GOOGLE Associate Cloud Engineer

Exam Questions

Total Questions: 300+
Demo Questions: 35
Version: Updated for 2025

Prepared and Verified by Cert Empire – Your Trusted IT Certification Partner

For Access to the full set of Updated Questions – Visit:
Google Associate Cloud Engineer Exam Questions by Cert Empire

Question: 1

Your company is moving its entire workload to Compute Engine. Some servers should be accessible
through the Internet, and other servers should only be accessible over the internal network. All servers need to be able to talk to each other over specific ports and protocols. The current on-premises network relies on a demilitarized zone (DMZ) for the public servers and a Local Area Network (LAN) for the private servers. You need to design the networking infrastructure on Google Cloud to match these requirements. What should you do?

A. 1. Create a single VPC with a subnet for the DMZ and a subnet for the LAN. 2. Set up firewall rules
to open up relevant traffic between the DMZ and the LAN subnets, and another firewall rule to allow
public ingress traffic for the DMZ.

B. 1. Create a single VPC with a subnet for the DMZ and a subnet for the LAN. 2. Set up firewall rules
to open up relevant traffic between the DMZ and the LAN subnets, and another firewall rule to allow
public egress traffic for the DMZ.

C. 1. Create a VPC with a subnet for the DMZ and another VPC with a subnet for the LAN. 2. Set up
firewall rules to open up relevant traffic between the DMZ and the LAN subnets, and another firewall
rule to allow public ingress traffic for the DMZ.

D. 1. Create a VPC with a subnet for the DMZ and another VPC with a subnet for the LAN. 2. Set up
firewall rules to open up relevant traffic between the DMZ and the LAN subnets, and another firewall
rule to allow public egress traffic for the DMZ.

Answer:

A

Explanation:

The most effective and standard design on Google Cloud for this scenario is to use a single Virtual Private Cloud (VPC). A single VPC acts as a global, private networking space. Within this VPC, you create separate subnets to organize resources, such as one for the DMZ and one for the LAN. This provides IP address range separation.
Crucially, security and traffic control are then managed by VPC firewall rules. You would create an

ingress rule to allow specific public traffic to instances in the DMZ subnet (typically by applying a network tag like web-server). You would then create other firewall rules to strictly control traffic between the DMZ and LAN subnets, again using tags or service accounts for precise targeting. This approach is efficient, manageable, and a documented best practice.

Why Incorrect Options are Wrong:

B. This option is incorrect because it specifies allowing public egress traffic for the DMZ. While egress is needed, the critical rule for a public-facing service is the ingress rule that allows incoming traffic from the internet.
C. Using two separate VPCs introduces unnecessary complexity. Communication between VPCs requires VPC Network Peering, which has limitations. For instance, firewall rules in one VPC cannot use network tags from the peered VPC, forcing less flexible IP-based rules.
D. This option is incorrect for two reasons: it uses the more complex two-VPC design and incorrectly focuses on public egress traffic instead of the necessary public ingress rule for the DMZ.

References:

1. Google Cloud Architecture Framework, Network design, VPC network segmentation: "For workloads that don't require the strict network isolation that's provided by separate VPC networks, use subnets and tags to create segmentation. This approach has the following benefits: Lower cost for network administration... Centralized control of firewall rules and routes in a single VPC network." This directly supports using subnets and tags within a single VPC (Option A) over separate VPCs (Option C) for this type of segmentation.
2. Google Cloud VPC Documentation, Use VPC firewall rules, Source and target filtering: "You can filter the sources for ingress rules and destinations for egress rules by using either network tags or service accounts." This highlights the flexibility of using tags within a single VPC, which is central to the solution in Option A.
3. Google Cloud VPC Documentation, VPC Network Peering, Details and properties: "Firewall rules. You cannot use network tags or service accounts from a peered VPC in a firewall rule." This statement confirms the limitation of the two-VPC approach (Option C), making it less manageable than the single-VPC design where tags can be used freely.

Question: 2

You have created a new project in Google Cloud through the gcloud command line interface (CLI) and
linked a billing account. You need to create a new Compute
Engine instance using the CLI. You need to perform the prerequisite steps. What should you do?

A. Create a Cloud Monitoring Workspace.

B. Create a VPC network in the project.

C. Enable the compute googleapis.com API.

D. Grant yourself the IAM role of Compute Admin.

Answer:

C

Explanation:

Before you can create resources for a specific Google Cloud service, such as Compute Engine, you must first enable its corresponding API for the project. When a new project is created, especially via the command-line interface (CLI), APIs are not enabled by default. The compute.googleapis.com API must be explicitly enabled to allow the creation and management of Compute Engine resources like VM instances. This is a fundamental prerequisite step for using the service.

Why Incorrect Options are Wrong:

A. Creating a Cloud Monitoring Workspace is for observing and alerting on resource performance; it is not a prerequisite for creating a Compute Engine instance.
B. New Google Cloud projects are automatically created with a default Virtual Private Cloud (VPC) network, so creating a new one is not a required step to launch an instance.
D. The user who creates a project is automatically granted the primitive Owner (roles/owner) IAM role, which includes all permissions necessary to create Compute Engine instances, making this step redundant.

References:

1. Enabling APIs: Google Cloud Documentation, "Enabling and disabling services," states: "Before you can use a Google Cloud service, you must first enable the service for your Google Cloud project... To enable an API for a project, you must have the serviceusage.services.enable permission for the project." The "Before you begin" section for creating a VM instance also lists enabling the Compute Engine API as a prerequisite.
Source: Google Cloud Documentation, Creating and starting a VM instance, "Before you begin" section.

2. Default VPC Network: Google Cloud Documentation, "VPC network overview," explains the default network: "When you create a new project, it starts with a default network. This network is an auto mode VPC network..." This confirms a network is already available.
Source: Google Cloud Documentation, VPC network overview, "Default network" section.
3. IAM Roles: Google Cloud Documentation, "Basic roles," describes the Owner role: "Owners can manage all resources in the project... When you create a Google Cloud project, you are automatically granted the Owner role for that project." This role includes the compute.instances.create permission.
Source: Google Cloud Documentation, IAM basic and predefined roles reference, "Basic roles" section.

Question: 3

Your company has developed a new application that consists of multiple microservices. You want to
deploy the application to Google Kubernetes Engine (GKE), and you want to ensure that the cluster
can scale as more applications are deployed in the future. You want to avoid manual intervention when each new application is deployed. What should you do?

A. Deploy the application on GKE, and add a HorizontalPodAutoscaler to the deployment.

B. Deploy the application on GKE, and add a VerticalPodAutoscaler to the deployment.

C. Create a GKE cluster with autoscaling enabled on the node pool. Set a minimum and maximum for
the size of the node pool.

D. Create a separate node pool for each application, and deploy each application to its dedicated node pool.

Answer:

C

Explanation:

The core requirement is to ensure the cluster can scale automatically as new applications are deployed, which translates to needing more compute nodes. The GKE cluster autoscaler is the feature designed specifically for this purpose. It automatically adds or removes nodes in a node pool based on the resource demands of pending pods. When a new application is deployed and there isn't enough capacity on existing nodes, the cluster autoscaler will provision new nodes (up to the defined maximum) to accommodate the new workload, thus avoiding manual intervention and ensuring scalability.

Why Incorrect Options are Wrong:

A. Horizontal Pod Autoscaler scales the number of application pods based on metrics like CPU usage. It does not add nodes to the cluster.
B. Vertical Pod Autoscaler adjusts the CPU and memory requests for existing pods. It does not add nodes to the cluster for new deployments.
D. This is an organizational strategy for workload isolation, not a scaling mechanism. It can lead to inefficient resource usage and still requires an autoscaling solution.

References:

1. Google Cloud Documentation, "Cluster autoscaler": "The GKE cluster autoscaler automatically resizes the number of nodes in a given node pool, based on the demands of your workloads. You don't need to manually add or remove nodes or overprovision your node pools. Instead, you specify a minimum and maximum size for the node pool, and the rest is automatic." This directly supports the chosen answer (C).
2. Google Cloud Documentation, "Autoscaling in GKE": This document contrasts the different autoscaling types. It clarifies that "Cluster autoscaler automatically resizes your GKE cluster's node pools based on your Pods' resource requests," whereas Horizontal and Vertical Pod Autoscalers manage the resources or replica count of the Pods themselves, not the underlying nodes. This distinction highlights why (A) and (B) are incorrect for scaling the cluster's capacity.
3. Google Cloud Documentation, "Horizontal pod autoscaling": "Horizontal Pod Autoscaling changes the shape of your Kubernetes workload by automatically increasing or decreasing the number of Pods in response to the workload's CPU or memory consumption...". This confirms that HPA scales pods, not nodes, making option (A) insufficient for the stated problem.

Question: 4

Your coworker has helped you set up several configurations for gcloud. You've noticed that you're running commands against the wrong project. Being new to the company, you haven't yet memorized any of the projects. With the fewest steps possible, what's the fastest way to switch to the correct configuration?

A. Run gcloud configurations list followed by gcloud configurations activate .

B. Run gcloud config list followed by gcloud config activate.

C. Run gcloud config configurations list followed by gcloud config configurations activate.

D. Re-authenticate with the gcloud auth login command and select the correct configurations on login.

Answer:

C

Explanation:

gcloud stores settings such as the active project in named "configurations."
The officially-documented way to view the available configurations and then make one current is:
1. gcloud config configurations list # see all configuration names

2. gcloud config configurations activate # make the right one active
This changes the active project and any other properties in a single step, without re-authentication or editing individual properties, and uses the only gcloud command group that actually exists in the Cloud SDK.

Why Incorrect Options are Wrong:

A. "gcloud configurations ..." is not a valid command group; invocation fails with "unknown group 'configurations'".
B. gcloud config list merely prints current settings, and there is no sub-command "gcloud config activate".
D. gcloud auth login renews credentials; it does not switch configurations and requires extra user interaction.

References:

1. Google Cloud SDK Documentation - "Managing gcloud CLI configurations", Section "List existing configurations" & "Activate a configuration":
https://cloud.google.com/sdk/docs/configurations#listexistingconfigurations
(commands: gcloud config configurations list / activate)
2. gcloud Reference - "gcloud config configurations list" & "gcloud config configurations activate", SDK version 465.0.0, Command synopsis paragraphs.

Question: 5

The storage costs for your application logs have far exceeded the project budget. The logs are currently being retained indefinitely in the Cloud Storage bucket myapp-gcp-ace-logs. You have been
asked to remove logs older than 90 days from your Cloud Storage bucket. You want to optimize ongoing Cloud Storage spend. What should you do?

A. Write a script that runs gsutil ls - - gs://myapp-gcp-ace-logs/** to find and remove items older than 90 days. Schedule the script with cron.

B. Write a lifecycle management rule in JSON and push it to the bucket with gsutil lifecycle set config-
json-file.

C. Write a lifecycle management rule in XML and push it to the bucket with gsutil lifecycle set config-
xml-file.

D. Write a script that runs gsutil ls -lr gs://myapp-gcp-ace-logs/** to find and remove items older than 90 days. Repeat this process every morning.

Answer:

B

Explanation:

The most efficient and cost-effective method to automatically manage the lifecycle of objects in a Cloud Storage bucket is to use the native Object Lifecycle Management feature. This feature allows you to define rules that are automatically applied to objects. A rule can be configured with a Delete action and a condition based on age (in days). This approach is fully managed by Google Cloud, requires no additional infrastructure like virtual machines or cron jobs, and is the recommended best practice for optimizing storage costs by removing old data. The configuration is defined in a JSON file and applied using the gsutil lifecycle set command.

Why Incorrect Options are Wrong:

A. This is a less optimal, self-managed solution. It requires a separate compute resource to run the cron job, incurring extra cost and management overhead, and is less efficient than the built-in lifecycle feature.
C. The gsutil lifecycle set command expects the configuration file to be in JSON format, not XML. XML is used for S3 compatibility but is not the native format for gsutil.
D. This approach is inefficient and relies on manual intervention ("Repeat this process every morning"), which is not a scalable or reliable method for ongoing optimization. Automation is key.

References:

1. Google Cloud Documentation - Object Lifecycle Management: "Object Lifecycle Management allows you to define a Lifecycle configuration for a bucket, which specifies a set of rules that apply to current and future objects in the bucket... One of the supported actions is Delete, which deletes an object when it meets the criteria you specify in the rule."
Source: Google Cloud Documentation, "Object Lifecycle Management", https://cloud.google.com/storage/docs/managing-lifecycles, "Introduction" section.
2. Google Cloud Documentation - gsutil lifecycle command: "The gsutil lifecycle command allows you to get or set the lifecycle management configuration for a bucket... The set sub-command sets the lifecycle configuration on one or more buckets. It takes a single JSON file name as a parameter."
Source: Google Cloud Documentation, "gsutil Command-line Tool", lifecycle command, https://cloud.google.com/storage/docs/gsutil/commands/lifecycle, "SET BUCKET LIFECYCLE CONFIGURATION" section.
3. Google Cloud Documentation - Example Lifecycle Configurations: This page provides a specific example of a JSON configuration to delete objects older than a specified number of days. The example shows a rule with action: "type": "Delete" and condition: "age": 90.
Source: Google Cloud Documentation, "Examples of lifecycle configurations", https://cloud.google.com/storage/docs/lifecycle-examples#delete, "Example 1: Delete objects older than 365 days" section (the principle applies to 90 days).

Question: 6

Users of your application are complaining of slowness when loading the application. You realize the
slowness is because the App Engine deployment serving the application is deployed in us-central whereas all users of this application are closest to europe-west3. You want to change the region of
the App Engine application to europe-west3 to minimize latency. What's the best way to change the
App Engine region?

A. Create a new project and create an App Engine instance in europe-west3

B. Use the gcloud app region set command and supply the name of the new region.

C. From the console, under the App Engine page, click edit, and change the region drop-down.

D. Contact Google Cloud Support and request the change.

Answer:

A

Explanation:

The region for a Google App Engine application is a permanent setting chosen during its initial creation within a Google Cloud project. This location cannot be changed after it has been set. A Google Cloud project can contain only one App Engine application. Therefore, to move the application to a new region, the only valid procedure is to create an entirely new Google Cloud project. Within this new project, you can then create a new App Engine application and select the desired europe-west3 region during the setup process before deploying your code.

Why Incorrect Options are Wrong:

B. The gcloud app region set command does not exist. The region is specified only once with the gcloud app create command and is immutable thereafter.
C. The Google Cloud Console UI does not offer an option to edit the region of an existing App Engine application, as this is a permanent, unchangeable setting.
D. Google Cloud Support cannot perform this action. The immutability of the App Engine region is a fundamental architectural constraint of the platform that support personnel cannot override.

References:

1. Google Cloud Documentation, App Engine Locations: "Important: When you create an app in your Google Cloud project, you must choose a location... The location you choose for your app is permanent and cannot be changed." This statement directly confirms the immutability of the App Engine region.

Source: Google Cloud Documentation, "App Engine Locations", https://cloud.google.com/appengine/docs/locations, Section: "Choosing a location".
2. Google Cloud SDK Documentation, gcloud app create: The documentation for the --region flag states: "The region to create the App Engine application in. You can't change the region after you set it." This confirms that the region is set at creation and cannot be modified with a command.
Source: Google Cloud SDK Command Reference, gcloud app create, https://cloud.google.com/sdk/gcloud/reference/app/create.
3. Google Cloud Documentation, Creating a Google Cloud Project: "Before you can deploy an app, you typically need to create a Google Cloud project... When you create your App Engine app, it is created in a specific region." This highlights the link between the project, the single App Engine app, and the region selection at creation time.
Source: Google Cloud Documentation, "Setting up your development environment", https://cloud.google.com/appengine/docs/standard/python3/setting-up-environment, Section: "Creating a Google Cloud project".

Question: 7

A company wants to build an application that stores images in a Cloud Storage bucket and wants to
generate thumbnails as well as resize the images. They want to use a google managed service that
can scale up and scale down to zero automatically with minimal effort. You have been asked to recommend a service. Which GCP service would you suggest?

A. Google Compute Engine

B. Google App Engine

C. Cloud Functions

D. Google Kubernetes Engine

Answer:

C

Explanation:

Cloud Functions is the most suitable service for this scenario. It is a serverless, event-driven compute platform that executes code in response to events, such as a new file being uploaded to a Cloud Storage bucket. This service is fully managed by Google, requires no server provisioning, and automatically scales based on load, including scaling down to zero. This "scale-to-zero" capability means you only pay for the compute time you consume, making it highly cost-effective and meeting the requirement for minimal effort and automatic scaling.

Why Incorrect Options are Wrong:

A. Google Compute Engine: This is an IaaS solution that requires manual management of virtual machines, including scaling and OS patching. It does not automatically scale to zero, incurring costs even when idle.
B. Google App Engine: While it is a managed platform that can scale to zero, it is designed for hosting web applications and APIs, not for single-purpose, event-driven tasks. It is less direct and requires more configuration than Cloud Functions for this specific use case.
D. Google Kubernetes Engine: This is a container orchestration service that provides significant power and flexibility but requires substantial management effort. The underlying cluster nodes do not scale to zero, making it unsuitable for this task's cost and simplicity requirements.

References:

1. Google Cloud Documentation - Cloud Functions Use Cases: The official documentation provides a specific tutorial for this exact use case: "This tutorial demonstrates how to build an event-driven, serverless function to process images. The function is triggered when a file is uploaded to Cloud Storage, and it generates a thumbnail of the image."
Source: Google Cloud, "Tutorial: Cloud Storage trigger: image thumbnailing".
2. Google Cloud Documentation - Choosing a compute option: This document compares Google's compute services. It describes Cloud Functions as ideal for "Event-driven applications that execute code in response to events" and highlights its serverless nature where "you don't need to manage servers or runtimes." In contrast, it positions GCE and GKE as requiring more infrastructure management.
Source: Google Cloud, "Choosing a compute option", Section: "Serverless computing".
3. Google Cloud Documentation - Cloud Functions Overview: "Cloud Functions is a serverless execution environment for building and connecting cloud services... your function is triggered and your code is executed in a fully managed environment. There is no need to provision any infrastructure or worry about managing any servers." This confirms the "minimal effort" and "managed service" requirements.
Source: Google Cloud, "Cloud Functions - Product Overview".

Question: 8

You are designing an application that lets users upload and share photos. You expect your application
to grow really fast and you are targeting a worldwide audience. You want to delete uploaded photos
after 30 days. You want to minimize costs while ensuring your application is highly available. Which
GCP storage solution should you choose?

A. Persistent SSD on VM instances.

B. Cloud Filestore.

C. Multiregional Cloud Storage bucket.

D. Cloud Datastore database.

Answer:

C

Explanation:

The scenario requires a highly available, globally accessible, and scalable storage solution for photos (binary objects) with a defined data lifecycle and a focus on cost-effectiveness. A multi-regional Cloud Storage bucket is the ideal choice. Cloud Storage is designed for unstructured data like images and scales massively. The multi-regional class ensures high availability and low latency for a worldwide audience by storing data redundantly across geographically separate regions. Crucially, Cloud Storage offers Object Lifecycle Management, which can be configured with a simple rule to automatically delete objects after 30 days, fulfilling a key requirement while minimizing operational overhead and cost.

Why Incorrect Options are Wrong:

A. Persistent SSD is block storage tied to a specific VM in a single zone. It is not globally available, doesn't scale easily for this use case, and lacks built-in lifecycle management.
B. Cloud Filestore is a managed NFS (file) storage service. It is a regional service, not designed for global content delivery, and is more expensive than object storage for this purpose.
D. Cloud Datastore is a NoSQL document database. It is designed for structured data (like metadata), not for storing large, unstructured binary files like photos, which would be inefficient and costly.

References:

1. Google Cloud Documentation, Cloud Storage Classes: "Multi-regions are a large geographic area, such as the United States, that contains at least two geographic places... Use Multi-Region Storage if you want to serve content to users outside of a single continent or to users across a large geographic area within a continent and want the performance benefits of serving content from a location that is geographically closer to your users." This supports the "worldwide audience" and "high availability" requirements.
Source: Google Cloud Documentation, "Storage classes", Section: "Multi-regions".
2. Google Cloud Documentation, Object Lifecycle Management: "Object Lifecycle Management allows you to define a Lifecycle configuration for a bucket... A lifecycle configuration is a set of rules that apply to current and future objects in a bucket... One of the actions you can specify in a rule is the Delete action... When the condition of this rule is met, Cloud Storage deletes the object." This directly addresses the requirement to "delete uploaded photos after 30 days."
Source: Google Cloud Documentation, "Object Lifecycle Management", Section: "Lifecycle configurations".
3. Google Cloud Documentation, Choosing a storage option: The documentation provides a decision tree that guides users. For "Unstructured data" (like photos) and "Worldwide" access, the primary recommendation is Cloud Storage. In contrast, Persistent Disk is for "Block storage for VMs" and Filestore is for "File storage for GKE and VMs".
Source: Google Cloud Documentation, "Store your data", Section: "Choosing the right option for your data".

Question: 9

You are designing an application that uses WebSockets and HTTP sessions that are not distributed
across the web servers. You want to ensure the application runs properly on Google Cloud Platform.
What should you do?

A. Meet with the cloud enablement team to discuss load balancer options.

B. Redesign the application to use a distributed user session service that does not rely on WebSockets and HTTP sessions.

C. Review the encryption requirements for WebSocket connections with the security team.

D. Convert the WebSocket code to use HTTP streaming.

Answer:

A

Explanation:

The application's design, which uses non-distributed HTTP sessions and WebSockets, requires that all requests from a single client are consistently sent to the same backend server. This is known as session affinity or "sticky sessions." Google Cloud offers several load balancing solutions, and their support for session affinity and the WebSocket protocol varies. For example, the External HTTP(S) Load Balancer supports both WebSocket proxying and multiple types of session affinity (e.g., generated cookie, client IP). Therefore, the critical first step is to evaluate and select the correct load balancer configuration to ensure the application's stateful nature is properly handled.

Why Incorrect Options are Wrong:

B. Redesigning the application is a significant architectural change. The immediate problem can be solved with the correct infrastructure configuration (a load balancer with session affinity) without rewriting the application.
C. While encryption is an important security consideration, it does not address the core functional requirement of routing stateful connections to a consistent backend, which is the cause of the problem.
D. Converting WebSockets to another technology is a major development effort that does not solve the separate, but related, problem of non-distributed HTTP session affinity.

References:

1. Session Affinity: Google Cloud documentation for External HTTP(S) Load Balancing states, "Session affinity sends all requests from the same client to the same backend VM or container, as long as the backend is healthy and has capacity." This directly addresses the need for non-distributed sessions.
Source: Google Cloud Documentation, "Session affinity overview," Section: "Session affinity types".
2. WebSocket Support: The overview for External HTTP(S) Load Balancing explicitly lists WebSocket support as a feature. "The load balancer can proxy WebSocket traffic with no configuration required... If you use WebSockets, the session affinity feature should be enabled to route requests for a particular session to the same backend."
Source: Google Cloud Documentation, "External HTTP(S) Load Balancing overview," Section: "WebSocket proxy support".
3. Load Balancer Selection: The choice of load balancer is critical for application functionality. The "Choose a load balancer" guide helps users select the right product based on requirements like traffic type, global vs. regional needs, and protocol support, which is the process implied by option A.
Source: Google Cloud Documentation, "Cloud Load Balancing overview," Section: "Choose a load balancer".

Question: 10

You have a number of compute instances belonging to an unmanaged instances group. You need to
SSH to one of the Compute Engine instances to run an ad hoc script. You've already authenticated
gcloud, however, you don't have an SSH key deployed yet. In the fewest steps possible, what's the
easiest way to SSH to the instance?

A. Run gcloud compute instances list to get the IP address of the instance, then use the ssh command.

B. Use the gcloud compute ssh command.

C. Create a key with the ssh-keygen command. Then use the gcloud compute ssh command.

D. Create a key with the ssh-keygen command. Upload the key to the instance. Run gcloud compute
instances list to get the IP address of the instance, then use the ssh command.

Answer:

B

Explanation:

The gcloud compute ssh command is a wrapper around the standard ssh command that automates the entire connection process. When executed, it checks for the existence of SSH keys. If no keys are found, it automatically generates a new key pair (/.ssh/googlecomputeengine), adds the public key to the project or instance metadata, and then initiates the SSH connection to the specified instance. This single command handles authentication, key generation, key distribution, and connection, making it the most efficient method with the fewest steps, as required by the question.

Why Incorrect Options are Wrong:

A. This method will fail. The standard ssh command requires a pre-existing private key on the local machine and a corresponding public key on the target instance, which the scenario states is not deployed.
C. This is an unnecessary step. While it would work, the gcloud compute ssh command will automatically generate the key pair for you if one doesn't exist, making the manual ssh-keygen step redundant.
D. This is the most complex and manual process. It involves multiple commands and manual key management, which is the exact opposite of the "easiest way" and "fewest steps" requirement.

References:

1. Google Cloud Documentation, gcloud compute ssh: "If you do not have a public SSH key, gcloud compute ssh generates one for you. When you generate a new key, gcloud compute ssh adds the key to your project or instance metadata and connects you to the VM." This confirms that the command handles key generation automatically.
Source: Google Cloud SDK Documentation, gcloud compute ssh, Command Description section.
2. Google Cloud Documentation, Connect to Linux VMs: "The gcloud command-line tool provides the simplest way to connect to your instances... When you connect to a Linux instance by using the gcloud command-line tool, Compute Engine creates an ephemeral SSH key for you and sets a username."
Source: Google Cloud Documentation Compute Engine User guides Connect to Linux VMs Connect using Google Cloud tools.
3. Google Cloud Documentation, Managing SSH keys in metadata: "When you use gcloud compute ssh to connect to an instance, gcloud can automatically create and add an SSH key to the project for you."
Source: Google Cloud Documentation Compute Engine User guides Access control Managing SSH keys in metadata How SSH keys work on Compute Engine instances.

Question: 11

You created a cluster.YAML file containing
resources:
name: cluster
type: container.v1.cluster
properties:
zone: europe-west1-b
cluster:
description: My GCP ACE cluster
initialNodeCount: 2
You want to use Cloud Deployment Manager to create this cluster in GKE. What should you do?

A. gcloud deployment-manager deployments create my-gcp-ace-cluster --config cluster.yaml

B. gcloud deployment-manager deployments create my-gcp-ace-cluster --type container.v1.cluster --
config cluster.yaml

C. gcloud deployment-manager deployments apply my-gcp-ace-cluster --type container.v1.cluster --
config cluster.yaml

D. gcloud deployment-manager deployments apply my-gcp-ace-cluster --config cluster.yaml

Answer:

A

Explanation:

The correct command to create a new deployment using Google Cloud Deployment Manager from a configuration file is gcloud deployment-manager deployments create. This command requires two main arguments: a name for the new deployment (in this case, my-gcp-ace-cluster) and the path to the configuration file, specified using the --config flag (--config cluster.yaml). The YAML file itself contains all the necessary resource definitions, including the resource type (container.v1.cluster) and its properties, so these do not need to be specified on the command line.

Why Incorrect Options are Wrong:

B. The --type flag is not a valid argument for the gcloud deployment-manager deployments create command. The resource type must be defined within the configuration file.
C. The apply subcommand is not a valid operation for gcloud deployment-manager deployments. Additionally, the --type flag is an invalid argument for this command.
D. The apply subcommand is not a valid operation for gcloud deployment-manager deployments.

The correct command to create a new deployment is create.

References:

1. Google Cloud Documentation, "Creating a deployment": This official guide explicitly shows the command syntax for creating a deployment from a configuration file.
Reference: Under the section "Creating a deployment with the gcloud tool," the documented command is: gcloud deployment-manager deployments create DEPLOYMENTNAME --config CONFIGFILE. This directly matches the structure of option A.
Source: Google Cloud, Deployment Manager Documentation, "Creating a deployment", https://cloud.google.com/deployment-manager/docs/deployments/creating-deployments#gcloud
2. Google Cloud SDK Documentation, gcloud deployment-manager deployments create: The official command-line reference details the syntax, positional arguments, and flags for the create command.
Reference: The documentation specifies DEPLOYMENTNAME as a required positional argument and --config=CONFIG as a required flag for providing a configuration file. It does not list an apply subcommand or a --type flag.
Source: Google Cloud SDK, gcloud deployment-manager deployments create, https://cloud.google.com/sdk/gcloud/reference/deployment-manager/deployments/create
3. Google Cloud Documentation, "Configuration fundamentals": This document explains that the type of a resource is a mandatory field within the resources section of a Deployment Manager configuration file.

Reference: The section "The structure of a configuration file" shows that type is a key-value pair under the resources list, confirming it belongs in the YAML file, not as a command-line flag.
Source: Google Cloud, Deployment Manager Documentation, "Configuration fundamentals", https://cloud.google.com/deployment-manager/docs/configuration/configuration-fundamentals#structure-of-a-configuration-file

Question: 12

You created a Kubernetes deployment by running kubectl run nginx image=nginx labels=app=prod.
Your Kubernetes cluster is also used by a number of other deployments. How can you find the identifier of the pods for this nginx deployment?

A. kubectl get deployments -output=pods

B. gcloud get pods -selector="app=prod"

C. kubectl get pods -l "app=prod"

D. gcloud list gke-deployments -filter=pod

Answer:

C

Explanation:

The kubectl command-line tool is used to manage resources within a Kubernetes cluster. The command kubectl get pods lists all pods in the current namespace. To filter this list and find specific pods, you use a label selector. The deployment was created with the label app=prod. The -l flag (or --selector) filters resources based on this label. Therefore, kubectl get pods -l "app=prod" correctly identifies and lists only the pods managed by the specified deployment. Note that the question contains a common typo, using a capital 'I' instead of a lowercase 'l' for the selector flag.

Why Incorrect Options are Wrong:

A. kubectl get deployments -output=pods
This command is syntactically incorrect. The --output flag is for formatting (e.g., -o yaml, -o json), and pods is not a valid value for this flag.
B. gcloud get pods -selector="app=prod"
The gcloud command is for managing Google Cloud resources, not for interacting with resources inside a Kubernetes cluster. The correct tool for this task is kubectl.
D. gcloud list gke-deployments -filter=pod
This command is invalid. gcloud does not have a list gke-deployments command, and this is not the correct syntax or tool for querying pods within a cluster.

References:

1. Kubernetes Documentation, "Labels and Selectors": This document explains that labels are key/value pairs attached to objects, such as pods. It states, "Via a label selector, the client/user can identify a set of objects." It also provides examples of using the -l or --selector flag with kubectl get to filter objects.
Reference: Kubernetes Documentation Concepts Overview Working with Objects Labels and Selectors.
2. Kubernetes Documentation, kubectl Cheat Sheet: This official cheat sheet lists common kubectl commands. Under the "Listing Resources" section, it shows kubectl get pods -l app=my-app as the standard way to list pods with a specific label.
Reference: Kubernetes Documentation Reference kubectl Cheat Sheet.
3. Google Cloud Documentation, "Deploying a stateless application": This GKE tutorial demonstrates the standard workflow for managing applications. It consistently uses kubectl get pods -l app= to check the status of pods created by a deployment, confirming this is the correct procedure for a GKE cluster.
Reference: Google Cloud Kubernetes Engine Documentation How-to guides Deploying a stateless application Step 4: Inspect the Deployment and Service.

Question: 13

You created a Kubernetes deployment by running kubectl run nginx image=nginx replicas=1. After a
few days, you decided you no longer want this deployment. You identified the pod and deleted it by
running kubectl delete pod. You noticed the pod got recreated.
$ kubectl get pods
NAME READY STATUS RESTARTS AGE
nginx-84748895c4-nqqmt 1/1 Running 0 9m41s
$ kubectl delete pod nginx-84748895c4-nqqmt
pod nginx-84748895c4-nqqmt deleted
$ kubectl get pods
NAME READY STATUS RESTARTS AGE
nginx-84748895c4-k6bzl 1/1 Running 0 25s
What should you do to delete the deployment and avoid pod getting recreated?

A. kubectl delete deployment nginx

B. kubectl delete -deployment=nginx

C. kubectl delete pod nginx-84748895c4-k6bzl -no-restart 2

D. kubectl delete inginx

Answer:

A

Explanation:

A Kubernetes Deployment is a controller that manages the lifecycle of Pods via a ReplicaSet. The Deployment's primary responsibility is to ensure that the actual state of the cluster matches the desired state defined in its configuration (e.g., replicas=1). When you manually delete a Pod managed by a Deployment, the ReplicaSet detects the deviation from the desired state and automatically creates a new Pod to replace it. To permanently remove the application and stop the Pods from being recreated, you must delete the controlling object, which is the Deployment itself. The command kubectl delete deployment nginx correctly targets and deletes the nginx Deployment, which then cascades the deletion to its underlying ReplicaSet and Pods.

Why Incorrect Options are Wrong:

B. kubectl delete -deployment=nginx is syntactically incorrect. The kubectl delete command expects the resource type and name as arguments, not as a flag.
C. kubectl delete pod nginx-84748895c4-k6bzl -no-restart 2 is syntactically invalid. The kubectl delete pod command has no --no-restart flag, and deleting the pod is the wrong approach anyway.

D. kubectl delete inginx is incorrect due to a typo (inginx instead of nginx) and because it omits the mandatory resource type (e.g., deployment, pod).

References:

1. Kubernetes Documentation, "Deployments": "A Deployment provides declarative updates for Pods and ReplicaSets... When you delete a Deployment, it cleans up the ReplicaSets and Pods it created. You can delete a Deployment by using kubectl delete." This is demonstrated with the example: kubectl delete deployment nginx-deployment.
Source: Kubernetes Documentation - Deployments
2. Kubernetes Documentation, kubectl Cheat Sheet: The syntax for deleting resources is shown as kubectl delete (-f FILENAME TYPE (NAME -l label --all)). This confirms that kubectl delete deployment nginx is the correct format, where deployment is the TYPE and nginx is the NAME.
Source: Kubernetes Documentation - kubectl Cheat Sheet
3. Google Cloud Documentation, "Deploying a containerized web application": In the "Cleaning up" section of this tutorial, the command to remove the deployment is kubectl delete deployment hello-web, reinforcing that the deployment object must be the target of the delete operation.
Source: Google Cloud Documentation - Deploying a containerized web application

Question: 14

You have a number of applications that have bursty workloads and are heavily dependent on topics
to decouple publishing systems from consuming systems. Your company would like to go serverless
to enable developers to focus on writing code without worrying about infrastructure. Your solution architect has already identified Cloud Pub/Sub as a suitable alternative for decoupling systems. You
have been asked to identify a suitable GCP Serverless service that is easy to use with Cloud Pub/Sub.
You want the ability to scale down to zero when there is no traffic in order to minimize costs. You want to follow Google recommended practices. What should you suggest?

A. Cloud Run for Anthos

B. Cloud Run

C. App Engine Standard

D. Cloud Functions.

Answer:

D

Explanation:

Cloud Functions is the most suitable service for this scenario. It is a fully managed, event-driven, serverless compute platform designed to run small, single-purpose code in response to events. It has a native, built-in trigger for Cloud Pub/Sub, making the integration extremely simple and aligning with the "easy to use" requirement. Cloud Functions automatically scales based on the number of incoming Pub/Sub messages, including scaling down to zero when there are no messages to process. This directly addresses the need to handle bursty workloads and minimize costs, allowing developers to focus solely on the function's code without managing any infrastructure.

Why Incorrect Options are Wrong:

A. Cloud Run for Anthos runs on a GKE cluster, which introduces infrastructure management overhead, contradicting the core requirement for developers to not worry about infrastructure.
B. Cloud Run is a valid serverless option, but Cloud Functions offers a more direct and simpler native integration for Pub/Sub triggers, making it the "easiest to use" choice as specified.
C. App Engine Standard is a serverless platform primarily designed for hosting web applications and APIs, not as a first-choice, event-driven compute service for Pub/Sub messages.

References:

1. Google Cloud Documentation, "Choosing a serverless option": This guide compares serverless services. It states, "Cloud Functions is ideal for event-driven workloads... For example, you can process file uploads to Cloud Storage, or react to log messages." This aligns with reacting to Pub/Sub messages. It positions Cloud Functions as the primary choice for event-based automation.
Source: Google Cloud Documentation, "Serverless computing options", Section: "Choosing a serverless option".
2. Google Cloud Documentation, "Cloud Functions: Cloud Pub/Sub triggers": This document details the native integration. It states, "You can trigger a function when a Cloud Pub/Sub message is sent to a topic. This allows you to process Pub/Sub messages asynchronously." This confirms the direct, easy-to-use integration.
Source: Google Cloud Documentation, "Cloud Functions Docs Triggers Cloud Pub/Sub triggers".
3. Google Cloud Documentation, "Cloud Functions: Execution Environment": This document explains the lifecycle and scaling. It notes, "Instances are allocated on-demand to handle incoming requests... When a function is idle, the number of idle instances might be reduced." This confirms the scale-to-zero capability for cost efficiency.
Source: Google Cloud Documentation, "Cloud Functions Docs Concepts Execution Environment", Section: "Scaling".

Question: 15

You have been asked to migrate a docker application from datacenter to cloud. Your solution architect has suggested uploading docker images to GCR in one project and running an application in
a GKE cluster in a separate project. You want to store images in the project img-278322 and run the
application in the project prod-278986. You want to tag the image as acmetrackntrace:v1. You want to follow Google-recommended practices. What should you do?

A. Run gcloud builds submit --tag gcr.io/img-278322/acmetrackntrace

B. Run gcloud builds submit --tag gcr.io/img-278322/acmetrackntrace:v1

C. Run gcloud builds submit --tag gcr.io/prod-278986/acmetrackntrace

D. Run gcloud builds submit --tag gcr.io/prod-278986/acmetrackntrace:v1

Answer:

B

Explanation:

The gcloud builds submit command uses Cloud Build to build and push a container image. The --tag flag specifies the full name for the image in Google Container Registry (GCR). The standard naming convention for GCR is HOSTNAME/PROJECT-ID/IMAGE:TAG.

The question explicitly states that the Docker images must be stored in the project img-278322. The image name is acmetrackntrace and the tag is v1. Therefore, the correct and complete image name is gcr.io/img-278322/acmetrackntrace:v1. This command correctly targets the specified image storage project, not the project where the GKE cluster runs.

Why Incorrect Options are Wrong:

A. This option is incorrect because it omits the required version tag :v1. Without a specified tag, the image would be tagged as :latest by default.
C. This option is incorrect because it targets the GKE project (prod-278986) for image storage and omits the required version tag :v1.
D. This option is incorrect because it targets the GKE project (prod-278986) for image storage, not the specified image repository project (img-278322).

References:

1. Google Cloud SDK Documentation, gcloud builds submit: The documentation for the command shows that the --tag flag is used to specify "The tag to use for the built image. The tag must be in a registry that the build service account can access." The examples provided use the format gcr.io/PROJECT-ID/IMAGE:TAG.

Source: Google Cloud SDK Documentation, gcloud builds submit, Command description and examples section. https://cloud.google.com/sdk/gcloud/reference/builds/submit
2. Google Cloud Build Documentation, "Build container images": This guide demonstrates building an image and pushing it to Container Registry. The example command is gcloud builds submit --tag gcr.io/PROJECT-ID/quickstart-image ., which establishes the required naming format.
Source: Google Cloud Documentation, "Build container images", "Build the image" section. https://cloud.google.com/build/docs/building/build-containers#buildtheimage
3. Google Container Registry Documentation, "Pushing and pulling images": This document explicitly defines the image naming convention. It states, "The full name of an image is in the format: HOSTNAME/PROJECT-ID/IMAGE:TAG". This confirms that the project ID where the registry is hosted must be part of the image name.
Source: Google Cloud Documentation, "Pushing and pulling images", "Tag the local image with the registry name" section. https://cloud.google.com/container-registry/docs/pushing-and-pulling#tagthelocalimagewiththeregistryname

Question: 16

You have files in a Cloud Storage bucket that you need to share with your suppliers. You want to restrict the time that the files are available to your suppliers to 1 hour. You want to follow Google recommended practices. What should you do?

A. Create a service account with just the permissions to access files in the bucket. Create a JSON key
for the service account. Execute the command gsutil signurl -m 1h gs:///*.

B. Create a service account with just the permissions to access files in the bucket. Create a JSON key
for the service account. Execute the command gsutil signurl -d 1h gs:///**.

C. Create a service account with just the permissions to access files in the bucket. Create a JSON key
for the service account. Execute the command gsutil signurl -p 60m gs:///.

D. Create a JSON key for the Default Compute Engine Service Account. Execute the command gsutil
signurl -t 60m gs:///***

Answer:

B

Explanation:

The most secure and effective method to provide temporary access to Cloud Storage objects is by using signed URLs. This approach follows Google's recommended practice of adhering to the principle of least privilege. By creating a dedicated service account with only the necessary permissions (e.g., roles/storage.objectViewer) to read the files, you limit potential exposure. The gsutil signurl command is the correct tool for this task. The -d 1h flag correctly sets the URL's validity duration to one hour. The gs:/// wildcard ensures that signed URLs are generated for all objects within the bucket, including those in subdirectories.

Why Incorrect Options are Wrong:

A. The -m flag in gsutil signurl is used to specify an HTTP method (e.g., GET, PUT), not the duration for which the URL is valid.
C. The -p flag is used to provide a password for an encrypted private key file, not to set the URL's expiration time. The URI gs:///. is also invalid.
D. Using the Default Compute Engine Service Account violates the principle of least privilege, as it often has broad permissions. The -t flag is not a valid option for specifying duration in gsutil signurl.

References:

1. Google Cloud Documentation, gsutil signurl command: The official documentation specifies the use of the -d flag to set the duration for which the signed URL is valid. It states, " is a number followed by 's', 'm', 'h', or 'd' to indicate seconds, minutes, hours, or days."
Source: Google Cloud SDK Documentation, gsutil signurl section.
2. Google Cloud Documentation, Service Account Best Practices: Google strongly recommends creating dedicated service accounts for specific workloads and granting them the minimum permissions necessary (principle of least privilege), rather than using the highly privileged default service accounts.
Source: Google Cloud, Identity and Access Management Documentation, "Best practices for working with service accounts," Section: "Grant service accounts the minimum permissions necessary."
3. Google Cloud Documentation, Wildcard Names: The wildcard is used to match objects recursively within a bucket. The documentation states, " matches any sequence of characters, including path separators. This allows for recursive wildcarding."
Source: Google Cloud, Cloud Storage Documentation, "Wildcard names."

Question: 17

You have a managed instance group comprised of preemptible VM's. All of the VM's keepdeleting and recreating themselves every minute. What is a possible cause of thisbehavior?

A. Your zonal capacity is limited, causing all preemptible VM's to be shutdown torecover capacity. Try
deploying your group to another zone.

B. You have hit your instance quota for the region.

C. Your managed instance group's VM's are toggled to only last 1 minute inpreemptible settings.

D. Your managed instance group's health check is repeatedly failing, either to amisconfigured health
check or misconfigured firewall rules not allowing the healthcheck to access the instance

Answer:

D

Explanation:

A Managed Instance Group (MIG) uses an autohealing policy to maintain the health and availability of its instances. This policy relies on a configured health check to determine if an instance is responsive. If an instance repeatedly fails the health check (e.g., due to a misconfigured application, a wrong port, or a firewall rule blocking the probe), the MIG's autohealing feature will declare the instance "unhealthy." It will then proactively delete and recreate the instance to restore service. This process creates a continuous cycle of recreation, which, depending on the health check's interval and threshold settings, can easily occur every minute.

Why Incorrect Options are Wrong:

A. While preemption due to capacity is possible, it is typically sporadic and not a rhythmic, one-minute cycle affecting all instances simultaneously.
B. Hitting an instance quota would prevent the MIG from creating new instances to meet its target size, but it would not cause the deletion of existing, running instances.
C. This setting does not exist. Preemptible VMs have a maximum lifetime of 24 hours, not a configurable one-minute lifespan.

References:

1. Google Cloud Documentation, "About autohealing for MIGs": "To set up autohealing, you must configure a health check... The MIG queries the health check status of each instance in the group. If an instance is unhealthy, the MIG recreates that instance." This directly supports that a failing health check triggers recreation.

Source: Google Cloud Compute Engine Documentation Instance groups Managed instance groups (MIGs) "About autohealing for MIGs".
2. Google Cloud Documentation, "Health check concepts": "When you configure a health check-based autohealing policy for a MIG, Compute Engine uses the health state to determine whether to proactively recreate a VM." This reinforces the link between health state and the recreation behavior described.
Source: Google Cloud Load Balancing Documentation Health checks "Health check concepts".
3. Google Cloud Documentation, "Firewall rules for health checks": "Health checks connect to VMs on your behalf... To allow these connections, you must create an ingress allow firewall rule that allows traffic from Google Cloud probers." This confirms that a misconfigured firewall is a common cause for health check failures.
Source: Google Cloud VPC network Documentation Using VPC "Firewall rules for health checks".
4. Google Cloud Documentation, "Preemptible VM instances": "A preemptible VM is an instance that you can create and run at a much lower price than normal instances. However, Compute Engine might stop (preempt) these instances if it requires access to those resources for other tasks... Preemptible instances have a maximum runtime of 24 hours." This confirms that a one-minute lifetime is not a feature and that preemption is not a guaranteed, cyclical event.
Source: Google Cloud Compute Engine Documentation Instances "Preemptible VM instances".

Question: 18

You deployed an application on a managed instance group in Compute Engine. The application accepts Transmission Control Protocol (TCP) traffic on port 389 and requires you to preserve the IP
address of the client who is making a request. You want to expose the application to the internet by
using a load balancer. What should you do?

A. Expose the application by using an external TCP Network Load Balancer.

B. Expose the application by using a TCP Proxy Load Balancer.

C. Expose the application by using an SSL Proxy Load Balancer.

D. Expose the application by using an internal TCP Network Load Balancer.

Answer:

B

Explanation:

The TCP Proxy Load Balancer is a global reverse proxy service designed for non-HTTP(S) TCP traffic from the internet. It terminates TCP connections at the Google Front End (GFE), which provides significant security benefits, including DDoS mitigation, by isolating backend instances from the public internet. To meet the critical requirement of preserving the client's original IP address, you must enable the PROXY protocol on the load balancer's backend service. This protocol adds a header to the request sent to the backend, containing the original source IP and port information. The application on the backend instances must be configured to parse this header. This architecture represents a robust and secure best practice for exposing TCP services globally.

Why Incorrect Options are Wrong:

A. While an external TCP Network Load Balancer preserves the source IP by default because it's a pass-through service, it is a regional service and offers fewer security features. The TCP Proxy Load Balancer is the more robust, global solution.
C. The SSL Proxy Load Balancer is incorrect because it is designed specifically for encrypted SSL/TLS traffic, not for general, unencrypted TCP traffic on a port like 389 (LDAP) as specified in the scenario.
D. An internal TCP Network Load Balancer is incorrect because it is used for traffic within a Google Cloud VPC network and cannot be used to expose an application directly to the public internet.

References:

1. Google Cloud Documentation - TCP Proxy Load Balancer overview: "TCP Proxy Load Balancing is a reverse proxy load balancer that distributes TCP traffic coming from the internet to virtual machine (VM) instances in your Google Cloud VPC network... By default, the original client IP address and port information is not preserved. You can preserve this information by enabling the PROXY protocol." This confirms that option B is a valid solution when the PROXY protocol is enabled.
2. Google Cloud Documentation - Choosing a load balancer: The feature comparison table shows that the TCP Proxy Load Balancer is a "Global" service, while the backend service-based Network Load Balancer is "Regional." For internet-facing applications, a global service is often the superior architectural choice. The table also highlights that proxy-based load balancers (like TCP Proxy) terminate client connections, which is key to their security posture.
3. Google Cloud Documentation - Backend service-based external TCP/UDP Network Load Balancers: "These load balancers are pass-through load balancers... The load balancer is not a proxy... The backends receive the original, unmodified packets from the clients, including the source and destination IP addresses." This explains why option A is a technically viable but less advanced alternative.

Question: 19

You are building a multi-player gaming application that will store game information in a database. As
the popularity of the application increases, you are concerned about delivering consistent performance. You need to ensure an optimal gaming performance for global users, without increasing the management complexity. What should you do?

A. Use Cloud SQL database with cross-region replication to store game statistics in the EU, US, and
APAC regions.

B. Use Cloud Spanner to store user data mapped to the game statistics.

C. Use BigQuery to store game statistics with a Redis on Memorystore instance in the front to provide
global consistency.

D. Store game statistics in a Bigtable database partitioned by username.

Answer:

B

Explanation:

Cloud Spanner is a unique, fully managed, relational database service designed for global scale and strong transactional consistency. For a multi-player gaming application with a global user base, Spanner provides low-latency reads and writes to users worldwide by distributing the database across multiple regions. It scales horizontally as the application's popularity increases, ensuring consistent performance. As a managed service, it abstracts away the complexities of sharding and replication, directly addressing the need to avoid increased management complexity while providing optimal performance for a global audience.

Why Incorrect Options are Wrong:

A. Cloud SQL with cross-region replication forces all write operations to a single primary region, creating high latency for global users not located near that region.
C. BigQuery is an analytical data warehouse (OLAP), not a transactional database (OLTP), making it unsuitable for the low-latency read/write operations required by a live game.
D. Cloud Bigtable is a regional service. While replication is possible, it provides eventual consistency, which may not be acceptable for critical game state data requiring strong consistency.

References:

1. Google Cloud Documentation - "What is Cloud Spanner?": "Cloud Spanner is a fully managed, mission-critical, relational database service that offers transactional consistency at global scale, schemas, SQL... and automatic, synchronous replication for high availability." This directly supports its use for global, consistent applications.
2. Google Cloud Solutions - "Gaming database solutions": This page explicitly recommends Cloud Spanner for "storing player data and game state for millions of users" due to its ability to "scale without downtime and provide a consistent view of data across the globe."
3. Google Cloud Documentation - "About replication in Cloud SQL": "All writes to a Cloud SQL instance go to the primary instance. You can use read replicas to scale read requests." This confirms the write-latency limitation for global applications mentioned in option A's analysis.
4. Google Cloud Documentation - "Overview of Cloud Bigtable replication": "Replication for Cloud Bigtable provides increased durability and availability for your data by copying it across multiple regions... Replication is eventually consistent." This highlights the eventual consistency model, which is a key differentiator from Spanner's strong consistency.

Question: 20

Your company has multiple projects linked to a single billing account in Google Cloud. You need to
visualize the costs with specific metrics that should be dynamically calculated based on company-
specific criteri
a. You want to automate the process. What should you do?

A. In the Google Cloud console, visualize the costs related to the projects in the Reports section.

B. In the Google Cloud console, visualize the costs related to the projects in the Cost breakdown section.

C. In the Google Cloud console, use the export functionality of the Cost table. Create a Looker Studiodashboard on top of the CSV export.

D. Configure Cloud Billing data export to BigOuery for the billing account. Create a Looker Studio dashboard on top of the BigQuery export.

Answer:

D

Explanation:

This solution meets all the requirements of the scenario. Exporting Cloud Billing data to BigQuery provides a detailed, granular dataset of all cost and usage information. BigQuery, a fully managed data warehouse, allows you to run complex SQL queries to create the required "specific metrics" and perform "dynamically calculated" aggregations based on any company-specific criteria. By connecting a Looker Studio (formerly Data Studio) dashboard directly to the BigQuery dataset, you create a powerful, customizable visualization tool. The entire pipeline, from data export to dashboard refresh, is automated, providing up-to-date insights without manual intervention.

Why Incorrect Options are Wrong:

A. The standard Reports section in the Cloud Console offers pre-defined views and filters, which lack the flexibility to create custom, dynamically calculated metrics based on unique business logic.
B. The Cost breakdown section, similar to the Reports section, provides a standard, hierarchical view of costs and does not support the creation of bespoke, company-specific calculations.
C. Exporting to a CSV file is a manual, one-time process. This approach fails the requirement to automate the process, as the CSV would need to be re-exported and re-uploaded for every update.

References:

1. Google Cloud Documentation - Export Cloud Billing data to BigQuery: "If you need to perform in-depth analysis of your Google Cloud billing data, you can export it to a BigQuery dataset... Exporting billing data to BigQuery provides a more detailed view of your costs than the online reports." This document highlights BigQuery as the solution for in-depth, custom analysis.
Source: Google Cloud, Cloud Billing Documentation, "Overview of Cloud Billing data export to BigQuery".
2. Google Cloud Blog - Visualize your Google Cloud spend with Looker Studio: "By exporting your detailed billing data into BigQuery, you can analyze your costs using Looker Studio... This allows you to create rich, interactive dashboards to explore your data and find cost-saving opportunities." This source explicitly recommends the combination of BigQuery and Looker Studio for advanced cost visualization.
Source: Google Cloud Blog, "Visualize your Google Cloud spend with Looker Studio," Getting Started section.
3. Google Cloud Documentation - Understand your spending with Cloud Billing reports: This document describes the capabilities of the built-in reports (Options A and B). It shows they are designed for viewing trends and high-level breakdowns using pre-set filters like project, service, or SKU, but it does not mention capabilities for creating custom, dynamically calculated metrics from raw data.
Source: Google Cloud, Cloud Billing Documentation, "View your billing reports and cost trends".

Question: 21

You have an application that runs on Compute Engine VM instances in a custom Virtual Private Cloud
(VPC). Your company's security policies only allow the use to internal IP addresses on VM instances
and do not let VM instances connect to the internet. You need to ensure that the application can access a file hosted in a Cloud Storage bucket within your project. What should you do?

A. Enable Private Service Access on the Cloud Storage Bucket.

B. Add slorage.googleapis.com to the list of restricted services in a VPC Service Controls perimeter
and add your project to the list to protected projects.

C. Enable Private Google Access on the subnet within the custom VPC.

D. Deploy a Cloud NAT instance and route the traffic to the dedicated IP address of the Cloud Storage
bucket.

Answer:

C

Explanation:

Private Google Access allows virtual machine (VM) instances with only internal IP addresses to reach the external IP addresses of Google APIs and services, including Cloud Storage. When enabled on a subnet, it provides a path for traffic from the VMs to services like storage.googleapis.com without requiring an external IP address on the VM or routing traffic through the public internet. The traffic remains within Google's private network, satisfying the security requirement of no internet connectivity while enabling access to necessary Google services.

Why Incorrect Options are Wrong:

A. Enable Private Service Access on the Cloud Storage Bucket.
Private Service Access is used to connect your VPC to Google-managed services (like Cloud SQL) that reside in a separate, Google-owned VPC. It is not applicable for accessing global APIs like Cloud Storage.
B. Add storage.googleapis.com to the list of restricted services in a VPC Service Controls perimeter and add your project to the list to protected projects.
VPC Service Controls is a security feature to prevent data exfiltration by creating a service perimeter. It does not provide the network connectivity needed for an internal-only VM to reach the service in the first place.

D. Deploy a Cloud NAT instance and route the traffic to the dedicated IP address of the Cloud Storage bucket.
Cloud NAT is primarily used to provide instances without external IPs with outbound access to the public internet. This violates the stated security policy. Private Google Access is the specific feature for accessing Google APIs privately.

References:

1. Google Cloud Documentation, "Private Google Access overview": "With Private Google Access, VMs that only have internal IP addresses (no external IP addresses) can reach the external IP addresses of Google APIs and services. .... You can use Private Google Access to access the external IP addresses of most Google APIs and services, including Cloud Storage..." This directly supports option C.
2. Google Cloud Documentation, "Choose a Cloud NAT product": "Cloud NAT enables Google Cloud virtual machine (VM) instances without external IP addresses and GKE clusters to connect to the internet." This confirms that Cloud NAT is for internet access, making option D incorrect as it violates the policy.
3. Google Cloud Documentation, "Private Service Access": "Private service access is a private connection between your VPC network and a network in a Google or third-party service. .... For example, you can use private service access to connect to Cloud SQL..." This shows that option A is for a different type of service connection.

4. Google Cloud Documentation, "VPC Service Controls overview": "VPC Service Controls helps you mitigate the risk of data exfiltration from your Google-managed services..." This confirms that VPC Service Controls (option B) is a security measure, not a connectivity solution for this scenario.

Question: 22

Your company completed the acquisition of a startup and is now merging the IT systems of both companies. The startup had a production Google Cloud project in their organization. You need to move this project into your organization and ensure that the project is billed lo your organization. You
want to accomplish this task with minimal effort. What should you do?

A. Use the projects. move method to move the project to your organization. Update the billing account of the project to that of your organization.

B. Ensure that you have an Organization Administrator Identity and Access Management (IAM) role
assigned to you in both organizations. Navigate to the Resource Manager in the startup's Google Cloud organization, and drag the project to your company's organization.

C. Create a Private Catalog tor the Google Cloud Marketplace, and upload the resources of the startup's production project to the Catalog. Share the Catalog with your organization, and deploy the
resources in your company's project.

D. Create an infrastructure-as-code template tor all resources in the project by using Terraform. and
deploy that template to a new project in your organization. Delete the protect from the startup's Google Cloud organization.

Answer:

A

Explanation:

The most direct and efficient method to transfer a project between organizations is using the Resource Manager's project move functionality. This is accomplished via the projects.move API method, which is also used by the gcloud projects move command. This process moves the project and all its resources, policies, and configurations intact. However, moving a project does not automatically change its associated billing account. To fulfill the requirement that the project is billed to the new organization, you must perform a second, distinct step: updating the project's billing account to one associated with the new organization. This two-step process is the standard procedure and represents the minimal effort required.

Why Incorrect Options are Wrong:

B: This option is incomplete. While having the correct permissions (like Organization Administrator) is necessary for the move, it omits the explicit and critical step of changing the project's billing account.

C: This is an overly complex and incorrect approach. Private Catalog is for managing and deploying approved solutions, not for migrating existing production projects. This method would require recreating all resources, leading to significant effort and downtime.

D: This method involves recreating the entire project's infrastructure from code, which is the opposite of "minimal effort." It doesn't move the existing project but rather creates a new one, requiring a separate, complex data migration strategy and causing service disruption.

References:

1. Official Google Cloud Documentation - Moving a project:
"When you move a project, its original billing account will continue to be used... To change the billing account, you must have the billing.projectManager role on the destination billing account and the resourcemanager.projectBillingManager role on the project." This confirms that moving the project and changing the billing account are two separate, required steps.
Source: Google Cloud Documentation, Resource Manager, "Moving a project", Section: "Effect on billing".

2. Official Google Cloud Documentation - gcloud projects move:
The command gcloud projects move is the command-line interface for the projects.move API method. The documentation outlines the process for moving a project to a new organization or folder.
Source: Google Cloud SDK Documentation, gcloud projects move.
3. Official Google Cloud Documentation - Modifying a project's billing account:
"You can change the billing account that is used to pay for a project." This page details the permissions and steps required to link a project to a different billing account, confirming it is a distinct action from moving the project's resource hierarchy.
Source: Google Cloud Billing Documentation, "Enable, disable, or change billing for a project".

Question: 23

All development (dev) teams in your organization are located in the United States. Each dev team has its own Google Cloud project. You want to restrict access so that each dev team can only create
cloud resources in the United States (US). What should you do?

A. Create a folder to contain all the dev projects Create an organization policy to limit resources in US
locations.

B. Create an organization to contain all the dev projects. Create an Identity and Access Management
(IAM) policy to limit the resources in US regions.

C. Create an Identity and Access Management

D. Create an Identity and Access Management (IAM)policy to restrict the resources locations in all
dev projects. Apply the policy to all dev roles.

Answer:

A

Explanation:

The Google Cloud Organization Policy Service is designed to provide centralized, programmatic control over an organization's cloud resources. To restrict the physical location of newly created resources, the gcp.resourceLocations constraint should be used. The most efficient and scalable method to apply this policy to a specific group of projects (like all development projects) is to group them into a folder. The organization policy is then applied to this folder, and all projects within it will inherit the constraint, ensuring resources are only created in the specified US locations.

Why Incorrect Options are Wrong:

B. Identity and Access Management (IAM) policies grant permissions to principals (who can do what), but they do not enforce constraints on resource attributes like location (where).
C. This is incorrect for the same reason as B; IAM is the wrong tool for enforcing location-based restrictions. This is the specific purpose of the Organization Policy Service.
D. This is incorrect for two reasons: IAM does not control resource locations, and policies are bound to resources (like projects or folders), not to roles.

References:

1. Google Cloud Documentation, "Restricting resource locations": "To restrict the locations where your organization's resources can be created, you can use a resource locations organization policy. The resource locations organization policy constraint is gcp.resourceLocations." This document explicitly states that the gcp.resourceLocations constraint is the correct tool.
2. Google Cloud Documentation, "Organization Policy Constraints", gcp.resourceLocations: "Defines the set of locations where location-based Google Cloud resources can be created... This constraint will be checked at resource creation time." This confirms the specific constraint and its function.
3. Google Cloud Documentation, "Resource hierarchy for access control": "The Google Cloud resource hierarchy allows you to group projects under folders, and folders and projects under the organization... Policies set at higher levels in the hierarchy are inherited by the resources below them." This supports using a folder to apply the policy to multiple projects efficiently.
4. Google Cloud Documentation, "Overview of IAM": "IAM lets you grant granular access to specific Google Cloud resources and helps prevent access to other resources... IAM lets you adopt the security principle of least privilege". This documentation clarifies that IAM's focus is on permissions, not resource configuration constraints like location.

Question: 24

You are configuring Cloud DNS. You want !to create DNS records to point home.mydomain.com, mydomain.com. and www.mydomain.com to the IP address of your Google Cloud load balancer. What should you do?

A. Create one CNAME record to point mydomain.com to the load balancer, and create two A records
to point WWW and HOME lo mydomain.com respectively.

B. Create one CNAME record to point mydomain.com to the load balancer, and create two AAAA records to point WWW and HOME to mydomain.com respectively.

C. Create one A record to point mydomain.com to the load balancer, and create two CNAME records
to point WWW and HOME to mydomain.com respectively.

D. Create one A record to point mydomain.com lo the load balancer, and create two NS records to
point WWW and HOME to mydomain.com respectively.

Answer:

C

Explanation:

The most appropriate configuration is to create an A record for the apex domain (mydomain.com) to point directly to the load balancer's IPv4 address. An A record is required for the apex domain because a CNAME record cannot be used at the zone apex (the root of a domain), as it must coexist with other records like SOA and NS.
For the subdomains (www.mydomain.com and home.mydomain.com), CNAME records should be created to point to the apex domain (mydomain.com). This approach is efficient because if the load balancer's IP address changes, only the single A record for mydomain.com needs to be updated, and the subdomains will automatically resolve to the new IP.

Why Incorrect Options are Wrong:

A. A CNAME record cannot be used for the zone apex (mydomain.com). Additionally, A records map hostnames to IP addresses, not to other hostnames.
B. A CNAME record cannot be used for the zone apex. AAAA records are for mapping to IPv6 addresses, not for aliasing one hostname to another.
D. NS (Name Server) records are used to delegate a DNS zone to a set of authoritative name servers, not to point a hostname to an IP address or another hostname.

References:

1. Google Cloud Documentation, Cloud DNS, "Add, modify, and delete records": Under the section for "CNAME record," the documentation states, "A CNAME record cannot exist at the zone apex." This directly invalidates options A and B. The documentation also defines an "A record" as mapping a domain name to an IPv4 address, which is the correct use case for the apex domain in this scenario.
2. Google Cloud Documentation, Cloud DNS, "Supported DNS record types": This page details the function of each record type. It confirms that A records are for IPv4 addresses, CNAMEs are for canonical names (aliases), and NS records are for name server delegation, supporting the reasoning for selecting C and rejecting D.
3. Internet Engineering Task Force (IETF), RFC 1034, "DOMAIN NAMES - CONCEPTS AND FACILITIES", Section 3.6.2: This foundational document for DNS specifies the CNAME rule: "If a CNAME RR is present at a node, no other data should be present". Since the zone apex must have SOA and NS records, a CNAME cannot be placed there. This provides the technical basis for why options A and B are incorrect.
4. Internet Engineering Task Force (IETF), RFC 1912, "Common DNS Operational and Configuration Errors", Section 2.4: This document clarifies common mistakes and states, "A CNAME record is not allowed to coexist with any other data. In other words, if suzy.podunk.xx is an alias for sue.podunk.xx, you can't also have an MX record for suzy.podunk.xx." This reinforces the rule against using a CNAME at the zone ape x

Question: 25

You have two subnets (subnet-a and subnet-b) in the default VPC. Your database servers are running
in subnet-
a. Your application servers and web servers are running in subnet-b. You want to configure a firewall
rule that only allows database traffic from the application servers to the database servers. What should you do?

A.
* Create service accounts sa-app and sa-db.
• Associate service account: sa-app with the application servers and the service account sa-db with
the database servers.
• Create an ingress firewall rule to allow network traffic from source service account sa-app to target
service account sa-db.

B.
• Create network tags app-server and db-server.
• Add the app-server lag lo the application servers and the db-server lag to the database servers.
• Create an egress firewall rule to allow network traffic from source network tag app-server to target
network tag db-server.

C.
* Create a service account sa-app and a network tag db-server.
* Associate the service account sa-app with the application servers and the network tag db-server with
the database servers.
• Create an ingress firewall rule to allow network traffic from source VPC IP addresses and target the
subnet-a IP addresses.

D.
• Create a network lag app-server and service account sa-db.
• Add the tag to the application servers and associate the service account with the database servers.
• Create an egress firewall rule to allow network traffic from source network tag app-server to target
service account sa-db.

Answer:

A

Explanation:

This solution correctly implements the principle of least privilege by using identity-based controls. An ingress firewall rule is created to control traffic entering the database servers. By specifying a target service account (sa-db) for the database servers and a source service account (sa-app) for the application servers, the rule precisely allows traffic only from instances with the sa-app identity to instances with the sa-db identity. This is a secure and scalable method that is not dependent on network location or IP addresses, which can change.

Why Incorrect Options are Wrong:

B: This option incorrectly describes an egress rule. The destination for an egress rule must be an IP CIDR range, not a target network tag. A target tag is used for ingress rules.
C: The firewall rule described is overly permissive. It allows traffic from all VPC IP addresses to the entire database subnet, which violates the requirement to only allow traffic from the application servers.
D: This option incorrectly describes an egress rule. The destination for an egress rule must be an IP CIDR range, not a target service account. A target service account is used for ingress rules.

References:

1. Google Cloud Documentation - VPC firewall rules: "For ingress rules, you can use service accounts to define the source. For egress rules, you can use service accounts to define the destination... Using service accounts for the source of ingress rules and the destination of egress rules is more specific than using network tags." This supports using service accounts for source/target specification.
2. Google Cloud Documentation - Use firewall rules: Under the "Components of a firewall rule" section, the table for "Source for ingress rules" lists "Source service accounts". The table for "Destination for egress rules" lists only "Destination IPv4 or IPv6 ranges". This confirms that options B and D, which specify a target tag or service account for an egress rule's destination, are invalid configurations.
3. Google Cloud Documentation - Firewall rules overview: "You can configure firewall rules by using network tags or service accounts... If you need stricter control over rules, we recommend that you use service accounts instead of network tags." This highlights that the approach in option A is a recommended best practice for secure configurations.

Question: 26

Your learn wants to deploy a specific content management system (CMS) solution lo Google Cloud.
You need a quick and easy way to deploy and install the solution. What should you do?

A. Search for the CMS solution in Google Cloud Marketplace. Use gcloud CLI to deploy the solution.

B. Search for the CMS solution in Google Cloud Marketplace. Deploy the solution directly from Cloud
Marketplace.

C. Search for the CMS solution in Google Cloud Marketplace. Use Terraform and the Cloud Marketplace ID to deploy the solution with the appropriate parameters.

D. Use the installation guide of the CMS provider. Perform the installation through your configuration
management system.

Answer:

B

Explanation:

Google Cloud Marketplace is specifically designed to offer pre-configured and optimized software solutions that can be deployed rapidly on Google Cloud. For a common application like a Content Management System (CMS), the Marketplace provides a streamlined, web-based interface that automates the creation of all necessary resources (e.g., Compute Engine instances, disks, firewall rules) and the installation of the software. This "few-click" deployment process is the most direct, quickest, and easiest method, perfectly aligning with the user's requirements.

Why Incorrect Options are Wrong:

A. While the gcloud CLI can deploy Marketplace solutions, it is generally more complex and less intuitive than using the graphical user interface, contradicting the "quick and easy" requirement.
C. Using Terraform is an excellent practice for infrastructure-as-code and repeatable deployments, but it requires writing configuration files and is more involved than a direct deployment from the Marketplace UI.
D. A manual installation using the provider's guide is the most time-consuming and complex option. It requires manually provisioning infrastructure and handling all software dependencies and configurations.

References:

1. Google Cloud Documentation, "Overview of Google Cloud Marketplace": The documentation states, "Google Cloud Marketplace lets you quickly deploy functional software packages that run on Google Cloud... Some solutions are free to use, and for others, you pay for the software, or for the Google Cloud resources that you use, or both." This confirms the Marketplace is the intended tool for quick deployments.
2. Google Cloud Documentation, "Deploying a VM-based solution": This guide details the process of deploying a solution directly from the Marketplace console. The steps involve selecting a product and filling out a simple web form, after which "Cloud Deployment Manager deploys the solution for you." This demonstrates the ease of use compared to CLI or manual methods.
3. Google Cloud Documentation, "Deploying a solution by using Terraform": This document outlines the multi-step process for using Terraform, which includes creating a .tf configuration file. This confirms that while possible, it is not the simplest or quickest method for a one-time deployment.

Question: 27

You are working for a startup that was officially registered as a business 6 months ago. As your customer base grows, your use of Google Cloud increases. You want to allow all engineers to create
new projects without asking them for their credit card information. What should you do?

A. Create a Billing account, associate a payment method with it, and provide all project creators with
permission to associate that billing account with their projects.

B. Grant all engineer's permission to create their own billing accounts for each new project.

C. Apply for monthly invoiced billing, and have a single invoice tor the project paid by the finance team.

D. Create a billing account, associate it with a monthly purchase order (PO), and send the PO to Google Cloud.

Answer:

A

Explanation:

The most effective and standard practice is to centralize billing management. This is achieved by creating a single Cloud Billing account for the organization and associating a corporate payment method. To allow engineers to create projects that are paid for by the company, they must be granted an IAM role on that billing account which includes the billing.projects.link permission. The Billing Account User role (roles/billing.user) is a predefined role that grants this permission. This setup enables engineers to create new projects and link them to the central billing account, solving the problem without requiring them to use personal credit cards and providing the company with centralized financial oversight.

Why Incorrect Options are Wrong:

B. This approach decentralizes billing, creating significant administrative overhead and making cost tracking nearly impossible. It also contradicts the goal of not using engineers' personal payment information.
C. Invoiced billing is a payment option, not the mechanism that enables project creation. A startup may not meet the eligibility criteria, and this option omits the crucial step of granting permissions.
D. A Purchase Order (PO) is a financial instrument used with invoiced billing for tracking purposes. It does not solve the core problem of granting engineers permission to use a central billing account.

References:

1. Google Cloud Documentation, "Overview of Cloud Billing concepts": This document states, "A Cloud Billing account is set up in Google Cloud and is used to pay for usage costs in your Google Cloud projects... To use Google Cloud resources in a project, billing must be enabled on the project. Billing is enabled when the project is linked to an active Cloud Billing account." This supports the fundamental need for a central billing account linked to projects.
2. Google Cloud Documentation, "Control access to Cloud Billing accounts with IAM," Section: "Billing account permissions": This page details the permissions required to manage billing. Specifically, the billing.projects.link permission "allows a user to link projects to the billing account." This is the exact permission needed by the engineers in the scenario.
3. Google Cloud Documentation, "Understand predefined Cloud Billing IAM roles," Section: "Billing Account User": The roles/billing.user role is described as granting permissions to link projects to a billing account. This is the standard role assigned to users who need to create projects under a corporate billing account.
4. Google Cloud Documentation, "Request invoiced billing," Section: "Eligibility requirements": This document outlines the criteria for invoiced billing, which includes being a registered business for at least one year and having a minimum spend, confirming that a 6-month-old startup might not qualify.

Question: 28

You recently received a new Google Cloud project with an attached billing account where you will work. You need to create instances, set firewalls, and store data in Cloud Storage. You want to follow
Google-recommended practices. What should you do?

A. Use the gcloud CLI services enable cloudresourcemanager.googleapis.com command to enable
all resources.

B. Use the gcloud services enable compute.googleapis.com command to enable Compute Engine and the gcloud services enable storage-api.googleapis.com command to enable the Cloud Storage
APIs.

C. Open the Google Cloud console and enable all Google Cloud APIs from the API dashboard.

D. Open the Google Cloud console and run gcloud init --project in a Cloud Shell.

Answer:

B

Explanation:

The principle of least privilege is a Google-recommended best practice, which dictates that you should only enable the specific APIs required for your tasks. The question requires creating instances and firewalls (managed by the Compute Engine API) and storing data (managed by the Cloud Storage API). Option B correctly identifies the specific gcloud commands to enable only these two necessary services: compute.googleapis.com for Compute Engine and storage-api.googleapis.com (or storage.googleapis.com) for Cloud Storage. This approach ensures that no unnecessary services are enabled, which minimizes the security attack surface and potential for unintended usage.

Why Incorrect Options are Wrong:

A. The cloudresourcemanager.googleapis.com API is for programmatically managing projects, folders, and organizations. It does not enable other services like Compute Engine or Cloud Storage.
C. Enabling all APIs is a significant security risk as it violates the principle of least privilege. It exposes the project to services that are not needed, increasing the potential attack surface.
D. The gcloud init command is used to initialize or configure settings for the gcloud command-line tool, such as the default project, account, and region. It does not enable any APIs.

References:

1. Official Google Cloud Documentation, Enabling and disabling services: "Before you can use a Google Cloud service, you must first enable the service's API for your Google Cloud project... We recommend that you enable APIs for only the services that your apps actually use." This supports the principle of enabling specific APIs. The page also provides the syntax gcloud services enable SERVICENAME, which matches option B.
Source: Google Cloud Documentation, "Enabling and disabling services".
2. Official Google Cloud Documentation, gcloud services enable command reference: This document confirms that gcloud services enable SERVICE... is the correct command to enable one or more APIs for a project.
Source: Google Cloud SDK Documentation, gcloud services enable.
3. Official Google Cloud Security Foundations Guide, Section 2.3, "Manage IAM permissions": This guide emphasizes the principle of least privilege. While discussing IAM, the principle extends to all resources, including enabling only necessary APIs. "Grant roles at the smallest scope... grant predefined roles instead of primitive roles... to enforce the principle of least privilege."
Source: Google Cloud Security Foundations Guide PDF, Page 13.
4. Official Google Cloud Documentation, gcloud init command reference: This document describes the function of gcloud init as: "Initializes or reinitializes gcloud CLI settings." It makes no mention of enabling APIs, confirming that option D is incorrect.
Source: Google Cloud SDK Documentation, gcloud init

Question: 29

Your company is using Google Workspace to manage employee accounts. Anticipated growth will increase the number of personnel from 100 employees to 1.000 employees within 2 years. Most employees will need access to your company's Google Cloud account. The systems and processes will
need to support 10x growth without performance degradation, unnecessary complexity, or security
issues. What should you do?

A. Migrate the users to Active Directory. Connect the Human Resources system to Active Directory.
Turn on Google Cloud Directory Sync (GCDS) for Cloud Identity. Turn on Identity Federation from Cloud Identity to Active Directory.

B. Organize the users in Cloud Identity into groups. Enforce multi-factor authentication in Cloud Identity.

C. Turn on identity federation between Cloud Identity and Google Workspace. Enforce multi-factor authentication for domain wide delegation.

D. Use a third-party identity provider service through federation. Synchronize the users from Google
Workplace to the third-party provider in real time.

Answer:

B

Explanation:

The company already uses Google Workspace, which means their user identities are managed by Cloud Identity. The most effective, scalable, and secure solution is to leverage this existing infrastructure. Organizing users into Google Groups and assigning IAM roles to these groups is a Google-recommended best practice for managing permissions at scale. As new employees join, they can be added to the appropriate groups to inherit necessary permissions, simplifying administration. Enforcing multi-factor authentication (MFA), known as 2-Step Verification in Google, is a critical security measure that protects user accounts from unauthorized access, which is essential as the organization grows. This approach avoids unnecessary complexity and cost while enhancing security and scalability.

Why Incorrect Options are Wrong:

A. This introduces significant and unnecessary complexity by adding Active Directory. Migrating users and setting up synchronization and federation is a major project when a suitable identity provider is already in place.

C. This is technically incorrect. Google Workspace and Cloud Identity are part of an integrated identity platform; you do not federate between them. It also misapplies MFA to domain-wide delegation instead of user accounts.

D. Introducing a third-party identity provider adds complexity, cost, and another system to manage. Synchronizing from Google Workspace to a third-party provider is also an unconventional and illogical data flow.

References:

1. Using Groups for Access Control: Google Cloud's official documentation on Identity and Access Management (IAM) explicitly recommends using groups to manage roles for multiple users. This simplifies administration and scales effectively.
Source: Google Cloud Documentation, "Best practices for using IAM", Section: "Use groups and roles to manage access".

2. Google Workspace and Cloud Identity Integration: Google Workspace accounts are inherently Cloud Identity accounts. This means there is no need for federation or a separate identity system.
Source: Google Cloud Documentation, "Overview of Cloud Identity and Access Management", Section: "Identities".

3. Enforcing Multi-Factor Authentication (MFA): The Google Workspace Admin help center details how to enforce 2-Step Verification (Google's term for MFA) for all users in an organization to enhance security.
Source: Google Workspace Admin Help, "Protect your business with 2-Step Verification", Section: "Deploy 2-Step Verification".

4. Complexity of Federation: Setting up federation with an external identity provider (as suggested in A and D) is a multi-step process intended for organizations that already have an established external IdP as their source of truth, not for those already using Google Workspace.
Source: Google Cloud Documentation, "Setting up identity federation", provides an overview of the required configuration, highlighting the added complexity.

Question: 30

Your application development team has created Docker images for an application that will be deployed on Google Cloud. Your team does not want to manage the infrastructure associated with
this application. You need to ensure that the application can scale automatically as it gains popularity. What should you do?

A. Create an Instance template with the container image, and deploy a Managed Instance Group with Autoscaling.

B. Upload Docker images to Artifact Registry, and deploy the application on Google Kubernetes Engine using Standard mode.

C. Upload Docker images to the Cloud Storage, and deploy the application on Google Kubernetes Engine using Standard mode.

D. Upload Docker images to Artifact Registry, and deploy the application on Cloud Run.

Answer:

D

Explanation:

Cloud Run is a fully managed, serverless platform that allows you to run stateless containers without managing the underlying infrastructure. It automatically scales the number of container instances up or down based on traffic, including scaling down to zero when there are no requests, which directly meets the requirements. Artifact Registry is the recommended, fully managed service in Google Cloud for storing, managing, and securing container images. This combination provides a completely serverless solution that requires no infrastructure management from the development team while ensuring automatic scalability.

Why Incorrect Options are Wrong:

A. A Managed Instance Group runs on Compute Engine virtual machines. This requires managing the underlying OS and instance configurations, which violates the "do not want to manage the infrastructure" requirement.
B. Google Kubernetes Engine (GKE) in Standard mode requires you to manage the worker node pools (the underlying VMs). This includes tasks like node upgrades and capacity planning, which constitutes infrastructure management.
C. Cloud Storage is designed for object storage, not as a primary repository for Docker images. Furthermore, GKE in Standard mode requires infrastructure management, making this option incorrect on two counts.

References:

1. Google Cloud Documentation, "Cloud Run overview": "Cloud Run is a managed compute platform that lets you run containers directly on top of Google's scalable infrastructure. You can deploy code written in any programming language on Cloud Run if you can build a container image from it. ... With Cloud Run, you don't need to manage infrastructure..."
2. Google Cloud Documentation, "Choosing a compute option": This document compares various compute services. It categorizes Cloud Run as "Serverless" and highlights "No infrastructure management." In contrast, it places Compute Engine (used in option A) under "Infrastructure as a Service (IaaS)" and GKE (used in options B and C) under "Containers as a Service (CaaS)," both of which involve more infrastructure management than serverless options.
3. Google Cloud Documentation, "Artifact Registry overview": "Artifact Registry is a single place for your organization to manage container images and language packages (such as Maven and npm). It is fully integrated with Google Cloud's tooling and runtimes..." This confirms Artifact Registry as the correct repository for Docker images.
4. Google Cloud Documentation, "Comparing GKE cluster modes: Autopilot and Standard": "In Standard mode, you manage your cluster's underlying infrastructure, which gives you node configuration flexibility." This statement confirms that GKE Standard mode involves infrastructure management, which the question explicitly seeks to avoid.

Question: 31

Your team is using Linux instances on Google Cloud. You need to ensure that your team logs in to these instances in the most secure and cost efficient way. What should you do?

A. Attach a public IP to the instances and allow incoming connections from the internet on port 22 for
SSH.

B. Use a third party tool to provide remote access to the instances.

C. Use the gcloud compute ssh command with the --tunnel-through-iap flag. Allow ingress traffic from the IP range 35.235.240.0/20 on port 22.

D. Create a bastion host with public internet access. Create the SSH tunnel to the instance through
the bastion host.

Answer:

C

Explanation:

Using Identity-Aware Proxy (IAP) for TCP forwarding is Google Cloud's recommended method for securing administrative access like SSH. This approach eliminates the need for bastion hosts or public IP addresses on your instances, significantly reducing the external attack surface. Access is controlled through granular IAM permissions (roles/iap.tunnelResourceAccessor) rather than network-level firewall rules open to the internet. The gcloud compute ssh --tunnel-through-iap command automates this secure connection. This is also the most cost-efficient solution as IAP for TCP forwarding is a zero-cost service, unlike a bastion host which requires a continuously running, billable VM.

Why Incorrect Options are Wrong:

A. Attaching a public IP and opening port 22 to the internet is highly insecure. It exposes the instance to constant brute-force attacks and automated scans from malicious actors.
B. Using a third-party tool introduces additional costs, potential security vulnerabilities, and management overhead. It is less integrated and secure than a native Google Cloud solution like IAP.
D. A bastion host is a valid security pattern but is not the most cost-efficient, as it requires running and maintaining an additional VM. It also has more operational overhead than the fully managed IAP service.

References:

1. Google Cloud Documentation - Use IAP for TCP forwarding: "Using IAP's TCP forwarding feature... you can control who can access administrative services like SSH and RDP on your backends from the public internet. This removes the need to run a bastion host..." This document also specifies the required firewall rule: "Create a firewall rule that allows ingress traffic from IAP's TCP forwarding IP range, 35.235.240.0/20, to the ports on your instances."
Source: Google Cloud Documentation, "Use IAP for TCP forwarding", Section: "When to use IAP TCP forwarding" and "Create a firewall rule".
2. Google Cloud Documentation - Securely connecting to VM instances: This guide compares connection methods and highlights the benefits of IAP. "IAP lets you establish a central authorization layer for applications that are accessed by HTTPS, so you can use an application-level access control model instead of relying on network-level firewalls." It positions IAP as a superior alternative to bastion hosts for securing access.
Source: Google Cloud Architecture Center, "Securely connecting to VM instances", Section: "Identity-Aware Proxy".
3. Google Cloud Documentation - Choose a connection option: This document explicitly contrasts IAP with bastion hosts. For IAP, it states you can connect "without requiring your VM instances to have external IP addresses." For bastion hosts, it notes the requirement to "provision and maintain an additional instance," which implies both cost and management overhead.
Source: Google Cloud Documentation, Compute Engine, "Choose a connection option".

Question: 32

You are migrating a business critical application from your local data center into Google Cloud. As part of your high-availability strategy, you want to ensure that any data used by the application will be immediately available if a zonal failure occurs. What should you do?

A. Store the application data on a zonal persistent disk. Create a snapshot schedule for the disk. If an
outage occurs, create a new disk from the most recent snapshot and attach it to a new VM in another
zone.

B. Store the application data on a zonal persistent disk. If an outage occurs, create an instance in another zone with this disk attached.

C. Store the application data on a regional persistent disk. Create a snapshot schedule for the disk. If
an outage occurs, create a new disk from the most recent snapshot and attach it to a new VM in another zone.

D. Store the application data on a regional persistent disk If an outage occurs, create an instance in
another zone with this disk attached.

Answer:

D

Explanation:

Regional persistent disks are designed for high availability by synchronously replicating data between two zones within the same region. If the primary zone containing the virtual machine (VM) fails, the application can be failed over to the other zone. You can force-attach the regional persistent disk to a new VM instance in the secondary zone. This process ensures that the data is immediately available with minimal downtime, which is critical for a business-critical application as specified in the scenario.

Why Incorrect Options are Wrong:

A. Zonal persistent disks are confined to a single zone. If that zone fails, the disk becomes inaccessible. Restoring from a snapshot introduces latency and potential data loss (RPO), failing the "immediately available" requirement.
B. A zonal persistent disk cannot be attached to a VM in a different zone. This action is technically impossible, making it an invalid solution for a zonal failure.
C. While using the correct disk type (regional), the recovery process is flawed. Restoring from a snapshot is unnecessary and slow; the primary benefit of a regional disk is its immediate

availability in the failover zone.

References:

1. Google Cloud Documentation, "High availability with regional persistent disks": "Regional persistent disks provide synchronous replication of data between two zones in a region. .... If your primary zone becomes unavailable, you can fail over to your secondary zone. In the secondary zone, you can force-attach the regional persistent disk to a new VM instance." (See section: "Failing over your regional persistent disk").
2. Google Cloud Documentation, "About persistent disk snapshots": "Snapshots are global resources... Use snapshots to protect your data from unexpected failures. .. Creating a new persistent disk from a snapshot takes time. ." This highlights that snapshots are for disaster recovery/backups, not instantaneous high-availability failover.
3. Google Cloud Documentation, "Persistent disk types": "Zonal persistent disks are located in a single zone. If a zone becomes unavailable, all zonal persistent disks in that zone are unavailable until the zone is restored." This confirms that options A and B are unsuitable for zonal failure scenarios.

Question: 33

The DevOps group in your organization needs full control of Compute Engine resources in your development project. However, they should not have permission to create or update any other resources in the project. You want to follow Google's recommendations for setting permissions for the DevOps group. What should you do?

A. Grant the basic role roles/viewer and the predefined role roles/compute.admin to the DevOps group.

B. Create an IAM policy and grant all compute. instanceAdmln." permissions to the policy Attach the
policy to the DevOps group.

C. Create a custom role at the folder level and grant all compute. instanceAdmln. * permissions to the role Grant the custom role to the DevOps group.

D. Grant the basic role roles/editor to the DevOps group.

Answer:

A

Explanation:

This approach adheres to the principle of least privilege, a core Google recommendation. The predefined role roles/compute.admin grants the DevOps group full control over all Compute Engine resources, fulfilling their primary requirement. The basic role roles/viewer provides necessary read-only access to other resources in the project, which is often needed for context (e.g., viewing network configurations or storage buckets), without granting permissions to modify them. This combination precisely meets the scenario's constraints by isolating modification permissions strictly to Compute Engine while allowing project-wide visibility.

Why Incorrect Options are Wrong:

B. This option misrepresents how IAM works. You do not grant permissions to a policy; a policy binds members to roles. The permission string is also invalid.
C. A predefined role (roles/compute.admin) already exists for this purpose, which is preferred over creating a custom role. Applying it at the folder level is also incorrect scope.
D. The roles/editor basic role grants broad permissions to create and update all resources in the project, directly violating the requirement to restrict modification permissions to Compute Engine.

References:

1. Google Cloud Documentation - IAM Basic and predefined roles reference: This document explicitly states that the Editor role (roles/editor) grants "permissions to create, modify, and delete all resources." It also describes the Compute Admin role (roles/compute.admin) as providing "Full control of all Compute Engine resources." This supports why option D is wrong and option A is correct. (See "Basic roles" and "Compute Engine roles" sections).
2. Google Cloud Documentation - IAM Overview, Principle of least privilege: "When you grant roles, grant the least permissive role that's required. For example, if a user only needs to view resources in a project, grant them the Viewer role, not the Owner role." This principle supports choosing the specific roles/compute.admin over the broad roles/editor.
3. Google Cloud Documentation - Understanding roles: "Predefined roles are created and maintained by Google... Google automatically updates their permissions as new features and services are added to Google Cloud. When possible, we recommend that you use predefined roles instead of custom roles." This supports the choice of a predefined role (as in option A) over a custom one (as in option C).

Question: 34

Your team is running an on-premises ecommerce application. The application contains a complex set
of microservices written in Python, and each microservice is running on Docker containers. Configurations are injected by using environment variables. You need to deploy your current application to a serverless Google Cloud cloud solution. What should you do?

A. Use your existing CI/CD pipeline Use the generated Docker images and deploy them to Cloud Run.
Update the configurations and the required endpoints.

B. Use your existing continuous integration and delivery (CI/CD) pipeline. Use the generated Docker
images and deploy them to Cloud Function. Use the same configuration as on-premises.

C. Use the existing codebase and deploy each service as a separate Cloud Function Update the configurations and the required endpoints.

D. Use your existing codebase and deploy each service as a separate Cloud Run Use the same configurations as on-premises.

Answer:

A

Explanation:

The most effective strategy is to leverage the existing containerized architecture. Cloud Run is Google Cloud's serverless platform designed specifically for running stateless containers. Since the application's microservices are already packaged as Docker images, they can be deployed directly to Cloud Run with minimal modification. This approach utilizes the existing CI/CD pipeline, simply retargeting the deployment step to Cloud Run. It correctly acknowledges that configurations, such as database connection strings and service endpoints managed via environment variables, will need to be updated for the new cloud environment, which is a standard and necessary part of any migration.

Why Incorrect Options are Wrong:

B: Cloud Functions is less suitable for running complex, containerized web services than Cloud Run. More importantly, it is unrealistic to assume the same on-premises configurations will work in the cloud without any changes.
C: This approach discards the significant advantage of having pre-built Docker containers. Refactoring the codebase for Cloud Functions would require unnecessary development effort compared to deploying the existing containers to Cloud Run.
D: While selecting Cloud Run is correct, this option is flawed because it incorrectly states that the

same on-premises configurations can be used. Migrating from on-premises to the cloud always requires configuration updates.

References:

1. Google Cloud Documentation, "What is Cloud Run?": "Cloud Run is a managed compute platform that lets you run containers directly on top of Google's scalable infrastructure. You can deploy code written in any programming language if you can build a container image from it. In fact, building container images is optional. If you're using Go, Node.js, Python, Java, .NET Core, or Ruby, you can use the source-based deployment option that builds the container for you." This confirms Cloud Run is the ideal platform for existing container images.
2. Google Cloud Documentation, "Choosing a serverless option": This document compares serverless options. It states, "Cloud Run is also a good choice if you want to migrate a containerized application from on-premises or from another cloud." This directly supports the scenario in the question. For Cloud Functions, it is positioned for "event-driven" applications, which is a less direct fit for a set of web microservices than Cloud Run.
3. Google Cloud Documentation, "Deploying container images": When deploying to Cloud Run, the documentation for the gcloud run deploy command shows flags like --set-env-vars for setting environment variables. This confirms that configurations are expected to be set or updated during deployment to the new cloud environment, invalidating the claims in options B and D that on-premises configurations can be used as-is.

4. Google Cloud Documentation, "Cloud Run common use cases": The documentation lists "APIs and microservices" as a primary use case, stating, "Quickly deploy microservices and scale them as needed without having to manage a Kubernetes cluster." This reinforces Cloud Run as the correct choice for a microservices-based application.

Question: 35

You are running a web application on Cloud Run for a few hundred users. Some of your users complain that the initial web page of the application takes much longer to load than the following pages. You want to follow Google's recommendations to mitigate the issue. What should you do?

A. Update your web application to use the protocol HTTP/2 instead of HTTP/1.1

B. Set the concurrency number to 1 for your Cloud Run service.

C. Set the maximum number of instances for your Cloud Run service to 100.

D. Set the minimum number of instances for your Cloud Run service to 3.

Answer:

D

Explanation:

The described issue, where an initial request is slow but subsequent ones are fast, is a classic symptom of a "cold start." In Cloud Run, services scale to zero instances by default during periods of no traffic. When a new request arrives, a new container instance must be provisioned, which introduces latency. To mitigate this, Google recommends setting a minimum number of instances. This ensures that a specified number of container instances are always running and "warm," ready to serve requests immediately. This directly eliminates the startup delay for incoming requests and resolves the slow initial page load problem.

Why Incorrect Options are Wrong:

A. Using HTTP/2 optimizes network transport for loading multiple assets but does not solve the server-side latency caused by provisioning a new container instance.
B. Setting concurrency to 1 would likely increase the frequency of cold starts, as more instances would be needed to handle the same amount of traffic.
C. Setting a maximum number of instances is a cost-control and safety feature to limit scaling; it does not keep instances warm to prevent cold starts.

References:

1. Google Cloud Documentation, "Configuring minimum instances": "To reduce latency and cold starts, you can configure Cloud Run to keep a minimum number of container instances running and ready to serve requests... When a request comes in that needs to be served by a new instance, Cloud Run starts a new instance. This is known as a cold start. Cold starts can result in high latency for the requests that require them."
2. Google Cloud Documentation, "General development tips for Cloud Run", Section: "Minimizing cold starts": "To permanently keep instances warm, use the min-instances setting. If you set the value of min-instances to 1 or greater, a specified number of container instances are kept running

and ready to serve requests, which reduces cold starts."
3. Google Cloud Documentation, "Container instance concurrency": "By default, each Cloud Run container instance can receive up to 80 requests at the same time... You may want to limit concurrency for instances where... your code cannot handle parallel requests." (This shows concurrency is not primarily for reducing initial latency).