

Practices for Lesson 1: Build, Deploy, and Run in Kubernetes

Practices for Lesson 1

Overview

In these practices, you configure your local OCI CLI to connect to OCI and kubectl to connect to OKE.

You will build a Docker image in your local Docker repository and push that Docker image into OCI Registry (OCIR) using your auth token to login.

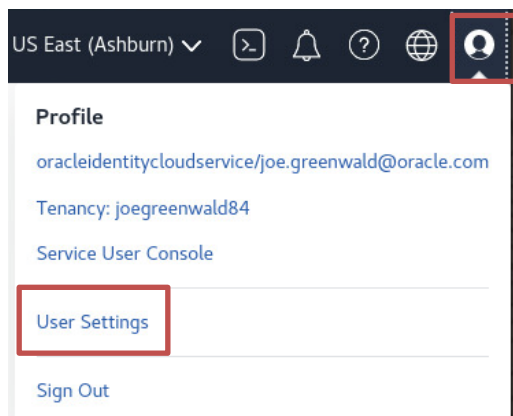
You will create a Kubernetes secret that OKE will use to pull the image and run it in a pod. You will also create a configmap to hold wallet connection details for Oracle ATP database. Lastly, you will create a new deployment and service and test this on Oracle Container Engine for Kubernetes (OKE).

Practice 1-1: Configuring Prerequisites of OCI Command Line Interface

In this practice, you will create the prerequisites for installing and configuring the OCI Command Line Interface (CLI).

Tasks

1. Sign in to **Oracle Cloud Infrastructure**.
 - a. If you are not already connected, connect to your VNC client and your remote Linux compute instance via SSH tunneling using your VNC client.
 - b. Open Firefox (**Applications > Favorites > Firefox**) on your remote Linux compute machine. Use the bookmark you created previously or connect to your console at the provided URL: **`https://cloud.oracle.com/?tenant=<tenancy name>&provider=OracleIdentityCloudService`**.
 - c. You now have access to OCI Console.
2. Create an API Key Pair for OCI.
 - a. Got to the user profile and select **User Settings**.



- b. Select the **API Keys** link and then click the **Add API Key** button.



- c. With **Generate API Key Pair** radio button selected, download and save the **Private Key** and then download and save the **Public Key** and click the **Add** button and then click **Close**.


Note: These files would be downloaded to /home/opc/Downloads folder

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



Download the private key. It will not be shown again. After you download it, [change the file permissions](#) so only you can view it.


 Download Private Key

 [Download Public Key](#)

Add

[Cancel](#)

Opening -08-09-18-26.pem

You have chosen to open:
 -08-09-18-26.pem
which is: plain text document (1.7 KB)
from: blob:

What should Firefox do with this file?

☐ Open with

Text Editor (default) ▼


☒ Save File

☐ Do this automatically for files like this from now on.

Cancel

OK

Opening -08-09-18-26_public.pem

You have chosen to open:
 -08-09-18-26_public.pem
which is: plain text document (458 bytes)
from: blob:

What should Firefox do with this file?

☐ Open with

Text Editor (default) ▼

☒ Save File

☐ Do this automatically for files like this from now on.

Cancel

OK

3. Configure the Oracle Cloud Infrastructure CLI.

- a. Create the OCI config file. In a terminal window, enter the command:

```
oci setup config
```

```
3.0.1
```

```
[opc@pcnjd-808557 ~]$ oci setup config
```

```
This command provides a walkthrough of creating a valid CLI config file.
```

```
The following links explain where to find the information required by this script:
```

```
User API Signing Key, OCID and Tenancy OCID:
```

```
https://docs.cloud.oracle.com/Content/API/Concepts/apisigningkey.htm#0t1
```

```
er
```

```
Region:
```

```
https://docs.cloud.oracle.com/Content/General/Concepts/regions.htm
```

```
General config documentation:
```

```
https://docs.cloud.oracle.com/Content/API/Concepts/sdkconfig.htm
```

```
Enter a location for your config [/home/opc/.oci/config]: █
```

- b. Accept the default location for the config file by pressing Enter. It will prompt you to enter User OCID, Tenancy OCID, and Region name.
- c. For **User OCID**, paste the User OCID you captured earlier and saved to your text file into the terminal in response to the prompt for user OCID.
- d. For the **Tenancy OCID**, paste the Tenancy OCID you captured earlier and saved to your text file into the terminal in response to the prompt for Tenancy OCID.
- e. For the home region, enter the one for your region - most likely **us-ashburn-1**.
- f. The sequence of prompts and responses will look similar to this:

```

Enter a location for your config [/home/opc/.oci/config]:
Enter a user OCID: ocid1.user.oc1..aaaaaaaazsqyx4bqu344gjb2qoyevhff2bvqziet7brpo
2lh65nsol6ekd7q
Enter a tenancy OCID: ocid1.tenancy.oc1..aaaaaaa6rkeountyhbxfrifaf7kcrrgxxfqeker
ttypkro7fwafs5pkq2mq
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: m
e-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]: █

```

- g. It asks whether to generate the RSA key (API Signing key pair):
You are not generating a new key here because you just downloaded one. Enter `n` and give the path as `/home/opc/Downloads/<private key name you downloaded>`.

	Name	Size	Modified
	oracleidentitycloudservice_joe.greenwald-08-25-16-56_public.pem	458 bytes	16:57
	oracleidentitycloudservice_joe.greenwald-08-25-16-56.pem	1.7 kB	16:57

Note: the key files you downloaded are in the `/home/opc/Downloads` folder.

```

Enter the location of your API Signing private key file: /home/opc/Downloads/oracleidentitycloudservice_joe.greenwald-08-25-16-56.
pem
Fingerprint: 5b:73:88:f0:55:f4:21:02:9e:b1:e0:db:2e:33:24:92
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>

```
[opc@pcnjd-808557 ~]$
```

4. The config file is now written to `/home/opc/.oci/config`. Verify the OCI CLI Connectivity.
 - a. Check the OCI CLI connectivity. In a terminal window, enter the command:
`oci os ns get`

Note: It displays your configured object storage namespace used by OCIR. This ensures that CLI has been configured successfully to the OCI, as shown in the following. Ignore the warnings if any.

```
[opc@pcnjd-808557 ~]$ oci os ns get
WARNING: Permissions on /home/opc/Downloads/oracleidentitycloudservice_joe.greenwald-08-25-16-56.pem are too open.
To fix this please try executing the following command:
oci setup repair-file-permissions --file /home/opc/Downloads/oracleidentitycloudservice_joe.greenwald-08-25-16-56.pem
Alternatively to hide this warning, you may set the environment variable, OCI_CLI_SUPPRESS_FILE_PERMISSIONS_WARNING:
export OCI_CLI_SUPPRESS_FILE_PERMISSIONS_WARNING=True

{
  "data": "idkxosvste4z"
}
[opc@pcnjd-808557 ~]$ █
```

- b. (Optional) If you want to stop the warnings, enter the command:

```
chmod 400 /home/opc/Downloads/<private key name>
```

Note: You may leave the terminal on your Linux machine open for the next practice.

This completes this practice.

Practice 1-2: Installing and Configuring `kubeconfig` on Oracle Cloud Infrastructure

Overview

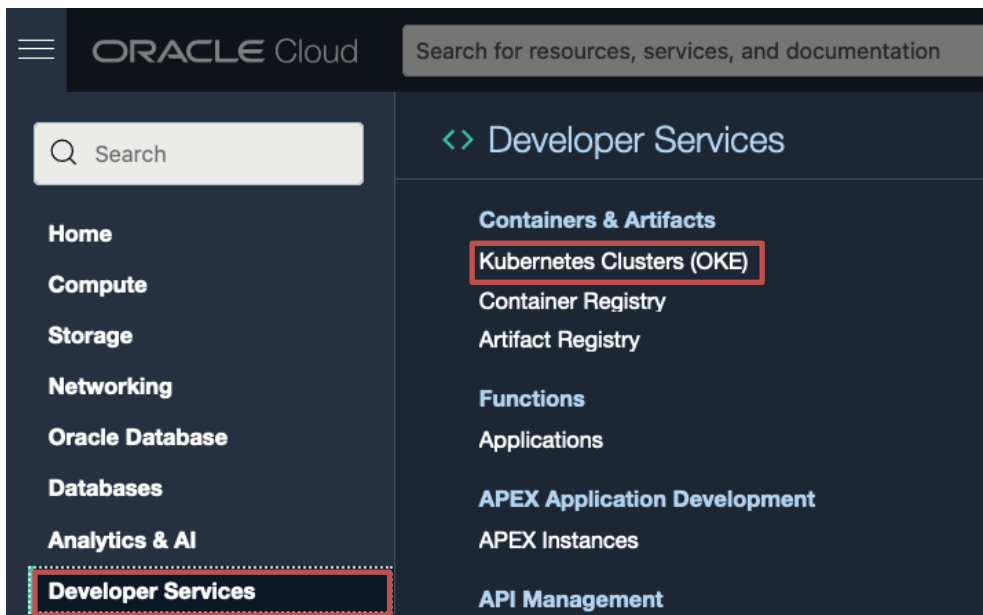
In this practice, you will install and configure `kubeconfig` on Oracle Cloud Infrastructure.

Assumptions

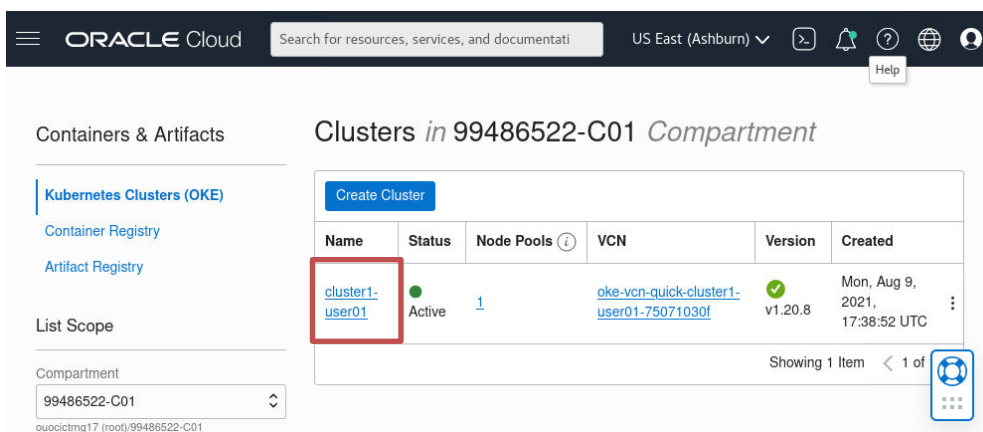
You have successfully completed practice 1-1.

Tasks

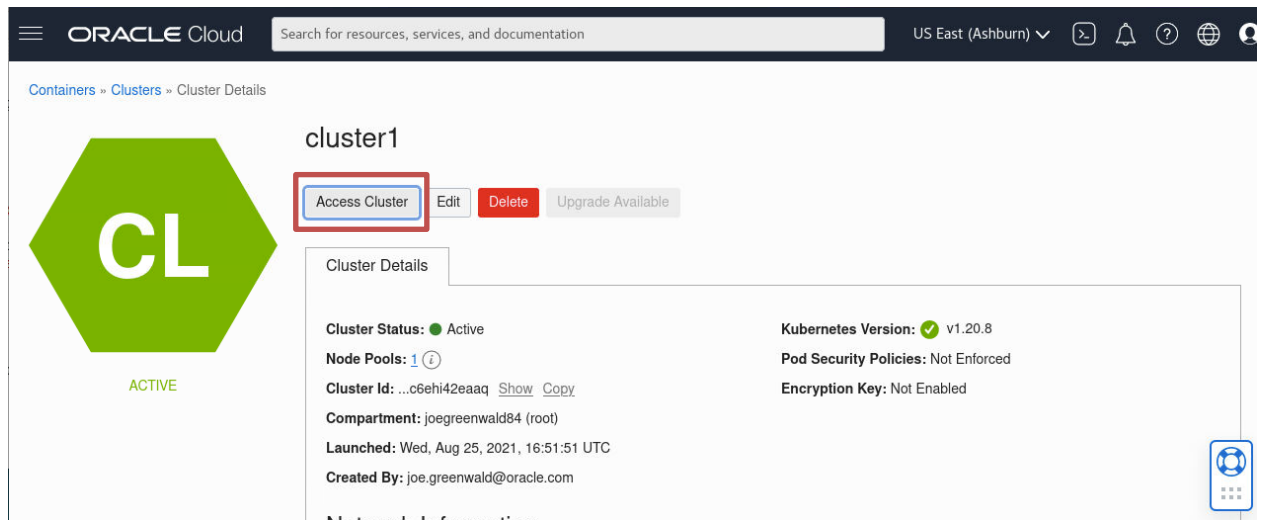
1. Download the `kubeconfig` file.
 - a. Navigate to the OCI console. On the left, you will see the Navigation **menu**; navigate to **Developer Services** and click **Kubernetes Clusters (OKE)**.



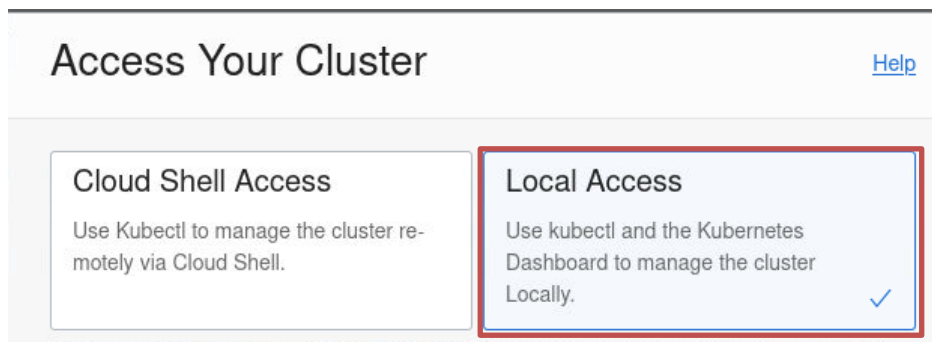
- b. On the **Cluster List** page, select the root compartment and click the name of the cluster in the root compartment.



- c. The **Cluster Details** page shows the details of the cluster. Click the **Access Cluster** button.



- d. This provides the commands to access `kubeconfig` for the respective cluster as shown below. Click **Local Access** to use your provided Linux machine as the host for CLI commands for kubectl.



e. Copy and paste and execute the oci ce cluster commands into your terminal window.

Note: Do not execute the command for a private endpoint.

newer version from [here](#). If you are not sure of the version of the OCI CLI you currently have installed, check with this command:

```
$ oci -v
```

1 Create a directory to contain the kubeconfig file.

```
$ mkdir -p $HOME/.kube
```

2 To access the kubeconfig for your cluster via the VCN-Native public endpoint, copy the following command:

```
$ oci ce cluster create-kubeconfig --cluster-id ocid1.cluster.oc1.iad.aaaaaaaant3xuchoy5lm2alpr5pnp7k6snf55pj3nd3f35txac6ehi42eaaq --file $HOME/.kube/config --region us-ashburn-1 --token-version 2.0.0 --kube-endpoint PUBLIC_ENDPOINT
```

To access the kubeconfig for your cluster via the VCN-Native private endpoint, copy the following command:

```
$ oci ce cluster create-kubeconfig --cluster-id ocid1.cluster.oc1.iad.aaaaaaaant3xuchoy5lm2alpr5pnp7k6snf55pj3nd3f35txac6ehi42eaaq --file $HOME/.kube/config --region us-ashburn-1 --token-version 2.0.0 --kube-endpoint PRIVATE_ENDPOINT
```

3 To set your KUBECONFIG environment variable to the file for this cluster, use:

```
$ export KUBECONFIG=$HOME/.kube/config
```

If you wish to save your new kubeconfig to a different location, please change the --file argument in the CLI command above with the new location path. You may also have to set or update your KUBECONFIG environment variable with this new

Note: The OCID of the cluster differs from cluster to cluster. For convenience, the command in the **How to Access Kubeconfig** dialog box already includes the respective cluster's OCID.

```
[opc@pcnjd-user01 ~]$ oci -v
3.0.0
[opc@pcnjd-user01 ~]$ mkdir -p $HOME/.kube
[opc@pcnjd-user01 ~]$ oci ce cluster create-kubeconfig --cluster-id ocid1.cluster.oc1.iad.aaaaaaaah2mko7a7bs3gqdr4vp3al56fj2uqm5bbj6xkxzljocgslqfbkmfa --file $HOME/.kube/config --region us-ashburn-1 --token-version 2.0.0 --kube-endpoint PUBLIC_ENDPOINT
New config written to the Kubeconfig file /home/opc/.kube/config
[opc@pcnjd-user01 ~]$ export KUBECONFIG=$HOME/.kube/config
[opc@pcnjd-user01 ~]$
```

2. Access the Kubernetes cluster through **kubectl** on Oracle Cloud Infrastructure.
 - a. In the terminal, confirm that **kubectl** is installed by entering the command:
kubectl version
 - b. Verify that you can use **kubectl** to connect to the new cluster that has been created. In the terminal window, enter the following command:
kubectl get nodes

c. You can see the details of the nodes running in the cluster, as shown in the following:

```
[opc@pcnjd-808557 ~]$ oci -v
3.0.1
[opc@pcnjd-808557 ~]$ mkdir -p $HOME/.kube
[opc@pcnjd-808557 ~]$ oci ce cluster create-kubeconfig --cluster-id ocid1.cluster.oc1.iad.aaaaaaaant3xuchoyslm2alpr5pnp7k6snf55pj3
nd3f35txac6ehi42eaaq --file $HOME/.kube/config --region us-ashburn-1 --token-version 2.0.0 --kubernetes-endpoint PUBLIC_ENDPOINT
New config written to the Kubeconfig file /home/opc/.kube/config
[opc@pcnjd-808557 ~]$ export KUBECONFIG=$HOME/.kube/config
[opc@pcnjd-808557 ~]$ kubectl version
Client Version: version.Info{Major:"1", Minor:"22", GitVersion:"v1.22.1", GitCommit:"632ed300f2c34f6d6d15ca4cef3d3c7073412212", Gi
tTreeState:"clean", BuildDate:"2021-08-19T15:45:37Z", GoVersion:"go1.16.7", Compiler:"gc", Platform:"linux/amd64"}
Server Version: version.Info{Major:"1", Minor:"20", GitVersion:"v1.20.8", GitCommit:"50317190d44dbdb51ae7ff430917b32ba96188b5", Gi
tTreeState:"clean", BuildDate:"2021-06-30T14:20:31Z", GoVersion:"go1.15.13 BoringCrypto", Compiler:"gc", Platform:"linux/amd64"}
WARNING: version difference between client (1.22) and server (1.20) exceeds the supported minor version skew of +/-1
[opc@pcnjd-808557 ~]$ kubectl get nodes
NAME                STATUS    ROLES    AGE   VERSION
10.0.10.31          Ready     node     11m   v1.20.8
[opc@pcnjd-808557 ~]$
```

This completes the practice.

Practice 1-3: Pulling an Image from Oracle Cloud Infrastructure Registry when Deploying a Load-Balanced Application to a Cluster

Overview

In this practice, you will create a named secret containing OCI credentials necessary to push an image to OKE and add them to the `dukelabs_oci.yaml` file that defines its Kubernetes deployment and service. Use this yaml file to deploy and verify the duplelabs application to a Kubernetes cluster. You will create a configmap that will contain the needed ATP wallet information for duplelabs to connect to the database using JDBC.

Assumptions

You have successfully completed practice 1-2.

Note: The following tasks need to be performed on the Linux compute instance you created.

Tasks

1. Create a secret for the tutorial.

Note: To enable Kubernetes to pull an image from OCI Registry when deploying an application, you need to create a Kubernetes secret. The secret includes all the login details you would provide while logging in to OCI Registry using the Docker login command, including your auth token.

- a. In a terminal window, run `kubectl` command to create a secret:

Hints: You need to substitute highlighted values within this command. Use the OCIR namespace and full username that you have previously saved into the text editor.

Format:

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

Where:

- `<secret name>`: The name of the secret to be created. For example: `ocirsecret`
- `<region code>`: It is the code for the OCI Registry region you are using. It is in the configuration text file you saved it to in a previous practice.
- `<ocir namespace/full oci username>`: It is the unique reference for the registry that is composed of the name of the container registry namespace followed by the slash symbol and the OCI account name (username) where the image will be pushed.
- `<oci auth token>`: It is the auth-token you created and saved in the text file in a previous practice.
- `<email address>`: Any dummy email address.

```
[opc@pcnjd-808557 ~]$ kubectl create secret docker-registry ocirsecret --docker-server=iad.ocir.io --docker-username="idkxosvste4z/oracleidentitycloudservice/joe.greenwald@oracle.com" --docker-password="KJY{h5Gm].nHx2QjAB>s" --docker-email="dummy@oracle.com"
secret/ocirsecret created
[opc@pcnjd-808557 ~]$ █
```

- b. Verify that the secret has been created. Enter the command:

```
kubectl get secrets
```

Note: Having created the secret, you can now refer to it in the application's yaml file.

2. Create the configmap from the database wallet.

Note: The content of the database wallet must be made available to the dukelabs application image running as a container inside a Kubernetes pod. Since this is read-only data and not very large in size, a configmap is a good choice for holding this information. Configmaps can be created from a single file and its contents, or an entire directory and it will automatically load the entire contents of the directory. That is what you are going to do.

- a. In a terminal window, enter the following command all on one line:

```
kubectl create configmap wallet --from-file=/home/opc/hol_labs/dukelabs_wallet
```

- b. Verify the configmap was created with the contents of the wallet directory:

```
kubectl describe configmap wallet
```

```
[opc@pcnjd-808557 ~]$ kubectl create configmap wallet --from-file=/home/opc/hol_labs/dukelabs_wallet
configmap/wallet created
[opc@pcnjd-808557 ~]$ kubectl describe configmap wallet
Name:         wallet
Namespace:    default
Labels:       <none>
Annotations:  <none>

Data
====
tnsnames.ora:
----
dukelabsdb_high = (description= (retry_count=20)(retry_delay=3)(address=(protocol=tcps)(port=1522)(host=adb.us-sanjose-1.oraclecloud.com))(connect_data=(service_name=g4546947bc4fdbc_dukelabsdb_high.adb.oraclecloud.com))(security=(ssl_server_cert_dn="CN=adb.us-sanjose-1.oraclecloud.com, OU=Oracle ADB SANJOSE, O=Oracle Corporation, L=Redwood City, ST=California, C=US"))))

dukelabsdb_low = (description= (retry_count=20)(retry_delay=3)(address=(protocol=tcps)(port=1522)(host=adb.us-sanjose-1.oraclecloud.com))(connect_data=(service_name=g4546947bc4fdbc_dukelabsdb_low.adb.oraclecloud.com))(security=(ssl_server_cert_dn="CN=adb.us-sanjose-1.oraclecloud.com, OU=Oracle ADB SANJOSE, O=Oracle Corporation, L=Redwood City, ST=California, C=US"))))

dukelabsdb_medium = (description= (retry_count=20)(retry_delay=3)(address=(protocol=tcps)(port=1522)(host=adb.us-sanjose-1.oraclecloud.com))(connect_data=(service_name=g4546947bc4fdbc_dukelabsdb_medium.adb.oraclecloud.com))(security=(ssl_server_cert_dn="CN=adb.us-sanjose-1.oraclecloud.com, OU=Oracle ADB SANJOSE, O=Oracle Corporation, L=Redwood City, ST=California, C=US"))))

dukelabsdb_tp = (description= (retry_count=20)(retry_delay=3)(address=(protocol=tcps)(port=1522)(host=adb.us-sanjose-1.oraclecloud.com))(connect_data=(service_name=g4546947bc4fdbc_dukelabsdb_tp.adb.oraclecloud.com))(security=(ssl_server_cert_dn="CN=adb.us-sanjose-1.oraclecloud.com, OU=Oracle ADB SANJOSE, O=Oracle Corporation, L=Redwood City, ST=California, C=US"))))
```

3. Build a new Docker image and push to OCIR.

- Build a new Docker image.
 - Tag and push the image to the OCIR registry
- a. In a terminal window, enter following commands:

```
cd /home/opc/hol_labs/dukelabs_atp
```

```
docker build -t dukeslabs .
```

```
[opc@pcnjd-808557 ~]$ cd /home/opc/hol_labs/dukelabs_atp/
[opc@pcnjd-808557 dukeslabs_atp]$ docker build -t dukeslabs .
Sending build context to Docker daemon 1.221MB
Step 1/13 : FROM maven:3.6-jdk-11 as build
Trying to pull repository docker.io/library/maven ...
3.6-jdk-11: Pulling from docker.io/library/maven
004f1eed87df: Pull complete
5d6f1e8117db: Pull complete
48c2faf66abe: Pull complete
234b70d0479d: Pull complete
d7eb6c022a4e: Pull complete
6c215442f70b: Pull complete
355e8215390f: Pull complete
cf5eb43522f6: Pull complete
4fee0489a65b: Pull complete
413646e6fa5d: Pull complete
Digest: sha256:1d29ccf46ef2a5e64f7de3d79a63f9bcffb4dc56be0ae3daed5ca5542b38aa2d
Status: Downloaded newer image for maven:3.6-jdk-11
----> e23b595c92ad
Step 2/13 : WORKDIR /helidon
----> Running in 99d7a42bd6a5
Removing intermediate container 99d7a42bd6a5
----> 35dc67e3a65a
Step 3/13 : ADD pom.xml .
----> ee4f4a649266
Step 4/13 : RUN mvn package -Dmaven.test.skip -Declipselink.weave.skip -Dversion.lib.eclipselink=2.7.9
----> Running in e5ab8230edc0
[INFO] Scanning for projects...
Step 13/13 : EXPOSE 8080
----> Running in ca57c6717669
Removing intermediate container ca57c6717669
----> 45e527f80f1f
Successfully built 45e527f80f1f
Successfully tagged dukeslabs:latest
[opc@pcnjd-808557 dukeslabs_atp]$
```

- b. In the terminal window, give a tag to the image that you're going to push to OCI Registry by entering the `docker tag` command all on one line.

```
docker tag dukeslabs <region code>.ocir.io/<ocir namespace>/dukelabs
```

```
[opc@pcnjd dukeslabs_atp]$ docker tag dukeslabs iad.ocir.io/idxosvste4z/dukelabs
[opc@pcnjd dukeslabs_atp]$ █
```

Note: The region code and the OCIR namespace should be in your configuration text file.

- c. Log into OCIR. Enter:

```
docker login <region code>.ocir.io
```

- d. Enter your username with namespace when prompted. Copy this from your configuration text file.
- e. Enter your Auth Token when prompted for **Password**. Copy this from your configuration text file.

```
[opc@pcnjd dukelabs_atp]$ docker login iad.ocir.io
Username: idkxosvste4z/oracleidentitycloudservice/joe.greenwald@oracle.com
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

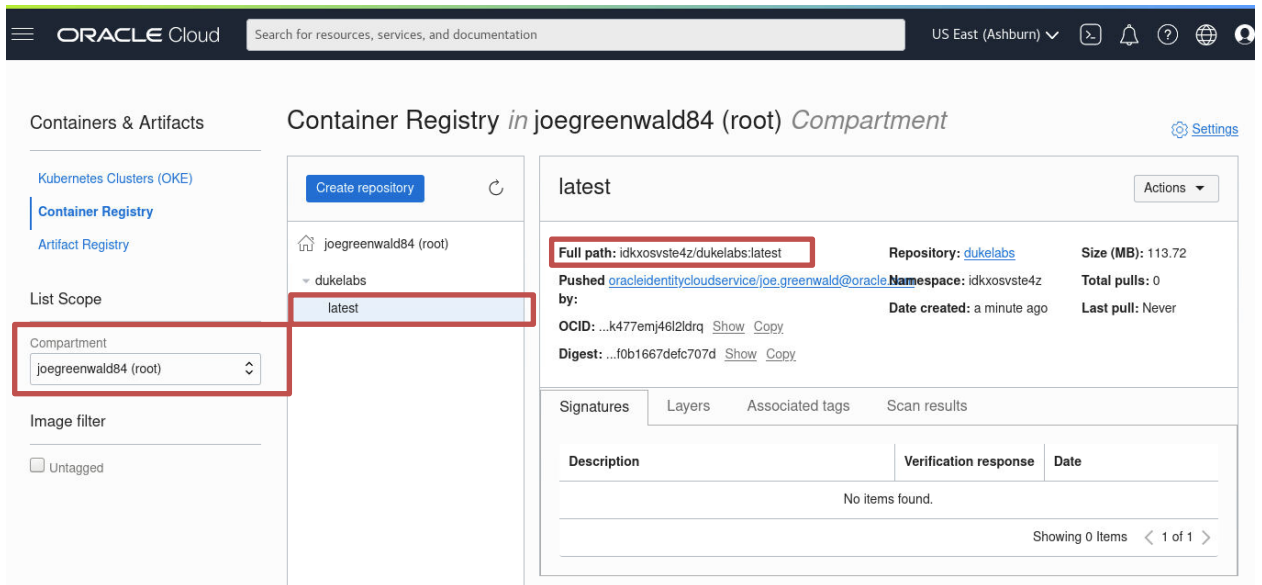
Login Succeeded
[opc@pcnjd dukelabs_atp]$
```

- f. In the terminal window, push the Docker image to OCIR:

```
docker push <region code>.ocir.io/<ocir namespace>/dukelabs
```

```
[opc@pcnjd dukelabs_atp]$ docker push iad.ocir.io/idkxosvste4z/dukelabs
The push refers to repository [iad.ocir.io/idkxosvste4z/dukelabs]
ea8315ca5fe1: Pushed
b9f9bcf448d7: Pushed
a21ed3219d67: Pushed
a292e95a5177: Pushed
a065e8f806e2: Pushed
8d5bc6a4ce7b: Pushed
f68ef921efae: Pushed
latest: digest: sha256:6816af1661da4b71d406526b57b8e07aaa79d9ffe78427843f0b1667defc707d size: 1786
[opc@pcnjd dukelabs_atp]$
```

4. Verify that the image has been pushed to OCI Registry.
- a. Navigate in the OCI console to the **Developer Services > Container Registry** page to view your repository.
- b. Copy the **Full Path** name of the **latest** image.
(In this example it is **idkxosvste4z/dukelabs:latest**)



5. Modify the `dukylabs_oci.yaml` file to prepare it for deployment to Kubernetes.

Notes: You need to make two changes:

- Change the image location to the ocir image repository.
 - Add the secret you created in the earlier step.
 - You should use values that match your ocir image repository location and your secret name.
- a. Open the `/home/opc/hol_labs/dukylabs_atp/dukylabs_oci.yaml` file in a text editor.
 - b. Change the container image and substitute highlighted values:
`<region code>.ocir.io/<ocir namespace>/dukylabs:latest`
 - c. Change the line in the file to include the name of the secret you created earlier.


```
Open [icon] dukelabs_oci.yaml
~/hol_labs/dukelabs_atp

apiVersion: v1
kind: Service
metadata:
  labels:
    app: dukelabs
    name: dukelabs
spec:
  ports:
  - port: 8080
    targetPort: 8080
  selector:
    app: dukelabs
  type: LoadBalancer
---
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: dukelabs
    name: dukelabs
spec:
  replicas: 1
  selector:
    matchLabels:
      app: dukelabs
  template:
    metadata:
      labels:
        app: dukelabs
    spec:
      containers:
      - image: iad.ocir.io/idxosvste4z/dukelabs:latest
        imagePullPolicy: Always
        name: dukelabs
        volumeMounts:
        - mountPath: /home/opc/ol_labs/dukelabs_wallet
          name: wallet-config
      imagePullSecrets:
      - name: ocirsecret
      volumes:
      - name: wallet-config
        configMap:
          name: wallet
```

d. Save your changes.

6. Deploy and test the dukelabs application.
 - a. In a terminal window, deploy the dukelabs application to the cluster by entering the following command:

```
kubectl create -f /home/opc/hol_labs/dukelabs_atp/dukelabs_oci.yaml
```

```
[opc@pcnjd dukelabs_atp]$ kubectl create -f /home/opc/hol_labs/dukelabs_atp/dukelabs_oci.yaml
service/dukelabs created
deployment.apps/dukelabs created
[opc@pcnjd dukelabs_atp]$
```

- b. Check to see if a pod is running with your application:

```
kubectl get pods
```

```
[opc@pcnjd-808557 dukelabs_atp]$ kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
dukelabs-559d679787-bbzvm	1/1	Running	0	84s

Note: You may see “Container Creating” the first couple of times you check on the pods. Eventually you will see the pod is running.

- c. Check the services:

```
kubectl get services
```

Note: You should see that the service is running and has an IP address assigned:

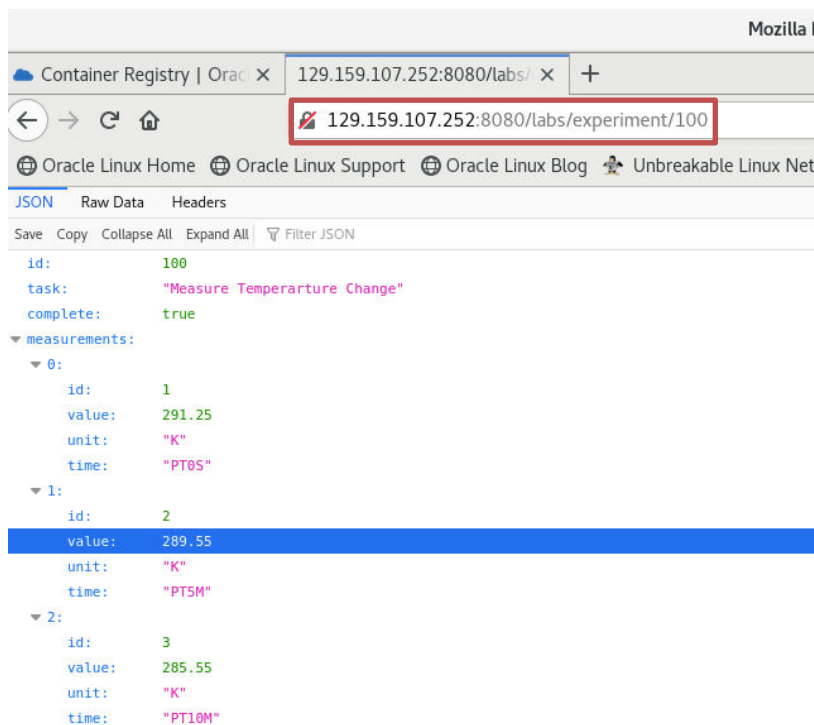
```
[opc@pcnjd-808557 dukelabs_atp]$ kubectl get services
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
dukelabs	LoadBalancer	10.96.39.204	129.159.107.252	8080:30492/TCP	2m15s
kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	41m

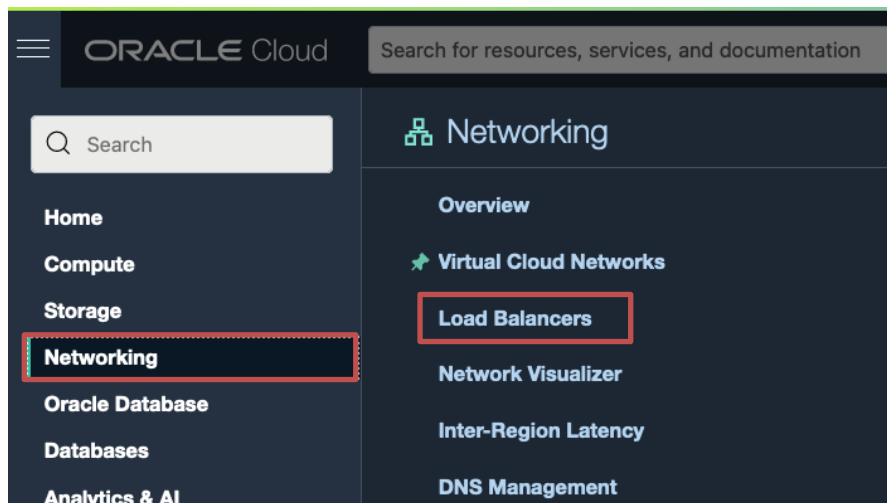
```
[opc@pcnjd-808557 dukelabs_atp]$
```

- d. Enter the EXTERNAL-IP address for your **dukelabs** service into Firefox and test the connection:

Example: 129.159.107.252:8080/labs/experiment/100



7. The dukelabs-service load balancer is implemented as an OCI **Load Balancer** with a back-end set to route incoming traffic to nodes in the cluster.
 - a. You can see the new load balancer on the **Load Balancers** page in the OCI Console. Navigate to **Networking**, and then select **Load Balancers**.



- b. Select your Compartment and you will see that the new load balancer is created in the OCI.

ORACLE Cloud

Search for resources, services, and documentation

US East (Ashburn)

Networking

Overview

Virtual Cloud Networks

Load Balancers

Network Visualizer

Inter-Region Latency

DNS Management

Customer Connectivity

IP Management

List Scope

Compartment

joegreenwald84 (root)

Load Balancers in joegreenwald84 (root) Compartment

Load Balancers provide automated traffic distribution from one entry point to multiple servers reachable from your virtual cloud network (VCN). They improve resource utilization, facilitate scaling, and help ensure high availability.

Create Load Balancer

Name	Type	State	IP Address	Shape	Overall Health	Created
b8902efb-3b59-43b7-aa99-e4928c243264	Load Balancer	Active	193.122.166.215 (Public)	100Mbps	OK	Wed, Aug 25, 2021, 15:54:31 UTC

Showing 1 Item < 1 of 1 >

This completes this practice.