Oracle

1Z0-1109-24 Exam

Oracle Cloud Infrastructure 2024 DevOps Professional

Questions & Answers
Demo

Version: 4.0

Question: 1

As a cloud engineer, you are responsible for managing a Kubernetes cluster on the Oracle Cloud Infrastructure (OCI) platform for your organization. You are looking for ways to ensure reliable operations of Kubernetes at scale while minimizing the operational overhead of managing the worker node infrastructure.
Which cluster option is the best fit for your requirement?

A. Using OCI OKE managed nodes with cluster autoscalers to eliminate worker node infrastructure management
B. Using OCI OKE virtual nodes to eliminate worker node infrastructure management
C. Using Kubernetes cluster add-ons to automate worker node management
D. Creating and managing worker nodes using OCI compute instances

Answer: B

Explanation:
Step 1: Understanding the Requirement
The goal is to ensure reliable operations of Kubernetes at scale while minimizing the operational overhead of managing worker node infrastructure. In this context, a solution is needed that abstracts away the complexity of managing, scaling, and maintaining worker nodes.

Step 2: Explanation of the Options
A . Using OCI OKE managed nodes with cluster autoscalers
While this option provides managed node pools and uses cluster autoscalers to adjust resources based on demand, it still requires some level of management for the underlying worker nodes (e.g., patching, upgrading, monitoring).
Operational overhead: Moderate.
B . Using OCI OKE virtual nodes
Virtual nodes in OCI OKE are a serverless option for running Kubernetes pods. They remove the need to manage underlying worker nodes entirely.
OCI provisions resources dynamically, allowing scaling based purely on pod demand.
There's no need for node management, patching, or infrastructure planning, which perfectly aligns with the requirement to minimize operational overhead.
Operational overhead: Minimal.
Best Fit for This Scenario: Since the requirement emphasizes minimizing operational overhead, this is the ideal solution.

C . Using Kubernetes cluster add-ons to automate worker node management
Kubernetes add-ons like Cluster Autoscaler or Node Problem Detector help in automating some aspects of worker node management. However, this still requires managing worker node infrastructure at the core level.
Operational overhead: Moderate to high.
D . Creating and managing worker nodes using OCI compute instances
This involves manually provisioning and managing compute instances for worker nodes, including scaling, patching, and troubleshooting.
Operational overhead: High.
Not Suitable for the Requirement: This option contradicts the goal of minimizing operational overhead.

Step 3: Why Virtual Nodes Are the Best Fit
Virtual Nodes in OCI OKE:
Virtual nodes provide serverless compute for Kubernetes pods, allowing users to run workloads without provisioning or managing worker node infrastructure.
Scaling: Pods are automatically scheduled, and the required infrastructure is dynamically provisioned behind the scenes.
Cost Efficiency: You only pay for the resources consumed by the running workloads.
Use Case Alignment: Eliminating the burden of worker node infrastructure management while ensuring Kubernetes reliability at scale.

Step 4: References and OCI Resources
OCI Documentation:
OCI Kubernetes Virtual Nodes
OCI Container Engine for Kubernetes Overview
Best Practices for Kubernetes on OCI:
Best Practices for OCI Kubernetes Clusters

Question: 2

How do OCI DevOps Deployment Pipelines reduce risk and complexity of production applications?

A. By reducing change-driven errors introduced by manual deployments
B. By scaling builds with service-managed build runners
C. By working with existing Git repositories and CI systems
D. By eliminating downtime of production applications

Answer: A

Explanation:

OCI DevOps Deployment Pipelines automate the process of deploying applications to production environments. By using automated, repeatable deployment processes, they help reduce the risk of

change-driven errors, which are often introduced during manual deployments. This automation reduces human errors and ensures consistency across environments, thus minimizing complexity and risk in production.

Question: 3

How does the Oracle Cloud Infrastructure Container Engine for Kubernetes (OKE) Cluster Autoscaler determine when to create new nodes for an OKE cluster?

A. When the CPU or memory utilization crosses a configured threshold.
B. When the resource requests from pods exceed a configured threshold.
C. When the custom metrics from the services exceed a configured threshold.
D. When the rate of requests to the application crosses a configured threshold.

Answer: B

Explanation:

The OKE Cluster Autoscaler automatically adjusts the number of worker nodes in an OKE cluster based on the resource requests made by Kubernetes pods. When there are not enough resources available (e.g., CPU or memory) on existing nodes to accommodate pending pods, the Cluster Autoscaler will create new nodes to meet the resource demand.

Question: 4

A team wants to deploy artificial intelligence and machine learning workloads in their OCI Container Engine for Kubernetes (OKE) cluster. They prioritize strong isolation, cost-efficiency, and the ability to leverage serverless capabilities.
Which solution is best suited for their requirements?

A. Virtual nodes in OKE
B. Self-Managed Nodes in OKE
C. Managed nodes in OKE
D. Container Instances in OCI

Answer: A

Explanation:

Virtual nodes in OKE provide a serverless experience for deploying Kubernetes workloads, which means you do not have to manage or scale the underlying infrastructure. This solution is particularly cost-efficient because you only pay for the resources used by the pods, and it provides strong isolation for workloads.
Virtual nodes are well suited for AI/ML workloads as they allow users to easily scale compute resources without being constrained by the limits of individual worker nodes.

Question: 5

Which command creates the docker registry secret required in the application manifests for OKE to pull images from Oracle Cloud Infrastructure Registry?
A)

```
kubectl create secret docker-registry <secret-name> --docker-server=
<region-key>.ocir.io --docker-username=<oci-username> --docker-
password='<oci-auth-token>' --docker-email=<email-address>
```

B)

```
kubectl create passwd docker-registry <secret-name> --docker-server=
<region-key>.oke.io --docker-username=<tenancy-namespace>/<oci-
username> --docker-password='<oci-auth-token>' --docker-email=<email-
address>
```

C)

```
kubectl create passwd docker-registry <secret-name> --docker-server=
<region-key>.oke.io --docker-username=<oci-username> --docker-
password='<oci-auth-token>' --docker-email=<email-address>
```

D)

```
kubectl create secret docker-registry <secret-name> --docker-server=
<region-key>.ocir.io --docker-username=<tenancy-namespace>/<oci-
username> --docker-password='<oci-auth-token>' --docker-email=<email-
address>
```

A. Option A
B. Option B
C. Option C
D. Option D

Answer: D

Explanation:

To create a Docker registry secret to pull images from the Oracle Cloud Infrastructure Registry (OCIR), you need to specify the correct parameters such as the region key, namespace, OCI username, and OCI authentication token.
Chosen command is correct because:
The kubectl create secret docker-registry command creates a Docker registry secret.
The --docker-server=<region-key>.ocir.io specifies the correct endpoint for OCIR.
The --docker-username=<tenancy-namespace>/<oci-username> provides both the tenancy namespace and the OCI username, which is the required format for authentication with OCIR.
The --docker-password='<oci-auth-token>' specifies the OCI auth token, which acts as a password for authentication.
The --docker-email=<email-address> is also included.
The other commands have errors, such as missing tenancy namespace or using incorrect flags (passwd instead of secret).