodate toil being done less often. Use this capacity to work on the Availability SLI engineering project.
d) Assign the Availability SLI engineering project to the Software Engineering team.

Answer: b

Study Guide to Crack Google Professional Cloud DevOps Engineer GCP-PCDE Exam:

- Getting details of the GCP-PCDE syllabus, is the first step of a study plan. This pdf is going to be of ultimate help. Completion of the syllabus is must to pass the GCP-PCDE exam.
- Making a schedule is vital. A structured method of preparation leads to success. A candidate must plan his schedule and follow it rigorously to attain success.
- Joining the Google provided training for GCP-PCDE exam could be of much help. If there is specific training for the exam, you can discover it from the link above.
- Read from the GCP-PCDE sample questions to gain your idea about the actual exam questions. In this PDF useful sample questions are provided to make your exam preparation easy.
- Practicing on GCP-PCDE practice tests is must. Continuous practice will make you an expert in all syllabus areas.