# Free Questions for 1Z0-1084-23

## Shared by Watson on 04-10-2024

For More Free Questions and Preparation Resources

Check the Links on Last Page

# Question 1

**Question Type: MultipleChoice**

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?

## Options:

**A-** Controlling access to the backend OCI resources.

**B-** Limiting the number of requests sent to the backend services.

**C-** Enabling Cross-Origin Resource Sharing (CORS) support.

**D-** Providing authentication and authorization.

## Answer:

A

## Explanation:

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.

# Question 2

**Question Type: MultipleChoice**

(CHK_4>2) You have a scenario where a DevOps team wants to store secrets in Oracle Cloud Infrastructure (OCI) Vault so that it can inject the secrets into an app's environment variables (for example, MYSQL_DB_PASSWD) at deployment time. Which is NOT valid about managing secrets in the OCI Vault service?

## Options:

**A-** New secret versions automatically expire in 90 days unless you configure an expiry rule.

**B-** You can manually create new secrets as well as new secret versions using the OCI Console:

**C-** A unique OCID is automatically generated for each secret and remains unchanged even when creating a new secret version.

**D-** A secret reuse rule prevents the use of secret contents across different versions of a secret.

## Answer:

C

## Explanation:

The correct answer is: 'A unique OCID is automatically generated for each secret and remains unchanged even when creating a new secret version.' The statement that is NOT valid about managing secrets in the OCI Vault service is: 'A unique OCID is automatically generated for each secret and remains unchanged even when creating a new secret version.' In OCI Vault, a secret is identified by its OCID (Oracle Cloud Identifier), which is a unique identifier for each resource in Oracle Cloud Infrastructure. However, when a new secret version is created for an existing secret, the OCID remains the same for the secret itself, but a new OCID is generated for the secret version. This allows you to track and manage different versions of a secret while maintaining a consistent OCID for the secret itself. The other statements mentioned are valid: You can manually create new secrets as well as new secret versions using the OCI Console. This means you have control over creating and managing secrets within the Vault service. A secret reuse rule prevents the use of secret contents across different versions of a secret. This ensures that each secret version maintains its own unique set of contents and avoids accidental reuse or sharing of secrets across versions. By default, new secret versions automatically expire in 90 days unless you configure an expiry rule. This helps enforce good security practices by automatically rotating secrets periodically, reducing the risk of unauthorized access in case of compromise. Therefore, the statement that is NOT valid is the one regarding the uniqueness and consistency of the OCID when creating new secret versions.

# Question 3

**Question Type:** MultipleChoice

You are developing a polyglot serverless application using Oracle Functions. Which language cannot be used to write your function code?

## Options:

**A-** PL/SQL

**B-** Python

**C-** Node.js

**D-** Go

**E-** Java

## Answer:

A

## Explanation:

Oracle Functions does not currently support PL/SQL as a language for writing function code. PL/SQL is a procedural language used in Oracle Database for developing stored procedures, triggers, and other database-related code. However, Oracle Functions supports several other popular programming languages such as Go, Node.js, Python, and Java, allowing developers to choose the language that best suits their application requirements and their familiarity with the language. While PL/SQL is powerful for working with the Oracle Database, it is not an option for writing function code in the Oracle Functions serverless architecture.

# Question 4

**Question Type: MultipleChoice**

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?

## Options:

**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.

# Question 5

**Question Type: MultipleChoice**

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

## Options:

**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 6

**Question Type:** **MultipleChoice**

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?

## Options:

**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 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.