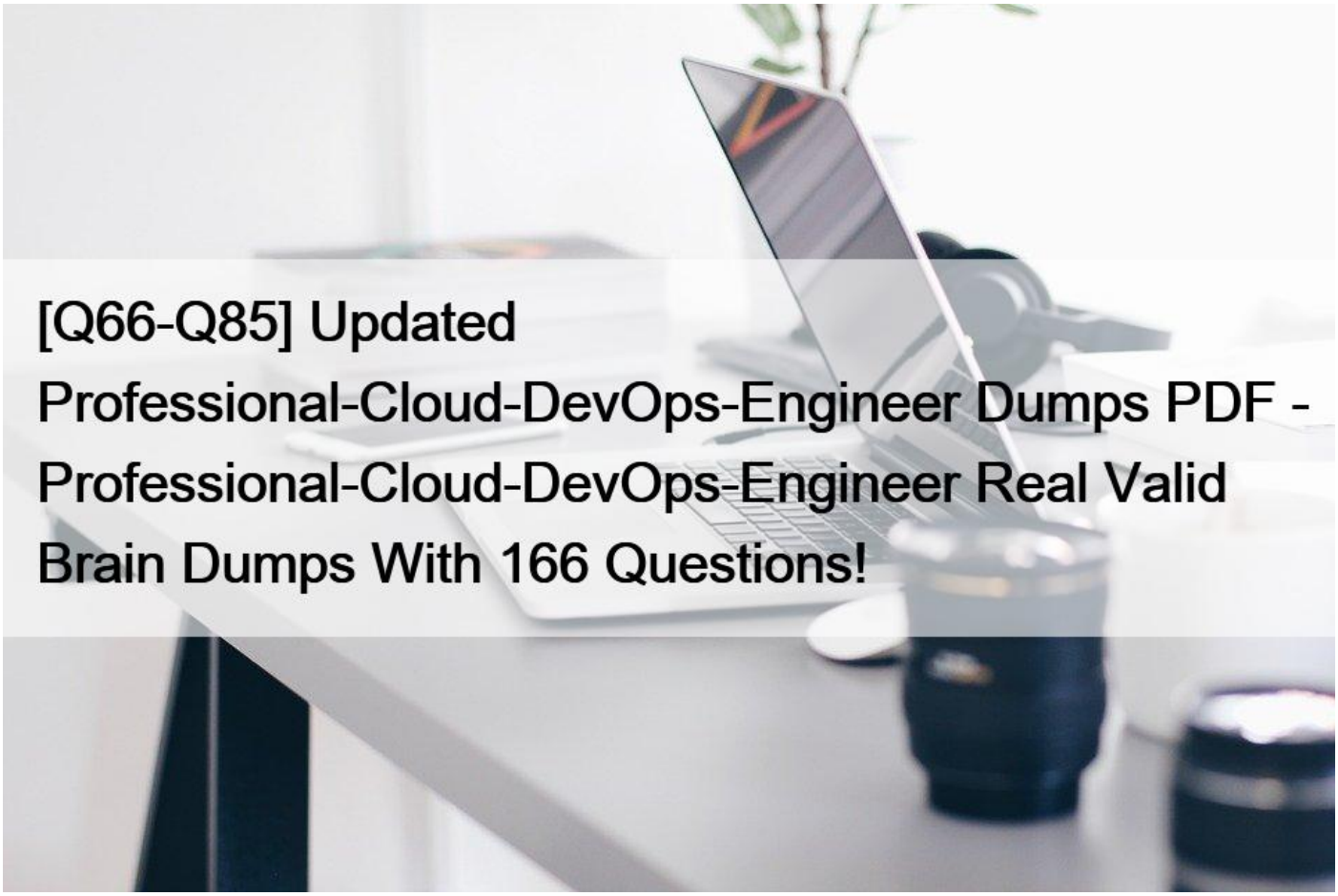
[Q66-Q85 Updated Professional-Cloud-DevOps-Engineer Dumps PDF - Professional-Cloud-DevOps-Engineer Real Valid Brain Dumps With 166 Questions!



Updated Professional-Cloud-DevOps-Engineer Dumps PDF - Professional-Cloud-DevOps-Engineer Real Valid Brain Dumps With 166 Questions! 100% Free Professional-Cloud-DevOps-Engineer Exam Dumps Use Real Cloud DevOps Engineer Dumps

The Google Cloud Certified - Professional Cloud DevOps Engineer Exam certification exam consists of multiple-choice questions and simulations that test the candidate's ability to apply their knowledge in real-world scenarios. Candidates must demonstrate their ability to design and implement solutions using Google Cloud Platform services such as App Engine, Kubernetes, Cloud Functions, and Cloud Storage. They must also demonstrate their ability to use tools such as Stackdriver, Cloud Logging, and Cloud Monitoring to manage and monitor their applications and services.

Q66. You support an application running on GCP and want to configure SMS notifications to your team for the most critical alerts in Stackdriver Monitoring. You have already identified the alerting policies you want to configure this for. What should you do?
* Download and configure a third-party integration between Stackdriver Monitoring and an SMS gateway.

Ensure that your team members add their SMS/phone numbers to the external tool.
* Select the Webhook notifications option for each alerting policy, and configure it to use a third-party integration tool. Ensure that your team members add their SMS/phone numbers to the external tool.
* Ensure that your team members set their SMS/phone numbers in their Stackdriver Profile. Select the SMS notification option for each alerting policy and then select the appropriate SMS/phone numbers from the list.
* Configure a Slack notification for each alerting policy. Set up a Slack-to-SMS integration to send SMS messages when Slack messages are received. Ensure that your team members add their SMS/phone numbers to the external integration.

Q67. You are designing a system with three different environments: development, quality assurance (QA), and production.

Each environment will be deployed with Terraform and has a Google Kubemetes Engine (GKE) cluster created so that application teams can deploy their applications. Anthos Config Management will be used and templated to deploy infrastructure level resources in each GKE cluster. All users (for example, infrastructure operators and application owners) will use GitOps. How should you structure your source control repositories for both Infrastructure as Code (laC) and application code?
* Cloud Infrastructure (Terraform) repository is shared: different directories are different environments GKE Infrastructure (Anthos Config Management Kustomize manifests) repository is shared: different overlay directories are different environments Application (app source code) repositories are separated: different branches are different features
* Cloud Infrastructure (Terraform) repository is shared: different directories are different environments GKE Infrastructure (Anthos Config Management Kustomize manifests) repositories are separated:

different branches are different environments

Application (app source code) repositories are separated: different branches are different features
* Cloud Infrastructure (Terraform) repository is shared: different branches are different environments GKE Infrastructure (Anthos Config Management Kustomize manifests) repository is shared: different overlay directories are different environments Application (app source code) repository is shared: different directories are different features
* Cloud Infrastructure (Terraform) repositories are separated: different branches are different environments GKE Infrastructure (Anthos Config Management Kustomize manifests) repositories are separated:

different overlay directories are different environments

Application (app source code) repositories are separated: different branches are different features
The correct answer is B, Cloud Infrastructure (Terraform) repository is shared: different directories are different environments. GKE Infrastructure (Anthos Config Management Kustomize manifests) repositories are separated: different branches are different environments. Application (app source code) repositories are separated: different branches are different features.

This answer follows the best practices for using Terraform and Anthos Config Management with GitOps, as described in the following sources:

For Terraform, it is recommended to use a single repository for all environments, and use directories to separate them. This way, you can reuse the same Terraform modules and configurations across environments, and avoid code duplication and drift. You can also use Terraform workspaces to isolate the state files for each environment12.

For Anthos Config Management, it is recommended to use separate repositories for each environment, and use branches to separate the clusters within each environment. This way, you can enforce different policies and configurations for each environment, and use pull requests to promote changes across environments. You can also use Kustomize to create overlays for each cluster that apply specific patches or customizations34.

For application code, it is recommended to use separate repositories for each application, and use branches to separate the features or bug fixes for each application. This way, you can isolate the development and testing of each application, and use pull requests to

merge changes into the main branch. You can also use tags or labels to trigger deployments to different environments5 .

Reference:

1: Best practices for using Terraform | Google Cloud

2: Terraform Recommended Practices – Part 1 | Terraform – HashiCorp Learn

3: Deploy Anthos on GKE with Terraform part 1: GitOps with Config Sync | Google Cloud Blog

4: Using Kustomize with Anthos Config Management | Anthos Config Management Documentation | Google Cloud

5: Deploy Anthos on GKE with Terraform part 3: Continuous Delivery with Cloud Build | Google Cloud Blog

6: GitOps-style continuous delivery with Cloud Build | Cloud Build Documentation | Google Cloud

Q68. 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 2 answers
* Your application servers’ logs
* Instrumentation coded directly in the client
* Metrics exported from the application servers
* GKE health checks for your application servers
* A synthetic client that periodically sends simulated user requests

Q69. Some of your production services are running in Google Kubernetes Engine (GKE) in the eu-west-1 region. Your build system runs in the us-west-1 region. You want to push the container images from your build system to a scalable registry to maximize the bandwidth for transferring the images to the cluster. What should you do?
* Push the images to Google Container Registry (GCR) using the gcr.io hostname.
* Push the images to Google Container Registry (GCR) using the us.gcr.io hostname.
* Push the images to Google Container Registry (GCR) using the eu.gcr.io hostname.
* Push the images to a private image registry running on a Compute Engine instance in the eu-west-1 region.

Q70. Your application runs on Google Cloud Platform (GCP). You need to implement Jenkins for deploying application releases to GCP. You want to streamline the release process, lower operational toil, and keep user data secure. What should you do?
* Implement Jenkins on local workstations.
* Implement Jenkins on Kubernetes on-premises
* Implement Jenkins on Google Cloud Functions.
* Implement Jenkins on Compute Engine virtual machines.
Explanation

Your application runs on Google Cloud Platform (GCP). You need to implement Jenkins for deploying application releases to GCP. You want to streamline the release process, lower operational toil, and keep user data secure. What should you do?

https://plugins.jenkins.io/google-compute-engine/

Q71. You manage several production systems that run on Compute Engine in the same Google Cloud Platform (GCP) project. Each system has its own set of dedicated Compute Engine instances. You want to know how must it costs to run each of the systems. What should you do?
* In the Google Cloud Platform Console, use the Cost Breakdown section to visualize the costs per system.
* Assign all instances a label specific to the system they run. Configure BigQuery billing export and query costs per label.
* Enrich all instances with metadata specific to the system they run. Configure Stackdriver Logging to export to BigQuery, and query costs based on the metadata.
* Name each virtual machine (VM) after the system it runs. Set up a usage report export to a Cloud Storage bucket. Configure the bucket as a source in BigQuery to query costs based on VM name.
Explanation

https://cloud.google.com/billing/docs/how-to/export-data-bigquery

Q72. You are managing the production deployment to a set of Google Kubernetes Engine (GKE) clusters. You want to make sure only images which are successfully built by your trusted CI/CD pipeline are deployed to production. What should you do?
* Enable Cloud Security Scanner on the clusters.
* Enable Vulnerability Analysis on the Container Registry.
* Set up the Kubernetes Engine clusters as private clusters.
* Set up the Kubernetes Engine clusters with Binary Authorization.

Q73. You are monitoring a service that uses n2-standard-2 Compute Engine instances that serve large files. Users have reported that downloads are slow. Your Cloud Monitoring dashboard shows that your VMS are running at peak network throughput. You want to improve the network throughput performance. What should you do?
* Deploy a Cloud NAT gateway and attach the gateway to the subnet of the VMS.
* Add additional network interface controllers (NICs) to your VMS.
* Change the machine type for your VMS to n2-standard-8.
* Deploy the Ops Agent to export additional monitoring metrics.
The correct answer is C, Change the machine type for your VMs to n2-standard-8.

According to the Google Cloud documentation, the network throughput performance of a Compute Engine VM depends on its machine type1. The n2-standard-2 machine type has a maximum egress bandwidth of 4 Gbps, which can be a bottleneck for serving large files. By changing the machine type to n2-standard-8, you can increase the maximum egress bandwidth to 16 Gbps, which can improve the network throughput performance and reduce the download time for users. You also need to enable per VM Tier_1 networking performance, which is a feature that allows VMs to achieve higher network performance than the default settings2.

The other options are incorrect because they do not improve the network throughput performance of your VMs. Option A is incorrect because Cloud NAT is a service that allows private IP addresses to access the internet, but it does not increase the network bandwidth or speed3. Option B is incorrect because adding additional network interfaces (NICs) or IP addresses per NIC does not increase ingress or egress bandwidth for a VM1. Option D is incorrect because deploying the Ops Agent can help you monitor and troubleshoot your VMs, but it does not affect the network throughput performance4.

Reference:

Cloud NAT overview, Cloud NAT overview. Network bandwidth, Bandwidth summary. Installing the Ops Agent, Installing the Ops Agent. Configure per VM Tier_1 networking performance, Configure per VM Tier_1 networking performance.

Q74. Your application images are built wing Cloud Build and pushed to Google Container Registry (GCR). You want to be able to specify a particular version of your application for deployment based on the release version tagged in source control. What would you do when you push the image?
* Reference the image digest in the source control tag.

* Supply the source control tag as a parameter within the image name.
* Use Cloud Build to include the release version tag in the application image.
* Use GCR digest versioning to match the image to the tag in source control.

Q75. You recently migrated an ecommerce application to Google Cloud. You now need to prepare the application for the upcoming peak traffic season. You want to follow Google-recommended practices. What should you do first to prepare for the busy season?
* Migrate the application to Cloud Run, and use autoscaling.
* Load test the application to profile its performance for scaling.
* Create a Terraform configuration for the application’s underlying infrastructure to quickly deploy to additional regions.
* Pre-provision the additional compute power that was used last season, and expect growth.

Explanation

The first thing you should do to prepare your ecommerce application for the upcoming peak traffic season is to load test the application to profile its performance for scaling. Load testing is a process of simulating high traffic or user demand on your application and measuring how it responds. Load testing can help you identify any bottlenecks, errors, or performance issues that might affect your application during the busy season1. Load testing can also help you determine the optimal scaling strategy for your application, such as horizontal scaling (adding more instances) or vertical scaling (adding more resources to each instance)2.

There are different tools and methods for load testing your ecommerce application on Google Cloud, depending on the type and complexity of your application. For example, you can use Cloud Load Balancing to distribute traffic across multiple instances of your application, and use Cloud Monitoring to measure the latency, throughput, and error rate of your application3. You can also use Cloud Functions or Cloud Run to create serverless load generators that can simulate user requests and send them to your application4.

Alternatively, you can use third-party tools such as Apache JMeter or Locust to create and run load tests on your application.

By load testing your ecommerce application before the peak traffic season, you can ensure that your application is ready to handle the expected load and provide a good user experience. You can also use the results of your load tests to plan and implement other steps to prepare your application for the busy season, such as migrating to a more scalable platform, creating a Terraform configuration for deploying to additional regions, or pre-provisioning additional compute power.

References:

1: Load Testing 101: How To Test Website Performance | BlazeMeter

2: Scaling applications | Google Cloud

3: Load testing using Google Cloud | Solutions | Google Cloud

4: Serverless load testing using Cloud Functions | Solutions | Google Cloud

Q76. You work for a global organization and are running a monolithic application on Compute Engine You need to select the machine type for the application to use that optimizes CPU utilization by using the fewest number of steps You want to use historical system metncs to identify the machine type for the application to use You want to follow Google-recommended practices What should you do?
* Use the Recommender API and apply the suggested recommendations
* Create an Agent Policy to automatically install Ops Agent in all VMs
* Install the Ops Agent in a fleet of VMs by using the gcloud CLI
* Review the Cloud Monitoring dashboard for the VM and choose the machine type with the lowest CPU utilization

The best option for selecting the machine type for the application to use that optimizes CPU utilization by using the fewest number

of steps is to use the Recommender API and apply the suggested recommendations. The Recommender API is a service that provides recommendations for optimizing your Google Cloud resources, such as Compute Engine instances, disks, and firewalls. You can use the Recommender API to get recommendations for changing the machine type of your Compute Engine instances based on historical system metrics, such as CPU utilization. You can also apply the suggested recommendations by using the Recommender API or Cloud Console. This way, you can optimize CPU utilization by using the most suitable machine type for your application with minimal effort.

Q77. You want to share a Cloud Monitoring custom dashboard with a partner team What should you do?
* Provide the partner team with the dashboard URL to enable the partner team to create a copy of the dashboard
* Export the metrics to BigQuery Use Looker Studio to create a dashboard, and share the dashboard with the partner team
* Copy the Monitoring Query Language (MQL) query from the dashboard; and send the MQL query to the partner team
* Download the JSON definition of the dashboard, and send the JSON file to the partner team
Explanation

The best option for sharing a Cloud Monitoring custom dashboard with a partner team is to provide the partner team with the dashboard URL to enable the partner team to create a copy of the dashboard. A Cloud Monitoring custom dashboard is a dashboard that allows you to create and customize charts and widgets to display metrics, logs, and traces from your Google Cloud resources and applications. You can share a custom dashboard with a partner team by providing them with the dashboard URL, which is a link that allows them to view the dashboard in their browser. The partner team can then create a copy of the dashboard in their own project by using the Copy Dashboard option. This way, they can access and modify the dashboard without affecting the original one.

Q78. You support a trading application written in Python and hosted on App Engine flexible environment. You want to customize the error information being sent to Stackdriver Error Reporting. What should you do?
* Install the Stackdriver Error Reporting library for Python, and then run your code on a Compute Engine VM.
* Install the Stackdriver Error Reporting library for Python, and then run your code on Google Kubernetes Engine.
* Install the Stackdriver Error Reporting library for Python, and then run your code on App Engine flexible environment.
* Use the Stackdriver Error Reporting API to write errors from your application to ReportedErrorEvent, and then generate log entries with properly formatted error messages in Stackdriver Logging.

Q79. You have a CI/CD pipeline that uses Cloud Build to build new Docker images and push them to Docker Hub.

You use Git for code versioning. After making a change in the Cloud Build YAML configuration, you notice that no new artifacts are being built by the pipeline. You need to resolve the issue following Site Reliability Engineering practices. What should you do?
* Disable the CI pipeline and revert to manually building and pushing the artifacts.
* Change the CI pipeline to push the artifacts to Container Registry instead of Docker Hub.
* Upload the configuration YAML file to Cloud Storage and use Error Reporting to identify and fix the issue.
* Run a Git compare between the previous and current Cloud Build Configuration files to find and fix the bug.
Explanation

“After making a change in the Cloud Build YAML configuration, you notice that no new artifacts are being built by the pipeline”- means something wrong on the recent change not with the image registry.

Q80. Your company is developing applications that are deployed on Google Kubernetes Engine (GKE). Each team manages a different application. You need to create the development and production environments for each team, while minimizing costs. Different teams should not be able to access other teams’ environments. What should you do?
* Create one GCP Project per team. In each project, create a cluster for Development and one for Production. Grant the teams IAM access to their respective clusters.
* Create one GCP Project per team. In each project, create a cluster with a Kubernetes namespace for Development and one for Production. Grant the teams IAM access to their respective clusters.
* Create a Development and a Production GKE cluster in separate projects. In each cluster, create a Kubernetes namespace per

team, and then configure Identity Aware Proxy so that each team can only access its own namespace.
* Create a Development and a Production GKE cluster in separate projects. In each cluster, create a Kubernetes namespace per team, and then configure Kubernetes Role-based access control (RBAC) so that each team can only access its own namespace.
Explanation

https://cloud.google.com/architecture/prep-kubernetes-engine-for-prod#roles_and_groups

Q81. You support a high-traffic web application and want to ensure that the home page loads in a timely manner. As a first step, you decide to implement a Service Level Indicator (SLI) to represent home page request latency with an acceptable page load time set to 100 ms. What is the Google-recommended way of calculating this SLI?
* Buckelize Ihe request latencies into ranges, and then compute the percentile at 100 ms.
* Bucketize the request latencies into ranges, and then compute the median and 90th percentiles.
* Count the number of home page requests that load in under 100 ms, and then divide by the total number of home page requests.
* Count the number of home page requests that load in under 100 ms. and then divide by the total number of all web application requests.
https://sre.google/workbook/implementing-slos/

In the SRE principles book, it’s recommended treating the SLI as the ratio of two numbers: the number of good events divided by the total number of events. For example: Number of successful HTTP requests / total HTTP requests (success rate)

Q82. Your company follows Site Reliability Engineering practices. You are the person in charge of Communications for a large, ongoing incident affecting your customer-facing applications. There is still no estimated time for a resolution of the outage. You are receiving emails from internal stakeholders who want updates on the outage, as well as emails from customers who want to know what is happening. You want to efficiently provide updates to everyone affected by the outage. What should you do?
* Focus on responding to internal stakeholders at least every 30 minutes. Commit to “next update” times.
* Provide periodic updates to all stakeholders in a timely manner. Commit to a “next update” time in all communications.
* Delegate the responding to internal stakeholder emails to another member of the Incident Response Team. Focus on providing responses directly to customers.
* Provide all internal stakeholder emails to the Incident Commander, and allow them to manage internal communications. Focus on providing responses directly to customers.
When disaster strikes, the person who declares the incident typically steps into the IC role and directs the high-level state of the incident. The IC concentrates on the 3Cs and does the following: Commands and coordinates the incident response, delegating roles as needed. By default, the IC assumes all roles that have not been delegated yet. Communicates effectively. Stays in control of the incident response. Works with other responders to resolve the incident. https://sre.google/workbook/incident-response/

Q83. You are responsible for creating and modifying the Terraform templates that define your Infrastructure. Because two new engineers will also be working on the same code, you need to define a process and adopt a tool that will prevent you from overwriting each other’s code. You also want to ensure that you capture all updates in the latest version. What should you do?
* * Store your code in a Git-based version control system.

* Establish a process that allows developers to merge their own changes at the end of each day.

* Package and upload code lo a versioned Cloud Storage bucket as the latest master version.
* * Store your code in a Git-based version control system.

* Establish a process that includes code reviews by peers and unit testing to ensure integrity and functionality before integration of code.

* Establish a process where the fully integrated code in the repository becomes the latest master version.
* * Store your code as text files in Google Drive in a defined folder structure that organizes the files.

* At the end of each day. confirm that all changes have been captured in the files within the folder structure.

* Rename the folder structure with a predefined naming convention that increments the version.
* * Store your code as text files in Google Drive in a defined folder structure that organizes the files.

* At the end of each day, confirm that all changes have been captured in the files within the folder structure and create a new .zip archive with a predefined naming convention.

* Upload the .zip archive to a versioned Cloud Storage bucket and accept it as the latest version.

Q84. You deploy a new release of an internal application during a weekend maintenance window when there is minimal user tragic. After the window ends, you learn that one of the new features isn’t working as expected in the production environment. After an extended outage, you roll back the new release and deploy a fix. You want to modify your release process to reduce the mean time to recovery so you can avoid extended outages in the future. What should you do? (Choose two.)
* Before merging new code, require 2 different peers to review the code changes.
* Adopt the blue/green deployment strategy when releasing new code via a CD server.
* Integrate a code linting tool to validate coding standards before any code is accepted into the repository.
* Require developers to run automated integration tests on their local development environments before release.
* Configure a CI server. Add a suite of unit tests to your code and have your CI server run them on commit and verify any changes.

Q85. Your team uses Cloud Build for all CI/CO pipelines. You want to use the kubectl builder for Cloud Build to deploy new images to Google Kubernetes Engine (GKE). You need to authenticate to GKE while minimizing development effort. What should you do?
* Assign the Container Developer role to the Cloud Build service account.
* Specify the Container Developer role for Cloud Build in the cloudbuild.yaml file.
* Create a new service account with the Container Developer role and use it to run Cloud Build.
* Create a separate step in Cloud Build to retrieve service account credentials and pass these to kubectl.