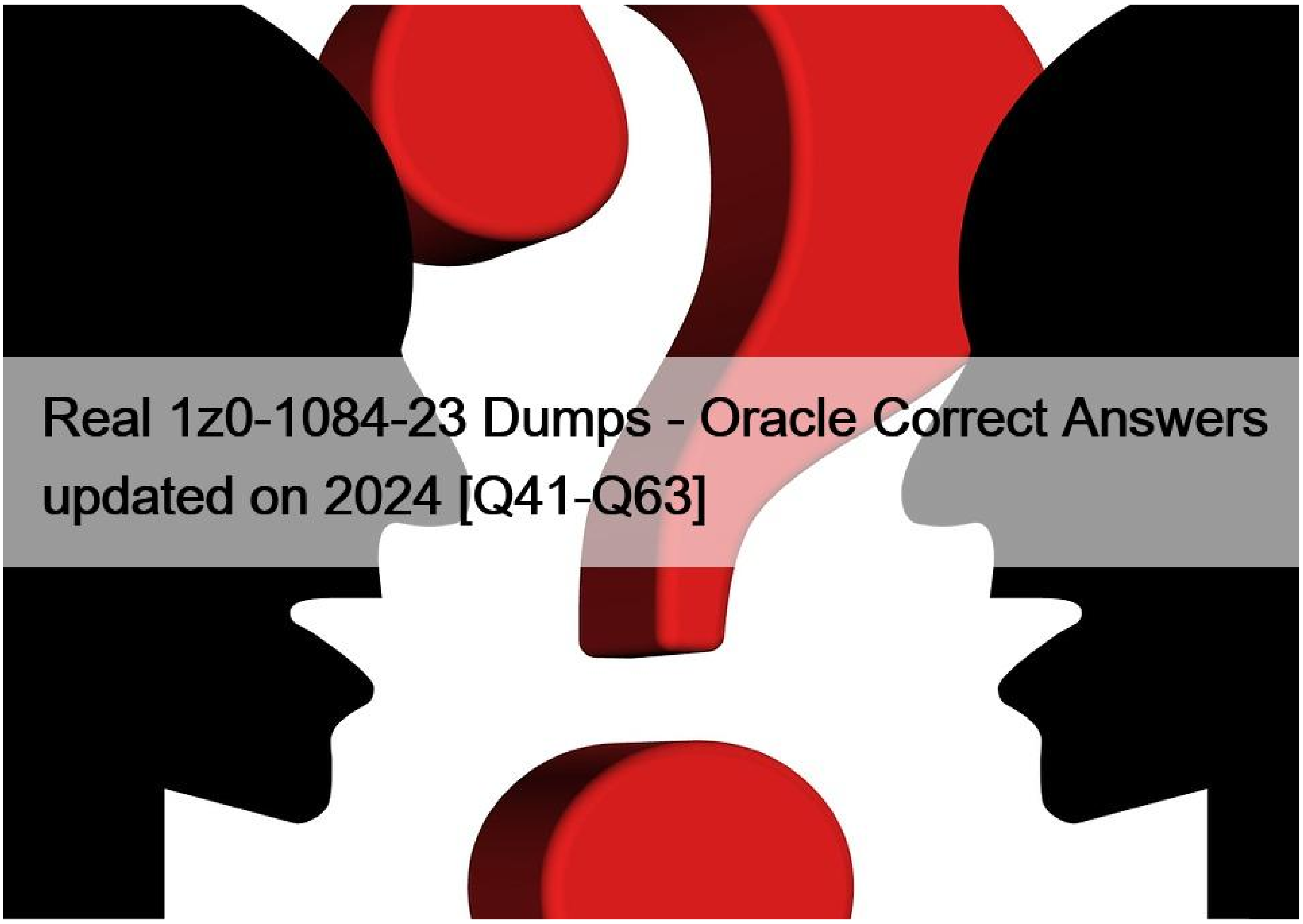
Real 1z0-1084-23 Dumps - Oracle Correct Answers updated on 2024 [Q41-Q63



Use Real 1z0-1084-23 Dumps - Oracle Correct Answers updated on 2024
Oracle Cloud 1z0-1084-23 Exam Practice Dumps

NO.41 You have just finished building and compiling the software required to implement the API microservice component. You need to rebuild the API docker image, and plan to tag it as: ocIdevops/api:latest Which docker command would re-create the API docker image?
* docker build -t OCIdevops/api:latest
* docker create -t OCIdevops/api:latest
* docker image -t OCIdevops/api:latest
* docker compile -t OCI devops/api:latest

The correct command to rebuild the API docker image and tag it as OCIdevops/api:latest is: docker build -t OCIdevops/api:latest The docker build command is used to build a Docker image from a Dockerfile. The -t flag is used to specify the name and optionally a tag for the image. In this case, the name of the image is OCIdevops/api and the tag is latest. By running this command, the Docker image will be recreated based on the instructions in the Dockerfile and tagged with the specified name and tag.

NO.42 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 &#8216;custom create&#8217; 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?
* Security List rule for TCP port 22 needs to be added to connect to the OCIR service.
* VCN hosting the OKE cluster worker nodes needs to have a NAT gateway to access OCIR repositories.
* Identity and Access Management (IAM) credentials need to be added for each user that deploys applications to the OKE cluster.
* OKE cluster needs to have a secret with the credentials of their OCIR repository and use that secret in the Kubernetes deployment manifest.
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&#8217;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.

NO.43 Which of the following step is NOT required for setting up the Container Engine for Kubernetes (OKE) cluster access using a local installation of kubectl?
* Set up the kubeconfig file.
* Install and configure the Oracle Cloud Infrastructure (OCI) CLI.
* Generate an API signing key pair (if you do not already have one) and upload the public key of the API signing key pair.
* Generate Auth token from the OCI console to access the OKE cluster using kubectl.
Explanation

The step that is NOT required for setting up the Container Engine for Kubernetes (OKE) cluster access using a local installation of kubectl is to generate an Auth token from the OCI console. The authentication for accessing the OKE cluster using kubectl can be performed using the OCI CLI configuration, specifically the API signing key pair and the kubeconfig file. Here are the correct steps for setting up the OKE cluster access using a local installation of kubectl: Set up the kubeconfig file: The kubeconfig file contains the necessary information to authenticate and access the OKE cluster using kubectl. It includes details such as the cluster endpoint, authentication method, and credentials. Generate an API signing key pair (if you do not already have one) and upload the public key of the API signing key pair: The API signing key pair is used for authentication with the OCI services. The public key of the key pair needs to be uploaded to the OCI Console to associate it with your user account. Install and configure the Oracle Cloud Infrastructure (OCI) CLI: The OCI CLI provides a command-line interface to interact with the OCI services. It needs to be installed and configured with your OCI credentials, including the user&#8217;s OCID, tenancy OCID, region, and the path to the API signing key pair. By completing these steps,you can configure kubectl to access and manage your OKE clusters from your local machine using the OCI CLI authentication configuration.

NO.44 (CHK_4>3) Your development team decides to create and deploy some business logic to serverless Oracle Functions. You are asked to help facilitate the monitoring, logging, and tracing of these services. Which is NOT valid about troubleshooting Oracle Functions?
* Oracle Functions invocation is enabled by default
* Oracle Functions invocation logs are enabled at the application level.

* Oracle Functions tracing is enabled at the function level.
* Oracle Functions metrics are available at both the function and application level.
Explanation

The option that is NOT valid about troubleshooting Oracle Functions is: &#8220;Oracle Functions tracing is enabled at the function level.&#8221; In Oracle Functions, tracing is not enabled at the function level. Instead, tracing is enabled at the application level. When you enable tracing for an application, it applies to all the functions within that application. Tracing allows you to capture detailedinformation about the execution flow and performance of the functions, helping you analyze and debug issues. The other options mentioned are valid:

Oracle Functions invocation logs are enabled at the application level. Invocation logs provide visibility into the details of function invocations, including input, output, duration, and any error messages. These logs are generated and stored by Oracle Functions, and you can access them for troubleshooting and monitoring purposes. Oracle Functions invocation is enabled by default. Once you deploy a function, it becomes invocable by default. You can configure different triggers to invoke the function, such as HTTP requests, scheduled events, or events from other Oracle Cloud Infrastructure services. Oracle Functions metrics are available at both the function and application level. Metrics provide insights into the usage, performance, and behavior of functions. They can include metrics such as invocations per minute, average duration, and error counts. These metrics can be viewed in the Oracle Cloud Infrastructure Console or accessed programmatically through APIs. It&#8217;s important to note that the specific configuration and behavior of monitoring, logging, and tracing in Oracle Functions may depend on the version, configuration, and options you have chosen. It is recommended to refer to the Oracle Functions documentation and consult the official documentation for accurate and up-to-date information on troubleshooting and monitoring Oracle Functions.

NO.45 You developed a microservices-based application that runs in an Oracle Cloud Infrastructure (OCI) Container Engine for Kubernetes (OKE) cluster. Your security team wants to use SSL termination for this application.

What should you do to create a secure SSL termination for this application using the fewest steps possible?
* Create a self-signed certificate and its corresponding key. Create a Kubernetes secret using the certificate and the key. Then add these annotations to the Kubernetes service: annotations:

service.beta.kubernetes.io/oci-load-balancer-ssl-ports: &#8220;443&#8221;service.beta.kubernetes.io/oci-load-balancer-tls-secret: ssl certificate-secret
* Create a self-signed certificate and its corresponding key. Create a Kubernetes secret using the certificate and the key. Then add these annotations to the Kubernetes service: annotations:

service.beta.kubernetes.io/oci-load-balancer-ssl-ports: &#8220;443&#8221;

service.beta.kubernetes.io/oci-load-balancer-security-list management-mode: &#8220;Frontend&#8221;
* Add these annotations to the kubernetes service: annotations:

service.beta.kubernetes.io/oci-load-balancer-ssl-ports: &#8220;443&#8221;

service.beta.kubernetes.io/oci-load-balancer-ssl-secret-key: ssl secret-key
* Generate a self-signed certificate using Let&#8217;s Encrypt. Use that certificate on OCI Load Balancer. Create the Kubernetes service using this load balancer.
Explanation

The correct answer is: &#8220;Create a self-signed certificate and its corresponding key. Create a Kubernetes secret using the certificate and the key. Then add these annotations to the Kubernetes service: annotations:

service.beta.kubernetes.io/oci-load-balancer-ssl-ports: &#8216;443&#8217;

service.beta.kubernetes.io/oci-load-balancer-tls-secret: ssl certificate-secret.&#8221; To create a secure SSL termination for your microservices-based application running in an OCI Container Engine for Kubernetes (OKE) cluster, you can follow these steps: Create a self-signed certificate and its corresponding key: Generate a self-signed SSL certificate and its private key using a tool like OpenSSL. Create a Kubernetes secret: Create a Kubernetes secret using the certificate and key obtained in the previous step. This secret will securely store the certificate and key within the Kubernetes cluster. Add annotations to the Kubernetes service: Modify the Kubernetes service that exposes your application and add the following annotations to enable SSL termination:

annotations: service.beta.kubernetes.io/oci-load-balancer-ssl-ports: &#8216;443&#8217; (specify the SSL port as 443) annotations: service.beta.kubernetes.io/oci-load-balancer-tls-secret: ssl certificate-secret (specify the name of the Kubernetes secret containing the certificate and key) By following these steps, you can create a secure SSL termination for your application using a self-signed certificate and Kubernetes secret. The annotations added to the Kubernetes service ensure that the SSL port is configured correctly and the TLS secret is utilized for SSL termination when traffic reaches the load balancer. The other options provided are not the most suitable approaches for achieving secure SSLtermination in an OCI Container Engine for Kubernetes (OKE) cluster:

Adding annotations related to the OCI load balancer SSL secret key is not the correct approach for SSL termination in this scenario. Using Let&#8217;s Encrypt to generate a self-signed certificate and configuring it on the OCI Load Balancer is not necessary when you can create and manage the SSL certificate within the Kubernetes cluster using a Kubernetes secret.

NO.46 You are developing a distributed application and you need a call to a path to always return a specific JSON content deploy an OCI API Gateway with the below API deployment specification. What is the correct value for type? { &#8220;routes&#8221; : [{ &#8220;path&#8221; : &#8220;/hello&#8221;, &#8220;methods&#8221; : [&#8220;Get&#8221;), &#8220;backend&#8221; : { &#8220;type&#8221; : &#8221; &#8212;&#8212;&#8212;&#8212;&#8212;- &#8220;, &#8220;status&#8221;: 200, &#8220;headers&#8221; : [{ &#8220;name&#8221; : &#8220;Content-Type&#8221;, &#8220;value&#8221; : &#8220;application/json&#8221; }] &#8220;body&#8221; : &#8220;{&#8220;myjson&#8221;:

&#8220;consistent response&#8221;}&#8221; }}]}
* JSON_BACKEND
* STOCK_RESPONSE_BACKEND
* CONSTANT_BACKEND
* HTTP_BACKEND
Explanation

The correct value for the &#8220;type&#8221; field in the API deployment specification is

&#8220;STOCK_RESPONSE_BACKEND&#8221;. By setting the &#8220;type&#8221; to &#8220;STOCK_RESPONSE_BACKEND&#8221;, you are indicating that the backend for the specified route should return a pre-defined response. This type of backend is commonly used when you want a specific response to be returned consistently, regardless of the actual backend service implementation. In this case, the API deployment specification is configured to have a single route with the path &#8220;/hello&#8221; and the method &#8220;GET&#8221;. The backend section specifies the type as

&#8220;STOCK_RESPONSE_BACKEND&#8221;. Additionally, it defines the response status code as 200, setsthe

&#8220;Content-Type&#8221; header to &#8220;application/json&#8221;, and provides the JSON content in the &#8220;body&#8221; field. Using this configuration, any request to the &#8220;/hello&#8221; path with the &#8220;GET&#8221; method will always receive a consistent JSON response with the content &#8220;{&#8220;myjson&#8221;: &#8220;consistent response&#8221;}&#8221;.

NO.47 You have a containerized application that requires access to an Autonomous Transaction Processing (ATP) Database. Which option is NOT valid when the container is deployed in an OKE cluster? (Choose the best answer.)

* Use Kubernetes secrets to configure environment variables on the container with ATP instance OCID, and OCI API credentials. Then use the CreateConnection API endpoint from the service runtime.
* Create a Kubernetes secret with contents from the instance Wallet files. Use this secret to create a volume mounted to the appropriate path in the application deployment manifest.
* Install the Oracle Cloud Infrastructure Service Broker on the Kubernetes cluster and deploy ServiceInstance and ServiceBinding resources for ATP. Then use the specified binding name as a volume in the application deployment manifest.
* Enable Oracle REST Data Services for the required schemas and connect via HTTPS.

Explanation

The option that is not valid for connecting to an Autonomous Transaction Processing (ATP) Database from a container in Kubernetes is: Install the Oracle Cloud Infrastructure Service Broker on the Kubernetes cluster and deploy ServiceInstance and ServiceBinding resources for ATP. Then use the specified binding name as a volume in the application deployment manifest. The Oracle Cloud Infrastructure Service Broker is not used for connecting to an ATP Database from a container in Kubernetes. The Service Broker is used for provisioning and managing cloud services directly from Kubernetes. It allows you to create and manage instances of OCI services using Kubernetesresources like ServiceInstance and ServiceBinding. To connect to an ATP Database from a container in Kubernetes, you can use one of the following valid options: Enable Oracle REST Data Services for the required schemas and connect via HTTPS. This involves enabling and configuring Oracle REST Data Services (ORDS) for the schemas in the ATP Database. You can then connect to the ATP Database using RESTful endpoints provided by ORDS. Use Kubernetes secrets to configure environment variables on the container with ATP instance OCID and OCI API credentials. Then use the CreateConnection API endpoint from the service runtime. This approach involves configuring the necessary environment variables on the container to provide the ATP instance OCID and OCI API credentials. The application can then use the OCI SDK or REST API (such as the CreateConnection endpoint) to establish a connection to the ATP Database. Create a Kubernetes secret with contents from the instance Wallet files. Use this secret to create a volume mounted to the appropriate path in the application deployment manifest. This method involves creating a Kubernetes secret that contains the necessary credentials from the ATP Database&#8217;s instance wallet files. The secret can then be mounted as a volume in the application deployment, allowing the application to access the required credentials for connecting to the ATP Database. Both options 1 and 3 provide valid approaches for connecting to an ATP Database from a container in Kubernetes, depending on the specific requirements and preferences of the application.

NO.48 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 include service B. Which approach should you take to test service A?
* There is no need to explicitly test APIs.
* Test against production APIs.
* Test using API mocks.
* Test the APIs in private environments.

Explanation

API mocking is a technique that simulates the behavior of real APIs without requiring the actual implementation or deployment of the dependent services1. API mocking allows you to test changes to service A without deploying all of its dependencies, such as service B, by creating mock responses for the APIs that service A relies on1. API mocking has several benefits, such as1:

* Faster testing: You can test your service A without waiting for service B to be ready or available, which reduces the testing time and feedback loop.

* Isolated testing: You can test your service A in isolation from service B, which eliminates the possibility of external factors affecting the test results or causing errors.

* Controlled testing: You can test your service A with different scenarios and edge cases by creating mock responses that mimic various situations, such as success, failure, timeout, etc.

NO.49 (CHK_1>3) You have an e-commerce application that loads customers&#8217; transactional data into the Oracle Cloud Infrastructure (OCI) Streaming service. The data must now be extracted and transformed before sending it to a third-party REST endpoint. You have been directed to leverage the OCI Service Connector Hub to automate this process. Which configuration option would address this requirement?
* Configure a new service connector as follows: * Source: Streaming * Task: None * Target: Notifications
* Configure a new service connector as follows: * Source: Streaming * Task: Functions * Target: API Gateway
* Configure a new service connector as follows: * Source: Streaming * Task: API Gateway * Target:Functions
* Configure a new service connector as follows: * Source: Streaming * Task: Functions * Target:

Functions
* Configure a new service connector as follows: * Source: Streaming * Task: API Gateway * Target:

Notifications
Explanation

To address the requirement of extracting and transforming data from the Oracle Cloud Infrastructure (OCI) Streaming service and sending it to a third-party REST endpoint using the OCI Service Connector Hub, the best configuration option is: Configure a new service connector as follows: * Source: Streaming * Task: None * Target: Notifications By selecting the Streaming service as the source, you can capture the transactional data from the stream. Since there is a need to transform and send the data to a third-party REST endpoint, you don&#8217;t need to specify any specific task in the connector. The target is set to Notifications, which allows you to send the transformed data to an endpoint outside of the OCI environment. Notifications can be configured to deliver the data to various supported destinations, including HTTP endpoints, email addresses, and more. This configuration enables you to automate the process of extracting data from the streaming service and sending it to the desired third-party REST endpoint, fulfilling the requirement of extracting, transforming, and forwarding the data.

NO.50 You are developing a real-time monitoring application for a fleet of vehicles, which will be deployed on Oracle Cloud Infrastructure (OCI). You need to choose between using OCI Queue or OCI Streaming to handle the real-time data feeds from the vehicles. Based on the scenario described, which is the most appropriate choice for handling real-time data feeds?
* OCI Queue, because it provides at-least-once message delivery, which is critical for real-time monitoring applications
* OCI Queue, because it is optimized for low-latency messaging and ideal for real-time applications
* OCI Streaming, because it is designed for high-volume, continuous ingestion and processing of data, making it the best choice for a fleet of vehicles
* OCI Streaming, because it offers exactly-once message delivery, which is necessary for real-time applications
Explanation

OCI Streaming is a fully managed, scalable, and durable messaging solution for ingesting continuous, high-volume streams of data that you can consume and process in real-time1. Streaming is suitable for any use case in which data is produced and processed continually and sequentially in a publish-subscribe messaging model1. Streaming can handle millions of messages per second with low latency2. Therefore, OCI Streaming is the most appropriate choice for handling real-time data feeds from a fleet of vehicles.Verified References: Overview of Streaming, Container Engine for Kubernetes

NO.51 To effectively test your cloud native applications for &#8220;unknown unknowns&#8221;, you need to employ various testing and deployment strategies. Which strategy involves exposing new functionality or features to only a small set of users?
* Canary Deployment
* Component Testing
* A/B Testing
* Blue/Green Deployment
Explanation

The strategy that involves exposing new functionality or features to only a small set of users is called Canary Deployment. Canary deployment is a technique used in software development and deployment where a new version of an application or feature is released to a small subset of users or a specific group of servers. This allows for testing and gathering feedback on the new functionality in a controlled and limited environment before making it available to a wider audience. In a canary deployment, a small portion of the traffic is routed to the new version while the majority of the traffic still goes to the stable version. This allows for monitoring and evaluation of the new functionality in real-world conditions while minimizing the impact of any potential issues or bugs. If the new version performs well and meets the desired criteria, it can then be gradually rolled out to a larger user base or all servers. By exposing the new functionality or features to a small set of users initially, canary deployment helps in identifying any unforeseen issues, gathering feedback, and ensuring the stability and reliability of the application before a full deployment.

NO.52 You are a developing a microservices application that will be a consumer of the Oracle CloudInfrastructure (OCI) Streaming service. Which API method should you use to read and process a stream?
* GetMessages
* GetStream
* ReadMessages
* ReadStream
* ProcessStream
Explanation

The correct API method to read and process a stream in the Oracle Cloud Infrastructure (OCI) Streaming service is &#8220;GetMessages&#8221;. When consuming messages from a stream in OCI Streaming, you use the

&#8220;GetMessages&#8221; API method. This method allows you to retrieve a batch of messages from the stream for processing. You can specify parameters such as the number of messages to retrieve, the maximum size of the messages, and the timeout for the request. By using the &#8220;GetMessages&#8221; API method, you can retrieve messages from the stream and then process them in your microservices application. This allows you to consume and handle the data in real-time as it becomes available in the stream. The &#8220;GetMessages&#8221; method provides flexibility in how you consume and process the messages, enabling you to implement custom logic and workflows based on your specific application requirements.

NO.53 You want to push a new image in the Oracle Cloud Infrastructure (OCI) Registry. Which TWO actions would you need to perform? (Choose two.)
* Generate an API signing key to complete the authentication via Docker CLI.
* Generate an auth token to complete the authentication via Docker CLI.
* Assign an OCI defined tag via OCI CLI to the image.
* Assign a tag via Docker CLI to the image.
* Generate an OCI tag namespace in your repository.
To push a new image to the Oracle Cloud Infrastructure (OCI) Registry, you would need to perform the following two actions: Assign a tag via Docker CLI to the image: Before pushing the image, you need to assign a tag to it using the Docker CLI. The tag helps identify the image and associate it with a specific version or label. Generate an auth token to complete the authentication via Docker CLI: To authenticate and authorize the push operation, you need to generate an auth token. This token is used to authenticate your Docker CLI with the OCI Registry, allowing you to push the image securely. Note: Generating an API signing key, assigning an OCI defined tag via OCI CLI, and generating an OCI tag namespace are not required steps for pushing a new image to the OCI Registry.

NO.54 Who is responsible for patching, upgrading, and maintaining the worker nodes in Oracle Cloud Infrastructure (OCI) Container Engine for Kubernetes (OKE)? (Choose the best answer.)
* Oracle Support
* It is automated
* The user
* Independent Software Vendors

The user is responsible for patching, upgrading, and maintaining the worker nodes in Oracle Cloud Infrastructure (OCI) Container Engine for Kubernetes (OKE). In OKE, the user has control over the worker nodes, which are the compute instances that run the Kubernetes worker components. As the user, you are responsible for managing and maintaining these worker nodes, including tasks such as patching the underlying operating system, upgrading Kubernetes versions, and performing any necessary maintenance activities. While Oracle provides the underlying infrastructure and support services, including managing the control plane and ensuring the availability of the OKE service, the responsibility for managing the worker nodes lies with the user. This allows you to have control and flexibility in managing your Kubernetes environment according to your specific needs and requirements.

NO.55 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?
* 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).
* Adding a custom certificate authority (CA) or custom CA bundle to your gateway&#8217;s trust store for mTLS is optional unless you need to reject certificates that do not contain particular values (such as a domain name).
* Custom CA or custom CA bundles can be added to your gateway&#8217;s trust store ONLY if they already exist in the OCI Certificates service.
* The mTLS request policy can only be enabled at the API deployment specification level, which then applies globally to ALL routes in that deployment.
Explanation

The correct answer is: &#8220;Adding a custom certificate authority (CA) or custom CA bundle to your gateway&#8217;s trust store for mTLS is optional unless you need to reject certificates that do not contain particular values (such as a domain name).&#8221; The statement that is NOT valid regarding the mTLS options in OCI API Gateway is:

&#8220;Adding a custom certificate authority (CA) or custom CA bundle to your gateway&#8217;s trust store for mTLS is optional unless you need to reject certificates that do not contain particular values (such as a domain name).&#8221; In OCI API Gateway, adding a custom certificate authority (CA) or custom CA bundle to the gateway&#8217;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&#8217;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.

NO.56 Which is NOT a valid option to execute a function deployed in Oracle Functions?
* Invoke from the Docker CLI.
* Invoke from the Fn Project CLI.
* Trigger by an event in the Oracle Cloud Infrastructure (OCI) Events service.
* Invoke from the OCI CLI.
* Send signed HTTP requests to the function&#8217;s invoke endpoint.
Explanation

The correct answer is: Invoke from the Docker CLI.Explanation: Executing a function deployed in Oracle Functions is typically done using the following options: Invoke from the Fn Project CLI: The Fn Project CLI provides a command-line interface specifically designed for interacting with Oracle Functions. You can use commands like fn invoke to invoke a function. Trigger by an event in the Oracle Cloud Infrastructure (OCI) Events service: You can configure events in OCI to trigger your function based on various criteria, such as object storage events, resource state changes, or scheduled events. Invoke from the OCI CLI: The OCI CLI

(Command Line Interface) allows you to interact with various services in Oracle Cloud Infrastructure, including Oracle Functions. You can use the fn invoke command to invoke a function. Send signed HTTP requests to the function&#8217;s invoke endpoint: Oracle Functions provides an HTTP endpoint that can be used to invoke functions. You can send signed HTTP requests to this endpoint using tools or programming languages that support making HTTP requests. On the other hand, invoking a function deployed in Oracle Functions using the Docker CLI is not a valid option. The Docker CLI is primarily used for managing Docker containers and images, and it does not provide a direct mechanism for invoking functions in Oracle Functions.

NO.57 Your Oracle Cloud Infrastructure (OCI) Container Engine for Kubernetes (OKE) administrator has created an OKE cluster with one node pool in a public subnet. You have been asked to provide a log file from one of the nodes for troubleshooting purpose. Which step should you take to obtain the log file?
* SSH into the node using the public key.
* It is impossible because OKE is a managed Kubernetes service.
* SSH into the nodes using the private key.
* Use the username opc and password to login.
Explanation

To obtain a log file from one of the nodes in an Oracle Cloud Infrastructure (OCI) Container Enginefor Kubernetes (OKE) cluster, you should SSH into the nodes using the private key. Here&#8217;s the step-by-step process: Obtain the private key: The private key is required to authenticate and access the nodes in the OKE cluster. You should obtain the private key from your administrator or the appropriate key pair used to create the cluster. SSH into the node: Use a secure shell (SSH) client, such as OpenSSH, to connect to the desired node in the cluster. The SSH command typically includes the private key file path and the public IP address or hostname of the node. Example command: ssh -i <private_key_file> opc@<node_public_ip> Replace

<private_key_file> with the path to the private key file and <node_public_ip> with the public IP address of the node you want to access. Navigate to the log file location: Once you have successfully connected to the node, navigate to the directory where the log file is located. The exact location and name of the log file may vary depending on the Kubernetes distribution and configuration. Copy or view the log file: You can either copy the log file from the node to your local machine using the scp command or view the contents directly on the node using tools like cat or less. By following these steps, you will be able to access the log file from the desired node in the OKE cluster for troubleshooting purposes.

NO.58 You are creating an API deployment in Oracle Cloud Infrastructure (OCI) API Gateway and you want to configure request policies to control access. Which is NOT available in OCI API Gateway?
* Controlling access to the backend OCI resources.
* Limiting the number of requests sent to the backend services.
* Enabling Cross-Origin Resource Sharing (CORS) support.
* Providing authentication and authorization.
The correct answer is: Controlling access to the backend OCI resources. OCI API Gateway does not provide direct control over access to backend OCI resources. It primarily focuses on managing and securing access to APIs exposed through the gateway. The gateway acts as a front-end for APIs and provides features such as authentication, authorization, rate limiting, and CORS support. While you can configure authentication and authorization policies, limit the number of requests, and enable CORS support in OCI API Gateway, it does not directly control access to backend OCI resources. Access to backend resources is typically managed through other means, such as IAM policies, network security rules, or resource-specific access controls.

NO.59 Which of the following is NOT a criterion that is usually met by a microservice?
* Organized around business capabilities.
* Tightly coupled
* Highly maintainable
* Independently deployable
The correct answer is: &#8220;Tightly coupled.&#8221; Tightly coupling is not a criterion that is usually met by a microservice. In fact, microservices are designed to be loosely coupled. Loosely coupling refers to reducing dependencies and minimizing the direct

interactions between different components or services. Microservices promote independence and autonomy, allowing each service to operate independently without being tightly bound to other services. The other options listed are criteria that are typically met by microservices: Organized around business capabilities: Microservices architecture suggests designing services around specific business capabilities or functionalities. This allows for focused and specialized services that align with the organization&#8217;s business needs. Independently deployable: Microservices are designed to be independently deployable units. Each microservice can be developed, tested, and deployed separately, without impacting other services. This enables agility and scalability in the deployment process. Highly maintainable: Microservices are often designed to be highly maintainable. They are smaller in scope and focused on specific tasks, making it easier to manage and maintain individual services. Additionally, microservices can be updated, patched, or replaced without affecting the entire system, facilitating easier maintenance and evolution of the application. Therefore, the criterion that is NOT typically met by a microservice is being tightly coupled.

NO.60 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?
* The number of requests to invoke a function that failed with an error response.
* The number of requests to invoke a function that failed due to throttling.
* The number of retries made by the function before failing due to an error.
* The length of time a function runs for.
Explanation

The option that is NOT available when viewing the Application metrics in the Oracle Cloud Infrastructure (OCI) Console is: &#8220;The number of retries made by the function before failing due to an error.&#8221; 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&#8217;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.

NO.61 You developed a microservices-based application that runs in an Oracle Cloud Infrastructure (OCI) Container Engine for Kubernetes (OKE) cluster. It has multiple endpoints that need to be exposed to the public internet.

What is the most cost-effective way to expose multiple application endpoints without adding unnecessary complexity to the application?
* Use a NodePort service type in Kubernetes for each of your service endpoints using the node&#8217;s public IP address to access the applications.
* Use a ClusterIP service type in Kubernetes for each of your service endpoints using a load balancer to expose the endpoints.
* Deploy an Ingress Controller and use it to expose each endpoint with its own routing endpoint.
* Create a separate load balancer instance for each service using the lowest 100 Mbps option.
Explanation

An Ingress Controller is a Kubernetes resource that provides advanced routing and load balancing for your applications running on a Kubernetes cluster1. An Ingress Controller allows you to define rules that specify how to route traffic to different services in your cluster based on the host name or path of the incoming request1. By deploying an Ingress Controller and using it to expose multiple application endpoints, you can achieve the following benefits1:

* Cost-effectiveness: You only need to create one load balancer instance per cluster, instead of one per service, which reduces the cost of exposing your applications.

* Simplicity: You only need to manage one set of routing rules for all your services, instead of configuring each service separately, which simplifies the application deployment and maintenance.

* Flexibility: You can use different types of Ingress Controllers, such as NGINX or Traefik, that offer various features and customization options for your routing needs.

NO.62 What are the TWO main reasons you would choose to implement a serverless architecture? (Choose two.)
* Easier to run long-running operations
* Improved in-function state management
* Reduced operational cost
* Automatic horizontal scaling
* No need for integration testing
Explanation

The two main reasons to choose a serverless architecture are: Automatic horizontal scaling: Serverless architectures allow for automatic scaling of resources based on demand. The infrastructure automatically provisions and scales resources as needed, ensuring that applications can handle varying workloads efficiently.

This eliminates the need for manual scaling and optimizes resource utilization. Reduced operational cost:

Serverless architectures follow a pay-per-use model, where you are billed only for the actual execution time and resources consumed by your functions. This leads to cost savings as you don&#8217;t have to pay for idle resources. Additionally, serverless architectures remove the need for managing and maintaining servers, reducing operational overhead and associated costs.Explanation: No need for integration testing: Integration testing is still necessary in serverless architectures to ensure that functions integrate correctly with other components and services. Serverless functions can interact with various event sources, databases, and APIs, and testing is required to verify the integration points. Improved in-function state management: Serverless architectures typically encourage stateless functions that operate on short-lived requests or events. While there are mechanisms to manage state within a function, serverless architectures are designed to be stateless by default, promoting scalability and fault tolerance. Easier to run long-running operations: Serverless functions are generally designed for short-lived operations rather than long-running tasks. If you have a requirement for long-running operations, a serverless architecture may not be the ideal choice, as it has execution time limits and may not provide the necessary resources for extended execution.

NO.63 A developer has created another version of a microservice and wants 10% of the traffic to flow towards it for testing purposes. The application is already configured using OCI (Oracle Cloud Infrastructure) Service Mesh. Which of the following steps is the right approach to achieve this goal?
* Create a new Kubernetes deployment for the new version of the microservice and set the traffic splitting percentage to 10% in the Kubernetes service manifest.
* Use Kubernetes HPA (Horizontal Pod Autoscaler) to scale the new version of the microservice to handle 10% of the traffic automatically.
* Create a new entry in the routeRules field of the ingress gateway route table manifest to configure traffic splitting between the old and new versions of the microservice and set the percentage to 10%.
* Create a new entry in the routeRules field of the virtual service route table manifest to configure traffic splitting between the old and new versions of the microservice and set the percentage to 10%.

Get ready to pass the 1z0-1084-23 Exam right now using our Oracle Cloud Exam Package:
https://www.examcollectionpass.com/Oracle/1z0-1084-23-practice-exam-dumps.html]