Deploying Oracle Digital Assistant Custom Component APIs to Oracle Kubernetes Cluster – Part 1 of 3. – Setting up an Oracle Kubernetes Cluster on OCI

Abhay Bhavsar, September 2023

This article is part 1 of the series of 3 articles that describes how to setup an Oracle Kubernetes Engine Cluster (i.e. OKE).

Here are the 3 articles.

Part 1 Set up an Oracle Kubernetes Engine Cluster on OCI.

Part 2 Deploy a sample custom component to OKE.

Part 3 Delegate deployment of custom components to a developer by granting necessary permissions to access and deploy to the 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-part1.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
- STACK.zip : The terraform modules to configure and deploy a Kubernetes Cluster to your OCI Tenancy
- OKE-ARTICLE.postman_collection.json : Postman collection to test the APIs through Oracle API Gateway
- part1-commands.txt: The text file that lists all the commands that are used in this article

1.3 Access Requirements

This article assumes that the person executing steps in this article is an OCI Administrator. In one of the steps in the subsequent sections of this article an OCI administrator account is created. This account is then used throughout this article and other parts of the series of this articles to run various scripts and OCI commands.

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

2 Setting up OKE Cluster using terraform scripts

This section guides you through the entire process of setting up an OKE infrastructure in your tenancy. In addition, this section will describe how an OCI Administrator can configure, run and verify the OKE setup in an OCI tenancy.

Ensure you have OCI admin role access to an OCI Universal Credits Tenancy. Also, ensure that you have the appropriate quota and limits to create Vault and keys before executing the terraform scripts. It is advisable to confirm that you have not exhausted quota or limit for all the type of resources including Compute, VMs, Policies, etc.

This OKE cluster configuration requires the following steps..

- Create an OCI compartment
- Create an IAM User (e.g. OKEAdmin) and add this user to Administrators group.
- Create an Auth Token for the OKEAdmin account.
- Create a API Key for the OKEAdmin account.
- Create a developer IAM user okedeveloper, create Auth token.
- Create a Developers group OKEDevelopersGroup and add the user to this group.
- Configure the cloud shell for docker-compose and download necessary terraform scripts
- Review the terraform scripts, make necessary customizations or configuration changes
- Run terraform init, plan and fix any issues if reported
- Run terraform apply to deploy the infrastructure
- Review the services created
- Access the cluster, configure the cloud shell to access cluster and login to OCIR
- Review the Deployments through OCIR and verify authenticator and vault clients pods
- Verify sample custom component API through Oracle API Gateway Cloud

2.1 Create an OCI compartment

Refer to the documentation for more details on how to create OCI compartments.

Before you create a compartment, review the checklist below. These steps will gather the information that will be used by the docker-compose file:

- Check which region you want to create the OKE setup
- Check how many availability domains exist in the region selected
- Note down the availability domain prefix

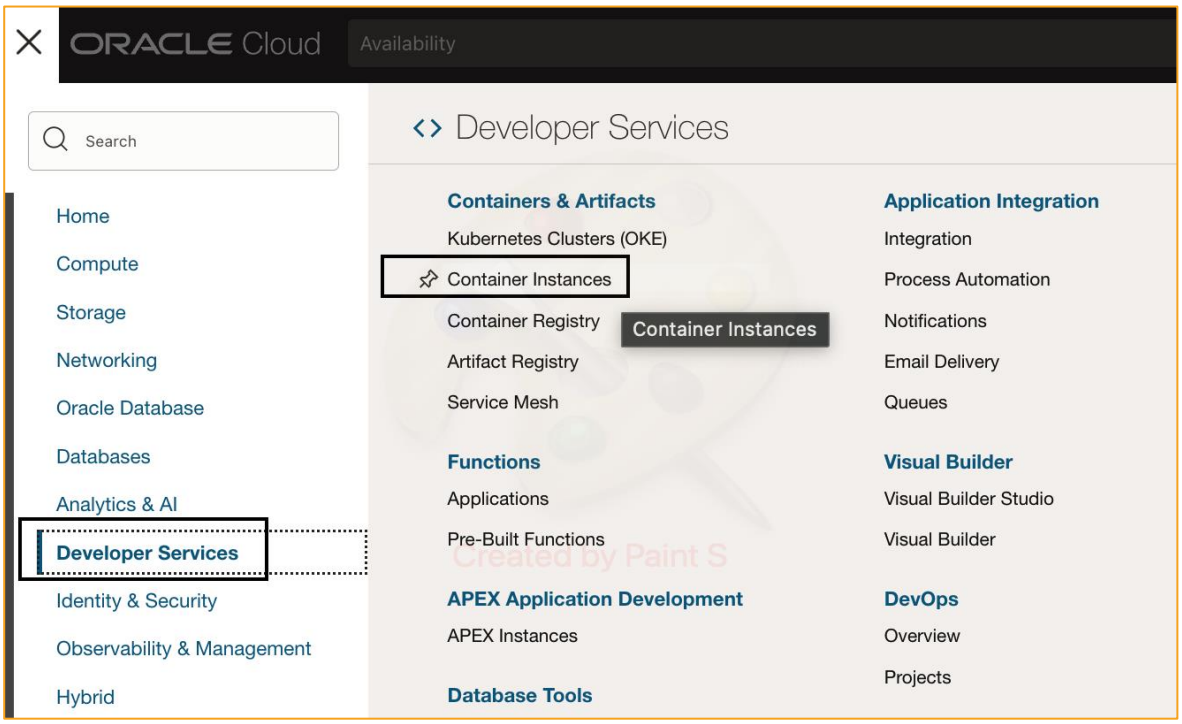


Figure 1: Developer Services → Container Instances

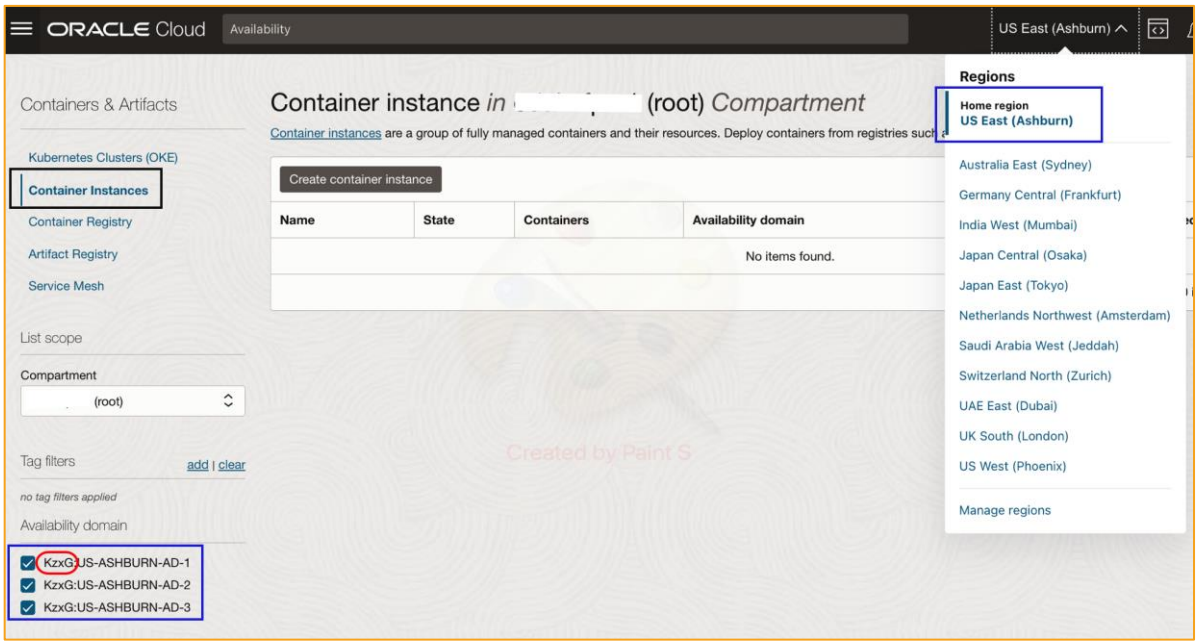


Figure 2: Choose Region, note Availability Domain

Note down the availability domains including the prefixes. This is required during configuration of the terraform scripts.

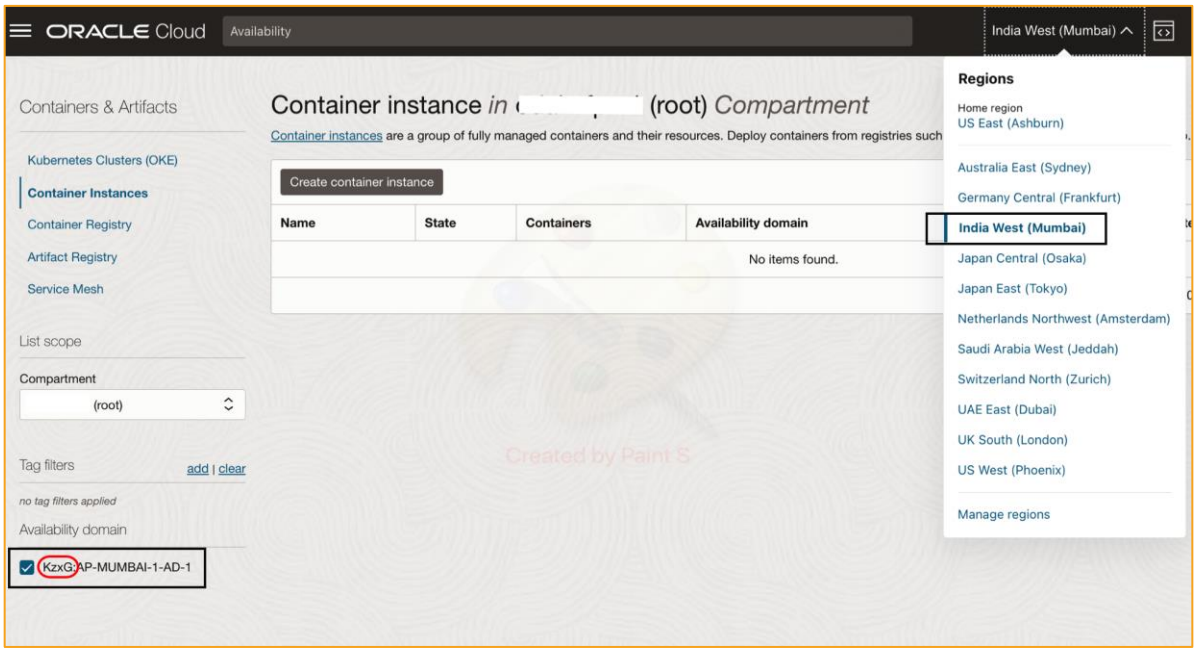


Figure 3: If required change Region, note Availability Domain

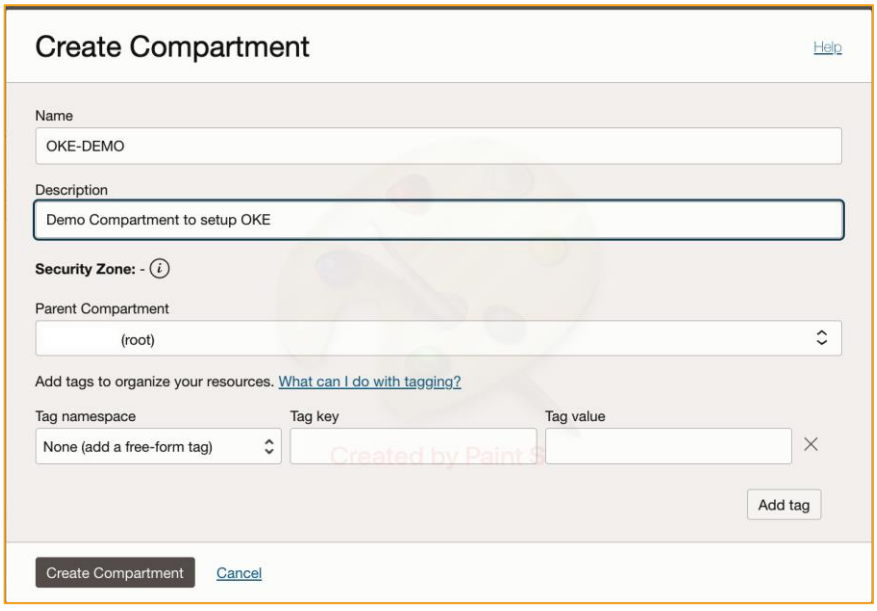


Figure 4: Create Compartment

2.2 Create an IAM User for example OKEAdmin and add this user to Administrators group.

The user OKEAdmin will be used for the admin user throughout this document. However, you can choose any name for the admin

Click on the main menu icon ≡ ORACLE Cloud. Choose **Users** from the **Identity** section,

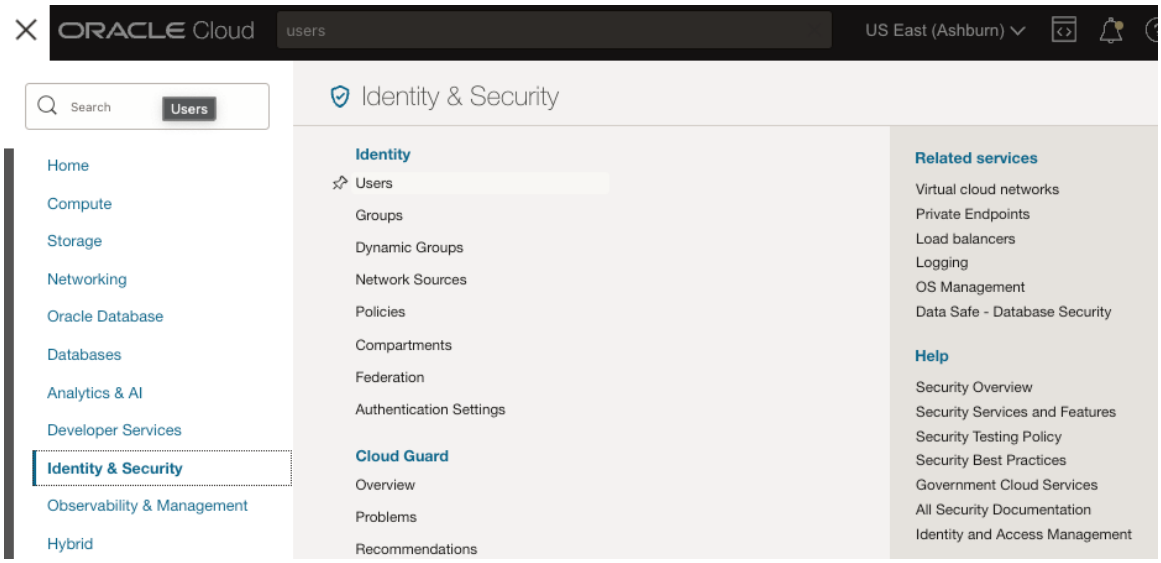


Figure 5-A Create IAM Admin User

Select the **Create User** button, then select **IAM User** option. In the Create IAM User dialog, enter a name (e.g. OKEAdmin) and description. Select Create to complete the new user creation.

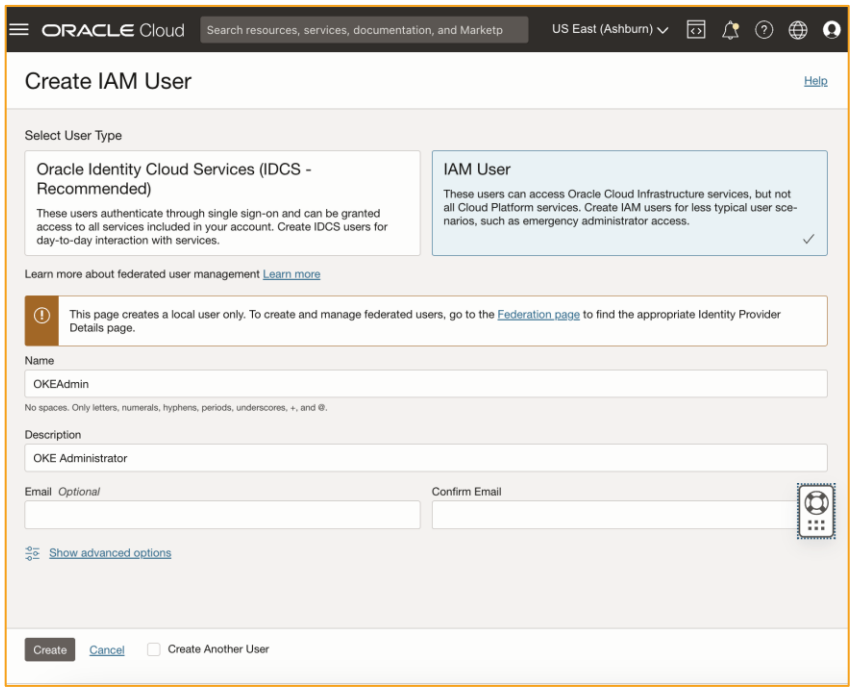


Figure 5-B: Create IAM Admin User

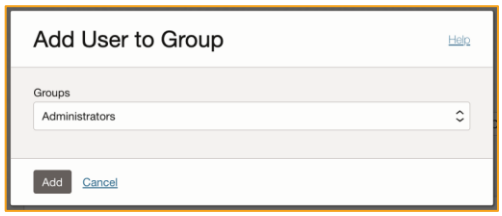


Figure 6: Add the Admin user to Administrators group

Once the new user has been created, under Groups, select the Add User to Group option. In the Add User to Group dialog, select the Administrators group and select Add.

For more information about managing IAM users and groups, refer to the documentation.

In the (created) admin user, select the **Create/Reset Password** button.

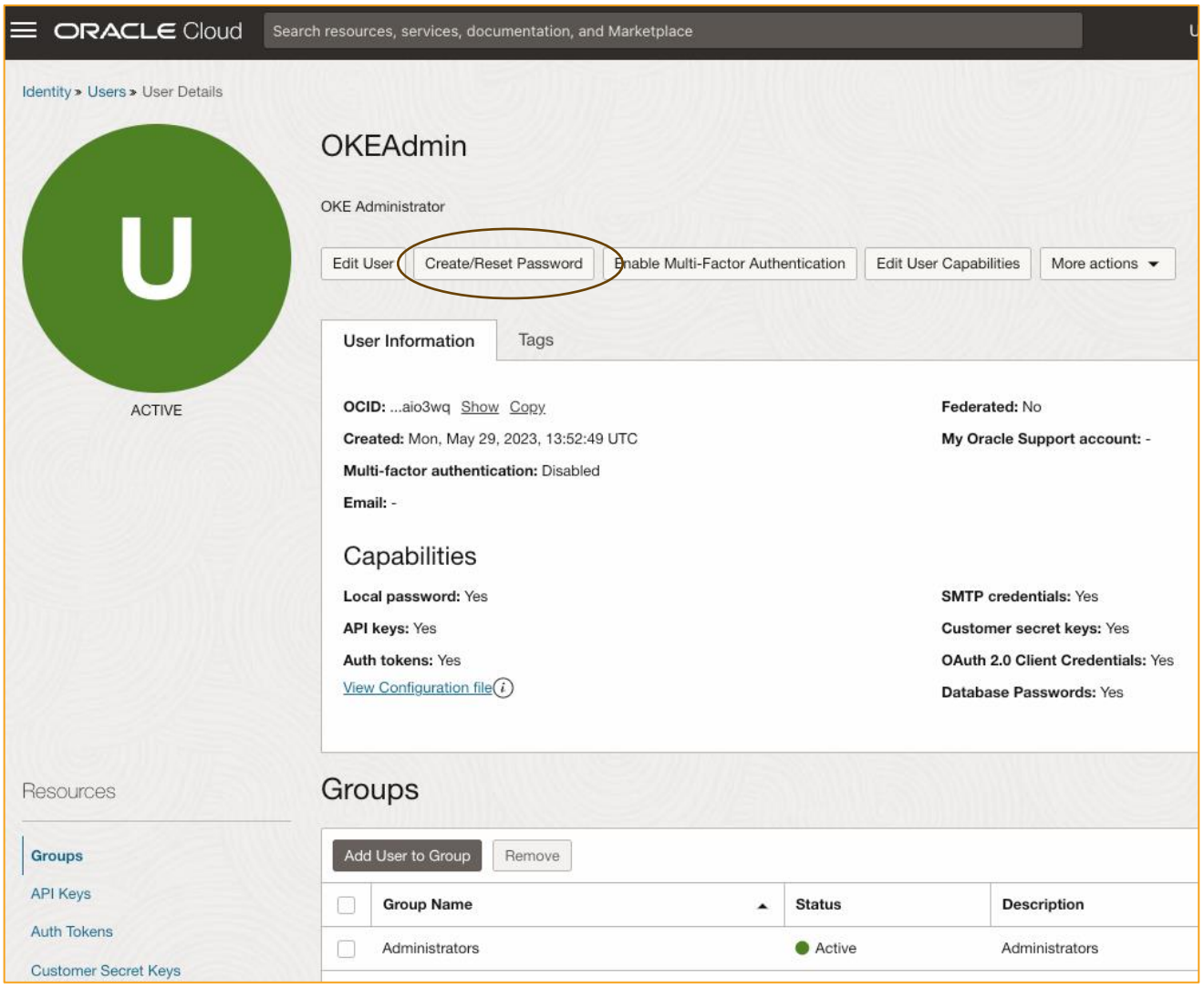
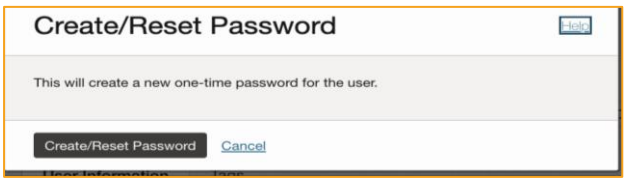


Figure 7-A: Locate Create/Reset Password

Create a new password for the OKEAdmin user. Select the Copy link to save the temporary password to the clipboard. Next, save this password to a temp file. This temporary password will be used to create your own password when accessing the OCI console for the first time as this user.



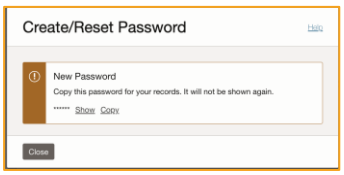


Figure 7-B: Setup a password

Logout of your current OCI Administrator account and then login using (new) OKEAdmin user.

This is important, since the entire setup and deployment will be done using this admin (e.g. OKEAdmin account). When logging back in, choose the Oracle Cloud Infrastructure Direct Sign-In option. For the first time use the temporary password to reset the password of your choice.

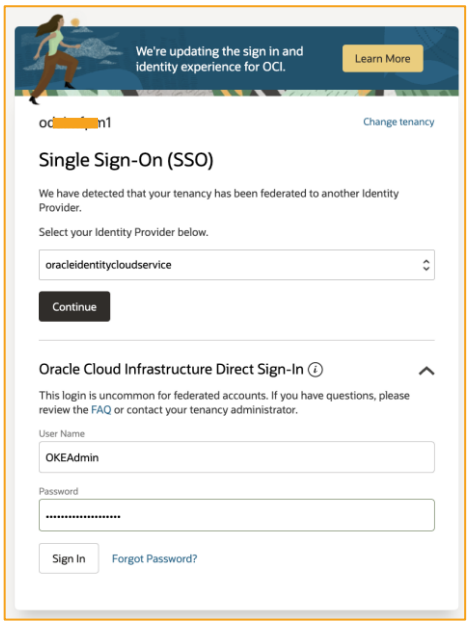


Figure 8: Login as OKAdmin user through the OCI Direct Sign In option

2.3 Create an Auth Token for the OKEAdmin account.

Once logged in as the OKEAdmin, locate the User Details for this user. Under the Resources section, select **Auth Token**, then select the **Generate Token**. In the Generate Token dialog, select the **Generate Token** and once the token is created, copy and store the token (string) in a secure place.

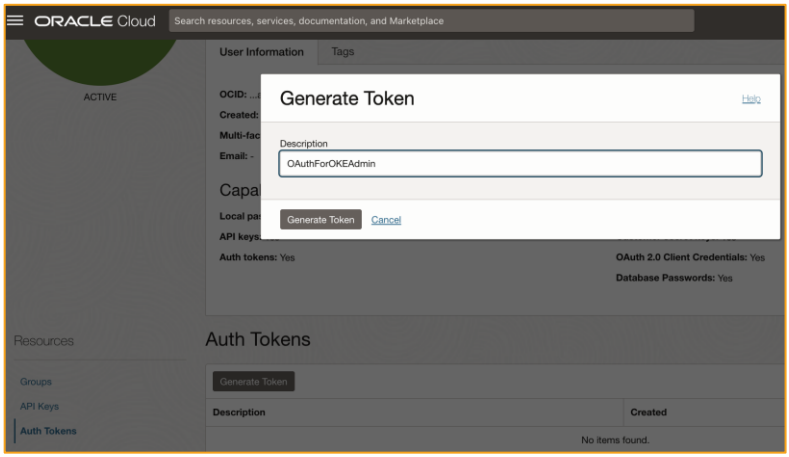


Figure 9: Generate Auth token

This Auth token us used to login into the OCI registry to deploy APIs to the registry.

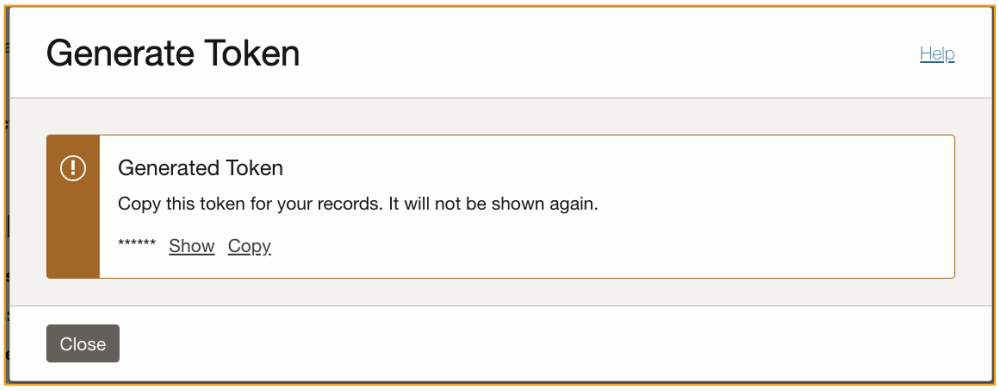


Figure 10: Copy and store the token in a secure place.

2.4 Create an API Key for the OKEAdmin account.

Under the Resources section, select **API Keys**, then select **Add API Key** option.

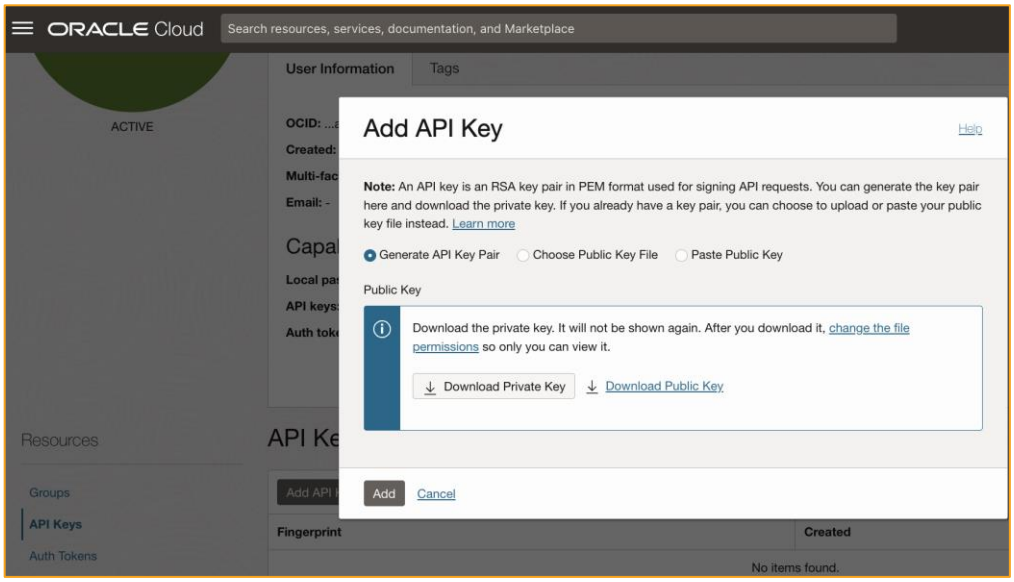


Figure 11: Add API Key

In the Add API Key dialog, select **Download Private Key**. Rename the key, for example, `okeadmin-priv.pem` and store this securely. Next, select the **Add** button. In the Configuration File Preview dialog, select the **Copy** button. Store the content into a secure place (e.g. config.txt on your local file system).

The API key is the primary authentication mechanism for OCI to create the necessary resources through the terraform script.

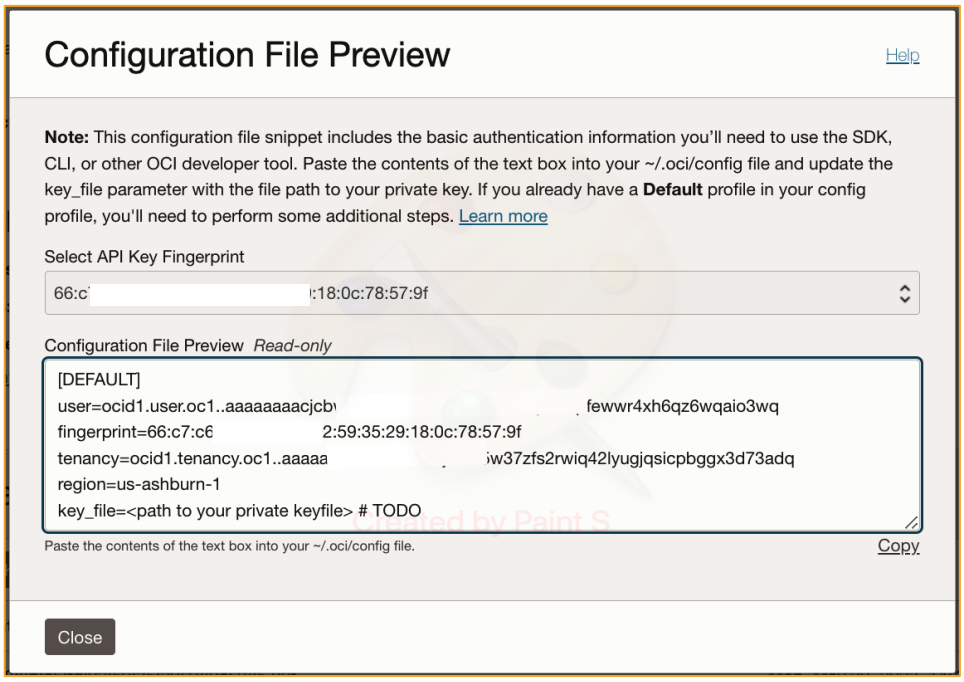


Figure 12: Capture configuration details

This article will continue to use the OKEAdmin account to configure the cluster. Since OKEAdmin user is part of the Administrator's group, this user has all of the necessary permissions to logon to the cloud shell, install the OKE cluster, access to the container registry (OCI Registry), access to the object storage and the access to the various service logs.

You can see that there is a default tenancy admin policy at the root compartment level named Tenant Admin Policy. You can find this under Identity > Policy [Root Compartment]

- cloud shell – This enables the OKEAdmin to open a cloud shell and run the terraform scripts and other kubectl and docker commands.
- OCI Registry (deploy images) – The terraform scripts deploy certain images to this OCI registry like the Authorizer an Vault client.
- OKE cluster, - OKEAdmin will run various OCI (cluster) commands once the cluster is built through the terraform scripts.
- Object Storage – Logs can be enabled at various levels, where these logs are stored in Object Storage.

2.5 Configure the Cloud Shell for docker-compose and download necessary terraform scripts

Open the Cloud Shell. When the shell is ready the Cloud Shell's system prompt will appear.

We will also use the OCI Code Editor later.

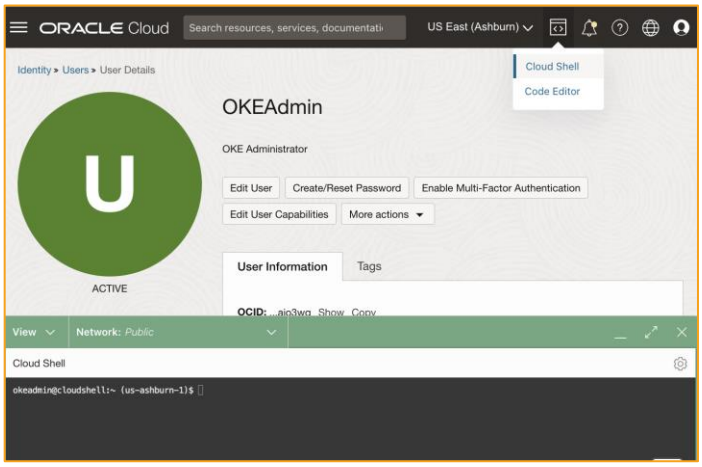


Figure 14: Menu for Cloud Shell and Code Editor

2.5.1 Upload Files

Download the resources-part1.zip file provided as part of the article and unzip the file to the local file system.

The zip includes among others these 2 files:

- docker-compose
- STACK.zip

Upload these two files (docker-compose & STACK.zip) to the Cloud Shell.

By default, the files are uploaded to the home directory of the logged in user, for example: /home/okeadmin. Note that while the user account is OKEAdmin the directory is lowercase okeadmin

Next, upload the private key file `okeadmin-priv.pem`, which was downloaded previously, to the Cloud Shell.

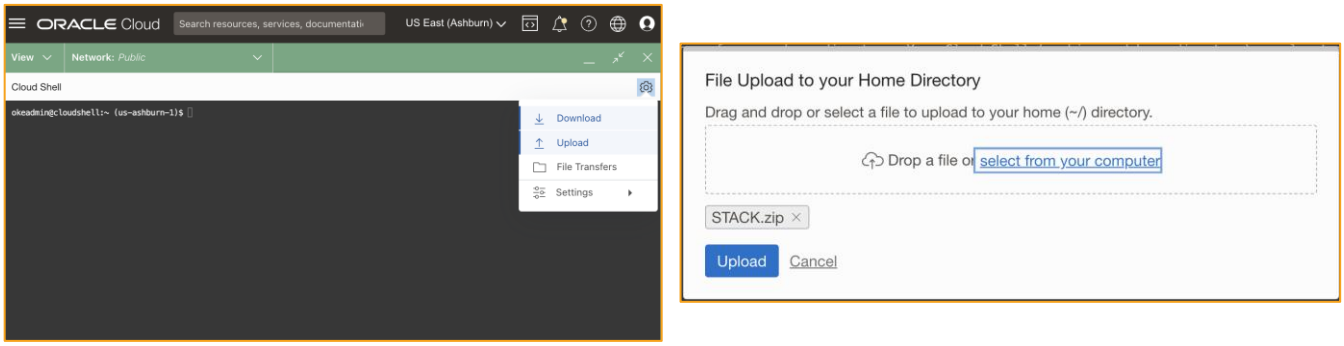


Figure 15: Upload files

2.5.2 Configure the Cloud Shell environment directories

This section guides you to configure the cloud shell with the necessary settings so you can run the terraform scripts successfully.

For each of the steps below, ensure that you are in the home directory (e.g. `/home/okeadmin`). This can be done by simply running the `cd` command.

Step 1:

In the Cloud Shell, run the following commands one after the other

```
cd
mkdir OKE
mv STACK.zip OKE
cd OKE
unzip STACK.zip
```

These commands creates a new directory named **OKE** and moves the **STACK.zip** file into this new directory., changes the working directory to OKE and then unzips the STACK.zip file.

Step 2:

Next, run the following commands.

```
cd
mkdir bin
mv docker-compose bin
chmod 700 ~/bin/docker-compose
```

These commands creates a new directory bin in the home directory i.e. `/home/okeadmin`. It then moves the `docker-compose` file to that directory and finally changes the permission on the file.

Step 3:

Ensure that you are in the home directory. This can be done by simply running the `cd` command. Run the following commands

```
cd
```

```
mkdir .oci
```

```
mv okeadmin-priv.pem ./oci
```

This creates a new directory `.oci` (the **.** is required) and moves the `okeadmin-priv.pem` file to this .oci directory. This private key provides the necessary permissions to allow the terraform scripts to sign-in when creating the OKE based resources.

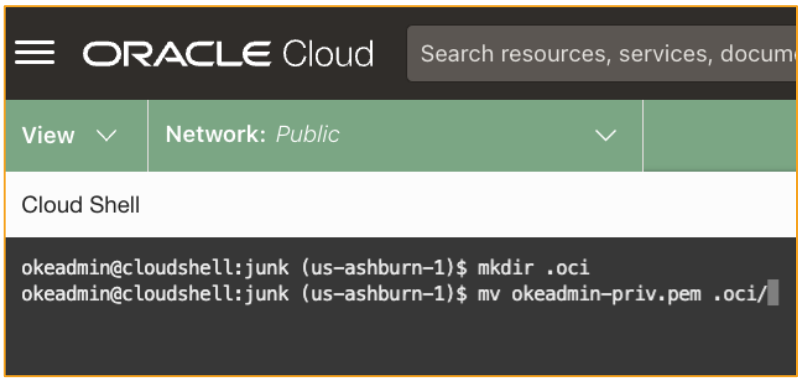


Figure 16: Store private key and note location

Step 4:

Update the .bashrc file and add the docker-compose to the Cloud Shell's path. You can use either the vi editor included in the Cloud Shell or the Cloud Editor.

To edit using the vi editor:

In the Cloud Shell, open the .bashrc file using the following command

```
vi .bashrc
```

Add the following line at the bottom of the .bashrc file, save and exit using **:wq** command.

```
export PATH=$PATH:~/bin
```

Figure 17-1: Configure PATH

*If you have already used the vi editor to make change to the .bashrc file, skip the Cloud Editor section and directly go to "**Ensure docker-compose in the path".***

To edit using Cloud Editor:

Open the Code Editor from the OCI console. By default hidden files are not shown in the code editor. To configure this click on File menu in the code editor and click on Preferences, select Open Preferences option as shown in figure below.

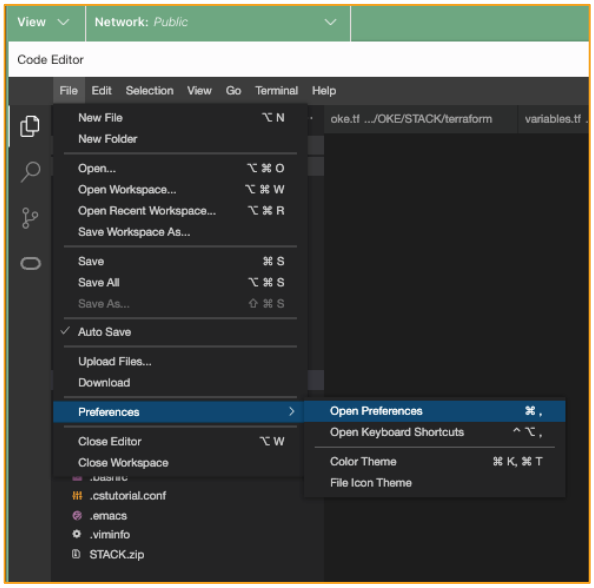


Figure 17-2: Open Code Editor Preferences

In Preferences screen expand the Text Editor option on the left and select Files as shown below. Select Edit in `settings.json` located on the right pane as shown below.

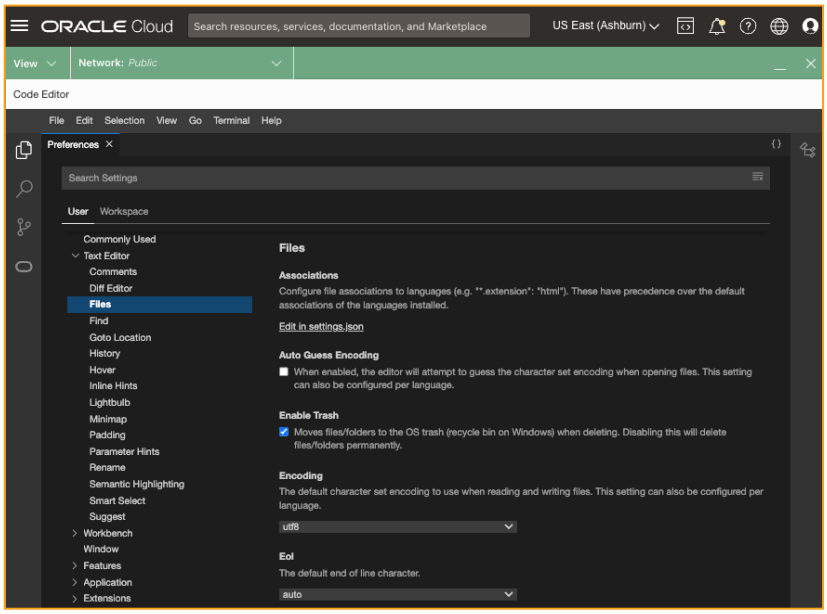


Figure 17-3: Text Editor → Files → Association → Edit in settings.json

This will open the `settings.json` configuration file in the Code Editor. Update the `settings.json`. Locate the `files.exclude` code block. The default value is `true`. Update this value to false and in the Code Editor main menu, select on File→Save option to save your changes.

If the files.exclude code block is not present, create the code block as show below

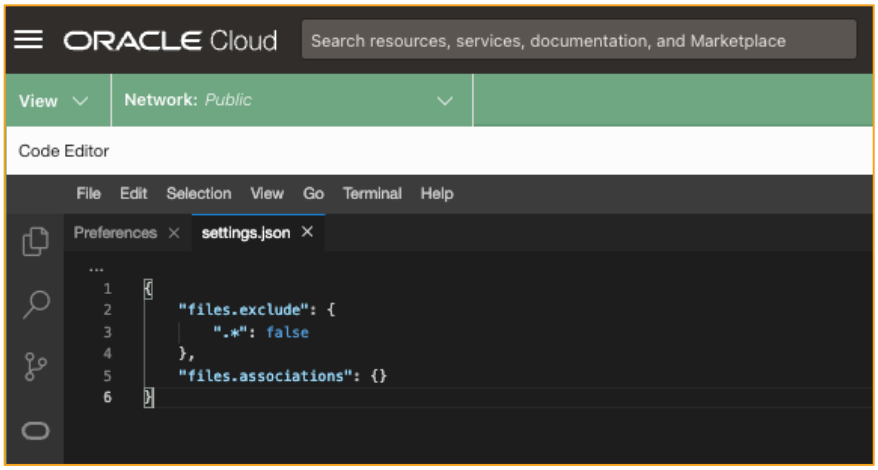


Figure 17-4: Locate `files.exclude`

In the code editor under the OKEADMIN directory (e.g. /home/okeadmin), locate and select the .bashrc file. In the editor pane to the right, add the following line to the end of the file, Save the file.

```
export PATH=$PATH:/~/bin
```

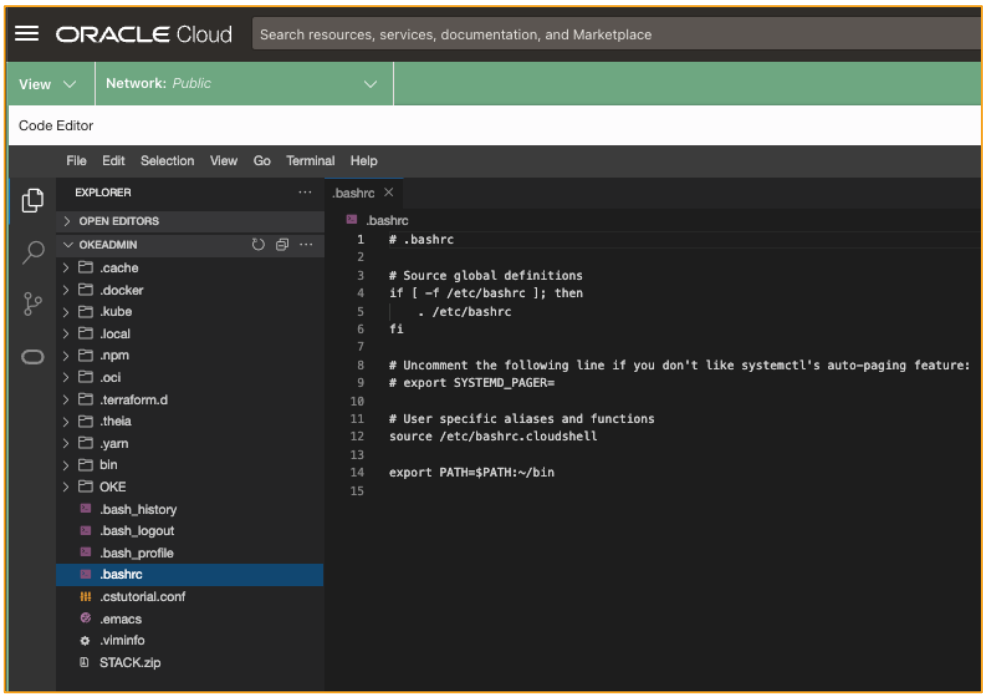


Figure 17-5: `Update .bashrc`

Ensure docker-compose in the path
Once the settings.json and the .bashrc file changes have been made, use the Cloud Shell to check that docker-compose in the path.

In the Cloud Shell, run the following command to make the change effective. Alternatively, close the Cloud Shell and reopen.

```
source .bashrc
```

Run the following command to confirm that docker-compose is on the path.

If you see a permission denied error ensure you run the `chmod 755 ~/bin/docker-compose` command.

```
docker-compose --version
```

Figure 18: Ensure docker-compose utility is in the path

Step 5:

Perform a docker login to ensure the OKEAdmin user can access the OCI registry

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

where `<namespace>` is obtained from the Object Storage of your tenancy.

To find the namespace, from within the OCI console, navigate to Object Storage.

e.g. https://cloud.oracle.com/object-storage/buckets

Select any compartment and then select any of the existing buckets. Note the namespace of the bucket.

If there are no existing buckets, create a DUMMY private bucket. Then use this bucket to capture the namespace.

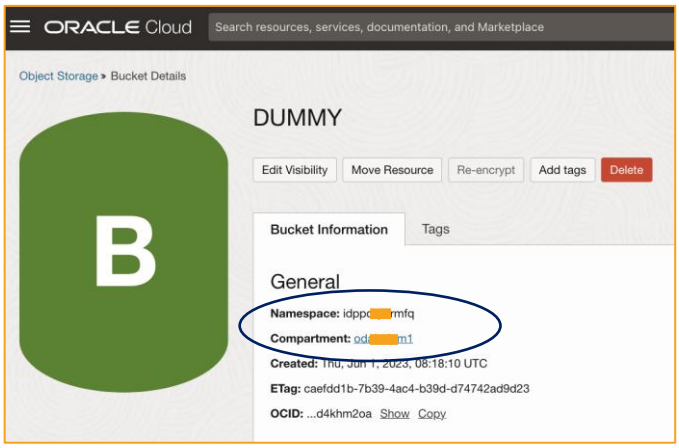


Figure 19: Note the namespace

While running the docker command, when prompted, provide the Auth Token password that was created for OKEAdmin.

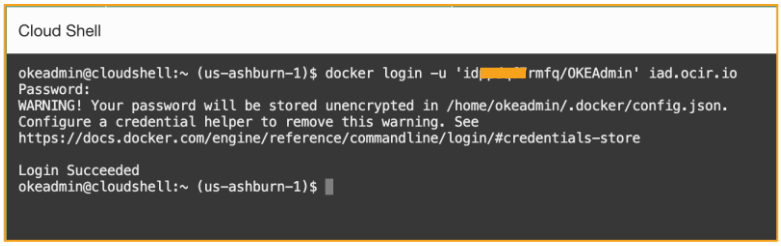


Figure 20: Perform docker login to ensure access to OCI Registry

If docker-compose is not in the path or the docker login failed the terraform scripts would also fail.

Step 6:

Unzip the terraform scripts using the commands shown below.

```
cd ~/OKE
unzip STACK.zip
```

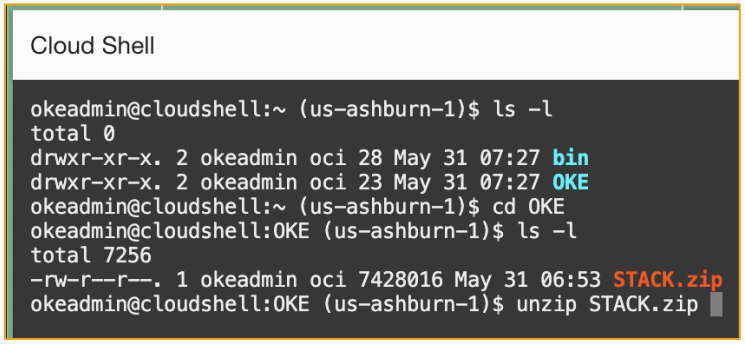


Figure 21: Extract content of STACK.zip

Step 7:

Open the Code Editor from the view menu and select the STACK directory. The terraform scripts are located under the ~/OKE/STACK/terraform directory.

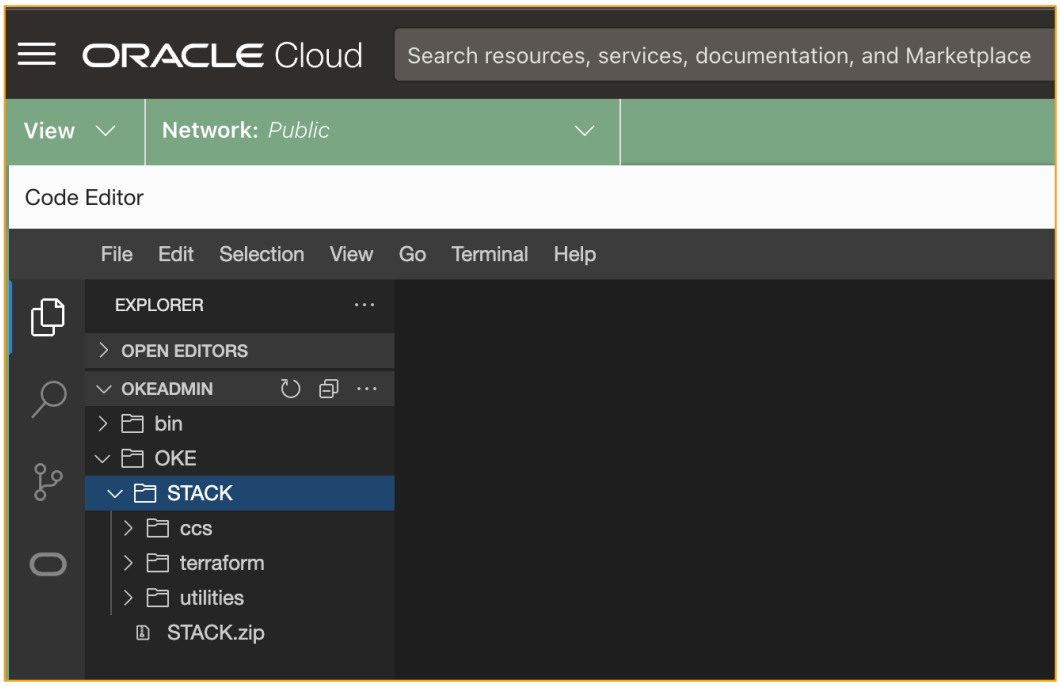


Figure 22: Locate STACK/terraform folder.

2.6 Review the terraform scripts, make necessary customizations or configuration changes

Next step is the use the Code Editor to make changes to a few configuration files. These changes are to provide the script information for the (OCI) region, tenancy, compartment, administrator user (and fingerprint) and a prefix for the artifacts that will be created as part of the setup.

2.6.1 Configure terraform configuration file

Step 1:
Open the terraform.tfvars file from the OKE/STACK/terraform directory in the code editor. Make changes to the configuration entries as indicated below.

Change prefix name as per your requirement (e.g. DEV-CLUSTER). The default is “DEMO-OKE”.
This is on line no. 10.

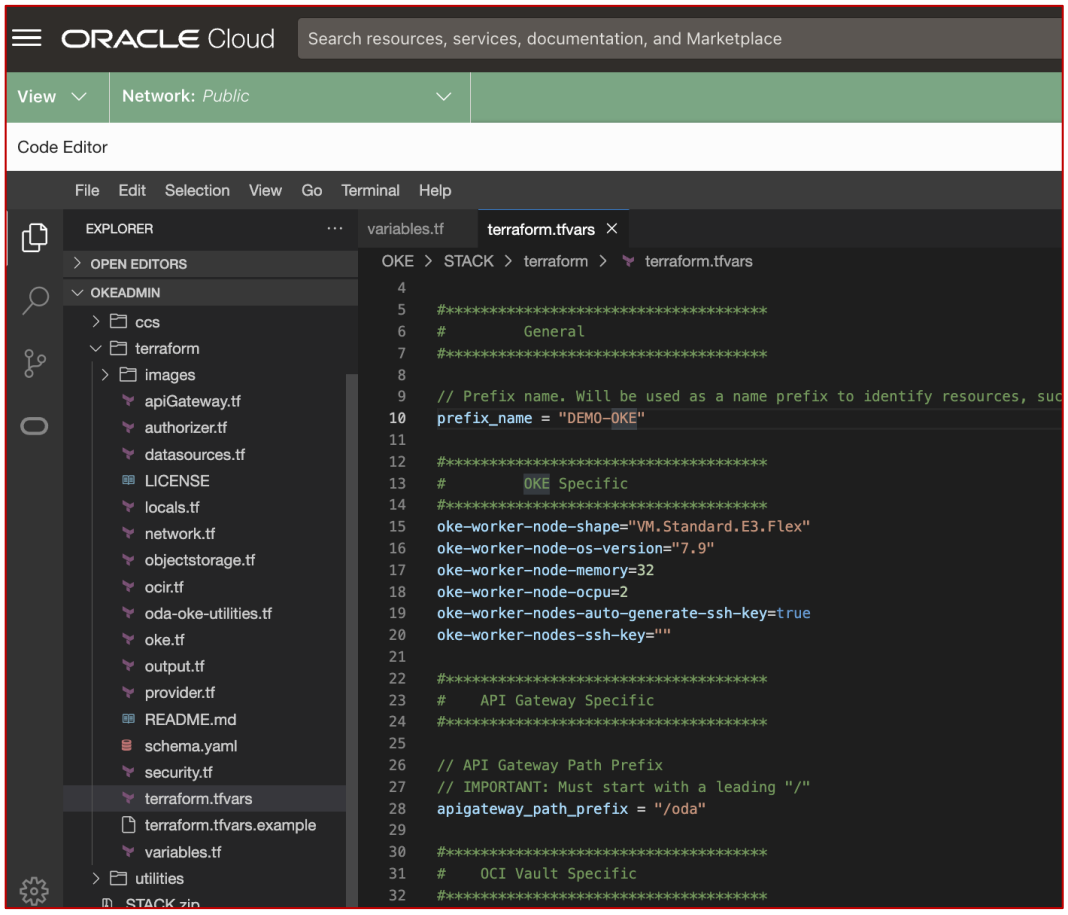


Figure 23: Configure `terraform.tfvars`

Step 2:

Change or confirm gateway path prefix, the default is “/oda”. This is on line no. 28.

Step 3:

- Change or confirm the region property.
- Change OCID for compartment to match the OCI of your compartment
- Provide OCID for OKEAdmin,
- Provide tenancy OCID and the fingerprint. These values were obtained in the Add API Key section ie. Section 2.4. You may have copied and pasted the contents to a (recommended) file named `config.txt`.
- Change the private_key_path to the correct location e.g. `/home/okeadmin/.oci/okeadmin-priv.pem`

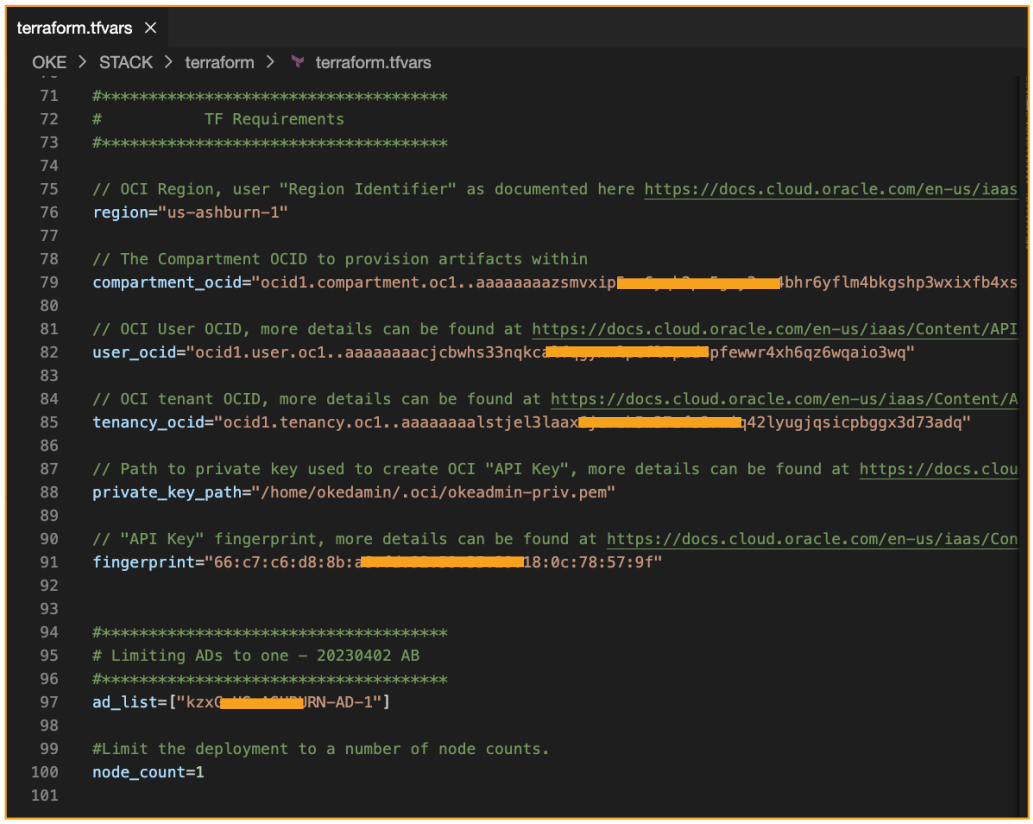


Figure 24: Configure important variables captured during API key creation

- Finally, update the availability domain (AD) name or provide a list of ADs where you would like to create the compute resources. This would be a comma separated list.

Note that every tenancy has a different prefix though the AD name in that region would be same e.g. US-ASHBURN-AD-1.

- Update the node_count variable to change the number of nodes to be created. Default is 1 – variable name is `node_count=1`

2.6.2 Review the changes

Before you move forward confirm that all necessary configuration settings have been completed.

Here is a check list.

1. docker-compose in the path
2. docker login to OCIR is successful
3. All required information has been updated in the `terraform.tfvars` file (not `terraform.tfvars.example`)
4. The private key file is stored in the ~/.oci folder and the `terraform.tfvars` has the correct path

If any of these steps are not correctly configured the configuration would fail. However, you can correct the errors in the configuration and re-run the terraform scripts.

2.7 Run terraform init, plan and fix any issues if reported

Open a new Cloud Shell and go to the `~/OKE/STACK/terraform` directory

```
cd ~/OKE/STACK/terraform
```

Step 1:

Run the following terraform command

```
terraform init
```

If there are any errors, fix them as per the suggestion provided in the message.

Most of the issues are syntax errors, so they are generally easy to fix. Also ensure you are running the command from the `~/OKE/STACK/terraform` directory.

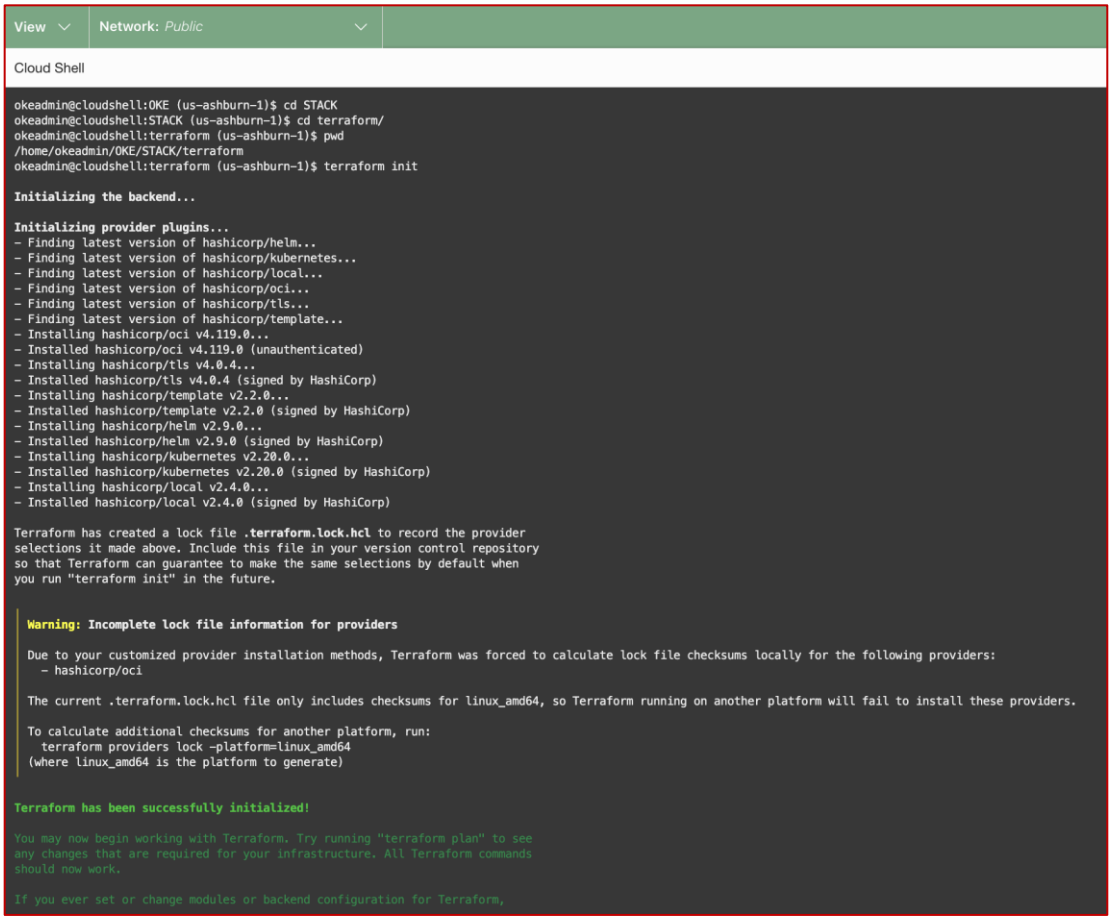


Figure 25: Initialize Terraform

Step 2:

Next, run the terraform plan command

```
terraform plan
```

If there are any errors, fix these errors. If there are no errors, you can proceed to the next step, which will perform the installation of the OKE cluster. Ensure you are running the command from the `~/OKE/STACK/terraform` directory.

2.8 Run terraform apply to deploy the infrastructure

```
terraform apply -auto-approve
```

This process could take anywhere from 20 to 30 minutes.

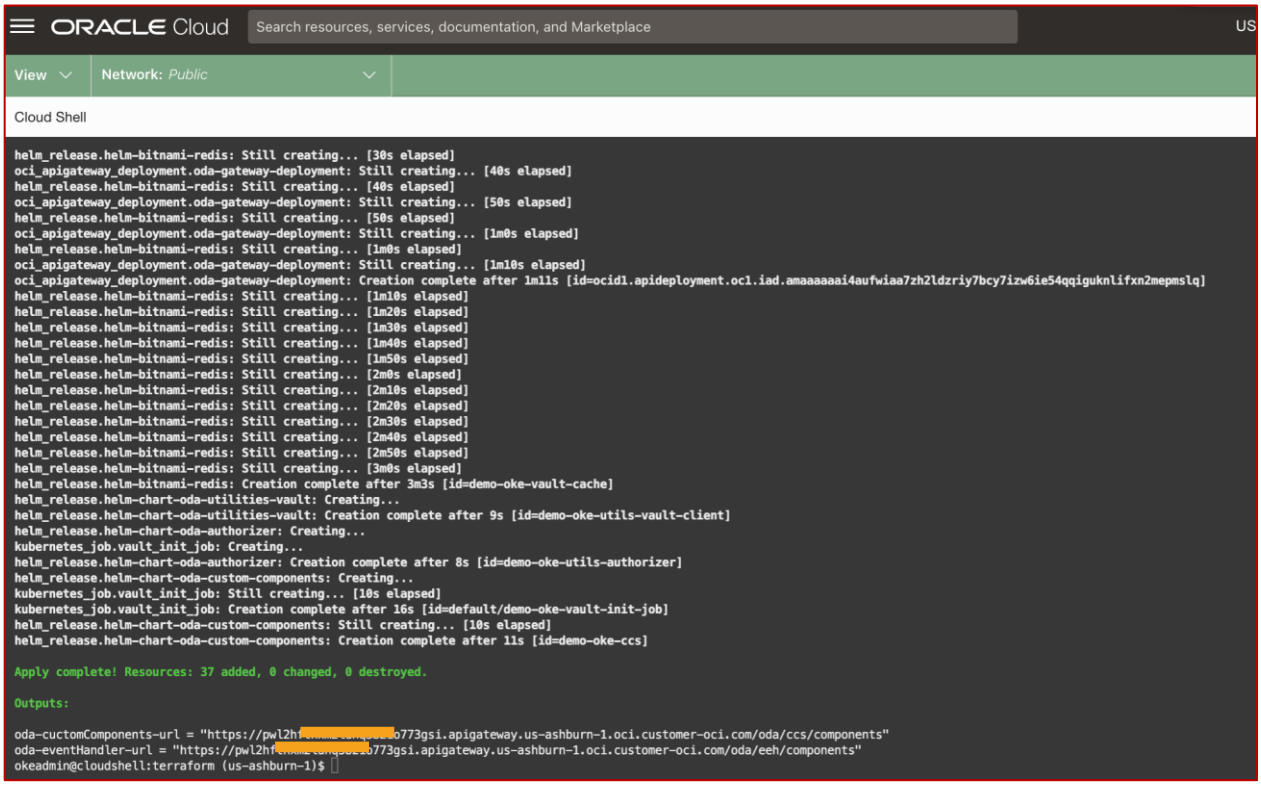


Figure 26: Note OCI API Gateway Cloud Endpoints

Once the script has successfully completed, the API gateway's endpoint URLs for both the sample custom component and the entity handler are listed. Copy these and save in a text file for quick reference.

e.g. https://pwl2hftnxm2lXXXXXXXXXXX3gsi.apigateway.us-ashburn-1.oci.customer-oci.com/oda/ccs/components

If the installation fails, the script will provide explanations on the issues. Fix the issues and then rerun the terraform plan and then the apply commands.

2.9 Review the services created

After successful completion of the terraform scripts verify that all the required services have been created.

To find any of resources, use any of the following methods:

- Use the dedicated URL provided in each section
- Navigate to the resource through the OCI Console's main menu
- Use the Search field that is located in the top of the OCI Console (next to Oracle Cloud logo)

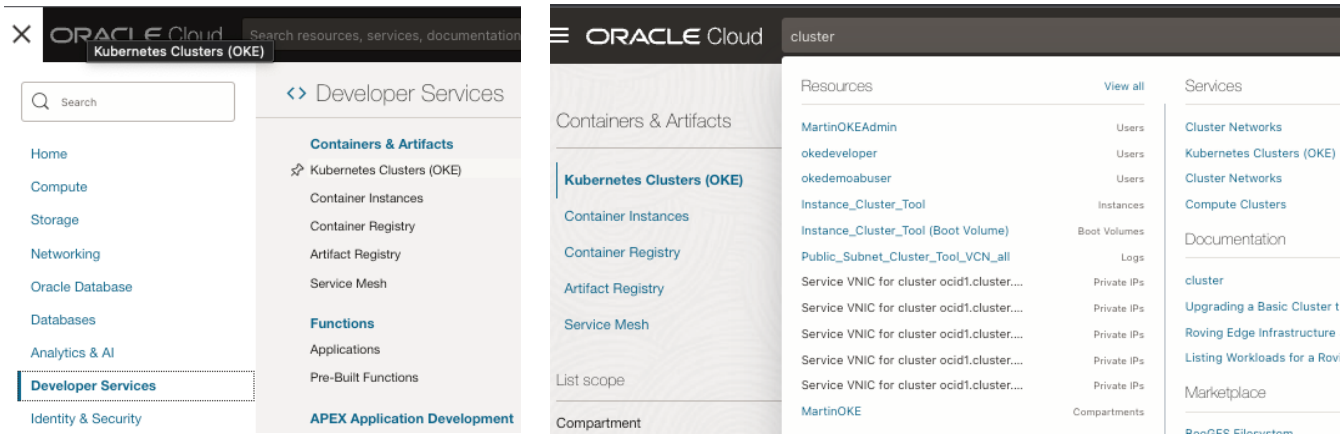


Figure 27-1: Look up for services

2.9.1 Cluster

Review the PKE Cluster that was created.

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

If the cluster is in another region, update the region in the URL to match the region where you deployed your OKE.

For example, the Mumbai region, the URL would be:

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

Change the region to match the region where you deployed your OKE.

e.g. for Mumbai region it would be

https://cloud.oracle.com/containers/clusters?region=ap-mumbai-1

Ensure you select the right compartment from the Compartment drop down to find your cluster.

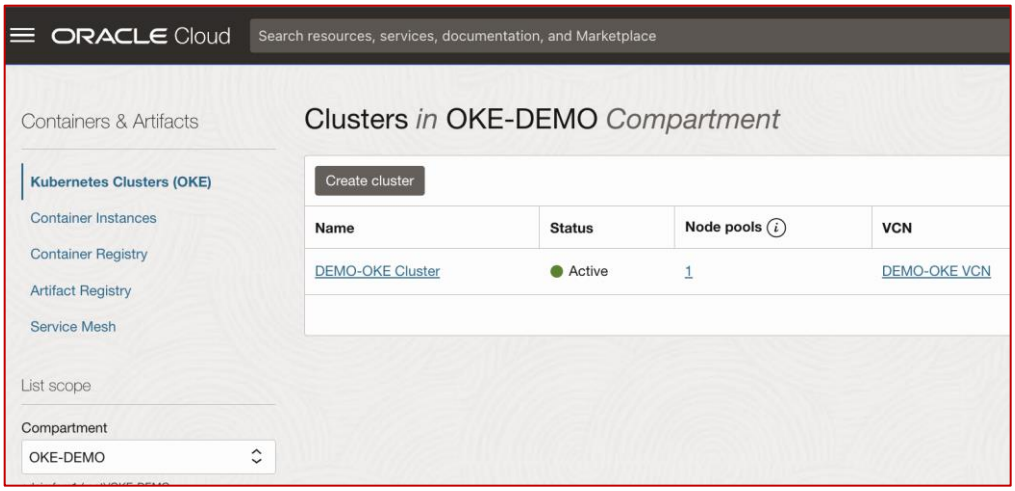


Figure 27-2: Cluster

2.9.2 OCI Registry

Review the OCI Registry (OCIR) that was created. Select the Container Registry menu option to locate your container repositories. Select within the **Repositories and images** drop down to view the resources that has been created.

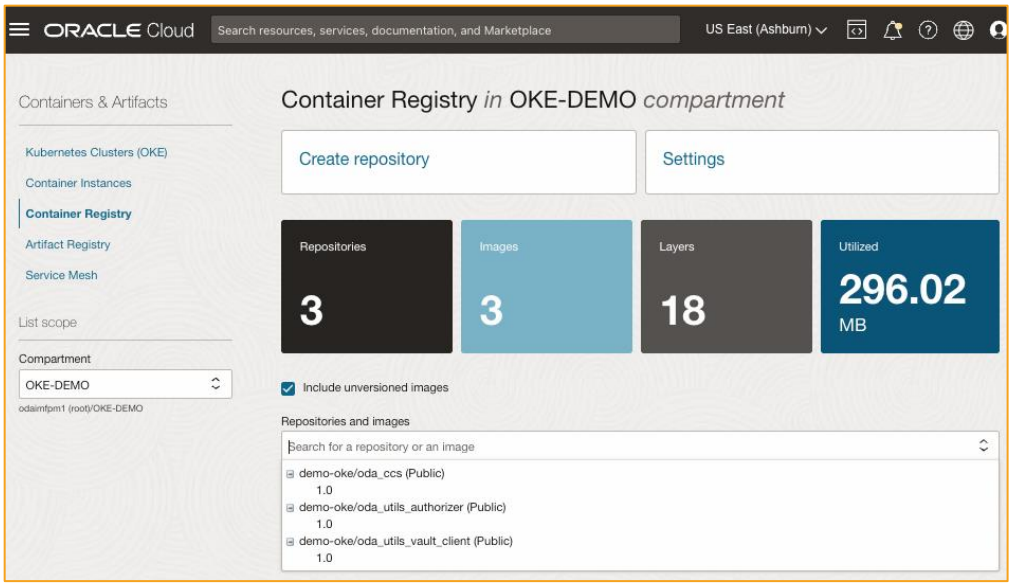


Figure 28: OCIR

Note that the terraform script has created 3 container repositories with the prefix name matching the prefix name given in the terraform.tfvars file.

2.9.3 VCN

Review the Virtual Cloud Network (VCN) that was created.

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

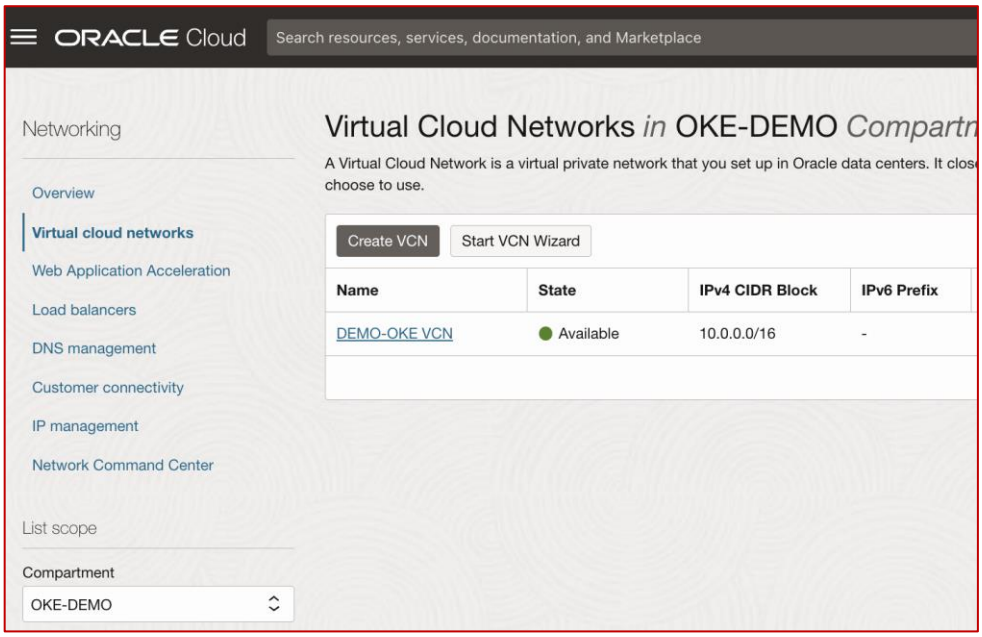


Figure 29: Virtual Cloud Network

2.9.4 Oracle API Gateway Cloud

Review the API Gateway that was created. Use the **Search** option to locate the API Gateway.

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

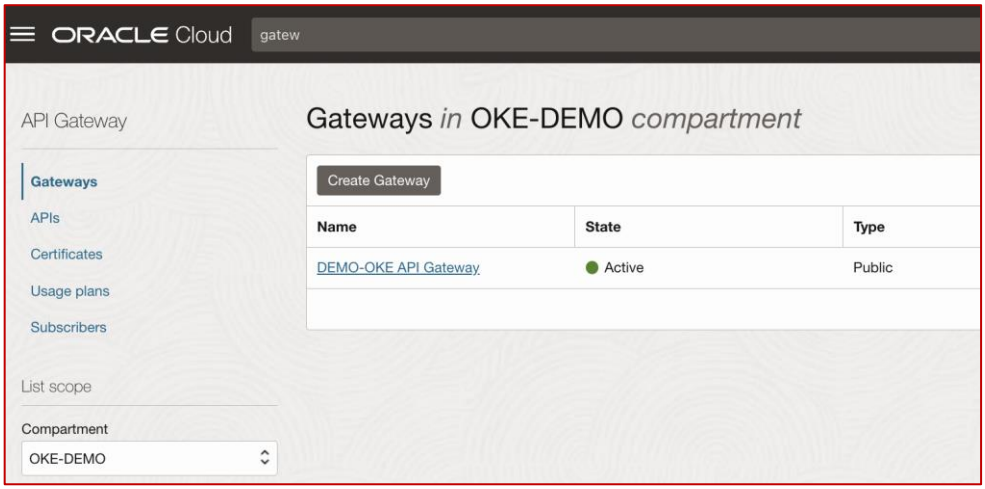


Figure 30: API Gateway Cloud

2.9.5 Load Balancer

Review the Load Balancer that was created. Use the **Search** option to locate the Load Balancer.

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

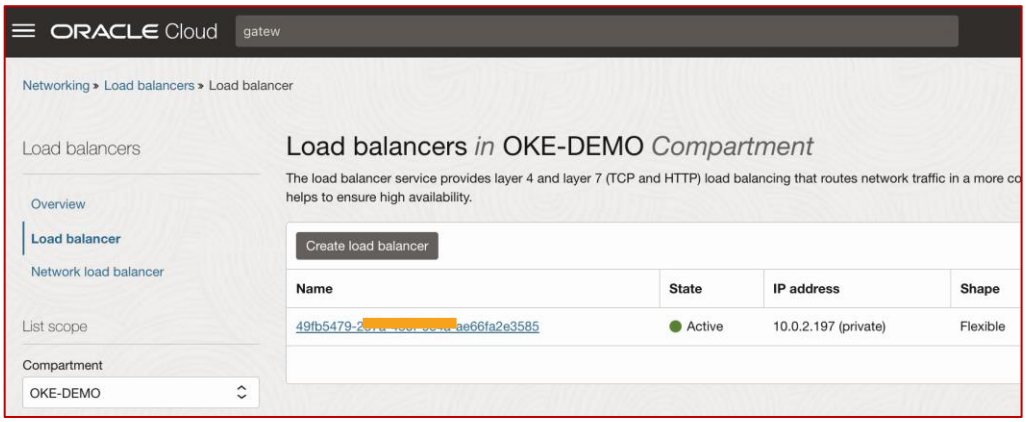


Figure 31: Flexible Load Balancer

2.9.6 OCI Vault

Review the OCI Vault that was created.

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

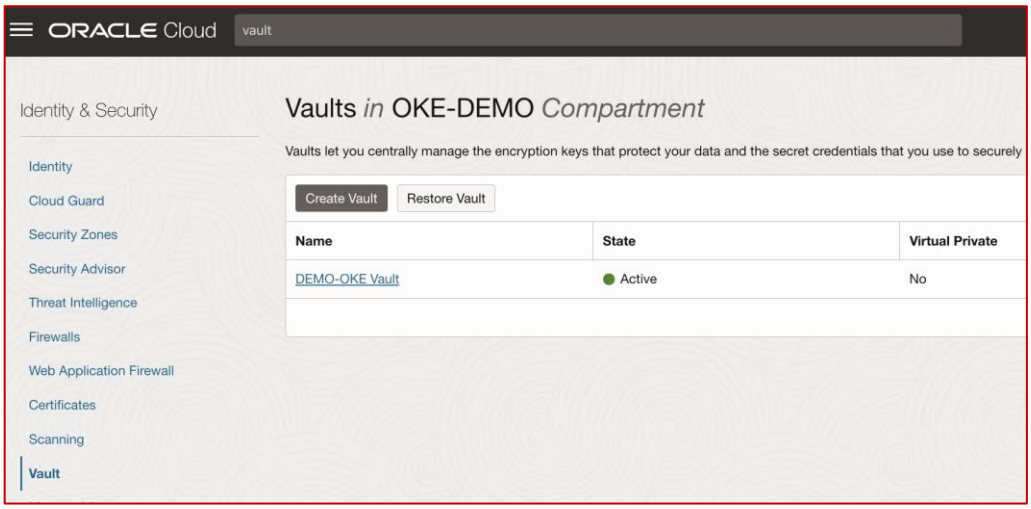


Figure 32-1: OCI Vault

Select the OCI vault link (e.g. DEMO-OKE Vault) to review the Master Encryption Key that was created to sign the keys in the vault.

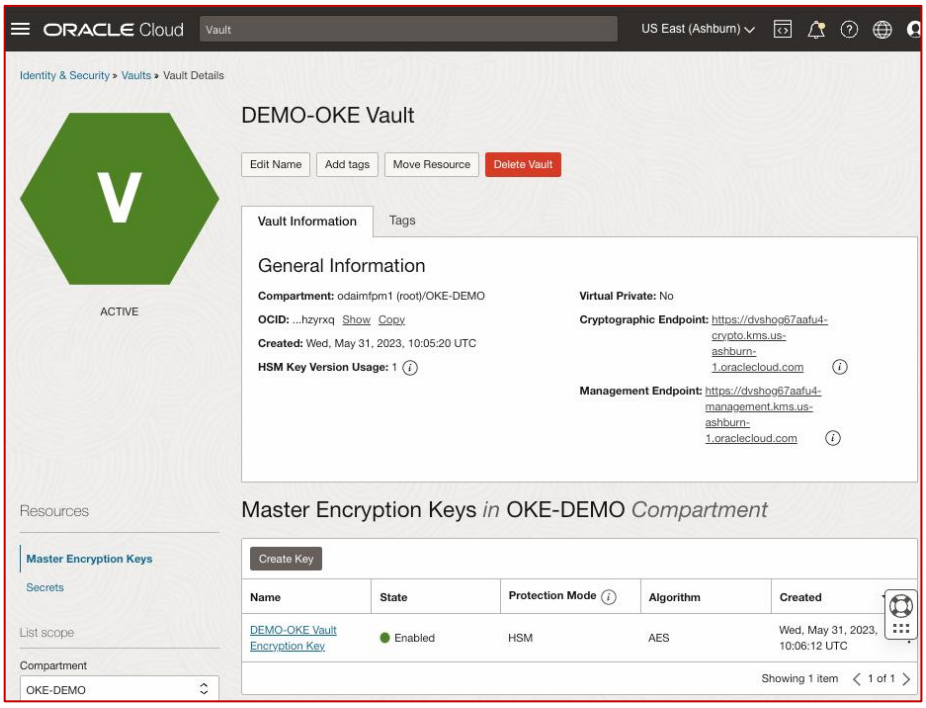


Figure 32-2: OCI Vault – Master Encryption Key

Select the Secret menu option located in the left pane (below the Master Encryption option). There are 2 secrets created with names CC_PASSWORD and CC_USER_NAME. These keys store the credentials to connect to the Oracle API gateway from the Oracle Digital Assistant. Part 2 of the series of this article will explain the usage of these keys in more detail.

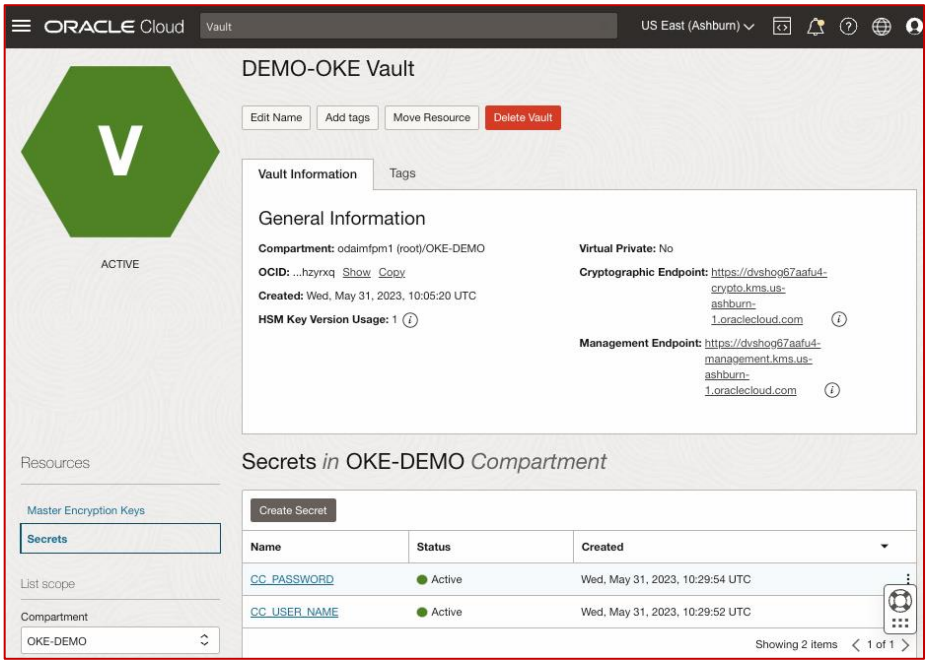


Figure 32-3: OCI Vault – Secrets

2.9.7 Dynamic Group

Review the Dynamic Group that was created.

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

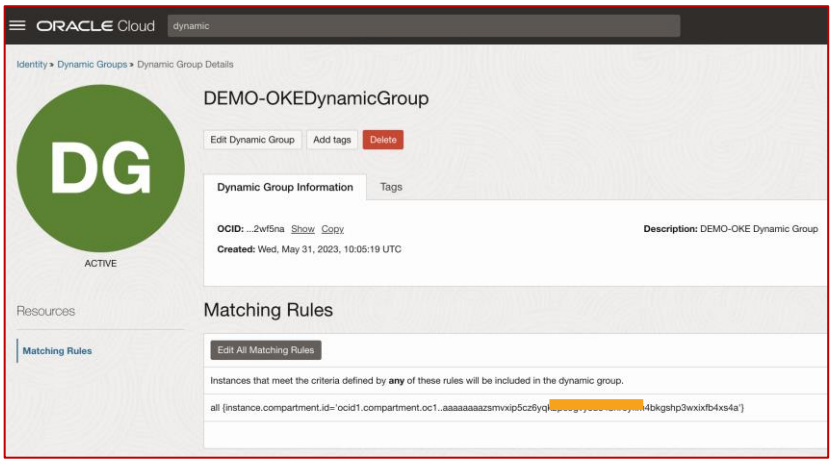


Figure 33: Dynamic Group

2.9.8 Policies

Review the Policies that were created.

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

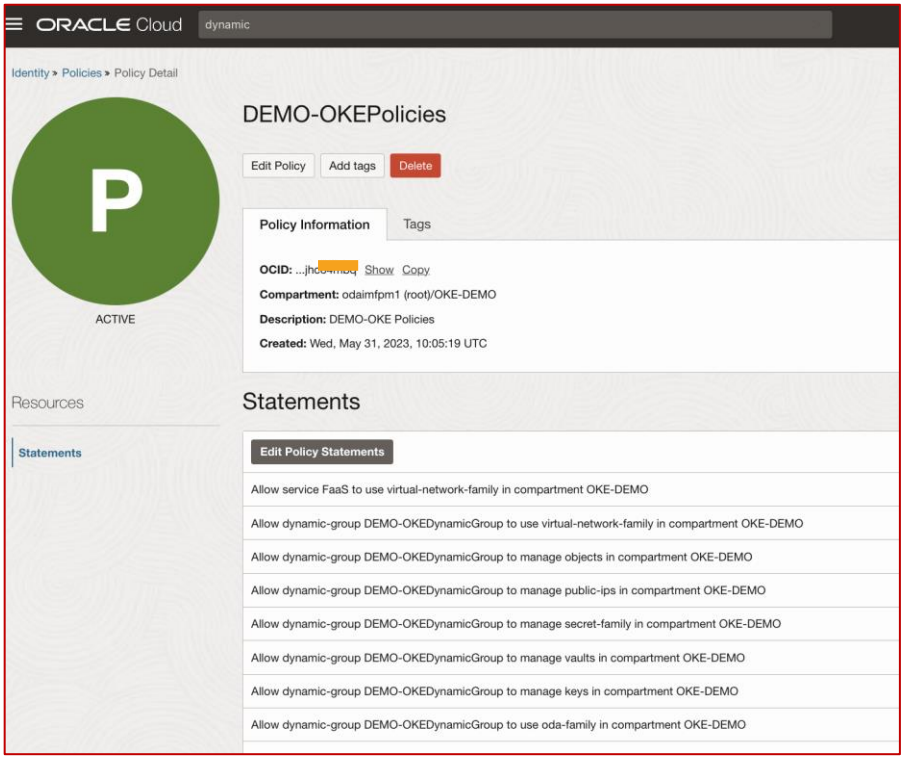


Figure 33: Policies

2.10 Configure the Cloud Shell to access the cluster and to login to OCIR

From the Cloud Console, select Developer Services → Kubernetes Clusters (OKE)

Ensure you have the right compartment selected and verify the cluster.

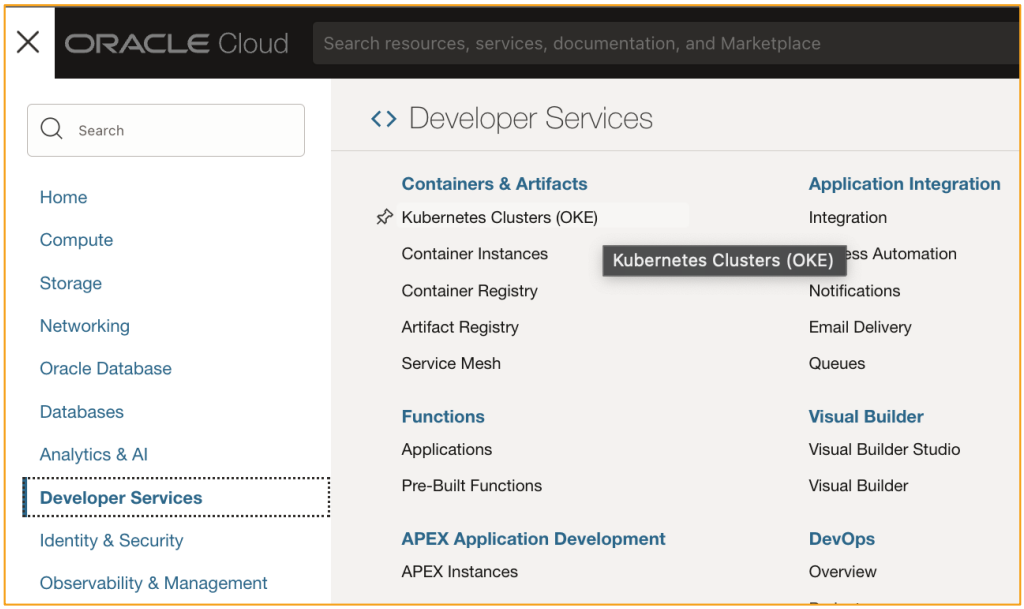


Figure 34: Kubernetes Cluster

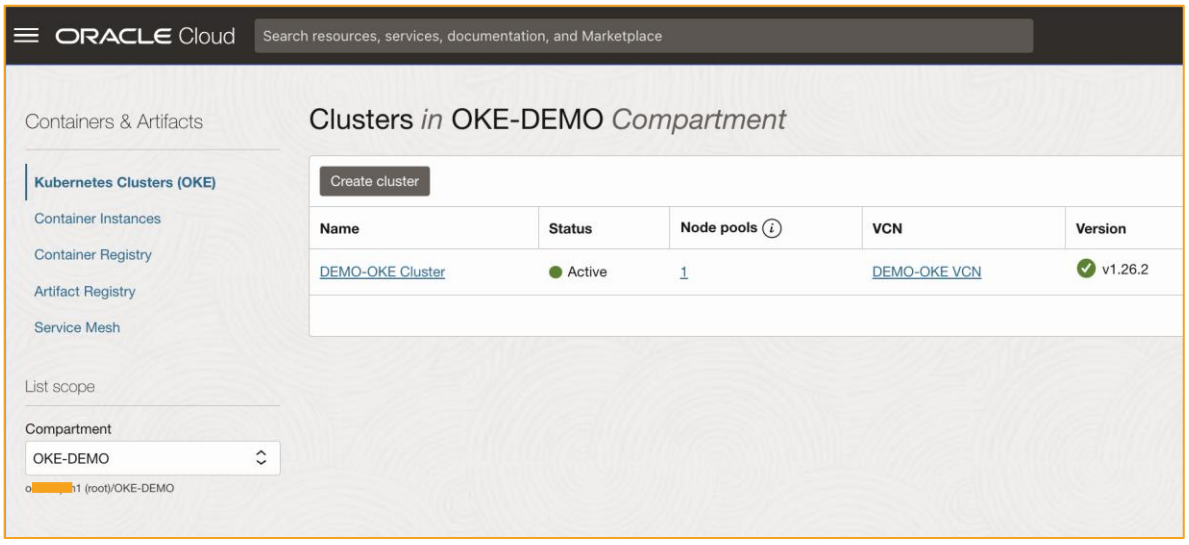


Figure 35: New Cluster Listed

Select the name of the cluster (e.g. DEMO-OKE Cluster) to go to the details screen.

Select the **Access Cluster button.** From the **Access Your Cluster** dialog, select the **Copy** link in step 2. This will copy the displayed command to the clipboard.

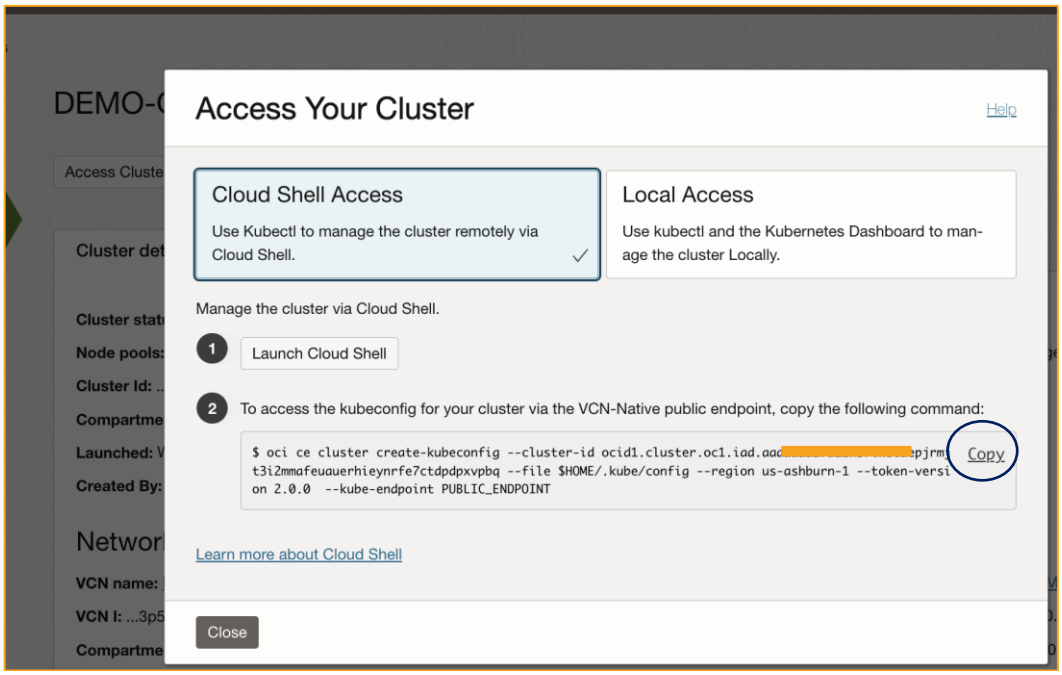


Figure 36: Access Cluster

Open the Cloud Shell and paste the copied command. A new file: `~/.kube/config` will get created in /home/okeadmin directory.

If a config file already exists, this command will update the same file with the cluster details.

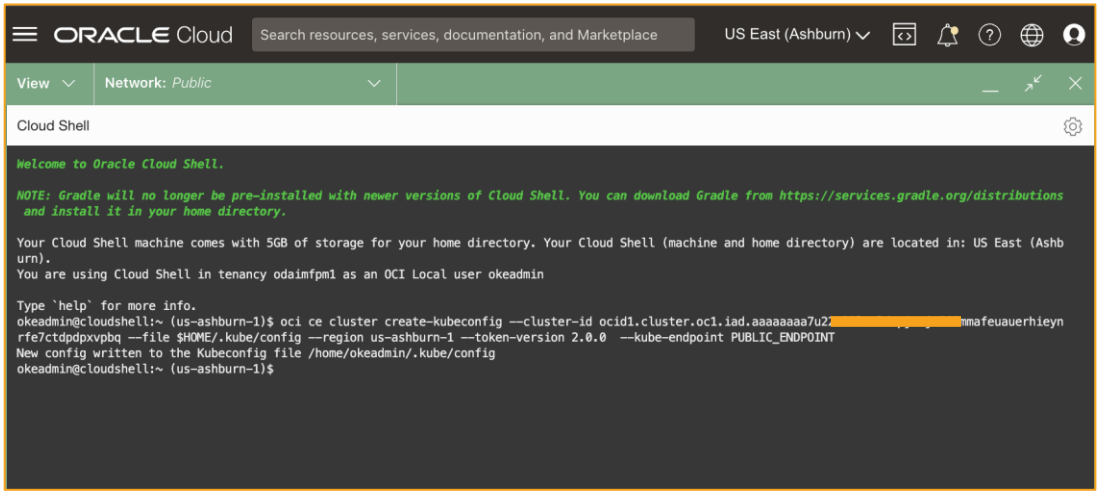


Figure 37: Configure access to cluster

Once the config file has been updated with the new cluster details, here are some useful commands:

List your cluster(s)

```
kubectl config get-contexts
```

Display the current context, which is primarily useful when you have multiple clusters.

```
kubectl config current-context
```

```
okeadmin@cloudshell:~ (us-ashburn-1)$ kubectl config get-contexts
CURRENT   NAME                   CLUSTER                AUTHINFO           NAMESPACE
*         context-ctdpdpxvpbq    cluster-ctdpdpxvpbq    user-ctdpdpxvpbq
okeadmin@cloudshell:~ (us-ashburn-1)$ kubectl config current-context
context-ctdpdpxvpbq
okeadmin@cloudshell:~ (us-ashburn-1)$
```

Figure 38: Check clusters available

The kubectl cheatsheet is a great reference for all the kubectl commands.

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

2.11 Review the Deployments through OCIR and verify authenticator and vault clients pods

Access the OCI Registry from within the OCI console to check the images deployed in your compartment. Ensure that the appropriate compartment is chosen from the dropdown list box as shown below.

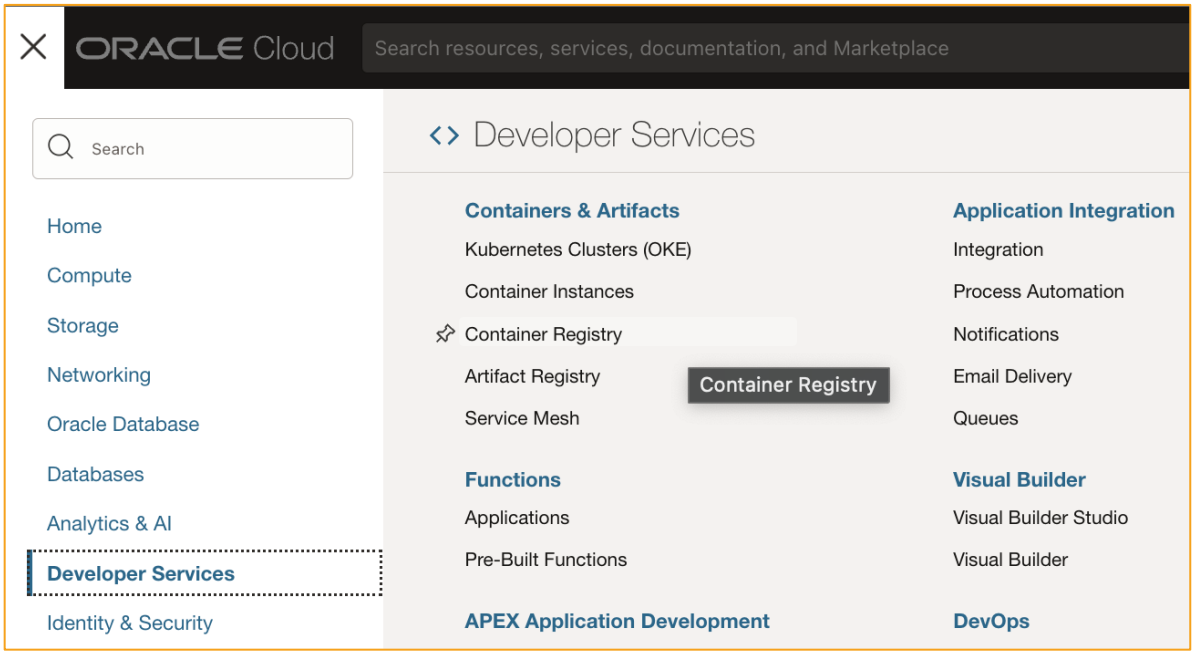


Figure 39: OCI Registry

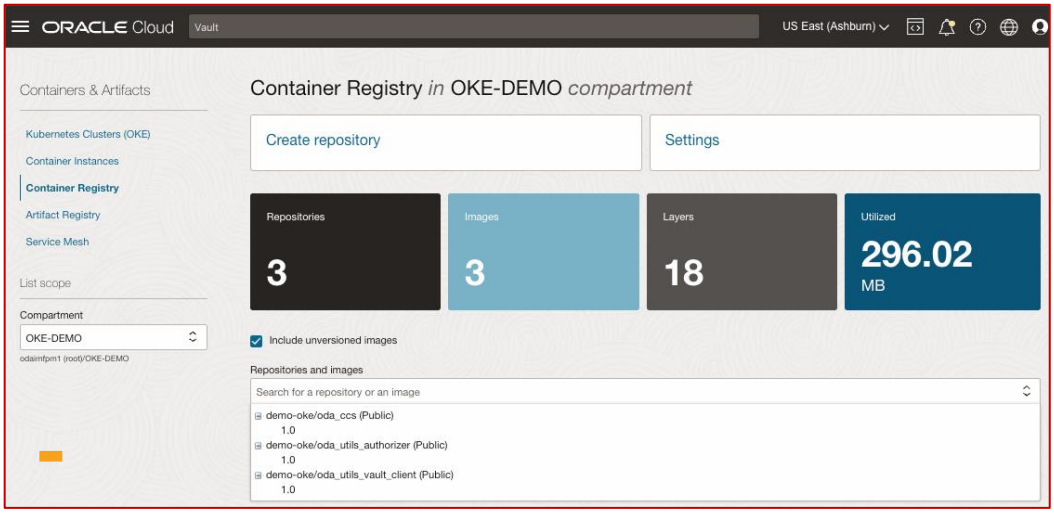


Figure 40: Confirm 3 Images

Ensure there are 3 images deployed to your compartment.

In the Cloud Shell, run the following command to display the list of pods that have been deployed as part of the terraform scripts.

```
kubectl get pods -o wide
```

```
okeadmin@cloudshell:~ (us-ashburn-1)$ kubectl get pods -o wide
NAME                                      READY   STATUS      RESTARTS   AGE   IP             NODE          NOMINATED NODE   READINESS GATES
demo-oke-ccs-7559f8f7f4-gj7fp             1/1     Running     0          18h   10.244.1.9     10.0.1.102    <none>           <none>
demo-oke-services-traefik-865f677b5-blbbj 1/1     Running     0          18h   10.244.1.5     10.0.1.102    <none>           <none>
demo-oke-utils-authorizer-796fbb9487-cbw7c 1/1    Running     0          18h   10.244.0.7     10.0.1.231    <none>           <none>
demo-oke-vault-cache-redis-master-0       1/1     Running     0          18h   10.244.0.132   10.0.1.15     <none>           <none>
demo-oke-vault-cache-redis-replicas-0     1/1     Running     0          18h   10.244.1.6     10.0.1.102    <none>           <none>
demo-oke-vault-cache-redis-replicas-1     1/1     Running     0          18h   10.244.0.6     10.0.1.231    <none>           <none>
demo-oke-vault-cache-redis-replicas-2     1/1     Running     0          18h   10.244.0.133   10.0.1.15     <none>           <none>
demo-oke-vault-client-79c798946b-zsfl9    1/1     Running     0          18h   10.244.1.7     10.0.1.102    <none>           <none>
demo-oke-vault-init-job-wrfzr             0/1     Completed   0          18h   10.244.1.8     10.0.1.102    <none>           <none>
ingress-nginx-controller-6b448794df-p5btd 1/1     Running     0          18h   10.244.0.131   10.0.1.15     <none>           <none>
okeadmin@cloudshell:~ (us-ashburn-1)$
```

Figure 41: Configured pods

All pods will use the prefix (name) that was configured in the terraform.tfvars file (default demo-oke) The prefix also always displays in lowercase even if the prefix was defined using uppercase letters.

Pod Name	Description
ingress-nginx-controller	The Ingress Controller i.e. a controller based on nginx web server setup to expose various APIs from OKE cluster.
traefik	The Traefik routing service that identifies which pod or application the incoming request should be relayed to.
utils-authorizer	The authorizer service that authenticates incoming request to access individual APIs or custom components.
vault-cache-redis	REDIS based in memory cache to access the vault service. This improves the performance by caching the credentials rather than hitting the OCI Vault for every request.
vault-client	Client application that does the authorization against the vault through the vault client service
vault-init-job	The initial pod that is started and completed to create the necessary username and password secrets inside the vault.

ccs	Custom Component Service

2.12 Verify sample custom component API through Oracle API Gateway Cloud

Import the Postman collection provided as part of this article into Postman. Open the OKE-ARTICLE collection and click on OKE Demo CC. Change the API gateway cloud URL to match your API gateway cloud URL that you noted at the end of 2.8 in this article.

https://pwl2hftnxm2lXXXXXXXXXXX3gsi.apigateway.us-ashburn-1.oci.customer-oci.com/oda/ccs/components

Sending the request as is, will give a 401 unauthorized error.

In Postman, configure the request for Basic Authentication and set the Username and Password to: `oda/oda`. These credentials were configured during the deployment. Re-send the request and a 200 response should return with content similar to the following.

Note that you can change both the username and password secrets in the OCI Vault through the Vault Client API installed as part of this setup.

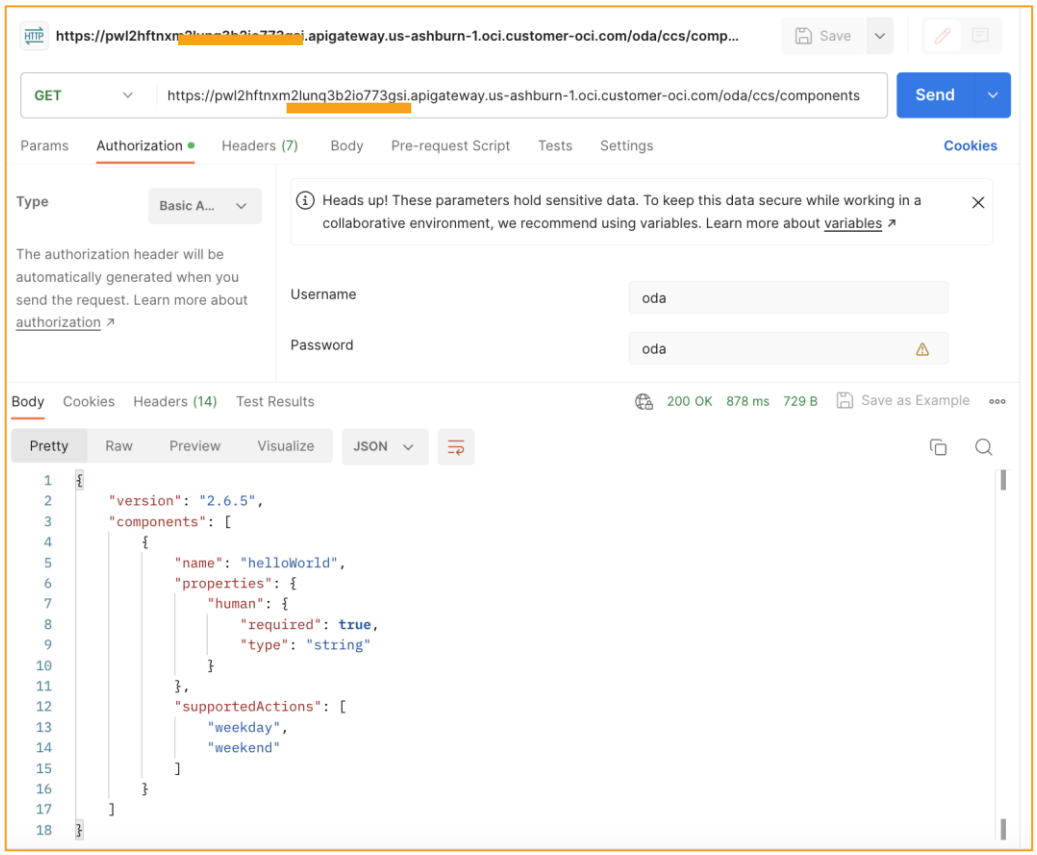


Figure 42: Confirm that API is available securely

This completes the cluster set up and the verification steps.

A 502 error means there is an issue with the setup. There is a complete appendix in this article that can help troubleshoot the issue

Conclusion

Deploying to Oracle Kubernetes Engine opens up a new door to a platform that provides a baseline for Oracle digital assistant to integrate with various services and other services to integrate with Oracle Digital Assistant. This article focuses primarily on how to setup OKE in your OCI environment. The part 2 of the series of this articles shows you how to build and deploy a simple custom component to the OKE cluster. While custom components can be deployed to the cluster you can also use this cluster as an API hosting platform e.g. you can deploy an API to this cluster a backend system e.g. PeopleSoft or CRM can call to generate events to trigger Application Initiated Conversations with MS Teams or Slack through ODA.

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

Design Camp Session - Unleash the power of OCI for ODA - Part 3 - Kubernetes and External Functions for Oracle Digital Assistant Developers

https://videohub.oracle.com/media/Oracle+Digital+Assistant+Design+CampA+Unleash+the+power+of+OCI+for+ODA+-+Part+3+-+Kubernetes+and+External+Functions+for+Oracle+Digital+Assistant+Developers/1_ig9c48jn

.