Deploying Oracle Digital Assistant Custom Component APIs to Oracle Kubernetes Cluster – Part 3 of 3. – Setting up a developer account for deploying custom components to OKE cluster

Abhay Bhavsar, September 2023

Part 1 of this series of 3 articles is a prerequisite for this article. Part 1 describes how to setup an Oracle Kubernetes Engine Cluster (i.e. OKE). Part 2 guides on how to deploy Oracle Digital Assistant custom components to OKE as an OCI Administrator. Part 3 will provide instructions on how to configure the necessary permissions to a *developer* account. This developer account will have a limited role in which to access and deploy to the OKE cluster.

Disclaimer

The article will list OCI resources that were automatically configured upon completion of the terraform scripts from Part 1 of this series. Please use the OCI Cost Estimator to perform cost calculation to understand the cost associated with this setup. The author is not responsible for the costs incurred by the setup created.

Cost Estimator - https://www.oracle.com/cloud/costestimator.html

1 Summary and Outline

1.1 Before you begin

The article assumes that the OCI administrator and the developers have a basic understanding of OCI resources like Virtual Cloud Network (VCN), Load Balancers, Oracle Cloud Infrastructure Registry (OCIR), OCI Compute, OCPU and Memory. Knowledge about OCIR Oracle API Gateway Cloud, OCI Vault is not necessary, but it would help. An OCI Administrator access is required to complete the setup. This article creates a new IAM user and adds that user to the Administrator group on OCI.

1.2 Resources

With this article you will find a **resources-part3.zip** file. Download and unzip the file. Here are the included files and their purpose that are applicable to this article.

- docker-compose : The docker compose command to build any artifact as a docker image
- OKE-ARTICLE.postman_collection.json : Postman collection to test the APIs through Oracle API Gateway
- Part3-commands.txt: file containing commands used in this article.

1.3 Access Requirements

In Part 1 of this series an OCI administrator account was created. This administrator account will be used throughout this article, like it was in Parts 1 and 2 of this series, to run the various scripts and the OCI commands.

Note: Configuration steps in this article require OCI administrator role access

2 Create an IAM developer account on OCI

2.1 Create a developer account okedeveloper

Within the OCI console, login, as user OKEAdmin, to the same tenancy where the OKE cluster was created (Part 1 of this series).

OKEAdmin refers to the administrator user that was created in Part 1 of this series. Use what is the administrator user that was created. However, for simplicity, this article will use OKEAdmin whenever the admin user is referred to.

https://cloud.oracle.com/?region=us-ashburn-1

Change the URL as per your region.

Select the main menu ☰ ORACLE Cloud icon. Next, select the Identity and Security menu option in the left menu pane, and then select Users.

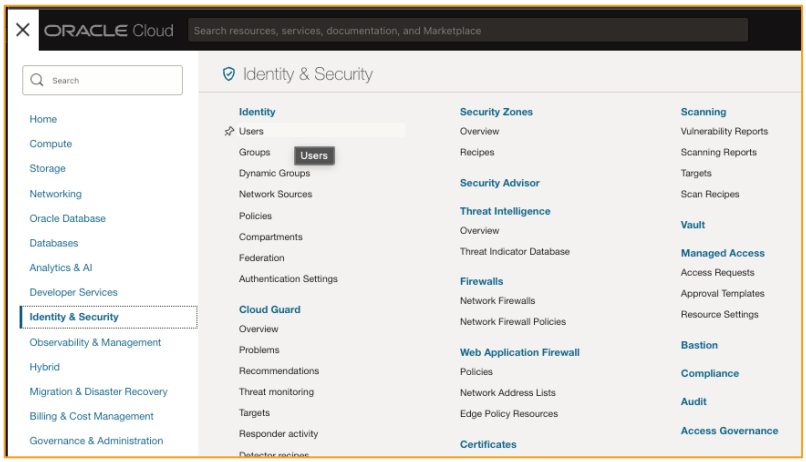

Figure 1: Visit User Management in Identity & Security

From the User screen, select the **Create User** button.

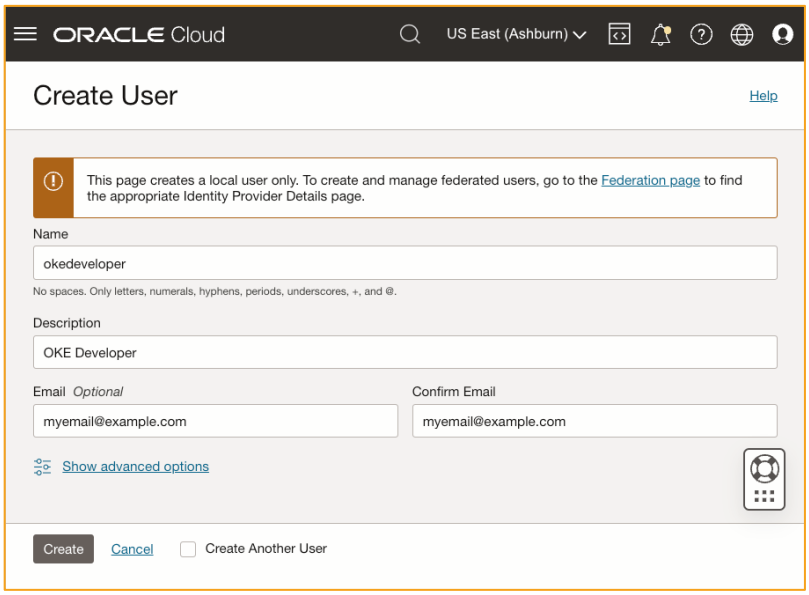

Figure 2: Create okedeveloper user account

This screenshot will be different, if the tenancy is secured through IDCS. The Create User dialog will also provide an option (IDCS or IAM) where to create the user. Choose the appropriate User Type.

In the Create User dialog, where the User Type is IAM, complete the following sections:

- Username: **okedeveloper**
- Email address (optional). Although optional, completing the email will allow the user to reset the password using Forgot Password option.

Next, select the **Create** button to complete the new user creation.

2.2 Create a temporary password

Once the new developer user is created, select **Create/Reset Password.** In the proceeding dialog, select the Create/Reset Password button, which will provide an option to copy the temporary password.

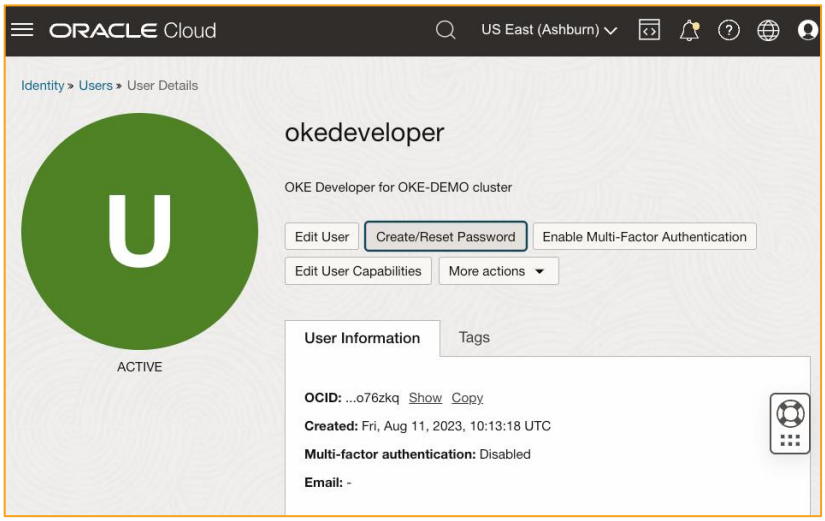

Figure 3: Create/Reset Password

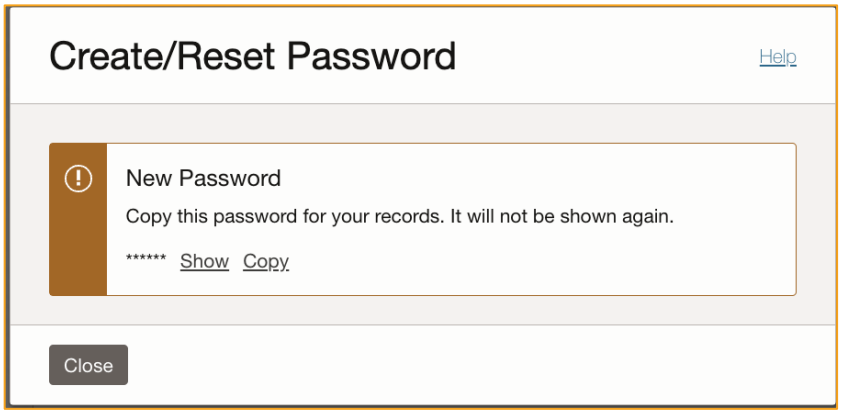

Figure 4: Copy the temporary password

Save this temporary password securely and provide it to the **okedeveloper** user. The okedeveloper will use this temporary password to reset the password once the user logs into the OCI console for the first time.

The next sections of this article will continues to use the OKEAdmin account to configure the necessary permissions for the new okedeveloper account.

2.3 Create an Auth token.

In the okedeveloper user screen, select the **Auth Tokens** menu option. The console screen will review the Auth Tokens section. Select the **Generate Token** button. This will bring up the Generate Token dialog where you can add in a Description similar to this: *This token will be required for user to perform a docker login to the OCIR (e.g. repository)*. Next, select the **Generate Token** button to complete the token creation.

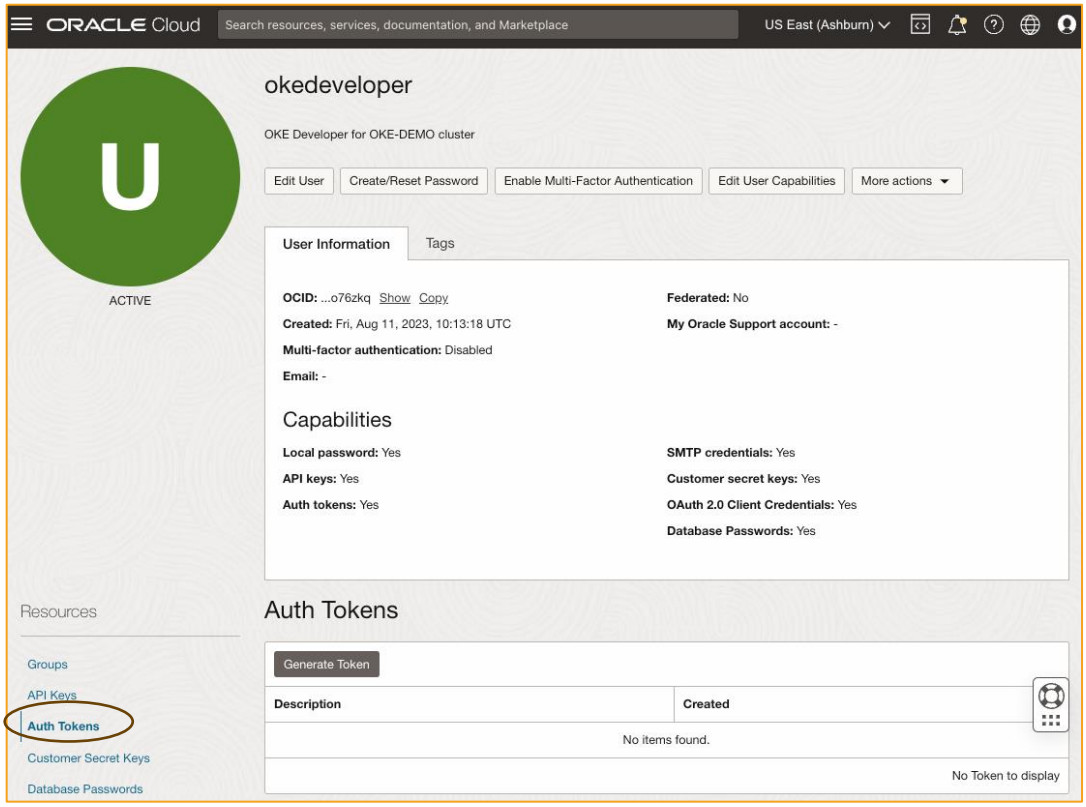

Figure 5: Auth Token

Generate the Auth token and save this securely to be supplied to the owner of the okedeveloper.

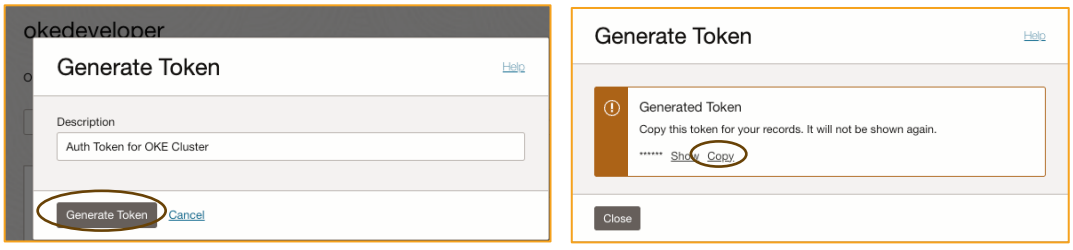

Figure 6: Generate, copy and save Auth token.

2.4 Create a group OKEDeveloperGroup and add okedeveloper to this group

In the Identity & Security section, create a new group named: **OKEDeveloperGroup** and add the **okedeveloper** user to this group.

Note the OCID for the group as shown below. This is OCID for the group and not for the user. Once the OKE cluster is ready. The OKEAdmin user can setup a role based, security access control for the cluster based on the OCID of the group.

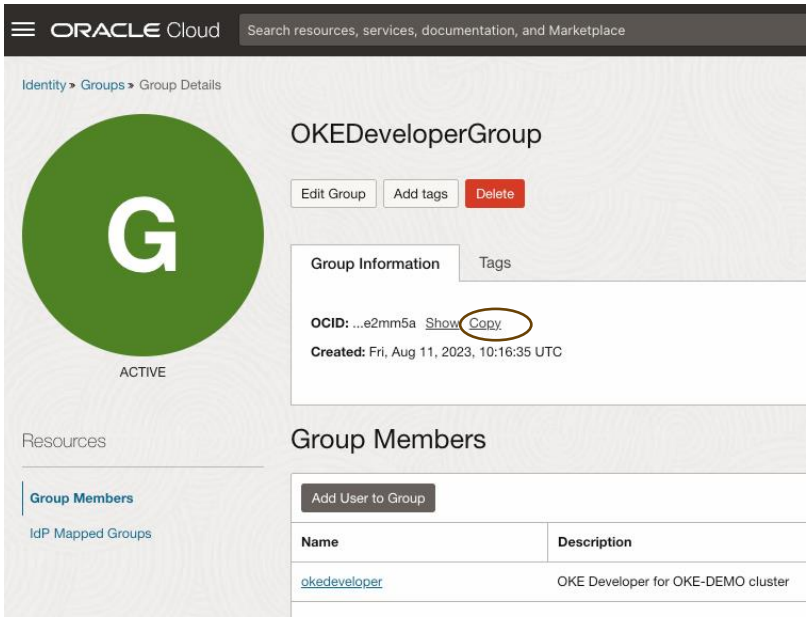

Figure 7: Capture OCID of the newly created compartment

Note the OCID of the OKEDeveloper group. This required to set up role based security for accessing OKE cluster.

2.5 Create Policies

Once the group is created, the next step is to create the policies for the group. These permissions will enable the execution of the various tasks such as accessing and deploying new application to the OKE cluster. The okedeveloper user, who is part of this group, will then be able to execute these tasks. Here are the list of permissions required:

- Cloud Shell & Code Editor – Run docker and kubectl commands and edit code in the code editor
- OCI Registry – Push docker images to OCI registry
- OKE cluster – Deploy pods to the cluster using kubectl commands
- Object storage - View logs

2.5.1 Cloud Shell Access

Navigate to the main Identity & Security page and in the left menu pane, select **Policies**. Next select the **Create Policy** button. In the Create Policy dialog, complete the required information. For the **Compartment,** select the (**root**) compartment.

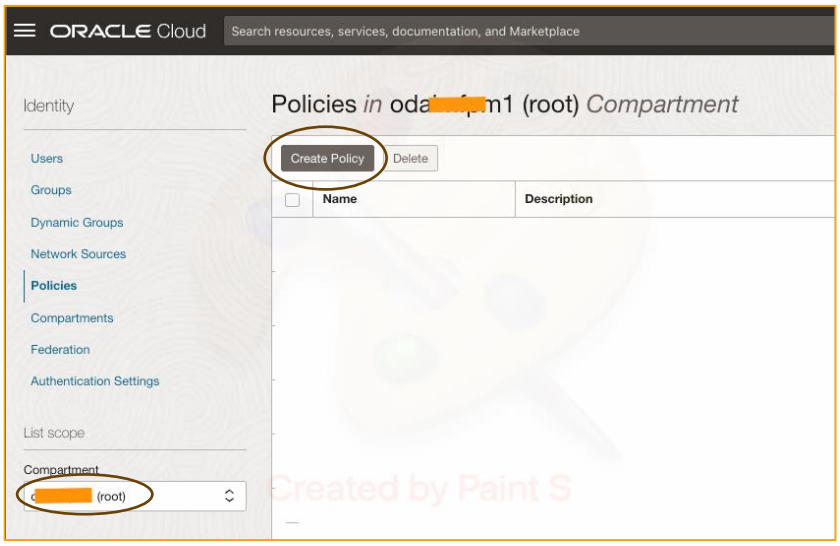

Figure 8: Policy for OKEDeveloperGroup

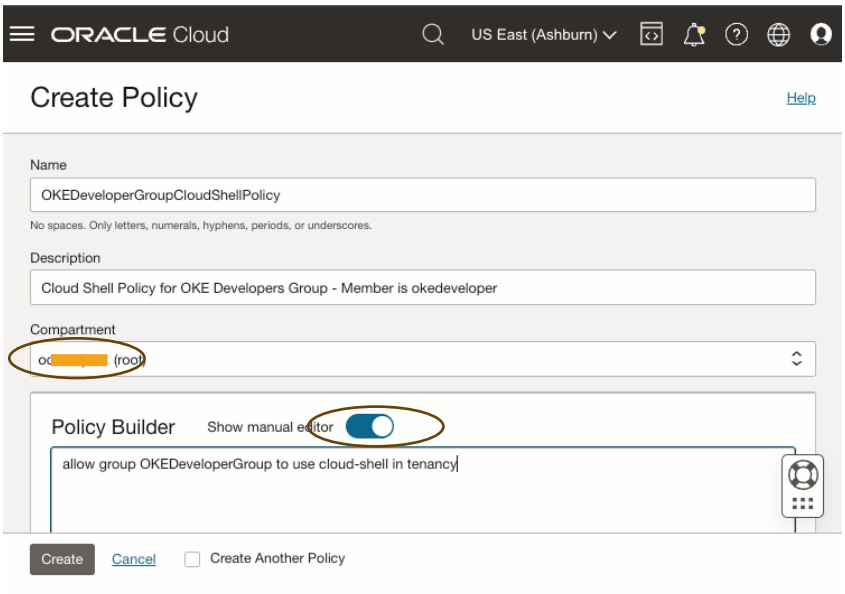

Figure 9: Create cloud shell policy for OKEDeveloperGroup

Next, in the **Policy Builder**, enable the **Show manual editor** option and then add the following statement:

```
allow group OKEDeveloperGroup to use cloud-shell in tenancy
```

Select the **Create** button to complete the Policy creation.

2.5.2 Compartment Level Access

This section will create the policies for the compartment where OKE cluster is configured (e.g. OKE-DEMO). Follow the same steps in 2.5.1 to create a new policy. However, select the **OKE-DEMO** compartment. In the Create Policy dialog, add in the following statements using the manual editor.

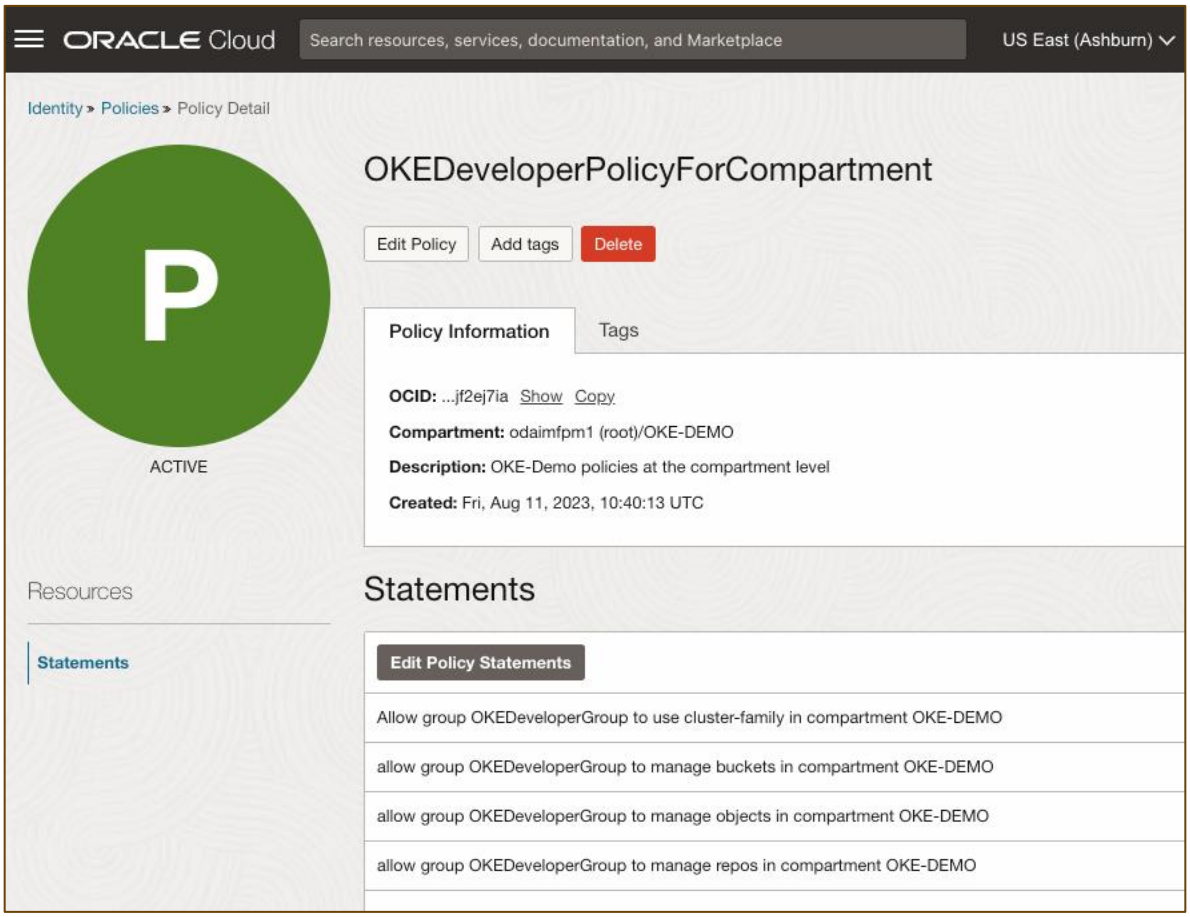

Figure 10: Compartment level policies

```
allow group OKEDeveloperGroup to use cluster-family in compartment OKE-DEMO
allow group OKEDeveloperGroup to manage buckets in compartment OKE-DEMO
allow group OKEDeveloperGroup to manage objects in compartment OKE-DEMO
allow group OKEDeveloperGroup to manage repos in compartment OKE-DEMO
```

Ensure you select the compartment where OKE cluster is configured. The managed verb used in the statement for both buckets and objects can be switched to use. However, in order to deploy images the verb managed must be used for repos (e.g. OCI registry).

You may choose to provide “use” access rather than “manage” for the buckets and objects. But in order to provide permissions to deploy images this user must have manage permissions on repos i.e. OCI Registry.

2.6 Implement role based security for your OKE Cluster

The OKEAdmin can control access to the OKE cluster by implementing role based access control.

The YAML script for supporting this is provided in the **resources-part3.zip** in the file **oke-rbac.yaml**.

Upload the oke-rbac.yaml script to the Cloud Shell and store the file in the `~/OKE/roles` directory.

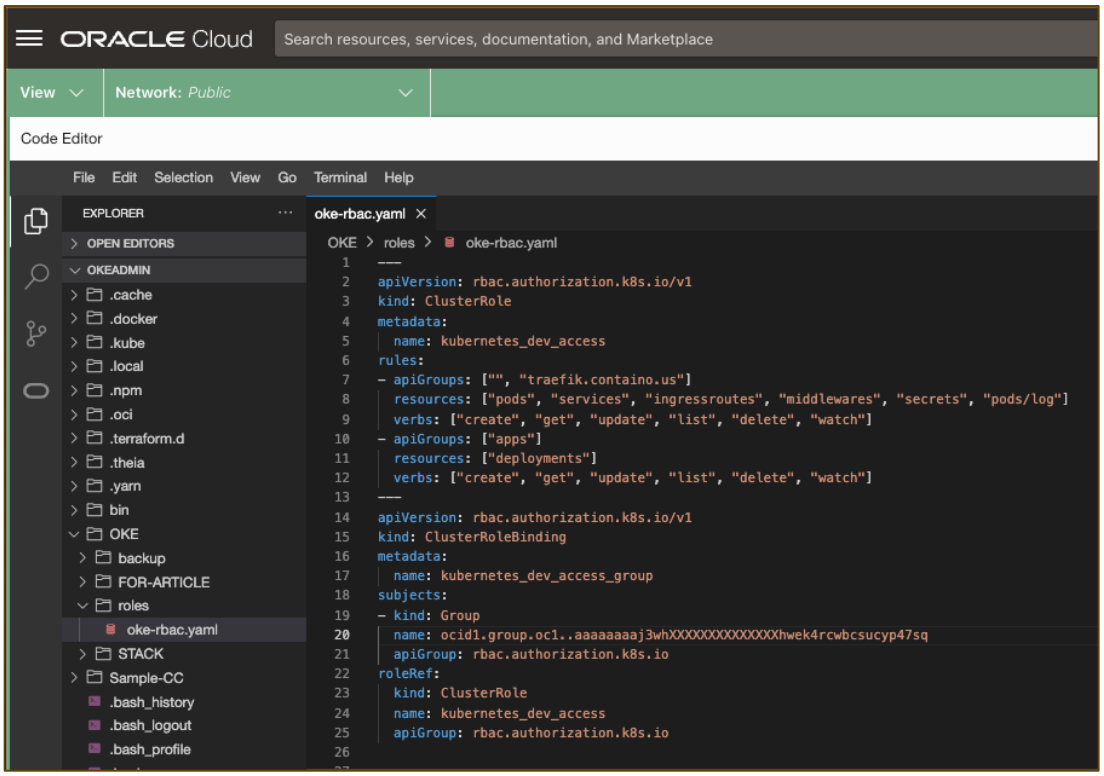

Figure 11: Edit the oke-rbac.yaml

In the Code Editor, edit the **oke-rbac.yaml** file (line 20) by replace the OCID of the group with the OCID of the OKEDeveloperGroup. The OCID was captured in step 2.4 of this article.

Open the Cloud Shell and navigate to the ~OKE/roles directory. Run the following command

```
kubectl apply -f oke-rbac.yaml
```

This will provide the necessary permissions for this group to access the cluster and to run various `kubectl` commands.

3 Verify access as okedeveloper

3.1 Login to OCI console

Log into tenancy using the **okedeveloper** user. If this user has not already done so, use the temporary password provided by the administrator to change the password to a permanent one.

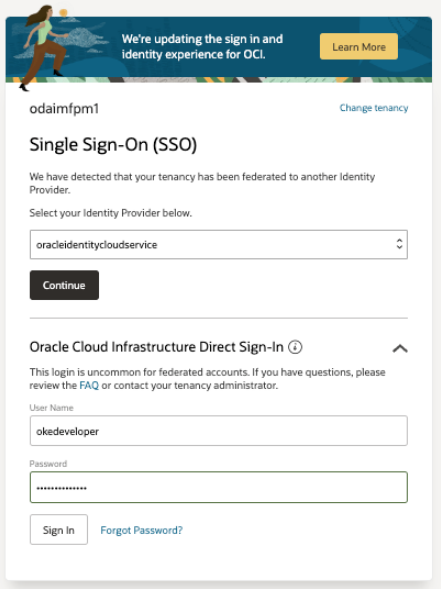

Figure 12: Login as okedeveloper using Oracle Cloud Infrastructure Direct Sign-In

3.2 Access OKE Cluster

Under Developer Services, select Kubernetes Clusters (OKE). In the proceeding screen, select the compartment (e.g. OKE-DEMO) to view the DEMO-OKE Cluster that was create in Part 1 of this series.

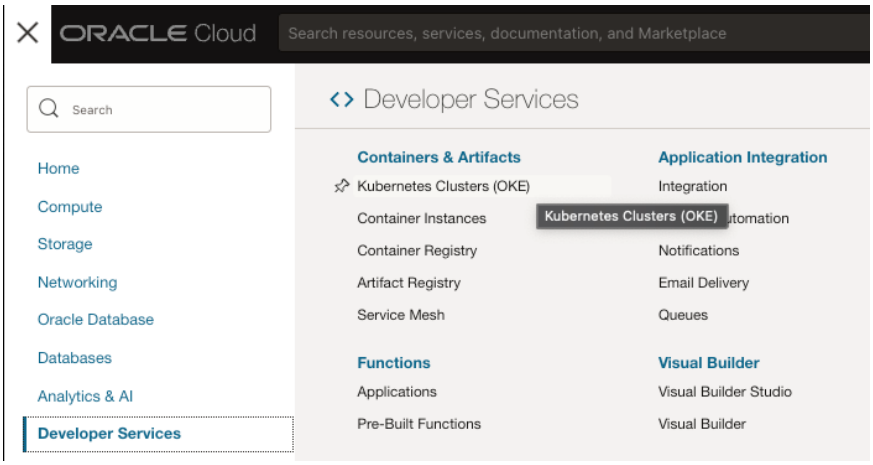

Figure 13: Access Kubernetes Cluster

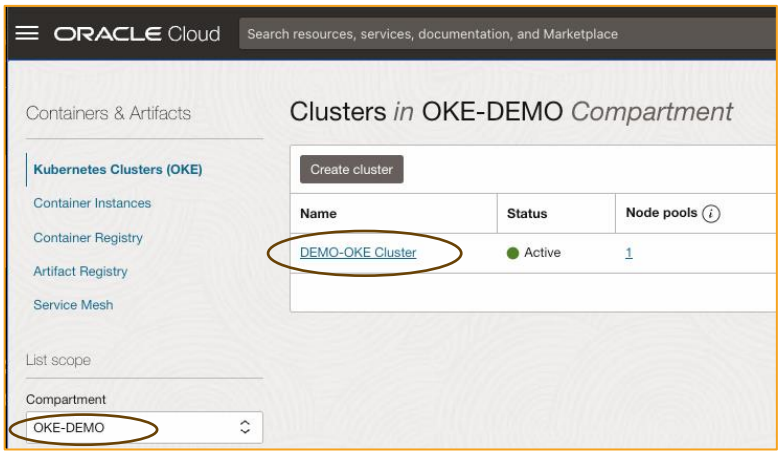

Figure 14: Find the cluster in the compartment

Next, slect the DEMO-OKE Cluster name and in the proceeding screen, select the **Access Cluster** button. In the Access Cluster dialog, ensure that the Cloud Shell Access is selected and then select the **Launch Cloud Shell** button. This will open the Cloud Shell below. Next, select the **Copy** link in the dialog's step 2. This will copy the cluster config command. Paste the contents into the Cloud Shell. Once done, access to the OKE cluster is enabled.

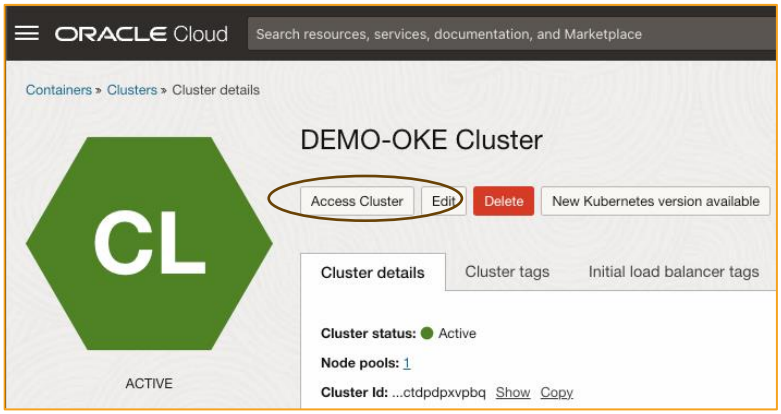

Figure 15: Access Cluster

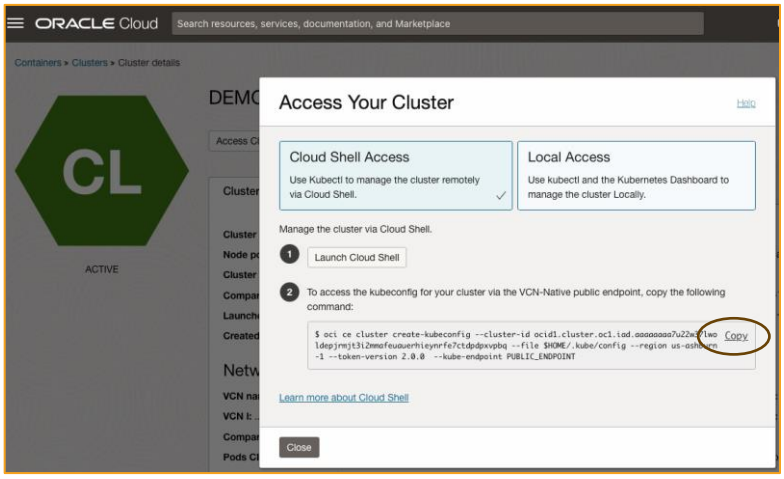

Figure 16: Launch Cloud Shell & Copy the cluster config command

3.3 Configure access to OKE cluster

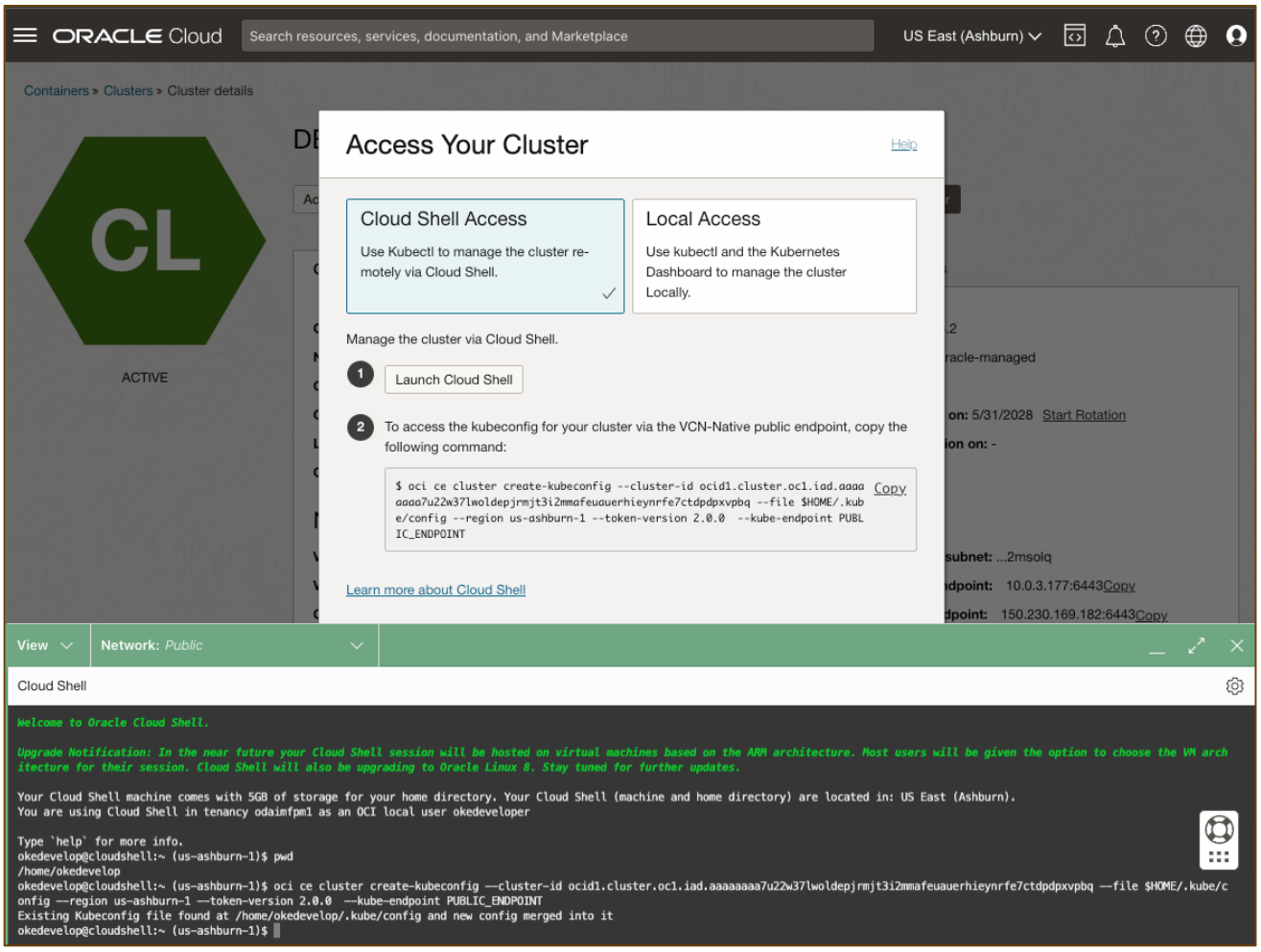

Figure 17: Configure OKE cluster access for okedeveloper account

3.4 List pods running

Run the following command to ensure that okedeveloper is able to list pods

```
kubectl get pods
```

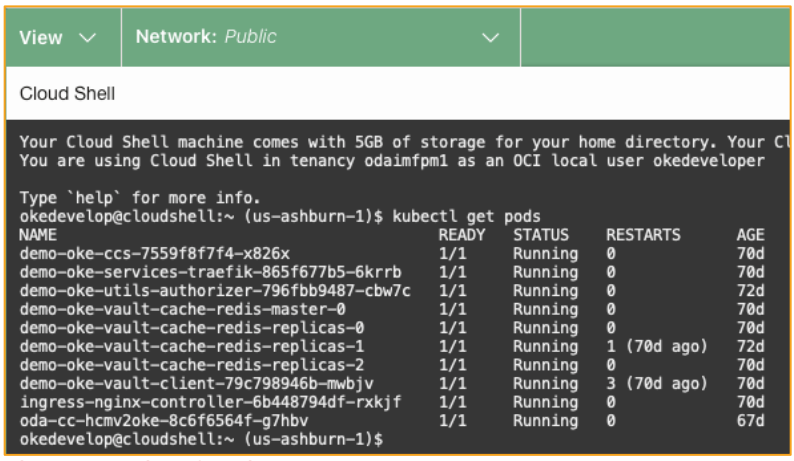

Figure 18: List of pods

If there is an authorization issue, there is a permissions issue. Contact the OKE administrator.

3.5 Verify access to OCI Registry

Access the OCI registry using the following URL:
https://cloud.oracle.com/compute/registry/containers?region=us-ashburn-1
Change the region code based on your region

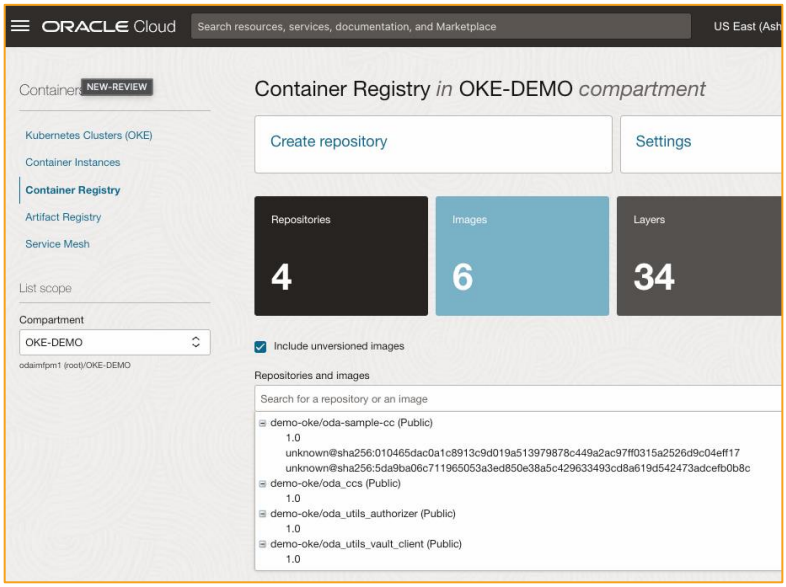

Figure 19: Access to OCI registry

3.6 Perform docker login

In the Cloud Shell, perform a docker login to check if the okedeveloper is able to login to the OCI registry.

Run the following command

```
docker login -u '<tenant namespace>/okedeveloper ' iad.ocir.io
```

When prompted, enter the Auth token that was created in Step 2.3 by the OKEAdmin.

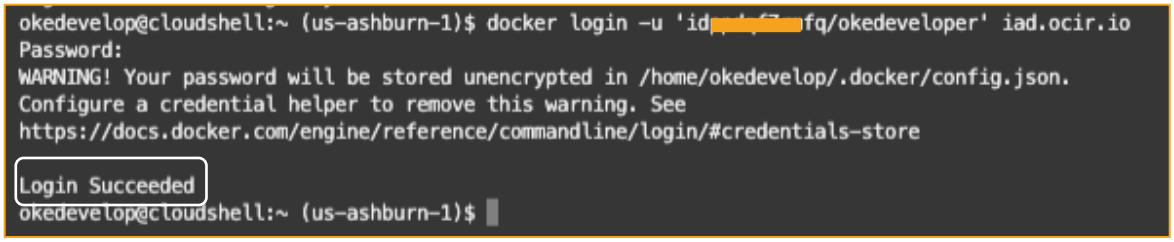

Figure 20: Successful docker login

The okedeveloper user can now follow the same steps in part 2 of the series, to build and deploy new or existing custom components and or other APIs to the OKE cluster.

This complete the configuration and verification for the okdeveloper account.

The developer owing the okedeveloper account can follow the same steps as indicated in part 2 of the series of articles to build and deploy new or existing custom components or other APIs to the OKE cluster. The steps provided in the part 2 are equally applicable to okedeveloper.

Conclusion

OKE cluster administrators can delegate the development activities to API developers for building and deploying custom components to the OKE cluster. By controlling access to the cluster through role based security and also implementing OCI policies at the compartment level, access and privileges can be controlled. But, still allowing for complete flexibility for developers to perform development and deployment activities.

Resources

Design Camp Session - Unleash The Power of OCI for ODA Part 1 - Custom Component Enterprise Deployment
https://videohub.oracle.com/media/Oracle+Digital+Assistant+Design+CampA+Unleash+The+Power+of+OCI+for+ODA+Part+1+-+Custom+Component+Enterprise+Deployment/1_rio2lqzq