CERT EMPIRE

Google Cloud Architect Exam Questions

Total Questions: 250+
Demo Questions: 30
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: Professional-Cloud-Architect Exam Dumps by Cert Empire

Question: 1

For this question, refer to the Mountkirk Games case study. Mountkirk Games wants you to design their new testing strategy. How should the test coverage differ from their existing backends on the other platforms?

A: Tests should scale well beyond the prior approaches.

B: Unit tests are no longer required, only end-to-end tests.

C: Tests should be applied after the release is in the production environment.

D: Tests should include directly testing the Google Cloud Platform (GCP) infrastructure.

Correct Answer:

A

Explanation:

The Mountkirk Games case study highlights that their existing monolithic backend struggles with scaling to meet unpredictable player demand. The new cloud-native architecture is explicitly designed to solve this problem. Consequently, the testing strategy must evolve to validate this new capability. The new approach requires performance, load, and stress testing that can operate at a scale far beyond their previous methods. This ensures the new system can handle the "several hundred thousand players" mentioned in the case study, a key difference from testing a traditional, less scalable monolith.

Why Incorrect Options are Wrong:

B: This is incorrect. In a microservices architecture, unit tests are more critical than ever to ensure the independent correctness of each service before integration, forming the base of the testing pyramid.

C: This is a high-risk anti-pattern. While some testing occurs in production (e.g., canary analysis), the primary testing effort must happen in pre-production environments to catch defects before they impact users.

D: This misinterprets the shared responsibility model. Google is responsible for testing its infrastructure. The customer (Mountkirk Games) is responsible for testing their application code and configuration that runs on that infrastructure.

References:

1. DevOps tech: Test automation I Google Cloud: In the section "Types of automated tests," the document discusses the importance of a balanced test portfolio, including performance

tests. It states, "Performance tests evaluate how a system performs under a particular workload... These tests can help you determine the reliability, speed, scalability, and responsiveness of an application." This supports the need for testing that can validate the new system's scalability (Answer A). The same document's "Test pyramid" section refutes option B by emphasizing a large base of unit tests.

2. Shared responsibility and fate I Security I Google Cloud: This document outlines the shared responsibility model. It clarifies that Google manages the security and reliability "of the cloud" (the infrastructure), while the customer is responsible for security and reliability "in the cloud" (their applications and configurations). This directly refutes the idea that Mountkirk should test GCP's infrastructure itself (Answer D).

3. Migrating a monolith to microservices on Google Kubernetes Engine I Google Cloud Architecture Center: This guide discusses the transition that Mountkirk is undertaking. In the "Testing" section, it notes, "In a microservices architecture, you must test the interactions between services... you must also perform integration testing and end-to-end testing." This reinforces that a comprehensive strategy is needed, not the elimination of test types (refuting B) or testing only in production (refuting C). The entire premise of moving to a scalable platform like GKE implies that testing for scale is a new, critical requirement.

Question: 2

For this question, refer to the Mountkirk Games case study. Mountkirk Games has deployed their new backend on Google Cloud Platform (GCP). You want to create a thorough testing process for new versions of the backend before they are released to the public. You want the testing environment to scale in an economical way. How should you design the process?

A: Create a scalable environment in GCP for simulating production load.

B: Use the existing infrastructure to test the GCP-based backend at scale.

C: Build stress tests into each component of your application using resources internal to GCP to simulate load.

D: Create a set of static environments in GCP to test different levels of load — for example, high, medium, and low.

Correct Answer:

A

Explanation:

The most effective and economical approach is to create a dedicated, scalable testing environment within Google Cloud. This allows for the simulation of production-level loads by dynamically scaling resources up for the duration of the test and scaling them down or terminating them afterward. This on-demand model is a core economic benefit of the cloud, ensuring that Mountkirk Games only pays for the testing infrastructure when it is actively in use, directly fulfilling the requirements for a scalable and economical process.

Why Incorrect Options are Wrong:

B: Using existing, non-GCP infrastructure to test a GCP-native backend is inefficient, introduces network latency, and fails to replicate the production environment accurately, leading to unreliable test results.

C: Building stress tests into individual components is a form of unit or integration testing; it does not address the need for end-to-end system load testing in a production-like environment.

D: Using static, pre-provisioned environments is not economical. These environments would incur costs even when idle, directly contradicting the requirement for a cost-effective solution.

References:

1. Google Cloud Architecture Framework, Reliability pillar, Testing for reliability: The framework emphasizes the importance of load testing to understand application behavior under stress. It states, "To get a realistic assessment of your application's performance, your test environment should mirror the production environment as closely as possible." This supports creating a dedicated, realistic environment in GCP. (Source: Google Cloud Documentation, "Architecture Framework: Reliability pillar", section: "Testing for reliability").

2. Google Cloud Blog, "Best practices for performance testing on Google Cloud": This article details the methodology for effective testing, stating, "A key principle of performance testing is to make your test environment as close to your production environment as possible... Google Cloud makes it easy to spin up a replica of your production environment, run your tests, and then tear it all down." This directly validates the approach in option A as both scalable and economical. (Source: Google Cloud Blog, "Best practices for performance testing on Google Cloud", Nov 19, 2020).

3. Google Cloud Solutions, "Disaster recovery planning guide": While focused on DR, this guide discusses testing methodologies that apply here. It notes the importance of testing in an isolated environment that mimics production to validate performance without impacting real users, reinforcing the need for a separate, scalable test environment. (Source: Google Cloud Documentation, "Disaster recovery planning guide", section: "Choosing a DR architecture").

Question: 3

For this question, refer to the Mountkirk Games case study. Mountkirk Games wants to set up a continuous delivery pipeline. Their architecture includes many small services that they want to be able to update and roll back quickly. Mountkirk Games has the following requirements: • Services are deployed redundantly across multiple regions in the US and Europe. • Only frontend services are exposed on the public internet. • They can provide a single frontend IP for their fleet of services. • Deployment artifacts are immutable. Which set of products should they use?

A: Google Cloud Storage, Google Cloud Dataflow, Google Compute Engine

B: Google Cloud Storage, Google App Engine, Google Network Load Balancer

C: Google Kubernetes Registry, Google Container Engine, Google HTTP(S) Load Balancer

D: Google Cloud Functions, Google Cloud Pub/Sub, Google Cloud Deployment Manager

Correct Answer:

C

Explanation:

This combination of products directly addresses all of Mountkirk Games' requirements for a continuous delivery pipeline for microservices. Google Kubernetes Engine (GKE, formerly Container Engine) is the ideal platform for deploying and managing containerized microservices, facilitating rapid updates and rollbacks. Google Container Registry (now part of Artifact Registry) serves as the repository for storing the immutable container images used as deployment artifacts. The External HTTP(S) Load Balancer is a global service that provides a single, stable Anycast IP address to distribute user traffic to the closest healthy backend service group across multiple regions (US and Europe), satisfying the networking requirements precisely.

Why Incorrect Options are Wrong:

A: Google Cloud Dataflow is a stream and batch data processing service, which is irrelevant for deploying a continuous delivery pipeline for application services.

B: The Google Network Load Balancer is a regional service. It cannot provide a single frontend IP for services deployed across multiple global regions as required.

D: This option describes a serverless, event-driven architecture. It does not provide the necessary global load balancing or container orchestration for a fleet of frontend services.

References:

1. Google Kubernetes Engine (GKE): "GKE is a good choice for composing your application from multiple, stateless microservices... GKE's rolling update mechanism lets you update your microservices with zero downtime."

Source: Google Cloud Documentation, "Choosing a Compute Option", Section: "Google Kubernetes Engine".

2. Artifact Registry (formerly Container Registry): "Artifact Registry is the evolution of Container Registry. Use Artifact Registry for container image storage and management... Artifacts are immutable, meaning that you cannot change an artifact's contents."

Source: Google Cloud Documentation, "Artifact Registry overview".

3. Cloud Load Balancing: The feature comparison table shows that the External HTTP(S) Load Balancer is "Global" and provides a "Single anycast IP address," while the External Network Load Balancer is "Regional." This makes the HTTP(S) Load Balancer the correct choice for the multi-region, single-IP requirement.

Source: Google Cloud Documentation, "Load Balancer features and types", Section: "Summary of Google Cloud load balancers".

4. Cloud Dataflow: "Cloud Dataflow is a fully managed service for executing Apache Beam pipelines within the Google Cloud ecosystem. Cloud Dataflow is a good choice for large-scale, data-processing tasks..."

Source: Google Cloud Documentation, "Choosing a Compute Option", Section: "Cloud Dataflow".

Question: 4

For this question, refer to the Mountkirk Games case study. Mountkirk Games' gaming servers are not automatically scaling properly. Last month, they rolled out a new feature, which suddenly became very popular. A record number of users are trying to use the service, but many of them are getting 503 errors and very slow response times. What should they investigate first?

A: Verify that the database is online.

B: Verify that the project quota hasn't been exceeded.

C: Verify that the new feature code did not introduce any performance bugs.

D: Verify that the load-testing team is not running their tool against production.

Correct Answer:

B

Explanation:

The core issue described is that the servers are "not automatically scaling properly" despite a surge in traffic. This indicates a problem with the scaling mechanism itself, not just the application load. When an autoscaler attempts to add new virtual machine instances to a managed instance group (MIG), the request can fail if the project has exhausted its quota for a required resource, such as CPUs for a specific region, IP addresses, or total instances. This failure to provision new resources directly explains why the system isn't scaling, leading to the existing servers being overwhelmed and returning 503 (Service Unavailable) errors and slow responses. Therefore, checking resource quotas is the most logical first step.

Why Incorrect Options are Wrong:

A: A database being offline would likely cause application errors, but it would not directly prevent the autoscaling mechanism from attempting to launch new instances in response to high traffic.

C: A performance bug would explain the high load and slow responses, but it doesn't explain why the autoscaler, which is designed to handle such load, is failing to scale out.

D: A load test, similar to a real user surge, would increase load. This should trigger a scale-out event; it does not explain why the scaling mechanism itself is failing.

References:

1. Google Cloud Documentation, "Troubleshooting autoscalers": Under the section "Autoscaler is not scaling out the MIG," a primary cause listed is "Resource quotas." The documentation states, "You might have run out of quota for the resources in your project... If you run out of quota, the autoscaler cannot scale out the MIG." This directly supports investigating quotas as a first step when scaling fails.

2. Google Cloud Documentation, "Resource quotas": This document explains that quotas are limits on Google Cloud resources that a project can use. It clarifies, "When you request a resource, your request is denied if you have insufficient quota." This confirms the mechanism by which an exceeded quota would prevent an autoscaler from creating new VM instances.

Question: 5

For this question, refer to the Mountkirk Games case study Mountkirk Games needs to create a repeatable and configurable mechanism for deploying isolated application environments. Developers and testers can access each other's environments and resources, but they cannot access staging or production resources. The staging environment needs access to some services from production. What should you do to isolate development environments from staging and production?

A: Create a project for development and test and another for staging and production.

B: Create a network for development and test and another for staging and production.

C: Create one subnetwork for development and another for staging and production.

D: Create one project for development, a second for staging and a third for production.

Correct Answer:

D

Explanation:

The most effective and secure method for isolating environments in Google Cloud is by using separate projects. Projects serve as the primary boundary for Identity and Access Management (IAM), billing, and API enablement. Creating distinct projects for development, staging, and production allows for granular IAM policies that prevent developers and testers from accessing staging or production resources. This structure aligns with Google's best practices for resource management and security, providing the strongest possible isolation while still allowing for controlled communication between environments (e.g., staging to production) when necessary.

Why Incorrect Options are Wrong:

A: Combining staging and production into a single project is a poor security practice. It increases the blast radius and the risk of accidental modifications to the production environment.

B: While separate VPC networks provide network-level isolation, they do not isolate IAM permissions. Resources in different networks within the same project still share the same IAM policies.

C: Subnetworks provide the weakest form of isolation. They exist within a single VPC and project, sharing the same network and IAM boundaries, making them unsuitable for separating entire environments.

References:

1. Google Cloud Architecture Framework: Security pillar: In the section "Define your resource hierarchy," the documentation states, "Use projects to group resources that share the same trust boundary... You can also use projects to separate your production environment from your non-production environments." This directly supports using separate projects for isolation.

2. Google Cloud Documentation, "Decide a resource hierarchy for your Google Cloud landing zone": In the section on "Project structure," a recommended approach is "A project for each environment, such as development, test, and production." This explicitly validates the strategy in the correct answer.

3. Google Cloud Documentation, "IAM overview": This document explains that IAM policies are applied at different levels of the resource hierarchy, with the project being a fundamental level. It notes, "A project organizes all your Google Cloud resources... Policies are inherited from parent to child." This highlights why project-level separation is crucial for enforcing distinct access controls for different environments.

Question: 6

The operations manager asks you for a list of recommended practices that she should consider when migrating a J2EE application to the cloud. Which three practices should you recommend? (Choose three.)

A: Port the application code to run on Google App Engine

B: Integrate Cloud Dataflow into the application to capture real-time metrics

C: Instrument the application with a monitoring tool like Stackdriver Debugger

D: Select an automation framework to reliably provision the cloud infrastructure

E: Deploy a continuous integration tool with automated testing in a staging environment

F: Migrate from MySQL to a managed NoSQL database like Google Cloud Datastore or Bigtable

Correct Answer:

C, D, E

Explanation:

The chosen practices represent foundational pillars for a successful and modern cloud migration, aligning with DevOps principles and the Google Cloud Architecture Framework.

Instrumenting the application for monitoring (C) is crucial for observability, allowing operators to understand the application's health and performance in the new cloud environment.

Automating infrastructure provisioning (D) using an Infrastructure as Code (IaC) approach ensures deployments are reliable, repeatable, and auditable.

Deploying a continuous integration (CI) pipeline with automated testing (E) accelerates development cycles and improves application quality and reliability by automating the build, test, and deployment process.

These three practices are universally recommended for migrating and operating applications effectively on any cloud platform.

Why Incorrect Options are Wrong:

A: Porting to App Engine is a specific re-platforming choice, not a universal practice. The application might be better suited for Google Kubernetes Engine or Compute Engine.

B: Cloud Dataflow is a service for large-scale data processing. Using it for application metrics is inappropriate; Cloud Monitoring is the correct tool for this purpose.

F: Migrating from MySQL to a NoSQL database is a significant re-architecture decision, not a general migration best practice. This may not be necessary or beneficial for all applications.

References:

1. Google Cloud Architecture Framework, Operational excellence pillar: This official framework outlines key principles for running workloads on Google Cloud.

Principle: Automate your infrastructure provisioning and management. This section explicitly recommends using tools like Terraform or Cloud Deployment Manager to "reliably provision the cloud infrastructure," directly supporting option D.

Principle: Use a CI/CD approach for all your artifacts. This section advocates for automating builds, tests, and deployments, which directly supports option E.

Principle: Define and implement a comprehensive monitoring and observability strategy. This section recommends using the Google Cloud operations suite (formerly Stackdriver) to gain insights into application performance and health, which directly supports option C.

2. Google Cloud Documentation, DevOps overview, "What is CI/CD?": This document describes the importance of continuous integration and automated testing as a core DevOps practice for improving software delivery speed and reliability, supporting option E.

3. Google Cloud Documentation, Cloud Debugger, "Overview": "Cloud Debugger is a feature of Google Cloud that lets you inspect the state of a running application in real time, without stopping or slowing it down." This describes the instrumentation mentioned in option C as part of a comprehensive observability strategy.

Question: 7

Your company has decided to make a major revision of their API in order to create better experiences for their developers. They need to keep the old version of the API available and deployable, while allowing new customers and testers to try out the new API. They want to keep the same SSL and DNS records in place to serve both APIs. What should they do?

A: Configure a new load balancer for the new version of the API

B: Reconfigure old clients to use a new endpoint for the new API

C: Have the old API forward traffic to the new API based on the path

D: Use separate backend pools for each API path behind the load balancer

Correct Answer:

D

Explanation:

The most precise and architecturally sound solution is to use a single Layer 7 load balancer that directs traffic based on the URL path. By configuring the load balancer's URL map, requests to a path like /v1/ can be routed to a backend service (pool) running the old API, while requests to /v2/ are sent to a different backend service with the new API. This pattern meets all requirements: it uses a single DNS record and SSL certificate, keeps both versions isolated and independently deployable, and provides a seamless way to direct traffic to the appropriate version.

Why Incorrect Options are Wrong:

A: A new load balancer would have a new IP address, violating the requirement to use the same DNS record for both APIs.

B: This places the burden of change on the client applications and does not solve the server-side routing requirement from a single endpoint.

C: Handling routing within the old API application is inefficient, creates an unnecessary dependency, and is not a scalable cloud-native pattern.

References:

1. Google Cloud Documentation, "External HTTP(S) Load Balancing overview": This document explains that the load balancer uses a URL map to direct requests to specific backend services based on the host and path of the incoming URL. This is the core

mechanism described in the correct answer. (See section: "How External HTTP(S) Load Balancing works").

2. Google Cloud Documentation, "URL maps overview": "After a load balancer receives a request, it uses the host and path of the request URL to determine which backend service should receive the request... A path rule specifies the URL paths that are mapped to a particular backend service." This directly supports using path-based routing to separate backend pools.

3. Google Cloud Documentation, "Backend services overview": A backend service defines how Cloud Load Balancing distributes traffic. The scenario requires two distinct backend services (one for each API version), which are then targeted by the URL map rules. (See section: "Backend services").

Question: 8

An application development team believes their current logging tool will not meet their needs for their new cloud-based product. They want a better tool to capture errors and help them analyze their historical log data. You want to help them find a solution that meets their needs. What should you do?

A: Direct them to download and install the Google StackDriver logging agent

B: Send them a list of online resources about logging best practices

C: Help them define their requirements and assess viable logging tools

D: Help them upgrade their current tool to take advantage of any new features

Correct Answer:

C

Explanation:

The most effective and professional first step is to collaborate with the development team to formally define their technical and business requirements for a logging solution. This foundational activity ensures that any proposed tool is evaluated against a clear set of criteria, such as log volume, retention policies, query capabilities, and integration needs. This requirements-driven approach is a core principle of solution architecture, preventing the premature adoption of a tool that may not be the best fit. Once requirements are established, a systematic assessment of viable tools, including Google's Cloud Logging, can be performed.

Why Incorrect Options are Wrong:

A: This option prematurely recommends a specific tool (the Cloud Logging agent) without first verifying if it meets the team's specific, undefined requirements.

B: Sending resources is a passive action that does not directly help the team evaluate and select a specific tool to solve their immediate problem.

D: The team has already expressed a lack of confidence in their current tool; focusing on an upgrade ignores the need for a comprehensive evaluation of alternatives.

References:

1. Google Cloud Architecture Framework: System design. The framework's first principle states, "Before you start to design your architecture, gather the business requirements and

goals for your workload." This directly supports the approach of defining requirements before selecting a solution. (Source: Google Cloud Documentation, "Google Cloud Architecture Framework", "System design", "Design principles", Principle 1).

2. Google Cloud Architecture Framework: Operational Excellence pillar. This pillar emphasizes the need to "Define and implement observability" which includes logging. The process of defining service level objectives (SLOs) and what needs to be logged is a form of requirements gathering that must precede implementation. (Source: Google Cloud Documentation, "Google Cloud Architecture Framework", "Operational excellence", "Define and implement observability").

3. Google Cloud Adoption Framework. This framework outlines a journey that begins with assessing the current state and defining the future state. Helping the team define requirements aligns with this initial assessment phase, ensuring the subsequent technology choices are well-founded. (Source: Google Cloud Documentation, "Google Cloud Adoption Framework", "Your cloud journey").

Question: 9

Your company wants to track whether someone is present in a meeting room reserved for a scheduled meeting. There are 1000 meeting rooms across 5 offices on 3 continents. Each room is equipped with a motion sensor that reports its status every second. The data from the motion detector includes only a sensor ID and several different discrete items of information. Analysts will use this data, together with information about account owners and office locations. Which database type should you use?

A: Flat file

B: NoSQL

C: Relational

D: Blobstore

Correct Answer:

B

Explanation:

The scenario describes a classic Internet of Things (IoT) use case characterized by high-volume, high-velocity data ingestion from a large number of distributed sensors (1000 writes per second). The data is semi-structured ("sensor ID and several different discrete items"). NoSQL databases are specifically designed for these requirements, offering horizontal scalability to handle high write throughput and a flexible schema to accommodate evolving sensor data without rigid structural constraints. A NoSQL database like Google Cloud Bigtable is purpose-built for ingesting and analyzing large-scale time-series and IoT data.

Why Incorrect Options are Wrong:

A: Flat file: This is a data storage format, not a database system. It lacks the indexing, querying capabilities, and scalability required for real-time analysis of 1000 events per second.

C: Relational: Relational databases enforce a strict schema and are not typically optimized for the massive, continuous write throughput characteristic of this IoT workload, which can lead to performance bottlenecks.

D: Blobstore: A blobstore (like Google Cloud Storage) is an object storage system for unstructured data. It is not a database and does not support the low-latency querying of individual records required by analysts.

References:

1. Google Cloud Documentation, "Cloud Bigtable - Overview": Cloud Bigtable is a NoSQL wide-column database. The documentation explicitly states its ideal use cases: "Cloud Bigtable is ideal for applications that need very high throughput and scalability for non-structured key/value data... What it's good for: ... Time series data... Internet of Things (IoT) data, such as energy consumption and temperature readings from a network of sensors." This directly matches the question's scenario.

2. Google Cloud Documentation, "Choosing a storage option": In the section comparing database options, NoSQL databases are recommended for "IoT (time series)" and "Large-scale, high-speed data processing." Relational databases (Cloud SQL) are recommended for "Web frameworks" and "OLTP," which are not the primary use case described.

3. Google Cloud Architecture Center, "Serverless IoT backend": This reference architecture for an IoT platform shows sensor data flowing through Pub/Sub and into a NoSQL database (in this case, Firestore) for storage and subsequent analysis, demonstrating this as a best-practice pattern. The document states, "Firestore is a NoSQL document database that lets you easily store, sync, and query data for your mobile and web apps at global scale."

Question: 10

You need to reduce the number of unplanned rollbacks of erroneous production deployments in your company's web hosting platform. Improvement to the QA/ Test processes accomplished an 80% reduction. Which additional two approaches can you take to further reduce the rollbacks? (Choose two.)

A: Introduce a green-blue deployment model

B: Replace the QA environment with canary releases

C: Fragment the monolithic platform into microservices

D: Reduce the platform's dependency on relational database systems

E: Replace the platform's relational database systems with a NoSQL database

Correct Answer:

A, C

Explanation:

The goal is to further reduce unplanned rollbacks after initial QA improvements.

A blue-green (or blue/green) deployment strategy directly addresses this by maintaining two identical production environments. The new version ("green") can be fully tested in production without affecting users. If an issue is found, traffic is never switched, or if it's switched and an issue arises, it can be instantly routed back to the stable "blue" environment. This makes the rollback process nearly instantaneous and low-risk.

Fragmenting a monolith into microservices reduces the "blast radius" of any single deployment. Each service can be deployed independently. An erroneous deployment only requires rolling back that specific small service, not the entire platform, thus reducing the number and impact of large-scale rollbacks.

Why Incorrect Options are Wrong:

B: Canary releases are a valuable technique but should supplement, not replace, a QA environment. Eliminating QA would likely increase the number of bugs reaching production, counteracting the goal.

D: The type of database (relational vs. non-relational) is not the root cause of deployment errors. The deployment and testing processes are the primary factors, not the database technology itself.

E: Replacing a database technology is a massive architectural undertaking. It does not directly address deployment rollbacks and would introduce significant new risks during the transition period.

References:

1. Blue-Green Deployments (A): Google Cloud Architecture Center, "Application deployment and testing strategies". This document describes the blue-green deployment strategy, stating, "This technique can eliminate downtime during a deployment. It also reduces risk: if the new version has issues, you can immediately roll back to the previous version." This directly supports option A.

2. Microservices (C): Google Cloud Architecture Center, "Migrating a monolith to microservices on Google Cloud". The document highlights a key benefit: "Independent deployments. Microservices introduce the ability to deploy each service independently of the other services... If a problem occurs, you can roll back the change for a single service instead of rolling back the entire application." This directly supports option C.

3. Canary Releases vs. QA (B): The "Application deployment and testing strategies" document positions canary testing as a production-level strategy, distinct from pre-production environments like QIt is a method for testing in production with a subset of users, not a replacement for foundational quality assurance.

Question: 11

You write a Python script to connect to Google BigQuery from a Google Compute Engine virtual machine. The script is printing errors that it cannot connect to BigQuery. What should you do to fix the script?

A: Install the latest BigQuery API client library for Python

B: Run your script on a new virtual machine with the BigQuery access scope enabled

C: Create a new service account with BigQuery access and execute your script with that user

D: Install the bq component for gcloud with the command gcloud components install bq.

Correct Answer:

B

Explanation:

The error described indicates an authentication failure. A Google Compute Engine (GCE) virtual machine uses its attached service account to make authorized API calls to other Google Cloud services. The permissions for this service account are governed by the access scopes configured on the VM instance. If the instance was created without the bigquery or cloud-platform access scope, the Python script, which uses Application Default Credentials (ADC) by default, will fail to authenticate to the BigQuery API. Creating a new VM with the correct access scope enabled grants the necessary permissions to the default service account, allowing the script to connect successfully.

Why Incorrect Options are Wrong:

A: A missing or outdated client library would typically result in an ImportError within the script, not a connection or permission-denied error from the API.

C: While creating a new service account is a valid authentication method, the more direct solution is to correct the permissions of the VM's existing default identity.

D: The bq command-line tool is a separate utility and its installation has no effect on Python client libraries used for programmatic API access.

References:

1. Google Cloud Documentation, Compute Engine, "Service accounts": This document explains how GCE instances use service accounts for authentication. It details access

scopes as the mechanism for granting permissions to the VM. It states, "Access scopes are the legacy method of specifying permissions for an instance... When you set access scopes on an instance, the permissions of the service account are limited by the scopes that you set." This directly supports that a missing scope causes permission failure.

2. Google Cloud Documentation, Authentication, "How Application Default Credentials works": This page describes the credential lookup order for ADFor GCE, it specifies that ADC uses the credentials from the GCE metadata server, which are tied to the service account and access scopes configured for the instance. This confirms why the script's authentication depends on the VM's configuration.

3. Google Cloud Documentation, Compute Engine, "Create and start a VM instance": In the section on "Identity and API access," the documentation outlines how to set access scopes during VM creation, including the specific scope for BigQuery (https://www.googleapis.com/auth/bigquery). This shows that enabling the scope is a required step for granting BigQuery access.

Question: 12

A production database virtual machine on Google Compute Engine has an ext4-formatted persistent disk for data files. The database is about to run out of storage space. How can you remediate the problem with the least amount of downtime?

A: In the Cloud Platform Console, increase the size of the persistent disk and use the resize2fs command in Linux.

B: Shut down the virtual machine, use the Cloud Platform Console to increase the persistent disk size, then restart the virtual machine

C: In the Cloud Platform Console, increase the size of the persistent disk and verify the new space is ready to use with the fdisk command in Linux

D: In the Cloud Platform Console, create a new persistent disk attached to the virtual machine, format and mount it, and configure the database service to move the files to the new disk

E: In the Cloud Platform Console, create a snapshot of the persistent disk restore the snapshot to a new larger disk, unmount the old disk, mount the new disk and restart the database service

Correct Answer:

A

Explanation:

The most efficient method with the least downtime is to perform an online resize. Google Cloud Persistent Disks can be resized while attached to a running virtual machine. After the underlying block device is enlarged, the resize2fs utility can be used on a mounted ext4 filesystem to extend it to occupy the newly available space without unmounting it or rebooting the instance. This combination of features in Google Cloud and Linux provides the fastest remediation with minimal to zero service interruption, directly addressing the question's core requirement.

Why Incorrect Options are Wrong:

B: Shutting down the virtual machine introduces significant, unnecessary downtime, which is explicitly contrary to the question's constraint of minimizing it.

C: The fdisk command is used to manage disk partitions, not to resize the filesystem itself. This option is incomplete and would not make the new space usable.

D: Creating a new disk and migrating data is a complex process that would require stopping the database service, resulting in considerable downtime.

E: The snapshot-and-restore process is a valid migration path but is significantly more complex and time-consuming, causing more downtime than a direct online resize.

References:

1. Google Cloud Documentation, Compute Engine, "Resize a persistent disk": "You can resize your disks at any time, even when they are attached to running VMs. You do not need to stop your VMs or detach your disks to resize them." This source confirms the first part of the correct answer.

2. Google Cloud Documentation, Compute Engine, "Resize a persistent disk", Section: "Resizing a file system and partitions": This section explicitly details the post-resize steps within the OS. For a disk with just a filesystem and no partition table, it states: "If your disk has only a file system on it, with no partition table, run the following command to extend the file system to the full size of the disk: sudo resize2fs /dev/[DEVICEID]". This confirms the second part of the correct answer.

3. Linux man page for resize2fs(8): "The resize2fs program will resize ext2, ext3, or ext4 file systems. ... If the filesystem is mounted, it can be used to expand the size of the mounted filesystem, assuming the kernel and the file system supports on-line resizing." This confirms that resize2fs supports the online operation required for minimal downtime.

Question: 13

Your customer is moving an existing corporate application to Google Cloud Platform from an on-premises data center. The business owners require minimal user disruption. There are strict security team requirements for storing passwords. What authentication strategy should they use?

A: Use G Suite Password Sync to replicate passwords into Google

B: Federate authentication via SAML 2.0 to the existing Identity Provider

C: Provision users in Google using the Google Cloud Directory Sync tool

D: Ask users to set their Google password to match their corporate password

Correct Answer:

B

Explanation:

Federating authentication via SAML 2.0 to the existing on-premises Identity Provider (IdP) is the optimal strategy. This approach allows users to authenticate using their current corporate credentials without creating new ones, ensuring minimal disruption. Most importantly, the authentication process occurs at the on-premises IdP. The IdP then sends a secure SAML assertion to Google Cloud, confirming the user's identity. This means passwords are never transmitted to or stored by Google, which directly satisfies the strict security requirement of keeping the existing, trusted password store as the single source of truth.

Why Incorrect Options are Wrong:

A: Using Password Sync replicates passwords to Google's infrastructure, which explicitly violates the strict security requirement of not storing passwords in a new system.

C: Google Cloud Directory Sync (GCDS) is a tool for provisioning user and group accounts, not for handling the authentication process itself. It synchronizes identities, not credentials.

D: Asking users to manually set matching passwords is a poor security practice, causes significant user disruption, and violates the principle of a single, authoritative password store.

References:

1. Google Cloud Documentation, "Federating Google Cloud with an external identity provider": This document explicitly states, "With federation, your existing identity provider is the single source of truth for your user identities... Passwords and other credentials are not synchronized to Google Cloud." This directly supports option B as the correct method for meeting the stated security and usability requirements.

2. Google Cloud Documentation, "Setting up single sign-on via SAML for a workforce identity pool": This guide details the SAML 2.0-based federation process, noting that it "lets your users sign in to Google Cloud services with their existing enterprise identity provider." This confirms that federation minimizes user disruption by leveraging existing credentials.

3. Google Cloud Documentation, "About Password Sync": This source describes Password Sync as a tool that "synchronizes your users' Active Directory passwords with their Google Accounts." This confirms that option A involves replicating passwords, which is contrary to the security requirement in the scenario.

4. Google Cloud Documentation, "About Google Cloud Directory Sync": This page clarifies the tool's purpose: "Google Cloud Directory Sync (GCDS) lets you synchronize your user data in your Google Account with your Microsoft Active Directory or LDAP server." It focuses on synchronizing user, group, and contact data, not the authentication mechanism itself, making option C an incomplete solution for the authentication strategy.

Question: 14

Your company plans to migrate a multi-petabyte data set to the cloud. The data set must be available 24hrs a day. Your business analysts have experience only with using a SQL interface. How should you store the data to optimize it for ease of analysis?

A: Load data into Google BigQuery

B: Insert data into Google Cloud SQL

C: Put flat files into Google Cloud Storage

D: Stream data into Google Cloud Datastore

Correct Answer:

A

Explanation:

The scenario requires a solution for a multi-petabyte dataset that is highly available and can be analyzed using a SQL interface. Google BigQuery is the most appropriate service as it is a serverless, highly scalable, and cost-effective cloud data warehouse designed specifically for petabyte-scale analytics. It provides a standard SQL interface, which meets the requirement for the business analysts, and is managed for high availability, fulfilling the 24/7 access need. Its architecture is optimized for running complex analytical queries over massive datasets, making it the ideal choice for this use case.

Why Incorrect Options are Wrong:

B: Insert data into Google Cloud SQL: Cloud SQL is a relational database service (OLTP) not designed or optimized for analytics on multi-petabyte datasets. Query performance would be inadequate.

C: Put flat files into Google Cloud Storage: Cloud Storage is an object storage service. While it can store petabytes of data, it does not offer a native SQL interface for analysis.

D: Stream data into Google Cloud Datastore: Cloud Datastore (now Firestore in Datastore mode) is a NoSQL database for transactional workloads, not a data warehouse for large-scale SQL-based analytics.

References:

1. Google Cloud Documentation - What is BigQuery?: "BigQuery is a fully managed enterprise data warehouse that helps you manage and analyze your data with built-in

features like machine learning, geospatial analysis, and business intelligence... BigQuery's serverless architecture lets you use SQL queries to answer your organization's biggest questions with zero infrastructure management." This source confirms BigQuery's purpose for petabyte-scale analytics using SQL.

2. Google Cloud Documentation - Cloud SQL Overview: "Cloud SQL is a fully managed relational database service for MySQL, PostgreSQL, and SQL Server." This highlights its role as a traditional relational database (OLTP), which is distinct from a data warehouse (OLAP) like BigQuery.

3. Google Cloud Documentation - Google Cloud Storage Overview: "Cloud Storage is a service for storing your objects in Google Cloud. An object is an immutable piece of data consisting of a file of any format." This defines it as a storage service, not an analytical engine.

4. Google Cloud Documentation - Firestore in Datastore mode overview: "Use Datastore mode for databases that will be used primarily by applications... It is not a relational database, and it does not support SQL queries." This explicitly states its unsuitability for SQL-based analysis.

Question: 15

A news feed web service has the following code running on Google App Engine. During peak load, users report that they can see news articles they already viewed. What is the most likely cause of this problem?

```
import news
from flask import Flask, redirect, request
from flask.ext.api import status
from google.appengine.api import users

app = Flask(_name_)
sessions = {}

@app.route(“/”)
def homepage():
      user = users.get_current_user()
      if not user:
            return “Invalid login”,
status.HTTP_401_UNAUTHORIZED

      if user not in sessions:
            sessions[user] = {“viewed”: []}

      news_articles = news.get_new_news (user, sessions [user]
[“viewed”])
      sessions [user] [“viewed”] +- [n[“id”] for n
in news_articles]

      return news.render(news_articles)

if _name_ == “_main_”:
      app.run()
```

A: The session variable is local to just a single instance

B: The session variable is being overwritten in Cloud Datastore

C: The URL of the API needs to be modified to prevent caching

D: The HTTP Expires header needs to be set to -1 stop caching

Correct Answer:

A

Explanation:

The Google App Engine standard environment is designed to be stateless and scales horizontally by creating or shutting down instances based on traffic. During peak load, multiple instances will be running. If the session variable is stored in the memory of a single instance, it is not shared with other instances. A user's requests can be routed to different instances. When a subsequent request from a user hits a new instance that does not have the updated lastseen session value, the application logic will fetch older articles, causing the user to see content they have already viewed.

Why Incorrect Options are Wrong:

B: The provided code snippet does not show any interaction with Cloud Datastore for session management, so assuming it's being overwritten there is speculative and not directly supported.

C: Modifying the API's URL is a technique to bypass client-side or CDN caching. The root cause here is server-side state management, not client-side caching.

D: Setting the Expires header is a mechanism to control HTTP caching by browsers and proxies. The problem lies with the application's internal state, not the caching of its responses.

References:

1. Google Cloud Documentation - App Engine - How Instances are Managed: "App Engine standard environment is designed to be stateless... Any request can be routed to any instance... Consequently, you shouldn't rely on the request being sent to the same instance. Relying on in-memory state makes an application non-scalable." This directly supports the conclusion that instance-local state is the cause of the problem during scaling.

2. Google Cloud Blog - Best practices for building scalable and resilient web apps on Google Cloud: In the section "Stateless compute layer," the article advises: "To build a scalable application, you must design it to be stateless. This means that any application instance can handle any user request at any time... Store session state in an external data

store like Memorystore, Redis, or Firestore." This reinforces that session state must not be local to an instance for a scalable application.

Question: 16

You set up an autoscaling instance group to serve web traffic for an upcoming launch. After configuring the instance group as a backend service to an HTTP(S) load balancer, you notice that virtual machine (VM) instances are being terminated and re-launched every minute. The instances do not have a public IP address. You have verified the appropriate web response is coming from each instance using the curl command. You want to ensure the backend is configured correctly. What should you do?

A: Ensure that a firewall rules exists to allow source traffic on HTTP/HTTPS to reach the load balancer.

B: Assign a public IP to each instance and configure a firewall rule to allow the load balancer to reach the instance public IP.

C: Ensure that a firewall rule exists to allow load balancer health checks to reach the instances in the instance group.

D: Create a tag on each instance with the name of the load balancer. Configure a firewall rule with the name of the load balancer as the source and the instance tag as the destination.

Correct Answer:

C

Explanation:

The behavior described—instances being terminated and re-launched cyclically—is a classic symptom of failing load balancer health checks. The managed instance group's autoscaler interprets the failed health checks as the instances being unhealthy and consequently replaces them. The load balancer's health check probes must be able to reach the backend instances over the network. This requires an ingress firewall rule that specifically allows traffic from Google's designated health check IP ranges to the backend instances on the appropriate port. Verifying the application with curl on the instance itself only confirms the application is running, not that it is reachable by the health checker.

Why Incorrect Options are Wrong:

A: This firewall rule applies to client traffic reaching the load balancer's frontend IP, not the health check traffic from the load balancer to the backend instances.

B: Backend instances do not need public IPs to be reached by the load balancer. This is an insecure practice and does not solve the internal connectivity issue for health checks.

D: While using tags on instances for firewall rules is a best practice, the source of the traffic is not the "name of the load balancer" but specific IP ranges used by Google's health check systems.

References:

1. Google Cloud Documentation, "Health checks overview," section "Firewall rules." This document explicitly states: "For health checks to work, you must create ingress allow firewall rules so that traffic from Google Cloud probers can connect to your backends... The following IP address ranges are used by Google Cloud probers: 35.191.0.0/16 and 130.211.0.0/22."

2. Google Cloud Documentation, "Managed instance groups (MIGs)," section "Health checking for MIGs." This page explains the consequence of failed checks: "If a health check determines that an application is not running on a VM, the MIG declares that VM instance as unhealthy... the MIG then recreates that unhealthy instance." This directly corroborates the instance cycling behavior described in the question.

3. Google Cloud Documentation, "Setting up an external HTTP(S) load balancer," section "Create a firewall rule for the backends." This tutorial provides a concrete example: "Create the fw-allow-health-check firewall rule. This is an ingress rule that allows traffic from the Google Cloud health checking systems (130.211.0.0/22 and 35.191.0.0/16)."

Question: 17

To reduce costs, the Director of Engineering has required all developers to move their development infrastructure resources from on-premises virtual machines (VMs) to Google Cloud Platform. These resources go through multiple start/stop events during the day and require state to persist. You have been asked to design the process of running a development environment in Google Cloud while providing cost visibility to the finance department. Which two steps should you take? (Choose two.)

A: Use the - -no-auto-delete flag on all persistent disks and stop the VM

B: Use the - -auto-delete flag on all persistent disks and terminate the VM

C: Apply VM CPU utilization label and include it in the BigQuery billing export

D: Use Google BigQuery billing export and labels to associate cost to groups

E: Store all state into local SSD, snapshot the persistent disks, and terminate the VM

F: Store all state in Google Cloud Storage, snapshot the persistent disks, and terminate the VM

Correct Answer:

A, D

Explanation:

The question presents two distinct requirements: a cost-effective method for managing development VMs with frequent start/stop cycles while preserving state, and a mechanism for cost visibility.

Option A addresses the first requirement. Stopping a Compute Engine VM is the most cost-effective approach for temporary shutdowns, as you are no longer charged for vCPU or memory. Using the --no-auto-delete flag on the persistent disk ensures that the state is preserved when the VM is stopped or even deleted, fulfilling the state persistence requirement.

Option D addresses the second requirement. Applying labels to resources (like VMs and disks) is the standard Google Cloud method for organizing resources and tracking costs. Exporting billing data to BigQuery allows for detailed, query-based analysis of costs, which can be grouped by these labels to provide the necessary visibility to the finance department.

Why Incorrect Options are Wrong:

B: Terminating the VM and using --auto-delete on disks would cause permanent data loss, violating the requirement that state must persist.

C: Labeling based on a dynamic metric like CPU utilization is not a standard or effective method for cost allocation; labels are for static metadata.

E: Local SSD data does not persist when a VM is stopped. The snapshot-and-terminate process is too slow and costly for daily start/stop cycles.

F: The snapshot-and-terminate process is inefficient for this use case. Re-architecting applications to use Cloud Storage is not implied and is overly complex.

References:

1. Stopping an instance and disk persistence: Google Cloud, Compute Engine Documentation, "Stop and start an instance." It states, "When you stop an instance... you are not charged for instance uptime after the instance is stopped. However, you are still charged for any resources that are attached to the instance, such as persistent disks..." This supports stopping VMs for cost savings while retaining disks.

2. Disk auto-deletion behavior: Google Cloud, Compute Engine Documentation, "Add or resize persistent disks." Under the "Deletion rule" section, it explains that the default behavior for boot disks is to be deleted with the instance, but this can be configured. This supports the use of the --no-auto-delete flag to preserve state.

3. Cost tracking with labels: Google Cloud, Billing Documentation, "Creating and managing resource labels." It states, "Labels are key-value pairs that you can attach to Google Cloud resources... Labels are forwarded to the billing system, so you can break down your billed charges by label."

4. Billing export for analysis: Google Cloud, Billing Documentation, "Export Cloud Billing data to BigQuery." The documentation states, "You can export detailed Google Cloud billing data... to a BigQuery dataset that you specify... Exporting billing data to BigQuery enables you to analyze your costs in detail." This confirms that the combination of labels and BigQuery export is the correct approach for cost visibility.

Question: 18

Your company has successfully migrated to the cloud and wants to analyze their data stream to optimize operations. They do not have any existing code for this analysis, so they are exploring all their options. These options include a mix of batch and stream processing, as they are running some hourly jobs and live- processing some data as it comes in. Which technology should they use for this?

A: Google Cloud Dataproc

B: Google Cloud Dataflow

C: Google Container Engine with Bigtable

D: Google Compute Engine with Google BigQuery

Correct Answer:

B

Explanation:

Google Cloud Dataflow is the most appropriate technology for this scenario. It is a fully managed service designed specifically for executing unified stream and batch data processing pipelines. It uses the open-source Apache Beam SDK, which allows developers to write a single data processing pipeline that can be executed in either streaming or batch mode. This directly addresses the requirement to handle both hourly jobs (batch) and live-processing (stream) for a new project with no existing code, providing a single, scalable, and serverless platform for both use cases.

Why Incorrect Options are Wrong:

A: Google Cloud Dataproc: This is a managed service for Apache Hadoop and Spark, best suited for migrating existing big data workloads, not for building new, unified pipelines from scratch.

C: Google Container Engine with Bigtable: This is an infrastructure-centric approach. It requires building and managing a custom processing application on Kubernetes, which is more complex than using a dedicated, managed service like Dataflow.

D: Google Compute Engine with Google BigQuery: This option separates compute from the analytics warehouse. It is not a unified processing service and would require manual orchestration of jobs on VMs for processing.

References:

1. Google Cloud Documentation, "Dataflow overview": "Dataflow is a managed service for executing a wide variety of data processing patterns. The documentation for this service refers to these patterns as pipelines. Dataflow pipelines can be of two types: batch (processing bounded data) and streaming (processing unbounded data)." This directly supports its dual-purpose nature.

2. Google Cloud Documentation, "Choosing a data processing option": In the section comparing services, it states, "Use Dataflow if you want a serverless approach to both stream and batch data processing." It contrasts this with Dataproc: "Use Dataproc if you want to deploy your existing Hadoop or Spark jobs to a managed service."

3. Google Cloud Documentation, "Apache Beam and the Dataflow SDK": "The Apache Beam SDKs provide a unified programming model that lets you develop both batch and streaming pipelines. You write a program with one of the Beam SDKs, and then you choose a runner, such as Dataflow, to execute your pipeline." This confirms the unified model for new development.

Question: 19

Your application needs to process credit card transactions. You want the smallest scope of Payment Card Industry (PCI) compliance without compromising the ability to analyze transactional data and trends relating to which payment methods are used. How should you design your architecture?

A: Create a tokenizer service and store only tokenized data

B: Create separate projects that only process credit card data

C: Create separate subnetworks and isolate the components that process credit card data

D: Streamline the audit discovery phase by labeling all of the virtual machines (VMs) that process PCI data

E: Enable Logging export to Google BigQuery and use ACLs and views to scope the data shared with the auditor

Correct Answer:

A

Explanation:

Tokenization is the most effective architectural strategy to minimize the scope of Payment Card Industry (PCI) compliance. This process replaces sensitive cardholder data (CHD) with a non-sensitive equivalent called a "token." The actual CHD is securely stored offsite by a compliant third-party payment processor. Your application then stores and processes only these tokens. This design removes your primary application infrastructure from the scope of most PCI Data Security Standard (DSS) requirements, as it no longer stores, processes, or transmits raw CHYou can still perform trend analysis using the tokens and associated non-sensitive metadata (e.g., card type, expiration status) without compromising security.

Why Incorrect Options are Wrong:

B: Creating separate projects isolates the Cardholder Data Environment (CDE) but does not minimize its scope; the project handling credit card data would still be fully in-scope for PCI DSS.

C: Isolating components in separate subnetworks is a required network segmentation control for a CDE, but like option B, it contains the scope rather than minimizing it.

D: Labeling VMs is an organizational best practice that aids in auditing but does not architecturally reduce or minimize the compliance scope itself.

E: Managing log access is a specific compliance control (PCI DSS Requirement 10), not a primary architectural design choice for minimizing the overall CDE scope.

References:

1. Google Cloud Whitepaper, "PCI DSS Compliance in Google Cloud": In the section "Reducing your PCI DSS scope," the document states, "One of the most effective ways to reduce the scope of your CDE is to reduce your interaction with cardholder data. Google recommends that you consider using a third-party payment processor that offers a tokenization solution... By using a tokenization service, you can avoid handling raw credit card data and significantly reduce the scope of your PCI DSS assessment." (Page 7).

2. Google Cloud Architecture Center, "Credit card tokenization reference architecture": This reference architecture explicitly details a solution using a third-party payment gateway to tokenize cardholder data before it reaches the merchant's backend on Google Cloud. The overview states, "This architecture helps you limit your PCI DSS compliance scope because your backend systems don't receive sensitive cardholder data."

3. Google Cloud Security Foundations Guide, "PCI-DSS": This guide discusses network segmentation (related to options B and C) as a method to "isolate the CDE from the rest of the organization's network," which is about containing scope, not minimizing it in the most effective way possible. Tokenization (Option A) is presented as the superior method for scope reduction.

Question: 20

Your customer is receiving reports that their recently updated Google App Engine application is taking approximately 30 seconds to load for some of their users. This behavior was not reported before the update. What strategy should you take?

A: Work with your ISP to diagnose the problem

B: Open a support ticket to ask for network capture and flow data to diagnose the problem, then roll back your application

C: Roll back to an earlier known good release initially, then use Stackdriver Trace and Logging to diagnose the problem in a development/test/staging environment

D: Roll back to an earlier known good release, then push the release again at a quieter period to investigate. Then use Stackdriver Trace and Logging to diagnose the problem

Correct Answer:

C

Explanation:

The most effective strategy follows standard incident response best practices. The immediate priority is to mitigate the user impact, which is best achieved by rolling back to the last known-good version of the application. Google App Engine's versioning capabilities are designed for this. Once service is restored, the faulty version should be debugged in a non-production environment (development, test, or staging) to prevent further impact. Cloud Trace is the specific Google Cloud service designed to diagnose latency bottlenecks in applications, making it the ideal tool to investigate the 30-second load time, supplemented by Cloud Logging for detailed event analysis.

Why Incorrect Options are Wrong:

A: The problem is correlated with a software update, making an ISP or network issue highly improbable as the root cause.

B: This approach incorrectly prioritizes diagnostics over immediate mitigation and focuses on network data, which is unlikely to be the source of an application performance issue.

D: Re-deploying a known-faulty version to the production environment, even during a quiet period, is a risky practice that can cause further user impact.

References:

1. App Engine Versions and Rollbacks: Google Cloud documentation on "Splitting Traffic" describes managing different versions and migrating traffic, which is the mechanism for a rollback. It states, "Migrating traffic switches the request routing from one or more versions to a single new version." This supports the "roll back" action.

Source: Google Cloud Documentation, "Splitting traffic", https://cloud.google.com/app-engine/docs/standard/python3/splitting-traffic (The concept is language-agnostic).

2. Cloud Trace for Latency Diagnosis: The official documentation for Cloud Trace states, "Cloud Trace is a distributed tracing system for Google Cloud that helps you understand how long it takes your application to handle incoming requests... It helps you find performance bottlenecks in your applications." This directly supports using Trace to diagnose the 30-second load time.

Source: Google Cloud Documentation, "Cloud Trace overview", https://cloud.google.com/trace/docs/overview.

3. Best Practices for Staging Environments: Google Cloud's architecture framework emphasizes the importance of using separate environments. "Isolate your production environment from your development and test environments... This isolation can limit the impact of security issues and configuration errors." This supports diagnosing the issue in a non-production environment.

Source: Google Cloud Architecture Framework, "System design", https://cloud.google.com/architecture/framework/system-design/design-for-scale-and-performance.

Question: 21

You are managing an application deployed on Cloud Run for Anthos, and you need to define a strategy for deploying new versions of the application. You want to evaluate the new code with a subset of production traffic to decide whether to proceed with the rollout. What should you do?

A: Deploy a new revision to Cloud Run with the new version. Configure traffic percentage between revisions.

B: Deploy a new service to Cloud Run with the new version. Add a Cloud Load Balancing instance in front of both services.

C: In the Google Cloud Console page for Cloud Run, set up continuous deployment using Cloud Build for the development branch. As part of the Cloud Build trigger, configure the substitution variable TRAFFIC_PERCENTAGE with the percentage of traffic you want directed to a new version.

D: In the Google Cloud Console, configure Traffic Director with a new Service that points to the new version of the application on Cloud Run. Configure Traffic Director to send a small percentage of traffic to the new version of the application.

Correct Answer:

A

Explanation:

The most direct and idiomatic method for performing a gradual rollout (canary deployment) in Cloud Run is to leverage its native traffic splitting capabilities. When you deploy a new version of your application's container image to an existing Cloud Run service, it creates a new, immutable revision. Cloud Run then allows you to configure how traffic is distributed between different revisions of that same service. By directing a small percentage of traffic to the new revision, you can evaluate its performance with live users before gradually migrating all traffic to it. This is the intended and most efficient pattern for this use case.

Why Incorrect Options are Wrong:

B: Creating an entirely new service and an external load balancer is an overly complex and non-standard approach for managing versions of a single application within Cloud Run.

C: This option describes the automation of a deployment (CI/CD) rather than the core traffic management strategy itself. The fundamental action is splitting traffic between revisions, which this option only alludes to.

D: Traffic Director is a powerful service mesh control plane, but using it for traffic splitting within a single Cloud Run service is unnecessary complexity, as this is a built-in feature of Cloud Run.

References:

1. Google Cloud Documentation: Cloud Run - "Rollbacks, gradual rollouts, and traffic migration": This document states, "By default, when you deploy a new revision, it replaces the previous revision and starts serving 100% of the traffic... However, you can change this behavior by splitting traffic to different revisions. Splitting traffic allows you to do gradual rollouts (canary deployments)..." This directly supports using revisions and traffic splitting.

2. Google Cloud Documentation: Cloud Run - "Splitting traffic": This guide provides the specific steps for the strategy described in the correct answer. It details how to "configure your service to split traffic to two or more revisions" and notes that "This is useful for testing a new revision with a small percentage of users (canary testing)..."

3. Google Cloud Documentation: Cloud Run - "About revisions": This page clarifies the core concept: "Each time you deploy to a service, a new revision is created. A revision consists of a specific container image, along with settings such as environment variables, memory limits, or concurrency value. Revisions are immutable." This establishes the foundation for why traffic is split between revisions.

Question: 22

A recent audit revealed that a new network was created in your GCP project. In this network, a GCE instance has an SSH port open to the world. You want to discover this network's origin.
What should you do?

A: Search for Create VM entry in the Stackdriver alerting console

B: Navigate to the Activity page in the Home section. Set category to Data Access and search for Create VM entry

C: In the Logging section of the console, specify GCE Network as the logging section. Search for the Create Insert entry

D: Connect to the GCE instance using project SSH keys. Identify previous logins in system logs, and match these with the project owners list

Correct Answer:

C

Explanation:

The most direct and reliable method to determine the origin of a created Google Cloud resource is by querying the Admin Activity audit logs. Cloud Logging is the designated service for this purpose. By navigating to the Logs Explorer ("Logging section"), you can construct a query to filter for the specific resource type (gcenetwork) and the API method that creates it (gce.networks.insert). The resulting log entries will contain the identity of the principal (user or service account) that performed the action, the timestamp, and the source IP address, definitively identifying the network's origin.

Why Incorrect Options are Wrong:

A: The Cloud Monitoring alerting console is used to configure and view alerts, not for conducting historical investigations of audit logs.

B: The Activity page provides a high-level summary. More importantly, resource creation is captured in "Admin Activity" logs, not "Data Access" logs, making the filter incorrect.

D: This investigates who has accessed the GCE instance via SSH, which is unrelated to who created the underlying network and the instance itself.

References:

1. Google Cloud Documentation, "Audit logs overview": This document explains that "Admin Activity audit logs contain log entries for API calls or other actions that modify the configuration or metadata of resources." The creation of a new network falls under this category. (Source: Google Cloud, Cloud Logging Documentation, "Overview of Cloud Audit Logs", https://cloud.google.com/logging/docs/audit).

2. Google Cloud Documentation, "Compute Engine audit logging information": This page lists the specific API calls that generate audit logs for Compute Engine. The method for creating a network is v1.compute.networks.insert, which is logged as an Admin Activity. This confirms that searching for a "Create" or "Insert" entry for a GCE Network is the correct procedure. (Source: Google Cloud, Cloud Logging Documentation, "Audited services", https://cloud.google.com/logging/docs/audit/gcp/compute#auditedoperations).

3. Google Cloud Documentation, "View logs by using the Logs Explorer": This guide details how to use the interface to build queries. It shows how to filter by resource type, such as "GCE Network," to narrow down search results, which is the action described in the correct answer. (Source: Google Cloud, Cloud Logging Documentation, "Query and view logs", https://cloud.google.com/logging/docs/view/logs-explorer-interface).

Question: 23

Your company has just recently activated Cloud Identity to manage users. The Google Cloud Organization has been configured as well. The security team needs to secure projects that will be part of the Organization. They want to prohibit IAM users outside the domain from gaining permissions from now on. What should they do?

A: Configure an organization policy to restrict identities by domain.

B: Configure an organization policy to block creation of service accounts.

C: Configure Cloud Scheduler to trigger a Cloud Function every hour that removes all users that don't belong to the Cloud Identity domain from all projects.

D: Create a technical user (e.g., [email protected]), and give it the project owner role at root organization level. Write a bash script that: ¢ Lists all the IAM rules of all projects within the organization. ¢ Deletes all users that do not belong to the company domain. Create a Compute Engine instance in a project within the Organization and configure gcloud to be executed with technical user credentials. Configure a cron job that executes the bash script every hour.

Correct Answer:

A

Explanation:

The most effective and direct way to prohibit IAM users from outside a specific domain from being granted permissions is to use the Organization Policy Service. The iam.allowedPolicyMemberDomains constraint is designed specifically for this purpose. By applying this policy at the organization level, you can specify one or more Cloud Identity or Google Workspace customer IDs. Once enforced, any attempt to add a principal to an IAM policy from a domain not on the allowed list will be denied, providing a proactive and preventative security control across all projects.

Why Incorrect Options are Wrong:

B: Blocking service account creation is unrelated to restricting external human users and would break legitimate automation and application functionality.

C: This is a reactive, not preventative, solution. It would remove unauthorized users after they have already been granted access, leaving a window of vulnerability.

D: This is a more complex, manually scripted version of option It is also reactive and introduces unnecessary operational overhead compared to a native policy service.

References:

1. Google Cloud Documentation, "Restricting identities by domain": This document explicitly describes the solution. It states, "You can use the Restrict domain sharing organization policy to limit the set of identities that are allowed to be used in Identity and Access Management (IAM) policies. This organization policy can be used to require that IAM policy members belong to a specific domain." It details the use of the iam.allowedPolicyMemberDomains list constraint.

2. Google Cloud Documentation, "Introduction to the Organization Policy Service": This document explains the purpose of the service: "The Organization Policy Service gives you centralized and programmatic control over your organization's cloud resources. As the organization policy administrator, you can define an organization policy, which is a set of constraints..." This confirms that using an organization policy is the correct approach for centralized enforcement.

3. Google Cloud Documentation, "Organization policy constraints": This page lists available constraints. The entry for iam.allowedPolicyMemberDomains confirms its function: "Defines the set of domains whose members can be added to IAM policies. By default, users from any domain can be added to IAM policies." This directly supports option A as the correct mechanism.

Question: 24

JencoMart has built a version of their application on Google Cloud Platform that serves traffic to Asia. You want to measure success against their business and technical goals. Which metrics should you track?

A: Error rates for requests from Asia

B: Latency difference between US and Asia

C: Total visits, error rates, and latency from Asia

D: Total visits and average latency for users from Asia

E: The number of character sets present in the database

Correct Answer:

C

Explanation:

To effectively measure success against both business and technical goals, a combination of metrics is necessary. "Total visits" is a key business metric that quantifies user traffic and engagement. "Error rates" and "latency" are fundamental technical metrics, often referred to as Service Level Indicators (SLIs), that measure the application's reliability and performance, respectively. This combination provides a comprehensive view of the application's health and its ability to meet user expectations in the specified region, directly aligning with the "Four Golden Signals" of monitoring advocated in Google's Site Reliability Engineering (SRE) principles.

Why Incorrect Options are Wrong:

A: This option is incomplete. It only tracks a single technical metric (reliability) and ignores business goals and performance (latency).

B: This is a comparative metric. The primary concern is the absolute performance for Asian users, not its difference from another region.

D: This option is incomplete as it omits error rates, which are a critical indicator of application reliability and user-facing problems.

E: This is an internal data-level detail, not a metric for measuring the success of application traffic or user experience.

References:

1. Google Cloud, Site Reliability Engineering Book, Chapter 6: Monitoring Distributed Systems. This chapter introduces the "Four Golden Signals" of monitoring: Latency, Traffic, Errors, and Saturation. Option C directly corresponds to the first three signals, which are considered essential for monitoring user-facing systems. "Total visits" represents Traffic, "error rates" represents Errors, and "latency" represents Latency.

2. Google Cloud Architecture Framework: Reliability pillar. The documentation for this pillar emphasizes defining and measuring Service Level Indicators (SLIs) to track reliability. It states, "For a user-facing system, good SLIs are often focused on availability, latency, and quality." Option C's metrics (error rates for availability, latency for performance) are prime examples of these SLIs.

3. Google Cloud Documentation, Cloud Monitoring: Metrics, time series, and resources. This documentation explains that key metrics to monitor for services include request counts (traffic/visits), error counts (errors), and latency. These are the fundamental metrics for understanding service health and performance.

Question: 25

You need to upgrade the EHR connection to comply with their requirements. The new connection design must support business-critical needs and meet the same network and security policy requirements. What should you do?

A: Add a new Dedicated Interconnect connection.

B: Upgrade the bandwidth on the Dedicated Interconnect connection to 100 G.

C: Add three new Cloud VPN connections.

D: Add a new Carrier Peering connection.

Correct Answer:

A

Explanation:

The requirement to support "business-critical needs" for an Electronic Health Records (EHR) system implies a strong need for high availability and redundancy, not just increased bandwidth. Adding a new, separate Dedicated Interconnect connection is the standard Google Cloud architecture for achieving a 99.99% uptime SLThis creates a fully redundant path for network traffic, ensuring that the failure of a single connection does not disrupt service to the critical EHR system. This approach directly addresses the need for a robust, highly available connection while allowing for the consistent application of existing network and security policies across both connections.

Why Incorrect Options are Wrong:

B: Upgrading bandwidth on a single connection does not improve its availability. It remains a single point of failure, which is unacceptable for a business-critical system.

C: Cloud VPN connections rely on the public internet and typically offer lower and less consistent performance than Dedicated Interconnect, making them an unsuitable upgrade for this use case.

D: While Carrier Peering is a valid connectivity option, adding a second Dedicated Interconnect is the most direct and consistent way to create redundancy for an existing Dedicated Interconnect setup.

References:

1. Google Cloud Documentation, Cloud Interconnect, Redundancy and SLA: In the section "Topology for production-level applications," the documentation explicitly states: "To achieve a 99.99% SLA for Dedicated Interconnect, you must provision at least two connections in one or more metropolitan areas." This directly supports adding a new connection for business-critical needs. (Source: Google Cloud, "Redundancy and SLA for Dedicated Interconnect", https://cloud.google.com/network-connectivity/docs/interconnect/concepts/redundancy#dedicated-redundancy)

2. Google Cloud Documentation, Cloud Interconnect, Product overview: This document positions Dedicated Interconnect as the solution for "Your most critical, data-intensive applications or workloads" due to its ability to provide a "direct, private connection" with high bandwidth and low latency, reinforcing its suitability for a critical EHR system over other options like Cloud VPN. (Source: Google Cloud, "Cloud Interconnect overview", https://cloud.google.com/network-connectivity/docs/interconnect/concepts/overview)

3. Google Cloud Documentation, Choosing a Network Connectivity product: This guide compares different connectivity options. It highlights that for "High availability and low latency," Dedicated Interconnect is the recommended choice, whereas Cloud VPN is suited for "Low-volume data connections." This contrast justifies eliminating Cloud VPN as an upgrade path for a critical system. (Source: Google Cloud, "Choose a product", https://cloud.google.com/network-connectivity/docs/choose-product)

Question: 26

For this question, refer to the EHR Healthcare case study. You are a developer on the EHR customer portal team. Your team recently migrated the customer portal application to Google Cloud. The load has increased on the application servers, and now the application is logging many timeout errors. You recently incorporated Pub/Sub into the application architecture, and the application is not logging any Pub/Sub publishing errors. You want to improve publishing latency. What should you do?

A: Increase the Pub/Sub Total Timeout retry value.

B: Move from a Pub/Sub subscriber pull model to a push model.

C: Turn off Pub/Sub message batching.

D: Create a backup Pub/Sub message queue.

Correct Answer:

C

Explanation:

The Pub/Sub client libraries, by default, use message batching to optimize for cost and throughput. This means the client waits to collect multiple messages before sending them in a single API request. While efficient, this process introduces latency for each individual message publish call. The application's timeout errors, despite no Pub/Sub publishing errors, suggest that application threads are waiting too long for the batched publish operation to complete. Turning off message batching (or tuning it to send messages immediately) will reduce this publishing latency, as each message will be sent in its own request without delay, which should alleviate the application's timeout issues.

Why Incorrect Options are Wrong:

A: Increasing the retry timeout would make the application wait even longer before a publish operation fails, likely exacerbating the application-level timeout errors, not improving latency.

B: The subscriber model (pull vs. push) concerns how messages are received from a subscription, which has no direct impact on the latency of publishing messages to a topic.

D: Creating a backup queue (topic) is a design pattern for availability or disaster recovery; it does not address or improve the performance or latency of the primary publishing path.

References:

1. Google Cloud Documentation, "Pub/Sub - Publish messages to a topic": In the section on "Batch settings," the documentation states, "When you publish a message, the client library can batch multiple messages together in a single request. Message batching can reduce the cost of publishing messages by reducing the number of API requests. However, it can also add latency to the publish request." This directly confirms that batching adds latency, and therefore disabling it would reduce latency.

2. Google Cloud Documentation, "Pub/Sub - Client libraries explained": The "Publishing messages" section explains the asynchronous nature of the publish call and the role of batching. It notes that for low-latency requirements, batching parameters should be configured to send messages immediately. For example, setting the element count threshold to 1 effectively disables batching.

Question: 27

The TerramEarth development team wants to create an API to meet the company's business requirements. You want the development team to focus their development effort on business value versus creating a custom framework. Which method should they use?

A: Use Google App Engine with Google Cloud Endpoints. Focus on an API for dealers and partners

B: Use Google App Engine with a JAX-RS Jersey Java-based framework. Focus on an API for the public

C: Use Google App Engine with the Swagger (Open API Specification) framework. Focus on an API for the public

D: Use Google Container Engine with a Django Python container. Focus on an API for the public

E: Use Google Container Engine with a Tomcat container with the Swagger (Open API Specification) framework. Focus on an API for dealers and partners

Correct Answer:

A

Explanation:

The primary goal is to enable developers to focus on business value rather than building a custom API framework. Google Cloud Endpoints is a distributed API management system that provides a complete API gateway solution, handling authentication, monitoring, logging, and quotas. This allows developers to concentrate on writing the application's business logic. Pairing Cloud Endpoints with Google App Engine, a fully managed serverless platform, further reduces operational overhead by abstracting away the underlying infrastructure. This combination is the most direct and efficient method to meet the stated requirements.

Why Incorrect Options are Wrong:

B: JAX-RS is a specification for building RESTful web services within the application code; it does not provide the external API management features (e.g., authentication, quotas) that Cloud Endpoints does.

C: The OpenAPI Specification (formerly Swagger) is a standard for defining and documenting APIs, not a management framework. It is used by services like Cloud Endpoints but is not a substitute for them.

D: Google Kubernetes Engine (GKE, formerly Container Engine) requires more operational management for clusters and containers compared to the serverless App Engine, conflicting with the goal of minimizing non-business-logic effort.

E: This option combines the higher operational overhead of GKE with the incorrect premise that the OpenAPI Specification is a management framework, making it the least suitable choice.

References:

1. Google Cloud Documentation, "About Cloud Endpoints": "Cloud Endpoints is an API management system that helps you secure, monitor, analyze, and set quotas on your APIs... By handling API management, Endpoints frees you to focus on your business logic." This directly supports the choice of Cloud Endpoints to reduce framework development.

2. Google Cloud Documentation, "App Engine overview": "Google App Engine is a fully managed, serverless platform for developing and hosting web applications at scale... Google manages the infrastructure concerns so you can focus on your code." This confirms App Engine as the choice for minimizing infrastructure overhead.

3. Google Cloud Documentation, "Choosing a compute option": This documentation typically positions App Engine as the platform-as-a-service (PaaS) offering with the highest level of management, ideal for scenarios where developer velocity and focus on application code are prioritized over infrastructure control, which is the case here. In contrast, GKE offers more control at the cost of increased management responsibility.

4. Google Cloud Documentation, "Cloud Endpoints for OpenAPI overview": "You describe the surface of your API in an OpenAPI specification... Cloud Endpoints uses the configuration file to create an Endpoints service." This clarifies that OpenAPI is a specification used to configure Cloud Endpoints, not a standalone management framework.

Question: 28

For this question, refer to the TerramEarth case study. A new architecture that writes all incoming data to BigQuery has been introduced. You notice that the data is dirty, and want to ensure data quality on an automated daily basis while managing cost. What should you do?

A: Set up a streaming Cloud Dataflow job, receiving data by the ingestion process. Clean the data in a Cloud Dataflow pipeline.

B: Create a Cloud Function that reads data from BigQuery and cleans it. Trigger the Cloud Function from a Compute Engine instance.

C: Create a SQL statement on the data in BigQuery, and save it as a view. Run the view daily, and save the result to a new table.

D: Use Cloud Dataprep and configure the BigQuery tables as the source. Schedule a daily job to clean the data.

Correct Answer:

D

Explanation:

Cloud Dataprep is an intelligent, serverless data service specifically designed for visually exploring, cleaning, and preparing data for analysis. It integrates directly with BigQuery as a source. Users can build data transformation "recipes" through an intuitive UI and then schedule jobs to run these recipes on a recurring basis, such as daily. This directly addresses the requirements to clean dirty data from BigQuery in an automated, daily fashion. As a managed service that runs Dataflow jobs on the backend, it provides a cost-effective solution without requiring users to manage the underlying infrastructure.

Why Incorrect Options are Wrong:

A: A streaming Dataflow job is incorrect because the data is already at rest in BigQuery. A batch job would be needed, but Dataprep provides a higher-level, more user-friendly abstraction for this specific task.

B: Using a Compute Engine instance solely to trigger a Cloud Function is an inefficient and costly architectural pattern. Cloud Scheduler is the appropriate tool for triggering scheduled serverless workloads.

C: While using a BigQuery view and scheduled queries is a valid technical approach, Cloud Dataprep is the more specialized and appropriate tool for data cleaning, offering a visual interface that simplifies the process.

References:

1. Cloud Dataprep Documentation: "Cloud Dataprep by Trifacta is an intelligent data service for visually exploring, cleaning, and preparing structured and unstructured data for analysis, reporting, and machine learning." It also states, "After you have defined the steps for data preparation, you can operationalize your recipe by running a job against your sample or source file."

Source: Google Cloud, "Dataprep documentation overview".

2. Scheduling Dataprep Jobs: "You can create schedules to run your jobs at specified times and frequencies... You can schedule a job to run once or on a repeating basis (every hour, day, week, or month)."

Source: Google Cloud, "Overview of scheduling".

3. BigQuery Integration: Dataprep can directly use BigQuery tables as a data source and write the cleaned results back to a new BigQuery table.

Source: Google Cloud, "BigQuery connection".

Question: 29

For this question, refer to the TerramEarth case study. You are migrating a Linux-based application from your private data center to Google Cloud. The TerramEarth security team sent you several recent Linux vulnerabilities published by Common Vulnerabilities and Exposures (CVE). You need assistance in understanding how these vulnerabilities could impact your migration. What should you do? (Choose two.)

A: Open a support case regarding the CVE and chat with the support engineer.

B: Read the CVEs from the Google Cloud Status Dashboard to understand the impact.

C: Read the CVEs from the Google Cloud Platform Security Bulletins to understand the impact.

D: Post a question regarding the CVE in Stack Overflow to get an explanation.

E: Post a question regarding the CVE in a Google Cloud discussion group to get an explanation.

Correct Answer:

A, C

Explanation:

To understand the impact of specific CVEs on a Google Cloud migration, the most appropriate actions are to consult official Google channels. The Google Cloud Platform Security Bulletins are the primary, authoritative source for public disclosures on how vulnerabilities affect Google Cloud services. For assistance specific to your project's configuration and migration plan, opening a support case allows you to engage directly with Google Cloud support engineers who can provide tailored, official guidance. These two methods ensure you receive accurate and reliable information directly from the vendor.

Why Incorrect Options are Wrong:

B: The Google Cloud Status Dashboard is for monitoring the availability and status of Google Cloud services, not for publishing security vulnerability information.

D: Stack Overflow is a community support forum and is not an official or verified source for Google Cloud security impact analysis.

E: Google Cloud discussion groups are community-based and do not serve as an official channel for security disclosures or authoritative advice.

References:

1. Google Cloud Security Bulletins: "Google Cloud publishes security bulletins for our customers to inform them about security vulnerabilities that may affect Google Cloud services." This is the designated location for CVE impact information.

Source: Google Cloud Documentation, "Security bulletins". (https://cloud.google.com/support/bulletins)

2. Google Cloud Support: "Get help from a Google expert... Get answers to your technical questions about Google Cloud..." Opening a support case is the official mechanism for direct technical assistance.

Source: Google Cloud Documentation, "Google Cloud Customer Care". (https://cloud.google.com/support)

3. Community Support vs. Official Support: Google documentation distinguishes between official support channels (like opening a case) and community resources (like Stack Overflow and Google Groups), which are not intended for official security advisories.

Source: Google Cloud Documentation, "Community support". (https://cloud.google.com/support/docs/community)

4. Google Cloud Status Dashboard: The dashboard's purpose is explicitly stated as providing "status information on the services that are part of Google Cloud." This confirms it is for service health, not security bulletins.

Source: Google Cloud Documentation, "Google Cloud Status Dashboard". (https://cloud.google.com/service-health/docs/dashboard-overview)

Question: 30

You are creating an App Engine application that uses Cloud Datastore as its persistence layer. You need to retrieve several root entities for which you have the identifiers. You want to minimize the overhead in operations performed by Cloud Datastore. What should you do?

A: Create the Key object for each Entity and run a batch get operation

B: Create the Key object for each Entity and run multiple get operations, one operation for each entity

C: Use the identifiers to create a query filter and run a batch query operation

D: Use the identifiers to create a query filter and run multiple query operations, one operation for each entity

Correct Answer:

A

Explanation:

The most efficient method to retrieve multiple Cloud Datastore entities when their identifiers are known is to use a batch get operation. This approach involves constructing the full Key for each entity and then fetching them all in a single API call. This single round trip to the database significantly minimizes latency and operational overhead compared to issuing multiple individual requests. Queries are inherently less efficient than direct key lookups because they involve the query planner and potentially index scans, even when filtering by key.

Why Incorrect Options are Wrong:

B: Running multiple get operations results in multiple network round trips to the database, which increases latency and cost, directly contradicting the goal of minimizing overhead.

C: A query is more computationally expensive and has higher latency than a direct get by key. Using a query when a direct lookup is possible is inefficient.

D: This is the least efficient option, as it combines the higher overhead of queries with the high latency of performing multiple, separate operations.

References:

1. Google Cloud Documentation, "Retrieving an entity": "To look up multiple entities from Datastore in a single request, use the lookup method with a list of Key objects. A batch lookup is more efficient than multiple individual get calls." This directly supports using a batch get operation (A) over multiple get operations (B).

2. Google Cloud Documentation, "Datastore Queries": This documentation describes how queries work, involving filters, sorts, and index scans. It states, "A Datastore query retrieves entities that meet a specified set of conditions." This process is inherently more complex than a direct key lookup, making options C and D less efficient than A.

3. Google Cloud Documentation, "Entities, Properties, and Keys": "Each entity in Datastore has a key that uniquely identifies it." This principle is why a get operation using a key is a direct, highly efficient lookup, as opposed to a query which must search for entities matching criteria.