



## Practice: Configure Oracle Functions

---

Try this hands-on lab with the **Oracle Cloud 30-day Free Trial account** or your own tenancy. If you do not have a free account, click [here](#) to get one.

### Overview

You will perform the following tasks in this practice:

- Launch Cloud Shell
- Generate an ssh Key
- Create a Virtual Cloud Network
- Create and Connect to a Compute Instance
- Configure OCI CLI
- Install Docker
- Create a Function Application
- Configure and Invoke a Function

## Practice: Launching Cloud Shell

---

### Overview

In this practice, you will launch a Cloud Shell session for the OCI user account assigned to you in preparation for the upcoming practices.

### Cloud Shell

Oracle Cloud Infrastructure (OCI) Cloud Shell is a web browser–based terminal accessible from the Oracle Cloud Console. Cloud Shell is free to use (within monthly tenancy limits), and provides access to a Linux shell with a pre-authenticated OCI CLI and other useful tools.

It provides:

- An ephemeral machine to use as a host for a Linux shell, preconfigured with the latest version of the OCI CLI and several useful tools
- 5 GB of storage for your home directory
- A persistent frame of the Console which stays active as you navigate to different pages of the Console

### Note:

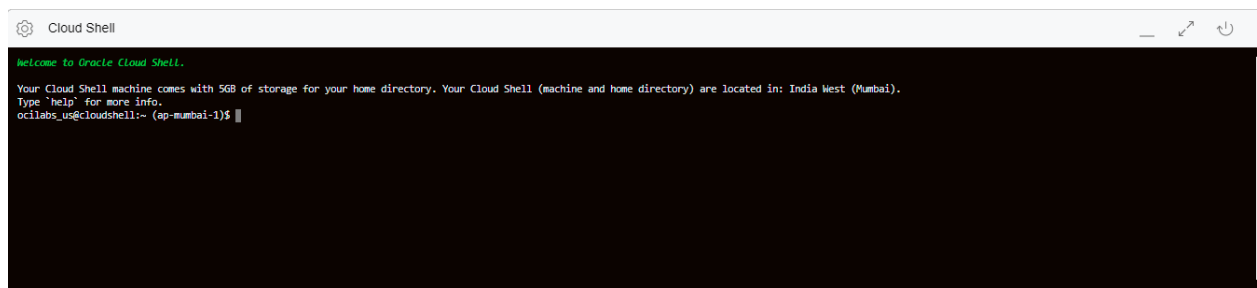
- The OCI CLI will execute commands against the region selected in the Console's Region selection menu when the Cloud Shell is started. Changing the region selection in the Console will not change the region for existing Cloud Shell instances; you will need to open a new Cloud Shell instance to change regions.
- Cloud Shell sessions have a maximum length of 24 hours, and time out after 20 minutes of inactivity. However, this should not impact this practice.

### Tasks

1. Log in to your [Oracle Cloud Account](#).
2. Click the **Cloud Shell** icon in the OCI Console header, highlighted in the following screenshot:



3. This will launch the Cloud Shell in a “drawer” at the bottom of the Console. When it is ready, you will see the terminal.



4. You can use the icons in the upper-right corner of the Cloud Shell window to minimize, maximize, and close your Cloud Shell session.

You can also use the menu icon in the upper-left corner of the Cloud Shell window to upload or download files, restart the Console, and use different setting options.

5. For clipboard operations:
  - Windows users can use `Ctrl-C` or `Ctrl-Insert` to copy, and `Ctrl-V` or `Shift-Insert` to paste.
  - For Mac OS users, use `Cmd-C` to copy and `Cmd-V` to paste.
6. To get started with Cloud Shell, you can run the following OCI CLI command. Your Cloud Shell comes with the OCI CLI pre-authenticated, so there is no setup to perform before you can start using it.

This command will display the namespace of your OCI Tenant.

```
$ oci os ns get
{
  "data": "bm6rwnfgnfbj"
}
```

This completes the task of launching Cloud Shell. Keep this session active for the next practice.

## Practice: Generating an ssh Key

---

### Overview

Instances use an ssh key pair instead of a password to authenticate a remote user. A key pair file contains a private key and a public key. You keep the private key on your computer and provide the public key every time you launch an instance. In this practice, you will generate an ssh key to be used later while launching an instance.

### Tasks

1. Launch the Cloud Shell session as described in the previous practice.
2. Execute the following commands to generate an ssh-key, which will be used to create a Compute instance. As long as an `id_rsa` and `id_rsa.pub` keypair is present, they can be reused. By default, these are stored in `~/ .ssh/` directly.

```
$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key
(/home/ocilabs_us/.ssh/id_rsa):
Created directory '/home/ocilabs_us/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in
/home/ocilabs_us/.ssh/id_rsa.
Your public key has been saved in
/home/ocilabs_us/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:zxQogNbEvQJIAbJb+3x1r8QEJN8ZWlUYHHyZge5lZ10
ocilabs_us@804708cc8ef6
The key's randomart image is:
+---[RSA 2048]---+
|=.+=o.. . o++*o+ |
|o.+ o..+ = o= + E|
|.... ..= +. . o|
| o .. .. . . o +|
|. . . S +. o o |
| o . B .. |
| o . = . |
| . . . |
| . . |
+-----[SHA256]-----+
```

3. Make sure permissions are restricted because sometimes ssh fails if private keys have permissive file permissions.

```
$ chmod 0700 ~/.ssh
$ chmod 0600 ~/.ssh/id_rsa
$ chmod 0644 ~/.ssh/id_rsa.pub
$ ls -l ~/.ssh
total 8
-rw-----. 1 ocilabs_us oci 1675 Jun 29 17:10 id_rsa
-rw-r--r--. 1 ocilabs_us oci 405 Jun 29 17:10 id_rsa.pub
```

4. Copy the contents of ~/.ssh/id\_rsa.pub to Notepad. This is your ssh key to connect to the instances.

```
$ cat ~/.ssh/id_rsa.pub
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQDEUsgq5R/5PcdS1+Mws2Y6vli0HcCw9g3l
uI0x/yFDwE+stlnfyzv4c73+uS35VD6kgFMO5izZKx3fV0JpqhUPjwuyigP9jc
6cgJmWjYhkbCHD8r8bFvrdVv0KuUPi+oKQUI4Zr4EtuTao3kkLywWz6aEJgS6GY2
19JSXqBH27QjgGk4l4sdeb9VuTuQ07Z7VzyAUfKK5oqlJfLC6a/JhdfTLYnv++W
y3lnVZUojEQK57bOD7jVDTTErs0PSWXzMedretrEXtsBU+Tm1DZBe7QWoqghMTkI
a3hegulqIwVxujfy7xDNPE1FHR/LG0978CyJwAfrShjXAYQtSwMF
ocilabs_us@804708cc8ef6
```


This completes the task of creating the ssh key.

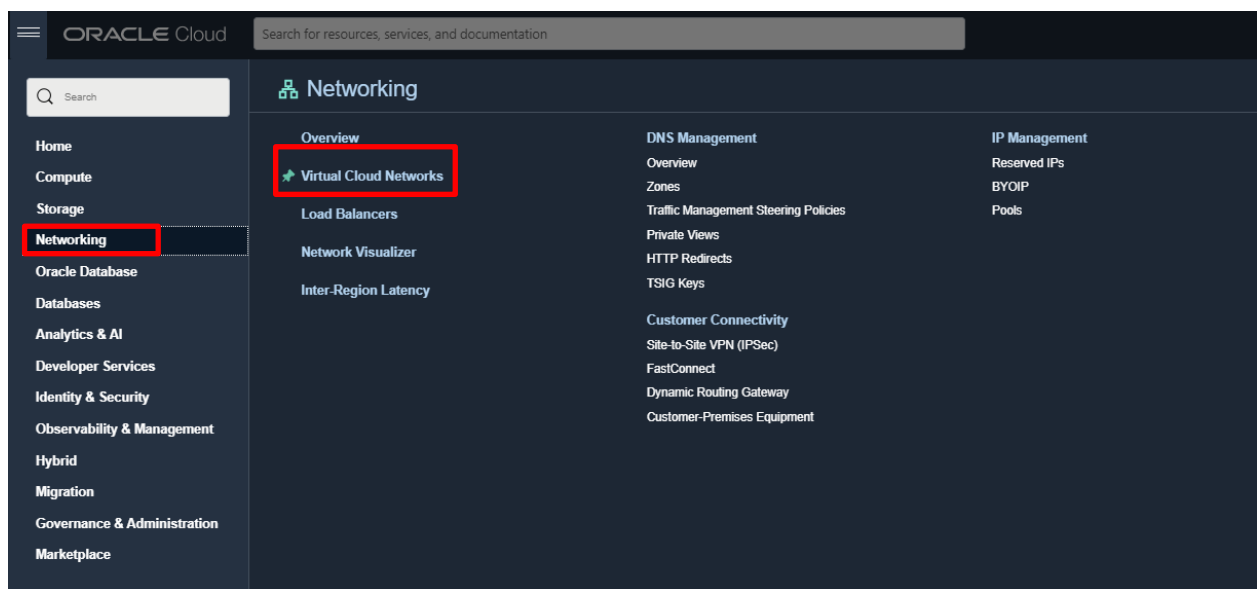
# Practice: Creating a Virtual Cloud Network

## Overview

In this practice, you will create a VCN and the required network resources.

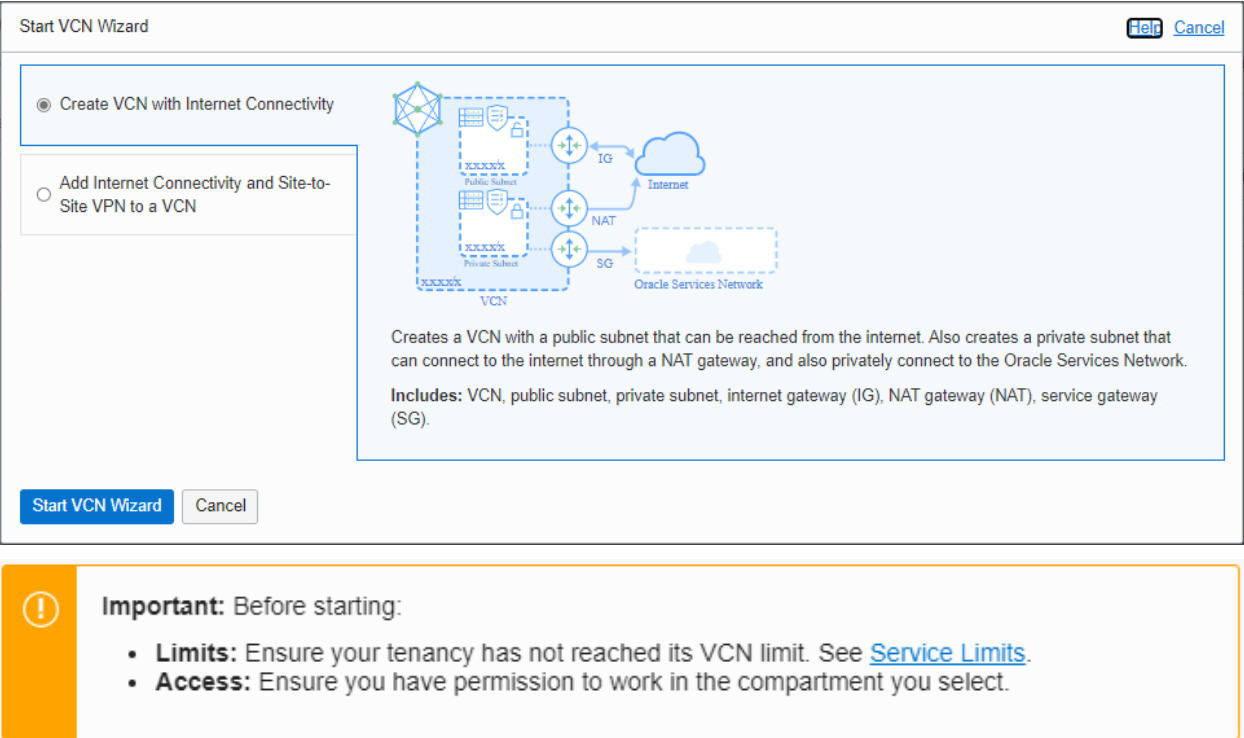
## Tasks

1. Log in to your [Oracle Cloud Account](#).
2. In the **Oracle Cloud Infrastructure** section, enter the cloud account **User Name** and **Password** assigned to you, and click **Direct Sign-In**.
3. At this point, you should be logged in to the **Oracle Cloud Infrastructure (OCI) Dashboard**, also called the OCI Console.
4. Click **Menu**  in the upper-left corner and explore the options available. You will use this navigation path throughout the practice.
5. In the Console, click **Menu > Networking > Virtual Cloud Networks**.



6. On the networking page, select the compartment name from the drop-down list.
7. On the Virtual Cloud Networks page, click **Start VCN Wizard**.

8. In the dialog box, select **Create VCN with Internet Connectivity**, and click **Start VCN Wizard**.



**Start VCN Wizard**

☒ Create VCN with Internet Connectivity

☐ Add Internet Connectivity and Site-to-Site VPN to a VCN

Creates a VCN with a public subnet that can be reached from the internet. Also creates a private subnet that can connect to the internet through a NAT gateway, and also privately connect to the Oracle Services Network.

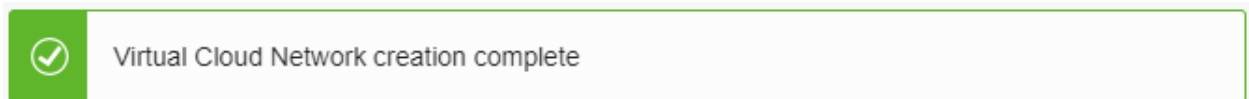
Includes: VCN, public subnet, private subnet, internet gateway (IG), NAT gateway (NAT), service gateway (SG).

**Start VCN Wizard** Cancel

**Important:** Before starting:

- **Limits:** Ensure your tenancy has not reached its VCN limit. See [Service Limits](#).
- **Access:** Ensure you have permission to work in the compartment you select.

9. Provide the basic information:
  - **VCN Name:** FunctionsVCN
  - **Compartment:** ocilabs
  - **VCN CIDR Block:** Enter 10.0.0.0/16
  - **Public Subnet CIDR Block:** Enter 10.0.1.0/24
  - **Private Subnet CIDR Block:** Enter 10.0.2.0/24
  - **Select the check box for** Use DNS Hostnames in this VCN.Click **Next**.
10. Review and click **Create**. A lot of useful information is available on this page for you to review.
11. The VCN is created along with Private and Public Subnets.



**Note:** This option is the quickest way to get a working cloud network in the fewest steps.

This completes the task of creating a VCN along with two subnets. You will use them in the upcoming practices.



# Practice: Creating and Connecting to a Compute Instance

---

## Overview

In this practice, you create Compute VM instances in each of the two subnets in your VCN.

An Oracle Cloud Infrastructure VM Compute instance runs on the same hardware as a Bare Metal instance, leveraging the same cloud-optimized hardware, firmware, software stack, and networking infrastructure.

## Tasks

1. In the OCI Console, navigate to **Menu > Compute > Instances**.
2. Click **Create Instance**.

**Note:** You should select your compartment before creating an instance.

3. Enter the following details for your Compute instance:
  - **Name:** `Functions_instance`
  - **Compartment:** `ocilabs`
  - **Placement:** `Select default`
  - **Image or Operating System:** `Select the default Oracle Linux image`
  - **Availability Domain:** `Select any Availability Domain`
  - **Shape:** `VM.Standard2.1`

**ORACLE** Linux Oracle Linux 7.9  
Image build: 2021.08.27-0 [Change image](#)

Shape

**intel** VM.Standard2.1  
Virtual machine, 1 core OCPU, 15 GB memory, 1 Gbps network bandwidth [Change shape](#)

**Note:**



*To change the Image, you can click **Change image**.*

*Click **Show Shape, Network, and Storage Options** and explore the details.*

- **Networking:** **FunctionsVCN**
  - **Add ssh Keys:** Select the **Paste ssh keys** option and paste the content of your Public ssh key copied in the previous practice. (Also available in `~/.ssh/id_rsa.pub`)
  - **Boot Volume:** Select default
4. Finally, click **Create** to create the Compute instance.
  5. When the instance state changes to **Running**, you can ssh to the Public IP address of the instance. To do this, make a note of the Public IP address that gets assigned to the **Functions\_instance**.
  6. You will use **Cloud Shell** to connect to the Compute instance. Bring up the minimized Cloud Shell terminal, or launch it again and enter the following command, and enter **Yes** when prompted to continue connecting.

```
$ ssh opc@<Public_IP_of_Compute>
```

**Note:** In general, for OCI Linux-based compute instances, the default username is **opc**.

**When successfully connected, you can see the change in the command prompt to ensure you are now logged in to your **Functions\_instance** Compute instance.**

This completes the task of creating a Compute instance. Remain logged in to the instance to continue with the next practice.

## Practice: Configuring OCI CLI

### Tasks

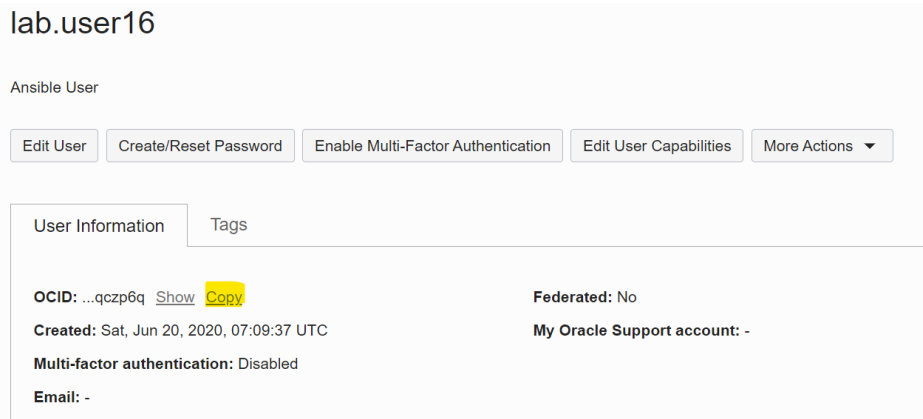
1. Log in to the Functions\_instance.
2. Check that OCI CLI is installed by entering the following command:

```
$ oci -v
```

The output will give you the version of the OCI CLI installed.

3. Next, you will need to gather some information so that you can configure oci. Record your OCID in a text file.

- a. Navigate to **Identity > Users** and copy the OCID for your user login (user\_ocid).



lab.user16

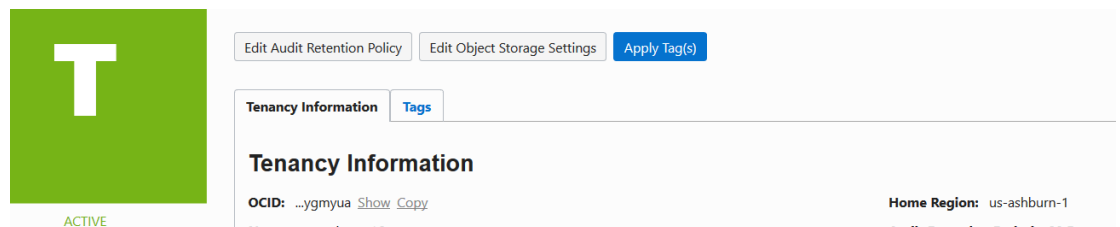
Ansible User

Edit User Create/Reset Password Enable Multi-Factor Authentication Edit User Capabilities More Actions ▼

User Information Tags

OCID: ...qczp6q [Show](#) [Copy](#) Federated: No  
Created: Sat, Jun 20, 2020, 07:09:37 UTC My Oracle Support account: -  
Multi-factor authentication: Disabled  
Email: -

- b. Click **Menu > Governance & Administration > Tenancy Details**. Copy and record the tenancy OCID.



ACTIVE

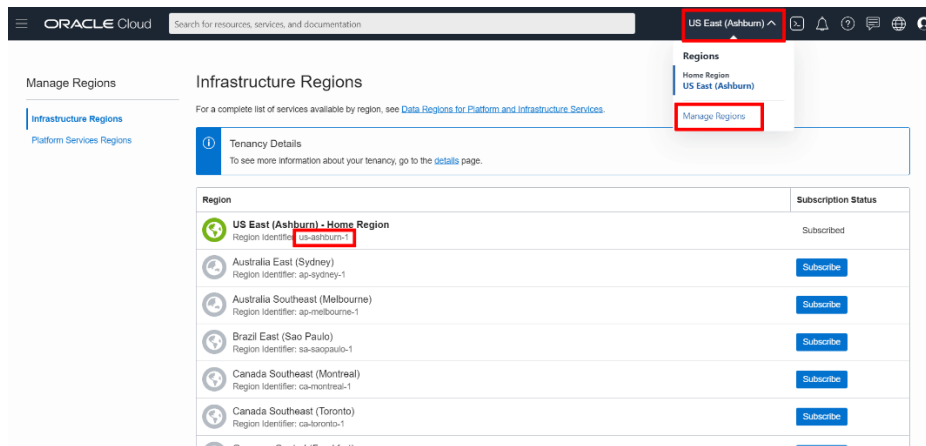
Edit Audit Retention Policy Edit Object Storage Settings Apply Tag(s)

Tenancy Information Tags

Tenancy Information

OCID: ...ygmyua [Show](#) [Copy](#) Home Region: us-ashburn-1  
Name: ocid1tenancy... Audit Retention Period: 90 Days

- c. Next, you will need to get your region identifier. Click your region and then click **Manage Regions**. Then copy your region identifier and record it.



4. To configure OCI CLI, enter the following command:

```
$ oci setup config
```

5. When prompted for a location for your config, press **Enter** to choose the default location. When prompted for your user OCID, tenancy OCID, and region ID, enter the appropriate information. When asked if you want to generate a new RSA key pair, enter **Y**. For all other prompts, press **Enter** to accept the default.

```
Enter a location for your config [/home/opc/.oci/config]:
Enter a user OCID: ocid1.user.oc1..aaaaaaa2ikgy6fwpu3jagmobf25hc4ij5akmw6detiwpv4enjxvczp6q
Enter a tenancy OCID: ocid1.tenancy.oc1..aaaaaaaaf5piu7vbf72x2ortivvc5wdisajtpwoupt66qkcdn5v74lgrcvq
Enter a region by index or name(e.g.
1: ap-chiyoda-1, 2: ap-chuncheon-1, 3: ap-hyderabad-1, 4: ap-melbourne-1, 5: ap-mumbai-1,
6: ap-osaka-1, 7: ap-seoul-1, 8: ap-sydney-1, 9: ap-tokyo-1, 10: ca-montreal-1,
11: ca-toronto-1, 12: eu-amsterdam-1, 13: eu-frankfurt-1, 14: eu-zurich-1, 15: me-dubai-1,
16: me-jeddah-1, 17: sa-santiago-1, 18: sa-saopaulo-1, 19: sa-vinhedo-1, 20: uk-cardiff-1,
21: uk-gov-cardiff-1, 22: uk-gov-london-1, 23: uk-london-1, 24: us-ashburn-1, 25: us-gov-ashburn-1,
26: us-gov-chicago-1, 27: us-gov-phoenix-1, 28: us-langley-1, 29: us-luke-1, 30: us-phoenix-1,
31: us-sanjose-1): us-ashburn-1
Do you want to generate a new API Signing RSA key pair? (If you decline you will be asked to supply the path to an existing key.) [Y/n]: Y
Enter a directory for your keys to be created [/home/opc/.oci]:
Enter a name for your key [oci_api_key]:
Public key written to: /home/opc/.oci/oci_api_key_public.pem
Enter a passphrase for your private key (empty for no passphrase):
Private key written to: /home/opc/.oci/oci_api_key.pem
Fingerprint: bd:2a:c9:19:10:22:a4:45:72:e4:fb:5b:32:b1:cb:2c
Config written to /home/opc/.oci/config

If you haven't already uploaded your API Signing public key through the
console, follow the instructions on the page linked below in the section
'How to upload the public key':

https://docs.cloud.oracle.com/Content/API/Concepts/apisigningkey.htm#How2
```

6. The `oci setup config` command also generates an API key. You will need to upload this API key into your OCI account for authentication of API calls.

```
$ cat ~/.oci/oci_api_key_public.pem
```

- Highlight and copy the content from the Oracle Cloud Shell. Click the human icon followed by your username. Then scroll down and click **API Keys**. On your user details page, click **Add Public Key**. In the dialog box, paste the public key content and click **Add**.

```
[opc@functions-instance ~]$ cat ~/.oci/oci_api_key_public.pem
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA1rrjwnwQmUU0h1L80pwi
S2zCJGgaSXDDH1H4dI/zF09E37GHooVa48dXJBZH7/S6zfDvRVwuxbrijc0RXeU
Drq0aaoJEYpy9s5r0uT1nsoetUMpKDgfUAQ3Hmg10BnseJjyoPMnrNCT/8/mjMw8
fw+pcSs2M1AhnSWrYjQtC5XRYw8te9U/S73/yhSoLeqUwafSb37EKwSYVwSi9fKP
RL9jTWAHtq+ZmOEGmL8Mw+W+iSOWRYJ8dfpg3zDb3BhOTOMmo5+XoJ2POUVX5fO
uojrTigqUXxSkYUePx0LSIPji+VZyifVE8iqIq1180/l/Dlm92eLNUElipEfQ/yw
gQIDAQAB
-----END PUBLIC KEY-----
```

The screenshot shows the Oracle Cloud console interface. At the top, there's a search bar and navigation links. The main content area is titled 'API Keys'. On the left sidebar, under 'Resources', the 'API Keys' link is highlighted with a red box. In the top right corner, the user profile 'lab.user16' is highlighted with a red box. Below the profile, there are links for 'Tenancy: ocoocictrng6', 'Change Password', 'User Settings', and 'Sign Out'. The 'Add API Key' button is highlighted with a red box. Below it, a table shows one API key with the following details:

Fingerprint	Created
ac:28:50:cf:e8:25:bb:ea:40:ae:d4:42:6e:b3:52:b1	Wed, Dec 9, 2020, 08:18:11 UTC

At the bottom right of the table, it says 'Displaying 1 API Key'.

## Add API Key

[Help](#)

**Note:** An API key is an RSA key pair in PEM format used for signing API requests. You can generate the key pair here and download the private key. If you already have a key pair, you can choose to upload or paste your public key file instead. [Learn more](#)

☐ Generate API Key Pair ☐ Choose Public Key File ☒ Paste Public Key

Public Key

```
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA1rrjwnwQmUU0hIL8OpWi
S2zCJCgaSXDDHIH4dI/zFO9E37GHooVa48dXJQBZH7/S6zfDvRVWuxbrijc0RXeU
DrqOaqoJEYpy9s5r0uT1nsoetUMpKDgfUAQ3HMglOBnseJjyoPMnrNCT/8/mjMw8
fW+pcSs2MIAhnSWrYjQtc5XRYw8te9U/S73/yhSoLeqUwafSb37EKwSYvWSi9fKP
RL9jTWAHtq+ZmOEGmL8NW+W+iSOWRYJ8dfpg3zDb3BhOTOMmmo5+XoJ2POUVX5fO
uojrTigqUXxSkYUePx0LsIPji+VZyifVE8iqLq1180//Dlm92eLNUElipEfQ/yW
gQIDAQAB
-----END PUBLIC KEY-----
```

Add

[Cancel](#)

This completes the task of configuring the OCI CLI. Remain logged in for the next practice.

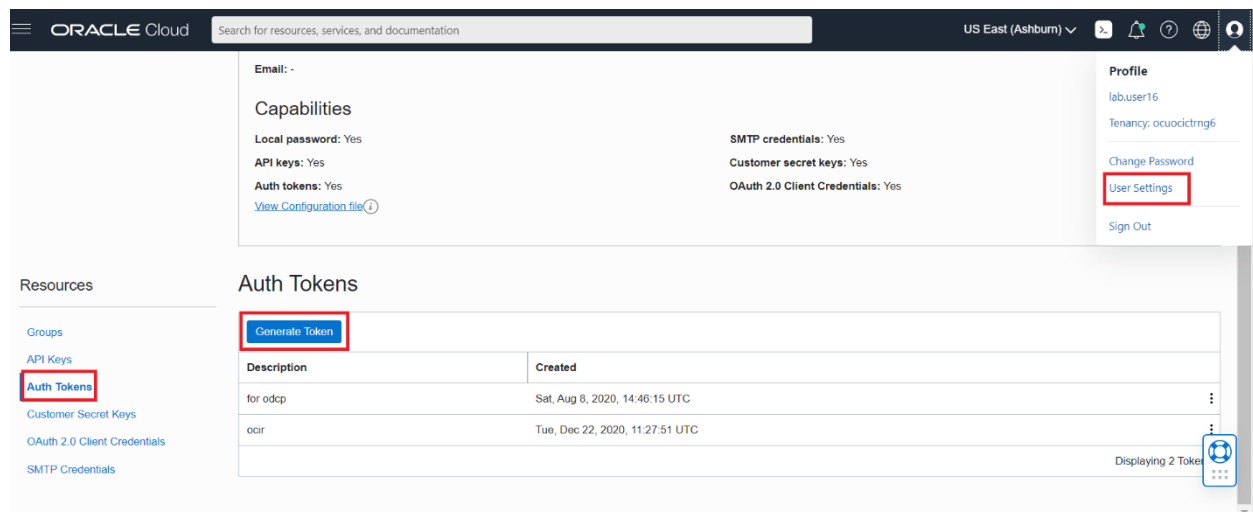
# Practice: Installing Docker

## Tasks

1. Generate an Auth Token. It is an Oracle-generated token that you can use to authenticate with third-party APIs and an Autonomous Database instance.

In the OCI Console, click the user icon (top right) and then click **User Settings**.

Under **Resources**, click **Auth Tokens**, and then click **Generate Token**. In the pop-up window, provide a description and then click **Generate Token**.



2. Click **Copy** and save the token in Notepad.

*Note: Do not close the window without saving the token as it cannot be retrieved later.*

3. Using your Cloud Shell connected to your compute instance, install **yum-utils** with the following command:

```
$ sudo yum install -y yum-utils device-mapper-persistent-data lvm2
```

4. Add the docker repo.

```
$ sudo yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
```

5. Edit the local `docker-ce.repo` file.

```
$ sudo vi /etc/yum.repos.d/docker-ce.repo
```

6. Replace the `baseurl` for `[docker-ce.repo]` with [https://download.docker.com/linux/centos/7/\\$basearch/stable](https://download.docker.com/linux/centos/7/$basearch/stable).

```
[docker-ce-stable]
name=Docker CE Stable - $basearch
baseurl=https://download.docker.com/linux/centos/7/$basearch/stable:
enabled=1
gpgcheck=1
gpgkey=https://download.docker.com/linux/centos/gpg

[docker-ce-stable-debuginfo]
name=Docker CE Stable - Debuginfo $basearch
baseurl=https://download.docker.com/linux/centos/$releasever/debug-$basearch/stable
```

7. Enable the Oracle Linux Development Packages repository and install `docker-engine`.

```
$ sudo yum-config-manager --enable ol7_developer
$ sudo yum install docker-engine -y
```

8. Enable and start `docker`.

```
$ sudo systemctl enable docker
Created symlink from /etc/systemd/system/multi-user.target.wants/docker.service to
/usr/lib/systemd/system/docker.service.

$ sudo systemctl start docker
```

9. Enable the `opc` user to use `docker` with the following command:

```
$ sudo usermod -aG docker opc
$ exit
```

10. `ssh` back to the compute instance and view `docker` images.

```
$ cd ~/.ssh
$ ssh -i ~/.ssh/id_rsa opc@PUBLIC-IP-OF-COMPUTE-1
$ docker images
```

```
[opc@functions-instance ~]$ docker images
REPOSITORY    TAG       IMAGE ID   CREATED   SIZE
```

11. Verify the `docker` version.

```
$ docker version
```



12. Launch the standard hello-world docker image as a container.

```
$ docker run hello-world
```

```
[opc@functions-instance ~]$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
b8dfde127a29: Pull complete
Digest: sha256:7d91b69e04a9029b99f3585aaaccae2baa80bcf318f4a5d2165a9898cd2dc0a1
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (amd64)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

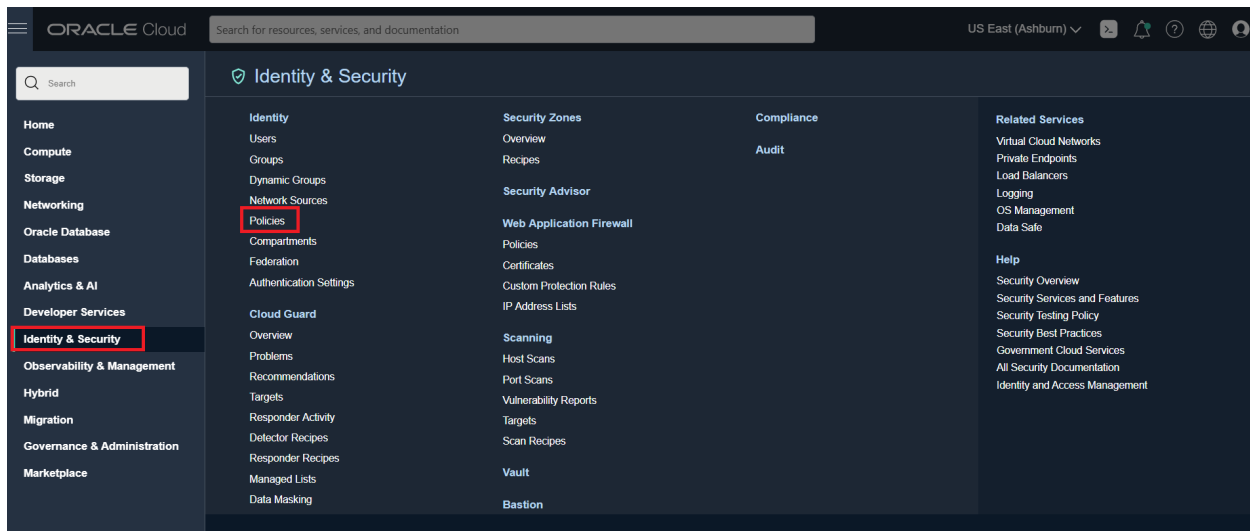
Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/
```

This completes the practice to install docker.

# Practice: Creating a Function Application

## Tasks

1. Click the **Navigation Menu** in the upper-left corner, navigate to **Identity & Security**, and select **Policies**.



2. Make sure that the compartment you selected is **root** and then click **Create Policy**.

Name: **FunctionPolicy**

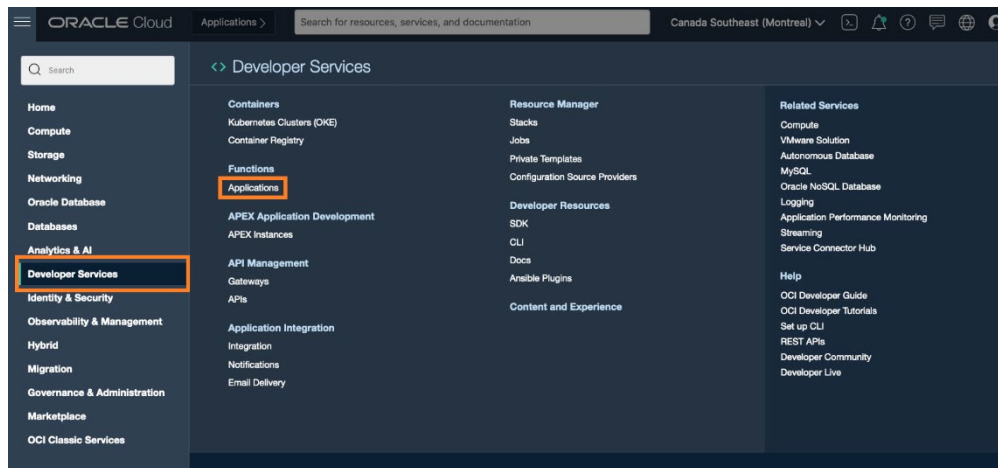
Description: **Allow functions to work**

Scroll down to the Policy Statements section. Enable **Show manual editor** on the Policy Builder tab.

Add these two policy statements. Click **Create**.

```
allow service FAAS to use virtual-network-family in tenancy
allow service FAAS to read repos in tenancy
```

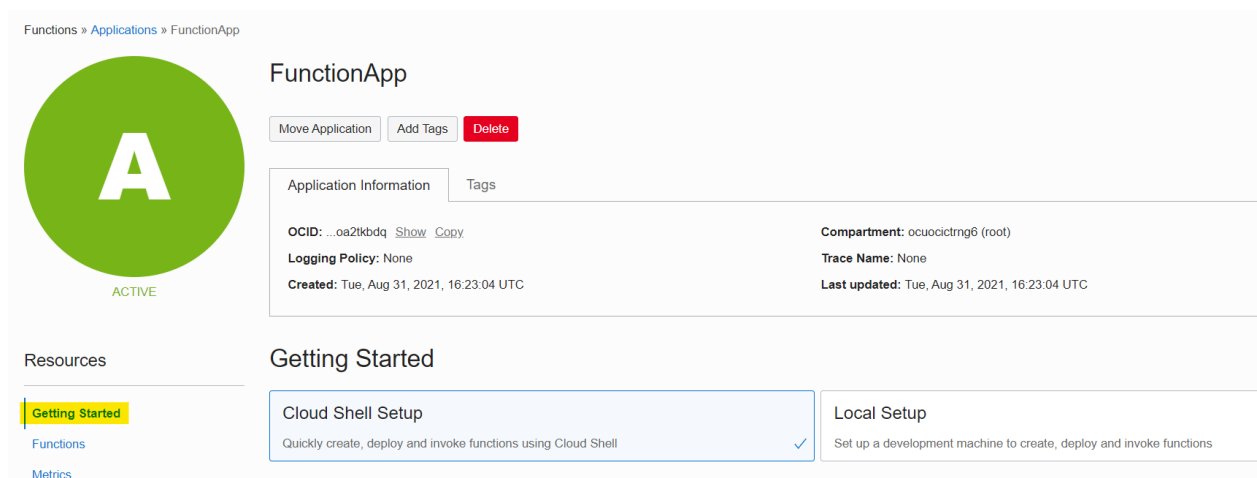
- Click the **Navigation Menu** in the upper-left corner, navigate to **Developer Services**, and select **Applications**.



- Click **Create Application** and enter the details in the dialog box.
  - NAME:** FunctionApp
  - VCN in:** Choose the compartment where your VCN was created
  - SUBNETS:** Choose your VCN's public subnet
  - LOGGING POLICY:** None

Click **Create**.

- Scroll down and click **Getting Started**.



This completes the practice of creating a function application.

## Practice: Configuring and Invoking a Function

---

### Tasks

1. Connect to the Function\_instance using Cloud Shell.

```
$ cd ~/.ssh
$ ssh -i SSH-KEY-NAME opc@PUBLIC-IP-OF-COMPUTE-1
```

2. Install Fn CLI, which is needed to execute function commands.

```
$ curl -Ls
https://raw.githubusercontent.com/fnproject/cli/master/install |
sh
$ fn version
Client version is latest version: 0.6.8
Server version: ?
$
```

3. Create the new Fn Project CLI context. CONTEXT-NAME can be a name that you can choose. For this practice, CONTEXT-NAME will be test-fn.

```
$ fn create context test-fn --provider oracle
Successfully created context: test-fn
$ fn use context test-fn
Now using context: test-fn
```

4. Configure the new context with the api-url endpoint to use when calling the OCI API. Replace REGION-IDENTIFIER with your region identifier.

```
$ fn update context api-url https://functions.REGION-IDENTIFIER.oraclecloud.com
```

```
[opc@functions-instance ~]$ fn update context api-url https://functions.us-ashburn-1.oraclecloud.com
Current context updated api-url with https://functions.us-ashburn-1.oraclecloud.com
```

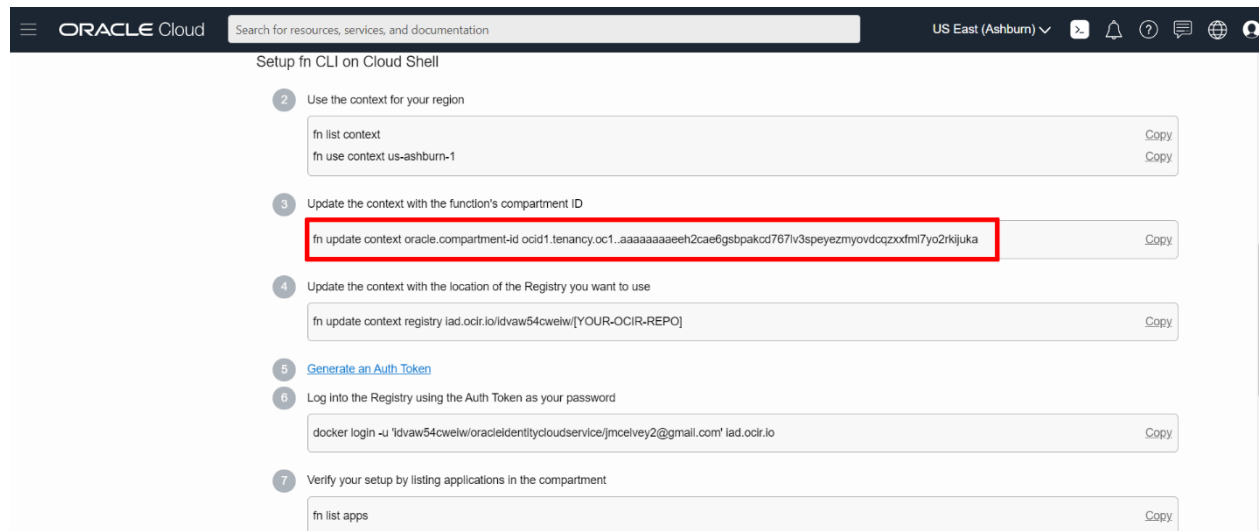
5. Configure the new context with the name of the profile you've created for use with Oracle Functions.

```
$ fn update context oracle.profile DEFAULT
```

6. Scroll down on the **Getting Started** section of your function application. Copy and paste command 3 into the Oracle Cloud Shell.

The command should be in the format:

```
fn update context oracle.compartment-id [COMPARTMENT-OCID]
```



7. Copy and paste command 4 into the Oracle Cloud Shell. Substitute OCIR-REPO with a name of your choice.
8. Copy and paste command 6 into the Oracle Cloud Shell.
9. When prompted for a password, paste your auth token into the shell and press **Enter**.

```
[opc@functions-instance ~]$ fn update context oracle.compartment-id ocid1.tenancy.oc1..aaaaaaah2cae6gsbpakcd767lv3speyzmyovdqzodmi7yo2rkijuka
Current context updated oracle.compartment-id with ocid1.tenancy.oc1..aaaaaaah2cae6gsbpakcd767lv3speyzmyovdqzodmi7yo2rkijuka
[opc@functions-instance ~]$ fn update context registry iad.ocir.io/ocucictrng6/functionrepo
Current context updated registry with iad.ocir.io/ocucictrng6/functionrepo
[opc@functions-instance ~]$ docker login -u 'ocucictrng6/lab.user16' iad.ocir.io
Password:
WARNING! Your password will be stored unencrypted in /home/opc/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
```

10. Create a function.

```
$ fn init --runtime java hello-java
```

A directory called **hello-java** is created, containing:

- A function definition file called `func.yaml`\*
- An `/src` directory containing source files and directories for the hello-java function\*
- A Maven configuration file called `pom.xml` that specifies the dependencies required to compile the function\*

```
[opc@functions-instance ~]$ fn init --runtime java hello-java
Creating function at: ./hello-java
Function boilerplate generated.
func.yaml created.
```

11. Change the directory to the hello-java directory created in the previous step.

```
$ cd hello-java
```

12. Build the function and its dependencies as a docker image called hello-java, push the image to the specified docker registry, and deploy the function to Oracle Functions.

```
$ fn -v deploy --app FunctionApp
```

```
Pushing iad.ocir.io/ocuoictrng6/functionrepo/hello-java:0.0.3 to docker registry...The push refers to repository [iad.ocir.io/ocuoictrng6/functionrepo/hello-java]
ae9ee2a5c9ea: Pushed
5e479f60ec86: Pushed
573b6604e218: Pushed
0d51b2ffbb94: Pushed
dd24b1fd977c: Pushed
10979b1f367e: Pushed
b129204e40d9: Pushed
1bd0aafd0b10: Pushed
814bff734324: Pushed
0.0.3: digest: sha256:77a6512cb7c9aa149e0a9c3cbdb27212f888e06402005dd193fac1ec9b64adec size: 2206
Updating function hello-java using image iad.ocir.io/ocuoictrng6/functionrepo/hello-java:0.0.3...
Successfully created function: hello-java with iad.ocir.io/ocuoictrng6/functionrepo/hello-java:0.0.3
```

13. Invoke the function.

```
$ fn invoke FunctionApp hello-java
```

14. Verify that the "Hello World!" message is displayed.

```
[opc@functions-instance hello-java]$ fn invoke FunctionApp hello-java
Hello, world!
[opc@functions-instance hello-java]$
```

This completes the practice of deploying and invoking a function.

Terminate Compute, the Function application, and the VCN that were created for the practices.