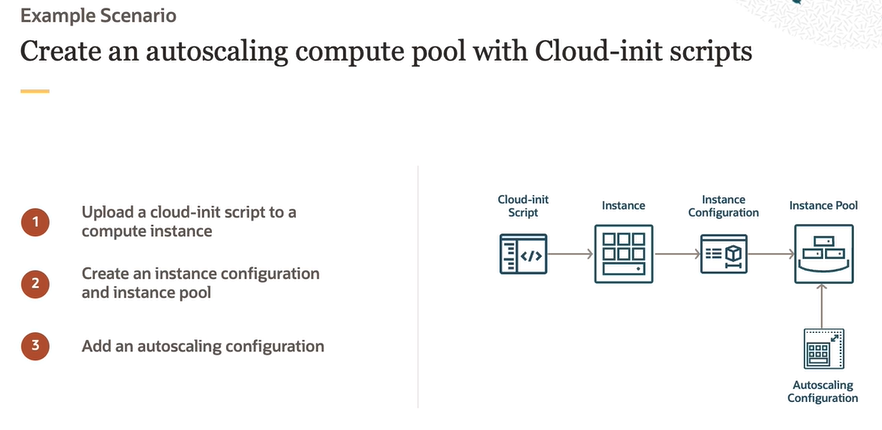
A screenshot of a computer

Description automatically generated



A diagram of a cloud storage system

Description automatically generated

A screenshot of a computer

Description automatically generated

A diagram of a cloud network

Description automatically generated

A diagram of a server

Description automatically generated

Overview

In this lab, you will deploy and manage a scalable, secure, DNS-mapped front-end *web application* that can be accessed by the internal users of your organization—either within your virtual cloud network or on-premises infrastructure.

This lab builds up a web application in several parts. First, you’ll host an Apache app on a single virtual machine (VM). Then, you’ll configure the host app on a pool of autoscaling compute instances behind a load balancer. And finally, you’ll add front-end SSL certification to the load balancer.

In this lab, you’ll:

a. Create a VCN with gateways, route tables, security lists, and subnets.

b. Create a VM that hosts an Apache application in two different ways:

1) SSH into the machine and use Bash to configure Apache

2) Provide a cloud-init script to execute when the VM launches

c. Create and auto-scale compute pools and attach to the load balancer.

d. Create a private DNS zone and configure front-end SSL for the load balancer.

1. In this first practice, you will *set up a VCN* and its common elements - gateways, security lists, and route tables, and subnets. While you could do this with the Cloud Console's VCN Wizard, this lab will guide you through the manual way to better understand the console and process.

1.Create VCN

Under IPv4 CIDR Blocks, add 10.0.0.0/16

2. Add Gateways to the VCN

To allow bi-directional traffic between the Internet and the VCN, you will add an *Internet gateway*. Later, you’ll attach this to a *public subnet*.

To allow private (egress-initiated only) traffic from the VCN into the Internet, you’ll create a *NAT gateway*. Later, you’ll attach this to a *private subnet*.

To allow traffic between the VCN and Oracle services, you will add a *service gateway*. Later, you’ll attach this to a *private subnet*. Strictly speaking, this gateway may not be needed for this lab. However, it will come in handy if you need to use the Bastion service to connect to private instances and debug.

3. Add Route Tables to the VCN

To direct traffic to these gateways, you need to create route tables to point to them.

First, you’ll create one for public Internet access. Click Create Route Table. RT-PUBLIC

Under Route Rules (Optional), click + Another Route Rule.

For Destination CIDR Block, type 0.0.0.0/0

Under Target Internet Gateway, select the Internet gateway. This should be named <REGION>-OP-LAB01-1-IG

Next, you’ll create one for private Internet and Oracle services access. Click Create Route Table RT-PRIVATE

Under Route Rules (Optional), click + Another Route Rule.

For Destination CIDR Block, type 0.0.0.0/0

Under Target NAT Gateway, select the NAT. This should be named <REGION>-OP-LAB01-1-NG

To add a second rule, for the service gateway, click + Another Route Rule

For Destination Services, select ALL <REGION> Services in Oracle Services Network, where <REGION> will be replaced by the appropriate region code.

Under Target Service Gateway, select the service gateway you created earlier. This should be named <REGION>-OP-LAB01-1-SG

4. Add Security Lists to the VCN

In order for subnets to accept traffic from these gateways, you’ll need to create security lists that specify allowed traffic.

First, you’ll create a security list for public Internet traffic SL-PUBLIC

Under Allow Rules for Ingress, click + Another Ingress Rule. A form for Ingress Rule 1 should appear.

a) Leave Stateless deselected.

b) For Source Type, leave CIDR selected.

c) For Source CIDR, enter 0.0.0.0/0 to allow traffic from the Internet.

d) For IP Protocol, leave TCP selected.

e) Leave Source Port Range empty.

f) For Destination Port Range enter 80,443 to allow HTTP and HTTPS traffic.

Next, you’ll create another security list for private traffic. SL-PRIVATE

For Source CIDR, enter 10.0.0.0/16 to limit traffic to that originating in the VCN.

5. Add Subnets to the VCN

three subnets to the VCN: one public and two private.

SNT-PUBLIC

3) For Subnet Type, select Regional (Recommended).

4) For CIDR Block, enter 10.0.0.0/24

5) For Route Table, select the route table you created earlier for public access. <REGION>-OP-LAB01-1-RT-PUBLIC

6) For Subnet Access, select Public Subnet.

7) Leave Use DNS hostnames in this SUBNET selected.

8) For DNS Label, enter SNTPUBLIC

9) For the DHCP options, select the default.

10) Under the box Security Lists, you will attach two security lists.

a) First, select the default security list in the drop-down to allow SSH and ICMP traffic.

b) Second, click + Another Security List to add another drop-down list.

c) Under the new drop-down list, select the security list you created earlier for public HTTP(S) access. This should be named <REGION>-OP-LAB01-1-SL-PUBLIC

SNT-PRIVATE1

3) For Subnet Type, select Regional (Recommended).

4) For CIDR Block, enter 10.0.1.0/24

5) For Route Table, select the route table for private access. <REGION>-OP-LAB01-1-RT-PRIVATE

6) For Subnet Access, select Private Subnet.

7) Leave Use DNS hostnames in this SUBNET selected.

8) For DNS Label, enter SNTPRIVATE1

9) For the DHCP options, select the default.

10) Under the box Security Lists, you will attach two security lists.

a) First, select the default security list in the drop-down list to allow SSH and ICMP traffic.

b) Second, click + Another Security List to add another drop-down list.

c) Under the new drop-down list, select the security list you created earlier for intra-VCN HTTP(S) access. This should be named <REGION>-OP-LAB01-1-SL-PRIVATE

SNT-PRIVATE2

3) For Subnet Type, select Regional (Recommended).

4) For CIDR Block, enter 10.0.2.0/24

5) For Route Table, select the route table for private access. <REGION>-OP-LAB01-1-RT-PRIVATE

6) For Subnet Access, select Private Subnet.

7) Leave Use DNS hostnames in this SUBNET selected.

8) For DNS Label, enter SNTPRIVATE2

9) For the DHCP options, select the default.

10) Under the box Security Lists, you will attach two security lists.

a) First, select the default security list in the drop-down list to allow SSH and ICMP traffic.

b) Second, click + Another Security List to add another drop-down list.

c) Under the new drop-down list, select the security list you created earlier for intra-VCN HTTP(S) access. This should be named <REGION>-OP-LAB01-1-SL-PRIVATE

You should now have a VCN with one public subnet and two private subnets. The public subnet is attached to an Internet gateway; the private subnets are attached to a NAT gateway and a Service gateway. All the subnets allow SSH and ICMP traffic due to the default security list. The public subnet also allows HTTP(S) traffic from anywhere. The private subnets allow HTTP(S) traffic originating from within the VCN.

**Create SSH Keys for Cloud Shell**

2. Use the following command in Cloud Shell to generate an SSH key pair

$ ssh-keygen -t rsa -b 4096

3. It will prompt you for the location to save the files. Leave it empty and press Enter to use the default.

4. It will prompt you for a password. For this lab, leave it empty and press Enter to skip. Press Enter again to confirm.

5. Add your keys to the ssh-agent with the following two commands:

$ eval “$(ssh-agent -s)”

$ ssh-add ~/.ssh/id\_rsa

6. Download the public key: Click the gear on the top right of Cloud Shell. Click Download.

c. Enter .ssh/id\_rsa.pub to specify the public key. Click Download to confirm. This is the key you will upload to compute instances when provisioning them.

**Set Up a Webserver with SSH**

you will add a VM instance to the public subnet in the VCN, then use SSH in Cloud Shell to install and configure an Apache webserver.

*Create a VM*

1. First, you’ll create a VM in your public subnet.

Click Create instance to open the instance creation form.

1) For Name, enter <REGION>-OP-LAB01-1-VM-01.

2) In the Placement box, select AD 1 for the first availability domain.

3) Leave the Image as Oracle Linux 8.

Click Change shape - For Instance type, leave Virtual Machine selected.

Shape series - Ampere. Select VM.Standard.A1.Flex. Number of OCPUs – 1 . Amount of memory (GB) - 6

4) In the Networking box:

a) Under Primary Network, choose Select existing virtual cloud network.

b) Under Virtual cloud network, select the VCN you created earlier. <REGION>-OP-LAB01-1-VCN-01

c) Subnet - Select existing subnet and choose the public subnet <REGION>-OP-LAB01-1-SNT-PUBLIC

d) Leave Assign a public IPv4 address selected.

5) Add SSH keys box - upload the SSH keys generated earlier or paste your Cloud Shell machine’s public key.

6) Leave the Boot volume options as the default.

d. Click Create. The console will take you to the instance’s details page.

2. Wait for the instance to finish provisioning to view its details.

a. Under Instance Access, copy the Public IP address.

Install and Configure Apache HTTP Server

3. Next, you will use Cloud Shell to SSH into the VM and install Apache.

a. Inthe Cloud Shell

$ ssh opc@<public\_ip\_address>

If you get a Connection refused error, wait a minute before trying again. It’s possible for the VM to show it is RUNNING but sshd hasn’t finished booting. If you get the error that your key has permissions that are too open, use this to update permissions:

$ chmod 600 <private\_key>

b. Install Apache webserver (httpd), configure it to start whenever the VM boots, and activate it.

$ sudo dnf install httpd

$ sudo systemctl enable httpd

$ sudo systemctl start httpd

c. Open the OS firewall to HTTP.

$ sudo firewall-cmd --permanent --add-service=http

$ sudo firewall-cmd --reload

4. To verify that the webserver is running, enter its public IP address into your browser’s address bar. It might take a few minutes after the instance says it is running. Apache's 1 default page should display.

5. Now that you have seen the process of manually starting an Apache webserver, you can do it through cloud-init automation instead. Terminate the above instance.

**Set Up Webserver with Cloud-init**

you’ll create a second VM instance. This time, you’ll use a cloud-init script instead of using SSH to install and activate an Apache webserver with the same firewall settings

Create VM

+ Show advanced options

i. Under the Management tab, choose Paste cloud-init script.

ii. Type the following commands:

#!/bin/sh

sudo dnf install httpd --assumeyes --quiet

sudo systemctl enable httpd

sudo systemctl start httpd

sudo firewall-offline-cmd --add-service=http

systemctl restart firewalld

echo '<!doctype html><html><body><h1>This is myoracle.com!</h1></body></html>' | sudo tee var/www/html/index.html

Note: There are a couple differences between these commands and the ones we

used through SSH:

• We added two flags for dnf to (1) assume we answer \yes\ when it asks whether to install httpd and dependencies and (2) limit logging.

• We used firewall-offline-cmd and systemctl instead of firewall-cmd because firewalld is not fully online during the boot process.

• We also added a line at the end to pass a basic webpage to Apache. This is to suppress warnings when we add a load balancer later.

Click Create instance.

To verify that the webserver is running, enter its public IP address into your browser’s address bar. It might take a few minutes after the instance says it is running. You should get the following reply from the webserver:

This is myoracle.com!

Now that you have manually provisioned a compute instance and automatically configured it through cloud-init, you can fully automate provisioning behind a load balancer.

**Create Private Load Balancer and Autoscale Compute Pools**

you’ll create a private load balancer and distribute the web traffic to an automatically scaling pool of VMs. You will create an instance configuration and pool based on an instance. You’ll then create an autoscaling configuration that will add an instance to that pool when a CPU threshold is hit.

Provision a Client Machine

For this part of the lab, you will place the load balancer and Apache hosts in private subnets. To access them, you will SSH into a machine in the public subnet that will then act as the HTTP client. It will use curl instead of a browser to retrieve the webpage.

Create VM VM-CLIENT

Provision a Load Balancer

6. Click Create Load Balancer at the top of the table. It will open a dialog box and ask you to select a load balancer type. Leave Load Balancer selected. Click Create Load Balancer at the bottom of the dialog box. This will take you to the load balancer creation form.

a. The first page of the form will be for configuring the load balancer itself.

1) For the Name, enter <REGION>-OP-LAB01-1-LB-01.

2) For Visibility Type, select Private.

3) Leave the Bandwidth section as the default.

4) Leave Enable IPv6 Address Assignment deselected.

5) In the Choose Networking box:

a) Select the VCN you created earlier. <REGION>-OP-LAB01-1-VCN-01

b) Select the first private subnet you created earlier. <REGION>-OP-LAB01-1-SNT-PRIVATE1

6) Click Next

b. The second page of the form will be for configuring the backends of the load balancer.

1) Specify Weighted Round Robin for the Load Balancing Policy.

2) Skip Add Backends since you’ll add a backend later.

3) Leave the box for Specify Health Check Policy as the default.

4) Leave Use SSL deselected.

5) Click Show Advanced Options. For the Backend Set Name, enter <REGION>-OP-LAB01-1-LBBS-01

6) Click Next

c. The third page of the form will be for configuring the listener.

1) For Listener Name, enter <REGION>-OP-LAB01-1-LSN-01

2) Change Listener traffic type to HTTP.

3) Leave the listener’s port at 80.

4) Click Next to go to the next page.

d. The fourth page will be to configure logging.

1) Disable Error Logs.

2) Leave Access Logs disabled.

3) Click Submit. Ignore the high-risk warning from Smart Check.

Create an Instance Configuration

7. Next, you will create an instance configuration to create the instance pool that you will attach to the load balancer. Navigate to the instance configuration list:

a. In the navigation menu, click Compute, then Instance Configurations.

8. Click Create instance configuration.

a. For Name, enter <REGION>-OP-LAB01-1-INST-CF

b. For Create in compartment, leave your compartment selected.

c. For Compartment to create instances in, leave your compartment selected.

d. In the Placement box, select AD 2 for the second availability domain.

e. In the Image and Shape box: Oracle Linux 8. Ampere VM.Standard.A1.Flex. (1,6)

f. In the Networking box:

1) Under Primary Network, choose Select existing virtual cloud network.

2) Under Virtual cloud network, select the VCN you created earlier. <REGION>-OP-LAB01-1-VCN-01

3) Subnet - Select existing subnet and choose the second private subnet <REGION>-OP-LAB01-1-SNT-PRIVATE2

4) Do not assign a public IPv4 address should now be selected.

g. Under the Add SSH keys box, upload or paste your Cloud Shell machine’s public key.

h. Leave the Boot volume options as the default.

i. Skip Block volumes.

j. Click Show advanced options.

1) Under the Management tab, choose Paste cloud-init script.

2) Copy the following commands:

#!/bin/sh

sudo dnf install httpd --assumeyes --quiet

sudo systemctl enable httpd

sudo systemctl start httpd

sudo firewall-offline-cmd --add-service=http

systemctl restart firewalld

echo '<!doctype html><html><body><h1>This is myoracle.com!</h1></body></html>' | sudo tee /var/www/html/index.html

k. Click Create. This will take you to the instance configuration’s details page.

Create an Instance Pool

9. Now, you can create an instance pool based on this instance configuration. Navigate to instance pool list.

click Compute. - Instance Pools. Click Create instance pool.

a. The first page will be for basic details.

1) For Name, enter <REGION>-OP-LAB01-1-INST-PL

2) Ensure that your compartment is selected.

3) For Instance configuration, select the instance configuration you created (<REGION>-OP-LAB01-1-INST-CF)

4) Set the number of instances to 1.

5) Click Next.

b. The second page will be for instance placement.

1) Under Availability domain selection 1:

a) Select AD 1 for the Availability domain.

b) Leave Fault domains as is.

c) Select your VCN (<REGION>-OP-LAB01-1-VCN-01).

d) Select the second private subnet (<REGION>-OP-LAB01-1-SNT-PRIVATE2).

2) Click + Another availability domain.

a) Select AD 2 for the Availability domain.

b) Leave Fault domains as is.

c) Select your VCN (<REGION>-OP-LAB01-1-VCN-01).

d) Select the second private subnet (<REGION>-OP-LAB01-1-SNT-PRIVATE2).

3) Select Attach a load balancer.

4) Under the Load balancer 1 box:

a) Leave Load balancer as the Load balancer type.

b) Select the load balancer you created earlier (<REGION>-OP-LAB01-1-LB-01).

c) Select the load balancer’s backend set (<REGION>-OP-LAB01-1-LBBS-01).

d) For Port, enter 80.

e) For VNIC, select Primary VNIC.

5) Click Next.

c. Review your configuration and click Create. This will take you to the instance pool’s details page.

Create an Autoscaling Configuration

10. You now have a load balancer with an instance pool in its backend. To set the instance pool to automatically scale, you’ll need to create an autoscaling configuration. Navigate to the autoscaling configuration list:

a. Compute. - Autoscaling Configurations.

11. Click Create autoscaling configuration.

a. The first page will be for basic details.

1) For Name, enter <REGION>-OP-LAB01-1-AS-CONFIG

2) Choose your compartment.

3) Select your instance pool (<REGION>-OP-LAB01-1-INST-PL). It may take a few minutes for it to appear.

4) Click Next.

b. On the Configure autoscaling policy page:

1) Ensure that Metric-based autoscaling is selected.

2) In the Configure autoscaling policy box:

a) Name the policy <REGION>-OP-LAB01-1-AS-POL

b) Leave the Cooldown in Seconds as 300. This is the minimum period of time between scaling actions.

c) Select CPU utilization for Performance Metric.

d) Enter the following for the Scale-out rule:

i) Scale-out Operator: Greater than (>)

ii) Threshold percentage: 75

iii) Number of instances to add: 1

e) Enter the following for the Scale-in rule:

i) Scale-in operator: Less than (<)

ii) Threshold percentage: 25

iii) Number of instances to remove: 1

f) Enter the following for the Scaling limits:

i) Minimum number of instances: 1

ii) Maximum number of instances: 2

iii) Initial number of instances: 1

3) Click Next.

c. Review your configuration and click Create. This will take you to the autoscaling configuration’s details page.

12. Navigate back to your instance pool’s details page.

a. Click Compute - Instance Pools

b. Wait for your instance pool to change state from Scaling to Running.

c. Click the pool’s name (<REGION>-OP-LAB01-1-INST-PL) to go to its details page.

13. Here, you will be able to view the instance pool’s instances. On the left navigation pane under Resources, select Attached instances. You should see one instance listed in the table.

Test the Autoscaling Configuration

14. Next, you will simulate high load on the instance pool to test the autoscaling configuration. To simulate load, you will SSH into the machine and run a common Linux stress-testing tool, stress-ng.

Note: Recall that the instance pool is in a private subnet. This means that we cannot access it directly. Instead, we will use the VM in the public subnet as both a bastion to SSH into the private instances and an HTTP client to test Apache on the private instances. For the rest of the document, we will refer to the instance in the public subnet as the client and the instance in the private subnet as the server.

a. Open Cloud Shell

b. SSH into the server through the client with the following command. The inner proxy command connects to the public VM, then the outer command connects to the private VM. Replace the angle bracket labels with the appropriate content.

$ ssh -o ProxyCommand=\ssh -W %h:%p -p 22 opc@<client-public-ip>\ -p 22 opc@<server-private-ip>

c. Enter yes twice to confirm that you want to SSH into both machines.

d. Run the following commands in succession to make the CPU perform matrix multiplication on all cores (only one in this instance) for 15 minutes.

$ sudo dnf install stress-ng --assumeyes

$ stress-ng --matrix 0 -t 15m > /dev/null 2>&1 &

$ top

Note: The code > /dev/null 2>&1 & suppresses the output and makes stress-ng run in the background. This way, you can view the running process with top.

15. Click Compute to also view the CPU utilization and Memory utilization graphs.

16. Wait 5-10 minutes for the total instance count to increase. If you go to the Instance Pools area in Compute, you’ll see a second instance has been created and your instance pool is in a Scaling state.

17. If you are still viewing top in Cloud Shell, press q to quit, then type exit to end the SSH session.

18. To test the webservers behind the load balancer. Log in to your client VM using SSH in Cloud Shell and use curl to retrieve the home page:

$ ssh opc@<client-public-ip-address>

$ curl http://<private-load-balancer-ip-address>

You should get the webpage in response:

<!doctype html><html><body><h1>This is myoracle.com!</h1></body></html>

**Create a Private Zone Using DNS Management**

you’ll add a private Domain Name System (DNS) zone to your VCN to use hostnames instead of IP addresses.

Create a Private Zone

1. . Under DNS Management, click Zones.

2. Click the Private Zones tab. You should see zones for your existing subnets.

3. Click Create Zone to open the zone creation dialog box.

a. For Zone Name, enter myoracle.com

b. Ensure that your compartment is selected.

c. Leave the Zone Type as the default

d. Choose Selecting existing DNS Private View and select your VCN’s private view (<REGION>-OP-LAB01-1-VCN-01). This will allow machines in your VCN to resolve DNS names in this private zone.

e. Skip the advanced options.

f. Click Create. This will take you to the zone’s details page.

Add a Zone Record

4. You should still be on the zone’s details page. If not, navigate to the zones in the main

menu under Networking and Zones. Click the Private Zones tab. Click your zone’s name.

5. Scroll down and click Add Record.

6. In the Add Record dialog box,

a. Select a record type: A - IPv4 Address from the drop-down list.

b. Name: Leave blank

c. Click the lock icon at the right to unlock the TTL fields.

d. TTL: 30

e. TTL Unit: Seconds

f. Rdata Mode: Basic

g. Address: <private-load-balancer-ip-address>

7. Click Submit.

8. Repeat Step 4 with following data:

a. Select a record type: CNAME from the drop-down list.

b. Name: web

c. TTL: 30

d. TTL Unit: Seconds

e. Rdata Mode: Basic

f. Target: myoracle.com

9. Click Submit.

10. Once your records have been added, click Publish Changes.

11. In the confirmation dialog box, click Publish Changes.

Test the Server

1. To test whether Private DNS is working, log in to your client VM using SSH in Cloud Shell and enter following commands in succession.

$ ssh opc@<client-machine-public-ip>

$ curl -k http://myoracle.com

$ curl -k http://web.myoracle.com

2. Both curl commands should return this:

<!doctype html><html><body><h1>This is myoracle.com!</h1></body></html>

3. Exit the SSH session:

$ exit

**Configure Front-end SSL with Load Balancing**

In the previous sections, you connected to the webserver(s) over HTTP. To use HTTPS, you’ll need to pass an RSA key pair to the load balancer and create a certificate to bind the public key to the hostname. In this practice, you will generate an RSA key pair and a self-signed certificate. Note that no major browser (nor curl) will trust actually a self-signed certificate, but you will use one for the purposes of this lab.

Create an SSL/TLS Certificate

1. Open Cloud Shell

2. Use the following command to generate an RSA key pair and associated certificate:

$ openssl req -x509 -newkey rsa:4096 -keyout key.pem -out cert.pem -sha256 -days 365 -nodes -subj '/CN=myoracle.com'

3. Enter ls and you should see both cert.pem and key.pem.

4. Download both cert.pem and key.pem:

Add the Certificate to the Load Balancer

1. In the navigation menu, select Networking, and then click Load Balancers.

2. Make sure your compartment is selected on the left under List Scope.

3. Click the name of your load balancer to go to its details page (<REGION>-OP-LAB01-1-LB-01).

4. Under Resources on the left, click Certificates (0).

5. Above the certificates table, there should be a field titled Certificate Resource. Use the drop-down to select Load Balancer Managed Certificate.

6. Click Add Certificate to open a new dialog box.

a. For Certificate Name, enter <REGION>-OP-LAB01-1-CERT.

b. Select Choose SSL Certificate File and upload cert.pem

c. Select the box for Specify Private Key and upload key.pem

d. Click Add Certificate and close the work request notification that will pop up.

7. Wait for the certificate to appear in the table.

Create an HTTPS Listener

1. You should still be on your load balancer’s details page. If not, navigate to it.

2. Under Resources on the left, click Listeners. The table should list the listener you created earlier to listen for HTTP (<REGION>-OP-LAB01-1-LSN-01)

3. Next, you will create a listener for HTTPS traffic. Click Create the listener.

a. Name: <REGION>-OP-LAB01-1-LSN-02

b. Protocol: HTTPS

c. Port: 443

d. Certificate Resource: Load Balancer Managed Certificate

e. Select the certificate: <REGION>-OP-LAB01-1-CERT

f. Backend set: Choose your backend set, <REGION>-OP-LAB01-1-LBBS-01

g. Idle Timeout in Seconds (optional): 60

h. Routing Policy (optional): Keep default

4. Click Create Listener and then click Close on the work request notification. Wait for the listener to appear in the table.

5. To test if the certificate is working, log in to the client VM (<REGION>-OP-LAB01-1-VM-

CLIENT) using SSH in Cloud Shell. Return to the Instance Details page in Compute if you need the IP address.

$ ssh opc@<client-public-IP-address>

6. Use curl to send an HTTPS request for the webpage. The -k flag is needed to prevent curl from requiring a verified (non-self-signed) certificate.

$ curl -k https://myoracle.com

You should get the following webpage as a response:

<!doctype html><html><body><h1>This is myoracle.com!</h1></body></html>

7. Repeat the curl command for the CNAME.

$ curl -k https://web.myoracle.com

You should get the same webpage as a response.

**Common Operational Activities: Compare Storage Types**

**Lab 2-1 Practices**

Overview

In this lab, you’ll create and compare three different types of storage: block volumes, file

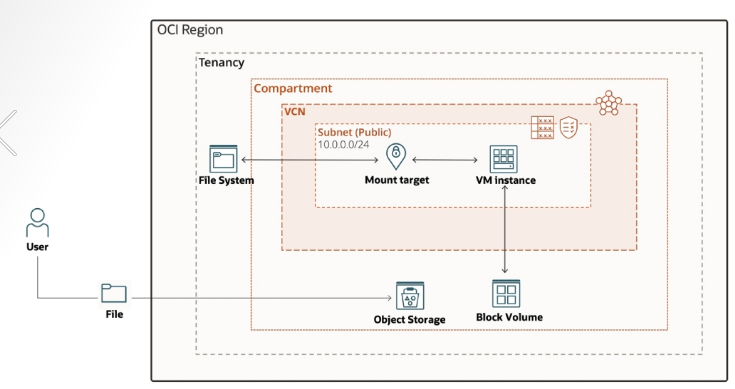
systems, and Object Storage buckets.

In this lab, you’ll:

a. Create a block volume and attach it to a compute instance.

b. Create a file system, export it, and mount in on a host.

c. Create a bucket and upload a file to it.



Before you begin the lab, you’ll need to create a VCN and compute instance.

**Create, Attach, and Mount a Block Volume to a Compute Instance**

Block volumes are often used to add storage capacity to an Oracle Cloud Infrastructure instance. You can create a block volume through the Cloud Console or an API after you have created a VCN and instance. Once created, the volume can be attached to an instance using a volume attachment. You can then connect to the volume from your instance's guest OS using iSCSI (which you will use in this lab) or paravirtualized mode. The volume can then be mounted and used by your instance.

Provision the Block Volume

1. In the navigation menu, navigate to Storage and click Block Storage.

2. Click Create Block Volume and provide these details:

• Name: IAD-OP-LAB02-1-BV-01

• Compartment: Your compartment

• Availability Domain: It must be the same AD as your instance (AD 1).

• Volume size: Choose a custom size of 50 GB.

• Target Volume Performance : Leave everything in this box as the default.

• Backup Policy: Select the No Backup Policy. This would configure automatic backups.

Note: There are three predefined backup policies: Bronze, Silver, and Gold. Each backup policy has a set backup frequency and retention periods. Bronze includes monthly incremental backups (12-month retention). Silver includes weekly (four-week retention) and monthly incremental backups. Gold includes daily (seven-day retention), weekly, and monthly incremental backups. All full backups are retained for five years.

3. Leave all other options (including cross-region replication as OFF) as their default values and click Create Block Volume. The volume will be ready to attach once its icon no longer lists it as Provisioning in the volume list.

Attach the Block Volume

Once the block volume is created, you can attach it to the VM instance you launched. When you attach a block volume to a VM instance, you have two options for attachment type, iSCSI or paravirtualized.

4. You can attach block volumes either from the block volume’s page or the compute

instance’s page. You’ll do it from the compute instance’s page for this lab.

a. Go to Compute, then Instances in the navigation menu.

b. Click your VM instance to go to its details page.

c. Click Attached block volumes to open the block volumes list.

5. Click Attach block volume.

a. Click Select volume and choose your volume (IAD-OP-LAB02-1-BV-01).

b. Device Path: Select /dev/oracleoci/oraclevdb

c. For Attachment mode, choose Paravirtualized. Relative to ISCSI, this requires more virtualization, but is simpler to work with.

d. For Access, choose Read/write. This will allow only one instance to attach to the block volume.

e. Click Attach.

Format and Mount the Device in the Operating System

Now that the hardware is virtually attached, you need to configure the operating system of the VM to work with the volume.

6. Open Cloud Shell

7. Connect to the instance through SSH. The public IP address of the instance is listed on its details page.

ssh opc@<public\_ip\_address>

8. Enter yes to confirm.

9. Once the disk is attached, run these commands to format the disk and mount it.

a. List the devices. You should see a device attached to oraclevdb.

$ ls -l /dev/oracleoci/oraclevd\*

b. Make an ext4 filesystem on the drive.

$ sudo mkfs -t ext4 /dev/oracleoci/oraclevdb

# Press y if prompted

c. Make a directory to mount the device on.

$ sudo mkdir /mnt/IAD-OP-LAB02-1-BV

d. Mount the device to the directory.

$ sudo mount /dev/oracleoci/oraclevdb /mnt/IAD-OP-LAB02-1-BV

e. Change directory and list the contents.

$ cd /mnt/IAD-OP-LAB02-1-BV

$ ls -l

f. List the disk space.

$ df -h

You should see a ~50G drive on /mnt/IAD-OP-LAB02-1-BV

10. Create a file on the disk.

$ sudo touch /mnt/IAD-OP-LAB02-1-BV/example\_file.txt

**Create a File System, Export It, and Mount in on a Host**

In this practice, you will create a file system in File Storage. When you create this file system, the Cloud Console will automatically create an associated mount target and export. A mount target is an NFS endpoint in a subnet of your choice and allows you to access your file system. It provides the IP address or DNS name used in the mount command when connecting NFS clients to a file system. An export controls how an endpoint accesses a file system and includes an export path that uniquely identifies that file system, so that many file systems can exist on a single mount target. A file system must have at least one export in one mount target for instances to mount the file system.

Open the Subnet to NFS

First, you need to create a security list that will allow the file storage system to accept connections to and from the subnet. For the purposes of this lab, the mount target and the compute instance must be in the same subnet for these rules to work.

1. In the main menu, navigate to Networking, then Virtual Cloud Network. Click into your

VCN, IAD-OP-LAB02-1-VCN-01.

2. In the left navigation pane, click Security Lists, then click your default security list.

3. Create the following ingress rules. All egress should already be allowed.

a. Stateful ingress from all ports in the source CIDR block (10.0.0.0/24) to TCP port 111

and ports 2048-2050

b. Stateful ingress from all ports in the source CIDR block (10.0.0.0/24) to UDP ports 111

and port 2048

Provision a File System

4. Open the navigation menu and click Storage. Under File Storage, click File Systems.

5. Make sure your correct compartment is listed at the left.

6. Click Create File System.

Note: File systems are encrypted by default. You cannot turn off encryption.

1. You can choose to accept the system defaults or change them by clicking Edit Details. Edit the file system name to be IAD-OP-LAB02-1-FS. Leave the rest of this area as default (AD, compartment, encryption).

Note on Encryption: File systems use Oracle-managed keys by default, which leaves all encryption-related matters to Oracle. Optionally, you can encrypt the data in this file system using your own Vault encryption key. Currently, only symmetric Advanced Encryption Standard (AES) keys are supported for file system encryption.

1. Note the export path, which is the file system name /IAD-OP-LAB02-1-FS. *(Export path: /FileSystem-20240222-1925-20)*
2. Create a new mount target and edit the name to be IAD-OP-LAB02-1-MNT-01 by clicking Edit Details at the right. Your pre-created VCN (IAD-OP-LAB02-1-VCN-01) and public subnet should be listed by default.
3. Click Create. Your file system, export and export path, and mount target will all be created.

7. Click your newly created file system, and then click Exports in the left navigation pane.

8. Find the export you just created, click the Actions menu (three vertical dots on the right), and then click Mount Commands.

9. In Image, ensure that Oracle Linux is selected.

10. Keep these commands open or note them in a text editor.

Top of Form

Image

Oracle LinuxUbuntuDebianCentOSRed Hat Linux

Command to install NFS client

sudo yum install nfs-utils

Command to create the mount point directory

sudo mkdir -p /mnt/FileSystem-20240222-1925-20

Command to mount the File System

sudo mount 10.0.0.121:/FileSystem-20240222-1925-20 /mnt/FileSystem-20240222-1925-20

Bottom of Form

Mount the File System to the Instance

11. Open Cloud Shell if it is not already open. SSH into the instance if you are not already connected.

$ ssh opc@<public\_ip\_address>

12. Install the NFS client by copying and pasting the Command to install NFS client from the Console or typing the following:

$ sudo yum install nfs-utils

13. Create a mount point by copying and pasting the Command to create the mount point directory from the Console or type the following.

$ sudo mkdir -p /mnt/IAD-OP-LAB02-1-FS

14. Mount the file system by copying and pasting the Command to mount the file system from the Console or type the following. Replace <private\_ip> with the private IP of the instance.

$ sudo mount <private\_ip>:/IAD-OP-LAB02-1-FS /mnt/IAD-OP-LAB02-1-FS

5. Create a file on the file system.

$ sudo touch /mnt/IAD-OP-LAB02-1-FS/example\_file.txt

sudo touch /mnt/ FileSystem-20240222-1925-20/example\_file.txt

16. You can see both the example file on the block volume and the example file on the file system:

$ sudo ls -R /mnt

**Create a Bucket and Upload a File to It**

Object Storage can store an unlimited amount of unstructured data of any content type, including analytic data and rich content such as images and videos. A regional service, Object Storage isn’t tied to any specific compute instance. You can access an Object Storage endpoint (for example, https://objectstorage.us-ashburn-1.oraclecloud.com in the Ashburn/IAD region) through the service gateway you created in your VCN. Object Storage buckets are logical containers for storing objects. Users or systems create buckets as needed within a region. In this practice, you’ll create an Object Storage bucket and upload a file to it.

Create a Bucket

1. In the navigation menu, under Storage and Object Storage, click Buckets.

2. Ensure your compartment is selected on the left.

3. Click Create Bucket.

4. In the Create Bucket dialog box, specify the attributes of the bucket:

a. Bucket Name: <your\_username>-IAD-OP-LAB02-1-BKT-01

Note: Your bucket name has to be unique within the entire tenancy. If you run into a conflict, add numbers to the end.

b. Default Storage Tier: Standard

c. Encryption: Encrypt using Oracle-managed keys

A few notes on bucket options:

Storage tier types:

− Standard is the default storage tier used for Object Storage service data and

allows for immediate access.

− Archive is the default storage tier used for Archive Storage service data.

Do not select the other options, except for Uncommitted Multipart Uploads Clean-up, which will delete uncommitted multipart uploads.

Encryption: Buckets are encrypted with keys managed by Oracle by default. You can optionally encrypt the data in this bucket using your own Vault encryption key, though that option does not exist for the purpose of this lab.

5. Click Create. The bucket is created immediately, and you can start uploading objects.

Create a Pre-Authenticated Request

To upload or download objects from your local machine, you could simply click upload or download on the bucket’s details page, similar to any other website that involves uploading or downloading content. For this lab, you will upload and download from the compute instance that you provisioned earlier. There are two main methods to do this: pre-authenticated requests (PARs) and the OCI CLI. This lab will use PARs for simplicity, but the OCI CLI would be more secure.

6. In the left navigation panel, click Pre-Authenticated Requests.

7. Click Create Pre-Authenticated Request above the table.

a. Name the PAR: IAD-OP-LAB02-1-PAR-01

b. For Pre-Authenticated Request Target, select Bucket.

c.

For Access Type, select Permit object reads and writes.

d. Keep Enable Object Listing deselected.

e. Leave the expiration date as the default.

f.

Click Create Pre-Authenticated Request at the bottom to confirm.

8. Save the PAR URL to a text file.

Pre-Authenticated Request URL

<https://objectstorage.us-ashburn-1.oraclecloud.com/p/p0tV2PkRWhuc6zjVtD13KXeaAqr9gUKr6w_5D-jKJhU_EyD2HWkOVsQO1bVkXRKj/n/ocuocictrng30/b/BKT-01/o/>

The current URL is deprecated and will no longer be supported in a future release of the console. A new URL will be used as shown below.

https://ocuocictrng30.objectstorage.us-ashburn-1.oci.customer-oci.com/p/p0tV2PkRWhuc6zjVtD13KXeaAqr9gUKr6w\_5D-jKJhU\_EyD2HWkOVsQO1bVkXRKj/n/ocuocictrng30/b/BKT-01/o/

Upload and Download to a Bucket from a VM Using a PAR

9. Open Cloud Shell if it is not already open. SSH into the instance if you are not already

connected.

$ ssh opc@<public\_ip\_address>

10. Change to your home directory.

$ cd ~

11. Make a file to upload and download.

$ echo \Some example data\ > example\_file.txt

$ ls

. Upload the file to Object Storage using an HTTP call. Substitute in the URL for the PAR

that you recorded earlier.

$ curl -X PUT --data-binary '@/home/opc/example\_file.txt' <PAR\_URL>example\_file.txt

curl -X PUT --data-binary '@/home/opc/example\_file.txt' <https://objectstorage.us-ashburn-1.oraclecloud.com/p/p0tV2PkRWhuc6zjVtD13KXeaAqr9gUKr6w_5D-jKJhU_EyD2HWkOVsQO1bVkXRKj/n/ocuocictrng30/b/BKT-01/o/>example\_file.txt

13. Look in the Console on your bucket’s details page for the example file.

14. Delete the file from the compute instance.

$ rm example\_file.txt

15. Download the file from Object Storage using an HTTP call. Substitute in the URL for the PAR that you recorded earlier.

$ curl -X GET <https://objectstorage.us-ashburn-1.oraclecloud.com/p/p0tV2PkRWhuc6zjVtD13KXeaAqr9gUKr6w_5D-jKJhU_EyD2HWkOVsQO1bVkXRKj/n/ocuocictrng30/b/BKT-01/o/>example\_file.txt

You should see the sample data you stored.

**Set Up Block Storage Disaster Recovery**

**Lab 3-1 Practices**

Overview

The block volume service offers a high level of durability—all volumes replicated across multiple storage servers with built-in repair mechanisms. However, block volume backups and replicas are still recommended for disaster recovery and business continuity for certain scenarios, such as application failure, availability domain failure, or operator error.

In this lab, you’ll:

a. Create and attach a block volume to a compute instance.

b. Enable regular backups for the block volume.

c. Enable cross-region replication of the block volume and add data.

d. Activate the volume replica in the destination region.

e. Attach the volume replica to a compute instance in the destination region and view the data.

A diagram of a diagram

Description automatically generated

**Create a Compute Instance in Cloud Console, CLI, and SDK**

**Lab 4-1 Practices**

Overview

In this lab, you will provision a compute instance using three different methods to compare

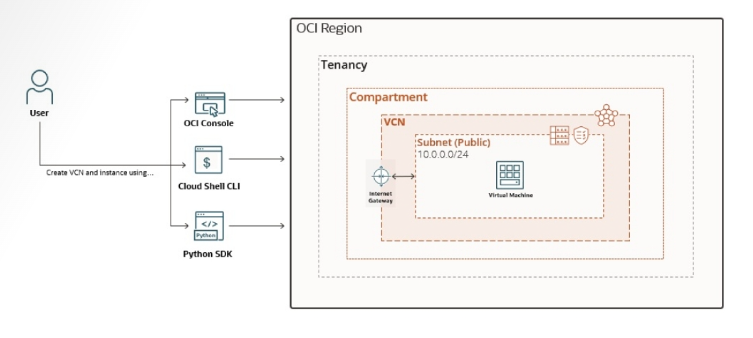
their functionality.

In this lab, you’ll:

a. Create a VM instance in the Cloud Console.

b. Create a VM instance using Cloud Shell in the CLI on the Cloud Console.

c. Create a VM instance using a Python SDK in Cloud Editor and Cloud Shell.



Create VCN, SSH Keys

**Create a VM Instance Using Cloud Shell Command Line Interface**

In this practice, you’ll use the CLI to access Oracle Cloud Infrastructure and carry out service-related tasks in the VCN you created.

Use the OCI CLI Help Option

1. Launch Cloud Shell

2. Most OCI CLI commands must specify a service, followed by a resource type and then an action. The basic command-line syntax is:

$ oci <service> <type> <action> <options>

For this practice, this syntax is applied as follows:

• Compute is the <service>.

• Instance is the resource <type>.

• Launch is the <action>.

• The rest of the command string consists of <options>.

3. To see the information for each command, use the --help option. For example, try the following:

$ oci --help

$ oci network --help

$ oci network vcn --help

$ oci network vcn list --help

Get Input Templates from the OCI CLI

4. Some options, such as --shape-config for launching a compute instance, take a JSON string as input. To see the form of this JSON string, use the --generate-param-json-input option:

$ oci compute instance launch --generate-param-json-input shape-config

We will use this template later.

Launch the Compute Instance

5. To see what is required to launch an instance, enter:

$ oci compute instance launch -h

6. Enter the OCI CLI interactive mode:

$ oci -i

7. You can use interactive mode to fill in details such as availability domain and compartment OCID by pressing the down arrow; for example, type the following without pressing enter the following with a trailing space:

$ oci compute instance launch --compartment-id

8. Wait a moment, then you can use the up and down arrows to find your compartment.

Press Enter and the CLI will autofill the OCID.

9. Enter the following in the OCI CLI interactive mode. Lines are separated by backslashes,

but do not actually separate them; enter the following as one line. They are separated for

display purposes. Compartment OCID, availability domain, subnet OCID, and image ID

should all be filled using interactive mode’s autofill feature.

$ oci compute instance launch \

--compartment-id <The ID of your compartment> \

--availability-domain <The name of the first AD> \

--subnet-id <The ID of \Public Subnet-IAD-OP-LAB04-1-VCN-01\>\

--display-name \IAD-OP-LAB04-1-VM-02\ \

--image-id <Latest Oracle Linux 8 Image for ARM (aarch64)> \

--shape \VM.Standard.A1.Flex\ \

--shape-config '{\ocpus\: \1\, \memoryInGbs\: \6\}' \

--assign-public-ip true \

--ssh-authorized-keys-file \/home/<shell-user>/.ssh/id\_rsa.pub\

For example, your command may look like this:

$ compute instance launch --compartment-id

ocid1.compartment.oc1..aaaaaaaa4zmhdgjpun266xpps4imcfroe42oxxa24lwv

y4wxxeimgbuafguq --availability-domain CPQY:US-ASHBURN-AD-1 --

subnet-id

ocid1.subnet.oc1.iad.aaaaaaaaabl3sc36muka6dlf5y7hcanvdomss6jthal6kp

h3j5whv7pkq74q --display-name \IAD-OP-LAB04-1-VM-02\ --image-id

ocid1.image.oc1.iad.aaaaaaaabta3gqhlfoam3l7rayd4inr6hhg4cmvcqb7p43q

adkiehnszyuka --shape VM.Standard.A1.Flex --shape-config '{\ocpus\:

\1\, \memoryInGbs\: \6\}' --assign-public-ip true– ssh -authorized-

keys-file \/home/x\_28012023/.ssh/id\_rsa.pub\

My code

x\_99070595@cloudshell:~ (us-ashburn-1)$ oci compute instance launch --generate-param-json-input shape-config

{

"baselineOcpuUtilization": "string",

"memoryInGBs": 0.0,

"nvmes": 0,

"ocpus": 0.0,

"vcpus": 0

}

x\_99070595@cloudshell:~ (us-ashburn-1)$ oci compute instance launch --compartment-id ocid1.compartment.oc1..aaaaaaaaiugoxedsjbk7kyvu724jaa6khknzvbtu5fdwftwbcfqbbve2i5ga --availability-domain ZIDs:US-ASHBURN-AD-1 --subnet-id ocid1.subnet.oc1.iad.aaaaaaaarjutobbu25y345sw6f3hwuqpxpisobefubcpprrne6fyggqkkehq --display-name “IAD-OP-LAB04-1-VM-02” --image-id ocid1.image.oc1.iad.aaaaaaaabta3gqhlfoam3l7rayd4inr6hhg4cmvcqb7p43qadkiehnszyuka --shape VM.Standard.A1.Flex --shape-config '{"ocpus": "1", "memoryInGbs": "6"}' --assign-public-ip true --ssh-authorized-keys-file "/home/x\_99070595/.ssh/id\_rsa.pub"

Record the instance OCID to use in the next few steps.

ocid1.instance.oc1.iad.anuwcljrusk3myic4umtx7apqmamdg3u5oqs62txp4t5dtgojswi2wdtk5kq

10. Navigate to compute instances in the navigation menu under Compute > Instances. You

should see an instance provisioning.

11. Get information about compute instance.

$ oci compute instance get --instance-id <instance-OCID>

*oci compute instance get --instance-id* *ocid1.instance.oc1.iad.anuwcljrusk3myic4umtx7apqmamdg3u5oqs62txp4t5dtgojswi2wdtk5kq*

12. Stop the compute instance.

$ oci compute instance action --action STOP --instance-id *ocid1.instance.oc1.iad.anuwcljrusk3myic4umtx7apqmamdg3u5oqs62txp4t5dtgojswi2wdtk5kq*

13. Terminate the compute instance.

$ oci compute instance terminate --instance-id *ocid1.instance.oc1.iad.anuwcljrusk3myic4umtx7apqmamdg3u5oqs62txp4t5dtgojswi2wdtk5kq*

14. Enter y to confirm.

15. Ensure that the compute instance has terminated under Compute > Instances in the

console.

**Create VM Instance Using Python SDK**

Next, you’ll create a VM instance using Python SDK. First, you’ll need to create a folder and file using Code Editor in the Cloud Console. You’ll then create the VM using a Python script.

Create a Folder and File in Code Editor

1. Open the console’s Code Editor, whose icon is at the top-right corner, to the right of the

CLI Cloud Shell icon.

2. Expand the Explorer panel with the top icon on the left panel. It looks like two overlapping

documents.

3. Expand the drop-down for your home directory if it isn’t already expanded. It’s okay if it is

empty.

4. Create a new folder by clicking File, then New Folder, and name it sdk\_example.

5. Create a file in that folder by clicking File, then New File, and name it

provision\_compute.py. This will also open the file.

Create the VM Using Python Script

1. Enter the following code:

import oci

# Load your tenancy, user, etc. information

config = oci.config.from\_file()

# Variables to fill in

compartment\_id = \\ # Fill with your compartment OCID

image\_id = \\ # Fill with the Oracle Linux 8 OCID for your region

subnet\_id = \\ # Fill with your public subnet OCID

display\_name = \IAD-OP-LAB04-1-VM-3\ # Fill with your display name

# Get first availability domain name

iam = oci.identity.IdentityClient(config)

availability\_domain =

iam.list\_availability\_domains(compartment\_id).data[0].name

# Package all settings together

launch\_instance\_details = oci.core.models.LaunchInstanceDetails(

availability\_domain=availability\_domain,

compartment\_id=compartment\_id,

display\_name=display\_name,

shape=\VM.Standard.A1.Flex\,

shape\_config=oci.core.models.LaunchInstanceShapeConfigDetails(

ocpus=1, memory\_in\_gbs=6),

source\_details=oci.core.models.InstanceSourceViaImageDetails(

image\_id=image\_id, source\_type=\image\),

create\_vnic\_details=oci.core.models.CreateVnicDetails(

assign\_public\_ip=True, subnet\_id=subnet\_id)

)

# Launch compute instance

compute = oci.core.ComputeClient(config)

compute.launch\_instance(launch\_instance\_details)

Fill in the required variables.

a. For compartment OCID, navigate to Identity & Security > Compartments (minimize

Code Editor) and find your compartment in the table.

b. For the image OCID, find the corresponding OCID for your region here:

https://docs.oracle.com/en-us/iaas/images/image/aaca995d-5c38-4d7b-9d9e-

0f4c3e05a66c/

c. For subnet OCID, navigate to Networking > Virtual Cloud Networks (minimize code

editor) and click into your VCN. Click the public subnet to go to its details. Find the OCID.

d. For display name, enter IAD-OP-LAB04-1-VM-03.

3. Save the file with by going to File, then clicking Save.

4. Open a new terminal by going to Terminal, then clicking New Terminal.

5. In the terminal, enter the following:

$ python3 sdk\_example/provision\_compute.py

6. Navigate to compute instances. You should see your compute instance provisioning.

7. Click the compute instance’s name to go to its details page.

**Automate Backing Up a File to Object Storage**

**Lab 5-1 Practices**

**Resource & Configuration Management: Create a Reusable VCN Configuration with Terraform**

**Lab 6-1 Practices**

Overview

There are multiple ways to create a VCN and subnet in the Oracle Cloud Console. Particularly if you want to launch several VCNs with the same configuration, it’s beneficial to use Terraform or Resource Manager to streamline and automate that process. Terraform can manage low-level components such as compute, storage, and networking resources, as well as high-level components such as DNS entries and SaaS features. In this lab, you’ll launch and destroy a VCN and subnet by creating Terraform automation scripts

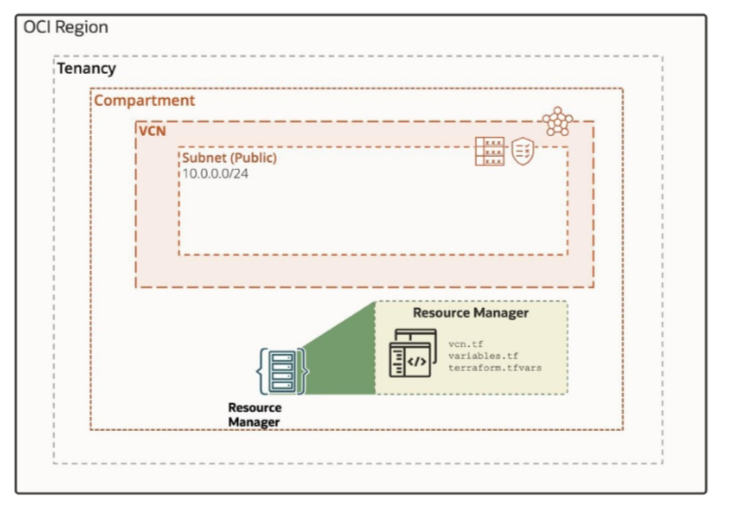
and issuing commands in Code Editor. Next, you’ll download those Terraform scripts and create a stack by uploading them into Oracle Cloud Infrastructure Resource Manager. You’ll then use that service to launch and destroy the same VCN and subnet.

In this lab, you’ll:

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.



**Create a Terraform Folder and File in Code Editor**

In this practice, you’ll create a folder and file to hold your Terraform scripts.

Tasks

1. Log in to your tenancy in the Cloud Console and open the Code Editor

2. Expand the Explorer panel with the top icon on the left panel

3. Expand the drop-down for your home directory if it isn’t already expanded.

4. Create a new folder by clicking File, then New Folder, and name it terraform-vcn.

5. Create a file in that folder by clicking File, then New File, and name it vcn.tf.

6. First, you’ll set up Terraform and the OCI Provider in this directory. Add these lines to the

file:

terraform {

required\_providers {

oci = {

source = “oracle/oci”

version = “>=4.67.3”

}

}

required\_version = “>= 1.0.0”

}

7. Save the changes by clicking File, then Save.

8. Now, run this code. Open a terminal panel in Cloud Editor by clicking Terminal, then New

Terminal.

9. Use pwd to check that you are in your home directory.

10. Enter ls and you should see your terraform\_vcn directory.

11. Enter cd terraform\_vcn/ to change to that directory with.

12. Use

terraform init

to initialize this directory for Terraform.

13. Use ls -a and you should see that Terraform created a hidden directory and file.

**Create and Destroy a VCN Using Terraform**

Terraform uses providers to interface between the Terraform engine and the supported cloud platform. The Oracle Cloud Infrastructure (OCI) Terraform provider is a component that connects Terraform to the OCI services that you want to manage. In this practice, you’ll create a Terraform script that will launch a VCN and subnet. You’ll then alter your script and create two additional files that will apply a compartment OCID variable to your Terraform script.

Write the Terraform

1. Open the OCI Provider documentation in the Terraform Registry https://registry.terraform.io/providers/oracle/oci/latest/docs

to familiarize yourself with the OCI Terraform provider. As you go along the lab, it may be helpful to try and find the relevant portions of the documentation.

2. Add the following code block to your Terraform script to declare a VCN, replacing <your\_compartment\_ocid> with the proper OCID. The only strictly required parameter is the compartment OCID, but you’ll add more later.

resource "oci\_core\_vcn" "example\_vcn" {

  compartment\_id = <your\_compartment\_ocid>

}

This snippet declares a resource block of type oci\_core\_vcn. The label that Terraform will use for this resource is example\_vcn.

In the terminal, run

*terraform plan*

and you should see that Terraform would create a VCN. Because most of the parameters were unspecified, terraform will list their values as “(known after apply). ” You can ignore the “-out option to save this plan” warning for this lab.

Note that *terraform plan* parses your Terraform configuration and creates an execution plan for the associated stack, while *terraform apply* applies the execution plan to create (or modify) your resources.

4. Add a display name and CIDR block (the bolded portion) to the code. Note that we want to set the cidr\_blocks parameter, rather than cidr\_block (which is deprecated). The region code IAD is used below, for the US East (Ashburn) region.

resource "oci\_core\_vcn" "example\_vcn" {

compartment\_id = "<your\_compartment\_ocid>"

**display\_name = "IAD-OP-LAB06-1-VCN-01"**

**cidr\_blocks = ["10.0.0.0/16"]**

}

5. Save the changes and run *terraform plan* again. You should see the display name and CIDR block reflected in Terraform’s plan.

6. Now add a subnet to this VCN. At the bottom of the file, add the following block:

resource "oci\_core\_subnet" "example\_subnet" {

compartment\_id = "<your\_compartment\_ocid>"

display\_name = "IAD-OP-LAB06-1-SNT-01"

vcn\_id = oci\_core\_vcn.example\_vcn.id

cidr\_block = "10.0.0.0/24"

}

Note the line where we set the VCN ID. Here we reference the OCID of the previously declared VCN, using the name we gave it to Terraform: example\_vcn. This dependency makes Terraform provision the VCN first, wait for OCI to return the OCID, then provision the subnet.

7. Run *terraform plan* to see that it will now create a VCN and subnet.

Add Variables

8. Before moving on there are a few ways to improve the existing code. Notice that the subnet and VCN both need the compartment OCID. We can factor this out into a variable. Create a file named variables.tf

9. In variables.tf, declare a variable named compartment\_id:

variable "compartment\_id" {

type = string

}

10. In vcn.tf, replace all instances of the compartment OCID with var.compartment\_id as follows:

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 = "IAD-OP-LAB06-1-VCN-01"

cidr\_blocks = ["10.0.0.0/16"]

}

resource "oci\_core\_subnet" "example\_subnet" {

compartment\_id = var.compartment\_id

display\_name = "IAD-OP-LAB06-1-SNT-01"

vcn\_id = oci\_core\_vcn.example\_vcn.id

cidr\_block = "10.0.0.0/24"

}

Save your changes in both vcn.tf and variables.tf

11. If you were to run terraform plan or apply now, Terraform would see a variable and provide you a prompt to input the compartment OCID. Instead, you’ll provide the variable value in a dedicated file. Create a file named exactly terraform.tfvars

12. Terraform will automatically load values provided in a file with this name. If you were to use a different name, you would have to provide the file name to the Terraform CLI. Add the value for the compartment ID in this file:

compartment\_id = "<your\_compartment\_ocid>"

Be sure to save the file.

13. Run *terraform plan* and you should see the same output as before.

Provision the VCN

14. Run *terraform apply* and confirm that you want to make the changes by entering yes at the prompt.

15. Navigate to VCNs in the console. Ensure that you have the right compartment selected. You should see your VCN. Click its name to see the details. You should see its subnet listed.

Terminate the VCN

16. Run *terraform destroy*. Enter yes to confirm. You should see the VCN terminate.

Refresh your browser if needed.

After destroy terraform.tfstate – remain only 1 code block

A screen shot of a computer

Description automatically generated

Appears terraform.tfstate.backup

A screen shot of a computer program

Description automatically generated

**Create and Destroy a VCN Using Resource Manager**

You can better manage the infrastructure provisioned through Terraform by migrating to Resource Manager instead of running Terraform locally in Cloud Shell or Code Editor. In this section, we will reuse the Terraform code but replace the CLI with Resource Manager.

1. Create a folder named terraform\_vcn on your host machine. Download the vcn.tf,

terraform.tfvars, and variables.tf files from Code Editor and move them to the

terraform\_vcn folder to your local machine. To download from Code Editor, right-click

the file name in the Explorer panel and select Download. You could download the whole

folder at once, but then you would have to delete Terraform’s hidden files.

Create a Stack

2. Navigate to Resource Manager under Developer Services. Go to the Stacks page.

3. Click Create stack.

a. The first page of the form will be for stack information.

1) For the origin of the Terraform configuration, keep My configuration selected.

2) Under Stack configuration, upload your terraform\_vcn folder.

3) Under Custom providers, keep Use custom Terraform providers deselected.

4) Name the stack and give it a description.

5) Ensure that your compartment is selected.

6) Click Next.

b. The second page will be for variables.

1) Because you uploaded a terraform.tfvars file, Resource Manager will auto-

populate the variable for compartment OCID.

2) Click Next.

c. The third page will be for review.

1) Keep Run apply deselected.

2) Click Create. This will take you to the stack’s details page.

Run a Plan Job

4. The stack itself is only a bookkeeping resource—no infrastructure was provisioned yet.

You should be on the stack’s page. Click Plan. A form will pop up.

a. Name the job RM-Plan-01.

b. Click Plan again at the bottom to submit a job for Resource Manager to run terraform plan. This will take you to the job’s details page.

5. Wait for the job to complete, and then view the logs. They should match what you saw when you ran Terraform in Code Editor.

Run an Apply Job

6. Go back to the stack’s details page (use the breadcrumbs). Click Apply. A form will pop up.

a. Name the job RM-Apply-01.

b. Under Apply job plan resolution, select the plan job we just ran (instead of “Automatically approve”). This makes it execute based on the previous plan, instead of running a new one.

c. Click Apply to submit a job for Resource Manager to run terraform apply. This will take you to the job’s details page.

7. Wait for the job to finish. View the logs and confirm that it was successful.

View the VCN

8. Navigate to VCNs in the Console.

9. You should see the VCN listed in the table. Click its name to go to its Details page.

10. You should see the subnet listed.

Run a Destroy Job

11. Go back to the stack’s details page in Resource Manager.

12. Click Destroy. Click Destroy again on the menu that pops up.

13. Wait for the job to finish. View the logs to see that it completed successfully.

14. Navigate back to VCNs in the Console. You should see that it has been terminated.

15. Go back to the stack in Resource Manager. Click the drop-down for More actions. Select

Delete stack. Confirm by selecting Delete.

You’ve now created a Terraform configuration for a VCN; created and destroyed the VCN

through Terraform running locally in Cloud Shell/Code Editor; and created and destroyed the

VCN through managed Terraform in Resource Manager.

**Resource & Configuration Management: Replicate an Existing Environment**

**Lab 7-1 Practices**

Overview

Resource Manager’s resource discovery allows you to generate Terraform based on existing infrastructure. This allows use cases such as manually provisioning infrastructure during a development cycle, then moving to Terraform for a deployment cycle. It also enables use cases such as migrating environments between regions or replicating environments for different purposes (for example, development, QA, or production).

In this lab, you’ll first manually provision a VCN with a compute instance. Then, you’ll use Resource Manager to generate Terraform for that infrastructure. Finally, you’ll use that Terraform to replicate the VCN and compute instance.

In this lab, you’ll:

a. Generate Terraform with Resource Manager.

b. Edit auto-generated Terraform.

c. Provision infrastructure based on the auto-generated Terraform.

A screenshot of a computer

Description automatically generated

Create VCN and Compute Instance

**Replicate an Existing Environment with Resource Manager**

In this practice, you’ll create a stack in Resource Manager based on an existing compartment. You’ll use this stack to generate a Terraform configuration that describes the compartment's resources. Finally, you’ll update the Terraform files to make them reusable.

Create a Stack from Existing Infrastructure

1. Developer Services - Stacks under Resource Manager.

2. Click Create stack.

a. For the origin of the Terraform configuration, select Existing compartment.

b. Under Stack configuration:

1) Select your compartment.

2) Select your region.

3) Choose Selected for Terraform provider services.

4) For Services, select core. Core services include Compute, Block Volume, and Networking.

c. Do not use a custom Terraform provider.

d. Name the stack IAD-OP-LAB07-1-Stack-1.

e. Add a short description.

f. Ensure that your compartment is selected for Create in compartment.

g. Click Next to Configure variables. There will be no variables to configure.

h. Click Next to Review.

i. Confirm that the only service listed for Terraform provider services is core.

j. Click Create.

3. Wait for the stack to finish creating. It will query each of the selected services for the resources in your compartment.

Download Terraform Configuration

1. Under the Stack information tab, click the download link for the Terraform configuration.

2. There will be three files in the downloaded folder: vars.tf, provider.tf and core.tf. Open core.tf.

3. Scroll through the code and identify different resources, such as the VCN and compute

instance. The VCN’s Terraform should look similar to this:

resource oci\_core\_vcn export\_IAD-OPS-LAB07-1-VCN-1 {

#cidr\_block = <<Optional value not found in discovery>>

cidr\_blocks = [

"10.0.0.0/16",

]

compartment\_id = var.compartment\_ocid

defined\_tags = {

Owner.CreatedOn = <Time created>

Owner.Creator = <Your username>

Owner.PrincipalType = user

}

display\_name = IAD-OPS-LAB07-1-VCN-1

dns\_label = iadopslab071vcn

freeform\_tags = {

VCN = #<auto-generated tag>

}

ipv6private\_cidr\_blocks = [

]

#is\_ipv6enabled = <<Optional value not found in discovery>>

}

Edit the Terraform Configuration to Be Reusable

Before you can reupload the Terraform generated by resource discovery, there are several fields that need to be removed. This is because resource discovery documents properties such as certain public IPs, private IPs, and instance launch options that are not manually configurable.

1. Remove the exported private IP for the compute instance. For this configuration to be reusable, it needs to allow OCI to allocate a new private IP to the instance instead of referring to the existing VNIC.

a. Find the resource block of type oci\_core\_private\_ip. It should look similar to this:

resource oci\_core\_private\_ip export\_IAD-OP-LAB07-1-VM-01 {

defined\_tags = {

\Oracle-Tags.CreatedBy\ = <Your username>

\Oracle-Tags.CreatedOn\ = <Time created>

}

display\_name = \IAD-OP-LAB07-1-VM-01\

freeform\_tags = {

}

hostname\_label = \iad-op-lab07-1-vm-01\

ip\_address = <The IP that OCI assigned>

#vlan\_id = <<Optional value not found in discovery>>

vnic\_id = <The VNIC OCID>

}

b. Comment or delete this entire block.

2. Remove the IP address, NVME count, and launch options from the compute instance.

a. Find the compute instance block (it will be of type oci\_core\_instance).

b. Inside of the compute instance block, find the field for private\_ip. It should look similar to this:

resource oci\_core\_instance export\_IAD-OP-LAB07-1-VM-01\_1 {

<Various fields…>

**private\_ip = <The IP that OCI assigned>**

<Various fields…>

}

c. Comment or delete the specific line for private\_ip, (bolded above).

d. Inside of the compute instance block, find the fields for launch\_options. look similar to this:

resource oci\_core\_instance export\_IAD-OP-LAB07-1-VM-01\_1 {

<Various fields…>

**launch\_options {**

**boot\_volume\_type = \PARAVIRTUALIZED\**

**firmware = \UEFI\_64\**

**is\_consistent\_volume\_naming\_enabled = \true\**

**is\_pv\_encryption\_in\_transit\_enabled = \true\**

**network\_type = \PARAVIRTUALIZED\**

**remote\_data\_volume\_type = \PARAVIRTUALIZED\**

**}**

<Various fields…>

}

e. Comment or delete this entire block of configurations (bolded above).

f. Inside of the compute instance block, find the fields for shape\_config look similar to this:

resource oci\_core\_instance export\_IAD-OP-LAB07-1-VM-01\_1 {

<Various fields…>

shape\_config {

**baseline\_ocpu\_utilization = \\**

memory\_in\_gbs = \6\

**nvmes = \0\**

ocpus = \1\

}

<Various fields…>

}

g. Comment or delete the lines for nvmes and baseline\_ocpu\_utilization (bolded above).

3. Remove the public IP address reference from the NAT gateway.

a. Find the block for the NAT gateway. It will look similar to this:

resource oci\_core\_nat\_gateway export\_NAT-Gateway-IAD-OP-LAB07-1-VCN-1 {

block\_traffic = \false\

compartment\_id = var.compartment\_ocid

defined\_tags = {

\Oracle-Tags.CreatedBy\ = <Your username>

\Oracle-Tags.CreatedOn\ = <Time created>

}

display\_name = \NAT Gateway-IAD-OP-LAB07-1-VCN-1\

freeform\_tags = {

\VCN\ = <Time created>

}

**public\_ip\_id = <The public IP that OCI assigned>**

#route\_table\_id = <<Optional value not found in discovery>>

vcn\_id = oci\_core\_vcn.export\_IAD-OP-LAB07-1-VCN-1.id

}

b. Comment or delete the line for the public\_ip\_id (bolded above).

Create a New Stack from the Terraform

1. Developer Services - Stacks (under Resource Manager). Click Create stack.

a. The first page of the form will be for stack information.

1) For the origin of the Terraform configuration, keep My configuration selected.

2) Under Stack configuration, select Folder as the source, and upload the folder containing vars.tf, provider.tf, and core.tf (move them to their own folder if needed).

3) Under Custom providers, keep Use custom Terraform providers deselected.

4) Name the stack IAD-OP-LAB07-1-Stack-2

5) Give it a short description.

6) Ensure that your compartment is selected

7) Click Next.

b. The second page will be for variables.

1) The variables should all be automatically populated.

2) Click Next.

c. The third page will be for review.

1) Select Run apply.

2) Click Create. This will take you to the stack’s details page.

d. Wait for the apply job to finish.

3. View the new VCN and new compute instance in the Console. Its will have the same names as the previous ones, but this could have been changed in the Terraform to a variable.

**Resource and Configuration Management: Use Ansible to Deploy Apache Application to Multiple Instances**

**Lab 8-1 Practices**

**A diagram of a cloud shell

Description automatically generated**

Overview

Oracle Cloud Infrastructure automation helps configure resources in a faster and more efficient way. In this lab, you will write an automation script to install and configure an httpd webserver. You will then write an Ansible playbook to spin up Apache applications and deploy them to two compute instances.

In this lab, you’ll:

a. Launch Code Editor and configure Ansible.

b. Write an Ansible playbook to install and configure Apache hosts.

Create VCN and 2 Compute Instances

Create VCN

First, you will create a VCN to house the compute instances.

Start VCN Wizard.

a. Name the VCN IAD-OP-LAB08-1-VCN-01.

b. Leave the CIDR blocks as their defaults.

c. Click Next d. Click Create.

e. Click View Virtual Cloud Network to go to its details page.

1. In the Security Lists - Click the Default Security List to go to its details page.

2. Here you need to open port 80 for HTTP. Click Add Ingress Rule to open a form.

a. Leave the box for Stateless deselected.

b. Leave Source Type as CIDR.

c. For Source CIDR, enter 0.0.0.0/0

d. For IP Protocol, select TCP.

e. Leave Source Port Range empty (this indicates All).

f. For Destination Port Range, enter 80

g. Click Add Ingress Rules at the bottom.

Launch Cloud Shell and Configure Ansible Clients

In this practice, you will set up Ansible clients, that you will install and configure webserver.

Open Code Editor

Create a new folder in the home directory, and name it

ansible\_apache\_lab.

Configure Ansible Hosts

1. Create a file in that folder and name it hosts.yaml.

2. In this file, you will record the public IP addresses of the VMs for Ansible. Add the following code, replacing the two IP addresses in the code with the ones from the VMs you provisioned earlier.

all:

hosts:

<First-IP-address>:

<Second-IP-address>:

Note: Make sure that you follow the indentation properly, because YAML files are sensitive to code indentation.

1. Save the changes by clicking File, then Save.

**Write the Ansible Playbook**

Note: The code below is added piece by piece to explain the process, but the entire code can be added at once. You can copy it from this page into a text editor to ensure that invisible characters aren’t present and line breaks are Linux-style.

1. Create another file in ansible\_apache\_lab by clicking File, then New File, and name it playbook.yaml.

2. Enter the following code to install and start Apache on the VMs.

- name: Install and Configure web servers

hosts: all

remote\_user: opc

tasks:

- name: Ensure apache is at the latest version

ansible.builtin.yum:

name: httpd

state: latest

become: yes

- name: Ensure apache is running

ansible.builtin.service:

name: httpd

state: restarted

become: yes

3. Open a terminal panel in Code Editor In the terminal, execute the playbook:

$ ansible-playbook -i ~/ansible\_apache\_lab/hosts.yaml ~/ansible\_apache\_lab/playbook.yaml

4. It will prompt you to confirm that you want to connect. Type yes and press Enter twice.

(Once for each machine). The two prompts may be intertwined.

The output should look similar to this (IP addresses have been censored):

A black screen with white text

Description automatically generated

5. Now, both VMs should have Apache installed, but you still need to open the Linux firewall to HTTP traffic. Edit the YAML script to configure the firewall to allow incoming HTTP traffic. Add the following block of code to playbook.yaml. Note that it’s indented to be one of the tasks:

- name: Permit traffic in default zone for http service

ansible.posix.firewalld:

service: http

permanent: yes

state: enabled

immediate: true

become: yes

6. Save and execute the playbook again in the terminal:

$ ansible-playbook -i ~/ansible\_apache\_lab/hosts.yaml ~/ansible\_apache\_lab/playbook.yaml

The output should look similar to this (IP addresses have been censored):

A black screen with white text

Description automatically generated

7. To test whether the webserver is running, enter the public IP addresses of the two Ansible

clients into your browser.

You should get a webpage that looks like this: