Google Cloud Professional Cloud Developer Practice Test (Sample)

Study Guide

Everything you need from our exam experts!

Questions

1. Which API architecture is most suitable for minimizing bandwidth costs and integrating easily with mobile apps?

A. RESTful APIs

B. MQTT for APIs

C. gRPC-based APIs

D. SOAP-based APIs

2. What firewall configuration should be implemented to secure access based on specific traffic tags?

A. Block all traffic on port 443

B. Allow all traffic into the network without restrictions

C. Allow traffic on port 443 based on specified tags

D. Permit all traffic on port 443 into the network

3. What is an advantage of using Cloud Functions in serverless architecture?

A. Requires manual scaling

B. Immediate cost with each function execution

C. Automatic scaling based on events

D. Full control of the underlying infrastructure

4. What design aspect is recommended when refactoring a monolithic application into microservices?

A. Develop the microservice code in the same programming language used by the microservice caller.

B. Create an API contract agreement between the microservice implementation and caller.

C. Require synchronous communications between all microservice implementations.

D. Implement a versioning scheme for future changes that could be incompatible.

5. What is a possible reason for a 502 Bad Gateway error when an application cannot connect to Memorystore?

A. The Memorystore instance was deployed without a public IP address.

B. The VPC Access connector is configured in a different region than App Engine.

C. A firewall rule allowing connection between App Engine and Memorystore is missing.

D. The application is using a VPC Access connector on a different subnet.

6. What is the recommended tool to develop applications for Google Kubernetes Engine in a development environment?

A. Use Cloud Code for application development.

B. Use the Cloud Shell integrated Code Editor.

C. Use a Cloud Notebook instance for data processing.

D. Use Cloud Shell to manage infrastructure.

7. You want to share a large read-only data set in a managed instance group ensuring low latency. What is the optimal solution?

A. Move the data to a Cloud Storage bucket and mount using Cloud Storage FUSE.

B. Move the data to a Cloud Storage bucket and copy it to the boot disk via a startup script.

C. Move the data to a Compute Engine persistent disk and attach it in read-only mode to multiple instances.

D. Move the data to a Compute Engine persistent disk, take a snapshot, and create multiple disks from that snapshot.

8. What are the benefits of Migrate for Anthos?

A. All of the above

B. Unified deployment model

C. Unified monitoring model

D. Application density

9. How does the architecture of an application handling orders operate effectively?

A. Multiple instances publish unique orders.

B. The Orders service publishes orders to the Orders topic.

C. Subscribers process orders at a reasonable pace due to buffering.

D. High rates of incoming messages overwhelm the Inventory service.

10. How should logs be structured for maximum efficiency in Google Cloud applications?

A. Use unstructured logs for breadth of data capture.

B. Utilize plain text formats for easy readability.

C. Standardize log formats to JSON for structured analysis.

D. Automatically export logs to a text file for record keeping.

Answers

1. C
2. C
3. C
4. B
5. B
6. A
7. C
8. A
9. A
10. C

Explanations

1. Which API architecture is most suitable for minimizing bandwidth costs and integrating easily with mobile apps?

A. RESTful APIs

B. MQTT for APIs

C. gRPC-based APIs

D. SOAP-based APIs

The selection of gRPC-based APIs is apt for minimizing bandwidth costs and facilitating seamless integration with mobile applications due to several key attributes inherent in gRPC. gRPC, which stands for Google Remote Procedure Call, employs Protocol Buffers for its serialization mechanism. Protocol Buffers are efficient in terms of size and speed, allowing for reduced payload sizes compared to other formats like JSON or XML. This is particularly advantageous in mobile contexts where bandwidth may be limited and usage is closely monitored. Additionally, gRPC operates over HTTP/2, which enhances performance through features such as multiplexing and header compression. Multiplexing allows multiple requests and responses to be sent over a single connection without blocking, which can further reduce latency and improve the responsiveness of mobile applications. The server can also push data to the client, providing real-time updates efficiently, which is highly desirable in mobile applications. Another strength of gRPC is its strong typing and generate support for multiple programming languages, promoting seamless integration and reducing errors during development. Overall, the combination of efficient serialization, reduced latency through HTTP/2, and strong integration capabilities makes gRPC particularly suitable for scenarios involving mobile applications that need to optimize bandwidth costs.

2. What firewall configuration should be implemented to secure access based on specific traffic tags?

A. Block all traffic on port 443

B. Allow all traffic into the network without restrictions

C. Allow traffic on port 443 based on specified tags

D. Permit all traffic on port 443 into the network

Allowing traffic on port 443 based on specified tags is an effective way to secure access, as it gives the ability to manage and filter network traffic according to specific criteria, and only permit communications that meet defined security or operational requirements. This approach utilizes labeled traffic, where each piece of traffic can be categorized with tags that reflect its purpose, source, or any other important aspect deemed relevant for security. By implementing this tagged traffic system, you can ensure that only legitimate and necessary traffic is allowed, which minimizes the potential entry points for malicious activities. This is particularly important for a secure network environment where controlling access is paramount to protecting sensitive data and applications. In contrast, blocking all traffic on a certain port or allowing unrestricted access would not provide the necessary granularity for security management. They can either result in overly restrictive access that may hinder functionality or create vulnerabilities by exposing the network to all types of traffic indiscriminately.

3. What is an advantage of using Cloud Functions in serverless architecture?

A. Requires manual scaling

B. Immediate cost with each function execution

C. Automatic scaling based on events

D. Full control of the underlying infrastructure

Using Cloud Functions in a serverless architecture provides the significant advantage of automatic scaling based on events. This means that the serverless environment can automatically adjust resources to accommodate the number of incoming requests or events without any manual intervention. When a function is triggered, the infrastructure seamlessly scales up to handle the increased load and then scales down when the demand decreases, optimizing resource usage and cost efficiency. This feature is particularly beneficial for applications that experience variable workloads, as it allows developers to focus on writing code without worrying about the underlying infrastructure or scaling challenges. Automatic scaling enables applications to respond quickly to traffic spikes while ensuring that resources are used effectively during quieter periods, thus maintaining performance and cost-effectiveness. In contrast, choices mentioning manual scaling, immediate costs with function execution, and full control of underlying infrastructure point towards limitations or characteristics that are not in alignment with the core benefits of a serverless model, which is designed to abstract away infrastructure management complexities and provide efficient resource utilization.

4. What design aspect is recommended when refactoring a monolithic application into microservices?

A. Develop the microservice code in the same programming language used by the microservice caller.

B. Create an API contract agreement between the microservice implementation and caller.

C. Require synchronous communications between all microservice implementations.

D. Implement a versioning scheme for future changes that could be incompatible.

Creating an API contract agreement between the microservice implementation and its caller is essential in a microservices architecture. This contract serves as a clear, defined communication protocol that outlines how the services will interact with each other. By establishing a contract, developers can ensure that any changes made to one microservice won't inadvertently break functionality in another service that relies on it. This is crucial in maintaining the independence of microservices, as each service can evolve at its own pace without tightly coupling changes to others. Furthermore, an API contract helps to standardize interactions and provides a framework for testing, documentation, and maintaining service boundaries, which are crucial elements in a microservices design. In addition, it allows for the implementation of strategies for versioning and backward compatibility, facilitating smoother integration and deployment processes across the entire system. The other options, while they may seem relevant, do not align as closely with the key principles of microservice design. For instance, developing microservice code in the same programming language can limit flexibility and the advantages of polyglot programming, while requiring synchronous communication can lead to tighter coupling and potential performance bottlenecks. Lastly, while implementing a versioning scheme is important, it often stems from the agreement established in the API contract rather than being a

5. What is a possible reason for a 502 Bad Gateway error when an application cannot connect to Memorystore?

A. The Memorystore instance was deployed without a public IP address.

B. The VPC Access connector is configured in a different region than App Engine.

C. A firewall rule allowing connection between App Engine and Memorystore is missing.

D. The application is using a VPC Access connector on a different subnet.

An application encountering a 502 Bad Gateway error when unable to connect to Memorystore can signify that the application server acting as a gateway or proxy received an invalid response from the upstream server, which in this case is the Memorystore. The chosen reason indicates a probable misconfiguration regarding the geographic regions of the services involved. When the VPC Access connector is configured in a different region than App Engine, it leads to cross-region communication issues. App Engine, when initialized in one region, expects to communicate with services within its own region or through properly configured networking elements. If the VPC Access connector is not in the same region, it can't successfully route requests to Memorystore, resulting in the application being unable to connect and hence triggering a 502 error. Having the VPC Access connector and App Engine in different regions creates a mismatch, leading to connection timeouts or failures. This underscores the importance of ensuring that all components that require low-latency communication are deployed within the same geographical region. The other reasons focus on different aspects, such as deployment configuration or firewall rules, which also could lead to connectivity issues, but the regional mismatch specifically targets the root cause of why the application cannot reach Memorystore, thus resulting in a 502 Bad Gateway error

6. What is the recommended tool to develop applications for Google Kubernetes Engine in a development environment?

A. Use Cloud Code for application development.

B. Use the Cloud Shell integrated Code Editor.

C. Use a Cloud Notebook instance for data processing.

D. Use Cloud Shell to manage infrastructure.

Using Cloud Code for application development is the recommended tool for developing applications within Google Kubernetes Engine (GKE) in a development environment. Cloud Code enhances the development process by providing integrated tools that allow developers to build, debug, and deploy applications directly to GKE with ease. Cloud Code supports a range of programming languages and frameworks, offering features such as smart code completion, real-time Kubernetes resource management, and integrated debugging tools that streamline the development workflow. By leveraging Cloud Code, developers can benefit from a meaningful context-aware development experience, making it simpler to manage Kubernetes configurations and work with cloud-native patterns. In contrast, other tools may not focus specifically on the nuances of Kubernetes application development. The Cloud Shell integrated Code Editor allows basic code editing and command-line operations but lacks the specialized features for GKE applications. A Cloud Notebook instance is better suited for machine learning and analytics tasks rather than typical application development, and while Cloud Shell is useful for infrastructure management commands, it does not combine those functionalities with application-centric development capabilities.

7. You want to share a large read-only data set in a managed instance group ensuring low latency. What is the optimal solution?

A. Move the data to a Cloud Storage bucket and mount using Cloud Storage FUSE.

B. Move the data to a Cloud Storage bucket and copy it to the boot disk via a startup script.

C. Move the data to a Compute Engine persistent disk and attach it in read-only mode to multiple instances.

D. Move the data to a Compute Engine persistent disk, take a snapshot, and create multiple disks from that snapshot.

The optimal solution utilizes a Compute Engine persistent disk attached in read-only mode to multiple instances because it provides a managed and reliable way to share a large data set among multiple instances in a managed instance group while ensuring low latency. By using a persistent disk, multiple instances can access the same disk simultaneously without needing to deal with network latency associated with accessing data over the internet or mount points. When the persistent disk is set to read-only mode, it guarantees that the data integrity remains intact, as no instance can modify the data. This approach maintains performance and reduces complexity since the persistent disk can easily scale with the instances without requiring additional configurations for access management. Additionally, persistent disks in Google Cloud are designed for high performance, allowing instances to access the data efficiently. This is particularly crucial when dealing with large data sets, as it facilitates faster read operations compared to other methods that may involve some form of data copying or network access. In contrast, while other options might provide ways to access data, they may introduce latency issues and additional management overhead that is less efficient for sharing large datasets across multiple compute instances.

8. What are the benefits of Migrate for Anthos?

A. All of the above

B. Unified deployment model

C. Unified monitoring model

D. Application density

Migrate for Anthos offers a variety of benefits that support the modernization and migration of applications to cloud-native environments. Selecting "All of the above" signifies recognition of the comprehensive advantages provided by the individual features listed. The unified deployment model allows organizations to seamlessly deploy applications across both on-premises and cloud environments. This consistency in deployment simplifies the management of applications and enables developers to leverage a unified toolset, ensuring that applications run reliably regardless of where they are hosted. The unified monitoring model is another critical feature that enhances application management and observability. It enables organizations to monitor their applications consistently across different environments. This serves as a foundation for identifying performance issues and maintaining reliability, thus ensuring that applications meet business requirements and user expectations. Application density is a notable benefit as well, which refers to the ability to effectively utilize resources by running multiple applications on the same infrastructure. This optimizes resource usage, reduces costs, and can lead to improved performance. By promoting efficient application deployment and resource management, Migrate for Anthos empowers businesses to scale effectively. In summary, selecting "All of the above" accurately reflects an understanding that Migrate for Anthos provides a robust set of capabilities, which include a unified deployment model, a unified monitoring model, and improved application

9. How does the architecture of an application handling orders operate effectively?

A. Multiple instances publish unique orders.

B. The Orders service publishes orders to the Orders topic.

C. Subscribers process orders at a reasonable pace due to buffering.

D. High rates of incoming messages overwhelm the Inventory service.

In the architecture of an application handling orders, the effective operation is characterized by multiple instances able to publish unique orders. This approach allows for horizontal scalability, meaning that as the volume of orders increases, more instances can be added to handle the additional load. This design principle is crucial in distributed systems where high availability and resilience are essential. By enabling multiple instances to publish unique orders, the system can better manage concurrent order submissions. This facilitates load balancing and ensures that no single instance becomes a bottleneck, which could lead to delays or failures in order processing. As a result, the entire system can more efficiently process orders, improving overall performance and user satisfaction. While other aspects of the options can contribute to the efficiency of an order processing architecture, the emphasis on multiple instances publishing orders highlights the importance of scalability and distribution in modern cloud-based applications. This ensures that the application can adapt to varying loads without compromising on speed or reliability.

10. How should logs be structured for maximum efficiency in Google Cloud applications?

A. Use unstructured logs for breadth of data capture.

B. Utilize plain text formats for easy readability.

C. Standardize log formats to JSON for structured analysis.

D. Automatically export logs to a text file for record keeping.

Standardizing log formats to JSON is the optimal approach for achieving maximum efficiency in Google Cloud applications. JSON (JavaScript Object Notation) is a lightweight data interchange format that is easy for machines to parse and generate. This structured format allows for better organization of log data, making it easier to extract specific fields and perform queries on them. When logs are in a structured format like JSON, developers and operations teams can leverage log analysis tools that can efficiently query and analyze the logs. This can significantly speed up troubleshooting and performance monitoring as developers can quickly identify patterns or issues within the logs. In contrast, unstructured logs lack specific format and organization, which makes it difficult to systematically analyze the data. While plain text formats are human-readable, they do not facilitate automated analysis and might require extra parsing to derive meaningful insights. Exporting logs to a text file does not inherently provide a structured format and can lead to challenges in data retrieval and analysis. Overall, using JSON as a standard log format enhances usability, supports better tooling capabilities, and enables efficient querying and debugging processes in cloud environments.