CERT EMPIRE

GOOGLE Professional Cloud Architect Exam Questions

Total Questions: 250+
Demo Questions: 35
Version: Updated for 2025

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

Question: 1

Your company wants to try out the cloud with low risk. They want to archive approximately 100 TB of their log data to the cloud and test the analytics features available to them there, while also retaining that data as a long-term disaster recovery backup. Which two steps should they take? Choose 2 answers

A. Load logs into Google BigQuery.

B. Load logs into Google Cloud SQL.

C. Import logs into Google Stackdriver.

D. Insert logs into Google Cloud Bigtable.

E. Upload log files into Google Cloud Storage.

Answer:

A, E

Explanation:

This scenario requires a two-part solution: a cost-effective, durable location for long-term archival and a powerful platform for large-scale analytics.
1. Google Cloud Storage (E) is the ideal initial destination. It is a highly scalable and durable object store. For long-term disaster recovery and archiving, the Coldline or Archive storage classes offer extremely low storage costs, fitting the "low risk" and "archive" requirements for 100 TB of data.

2. Google BigQuery (A) is Google's serverless, petabyte-scale data warehouse designed for fast SQL queries over large datasets. It directly addresses the requirement to "test the analytics features." Data can be easily loaded from Cloud Storage into BigQuery for high-performance analysis, or queried in place using external tables.
This combination provides a standard, cost-effective, and low-risk pattern for archiving and analyzing large data volumes on Google Cloud.

Why Incorrect Options are Wrong:

B. Load logs into Google Cloud SQL.
Cloud SQL is a managed relational database (OLTP) service not designed or priced for storing and analyzing 100 TB of log data.
C. Import logs into Google Stackdriver.
Now called Cloud Logging, this service is for real-time operational log management and monitoring, not for long-term, large-scale archival and historical analysis.
D. Insert logs into Google Cloud Bigtable.
Bigtable is a NoSQL wide-column store optimized for high-throughput, low-latency workloads,

making it more expensive and less suitable for long-term archival and ad-hoc SQL analytics compared to the chosen combination.

References:

1. Google Cloud Documentation - Cloud Storage Classes: "Archive Storage is our lowest-cost, highly durable storage service for data archiving, online backup, and disaster recovery... you can store massive amounts of data that you access less than once a year at a very low cost." This supports using Cloud Storage for the archival and DR requirement.
Source: Google Cloud Documentation, "Storage Classes", Section on "Archive Storage".
2. Google Cloud Documentation - Introduction to 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." This confirms BigQuery as the tool for the analytics requirement.
Source: Google Cloud Documentation, "What is BigQuery?".
3. Google Cloud Documentation - Loading data from Cloud Storage: "You can load data from Cloud Storage into BigQuery... Google Cloud Storage is a versatile and cost-effective place to stage data before loading it into BigQuery." This document explicitly describes the recommended pattern of using Cloud Storage as a source for BigQuery, validating the combination of answers A and E.
Source: Google Cloud Documentation, "Introduction to loading data from Cloud Storage".

4. Google Cloud Documentation - Choosing a storage option: This guide compares different storage services. It positions Cloud SQL for "Web frameworks, OLTP workloads" and BigQuery for "Data warehousing, BI, analytics," clearly distinguishing their use cases and showing why Cloud SQL is inappropriate for this scenario.
Source: Google Cloud Documentation, "Cloud Storage Products".

Question: 2

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 rule 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.

Answer:

C

Explanation:

The behavior described-instances being terminated and re-launched every minute-is a classic symptom of failing load balancer health checks. The autoscaler is configured to maintain a certain number of healthy instances. When the load balancer's health check probes cannot reach the backend instances, it reports them as unhealthy. The autoscaler then terminates these "unhealthy" instances and creates new ones to replace them, leading to a continuous cycle. Since the application is confirmed to be working locally on each instance, the problem lies in the network path between the Google Cloud health checkers and the instances. A firewall rule is required to explicitly allow ingress traffic from Google's health check IP ranges to the backend instances.

Why Incorrect Options are Wrong:

A. Firewall rules apply to VM instances, not the managed load balancer service. The load balancer is designed to receive public traffic by default.
B. Backend instances behind a load balancer should not have public IPs for security best practices. Communication from the load balancer to backends occurs over Google's internal network.

D. A load balancer's name cannot be used as a source in a firewall rule. Sources are defined by IP ranges, network tags, or service accounts.

References:

1. Google Cloud Documentation, Health checks overview, "Firewall rules": "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 IP address ranges to allow are 35.191.0.0/16 and 130.211.0.0/22." This directly supports the need for a firewall rule to allow health check traffic.
2. Google Cloud Documentation, External HTTP(S) Load Balancing overview, "Health checks": "Each backend service must have a health check associated with it. The health check probes the backends to determine if they can receive new connections or requests... The health status of backends determines their eligibility to receive new requests." This explains the role of health checks in determining instance availability.
3. Google Cloud Documentation, Autoscaling groups of instances, "Health checking": "If you are using load balancing with your managed instance group, the autoscaler might rely on the load balancer's health check to determine the health of instances... If an instance fails the application-based health check, the autoscaler considers the instance unhealthy and might recreate it." This document explicitly links failing health checks to the autoscaler's action of recreating instances.

Question: 3

You want to optimize the performance of an accurate, real-time, weather-charting application. The data comes from 50,000 sensors sending 10 readings a second, in the format of a timestamp and sensor reading. Where should you store the data?

A. Google BigQuery

B. Google Cloud SQL

C. Google Cloud Bigtable

D. Google Cloud Storage

Answer:

C

Explanation:

The scenario describes a classic time-series use case with a very high write throughput (50,000 sensors 10 readings/sec = 500,000 writes/sec). The application requires low latency for both writing data and reading it for real-time charting. Google Cloud Bigtable is a NoSQL, wide-column database specifically designed for large-scale, low-latency, and high-throughput workloads. It excels at ingesting and serving time-series data, such as that from IoT sensors, making it the optimal choice for this application's performance requirements.

Why Incorrect Options are Wrong:

A. Google BigQuery: This is an enterprise data warehouse optimized for large-scale analytical queries (OLAP), not for high-throughput transactional writes (OLTP) with low latency. It is not suitable for real-time ingestion at this rate.
B. Google Cloud SQL: This is a managed relational database. While excellent for transactional workloads, it cannot scale to handle the 500,000 writes per second required by the application while maintaining low latency.
D. Google Cloud Storage: This is a scalable object store, not a database. It is designed for unstructured data (blobs) and is not suitable for the low-latency, indexed queries of individual records needed for a real-time charting application.

References:

1. Google Cloud Documentation, "About Cloud Bigtable": Under the section "What it's good for," the documentation explicitly lists "Time series data" and "Internet of Things (IoT) data" as ideal workloads. It states, "Cloud Bigtable is ideal for applications that need very high throughput and scalability for non-structured key/value data... It supports high read and write throughput at low latency."
2. Google Cloud Architecture Center, "Time series on Google Cloud: A reference architecture":

This document outlines architectures for time-series data. In the section "Choosing a storage product," it recommends Cloud Bigtable for use cases requiring "low-latency reads and high-throughput writes," which directly matches the question's requirements.
3. Google Cloud Documentation, "Choosing a storage option": This guide compares storage services. For Cloud Bigtable, it lists common use cases as "IoT, finance, ad tech" and workload types as "Large analytical and operational workloads" and "Heavy read/write," confirming its suitability over other options for this scenario.

Question: 4

The database administration team has asked you to help them improve the performance of their new database server running on Google Compute Engine. The database is for importing and normalizing their performance statistics and is built with MySQL running on Debian Linux. They have an n1-standard-8 virtual machine with 80 GB of SSD persistent disk. What should they change to get better performance from this system?

A. Increase the virtual machine's memory to 64 GB.

B. Create a new virtual machine running PostgreSQL.

C. Dynamically resize the SSD persistent disk to 500 GB.

D. Migrate their performance metrics warehouse to BigQuery.

E. Modify all of their batch jobs to use bulk inserts into the database.

Answer:

C

Explanation:

The performance of Google Cloud SSD Persistent Disks (PD-SSD) is directly determined by their provisioned size. A larger disk receives a higher baseline of Input/Output Operations Per Second (IOPS) and throughput. The existing 80 GB disk has relatively low performance limits, which is a common bottleneck for write-intensive database workloads like importing and normalizing data. By increasing the disk size to 500 GB, the IOPS and throughput allocated to the virtual machine's storage will increase significantly, directly addressing the most probable cause of poor performance. This resizing can be done dynamically without detaching the disk, making it an efficient solution.

Why Incorrect Options are Wrong:

A. Increasing memory helps with caching reads, but a data import workload is primarily write-heavy, making disk I/O the more likely bottleneck.
B. Migrating to a different database engine is a complex project and does not address the underlying infrastructure performance limitation, which would likely affect PostgreSQL as well.
D. Migrating to BigQuery is a major architectural shift to a different service, not a performance improvement for the existing Compute Engine-based database system.
E. While a good practice, application-level changes like bulk inserts won't overcome a fundamental I/O performance bottleneck at the underlying disk infrastructure level.

References:

1. Google Cloud Documentation, "Configure disks to meet performance requirements": This document explicitly states, "Persistent disk performance scales with the size of the disk... For SSD persistent disks, the IOPS and throughput limits grow as you increase the size of the disk." The performance table shows an 80 GB PD-SSD has 2,400 read/write IOPS, while a 500 GB disk has 15,000 read/write IOPS.
Source Location: In the section "Performance by persistent disk size".
2. Google Cloud Documentation, "Resize a persistent disk": This document confirms that you can increase the size of a persistent disk while it is attached to a running VM. "You can resize your zonal persistent disks at any time, regardless of whether they are attached to a running virtual machine (VM) instance."
Source Location: In the "Before you begin" section.
3. Google Cloud Documentation, "N1 machine series": This page details the specifications for predefined machine types. The n1-standard-8 machine type has 8 vCPUs and 30 GB of memory, confirming the initial system configuration.
Source Location: In the table under the "N1 standard machine types" section.

Question: 5

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.

Answer:

A

Explanation:

Tokenization is a security process that replaces sensitive cardholder data (CHD), such as the Primary Account Number (PAN), with a non-sensitive equivalent called a "token." By implementing a tokenizer service, the primary application avoids directly processing or storing raw CHD. This dramatically reduces the Payment Card Industry Data Security Standard (PCI DSS) compliance scope to only the isolated tokenizer component. The application can then safely store and use these tokens for transaction analysis, tracking customer behavior, and analyzing payment method trends without exposing sensitive data, thus achieving the smallest possible compliance footprint while meeting business requirements.

Why Incorrect Options are Wrong:

B. This isolates the PCI environment but doesn't reduce its scope. The systems within the separate projects would still be fully in-scope as they process raw credit card data.
C. This is a network segmentation control, which is required by PCI DSS to isolate the Cardholder Data Environment (CDE), but it does not reduce the number of systems within that CDE.
D. Labeling is an administrative aid for identifying in-scope resources during an audit. It does not architecturally reduce the compliance scope itself.
E. This addresses logging and auditing requirements (PCI DSS Requirement 10) but does not reduce the scope of the systems that must be audited in the first place.

References:

1. Google Cloud Whitepaper, "PCI Data Security Standard Compliance in Google Cloud": In the section "Reducing PCI DSS scope" (Page 6), the document states, "One of the most effective ways to reduce the scope of your CDE is to use a third-party payment processor... Another way to reduce scope is to use tokenization." This directly validates that tokenization is a primary strategy for minimizing PCI scope.
2. PCI Security Standards Council, "Information Supplement: PCI DSS Tokenization Guidelines": Section 3.1, "Tokenization and PCI DSS Scope," explains that when tokenization is properly implemented and an organization no longer stores, processes, or transmits cardholder data, the systems handling only tokens may be removed from PCI DSS scope. This confirms that tokenization is the most effective method presented for scope reduction.
3. Google Cloud Architecture Center, "Architecture for PCI DSS compliance on Google Cloud": The "Tokenization proxy" section describes a reference architecture and states, "This architecture can help you reduce your PCI DSS scope because your backend systems don't store or process sensitive CHD." This provides a concrete architectural pattern supporting the chosen answer.

Question: 6

You have been asked to select the storage system for the click-data of your company's large portfolio of websites. This data is streamed in from a custom website analytics package at a typical rate of 6,000 clicks per minute, with bursts of up to 8,500 clicks per second. It must been stored for future analysis by your data science and user experience teams. Which storage infrastructure should you choose?

A. Google Cloud SQL

B. Google Cloud Bigtable

C. Google Cloud Storage

D. Google cloud Datastore

Answer:

B

Explanation:

Google Cloud Bigtable is the most appropriate choice for this scenario. It is a fully managed, wide-column NoSQL database designed specifically for ingesting and analyzing very large datasets with high throughput and low latency. The requirement to handle a high-velocity stream, with bursts up to 8,500 writes per second, aligns perfectly with Bigtable's core strengths. Clickstream data is a classic example of time-series data, for which Bigtable's data model and row key design are exceptionally well-suited. It integrates seamlessly with Google's data analysis ecosystem, including BigQuery and Dataproc, enabling the future analysis required by the data science teams.

Why Incorrect Options are Wrong:

A. Google Cloud SQL: This is a relational database (OLTP) and is not designed for the high-velocity, semi-structured ingestion of clickstream data. It would face significant performance and scaling challenges at 8,500 writes per second.
C. Google Cloud Storage: While Cloud Storage is a cost-effective data lake foundation, it is an object store, not a database. It cannot directly ingest a stream of 8,500 individual clicks per second; it requires an intermediate service like Dataflow to batch events into files, adding complexity. Bigtable is purpose-built for ingesting such high-velocity event streams directly.
D. Google Cloud Datastore: This is a document database optimized for transactional application workloads (e.g., user profiles, shopping carts). It is not designed or cost-effective for high-throughput ingestion and large-scale analytical scans of time-series data.

References:

1. Official Google Cloud Documentation - Bigtable Use Cases: "Bigtable is ideal for applications that need to handle large amounts of data with very high throughput... Common use cases for Bigtable include... Time series data... This includes... data about user behavior on a website." This directly validates the use of Bigtable for the clickstream data described in the question.
Source: Google Cloud Documentation, "About Cloud Bigtable", Section: "What it's good for".
2. Official Google Cloud Documentation - Choosing a Storage Option: In the decision tree for choosing a storage option, workloads characterized by "Large-scale structured data (terabytes or more)" and "Analytics" with a need for "Low-latency read/write access" point directly to Cloud Bigtable. The burst rate of 8,500 clicks/sec implies a high-throughput, low-latency write requirement.
Source: Google Cloud Documentation, "Choose a storage option", Section: "Summary of Google Cloud storage options".
3. Google Cloud Architecture Center - Streaming IoT data to Google Cloud: This reference architecture, while for IoT, describes an identical data pattern (high-volume, time-series event stream). The recommended architecture for real-time processing and storage is Pub/Sub - Dataflow - Bigtable. This demonstrates that for high-velocity streams intended for analysis, Bigtable is the designated storage system.
Source: Google Cloud Architecture Center, "Streaming IoT data to Google Cloud", Document ID: streaming-iot-data-google-cloud.

4. Peer-Reviewed Publication - "Google's Bigtable: A Distributed Storage System for Structured Data": This foundational paper describes the design goals of Bigtable, which include managing massive amounts of data (petabytes) across thousands of commodity servers and handling high-throughput workloads. This academic source underpins why Bigtable is technically suited for the scale mentioned in the question.
Source: Chang, F., Dean, J., Ghemawat, S., et al. (2008). Bigtable: A Distributed Storage System for Structured Data. ACM Transactions on Computer Systems, 26(2), Article 4. (OSDI '06). DOI: https://doi.org/10.1145/1365815.1365816

Question: 7

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.

Answer:

C

Explanation:

The most effective strategy follows standard incident response best practices. The immediate priority is to mitigate user impact. Since the issue began after a recent update, rolling back to the previous, stable version is the fastest way to restore service. Once the production environment is stable, the problematic new version can be deployed to a non-production environment (development, test, or staging) for root cause analysis. Cloud Trace is the ideal tool for diagnosing latency bottlenecks within an application, and Cloud Logging provides the necessary application-level logs to identify errors or anomalous behavior. This approach resolves the user-facing issue quickly while enabling a safe and thorough investigation.

Why Incorrect Options are Wrong:

A. The problem is correlated with a code update, making an application issue far more likely than an ISP problem. Investigating the ISP first is inefficient.
B. This option incorrectly prioritizes a slow diagnosis (opening a support ticket) over immediate mitigation (rolling back), prolonging the negative user experience.
D. Re-deploying a known-bad release to production, even during a quiet period, is risky and unnecessarily impacts users. Investigation should occur in a non-production environment.

References:

1. App Engine Versioning and Rollbacks: Google Cloud's official documentation on App Engine traffic splitting explains how to manage multiple versions and route traffic. A rollback is achieved by migrating 100% of traffic back to a previous, known-good version.
Source: Google Cloud Documentation, "Splitting traffic". Section: "Migrating traffic from one version to another".
2. Using Cloud Trace for Latency: Cloud Trace is designed to analyze latency in applications. It collects latency data to help developers understand how requests propagate through their application and identify performance bottlenecks.
Source: Google Cloud Documentation, "Cloud Trace Overview". Section: "What is Cloud Trace?".
3. Incident Response Principles: The Google SRE handbook, a foundational text for cloud operations, emphasizes restoring service as the top priority during an outage. Quick, simple, and safe actions like a rollback are preferred over complex diagnostics on a live system.
Source: "Site Reliability Engineering: How Google Runs Production Systems", O'Reilly, 2016. Chapter 12, "Emergency Response", Section: "Have a simple, fast, and safe way to stop the bleeding."
4. Debugging in Non-Production Environments: Best practices dictate that debugging and testing should be performed in an environment that mirrors production but does not serve live traffic, to prevent impacting users.
Source: Google Cloud Architecture Framework, "System design". Section: "Use multiple environments".

Question: 8

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

Answer:

B

Explanation:

Google Cloud Dataflow is the optimal choice as it is a fully managed, serverless service designed for unified stream and batch data processing. It utilizes the Apache Beam SDK, which provides a single programming model to build pipelines that can run in either streaming (live-processing) or batch (hourly jobs) mode. This directly addresses the company's need for a mixed-processing solution without requiring them to manage underlying infrastructure or write separate code for batch and stream jobs. Since they have no existing code, adopting the unified Beam model on Dataflow is the most efficient and scalable approach.

Why Incorrect Options are Wrong:

A. Google Cloud Dataproc is a managed service for running open-source frameworks like Apache Spark and Hadoop. It is best suited for migrating existing workloads, not for building new, cloud-native unified pipelines.
C. Google Container Engine with Bigtable provides container orchestration and a NoSQL database. This is an infrastructure-level solution that would require extensive custom development to build a data processing system.
D. Google Compute Engine with Google BigQuery combines IaaS with a data warehouse. While BigQuery excels at large-scale batch analytics and supports streaming ingestion, it is not a unified processing engine for arbitrary stream and batch logic.

References:

1. Google Cloud Documentation, "Dataflow overview": "Dataflow is a managed service for executing a wide variety of data processing patterns. The documentation on this site shows you how to deploy your batch and streaming data processing pipelines using Dataflow... Dataflow is based on the open source project Apache Beam." This source confirms Dataflow's dual capability for batch and stream processing.
2. Google Cloud Documentation, "Choose a data processing option": In the comparison table, Dataflow is recommended for "Batch and stream processing" with a "Unified development model." In contrast, Dataproc is recommended for "Batch and stream processing with open source software (OSS) like Apache Spark," highlighting its role in managing specific frameworks rather than providing a native unified model.
3. Google Cloud Documentation, "Apache Beam and the Dataflow SDK": "The Apache Beam SDKs provide a unified programming model that lets you implement and run data processing jobs in either batch or streaming mode." This directly supports the core reason for choosing Dataflow-the unified model that satisfies the question's requirements.

Question: 9

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.

Answer:

B

Explanation:

Federating authentication via SAML 2.0 to the existing on-premises Identity Provider (IdP) is the optimal strategy. This approach establishes a trust relationship where Google Cloud delegates the authentication process to the customer's IdP. Users are redirected to their familiar corporate login page, ensuring minimal disruption through a single sign-on (SSO) experience. Most importantly, user passwords are never transmitted to or stored by Google. The credentials remain exclusively within the on-premises environment, managed by the existing IdP, thereby satisfying the strict security requirements for password storage.

Why Incorrect Options are Wrong:

A. This replicates password hashes to Google's systems, which directly violates the strict security requirement to keep passwords stored only in the existing corporate system.
C. Google Cloud Directory Sync (GCDS) is a tool for provisioning users and groups, not for authenticating them. It would need to be paired with another solution for authentication.
D. This manual approach is insecure, unmanageable, and causes significant user disruption, failing to meet either of the core requirements.

References:

1. Google Cloud Architecture Framework, Security, privacy, and compliance design pillar, Identity and access management: In the section "Centralize identity management," the framework recommends: "To let your employees use their existing corporate credentials to authenticate to Google Cloud, you can federate your existing identity provider with Cloud Identity." This directly supports the federation approach for leveraging existing corporate identities.
2. Google Cloud Documentation, Cloud Identity, "Choosing an authentication method": This document compares authentication methods. For federation, it states: "Your external IdP remains the single source of truth for identity... Passwords and password policies remain under your

control." This confirms that federation meets the strict security requirement.
3. Google Cloud Documentation, Cloud Identity, "About Google Cloud Directory Sync": The overview clearly states, "You use GCDS to synchronize your Google user data with your Microsoft Active Directory or LDAP server. GCDS doesn't synchronize passwords." This clarifies that GCDS is for provisioning, not authentication, making option C an incomplete solution.
4. Google Workspace Admin Help, "About G Suite Password Sync": The documentation for this tool (now Google Workspace Password Sync) explains that it "synchronizes users' Microsoft Active Directory passwords with their... Cloud Identity passwords." This confirms that option A involves replicating passwords, which is what the security team in the scenario wants to avoid.

Question: 10

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

Answer:

B

Explanation:

The scenario describes a classic Internet of Things (IoT) use case characterized by high-velocity (1000 writes/second) and high-volume data ingestion from globally distributed sensors. NoSQL databases are purpose-built for these requirements. They offer horizontal scalability to handle massive write throughput and flexible data models (e.g., key-value, wide-column) that are ideal for simple, discrete sensor data. A wide-column NoSQL database like Cloud Bigtable is particularly well-suited for storing and querying this type of time-series data efficiently at scale, making it the most appropriate choice.

Why Incorrect Options are Wrong:

A. Flat file: Inefficient for querying and managing a high-velocity stream of individual data points; better suited for batch data, not real-time ingestion and analysis.
C. Relational: Can struggle with high-velocity write ingestion at this scale without significant cost and complexity. The rigid schema is less ideal for simple sensor data.
D. Blobstore: Designed for unstructured binary objects (blobs), not for storing and querying structured or semi-structured transactional data like individual sensor readings.

References:

1. Google Cloud Documentation, "Choosing a storage option": In the comparison table under "Which database should I use?", NoSQL databases like Cloud Bigtable are recommended for "IoT, time series, monitoring, and ad tech" use cases. This directly aligns with the sensor data scenario described in the question. The documentation highlights its strength in handling "heavy

reads and writes."
2. Google Cloud Documentation, "Bigtable Documentation Overview": This document explicitly states, "Bigtable is ideal for applications that need very high throughput and scalability for key/value data... Use cases include IoT, user analytics, and financial data analysis." The scenario of 1000 sensors reporting every second is a high-throughput IoT use case.
3. Google Cloud Architecture Center, "Architecture for connecting IoT devices to Google Cloud": This official reference architecture shows a common pattern where IoT device data is streamed through Pub/Sub and ingested into Cloud Bigtable for "storage, and as a source for stream processing and analytics." This validates the use of a NoSQL database as the standard solution for this problem.

Question: 11

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 2 answers

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.

Answer:

A, C

Explanation:

The goal is to further reduce production rollbacks beyond what QA/Test improvements have already achieved. This requires implementing strategies that either de-risk the deployment process itself or change the application architecture to limit the impact of a faulty deployment.
A blue-green deployment model directly addresses this by maintaining two identical production environments. The new version is deployed to the inactive ("green") environment for final testing. Once verified, traffic is switched over. If issues arise, traffic can be switched back to the original ("blue") environment almost instantly, making the rollback process controlled and minimally disruptive.
Fragmenting a monolith into microservices reduces the "blast radius" of any single deployment. Each service can be deployed and rolled back independently. An error in one service will not necessitate a rollback of the entire platform, thus reducing the number of large-scale, unplanned rollbacks.

Why Incorrect Options are Wrong:

B. Canary releases are a valid strategy, but this option suggests replacing the QA environment. Canary testing is a production strategy that complements a robust QA process, it does not replace it.
D. Reducing dependency on relational databases is an architectural decision not directly tied to deployment safety. Rollbacks are caused by erroneous code or configuration, not the database type itself.
E. Replacing a relational database with NoSQL is a specific technology change. It does not inherently reduce deployment errors and could introduce new complexities, potentially increasing

rollback risk.

References:

1. Blue-Green Deployments: Google Cloud Architecture Center, "Application deployment and testing strategies," discusses Blue/Green deployments as a method to "reduce the risk of introducing a bug into production" by enabling a rapid rollback capability. (Source: Google Cloud, "Application deployment and testing strategies," Section: Blue/Green deployments).
2. Microservices for Deployment Safety: Google Cloud Architecture Center, "Migrating a monolith to microservices," explains a key benefit: "Services can be deployed independently... If a problem occurs in a service, you only need to roll back that service's deployment, rather than the entire application's deployment." This directly supports the goal of reducing platform-wide rollbacks. (Source: Google Cloud, "Migrating a monolith to microservices on Google Kubernetes Engine," Section: Benefits of a microservices architecture).
3. Canary Releases vs. QA: The Google Site Reliability Engineering book clarifies that canarying is a form of production testing for changes that have already passed other tests. It states, "Once a binary has passed all of its automated tests... we begin a series of progressive rollouts to the production fleet... The first stage of any production deployment is a canary." This shows canarying follows, and does not replace, prior testing. (Source: Beyer, B., et al. (2016). Site Reliability Engineering. O'Reilly. Chapter 8, "Release Engineering," Section: "Rollout Process").

Question: 12

An application development team believes their current logging tool will not meet their needs for their new cloud-based product. They want a bettor tool to capture errors and help them analyze their historical log dat a. 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.

Answer:

C

Explanation:

As a Cloud Architect, the primary responsibility is to translate business and technical needs into a well-designed solution. The development team has expressed a need but has not defined the specific requirements for a "better tool." Jumping directly to a solution like installing the Cloud Logging agent (A) or upgrading the current tool (D) is premature. The most effective and professional first step is to engage with the team to formally define their requirements. This includes understanding specific pain points with the current tool, desired analysis capabilities, retention policies, and budget constraints. Once these requirements are clear, a proper evaluation of viable tools, including Google Cloud's operations suite and third-party options, can be conducted to find the best fit.

Why Incorrect Options are Wrong:

A. Direct them to download and install the Google StackDriver logging agent.
This is a solution-first approach. It presumes Cloud Logging is the best fit without first understanding the team's specific functional, performance, or cost requirements.
B. Send them a list of online resources about logging best practices.
This is passive and unhelpful. The architect's role is to actively guide the team to a solution, not to assign them a research project.
D. Help them upgrade their current tool to take advantage of any new features.
This option prematurely focuses on one potential solution. The team's core dissatisfaction might be with the tool's fundamental architecture, which an upgrade may not fix.

References:

1. Google Cloud Architecture Framework - System design pillar: The framework emphasizes that a key principle of system design is to "Understand your requirements." It states, "Before you design your architecture, gather the business requirements and goals for your product." This directly supports option C, which prioritizes defining requirements before assessing tools.
Source: Google Cloud Documentation, "System design: The foundations for your architecture," Section: "Understand your requirements."
2. Google Cloud Adoption Framework: This framework outlines a structured journey to the cloud. The initial phases involve learning, assessing, and planning, which align with the process of defining requirements before implementing a specific technology. The architect's role is to facilitate this process.
Source: Google Cloud Whitepaper, "The Google Cloud Adoption Framework," Section: "Your cloud maturity," describing the "Tactical" vs. "Strategic" phases. A strategic approach involves planning and requirements definition.
3. Carnegie Mellon University, Software Engineering Institute, "Requirements Engineering": Academic principles of software and systems engineering universally establish requirements definition as the foundational step. "Requirements engineering is the process of defining, documenting, and maintaining requirements... It is a crucial part of the system design process, as a project will likely fail if the requirements are not correct." This principle applies directly to selecting a critical tool like a logging solution.
Source: SEI Digital Library, "Requirements Engineering."

Question: 13

A news teed 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 to stop caching.

Answer:

A

Explanation:

The application is running on Google App Engine, which automatically scales by creating new server instances during peak load. The code uses a Flask session object to store the list of seenarticles. By default, this session data is often stored in the local memory of the specific instance handling the request. When App Engine scales out, a user's subsequent requests can be routed to different instances. A new instance will not have the session data from the previous instance, causing it to re-initialize an empty seenarticles list and serve articles the user has already viewed on a different instance. This explains why the issue occurs specifically during peak load when scaling is active.

Why Incorrect Options are Wrong:

B. The code reads from Datastore; it does not write or overwrite session data there. The problem lies with the ephemeral nature of the session state, not the article persistence layer.
C. Modifying the API's URL is a cache-busting technique. The issue is a server-side state management failure across instances, not client-side or intermediary caching.
D. Setting HTTP Expires headers controls browser or proxy caching. This will not solve the underlying problem of the server application losing the user's state between requests.

References:

1. Google Cloud Documentation, App Engine, "How Instances are Managed": In the "Instance state" section, the documentation states, "When the App Engine scheduler creates a new instance, the instance does not have knowledge of the state of other instances." This directly supports the reasoning that session data stored locally on one instance is not available to others. (Source: Google Cloud, App Engine Standard Environment Documentation, "How Instances are Managed").
2. Google Cloud Documentation, App Engine, "Using Sessions" (Python 3): This guide explicitly warns about this scenario: "Because App Engine creates and destroys instances automatically, you can't rely on in-memory session data. An app that receives a request might not be the same app that receives the next request from the same user... To use sessions, you must have a place to store the session data. App Engine provides a session implementation that uses Memorystore for Redis, Firestore, or Cloud Storage as the backend for session data." This confirms that the default local session is the problem and a centralized store is the solution. (Source: Google Cloud, App Engine, Python 3 runtime, "Using Sessions").
3. Google Cloud Architecture Center, "The 12-factor app on Google Cloud": Factor VI, "Processes," states that applications should be executed as one or more stateless processes. It explains, "Any data that needs to persist must be stored in a stateful backing service... For

example, avoid using sticky sessions, which is a feature available in many load balancers that directs requests from a given user to the same server. Sticky sessions can mask problems in your application's session-handling logic." Storing session data in instance memory violates this principle. (Source: Google Cloud Architecture Center, "Modern application architecture").

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.

Answer:

A

Explanation:

Google BigQuery is a serverless, fully-managed enterprise data warehouse designed specifically for large-scale data analytics. It can seamlessly handle multi-petabyte datasets and is optimized for running complex analytical queries with high performance. Crucially, it provides a standard SQL interface (ANSI:2011 compliant), which allows business analysts to leverage their existing skills without needing to learn new query languages or tools. Its serverless nature also ensures high availability (24/7) without any infrastructure management overhead, making it the ideal choice for this scenario.

Why Incorrect Options are Wrong:

B. Insert data into Google Cloud SQL.
Cloud SQL is a managed relational database service (OLTP) and is not designed or cost-effective for petabyte-scale analytical workloads (OLAP). Its storage capacity is limited and insufficient for this use case.
C. Put flat files into Google Cloud Storage.
Google Cloud Storage is an object storage service. While it can store petabytes of data, it does not offer a native SQL interface for analysis, failing to meet the analysts' requirements.
D. Stream data into Google Cloud Datastore.
Cloud Datastore (now Firestore in Datastore mode) is a NoSQL document database optimized for transactional application workloads, not for large-scale analytics. It does not use a standard SQL interface.

References:

1. Google Cloud Documentation, "What is BigQuery?": "BigQuery is a fully managed enterprise data warehouse that helps you manage and analyze your data... 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 as an analytics data warehouse with a SQL interface.
2. Google Cloud Documentation, "Introduction to BigQuery architecture": "BigQuery is Google's fully managed, petabyte scale, low cost analytics data warehouse." This reference explicitly states its capability to handle petabyte-scale data.
3. Google Cloud Documentation, "Cloud SQL quotas and limits": This page details the maximum storage capacity for Cloud SQL instances (e.g., 64 TB for MySQL and PostgreSQL), which is significantly less than the multi-petabyte scale required by the question.
4. Google Cloud Documentation, "Firestore in Datastore mode overview": "It is a NoSQL document database built for automatic scaling, high performance, and ease of application development... Datastore mode is ideal for databases that will be used primarily by applications." This highlights its unsuitability for the described analytics use case.
5. Carnegie Mellon University, School of Computer Science, Course 15-319/619, "Cloud Computing", Lecture 18: Data Analytics & GCP: The course materials differentiate between services like Cloud SQL (for OLTP) and BigQuery (for OLAP), stating, "BigQuery: Interactive analysis of petabyte-scale datasets... Uses a columnar storage format... SQL-like query language." This academic source validates the choice of BigQuery for large-scale SQL-based analysis over other GCP database options.

Question: 15

Your customer is moving their corporate applications to Google Cloud Platform. The security team wants detailed visibility of all projects in the organization. You provision the Google Cloud Resource Manager and set up yourself as the org admin. What Google Cloud Identity and Access Management (Cloud IAM) roles should you give to the security team'?

A. Org viewer, project owner

B. Org viewer, project viewer

C. Org admin, project browser

D. Project owner, network admin

Answer:

B

Explanation:

To achieve "detailed visibility of all projects," the security team needs two types of read-only access granted at the organization level. First, the Organization Viewer role (roles/resourcemanager.organizationViewer) allows them to see the organization's resource hierarchy, including all folders and projects. Second, the Project Viewer role (roles/viewer), when granted at the organization level, is inherited by all child projects. This provides the necessary read-only access to view the resources within every project. This combination perfectly fulfills the requirement while adhering to the principle of least privilege by not granting any modification permissions.

Why Incorrect Options are Wrong:

A. The Project Owner role grants full control over projects, including modifying and deleting resources, which is excessive for a visibility requirement and violates the principle of least privilege.
C. The Org Admin role is highly privileged, granting full control over the organization's policies and hierarchy. The Project Browser role is insufficient as it only allows listing projects, not viewing their internal resources.
D. The Project Owner role is excessively permissive. Furthermore, granting roles at the project level would not provide the required organization-wide visibility from a single policy assignment.

References:

1. Google Cloud Documentation, "Access control for organizations using IAM", Resource Manager roles: The Organization Viewer role (roles/resourcemanager.organizationViewer) is described as providing "Read access to the organization and all its resources." This covers the need to see all projects.

Source: Google Cloud Documentation, IAM, "Access control for organizations using IAM".
2. Google Cloud Documentation, "Understanding roles", Basic roles: The Viewer role (roles/viewer) is defined as granting "Permissions for read-only actions that do not affect state, such as viewing (but not modifying) existing resources or data."
Source: Google Cloud Documentation, IAM, "Understanding roles", Section: "Basic roles".
3. Google Cloud Documentation, "Resource hierarchy overview", IAM policy inheritance: "When you set an allow policy at a level of the resource hierarchy, the permission is inherited by the resources below it. For example, if you set an allow policy for a user at the organization level, the user has that permission for all folders and projects in the organization." This confirms that granting the Project Viewer role at the organization level provides visibility into all projects.
Source: Google Cloud Documentation, Resource Manager, "Resource hierarchy overview", Section: "IAM policy inheritance".
4. Google Cloud Documentation, "Understanding roles", Predefined roles: The Browser role (roles/browser) "doesn't include permissions to view resources in the project," confirming it is insufficient for "detailed visibility." The Owner role (roles/owner) "includes permissions to read and write all of a project's resources," confirming it is excessive.
Source: Google Cloud Documentation, IAM, "Understanding roles", Sections: "Basic roles" and "General roles".

Question: 16

Your company places a high value on being responsive and meeting customer needs quickly. Their primary business objectives are release speed and agility. You want to reduce the chance of security errors being accidentally introduced. Which two actions can you take? Choose 2 answers

A. Ensure every code check-in is peer reviewed by a security SME.

B. Use source code security analyzers as part of the CI/CD pipeline.

C. Ensure you have stubs to unit test all interfaces between components.

D. Enable code signing and a trusted binary repository integrated with your CI/CD pipeline.

E. Run a vulnerability security scanner as part of your continuous-integration /continuous-delivery (CI/CD) pipeline.

Answer:

B, E

Explanation:

The core business objectives are release speed and agility. To reduce the accidental introduction of security errors without compromising these objectives, automated security checks must be integrated into the development lifecycle. This approach is often called "shifting left." Using source code security analyzers (Static Application Security Testing - SAST) and vulnerability scanners (Software Composition Analysis - SCA) as part of the CI/CD pipeline automates the detection of security flaws in both custom-written code and third-party dependencies. These automated checks provide rapid feedback to developers, allowing them to fix issues early without creating manual bottlenecks, thus supporting both security and agility.

Why Incorrect Options are Wrong:

A. A manual peer review by a security Subject Matter Expert (SME) for every check-in would create a significant bottleneck, directly contradicting the primary business objectives of release speed and agility.
C. Using stubs for unit testing is a practice for ensuring functional correctness and component isolation, not for directly identifying security vulnerabilities in the code.
D. Code signing and trusted repositories (like Google Cloud's Binary Authorization) are crucial for supply-chain security. They ensure artifact integrity at deployment time but do not prevent security errors from being introduced into the source code during development.

References:

1. Google Cloud Documentation - Shifting left on security: In the "Recommendations" section, it explicitly states, "Integrate security tools into your CI/CD pipeline. Use tools for static application security testing (SAST), software composition analysis (SCA), and dynamic application security testing (DAST) to automatically scan your code and applications for vulnerabilities." This directly supports options B and E as best practices for integrating automated security into a fast-paced development cycle.
Source: Google Cloud, Security best practices center, "Shifting left on security".
2. Google Cloud Documentation - DevSecOps on Google Cloud: This document outlines security practices within a CI/CD context. Under the "Secure" section, it details different testing types. It describes Static application security testing (SAST) as tools that "analyze static source code for security vulnerabilities" (supporting option B) and Software composition analysis (SCA) as tools that "can scan for vulnerabilities in third-party libraries and frameworks" (supporting option E).
Source: Google Cloud, Architecture Center, "DevSecOps on Google Cloud", Section: "Secure".
3. Google Cloud Documentation - Binary Authorization overview: This document explains that Binary Authorization focuses on deploy-time security policy enforcement to ensure only trusted container images are deployed. This confirms that its role is to protect the release process, not to find vulnerabilities in the source code itself, which is the focus of the question. This supports the reasoning for why option D is incorrect for this specific scenario.
Source: Google Cloud, Documentation, "Binary Authorization overview".

Question: 17

You are helping the QA team to roll out a new load-testing tool to test the scalability of your primary cloud services that run on Google Compute Engine with Cloud Bigtable. Which three requirements should they include? Choose 3 answers

A. Ensure that the load tests validate the performance of Cloud Bigtable.

B. Create a separate Google Cloud project to use for the load-testing environment.

C. Schedule the load-testing tool to regularly run against the production environment.

D. Ensure all third-party systems your services use are capable of handling high load.

E. Instrument the production services to record every transaction for replay by the load-testing tool.

F. Instrument the load-testing tool and the target services with detailed logging and metrics collection.

Answer:

A, B, F

Explanation:

A robust load-testing strategy for a cloud-native application requires a dedicated, isolated environment, a comprehensive test scope, and detailed observability. Creating a separate Google Cloud project (B) provides essential isolation for resources, billing, and IAM, preventing any impact on the production environment. The load test must validate the performance of all critical components, including the Cloud Bigtable database (A), as it is integral to the overall service's scalability. Finally, instrumenting both the load-testing tool and the target services with detailed logging and metrics (F) is fundamental. Without this data, it is impossible to measure performance, identify bottlenecks, or validate the test's success.

Why Incorrect Options are Wrong:

C: Running a new load-testing tool against a production environment is extremely risky and can cause service disruptions or outages for real users.
D: While considering third-party systems is important, the primary goal is to test your services. Often, third-party dependencies are mocked or stubbed to isolate the test.
E: Recording every production transaction is often impractical due to performance overhead and storage costs. Generating synthetic load or using a sample of traffic is a more common practice.

References:

1. Separate Project (B): Google Cloud's "Security foundations guide" recommends using a resource hierarchy with separate projects for different environments (e.g., development, test, production) to provide isolation and manage access controls and quotas independently. This is a foundational best practice. (Source: Google Cloud Documentation, Security foundations guide, "Resource hierarchy" section).
2. Instrumentation and Metrics (F): The Google SRE book emphasizes that a service's reliability and performance can only be understood through measurement. Chapter 6, "Monitoring Distributed Systems," states, "Without monitoring, you have no way to tell whether the service is even working." This principle is core to any testing. (Source: "Site Reliability Engineering: How Google Runs Production Systems," O'Reilly, 2016, Chapter 6).
3. Validate Bigtable Performance (A): Cloud Bigtable performance is highly dependent on schema design and usage patterns. Google's documentation on "Designing your schema" and "Understanding Cloud Bigtable performance" highlights the need to test and validate that your design meets latency and throughput requirements under load. An incomplete test that ignores the database would yield misleading results. (Source: Google Cloud Documentation, Cloud Bigtable, "Schema design best practices").

Question: 18

You want to make a copy of a production Linux virtual machine in the US-Central region. You want to manage and replace the copy easily if there are changes on the production virtual machine. You will deploy the copy as a new instances in a different project in the US-East region. What steps must you take?

A. Use the Linux dd and netcat command to copy and stream the root disk contents to a new virtual
machine instance in the US-East region.

B. Create a snapshot of the root disk and select the snapshot as the root disk when you create a new
virtual machine instance in the US-East region.

C. Create an image file from the root disk with Linux dd command, create a new disk from the image
file, and use it to create a new virtual machine instance in the US-East region

D. Create a snapshot of the root disk, create an image file in Google Cloud Storage from the snapshot, and create a new virtual machine instance in the US-East region using the image file for the
root disk.

Answer:

D

Explanation:

This option describes the most robust and standard Google Cloud-native procedure for duplicating a virtual machine across different projects and regions. The process involves:

1. Creating a snapshot of the production VM's disk to capture its state at a specific point in time.
2. Creating a reusable, bootable custom image from that snapshot. Custom images are the standard mechanism for packaging a boot disk configuration.
3. Sharing the custom image with the destination project.
4. Using the shared custom image in the destination project to create a new VM instance in the desired region (US-East).

This entire workflow is manageable, repeatable, and leverages Google Cloud's infrastructure, fulfilling the requirements of the question.

Why Incorrect Options are Wrong:

A. Using dd and netcat is a manual, low-level data transfer method. It is not a scalable, reliable, or manageable cloud-native solution and is prone to errors.
B. Snapshots are regional resources and are scoped to a single project. This option fails to address the cross-project and cross-region requirements without additional steps, such as creating an image.
C. Creating a disk image file using dd within the guest OS is inefficient, requires significant temporary disk space on the source VM, and is not a recommended Google Cloud practice.

References:

1. Create a custom image from a snapshot: The first two steps of the correct answer (snapshot and image creation) are documented by Google Cloud. "You can create a custom image from a snapshot. Before you can create an image from a snapshot, you must first create a snapshot of a persistent disk."
Source: Google Cloud Documentation, "Create a custom image from a snapshot," in Compute Engine Docs Images Creating custom images.
2. Sharing images across projects: The key to solving the cross-project requirement is sharing the custom image. "To share a custom image with other projects, you grant the roles/compute.imageUser IAM role for that image to the principals that need to use the image."
Source: Google Cloud Documentation, "Sharing custom images," in Compute Engine Docs Images.

3. Creating a VM from a custom image: The final step is creating the new instance in the target project and region from the shared image. "After you share a custom image with another project, principals in that project can use the shared image to create VMs." The command gcloud compute instances create VMNAME --image=IMAGENAME --image-project=SOURCEPROJECTID demonstrates this capability.
Source: Google Cloud Documentation, "Creating a VM from a shared custom image," in Compute Engine Docs Images Sharing custom images.

Question: 19

Your company runs several databases on a single MySQL instance. They need to take backups of a specific database at regular intervals. The backup activity needs to complete as quickly as possible and cannot be allowed to impact disk performance. How should you configure the storage?

A. Configure a cron job to use the gcloud tool to take regular backups using persistent disk snapshots.

B. Mount a Local SSD volume as the backup location. After the backup is complete, use gsutil to move the backup to Google Cloud Storage.

C. Use gcsfuse to mount a Google Cloud Storage bucket as a volume directly on the instance and
write backups to the mounted location using mysqldump

D. Mount additional persistent disk volumes onto each virtual machine (VM) instance in a RAID10 array and use LVM to create snapshots to send to Cloud Storage.

Answer:

B

Explanation:

This solution directly addresses all constraints. Using a Local SSD as a temporary backup location provides the highest possible I/O performance, ensuring the mysqldump operation completes as quickly as possible. Because the Local SSD is a separate physical device from the primary persistent disk holding the MySQL data, the backup write operations are completely isolated, causing no performance impact on the active database. After the dump is complete, gsutil provides a reliable and efficient method to move the backup file to Google Cloud Storage for durable, long-term, and cost-effective retention.

Why Incorrect Options are Wrong:

A. Persistent disk snapshots are block-level copies of the entire disk, not a specific database. This method also causes I/O on the source disk, which violates the no-impact requirement.
C. Writing directly to a Cloud Storage bucket via gcsfuse would be extremely slow due to the high latency of translating file system operations into API calls, failing the "quickly as possible" requirement.
D. This is overly complex. LVM snapshots, like persistent disk snapshots, are block-level and do not provide the required granularity to back up a single, specific database.

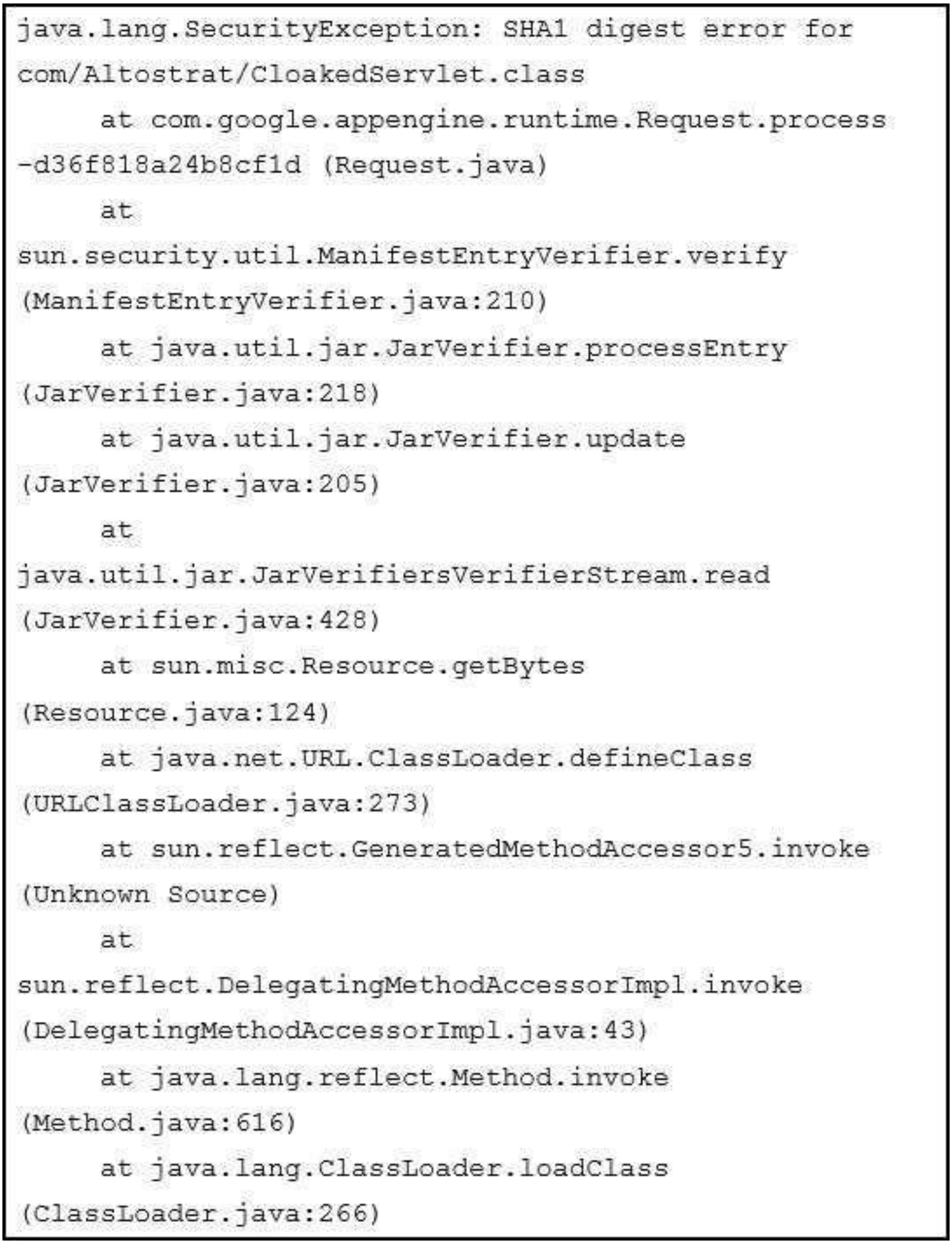
References:

1. Local SSD Performance and Use Case: Google Cloud official documentation states, "Local SSDs are suitable for temporary storage such as caches, processing space, or low-value data... Local SSDs offer very high IOPS and very low latency." This supports using it as a high-speed temporary location for the backup file.
Source: Google Cloud Documentation, "About Local SSDs", Section: "When to use Local SSDs".
2. Isolation from Primary Disk: Local SSDs are physically attached to the host server. Writing to a Local SSD uses a separate I/O path from a network-attached Persistent Disk, thus isolating the backup workload from the primary database disk.
Source: Google Cloud Documentation, "Storage options for Compute Engine", Section: "Storage option comparison table".
3. Persistent Disk Snapshot Granularity: Snapshots operate at the entire disk level. "A snapshot is a copy of the data on your persistent disk at a specific point in time." This confirms they are not suitable for backing up a specific database within an instance.
Source: Google Cloud Documentation, "About persistent disk snapshots", Section: "What are disk snapshots?".
4. gcsfuse Performance Limitations: The official documentation for gcsfuse warns about its performance characteristics: "gcsfuse has much higher latency than a local file system. As such, the rate of I/O for small files is likely to be slow." This makes it unsuitable for time-sensitive database dump operations.

Source: Google Cloud Documentation, "Cloud Storage FUSE", Section: "Performance".

Question: 20

You deploy your custom Java application to Google App Engine. It fails to deploy and gives you the following stack trace.



```
java.lang.SecurityException: SHA1 digest error for
com/Altostrat/CloakedServlet.class
     at com.google.appengine.runtime.Request.process
-d36f818a24b8cf1d (Request.java)
     at
sun.security.util.ManifestEntryVerifier.verify
(ManifestEntryVerifier.java:210)
     at java.util.jar.JarVerifier.processEntry
(JarVerifier.java:218)
     at java.util.jar.JarVerifier.update
(JarVerifier.java:205)
     at
java.util.jar.JarVerifiersVerifierStream.read
(JarVerifier.java:428)
     at sun.misc.Resource.getBytes
(Resource.java:124)
     at java.net.URL.ClassLoader.defineClass
(URLClassLoader.java:273)
     at sun.reflect.GeneratedMethodAccessor5.invoke
(Unknown Source)
     at
sun.reflect.DelegatingMethodAccessorImpl.invoke
(DelegatingMethodAccessorImpl.java:43)
     at java.lang.reflect.Method.invoke
(Method.java:616)
     at java.lang.ClassLoader.loadClass
(ClassLoader.java:266)
```

What should you do?

A. Upload missing JAR files and redeploy your application.

B. Digitally sign all of your JAR files and redeploy your application

C. Recompile the CLoakedServlet class using and MD5 hash instead of SHA1

Answer:

B

Explanation:

The provided stack trace shows a java.lang.SecurityException with two key messages: "digest ... does not match the digest specified in the manifest" and "has unsigned entries." This indicates that a signed JAR file has been tampered with or improperly packaged after its manifest and signature were created. The Java runtime, as part of the Google App Engine security sandbox, verifies the integrity of signed code upon deployment. It found that the content of a class file does not match the cryptographic hash stored in the JAR's manifest, and that an unsigned class was present in a signed JAR. Digitally signing all JAR files correctly will regenerate the manifests and signatures, ensuring the integrity of the package and resolving this security exception.

Why Incorrect Options are Wrong:

A. Upload missing JAR files and redeploy your application.
The stack trace indicates a SecurityException due to a file integrity issue, not a ClassNotFoundException or NoClassDefFoundError, which would signal a missing file.
C. Recompile the CLoakedServlet class using and MD5 hash instead of SHA1.
The problem is with the JAR package's digital signature and manifest, not the compilation of an individual class file or the specific hash algorithm used.

References:

1. Oracle, The JavaTM Tutorials, "Signing and Verifying JAR Files": This document explains the purpose of JAR signing. The runtime verifies that the JAR file has not been altered since it was signed. The error "digest ... does not match" is a direct result of a failed integrity check, which this verification process performs. The tutorial states, "When a signed JAR file is being verified, the digests of each of its entries are re-computed and compared with the digests recorded in the manifest."
Source: Oracle Java Documentation, https://docs.oracle.com/javase/tutorial/deployment/jar/signing.html, Section: "Verifying a Signed JAR File".
2. Google Cloud Documentation, "The Java 8 runtime environment": The App Engine standard environment runs applications in a security sandbox. This restricted environment enforces strict

security checks, including the verification of code integrity from signed JARs, which is why this SecurityException is thrown during deployment.
Source: Google Cloud Documentation,
https://cloud.google.com/appengine/docs/standard/java/runtime, Section: "The sandbox".
3. Oracle, Java SE Tools Reference, "jarsigner": The jarsigner tool is used to sign JAR files and verify their signatures. The error message "has unsigned entries" is a common problem when a signed JAR is modified without being re-signed. Using jarsigner to properly sign all application JARs is the correct procedure to fix this issue.
Source: Oracle Java Documentation,
https://docs.oracle.com/en/java/javase/17/docs/specs/man/jarsigner.html, Description section.

Question: 21

The application reliability team at your company has added a debug feature to their backend service to send all server events to Google Cloud Storage for eventual analysis. The event records are at least 50 KB and at most 15 MB and are expected to peak at 3,000 events per second. You want to minimize data loss. Which process should you implement?

A. • Append metadata to file body.
• Compress individual files.
• Name files with serverName-Timestamp.
• Create a new bucket if bucket is older than 1 hour and save individual files to the new bucket. Otherwise, save files to existing bucket

B. • Batch every 10,000 events with a single manifest file for metadata.
• Compress event files and manifest file into a single archive file.
• Name files using serverName-EventSequence.
• Create a new bucket if bucket is older than 1 day and save the single archive file to the new bucket. Otherwise, save the single archive file to existing bucket.

C. • Compress individual files.
• Name files with serverName-EventSequence.
• Save files to one bucket
• Set custom metadata headers for each object after saving.

D. • Append metadata to file body.
• Compress individual files.
• Name files with a random prefix pattern.
• Save files to one bucket

Answer:

D

Explanation:

The core challenge is handling a peak write rate of 3,000 objects per second to Cloud Storage while minimizing data loss. High, sustained write rates with sequential or predictable object names (like timestamps or sequences) can cause "hotspotting," where requests are concentrated on a few backend servers, leading to increased latency, errors, and potential data loss.
Option D is the only solution that addresses this critical performance issue by naming files with a random prefix. This practice distributes the write load across many different backend servers, maximizing ingestion performance and reliability. Appending metadata to the file body is also more efficient than a separate API call, and using a single bucket is the standard, scalable approach.

Why Incorrect Options are Wrong:

A: Naming files with serverName-Timestamp creates a sequential pattern that leads to hotspotting. Creating a new bucket every hour is unnecessary operational overhead and does not solve the hotspotting problem.
B: Batching 10,000 events into a single large archive file significantly increases the risk of data loss. If the batching process fails before writing the archive, all 10,000 events are lost.
C: Naming files with serverName-EventSequence is a sequential pattern that causes hotspotting. Setting metadata after saving the object requires a second API call per file, doubling the operations and increasing the potential for failure.

References:

1. Google Cloud Documentation - Cloud Storage Best Practices, "Request rate and access distribution guidelines": This document explicitly recommends randomizing object names to improve performance for high request rates. It states, "A high request rate to the same bucket can lead to request-rate limits... To avoid hotspotting, design your applications to upload to a wide range of object names across many different prefixes... a good way to introduce randomness is to hash the original object name." This directly supports the use of a "random prefix pattern" in option D and invalidates the sequential naming in A, B, and C.
2. Google Cloud Documentation - Cloud Storage, "Object naming guidelines": This guide details how to choose object names for optimal performance. It advises, "Using sequential names... can lead to hotspotting... To avoid such issues, you should avoid using sequential names. To get the best performance, use completely random object names."
3. Google Cloud Documentation - Cloud Storage, "Viewing and editing object metadata": This page describes two ways to set metadata: during the upload or as a separate update. For efficiency and atomicity at high throughput, including metadata in the initial upload request (either in the body or as a header) is superior to a separate update call, making the approach in D (append to body) more robust than C (set after saving).

Question: 22

A lead software engineer tells you that his new application design uses websockets and HTTP sessions that are not distributed across the web servers. You want to help him ensure his application will run property on Google Cloud Platform. What should you do?

A. Help the engineer to convert his websocket code to use HTTP streaming.

B. Review the encryption requirements for websocket connections with the security team.

C. Meet with the cloud operations team and the engineer to discuss load balancer options.

D. Help the engineer redesign the application to use a distributed user session service that does not
rely on websockets and HTTP sessions.

Answer:

C

Explanation:

The application's architecture has two specific requirements that must be addressed at the infrastructure level: support for the WebSocket protocol and a mechanism to handle non-distributed HTTP sessions. The latter requirement implies the need for session affinity (or "sticky sessions"), ensuring that requests from a specific user are consistently routed to the same backend server.

Google Cloud Load Balancing services are designed to manage this type of traffic distribution. Specifically, services like the External HTTP(S) Load Balancer natively support the WebSocket protocol and provide multiple options for configuring session affinity (e.g., based on generated cookies, client IP, or HTTP headers). Therefore, the most direct and appropriate action is to discuss these load balancer options to find a configuration that supports the application's existing design without requiring a code rewrite.

Why Incorrect Options are Wrong:

A. This suggests an unnecessary application redesign. Google Cloud's load balancers provide native support for websockets, so converting the code to a different protocol is not required.

B. While security is important, reviewing encryption does not solve the core architectural problem of routing traffic correctly to support websockets and non-distributed sessions.

D. This proposes a significant and costly application redesign as the first step. A Cloud Architect should first seek to support the existing application requirements with appropriate infrastructure before recommending a complete rewrite.

References:

1. Google Cloud Documentation, "External HTTP(S) Load Balancer overview": This document explicitly states the load balancer's capabilities. Under the "Features" section, it lists "WebSocket support." The documentation explains: "Google Cloud Armor and Cloud CDN can be used with WebSockets. The WebSocket protocol... provides a full-duplex communication channel between a client and a server. The channel is initiated from an HTTP(S) request. The External HTTP(S) Load Balancer has native support for the WebSocket protocol."
2. Google Cloud Documentation, "Session affinity": This page details how to configure session affinity for various load balancers. For the Global External HTTP(S) Load Balancer, it states: "Session affinity sends all requests from the same client to the same virtual machine (VM) instance or endpoint... This is useful for applications that require stateful sessions." It then describes the different types, including Generated cookie affinity, Header field affinity, and HTTP cookie affinity, which directly address the "non-distributed sessions" requirement.
3. Google Cloud Architecture Framework, "System design pillar": This framework emphasizes selecting the right Google Cloud products and services to meet design requirements. The "Networking and connectivity principles" section guides architects to "Choose the right load balancer for your needs." This aligns with option C, which involves evaluating load balancer options to fit the application's specific websocket and session state requirements.

Question: 23

You want to enable your running Google Container Engine cluster to scale as demand for your application changes. What should you do?

A. Add additional nodes to your Container Engine cluster using the following command:
gcloud container clusters resize CLUSTERNAME --size 10

B. Add a tag to the instances in the cluster with the following command:
gcloud compute instances add-tags INSTANCE --tags enable --autoscaling max-nodes-10

C. Update the existing Container Engine cluster with the following command:
gcloud alpha container clusters update mycluster --enable-autoscaling --min-nodes=1 --max-nodes=10

D. Create a new Container Engine cluster with the following command:
gcloud alpha container clusters create mycluster --enable-autocaling --min-nodes=1
--max-nodes=10
and redeploy your application.

Answer:

C

Explanation:

The most direct and appropriate method to enable automatic scaling on an existing, running Google Kubernetes Engine (GKE) cluster is to update its configuration. The gcloud container clusters update command is designed for this purpose. Using the --enable-autoscaling flag along with --min-nodes and --max-nodes parameters allows the cluster autoscaler to be activated and configured with the desired scaling boundaries. This modifies the cluster in-place without requiring workload migration.

Why Incorrect Options are Wrong:

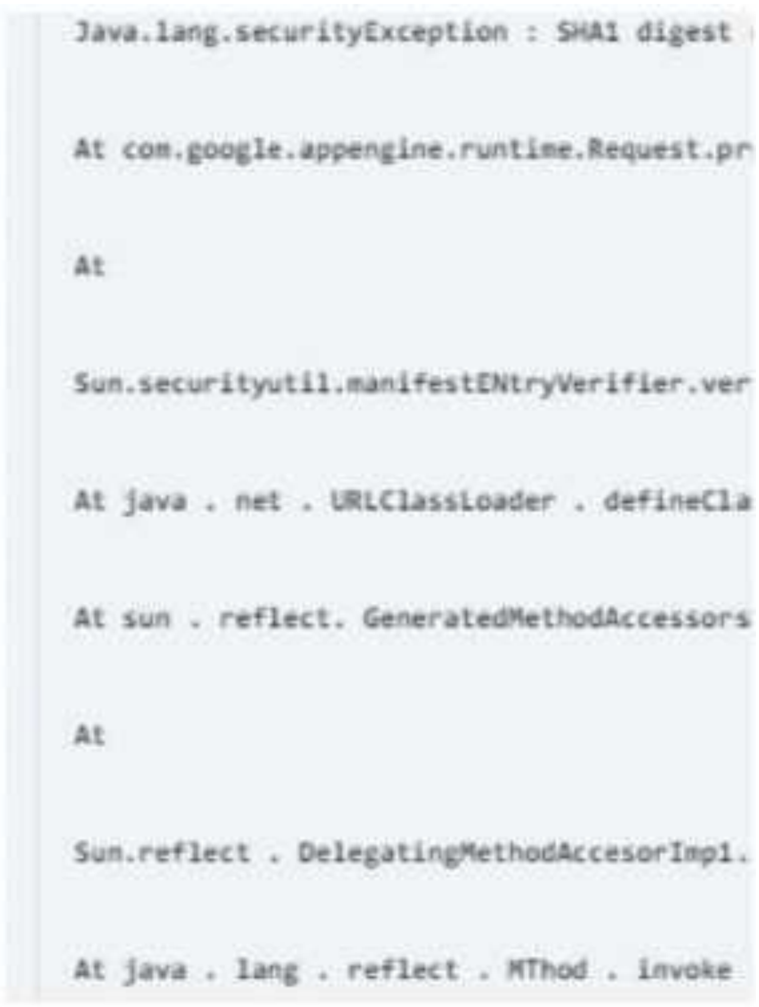
A: The resize command performs a one-time, manual scaling operation to a fixed number of nodes. It does not enable the cluster to scale automatically based on workload demand.
B: Adding tags to individual Compute Engine instances is used for networking rules or organization, not for enabling the GKE cluster autoscaler, which is a managed feature of the cluster itself.
D: This command creates an entirely new cluster. While it correctly enables autoscaling, it does not modify the running cluster as requested and would require a disruptive migration of all applications.

References:

1. Google Cloud Documentation - Autoscaling a cluster: The official documentation explicitly provides the command for enabling cluster autoscaling on an existing cluster: gcloud container clusters update CLUSTERNAME --enable-autoscaling --min-nodes=MINNODES --max-nodes=MAXNODES. This directly supports option C as the correct procedure. (See "Enabling cluster autoscaling for an existing cluster" section).
Source: Google Cloud, "Autoscaling a cluster", cloud.google.com/kubernetes-engine/docs/how-to/cluster-autoscaler.
2. Google Cloud SDK Documentation - gcloud container clusters update: The reference for this command confirms its purpose is to "Update settings for a cluster" and lists --enable-autoscaling, --max-nodes, and --min-nodes as valid flags for managing the autoscaler.
Source: Google Cloud SDK, "gcloud container clusters update", cloud.google.com/sdk/gcloud/reference/container/clusters/update.
3. Google Cloud SDK Documentation - gcloud container clusters resize: This documentation clarifies that the resize command is for manual scaling: "This command is used for manual scaling. You can use this command to increase or decrease the number of nodes in a cluster." This confirms why option A is incorrect for automatic scaling.
Source: Google Cloud SDK, "gcloud container clusters resize", cloud.google.com/sdk/gcloud/reference/container/clusters/resize.

Question: 24

You deploy your custom java application to google app engine. It fails to deploy and gives you the following stack trace:



```
Java.lang.securityException : SHA1 digest

At com.google.appengine.runtime.Request.pr

At

Sun.securityutil.manifestENtryVerifier.ver

At java . net . URLClassLoader . defineCla

At sun . reflect. GeneratedMethodAccessors

At

Sun.reflect . DelegatingMethodAccesorImpl.

At java . lang . reflect . MThod . invoke
```

A. Recompile the CLoakedServlet class using and MD5 hash instead of SHA1

B. Digitally sign all of your JAR files and redeploy your application.

C. Upload missing JAR files and redeploy your application

Answer:

B

Explanation:

The stack trace indicates a java.lang.SecurityException because the SHA1 digest of a JAR file does not match the expected digest listed in the application's deployment descriptor (META-INF/application.xml). This is a file integrity verification failure. The application server, in this case, the App Engine runtime, is unable to verify that the JAR file is authentic and has not been tampered with. Digitally signing the JAR files with a trusted certificate creates a manifest with correct digests and a signature to guarantee their integrity. Redeploying with properly signed JARs will allow the runtime to successfully verify the files, resolving the security exception.

Why Incorrect Options are Wrong:

A. The error is with the javax.servlet-api-3.0.1.jar file's integrity, not a custom class. Changing the hash algorithm for a single class is irrelevant to the JAR verification process.
C. The error is a digest mismatch (does not match), not a ClassNotFoundException. This confirms the file is present but has failed an integrity check, rather than being missing.

References:

1. Oracle Java Documentation, "Signing and Verifying JAR Files": This document explains the purpose of JAR signing. It states, "You can sign JAR files to ensure their integrity and authenticity... When a signed JAR file is loaded, the Java runtime can verify the signature to ensure that the file's contents have not been changed since it was signed." The SecurityException in the question is a direct result of this verification failing. (Source: Oracle, JDK 8 Documentation, The Java Tutorials, Deployment, Signing and Verifying JAR Files).
2. Google Cloud Documentation, "Java 8 Runtime Environment": The error log references the WEB-INF/lib/ directory, which is a standard part of the required application directory structure for Java applications on App Engine. This confirms the context is a standard Java web application deployment where such integrity checks are common. (Source: Google Cloud, App Engine standard environment for Java 8 documentation, "The Java 8 Runtime Environment", Section: "Organizing your files").
3. Princeton University, COS 432: Information Security, Lecture 18: Java Security: This courseware discusses the Java security model, including the sandbox, SecurityManager, and code signing. It explains that the SecurityManager is responsible for throwing a SecurityException when a security policy is violated, such as when code integrity cannot be verified via its signature. (Source: Princeton University, Department of Computer Science, COS 432, Lecture 18, Slides 15-18 on "Code Signing").

Question: 25

You are designing a mobile chat application. You want to ensure people cannot spoof chat messages, by providing a message were sent by a specific user. What should you do?

A. Tag messages client side with the originating user identifier and the destination user.

B. Encrypt the message client side using block-based encryption with a shared key.

C. Use public key infrastructure (PKI) to encrypt the message client side using the originating user's private key.

D. Use a trusted certificate authority to enable SSL connectivity between the client application and the server.

Answer:

C

Explanation:

The question requires a solution to prevent message spoofing by proving a message was sent by a specific user. This is a classic use case for digital signatures, which provide authenticity and non-repudiation. The process described in option C, using Public Key Infrastructure (PKI) to encrypt with the originating user's private key, is the definition of creating a digital signature. The recipient can then use the sender's public key to verify the signature, confirming that only the holder of the private key could have sent the message. This cryptographically proves the sender's identity and prevents them from denying they sent the message.

Why Incorrect Options are Wrong:

A. Tagging messages with a user identifier on the client side is insecure metadata. A malicious user can easily modify the client application to forge this tag.
B. Shared key (symmetric) encryption provides confidentiality, ensuring only those with the key can read the message. It does not prove origin, as anyone with the shared key could have created the message.
D. SSL/TLS secures the communication channel between the client and the server (data in transit). It does not cryptographically sign the individual messages to prove the user's identity to other chat participants.

References:

1. Google Cloud Documentation, Cloud Key Management Service, "Digital signatures": "Digital signatures are commonly used to verify the integrity and authenticity of data. For example, you can use digital signatures to verify that a binary was released by a specific developer... A private key is used to create a digital signature, and the corresponding public key is used to validate the signature." This directly supports the mechanism described in option C for proving origin.

2. Google Cloud Documentation, Cloud Key Management Service, "Asymmetric encryption": This document distinguishes between symmetric and asymmetric keys, stating, "Asymmetric keys can be used for either asymmetric encryption or for creating digital signatures." This clarifies that the PKI approach (asymmetric keys) is the correct tool for signatures, unlike the symmetric approach in option B.
3. MIT OpenCourseWare, 6.857 Computer and Network Security, Fall 2017, Lecture 8: Public-Key Cryptography: The lecture notes state that a digital signature scheme provides "(1) Authentication (of origin), (2) Integrity (of message), (3) Non-repudiation (by origin)." The process is defined as Sign(SK, M) where SK is the secret (private) key, which aligns perfectly with option C's methodology.
4. Google Cloud Security Whitepaper, "Encryption in Transit in Google Cloud": This paper details how Google Cloud uses TLS to secure data in transit (Section: "Default encryption in transit"). This supports the reasoning for why option D is incorrect, as TLS secures the transport layer between two points (e.g., client and server), not the authenticity of the application-layer message itself for end-to-end verification between users.

Question: 26

You created a pipeline that can deploy your source code changes to your infrastructure in instance groups for self healing. One of the changes negatively affects your key performance indicator. You are not sure how to fix it and investigation could take up to a week. What should you do?

A. Log in to a server, and iterate a fix locally

B. Change the instance group template to the previous one, and delete all instances.

C. Revert the source code change and rerun the deployment pipeline

D. Log into the servers with the bad code change, and swap in the previous code

Answer:

C

Explanation:

The most effective and reliable strategy is to treat the rollback as a new deployment. By reverting the problematic change in the source code repository, you maintain a clear and auditable history of the application's state. Rerunning the deployment pipeline leverages your existing automation to build and deploy the last known-good version of the code. This ensures the change is applied consistently across all instances in the self-healing group and aligns with DevOps and Continuous Integration/Continuous Deployment (CI/CD) best practices, where the source control system is the single source of truth.

Why Incorrect Options are Wrong:

A. Logging in to a server to iterate a fix is a manual process that is not scalable and will be undone when the instance group's self-healing mechanism replaces the instance.
B. While changing the instance template would work, it's a manual infrastructure-level intervention. The root cause is the application code, and the best practice is to fix the source of truth (the code) and let the pipeline manage the infrastructure changes.
D. Manually swapping code on live servers is an anti-pattern. It is not repeatable, not auditable, and any changes will be lost during self-healing events or subsequent automated deployments.

References:

1. Google Cloud Documentation, DevOps tech: Continuous delivery: "A key goal of continuous delivery is to make your release process a low-risk event that you can perform at any time and on demand..... Because you are deploying smaller changes, you can more easily pinpoint and address bugs and roll back changes if necessary." This supports the principle of rolling back a problematic change through the established process.
Source: Google Cloud, "DevOps tech: Continuous delivery", Section: "What is continuous

delivery?".
2. Google, Site Reliability Engineering, Chapter 8 - Release Engineering: "A key feature of our release system is the ability to quickly and safely roll back a release that is found to be bad. ... Rollbacks use the same infrastructure as rollouts, but in reverse." This highlights the best practice of using the same automated system (the pipeline) for rollbacks as for deployments, which is achieved by reverting the code and re-running the pipeline.
Source: Beyer, B., Jones, C., Petoff, J., & Murphy, N. R. (2016). Site Reliability Engineering: How Google Runs Production Systems. O'Reilly Media. Chapter 8, Section: "Rollout Policies".
3. Google Cloud Documentation, Managed instance groups (MIGs): "A managed instance group (MIG) ... maintains high availability of your apps by proactively keeping your VMs (instances) running. If a VM in the group stops, crashes, or is deleted... the MIG automatically recreates it in accordance with the group's instance template". This confirms that any manual changes made directly to an instance (as suggested in options A and D) will be lost.
Source: Google Cloud, "Managed instance groups (MIGs)", Section: "High availability".

Question: 27

Your organization wants to control IAM policies for different departments independently, but centrally. Which approach should you take?

A. Multiple Organizations with multiple Folders

B. Multiple Organizations, one for each department

C. A single Organization with Folder for each department

D. A single Organization with multiple projects, each with a central owner

Answer:

C

Explanation:

The Google Cloud resource hierarchy is designed for this exact use case. A single Organization node serves as the root, enabling centralized governance and application of organization-wide IAM policies. Folders are used to group resources, such as projects, that share common IAM policies. By creating a Folder for each department, you can delegate administrative rights to departmental teams, allowing them to manage their own projects and resources independently within their Folder. This structure provides both the central control required by the organization and the departmental autonomy needed for efficient operations.

Why Incorrect Options are Wrong:

A. Multiple Organizations with multiple Folders: Using multiple Organizations breaks the principle of central control, as each Organization is a distinct root entity with its own separate policies.
B. Multiple Organizations, one for each department: This approach completely fragments governance, creating isolated silos for each department and making central IAM policy enforcement impossible.
D. A single Organization with multiple projects, each with a central owner: While this provides central control, it lacks the intermediate grouping layer (Folders) for departments, making policy management at scale inefficient and difficult to delegate.

References:

1. Google Cloud Documentation, Resource Manager, "Cloud Platform resource hierarchy": "Folders are an additional grouping mechanism on top of projects... Folders are commonly used to model different departments, teams, or legal entities within a company. For example, a first level of folders could represent the major departments in your organization." This directly supports using folders to represent departments.
2. Google Cloud Documentation, Resource Manager, "Creating and managing folders", Section: "Using folders for access control": "You can use folders to isolate resources for different

departments... Access to resources can be limited by department by assigning IAM roles at the folder level. All projects and folders within a parent folder inherit the IAM policies of that folder." This confirms that folders are the correct tool for delegating departmental control.
3. Google Cloud Documentation, IAM, "Policy inheritance": "The resource hierarchy for policy evaluation includes the organization, folders, projects, and resources... The child resource inherits the parent's policy." This explains the mechanism for central control from the Organization node downwards.
4. Google Cloud Architecture Framework, "Design a resource hierarchy for your Google Cloud landing zone": In the "Folder structure" section, a common pattern recommended is: "A folder for each department, such as Dept-A and Dept-B." This establishes the chosen answer as a documented best practice.

Question: 28

A recent audit that a new network was created in Your GCP project. In this network, a GCE instance has an SSH port open 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.

Answer:

C

Explanation:

To investigate the origin of a newly created Google Cloud resource, the correct tool is Cloud Logging, which captures Admin Activity audit logs. These logs record API calls that modify the configuration or metadata of resources, such as creating a VPC network. By filtering the logs in the Logs Explorer for the resource type gcenetwork and the API method that creates the network (compute.networks.insert), you can pinpoint the exact log entry. This entry will contain the identity of the principal (user, group, or service account) that made the call, the source IP address, and the timestamp, thereby revealing the network's origin.

Why Incorrect Options are Wrong:

A. The Cloud Monitoring alerting console is used for configuring and viewing alerts based on metrics, logs, and uptime checks, not for retrospectively searching historical audit logs for specific events.
B. Resource creation events are captured in Admin Activity logs, not Data Access logs. Data Access logs track when data within a resource is read or written, which is not relevant here.
D. Checking system logs on the GCE instance would only show who has logged into the virtual machine, not who created the underlying network or the instance itself. The creator may have never accessed the instance.

References:

1. Cloud Audit Logs Overview: "Admin Activity audit logs contain log entries for API calls or other actions that modify the configuration or metadata of resources. For example, these logs record when users create VM instances or change Identity and Access Management permissions." This confirms that creating a network is an Admin Activity.
Source: Google Cloud Documentation, "Cloud Audit Logs", Overview section.
2. Querying Logs in Logs Explorer: The Logs Explorer allows for building queries to find specific log entries. A query to find network creation events would look like: resource.type="gcenetwork" and protoPayload.methodName="v1.compute.networks.insert". This demonstrates the method described in the correct answer.
Source: Google Cloud Documentation, "Build queries in the Logs Explorer", Query for a resource type and log name section.
3. Compute Engine Audited Operations: The official documentation lists the specific API methods that are logged. For creating a network, the method is v1.compute.networks.insert. This validates that searching for an "insert" entry for a GCE Network is the correct procedure.
Source: Google Cloud Documentation, "Compute Engine audit logging information", Audited operations table.

Question: 29

As part of implementing their disaster recovery plan, your company is trying to replicate their production MySQL database from their private data center to their GCP project using a Google Cloud VPN connection. They are experiencing latency issues and a small amount of packet loss that is disrupting the replication. What should they do?

A. Configure their replication to use UDP.

B. Configure a Google Cloud Dedicated Interconnect.

C. Restore their database daily using Google Cloud SQL.

D. Add additional VPN connections and load balance them.

E. Send the replicated transaction to Google Cloud Pub/Sub.

Answer:

B

Explanation:

The core issue is the use of a Cloud VPN connection over the public internet for a latency-sensitive and loss-intolerant workload like production database replication. The public internet provides "best-effort" delivery, which leads to unpredictable latency and packet loss, disrupting the replication process. Google Cloud Dedicated Interconnect establishes a private, direct physical connection to Google's network, bypassing the public internet. This provides a stable, low-latency, and high-bandwidth connection with a Service Level Agreement (SLA), making it the appropriate solution for reliable, enterprise-grade disaster recovery scenarios.

Why Incorrect Options are Wrong:

A. Configure their replication to use UDP.
UDP is an unreliable, connectionless protocol that does not guarantee packet delivery or order. Database replication requires the guaranteed, in-order delivery provided by TCP to prevent data corruption.
C. Restore their database daily using Google Cloud SQL.
This is a backup-and-restore strategy, not replication. A daily restore implies a Recovery Point Objective (RPO) of up to 24 hours, which is typically unacceptable for a production database DR plan.
D. Add additional VPN connections and load balance them.
While this can increase aggregate throughput and add redundancy, all connections still traverse the unreliable public internet. It does not solve the fundamental problems of inconsistent latency and packet loss for a single replication stream.
E. Send the replicated transaction to Google Cloud Pub/Sub.

This introduces significant architectural complexity, requiring custom producers and consumers. It does not address the underlying network instability between the on-premises data center and GCP, which is the root cause of the problem.

References:

1. Google Cloud Documentation, "Choosing a Network Connectivity product": This document directly compares Cloud VPN and Cloud Interconnect. It states, "Cloud Interconnect provides low latency, high availability connections that enable you to reliably transfer data between your on-premises and Virtual Private Cloud (VPC) networks." It also notes for Cloud VPN, "Because this connection traverses the internet, its performance...can be inconsistent." This supports choosing Interconnect (B) over VPN (D).
2. Google Cloud Documentation, "Dedicated Interconnect overview": This page highlights the key benefits of Dedicated Interconnect: "Lower latency. Traffic between your on-premises and VPC networks doesn't touch the public internet. Instead, it travels over a dedicated connection with lower latency." This directly addresses the problem described in the question.
3. MySQL 8.0 Reference Manual, "Section 19.1.1 Replication Implementation Overview": The MySQL documentation describes the replication process where a replica's I/O thread connects to the source server over the network to read the binary log. This connection relies on a stable, reliable transport protocol like TCP, making UDP (A) an unsuitable choice.
4. Kurose, J. F., & Ross, K. W. (2017). Computer Networking: A Top-Down Approach (7th ed.). Pearson. In Chapter 3, Section 3.3, UDP is described as providing an "unreliable data transfer service," while Section 3.5 describes TCP's "reliable data transfer" service, reinforcing why UDP is incorrect for database replication.

Question: 30

Your customer support tool logs all email and chat conversations to Cloud Bigtable for retention and analysis. What is the recommended approach for sanitizing this data of personally identifiable information or payment card information before initial storage?

A. Hash all data using SHA256

B. Encrypt all data using elliptic curve cryptography

C. De-identify the data with the Cloud Data Loss Prevention API

D. Use regular expressions to find and redact phone numbers, email addresses, and credit card numbers

Answer:

C

Explanation:

The most appropriate and recommended approach is to use the Cloud Data Loss Prevention (DLP) API. This is a fully managed Google Cloud service specifically designed to discover, classify, and de-identify sensitive data like Personally Identifiable Information (PII) and Payment Card Information (PCI) within unstructured text streams or storage. Cloud DLP uses pre-trained classifiers (infoType detectors) to accurately identify sensitive data and offers various transformation techniques such as redaction, masking, and tokenization. This allows for the sanitization of sensitive information while preserving the utility of the non-sensitive data for analysis, directly addressing the question's requirements before the data is stored in Bigtable.

Why Incorrect Options are Wrong:

A. Hash all data using SHA256: Hashing the entire log entry would render the non-sensitive data unusable for the stated purpose of retention and analysis, as it's a one-way, irreversible process.
B. Encrypt all data using elliptic curve cryptography: Encryption protects data but does not sanitize or de-identify it. The sensitive information still exists in an encrypted form and is not removed or transformed for analysis.
D. Use regular expressions to find and redact phone numbers, email addresses, and credit card numbers: This approach is brittle, difficult to maintain, and prone to errors. It cannot reliably detect all formats of sensitive data (e.g., international phone numbers, names) or validate them (e.g., credit card checksums), unlike the managed and sophisticated detectors in Cloud DLP.

References:

1. Cloud Data Loss Prevention Documentation - De-identification of sensitive data: "You can use Cloud DLP to de-identify sensitive data in your content. De-identification is the process of removing identifying information from data. Its goal is to allow the sharing and use of personal data while protecting privacy." This page details methods like redaction and masking, which are ideal for the scenario.
Source: Google Cloud Documentation, "De-identification of sensitive data".
2. Cloud Data Loss Prevention Documentation - InfoType detector reference: This document lists the extensive built-in detectors for PII and PCI data, such as CREDITCARDNUMBER, EMAILADDRESS, and PHONENUMBER. This demonstrates why Cloud DLP is superior to custom regular expressions.
Source: Google Cloud Documentation, "InfoType detector reference".
3. Google Cloud Architecture Framework - Security, privacy, and compliance: "Use Cloud DLP to discover, classify, and redact sensitive data. For example, you can use Cloud DLP to find and redact credit card numbers from a chat transcript before you store the transcript." This directly cites the use case described in the question as a recommended practice.
Source: Google Cloud Architecture Framework, Security pillar, "Implement least privilege".

Question: 31

You are using Cloud Shell and need to install a custom utility for use in a few weeks. Where can you store the file so it is in the default execution path and persists across sessions?

A. /bin

B. Cloud Storage

C. /google/scripts

D. /usr/local/bin

Answer:

A

Explanation:

Google Cloud Shell provides a 5 GB persistent disk mounted as your $HOME directory. This ensures that any files and configurations within $HOME are preserved across sessions. The Cloud Shell environment is pre-configured so that if a /bin directory exists, it is automatically added to the system's execution path ($PATH). Therefore, placing a custom utility in /bin meets both requirements: the file will persist for future use, and it can be executed directly by name from any location without specifying the full path.

Why Incorrect Options are Wrong:

B. Cloud Storage: This is an object storage service, not a local filesystem directory in the execution path. The utility would need to be downloaded to the VM before it could be run.
C. /google/scripts: This directory is part of the ephemeral Cloud Shell virtual machine instance. Any files placed here will be lost when your session ends.
D. /usr/local/bin: While this directory is in the default execution path, it resides on the ephemeral VM's filesystem. It does not persist across sessions, so the utility would be deleted.

References:

1. Google Cloud Documentation, "How Cloud Shell works", Section: Persistent disk storage: "Cloud Shell provisions 5 GB of persistent disk storage on your temporarily allocated virtual machine. This storage is located at your $HOME directory and persists between sessions... Any modifications you make to your home directory, including installed software, scripts, and user configuration files like .bashrc and .vimrc, persist between sessions."
2. Google Cloud Documentation, "Cloud Shell features", Section: A pre-configured gcloud CLI and other utilities: This section implies a standard Linux environment. In standard Linux configurations (like the Debian base for Cloud Shell), the default .profile script adds $HOME/bin to the $PATH if the directory exists. This behavior, combined with the persistence of the $HOME directory, makes /bin the correct location.

3. Google Cloud Documentation, "Customizing your Cloud Shell environment": This guide explains how to make persistent customizations. It states, "When you start Cloud Shell, a bash shell is run and any commands in /.bashrc and /.profile are executed." This confirms that standard shell startup scripts, which typically configure the path to include /bin, are honored and persist.

Question: 32

You want to create a private connection between your instances on Compute Engine and your on- premises data center. You require a connection of at least 20 Gbps. You want to follow Google- recommended practices. How should you set up the connection?

A. Create a VPC and connect it to your on-premises data center using Dedicated Interconnect.

B. Create a VPC and connect it to your on-premises data center using a single Cloud VPN.

C. Create a Cloud Content Delivery Network (Cloud CDN) and connect it to your on-premises data
center
using Dedicated Interconnect.

D. Create a Cloud Content Delivery Network (Cloud CDN) and connect it to your on-premises datacenter
using a single Cloud VPN.

Answer:

A

Explanation:

The requirement is for a private, high-bandwidth connection (at least 20 Gbps) between an on-premises data center and a Google Cloud VPC. Dedicated Interconnect is the Google-recommended service for this use case. It provides a direct, private, physical connection to Google's network. To achieve the 20 Gbps throughput requirement, you can provision two 10 Gbps Dedicated Interconnect circuits. This solution connects your on-premises environment directly to your VPC, providing the necessary performance and privacy.

Why Incorrect Options are Wrong:

B. A single Cloud VPN tunnel provides a maximum of 3 Gbps of bandwidth, which does not meet the 20 Gbps requirement.
C. Cloud CDN is a content delivery network used for caching and distributing web content to users globally; it is not a service for establishing private network connectivity.
D. This option is incorrect for two reasons: Cloud CDN is the wrong service for this purpose, and a single Cloud VPN does not meet the bandwidth requirement.

References:

1. Google Cloud Documentation, "Choosing a Network Connectivity product": In the comparison table under "Features," Cloud Interconnect is listed with a bandwidth of "10 Gbps or 100 Gbps per link," suitable for high-throughput needs. In contrast, Cloud VPN is listed with a bandwidth of "Up to 3 Gbps per tunnel." This directly supports the choice of Interconnect over VPN for the 20 Gbps

requirement.
2. Google Cloud Documentation, "Dedicated Interconnect overview": This document states, "Dedicated Interconnect provides a direct physical connection between your on-premises network and Google's network... Connections are offered as one or more 10-Gbps or 100-Gbps Ethernet connections." This confirms that multiple 10 Gbps connections can be used to meet the 20 Gbps requirement.
3. Google Cloud Documentation, "Cloud CDN overview": This document describes Cloud CDN as a service that "uses Google's global edge network to bring content closer to your users." This clarifies that its purpose is content distribution, not establishing a private network link for backend services, making options C and D incorrect.

Question: 33

You are analyzing and defining business processes to support your startup's trial usage of GCP, and you don't yet know what consumer demand for your product will be. Your manager requires you to minimize GCP service costs and adhere to Google best practices. What should you do?

A. Utilize free tier and sustained use discounts. Provision a staff position for service cost management.

B. Utilize free tier and sustained use discounts. Provide training to the team about service cost management.

C. Utilize free tier and committed use discounts. Provision a staff position for service cost management.

D. Utilize free tier and committed use discounts. Provide training to the team about service cost management.

Answer:

D

Explanation:

Google Cloud's best practices for cost management are centered around the principles of FinOps, which promotes a culture of cost accountability across the organization. For a startup with unknown future demand, the most effective strategy involves two key elements. First, providing training to the entire team ensures that developers and operators understand the cost implications of their architectural choices. Second, while demand is unpredictable now, the business process should be built to proactively identify a stable, baseline workload as soon as it emerges. Once identified, applying Committed Use Discounts (CUDs) to this baseline will yield the most significant savings (up to 70%). This proactive approach of planning for CUDs is superior to passively relying on less impactful, automatic discounts.

Why Incorrect Options are Wrong:

A. Utilize free tier and sustained use discounts. Provision a staff position for service cost management.
Hiring a dedicated staff position for a startup in a trial phase is not cost-effective and contradicts the goal of minimizing costs.
B. Utilize free tier and sustained use discounts. Provide training to the team about service cost management.
Sustained Use Discounts (SUDs) are automatic and less impactful than CUDs. A best-practice business process should be proactive, planning to use the most effective tools, not just relying on passive, automatic ones.
C. Utilize free tier and committed use discounts. Provision a staff position for service cost

management.
As with option A, provisioning a dedicated staff position is an unnecessary expense for a startup and is not a recommended best practice for this scenario.

References:

1. Google Cloud Cost Management Solutions: The official documentation outlines key principles, including empowering teams and using intelligent pricing models. It states, "Empower your teams with the training, resources, and tools they need to operate with cost-efficiency" and "Take advantage of intelligent recommendations to... use pricing models like committed use discounts to optimize costs." This directly supports the combination of training and CUDs in option D. (Source: Google Cloud, "Cost Management", Section: "Key principles of cloud cost management").
2. Committed Use Discounts (CUDs) Documentation: CUDs are Google's primary tool for reducing costs on predictable workloads. The documentation states, "Committed use discounts (CUDs) provide deeply discounted prices in exchange for your commitment... The discounts are ideal for workloads with predictable resource needs." A best practice business process involves identifying these predictable needs and applying CUDs. (Source: Google Cloud, "Committed use discounts", Overview section).
3. Sustained Use Discounts (SUDs) Documentation: This page explains that SUDs are automatic and apply when resources run for a significant portion of the month. While beneficial, they are a passive mechanism and have been largely superseded by the more strategic CUDs for vCPU and memory, making them a less central part of a proactive cost management process. (Source: Google Cloud, "Sustained use discounts", Overview section).

Question: 34

You are building a continuous deployment pipeline for a project stored in a Git source repository and want to ensure that code changes can be verified deploying to production. What should you do?

A. Use Spinnaker to deploy builds to production using the red/black deployment strategy so that changes can easily be rolled back.

B. Use Spinnaker to deploy builds to production and run tests on production deployments.

C. Use Jenkins to build the staging branches and the master branch. Build and deploy changes to production for 10% of users before doing a complete rollout.

D. Use Jenkins to monitor tags in the repository. Deploy staging tags to a staging environment for testing.
After testing, tag the repository for production and deploy that to the production environment.

Answer:

D

Explanation:

This option describes a robust and standard Continuous Integration/Continuous Deployment (CI/CD) pipeline. It correctly separates the environments, using a staging environment for verification, which directly addresses the question's core requirement to "ensure that code changes can be verified deploying to production." Using Git tags (staging, production) is a common and effective practice to trigger deployments of specific, immutable versions of the code to the corresponding environments. This ensures that the exact code that was tested in staging is the code that gets deployed to production, following a successful verification process.

Why Incorrect Options are Wrong:

A. Red/black (or blue/green) is a production deployment strategy that minimizes downtime. It does not, by itself, provide a pre-production verification stage, which is the primary requirement.
B. Running tests on production deployments is a high-risk practice. The goal is to find and fix issues before they reach production and impact users, not after.
C. A canary deployment (releasing to 10% of users) is a strategy for a phased rollout to production. Verification happens on a subset of live users, not in a dedicated test environment before production.

References:

1. Google Cloud Documentation - CI/CD on Google Cloud: This documentation outlines the typical stages of a software delivery pipeline: Source, Build, Test, and Deploy. Option D aligns perfectly with this model by including a dedicated "Test" stage (in the staging environment) before the final "Deploy" stage (to production). The document emphasizes that "Each stage acts as a gate that vets a new change for quality."
Source: Google Cloud Documentation, "CI/CD modernization: A guide to building a software delivery pipeline", Section: "Stages of a software delivery pipeline".
2. Google Cloud Solutions - Continuous deployment to GKE using Jenkins: This official tutorial demonstrates a multi-environment pipeline. It uses separate Git branches (dev, production) to trigger deployments to different environments. Using tags, as described in option D, is an analogous and widely accepted best practice for managing releases, where a specific tag triggers the promotion of a build through the pipeline from staging to production.
Source: Google Cloud Solutions, "Continuous deployment to GKE using Jenkins", Section: "Understanding the application development and deployment workflow".
3. Google Cloud Documentation - Release and deployment strategies: This page describes strategies like blue/green (red/black) and canary deployments. It positions them as methods for the final step of deploying to production to reduce risk, which confirms that they are distinct from the pre-production verification step described in option D.
Source: Google Cloud Documentation, "Application deployment and testing strategies", Sections: "Blue/green deployments" and "Canary deployments".

Question: 35

You have an outage in your Compute Engine managed instance group: all instance keep restarting after 5 seconds. You have a health check configured, but autoscaling is disabled. Your colleague, who is a Linux expert, offered to look into the issue. You need to make sure that he can access the VMs. What should you do?

A. Grant your colleague the IAM role of project Viewer

B. Perform a rolling restart on the instance group

C. Disable the health check for the instance group. Add his SSH key to the project-wide SSH keys

D. Disable autoscaling for the instance group. Add his SSH key to the project-wide SSH Keys

Answer:

C

Explanation:

The constant restarting of instances in a Managed Instance Group (MIG) with a health check configured is characteristic of the autohealing feature. The MIG detects that the instances are unhealthy (failing the health check within 5 seconds) and attempts to "heal" the group by recreating them. To stop this restart loop and allow for debugging, the autohealing mechanism must be paused by disabling or removing the health check from the MIG. Once the instances are stable, adding the colleague's public SSH key to the project-wide metadata is a standard and effective method to grant them SSH access to all instances in the project for troubleshooting.

Why Incorrect Options are Wrong:

A. The IAM Viewer role is a read-only role and does not grant the necessary permissions (compute.instances.osLogin or compute.instances.setMetadata) to connect to a VM via SSH.
B. Performing a rolling restart will not solve the issue. The new instances created during the restart will still fail the health check and enter the same restart loop.
D. The question explicitly states that autoscaling is already disabled. The issue is caused by autohealing, which is a separate feature from autoscaling, even though both are part of MIGs.

References:

1. Autohealing and Health Checks: Google Cloud documentation states, "Autohealing relies on a health check to determine if an application on an instance is responding as expected... If the health check determines that an application has failed, the group automatically recreates that instance." To stop this, the health check must be removed.
Source: Google Cloud Documentation, "Setting up health checking and autohealing," Section: "How autohealing works."

2. Updating a Managed Instance Group: To disable the health check, you must update the instance group's configuration. The documentation outlines procedures for updating MIGs, which includes removing an associated health check.
Source: Google Cloud Documentation, "Updating managed instance groups (MIGs)," Section: "Updating a health check for a MIG."
3. Managing SSH Keys: Project-wide public SSH keys can be used to grant access to all instances in a project. "When you add a public SSH key to a project, any user who has the private key can connect to any VM in that project that is configured to accept project-wide keys."
Source: Google Cloud Documentation, "Managing SSH keys in metadata," Section: "Adding and removing project-wide public SSH keys."
4. IAM Roles for Compute Engine: The documentation for the Viewer role (roles/viewer) confirms it does not include permissions like compute.instances.setMetadata or IAP-based access, which are required for SSH connections.
Source: Google Cloud Documentation, "Compute Engine IAM roles," Section: "Project roles."