# Oracle

## 1Z0-1084-24

## Oracle Cloud Infrastructure 2024 Developer Professional

**Questions And Answers PDF Format:**

**For More Information – Visit link below:**
**https://www.certsgrade.com/**

*Version* = **Product**

# Latest Version: 6.0

## Question: 1

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?

A. Use the username opc and password to login.
B. It is impossible because OKE is a managed Kubernetes service.
C. SSH into the nodes using the private key.
D. SSH into the node using the public key.

**Answer: C**

Explanation:
To obtain a log file from one of the nodes in an Oracle Cloud Infrastructure (OCI) Container Engine for Kubernetes (OKE) cluster, you should SSH into the nodes using the private key. Here'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.

## Question: 2

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?

A. 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: "443" service.beta.kubernetes.io/oci-load-balancer-tls-secret: ssl certificate-secret
B. 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: "443" service.beta.kubernetes.io/oci-load-balancer-security-list management-mode: "Frontend"
C. Add these annotations to the kubernetes service: annotations: service.beta.kubernetes.io/oci-load-balancer-ssl-ports: "443" service.beta.kubernetes.io/oci-load-balancer-ssl-secret-key: ssl secret-key
D. Generate a self-signed certificate using Let's Encrypt. Use that certificate on OCI Load Balancer. Create the Kubernetes service using this load balancer.

**Answer: A**

Explanation:
The correct answer is: "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: '443' service.beta.kubernetes.io/oci-load-balancer-tls-secret: ssl certificate-secret." 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: '443' (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 SSL termination 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'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.

## Question: 3

You are building a cloud native serverless travel application with multiple Oracle Functions in Java, Python, and Node.js. You need to build and deploy these functions to a single application named travel-app. Which command will help you complete this task successfully?

A. fn function deploy app travel-app--all
B. fn app deploy --app travel-app --all
C. fn app --app travel-app deploy --ext java pyljs
D. fn deploy--app travel-app --all

**Answer: D**

Explanation:
The correct answer is: fn deploy --app travel-app --all Explanation:: To build and deploy multiple Oracle Functions as part of a single application named "travel-app," you can use the fn deploy command with the appropriate options. The command fn deploy --app travel-app --all is the correct syntax. Here's what each part of the command does: fn deploy: This command is used to deploy functions and applications in Oracle Functions. --app travel-app: This option specifies the application name as "travel-app," indicating that you want to deploy functions to this application. --all: This option indicates that you want to deploy all the functions within the application. By using fn deploy --app travel-app --all, you can build and deploy all the functions in your travel application across different programming languages (Java, Python, and Node.js) to the "travel-app" application in Oracle Functions.

## Question: 4

Which of the following is NOT a criterion that is usually met by a microservice?

A. Organized around business capabilities.
B. Tightly coupled
C. Highly maintainable
D. Independently deployable

**Answer: B**

Explanation:
The correct answer is: "Tightly coupled." 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'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.

## Question: 5

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?

A. docker build -t OCIdevops/api:latest
B. docker create -t OCIdevops/api:latest
C. docker image -t OCIdevops/api:latest
D. docker compile -t OCI devops/api:latest

## Answer: A

Explanation:
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.

For More Information – Visit link below:

https://www.certsgrade.com/

# PRODUCT FEATURES

- 100% Money Back Guarantee
- 90 Days Free updates
- Special Discounts on Bulk Orders
- Guaranteed Success
- 50,000 Satisfied Customers
- 100% Secure Shopping
- Privacy Policy
- Refund Policy

**16 USD Discount Coupon Code: NB4XKTMZ**