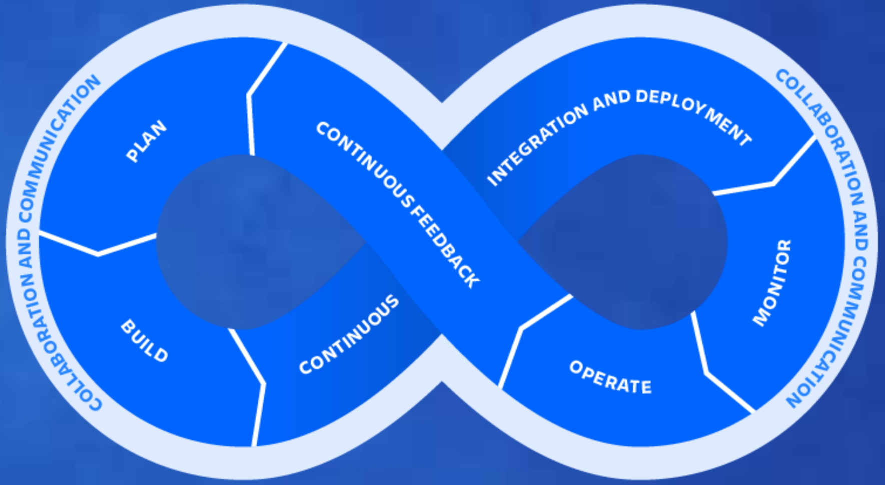

PLAN
CONTINUOUS FEEDBACK
INTEGRATION AND DEPLOYMENT
MONITOR
OPERATE
CONTINUOUS
BUILD
COLLABORATION AND COMMUNICATION
COLLABORATION AND COMMUNICATION

Session 19.
Working with Kubernetes Clusters

Ram N Sangwan

www.rnsangwan.com

Agenda

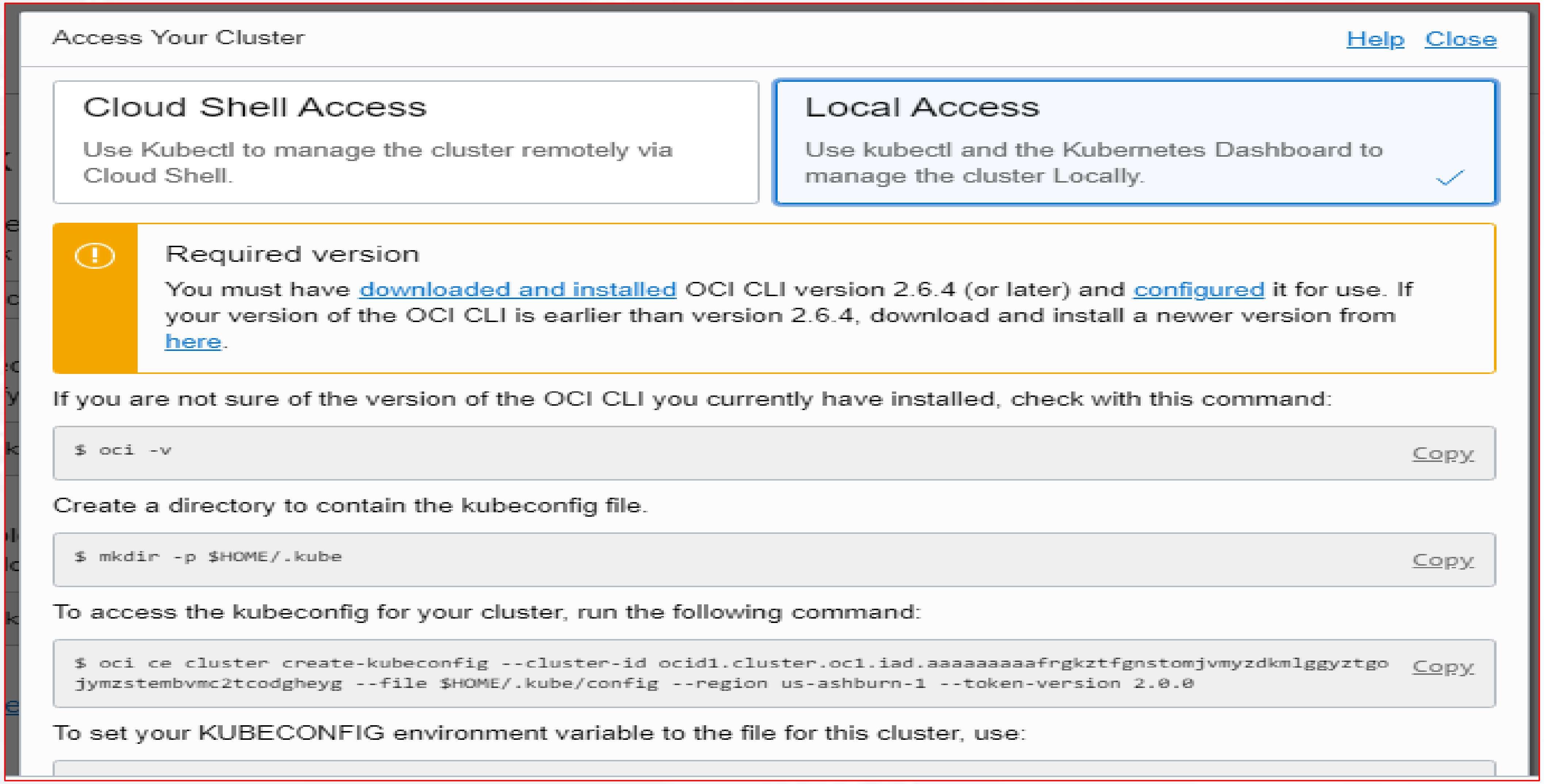
- Accessing a Cluster Using Kubectl
- Starting the Kubernetes Dashboard
- Adding a Service Account Authentication Token to a Kubeconfig File
- Deploying a Sample Nginx App on a Cluster Using Kubectl
- Pulling Images from Registry during Deployment
- Encrypting Kubernetes Secrets at Rest in Etcd
- About Access Control and Container Engine for Kubernetes

Accessing a Cluster Using Kubectl

- To access a cluster using kubectl and the Kubernetes Dashboard, you have to set up a Kubernetes configuration file for the cluster, by default named config, and stored in ***$HOME/.kube***.
- Once kubeconfig file is set, you can start using ***kubectl*** and the Kubernetes Dashboard.

Steps for Setting up kubeconfig File

- Depend on how you want to access the cluster:
  - To access the cluster using ***kubectl*** in Cloud Shell,
    - run an OCI CLI command in the Cloud Shell window to set up the ***kubeconfig*** file.
    - You cannot run the Kubernetes Dashboard in Cloud Shell.
  - To access the cluster using a local installation of ***kubectl*** and the Kubernetes Dashboard:
    - Generate an API signing key pair (if you don't already have one).
    - Upload the public key of the API signing key pair.
    - Install and configure the OCICLI.
    - Set up the ***kubeconfig*** file.


Access Your Cluster

Help Close

Cloud Shell Access

Use Kubectl to manage the cluster remotely via Cloud Shell.

Local Access ✓

Use kubectl and the Kubernetes Dashboard to manage the cluster Locally.

Required version

You must have downloaded and installed OCI CLI version 2.6.4 (or later) and configured it for use. If your version of the OCI CLI is earlier than version 2.6.4, download and install a 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
```

Copy

Create a directory to contain the kubeconfig file.

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

Copy

To access the kubeconfig for your cluster, run the following command:

```
$ oci ce cluster create-kubeconfig --cluster-id ocid1.cluster.oc1.iad.aaaaaaaaafrgkztfgnstomjvmyzdkmlggyztgo
jymzstembvmc2tcodgheyg --file $HOME/.kube/config --region us-ashburn-1 --token-version 2.0.0
```

Copy

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

Step 1. Set up the kubeconfig file

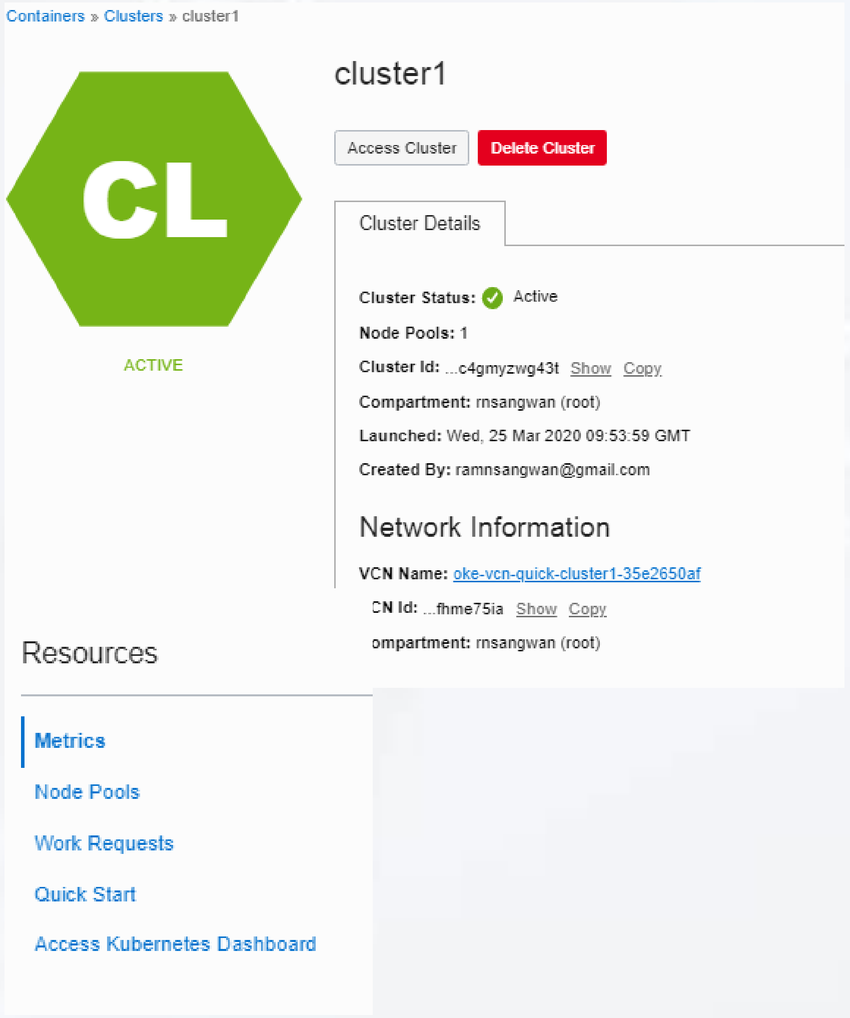
1. Open ***Solutions and Platform -> Developer Services -> Kubernetes Clusters***.
2. On the Cluster List page, click the name of the cluster you want to access using ***kubectl***.
3. Click the Access Cluster button to display the Access Your Cluster dialog box.
4. Click Cloud Shell Access.
5. Click Launch Cloud Shell to display the Cloud Shell window.
6. Run the OCI CLI command to set up the ***kubeconfig*** file. E.g. :

   ```
   $ oci ce cluster create-kubeconfig --cluster-id ocid1.cluster.oc1.phx.. --file $HOME/.kube/config  --region us-phoenix-1 --token-version 2.0.0
   ```

7. If you don't want to use $HOME/.kube/config, set the value of KUBECONFIG environment variable to point to the name and location of the kubeconfig file:

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

Accessing the K8s Cluster with kubectl



Containers » Clusters » cluster1

cluster1

Access Cluster | Delete Cluster

Cluster Details

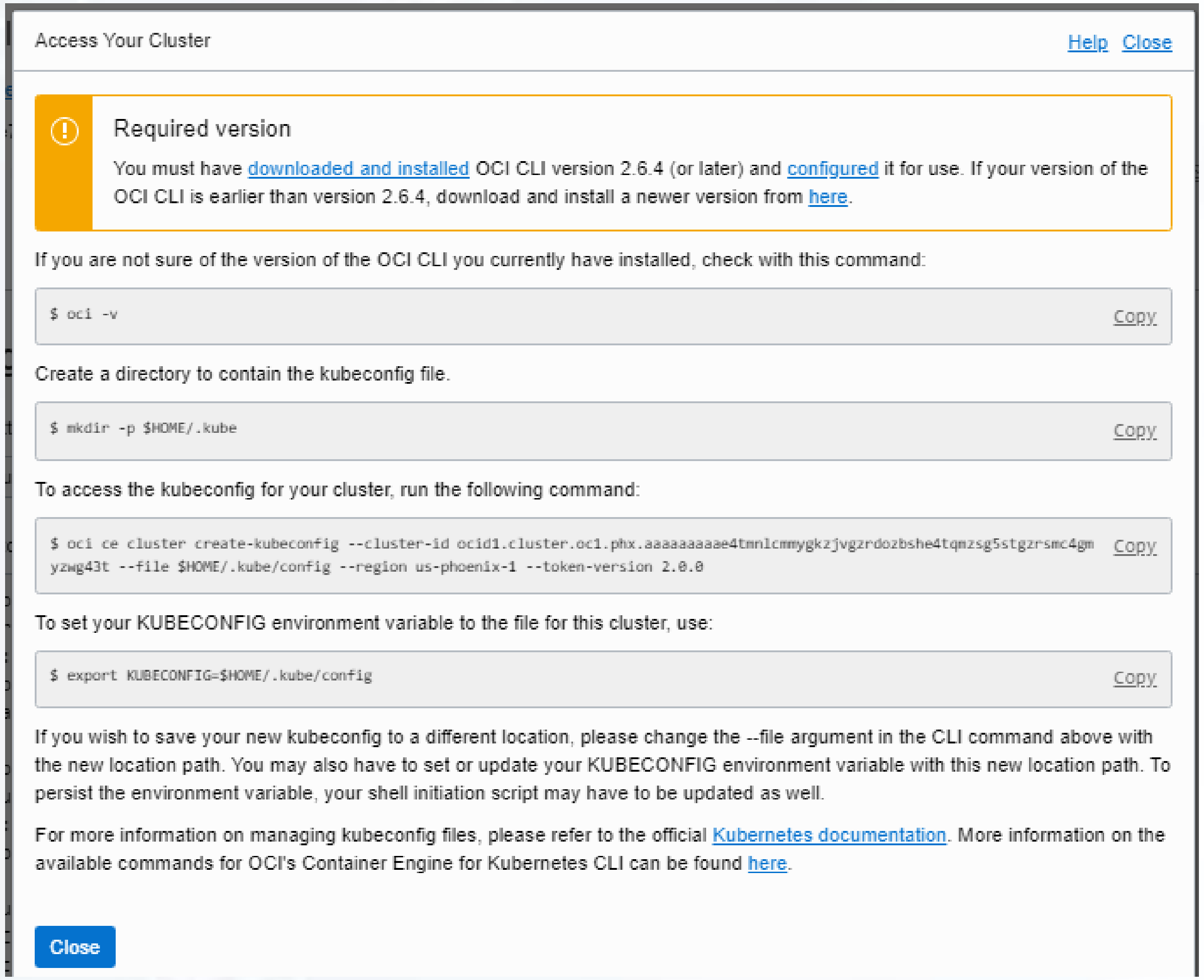
Cluster Status: ✓ Active
Node Pools: 1
Cluster Id: ...c4gmyzwg43t Show Copy
Compartment: rnsangwan (root)
Launched: Wed, 25 Mar 2020 09:53:59 GMT
Created By: ramnsangwan@gmail.com

Network Information

VCN Name: oke-vcn-quick-cluster1-35e2650af
CN Id: ...fhme75ia Show Copy
ompartment: rnsangwan (root)

Resources

- Metrics
- Node Pools
- Work Requests
- Quick Start
- Access Kubernetes Dashboard



Access Your Cluster — Help Close

Required version

You must have downloaded and installed OCI CLI version 2.6.4 (or later) and configured it for use. If your version of the OCI CLI is earlier than version 2.6.4, download and install a 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
```

Create a directory to contain the kubeconfig file.

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

To access the kubeconfig for your cluster, run the following command:

```
$ oci ce cluster create-kubeconfig --cluster-id ocid1.cluster.oc1.phx.aaaaaaaaae4tmnlcmmygkzjvgzrdozbshe4tqmzsg5stgzrsmc4gm
yzwg43t --file $HOME/.kube/config --region us-phoenix-1 --token-version 2.0.0
```

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 location path. To persist the environment variable, your shell initiation script may have to be updated as well.

For more information on managing kubeconfig files, please refer to the official Kubernetes documentation. More information on the available commands for OCI's Container Engine for Kubernetes CLI can be found here.

Close

Step 2: Verify that kubectl can access the cluster

- Enter the following command in the Cloud Shell window:

```
$ kubectl get nodes
```

  Information about the nodes in the cluster is shown.

- You can now use kubectl to perform operations on the cluster.

The Kubeconfig Files

- A single kubeconfig file can include the details for multiple clusters, specified by the ***current-context:*** element.
- A kubeconfig file includes an OCI CLI command that dynamically generates an authentication token and inserts it when you run a ***kubectl*** command.
- The OCI CLI must be available on your shell's executable path.
- The OCI CLI command in the kubeconfig file uses your current CLI profile when generating an authentication token.
- If you have defined multiple profiles in different tenancies in ***~/.oci/config***, specify which profile to use when generating the authentication token.

The kubeconfig file Example

```
user:
  exec:
    apiVersion: client.authentication.k8s.io/v1beta1
    args:
    - ce
    - cluster
    - generate-token
    - --cluster-id
    - <cluster ocid>
    - --profile
    - <profile-name>
    command: oci
    env: []
```

- Set the OCI_CLI_PROFILE environment variable to the name of the profile defined:

```
$ export OCI_CLI_PROFILE=<profile-name>
$ kubectl get nodes
```

Starting Kubernetes Dashboard

- Kubernetes Dashboard is a web-based user interface used to:
  - deploy containerized applications to a Kubernetes cluster
  - troubleshoot your containerized applications
  - get an overview of applications running on a cluster and create/modify individual Kubernetes resources.
- You have to deploy the dashboard manually.

Start the Kubernetes Dashboard

Deploy the Dashboard manually:

```
# kubectl apply -f https://raw.githubusercontent.com/kubernetes/dashboard/v2.0.3/aio/deploy/recommended.yaml
```

1. In a text editor, create a file (e.g. oke-admin-service-ccount.yaml):

   It defines an administrator service account and a clusterrolebinding, both called oke-admin.

2. Create the service account & clusterrolebinding in the cluster:

   ```
   # kubectl apply -f oke-admin-service-account.yaml
   ```

- You can now use the oke-admin service account to view and control the cluster, and to connect to the Kubernetes dashboard.

Start the Kubernetes Dashboard

3. Obtain an authentication token for the oke-admin service account :

```
# kubectl -n kube-system describe secret $(kubectl -n kube-system get secret | grep oke-admin | awk '{print $1}')
```

*The output includes an authentication token as the value of **token: element**:*

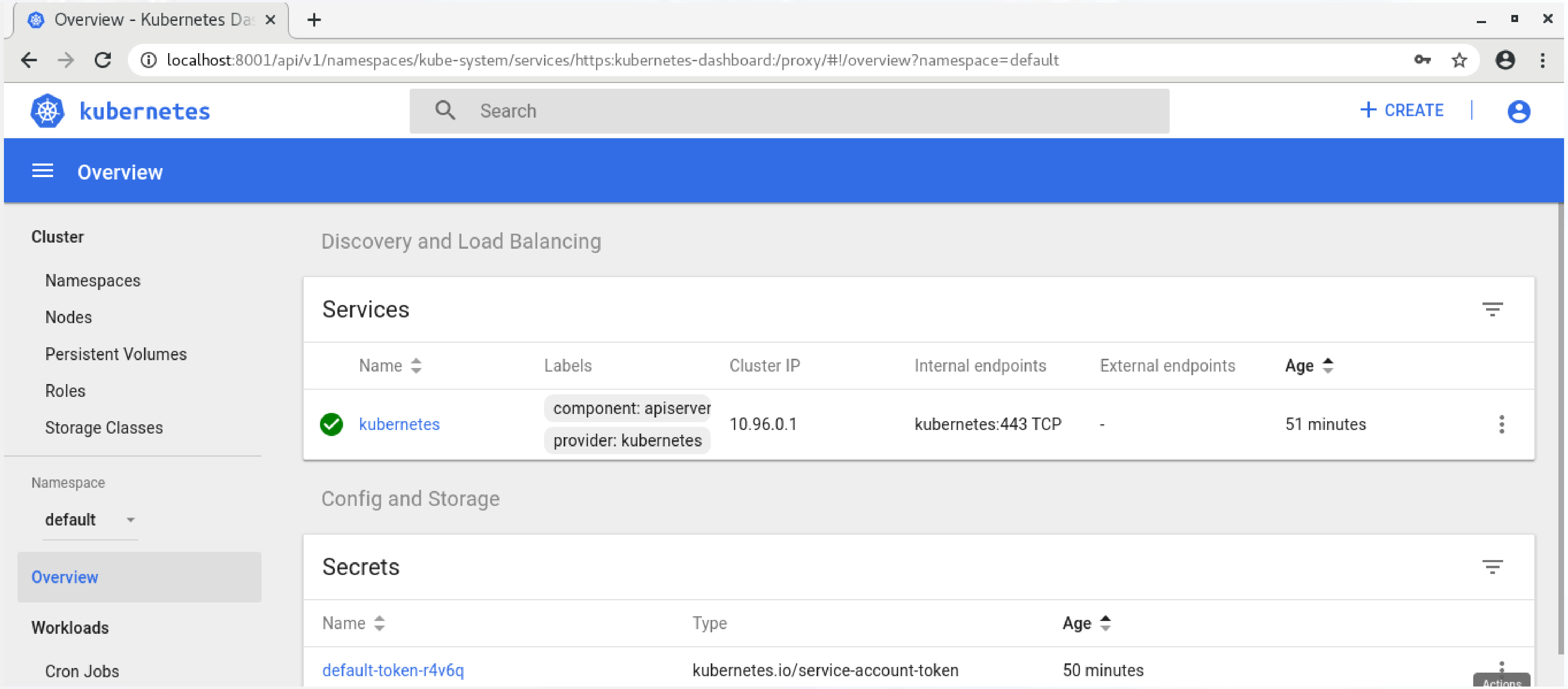
```
Name: oke-admin-token-gwbp2
…….
namespace: 11 bytes
token: eyJh______px1Q
…….
```

4. Copy the value of the token: element from the output.
5. In a terminal window, enter ***kubectl proxy*** to start the Kubernetes Dashboard.

Start the Kubernetes Dashboard

6. Open a browser and visit
   http://localhost:8001/api/v1/namespaces/kubernetes-dashboard/services/https:kubernetes-dashboard:/proxy/
7. In the Kubernetes Dashboard, select Token and paste the value of the token: element you copied earlier into the Token field.
8. In the Kubernetes Dashboard, click Sign In, and then click Overview to see the applications deployed on the cluster.

Dashboard



Overview - Kubernetes Das
localhost:8001/api/v1/namespaces/kube-system/services/https:kubernetes-dashboard:/proxy/#!/overview?namespace=default
kubernetes
Search
+ CREATE
Overview
Cluster
Namespaces
Nodes
Persistent Volumes
Roles
Storage Classes
Namespace
default
Overview
Workloads
Cron Jobs

Discovery and Load Balancing

Services

	Name	Labels	Cluster IP	Internal endpoints	External endpoints	Age
✓	kubernetes	component: apiserver provider: kubernetes	10.96.0.1	kubernetes:443 TCP	-	51 minutes

Config and Storage

Secrets

Name	Type	Age
default-token-r4v6q	kubernetes.io/service-account-token	50 minutes

https://kubernetes.io/docs/reference/kubectl/kubectl/

Adding Service Account Auth Token to Kubeconfig

- A service account has an associated authentication token, which is stored as a Kubernetes secret.
- You bind service account to a clusterrolebinding that has cluster administration permissions.
- You can then add the service account and its authentication token as a user definition in the kubeconfig file itself.

https://docs.cloud.oracle.com/en-us/iaas/Content/ContEng/Tasks/contengaddingserviceaccttoken.htm

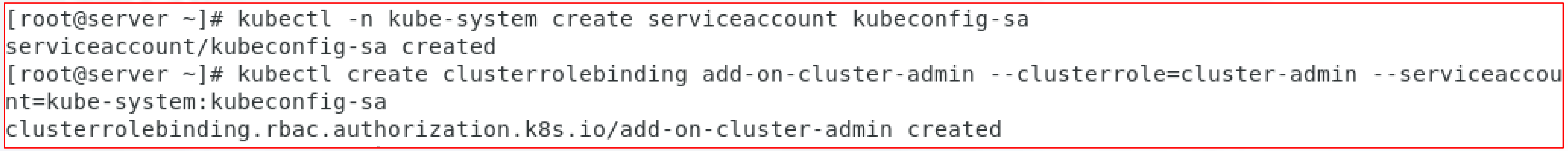
Adding Service Account Auth Token to Kubeconfig

1. Create a new service account, kubeconfig-sa, in the kube-system namespace with ***kubectl***:

   # kubectl -n kube-system create serviceaccount kubeconfig-sa

2. Create a new clusterrolebinding with cluster administration permissions and bind it to the service account you just created:

   # kubectl create clusterrolebinding add-on-cluster-admin --clusterrole=cluster-admin --serviceaccount=kube-system:kubeconfig-sa



```
[root@server ~]# kubectl -n kube-system create serviceaccount kubeconfig-sa
serviceaccount/kubeconfig-sa created
[root@server ~]# kubectl create clusterrolebinding add-on-cluster-admin --clusterrole=cluster-admin --serviceaccou
nt=kube-system:kubeconfig-sa
clusterrolebinding.rbac.authorization.k8s.io/add-on-cluster-admin created
```

Adding Service Account Auth Token to Kubeconfig

3. Obtain the name of the service account authentication token and assign its value to an environment variable:

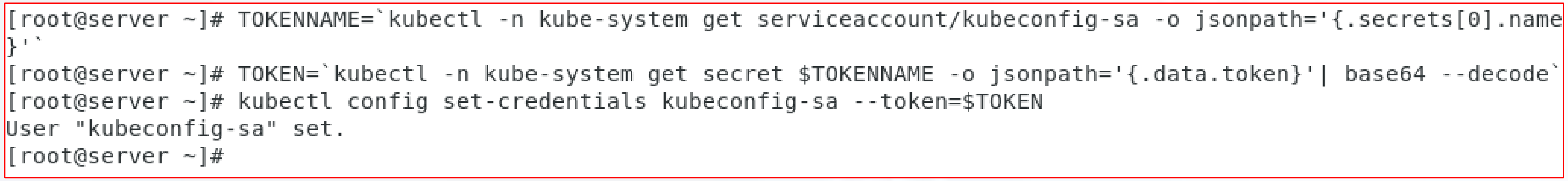
   # TOKENNAME=`kubectl -n kube-system get serviceaccount/kubeconfig-sa -o jsonpath='{.secrets[0].name}'`

4. Obtain the value of the service account authentication token and assign its value (decoded from base64) to an environment variable:

   # TOKEN=`kubectl -n kube-system get secret $TOKENNAME -o jsonpath='{.data.token}'| base64 --decode`

5. Add the service account (and its authentication token) as a new user definition in the kubeconfig file by entering the following kubectl command:

   # kubectl config set-credentials kubeconfig-sa --token=$TOKEN



```
[root@server ~]# TOKENNAME=`kubectl -n kube-system get serviceaccount/kubeconfig-sa -o jsonpath='{.secrets[0].name
}'`
[root@server ~]# TOKEN=`kubectl -n kube-system get secret $TOKENNAME -o jsonpath='{.data.token}'| base64 --decode`
[root@server ~]# kubectl config set-credentials kubeconfig-sa --token=$TOKEN
User "kubeconfig-sa" set.
[root@server ~]#
```

Adding Service Account Auth Token to Kubeconfig

6. Set the user specified in the kubeconfig file for the current context to be the new service account user you created:

```
$ kubectl config set-context --current --user=kubeconfig-sa
```

7. To verify that authentication works as expected, run a ***kubectl*** command.

```
# kubectl get pods  --all-namespaces
```

Deploying a Sample Nginx App on a Cluster Using Kubectl

- The Quick Start tab (on Cluster page) makes it easy to view and copy the commands to:
  - set up access to the cluster
  - download and deploy a sample Nginx application using the Kubernetes command line tool kubectl from the instructions in a manifest file

Deploy a sample nginx Application

1. In a terminal window, deploy the sample Nginx application by entering

```
kubectl create -f https://k8s.io/examples/application/deployment.yaml
```

2. Use the Kubernetes Dashboard or kubectl to confirm that the sample application has deployed successfully. For example:
   a. Enter kubectl proxy to start the Kubernetes Dashboard.
   b. Open a browser and go to

      http://localhost:8001/api/v1/namespaces/kubesystem/services/https:kubernetes-dashboard:/proxy/
   c. Click Overview to see the applications deployed on the cluster.

      You can see the Nginx sample application has been deployed as two pods, on two nodes in the cluster.

Pulling Images from Registry During Deployment

- Two steps required:
  - Use ***kubectl*** to create a Docker registry secret.
  - Specify the image to pull from OCI Registry, including the repository location and the Docker registry secret to use, in the application's manifest file.

Create a Docker registry secret

In a terminal window, enter:

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

- ***<region-key>*** is the key for the OCI Registry region you're using.
- ***ocir.io*** is the OCI Registry name.
- ***<oci-username>*** is the username to use when pulling the image, e.g rnsangwan@gmail.com .

For example:

```
$ kubectl create secret docker-registry ocirsecret --docker-server=phx.ocir.io \
--docker-username='idaffnzwmy7k/rnsangwan@gmail.com'  \
--docker-password='k]j64r{1sJSSF-;)K8'  \
--docker-email='rnsangwan@gmail.com’
```

Specify The image and Secret

1. Open the application's manifest file in a text editor.
2. Add the following sections:
   a. Add a containers section that specifies the name and location of the container you want to pull from OCIRegistry, along with other deployment details.
   b. Add an imagePullSecrets section to the manifest file that specifies the name of the Docker secret you created to access the OCIRegistry.

Specify The image and Secret

Example of manifest with imagePullSecrets sections:

```
apiVersion: v1
kind: Pod
metadata:
  name: ngnix-image
spec:
  containers:
    - name: ngnix
      image: phx.ocir.io/idaffnzwmy7k/project01/ngnix-lb:latest
      imagePullPolicy: Always
      ports:
      - name: nginx
        containerPort: 8080
        protocol: TCP
  imagePullSecrets:
    - name: ocirsecret
```

3. Save and close the manifest file.

Encrypting Kubernetes Secrets at Rest in Etcd

- Before you can create a cluster where Kubernetes secrets are encrypted, you have to:
  - know the name and OCID of a suitable master encryption key in Vault
  - create a dynamic group that includes all clusters in the compartment in which you are going to create the new cluster
  - create a policy authorizing the dynamic group to use the master encryption key
- Policies must have been defined to authorize Container Engine for Kubernetes
- After you've specified a master encryption key for a new cluster and created the cluster, do not subsequently delete the master encryption key in the Vault service.

About Access Control and Container Engine for Kubernetes

- To perform operations on a Kubernetes cluster, you must have appropriate permissions to access the cluster.
- IAM provides control over:
  - whether a user can create or delete clusters
  - whether a user can add, remove, or modify node pools
  - which Kubernetes object create/delete/view operations a user can perform on all clusters within a compartment or tenancy

Thank You