PassQueen
a.k.a dumpsqueen.com

Oracle Cloud Infrastructure 2026 Cloud Ops Professional

Oracle 1z0-1067-25

Version Demo

Total Demo Questions: 4

Total Premium Questions: 47

Buy Premium PDF

https://passqueen.com

support@passqueen.com

Topic Break Down

Topic	No. of Questions
Topic 1, Identity and Access Management (IAM)	9
Topic 2, Networking	11
Topic 3, Compute	5
Topic 4, Storage	7
Topic 5, Database	1
Topic 6, Security	1
Topic 7, Observability and Management	9
Topic 8, Governance and Cost Management	4
Total	47

QUESTION NO: 1 - (SIMULATION)

Scenario: 2 (Oracle Cloud-init and AutoScaling: Use cloud-init to Configure Apache on Instances in an Autoscaling Instance Pool)

Scenario Description: (Hands-On Performance Exam Certification)

You ' re deploying an Apache-based web application on OCI that requires horizontal autoscaling.

To configure instances upon provisioning, write a cloud-init script for Oracle Linux 8 that installs and enables Apache (httpd), and opens the firewall for HTTP on TCP port 80. Create an instance configuration and include the cloud-init script in it. Use this instance configuration to create an instance pool and autoscaling configuration.

Pre-Configuration:

To fulfill this requirement, you are provided with the following:

Access to an OCI tenancy, an assigned compartment, and OCI credentials

A VCN Cloud-Init Challenge VCN with an Internet gateway and a public subnet. The security list for the subnet allows ingress via TCP ports 22 and 80 (SSH and HTTP). The route table forwards all egress to the Internet gateway.

Access to the OCI Console

Required IAM policies

An SSH key pair for the compute instance

Public Key https://objectstorage.us-ashburn-1.oraclecloud.com/n/tenancyname/b/PBT_Storage/o/PublicKey.pub

Private Key https://objectstorage.us-ashburn-1.oraclecloud.com/n/tenancyname/b/PBT_Storage/o/PKey.key

Note: Throughout your exam, ensure to use assigned Compartment , User Name , and Region.

Complete the following tasks in the provisioned OCI environment:

Task 1(a): Develop the cloud-init Script:

Task 1(b): Use cloud-init to Configure Apache on Instances in an Autoscaling Instance Pool:

ANSWER: See the explanation for the answer

Explanation:

Use the assigned compartment and region for every resource.

Create the cloud-init user-data script below. Paste it as plain text under **Advanced options** / **Management** / **Cloud-init script** when creating both the standalone test instance and the instance configuration.

```
#!/bin/bash
dnf install -y httpd
systemctl enable httpd
systemctl start httpd
firewall-offline-cmd --add-service=http
systemctl restart firewalld
```

Cloud-init runs as root, so `sudo` is not required. The subnet security list already permits TCP/80; the script additionally opens the guest OS firewall service named `http`.

Create instance `pbt_cloud_init_vm_01` in `Cloud-Init Challenge SNT`.
Select **Oracle Linux 8**, shape **VM.Standard.A1.Flex**, and configure **1 OCPU** and **6 GB** memory.
Select any available AD, assign a public IPv4 address, and paste/upload the supplied public SSH key.
Add the preceding cloud-init script and create the instance. After it is running, HTTP should be reachable through its public IP on port 80.

Next, create instance configuration `pbt_cloud_init_config_01` with the same settings:

Oracle Linux 8; `VM.Standard.A1.Flex`; 1 OCPU; 6 GB memory; any AD.

Public subnet `Cloud-Init Challenge SNT` and a public IPv4 assignment.
The supplied SSH public key.
The exact cloud-init script shown above.

Create instance pool `pbt_cloud_init_pool_01` using `pbt_cloud_init_config_01`. Set its initial instance count to **1**, select an available AD, use the specified public subnet/VCN for the primary VNIC, leave fault domain unselected, and do not attach a load balancer.

Create and attach autoscaling configuration `pbt_cloud_autoscaling_config_01` for `pbt_cloud_init_pool_01` with these values:

Autoscaling type: **Metric-based autoscaling**
Performance metric: **CPU utilization**
Cooldown: **300 seconds**
Scale-out: operator **Greater than (>)**, threshold **75%**, add **1** instance
Scale-in: operator **Less than (<)**, threshold **25%**, remove **1** instance
Minimum instances: **1**
Maximum instances: **2**
Initial instances: **1**

This ensures every newly provisioned pool instance runs the Apache installation, enablement, startup, and HTTP firewall configuration during first boot.

QUESTION NO: 2 - (SIMULATION)

Scenario: 1 (Create a reusable VCN Configuration with Terraform)

Scenario Description: (Hands-On Performance Exam Certification)

You'll launch and destroy a VCN and subnet by creating Terraform automation scripts and issuing commands in Code Editor. Next, you'll download those Terraform scripts and create a stack by uploading them into Oracle Cloud Infrastructure Resource Manager.

You'll then use that service to launch and destroy the same VCN and subnet.

In this scenario, you will:

a. Create a Terraform folder and file in Code Editor.

b. Create and destroy a VCN using Terraform.

c. Create and destroy a VCN using Resource Manager.

ANSWER: See the explanation for the answer

Explanation:

Answer: Create the Terraform configuration below, use it first from Code Editor/Cloud Shell to apply and destroy the VCN, then upload the three configuration files to a Resource Manager stack and run Plan, Apply (using that plan), and Destroy.

Create a folder named `terraform_vcn` and add these files.

vcn.tf

```
terraform {
  required_providers {
    oci = {
      source  = "oracle/oci"
      version = ">= 4.67.3"
    }
  }
  required_version = ">= 1.0.0"
}
```

```
resource "oci_core_vcn" "example_vcn" {
  compartment_id = var.compartment_id
  display_name   = "VCN-01"
  cidr_blocks    = ["10.0.0.0/16"]
}

resource "oci_core_subnet" "example_subnet" {
  compartment_id = var.compartment_id
  display_name   = "SNT-01"
  vcn_id         = oci_core_vcn.example_vcn.id
  cidr_block     = "10.0.0.0/24"
}
```

variables.tf

```
variable "compartment_id" {
  type = string
}
```

terraform.tfvars

```
compartment_id = "<your_compartment_OCID>"
```

Replace `<your_compartment_OCID>` with the target compartment OCID. The subnet references `oci_core_vcn.example_vcn.id`, which causes Terraform to create the VCN before creating the subnet.

In the Code Editor terminal, change to the configuration directory and run:

```
cd ~/terraform_vcn
terraform init
terraform plan
terraform apply
# Type: yes
terraform destroy
# Type: yes
```

Verify after `apply` that VCN `VCN-01` and subnet `SNT-01` exist in the selected compartment. Verify after `destroy` that they are terminated.

Resource Manager steps:

Download/upload only `vcn.tf`, `variables.tf`, and `terraform.tfvars` in the `terraform_vcn` folder. Do not include `.terraform` or Terraform state files.
Open **Developer Services > Resource Manager > Stacks** and select **Create stack**.
Select **My configuration**, upload the `terraform_vcn` folder, leave **Use custom Terraform providers** unselected, specify a stack name/description and the target compartment, then select **Next**.
Confirm that `compartment_id` is populated from `terraform.tfvars`, select **Next**, leave **Run apply** deselected, and select **Create**.
On the stack, select **Plan**, name the job `RM-Plan-01`, and submit it. Wait for the plan job to succeed.
Select **Apply**, name the job `RM-Apply-01`, and under **Apply job plan resolution** select the completed `RM-Plan-01` job rather than **Automatically approve**. Submit the apply job and wait for success.
Confirm `VCN-01` and `SNT-01` were created.
Return to the stack, select **Destroy**, confirm **Destroy**, and wait for the destroy job to succeed.
Finally select **More actions > Delete stack** and confirm deletion.

The essential Resource Manager requirement is to apply the previously completed plan job, not automatically approve a newly generated plan.

QUESTION NO: 3

Which option is NOT a possible return value for an OCI health check?

A. REGEX_MISMATCH

B. UNKNOWN

C. UNREACHABLE

D. INVALID_STATUS_CODE

E. TIMED_OUT

ANSWER: C

Explanation:

UNREACHABLE is not a defined OCI Health Checks return value. OCI Health Checks reports the outcome of an HTTP probe using documented result and failure classifications. When a probe cannot establish or complete the expected check, the service records a specific supported result such as a timeout, an unexpected HTTP status code, or a response-body regular-expression mismatch. These values allow an operator to distinguish an endpoint that did not respond in time from one that responded but failed the configured HTTP or content-validation rule.

Using the documented return vocabulary is important when interpreting health-check results programmatically, building Monitoring queries and alarms, or automating incident handling. Automation should evaluate the exact strings emitted by the Health Checks service rather than infer a general network-state label. In particular, OCI represents reachability-related probe failures through its supported, more specific return classifications; it does not return the generic value **UNREACHABLE**.

Oracle documents the Health Checks service and its HTTP monitor/probe-result model in the OCI Health Checks documentation and the HTTP probe result API model.

QUESTION NO: 4

The general syntax for an IAM policy is: Allow < identity_domain_name > / < subject > to < verb > < resource-type > in < location > where < conditions > Which two are valid values for < verb > ?

A. destroy

B. create

C. manage

D. read

E. alter

ANSWER: C D

Explanation:

manage and **read** are valid OCI IAM policy verbs. OCI policy statements use standardized verbs to assign progressively broader permission levels for a resource type. The principal policy verbs are *inspect*, *read*, *use*, and *manage*. The **read** verb grants permissions to view details of resources and retrieve associated information without permitting modifications. It is appropriate for users or groups that need visibility into configured resources and their attributes.

The **manage** verb is the broadest standard policy verb. It includes the permissions granted by the lower-level verbs and generally permits full lifecycle administration of the specified resource type, subject to the resource family and any conditions included in the policy statement. For example, a policy granting a group permission to manage a resource type in a compartment enables that group to create, update, move, and delete resources when those operations are defined for the resource type. Oracle documents these standardized policy verbs and their relationship to API permissions in its IAM policy reference: OCI IAM Policies and Advanced Policy Features.