Free Questions for 1Z0-1084-23 by dumpssheet

Shared by Stark on 09-08-2024

For More Free Questions and Preparation Resources

Check the Links on Last Page

Question 1

Question Type: MultipleChoice

You have two microservices, A and B running in production. Service A relies on APIs from service B. You want to test changes to service A without deploying all of its dependencies, which includes service B. Which approach should you take to test service A?

Options:

A- Test using a previous test version of service B.

B- Test using an API mock of service B.

C- Test using the current production version of service B.

D- This is not possible because service B is a dependency.

Answer:

B

Explanation:

The correct answer is: Test using an API mock of service B. To test service A without deploying all of its dependencies, including service B, you can use an API mock of service B. An API mock is a simulated version of the API that mimics the behavior of the actual service.

By using an API mock, you can isolate the testing of service A and simulate the responses and behavior of service B's APIs. With an API mock, you can define the expected responses and behavior of service B's APIs, allowing you to test the integration between service A and the mocked version of service B. This approach enables you to verify the functionality of service A without relying on the availability or changes in the actual service B. By decoupling the dependencies and using an API mock, you can perform independent testing of service A, ensuring its functionality in isolation.

Question 2

Question Type: MultipleChoice

With the volume of communication that can happen between different components in cloud-native applications, it is vital to not only test functionality, but also service resiliency. Which statement is true regarding service resiliency?

Options:

A- Resiliency is about avoiding failures.

B- Resiliency testing can be done only in a test environment.

C- Resiliency is about recovering from failures without downtime or data loss.

D- Resiliency is about not bringing a service to a functioning state after a failure.

Answer:

C

Explanation:

The correct answer is: 'Resiliency is about recovering from failures without downtime or data loss.' Service resiliency, in the context of cloud-native applications, is the ability of a service or system to recover from failures and continue functioning without downtime or data loss. It involves designing and implementing mechanisms to handle failures, such as network outages, hardware failures, or software errors, in a way that minimizes the impact on the overall system. The goal of resiliency is to ensure that the application or service can continue to operate and provide a certain level of functionality, even in the face of failures. This typically involves techniques such as redundancy, fault tolerance, and graceful degradation. By implementing resiliency measures, a cloud-native application can recover and adapt to failures, maintain availability, and preserve data integrity. The other statements are not accurate regarding service resiliency: Resiliency is not about not bringing a service to a functioning state after a failure. Instead, it is about recovering from failures and ensuring continued functionality. Resiliency is not about avoiding failures entirely. While it is desirable to prevent failures, resiliency focuses on the ability to handle and recover from failures when they do occur. Resiliency testing is not limited to a test environment. It is important to test and validate the resiliency measures in both test environments and production environments to ensure the application can effectively handle failures in real-world scenarios.

Question 3

Question Type: MultipleChoice

To enforce mutual TLS (mTLS) authentication for clients of your microservices, your team has chosen to leverage the Oracle Cloud Infrastructure (OCI) API Gateway service to create new API Deployments that will direct requests to your microservices. Which is NOT valid regarding the mTLS options in OCI API Gateway?

Options:

A- Custom CA or custom CA bundles can be added to your gateway's trust store ONLY if they already exist in the OCI Certificates service.

B- Adding a custom certificate authority (CA) or custom CA bundle to your gateway's trust store for mTLS is optional unless you need to reject certificates that do not contain particular values (such as a domain name).

C- The mTLS request policy can only be enabled at the API deployment specification level, which then applies globally to ALL routes in that deployment.

D- Once the mTLS request policy is enabled, ALL requests with valid certificates are routed to the backend unless you have defined one or more particular values (such as a domain name).

Answer:

B

Explanation:

The correct answer is: 'Adding a custom certificate authority (CA) or custom CA bundle to your gateway's trust store for mTLS is optional unless you need to reject certificates that do not contain particular values (such as a domain name).' The statement that is NOT valid regarding the mTLS options in OCI API Gateway is: 'Adding a custom certificate authority (CA) or custom CA bundle to your gateway's trust store for mTLS is optional unless you need to reject certificates that do not contain particular values (such as a domain name).' In OCI API Gateway, adding a custom certificate authority (CA) or custom CA bundle to the gateway's trust store is not optional. It is a necessary step when configuring mTLS authentication. The trust store in the gateway is used to validate the client certificates presented during mTLS authentication. The other options listed are valid regarding the mTLS options in OCI API Gateway: Once the mTLS request policy is enabled, all requests with valid certificates are routed to the backend unless specific values (such as a domain name) are defined. This means that only requests with valid client certificates will be allowed to access the backend microservices. The mTLS request policy can only be enabled at the API deployment specification level, and it applies globally to all routes in that deployment. This ensures consistent mTLS authentication across all routes and endpoints in the API deployment. Custom CA or custom CA bundles can be added to the gateway's trust store, but only if they already exist in the OCI Certificates service. This allows you to include trusted CAs or CA bundles to validate client certificates during mTLS authentication.

Question 4

Question Type: **MultipleChoice**

A developer using Oracle Cloud Infrastructure (OCI) API Gateway needs to authenticate the API requests to their web application. The authentication process must be implemented using a custom scheme which accepts string-based parameters from the API caller. Which approach should the developer use in this scenario?

Options:

A- Create a cross account functions authorizer.

B- Create an authorizer function using OCI Identity and Access Management 91AM) based authentication.

C- Create an authorizer function using request header authorization.

D- Create an authorizer function using token-based authorization.

Answer:

D

Explanation:

In the given scenario, the developer should use the approach of creating an authorizer function using token-based authorization. Token-based authorization is a commonly used approach for authenticating API requests. It involves generating and issuing tokens to API callers, which they can then include in the requests they make to the API. The tokens serve as proof of authentication and are validated by the server to ensure the caller's identity and access rights. By creating an authorizer function using token-based authorization, the developer can implement a custom scheme that accepts string-based parameters from the API caller. This allows the developer to define their own authentication logic and validate the provided tokens according to their requirements. The authorizer function can be configured in the OCI API Gateway to be invoked before forwarding the request to the web application. It will perform the necessary token validation and authentication checks, allowing only authorized requests to access the protected resources of the web application.

Question 5

Question Type: MultipleChoice

Which TWO statements accurately describe an Oracle Functions application? (Choose two.)

Options:

A- A common context to store configuration variables that are available to all functions in the application. A Docker image containing all the functions that share the same configuration.

B- An application based on Oracle Functions, Oracle Cloud Infrastructure (OCI) Events, and OCI API Gateway services.

C- A small block of code invoked in response to an OCI Events service.
A logical group of functions.

D- A Docker image containing all the functions that share the same configuration.

Answer:

A, C

Explanation:

The correct statements are: A common context to store configuration variables that are available to all functions in the application. A Docker image containing all the functions that share the same configuration. A logical group of functions. An Oracle Functions application provides a common context for functions within the application. It allows you to store configuration variables that are accessible by all the functions in the application. Functions within the same application can share the same Docker image, which contains the common configuration and dependencies. An Oracle Functions application serves as a logical group that organizes related functions. Functions within the same application can be managed collectively, and they can interact and share resources within the application context.

Question 6

Question Type: MultipleChoice

Which TWO are required to access the Oracle Cloud Infrastructure (OCI) Container Engine for Kubernetes (OKE) cluster from the kubectl CLI? (Choose two.)

Options:

A- Tiller enabled on the OKE cluster.

B- An SSH key pair with the public key added to the cluster worker nodes.

C- Install and configure the OCI CLI.

D- A configured OCI API signing key pair.

E- OCI Identity and Access Management (IAM) Auth Token.

Answer:

C, D

Explanation:

The correct options are: A configured OCI API signing key pair: The API signing key pair is used for authentication and authorization to access OCI resources, including the OKE cluster. The private key should be configured on your local machine to authenticate API requests. An SSH key pair with the public key added to the cluster worker nodes: This is required for secure SSH access to the worker nodes in the OKE cluster. You need to generate an SSH key pair and add the public key to the cluster's worker node pool during cluster creation or update. Therefore, the correct options are having a configured OCI API signing key pair and an SSH key pair with the public key added to the cluster worker nodes.

Question 7

Question Type: MultipleChoice

Oracle Functions monitors all deployed functions and collects and reports various metrics. Which is NOT available when viewing the Application metrics in the Oracle Cloud Infrastructure (OCI) Console?

Options:

A- The length of time a function runs for.

B- The number of retries made by the function before failing due to an error.

C- The number of requests to invoke a function that failed due to throttling.

D- The number of requests to invoke a function that failed with an error response.

Answer:

B

Explanation:

The option that is NOT available when viewing the Application metrics in the Oracle Cloud Infrastructure (OCI) Console is: 'The number of retries made by the function before failing due to an error.' When viewing the Application metrics in the OCI Console for Oracle Functions, you can typically see metrics related to the performance and usage of your functions. These metrics provide insights into how your functions are performing and being utilized. The following metrics are usually available: The number of requests to invoke a function that failed due to throttling: This metric indicates the number of requests that were not processed by the function due to reaching the configured concurrency limit or throttling settings. The length of time a function runs for: This metric represents the duration of each function invocation, measuring the time it takes for the function to complete its execution. The number of requests to invoke a function

that failed with an error response: This metric counts the number of requests that encountered an error during the function invocation, resulting in a failed response. However, the number of retries made by the function before failing due to an error is not typically available as part of the Application metrics in the OCI Console. The retries made by the function are usually handled at the invoker level, and the specific details of retries may not be captured as part of the application-level metrics. It's important to note that the availability of metrics and their specific details may vary depending on the version and configuration of Oracle Functions and the monitoring setup. It is recommended to refer to the Oracle Functions documentation and consult the official documentation for accurate and up-to-date information on available metrics.

Question 8

Question Type: MultipleChoice

A Docker image consists of one or more layers, each of which represents a Dockerfile instruction. The layers are stacked and each one is a delta of the changes from the previous layer. What permission is associated with these layers?

Options:

A- read mostly

B- write only

C- movable

D- read only

E- write once

Answer:

D

Explanation:

The correct answer is: 'read only.' The layers of a Docker image are read-only. Once a layer is created, it cannot be modified. Each layer represents a Dockerfile instruction, and it is stacked on top of the previous layer, forming a stack of immutable layers. These layers are designed to be read-only to ensure consistency and integrity of the image. When a Docker image is built, each instruction in the Dockerfile creates a new layer. Each layer represents the changes made by that instruction relative to the previous layer. The layers are stacked on top of each other to form the complete image. This layer-based approach allows for efficient storage and distribution of Docker images. Because the layers are read-only, any changes or modifications to the image result in the creation of new layers rather than modifying the existing ones. This immutability ensures that each layer remains intact and preserves the integrity of the image. It also enables Docker's caching mechanism, where previously built layers can be reused if the corresponding instructions haven't changed, speeding up the image build process. The other options mentioned, such as 'write only,' 'write once,' 'movable,' and 'read mostly,' do not accurately describe the permission associated with Docker image layers. Docker image layers are specifically designed to be read-only.

Question 9

Question Type: MultipleChoice

Your team has created a serverless application deployed in Oracle Functions. It uses a Python function leveraging the Oracle Cloud Infrastructure (OCI) Python SDK to stop any OCI compute instance that does not comply with your corporate security standards. Although there are three non-compliant OCI compute instances, when you invoke this function, none of the instances were stopped. With respect to this issue, which of the following is a valid troubleshooting strategy?

Options:

A- Enable function logging in the OCI console, add some print statements in your function code, and then view the logs to troubleshoot.

B- Enable function remote debugging in the OCI console, and then use your favorite IDE to inspect the function running on Oracle Functions.

C- Ensure that the application is deployed within the same OCI compartment as the instance, because you cannot enable function execution data from the OCI console.

D- Enable function tracing in the OCI console, and then go to the OCI Monitoring console to view the function stack trace.

Answer:

A

Explanation:

The valid troubleshooting strategy in this scenario is to enable function logging in the OCI console, add some print statements in your function code, and then view the logs to troubleshoot. Enabling function logging allows you to capture and store logs generated by your function during its execution. By adding print statements or log statements in your function code, you can output relevant information and debug messages to the logs. This helps you understand the execution flow, identify any errors or issues, and gather more information about the function's behavior. To troubleshoot the issue of the Python function not stopping the non-compliant OCI compute instances, you can follow these steps: Enable function logging in the OCI console: Enable logging for your function to ensure that logs are captured during its execution. Modify your function code: Add relevant print statements or log statements at key points in your code to output debug information or verify the execution flow. For example, you can print the instance details that are being evaluated for compliance. Invoke the function: Trigger the function execution either through an event or manually. View the logs: Access the function logs in the OCI console or retrieve them programmatically. Look for the expected print statements or log entries that indicate the status of each instance and the decisions made by the function. By reviewing the logs, you can analyze the output and identify any issues or discrepancies. It can help you determine if the function is correctly evaluating the compliance criteria, retrieving the instance details, or making the necessary API calls to stop the instances. You may need to adjust your code logic or investigate further based on the information provided in the logs. Enabling function remote debugging is not a suitable strategy in this case because it is primarily used for inspecting and debugging the function code during development, rather than troubleshooting issues in a deployed function. Enabling function tracing can provide insights into the execution flow and performance of the function but may not directly address the issue of the instances not being stopped. Ensuring that the application is deployed within the same OCI compartment as the instance is not directly related to troubleshooting the issue with the non-compliant instances. It is a consideration for access and permissions but does not provide specific insights into the problem at hand. Remember to refer to the Oracle Functions documentation and consult the official resources for detailed instructions and best practices on troubleshooting and monitoring Oracle Functions.

Question 10

Question Type: MultipleChoice

Your organization has deployed their e-commerce application on Oracle Container Engine for Kubernetes (OKE) and they are using the Oracle Cloud Infrastructure Registry (OCIR) service as their Docker image repository. They have deployed the OKE cluster using the 'custom create' option, and their Virtual Cloud Network (VCN) has three public subnets with associated Route Tables, Security Lists, and Internet Gateway. However, their application containers are failing to deploy. On investigation, they discover that the images are not being pulled from the designated OCIR repository, even though the YAML configuration has the correct path to the images. What is a valid concern here that needs to be further investigated?

Options:

A- Security List rule for TCP port 22 needs to be added to connect to the OCIR service.

B- VCN hosting the OKE cluster worker nodes needs to have a NAT gateway to access OCIR repositories.

C- Identity and Access Management (IAM) credentials need to be added for each user that deploys applications to the OKE cluster.

D- OKE cluster needs to have a secret with the credentials of their OCIR repository and use that secret in the Kubernetes deployment manifest.

Answer:

D

Explanation:

A valid concern that needs to be further investigated in this scenario is whether the OKE cluster has a secret with the credentials of the Oracle Cloud Infrastructure Registry (OCIR) repository and if that secret is being used in the Kubernetes deployment manifest. Here's why this concern is relevant: Access to the OCIR repository: In order for the OKE cluster to pull images from the OCIR repository, it needs proper authentication credentials. These credentials are typically provided in the form of a secret, which contains the necessary information to authenticate with the registry. Secret in the deployment manifest: The Kubernetes deployment manifest defines how the application containers should be deployed. It includes specifications such as the container image, resource requirements, and environment variables. To pull images from a private repository like OCIR, the deployment manifest needs to reference the appropriate secret that contains the registry credentials. If the images are not being pulled from the designated OCIR repository, it suggests that either the secret with the OCIR credentials is missing or it is not properly referenced in the deployment manifest. Further investigation should focus on verifying the presence and correctness of the secret, as well as confirming that it is correctly referenced in the deployment manifest for the application containers. By ensuring the presence of the secret and proper configuration in the deployment manifest, the OKE cluster will have the necessary credentials to access the OCIR repository and successfully deploy the application containers.

Question 11

Question Type: MultipleChoice

You are building a container image and pushing it to Oracle Cloud Infrastructure Registry (OCIR). You need to ensure that these images never get deleted from the repository. Which action should you take?

Options:

A- Write a policy to limit access to the specific repository in your compartment.

B- Create a group and assign a policy to perform lifecycle operations on images.

C- Set global policy of image retention to 'Retain All Images'.

D- Edit the tenancy global retention policy.

Answer:

D

Explanation:

The correct answer is: 'Edit the tenancy global retention policy.' To ensure that container images never get deleted from the Oracle Cloud Infrastructure Registry (OCIR), you should edit the tenancy global retention policy. The tenancy global retention policy is a setting that determines the retention behavior for all the images in the OCIR across the entire tenancy. By editing this policy, you can define the retention behavior that suits your requirements. To edit the tenancy global retention policy, you would typically perform the following steps: Access the Oracle Cloud Infrastructure Console and navigate to the OCIR service. Go to the 'Policies' section or 'Settings' section in the OCIR service. Locate the tenancy global retention policy settings. Modify the retention policy to specify the desired retention behavior. In this case, you would set the policy to retain all images, ensuring they are never deleted from the repository. By setting the global policy of image retention to 'Retain All Images,' you can ensure that the container images in your OCIR repository are permanently retained and not subject to deletion based on any default or automatic retention rules. The other options mentioned are not

directly related to ensuring that container images are never deleted from the repository: Creating a group and assigning a policy to perform lifecycle operations on images or writing a policy to limit access to the specific repository in your compartment are access control measures and do not address the retention of images. Setting the global policy of image retention to 'Retain All Images' is the correct action to achieve the desired outcome of preventing image deletion from the repository.