OCI Setup MLFlow Tracking Server

Referenced Documentation: https://oci-mlflow.readthedocs.io/en/latest/tracking_server.html

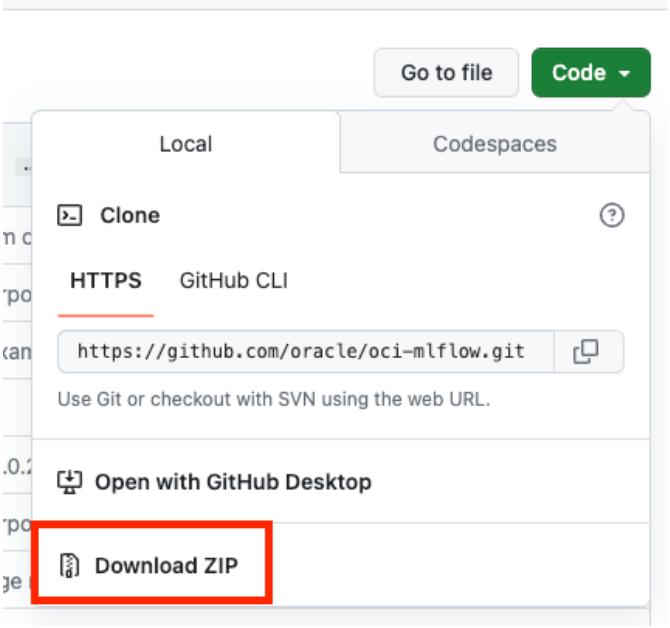
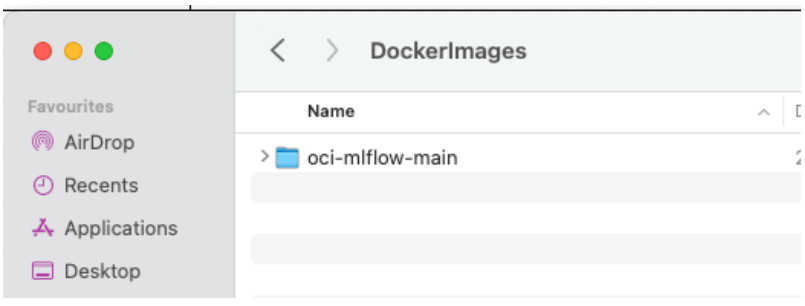
Pre-requisite: A running OCI Data Science Notebook Session within Private Network.

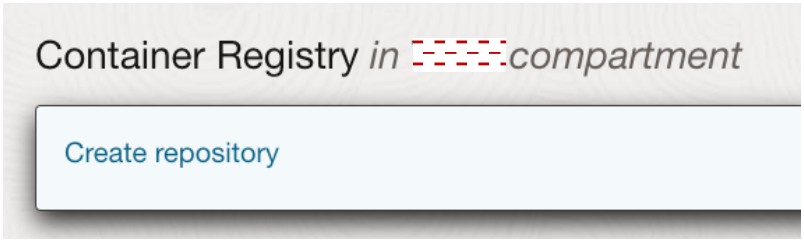
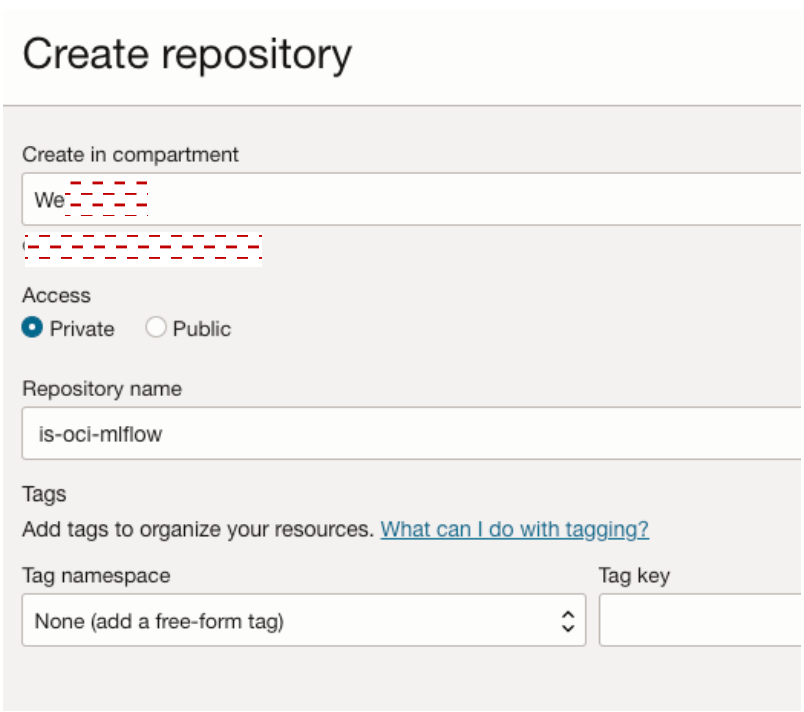
There are two ways to set up the tracker server within OCI:

1. Running with the default options will assume you want to store the artifacts in your local file system and use SQLite for recording experiments data.
2. Configure setup with Object Storage to save artifacts and MySQL Database to save experiments data.

We will attempt both options using the ***oci-mlflow*** container image which we will deploy using the OCI Container Instances.

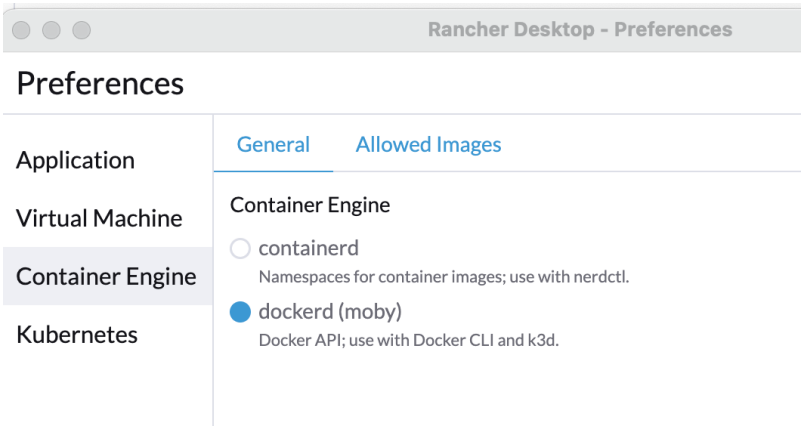
Approach 1: Store Artifacts Locally and Experiments in SQLite

Step	Screenshot
Download the ***oci-mlflow*** docker file from the Oracle GitHub Repository. https://github.com/oracle/oci-mlflow/tree/main	
Unzip the folder into a Directory on your Laptop.	

Before we build our Docker Image, let's create a Container Registry to Store our MLFlow Image within Oracle Container Registry. ***Navigate from OCI Menu > Developer Services > Container Registry.***	
Click ***Create Repository.***	
Select your ***Compartment***. Select access as ***Private***. Give the Repo a ***Name***. Make note of the repo name as we will need this later. ***Click Create***.	

To build our MLFlow Docker Image I will be using Rancher Desktop. Download and install Rancher Desktop. https://rancherdesktop.io/

When you open the application select the container engine as ***dockerd (moby).***



Once Rancher Desktop is up and running, we can build our Docker Image while in the unzipped directory we downloaded from GitHub.

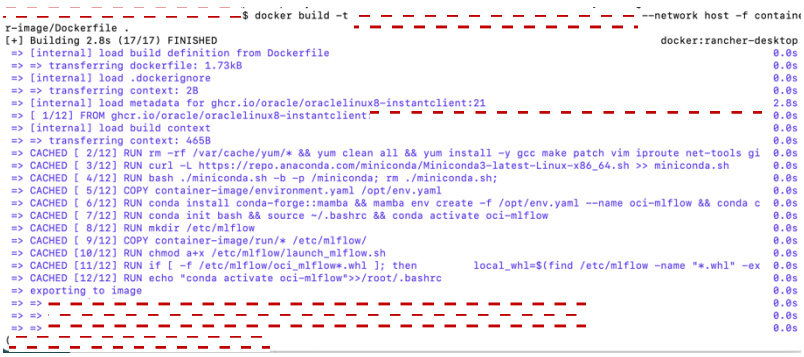
We will build the image using the following format:

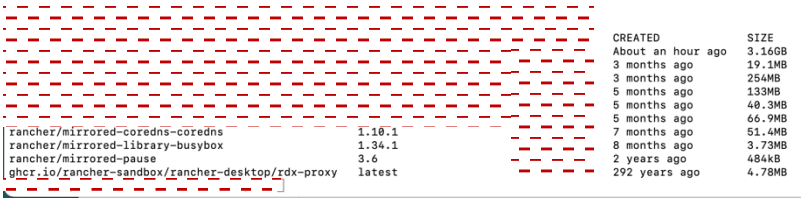
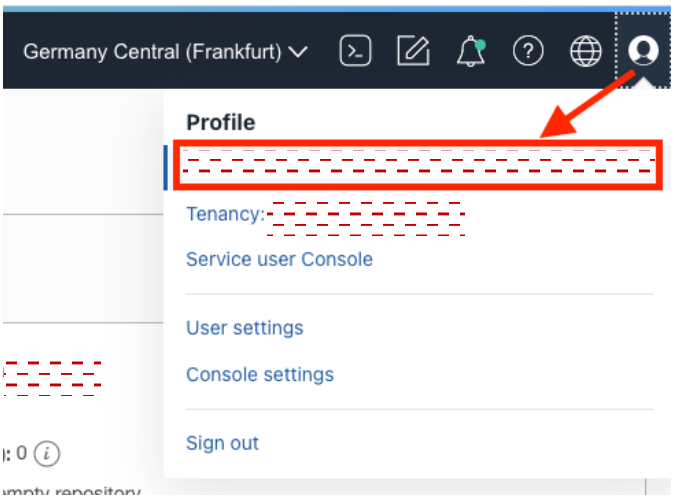
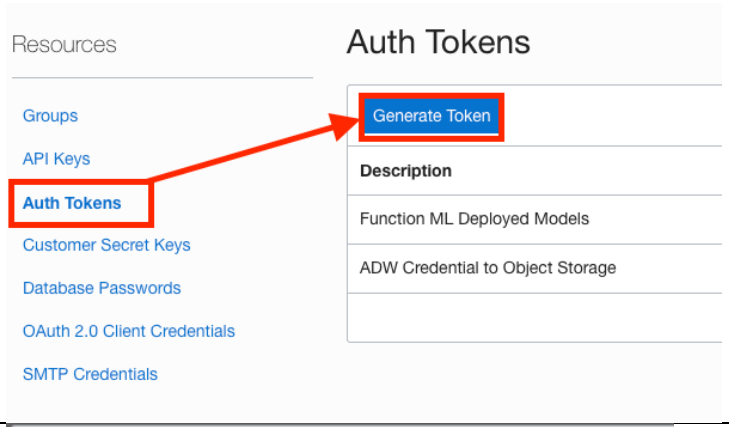
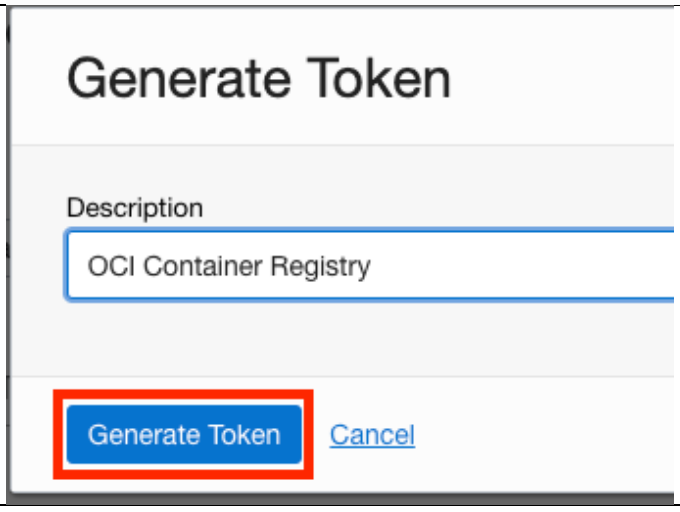
docker build -t {region}.ocir.io/{tenancy-namespace}/{repo-name}:{tag} --network host -f container-image/Dockerfile .

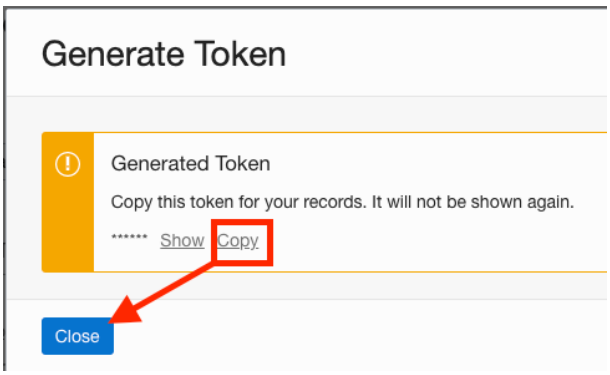
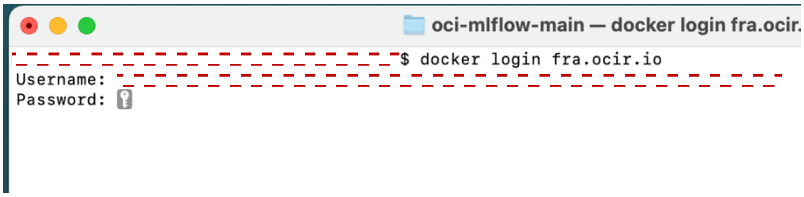
See parameter definitions to the right.

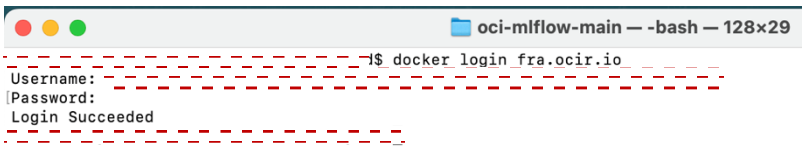
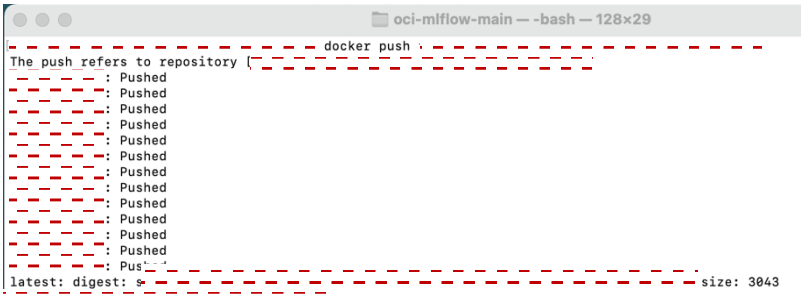
Parameter Definitions:

- ***<region-key>*** is the key for the Container Registry region you're using. For example, ***fra***. See Availability by Region.
- ocir.io is the Container Registry name.
- ***<tenancy-namespace>*** is the auto-generated Object Storage namespace string of the tenancy that owns the repository to which you want to push the image (as shown on the **Tenancy Information** page).
- ***<repo-name>*** is the name of the target repository to which you want to push the image (for example, project01/acme-web-app).
- ***<tag>*** is an image tag you want to give the image in Container Registry (for example, version2.0.test).



Once the Image is built, we can list our images using: ***docker image ls***	
Now we can push our image to the OCI Registry. We then need to Generate an Auth Token for Authentication from our Local Docker to Container Registry. Click on your ***Profile Icon*** then your ***Username***.	
Navigate to ***Auth Tokens*** and Click on ***Generate Token.***	
Give it a ***Description*** and Click on ***Generate Token***.	

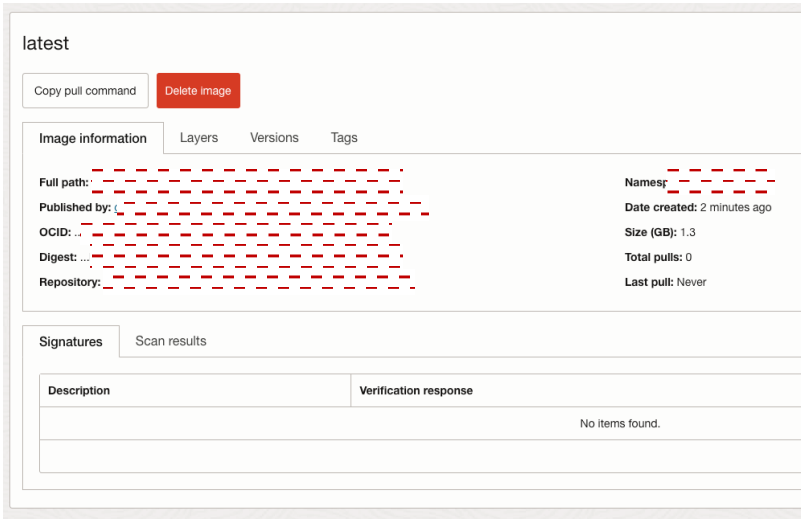
Copy the ***Token*** as it will be needed later to authenticate. You can't view this again.	
Ensure your local Docker installation is running, before doing the following tasks. In a terminal window on the client machine that is running Docker, log into your Container Registry by entering: ***docker login <region-key>.ocir.io*** Where ***<region-key>*** corresponds to the key for the Container Registry Region you're using. For example, Frankfurt has the key ***fra***.	
When prompted for a username, enter your username in the format ***<tenancy-namespace>/<username>*** If your tenancy is federated with Oracle IDCS, use the format ***<tenancy-namespace>/oracleidentitycloudservice/<username>***	

When prompted for a password, ***enter*** the ***Auth Token*** you generated and copied earlier.	
Push the local Docker Image to the OCI Container Registry. ***docker push <repository-name>:<tag>*** If you need a reminder of the above details, you can always run ***docker image ls***	Format of ***<repository-name>*** must be ***<region-key>.ocir.io/<tenancy-namespace>/<repo-name>*** where: ○ ***<region-key>*** is the key for the Container Registry region you're using. For example, fra. See Availability by Region. ○ ocir.io is the Container Registry name. ○ ***<tenancy-namespace>*** is the auto-generated Object Storage namespace string of the tenancy that owns the repository to which you want to push the image (as shown on the **Tenancy Information** page). ○ ***<repo-name>*** is the name of the target repository to which you want to push the image (for example, project01/acme-web-app). ○ ***<tag>*** is an image tag you want to give the image in Container Registry (for example, version2.0.test). 

We can then view our Image in the OCI Container Registry.

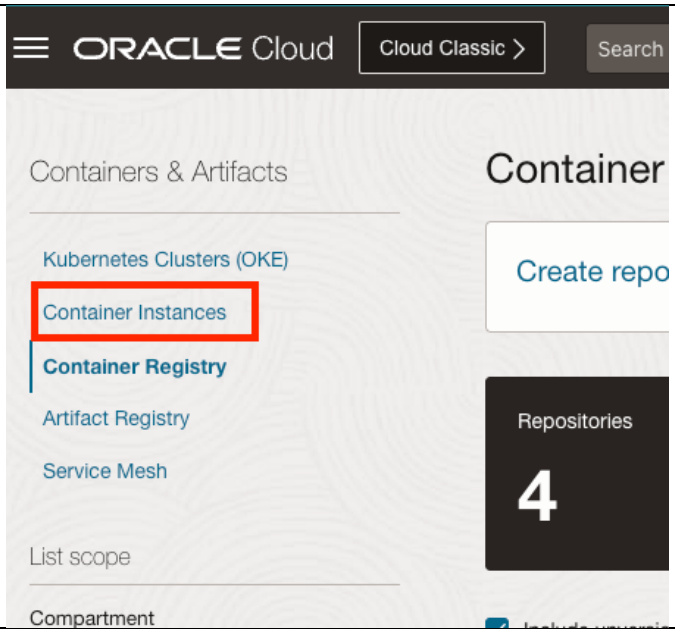
Click on ***OCI Menu > Developer Services > Container Registry***

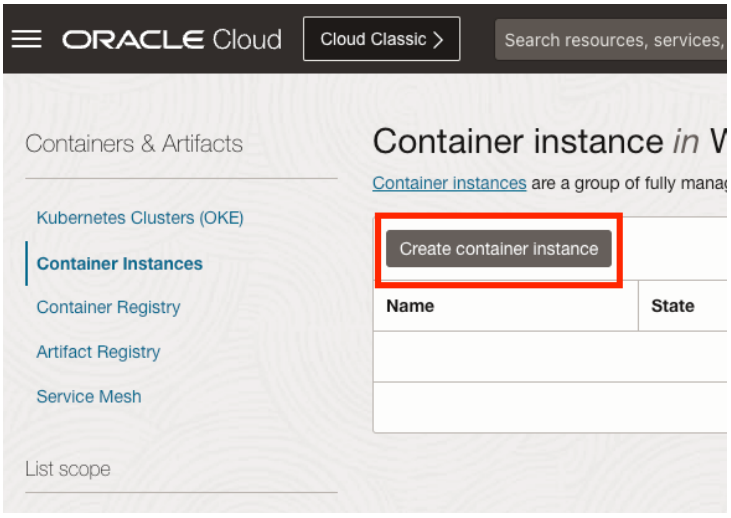
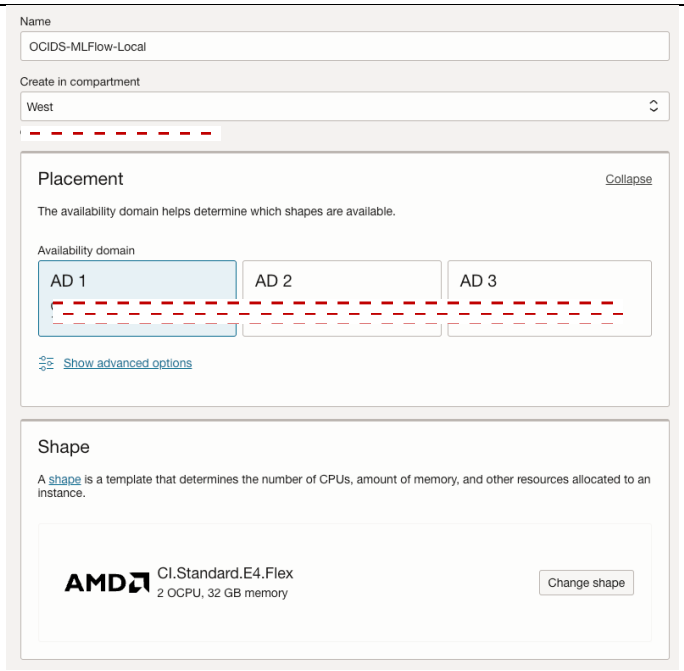
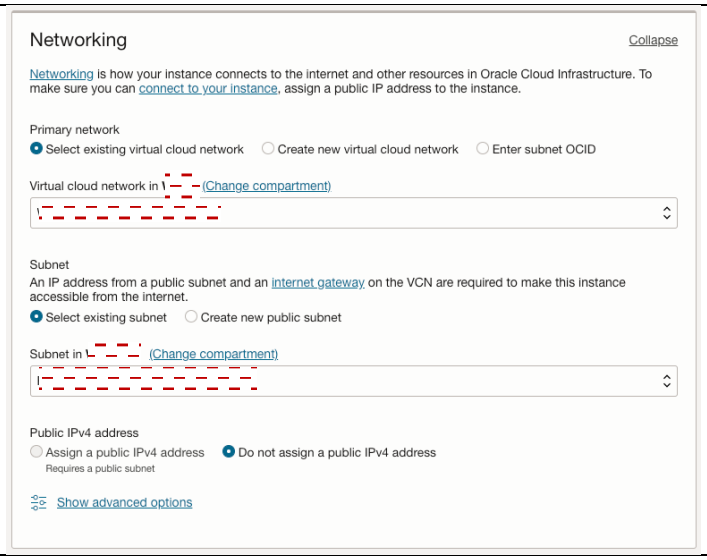
We can then select our Repository and see our uploaded Docker Image.

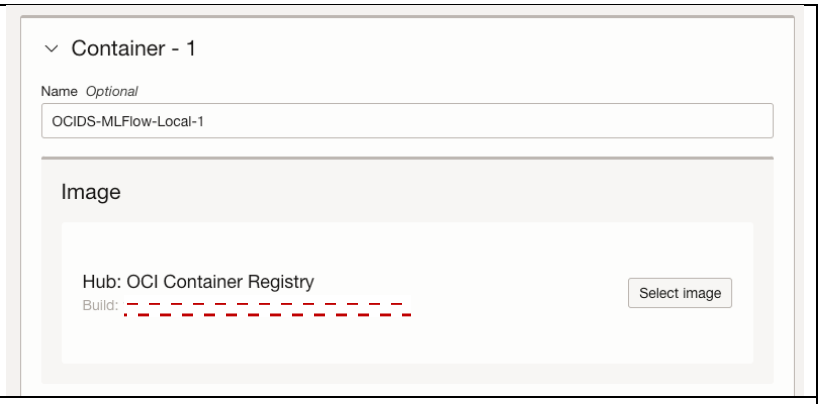
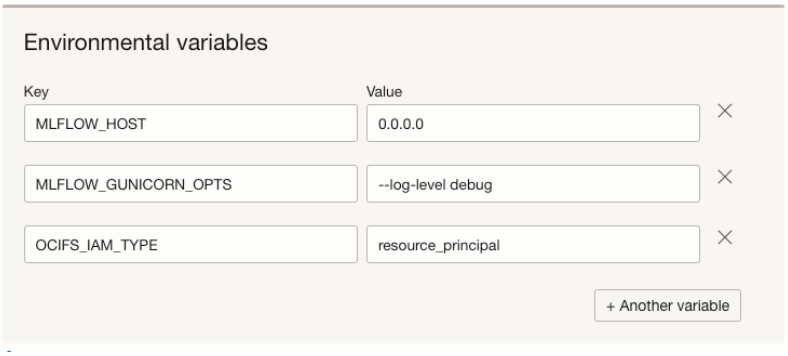
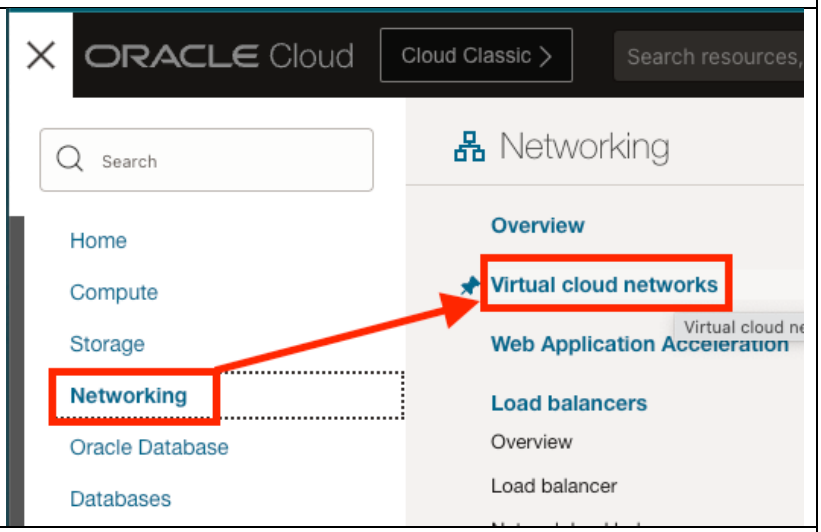
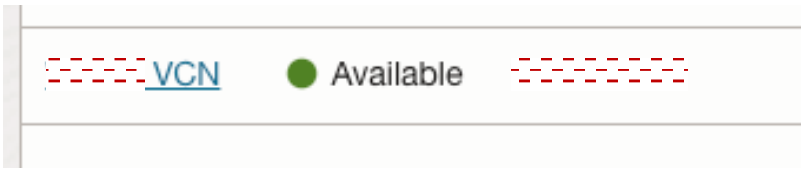


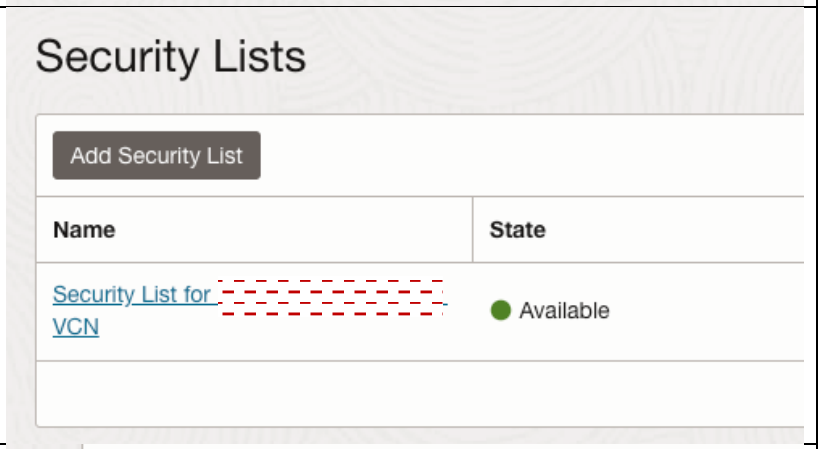
Next, we need to deploy and run this container.

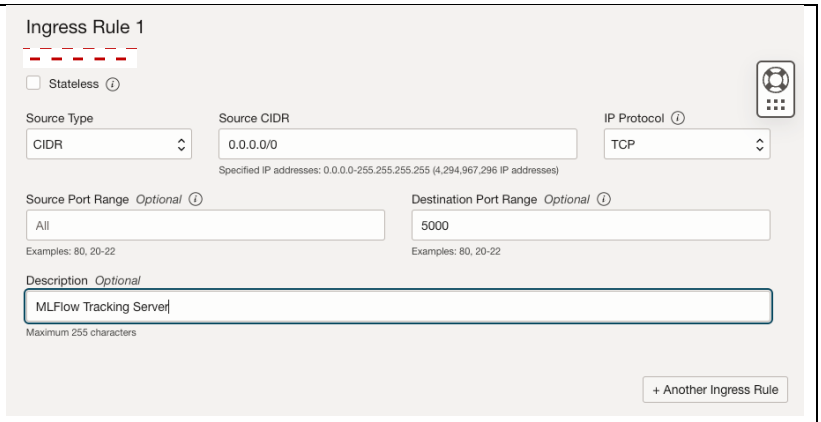
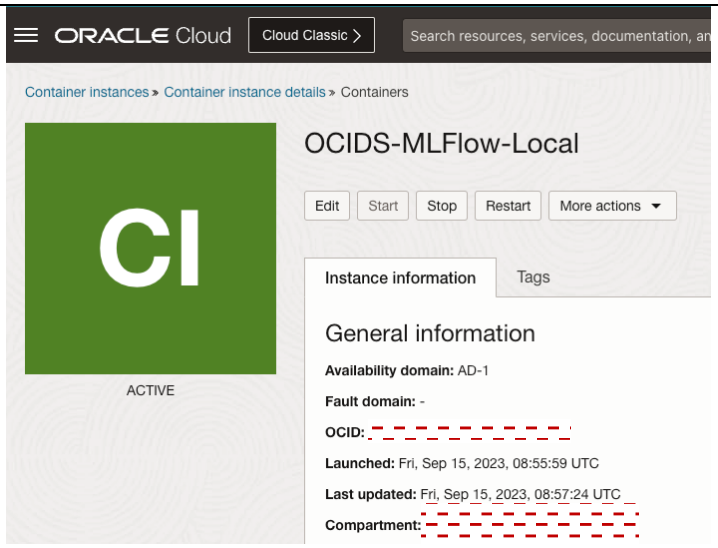
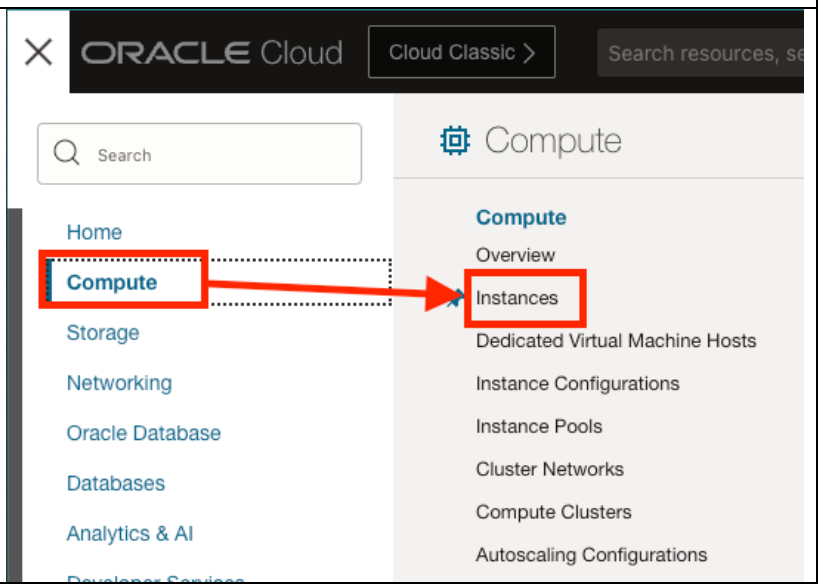
Navigate under the ***Containers & Artifacts Menu*** to ***Container Instances***.

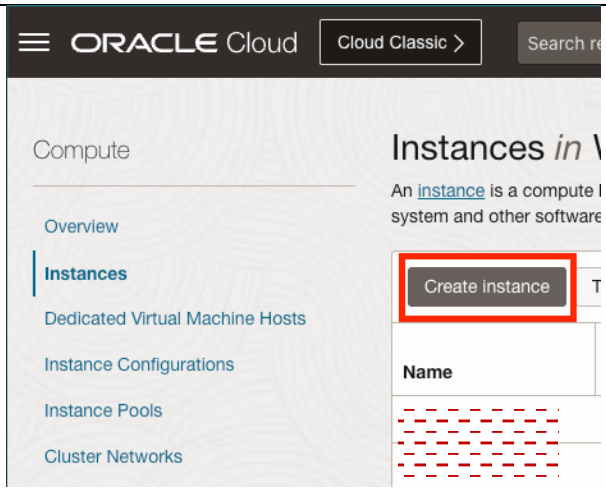
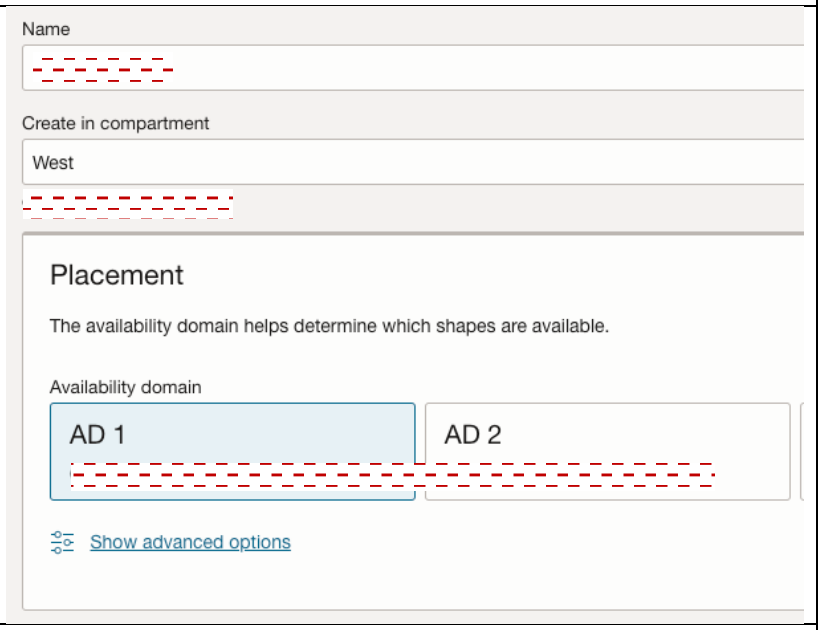
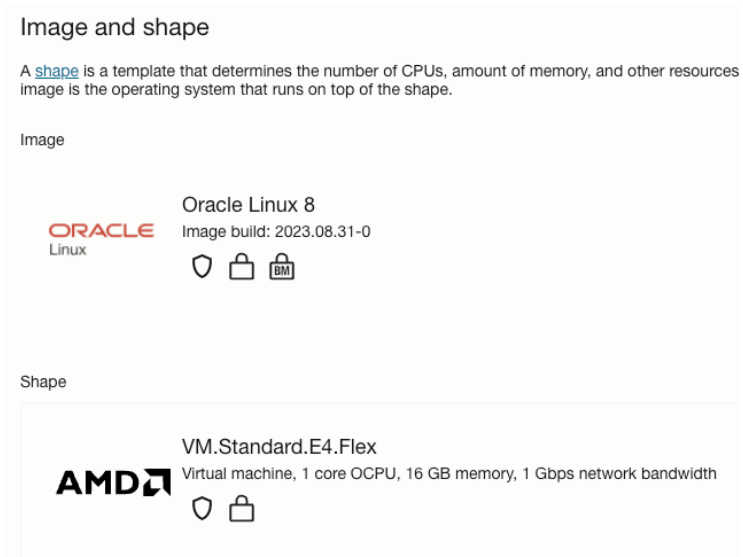


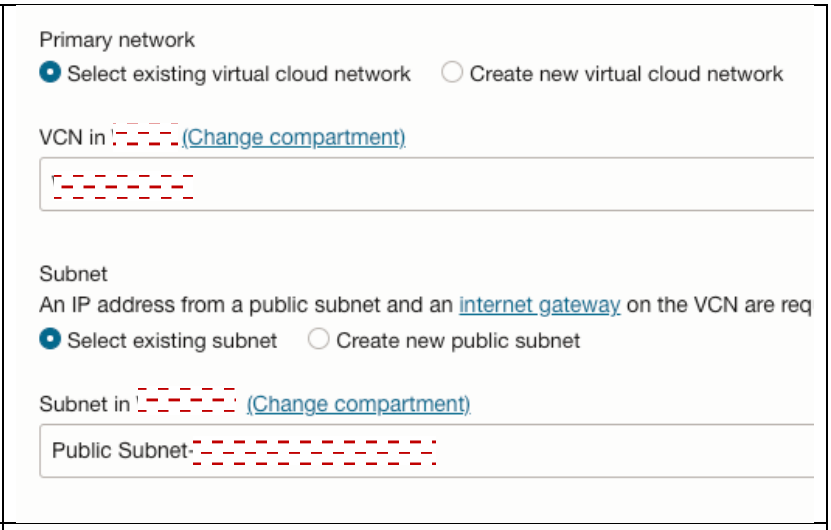
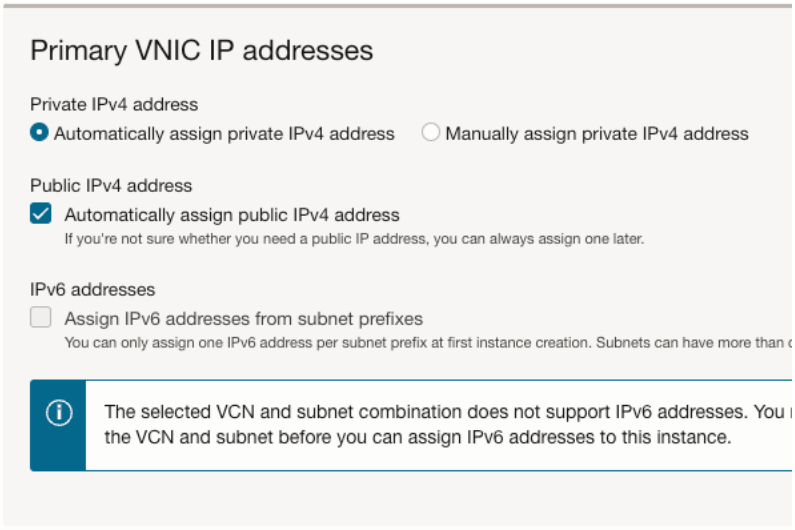
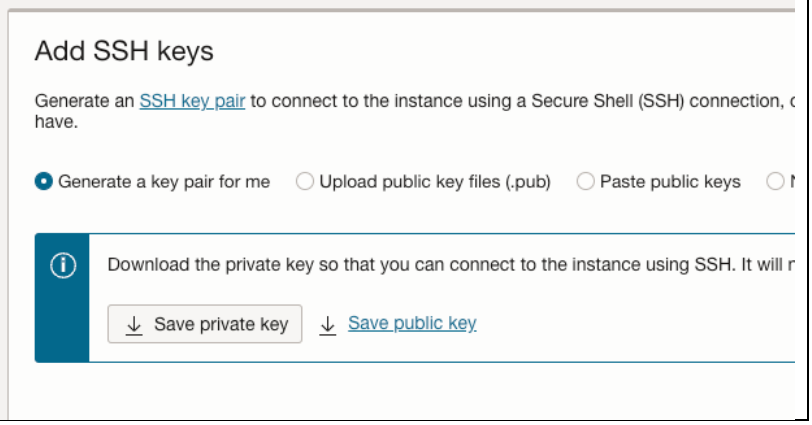
Click ***Create Container Instance.***	
Give the Container: • Name • Compartment • Availability Domain • Compute Shape	
Select the ***VCN*** And ***Private Subnet*** of where your OCI Data Science Notebook Session is located. Click ***Next***.	

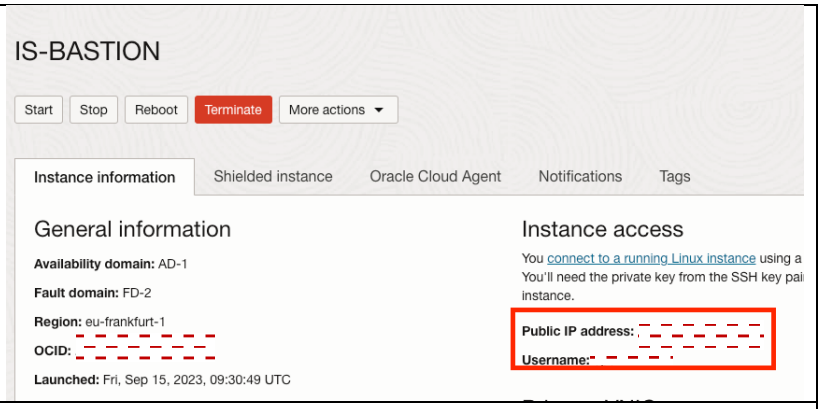
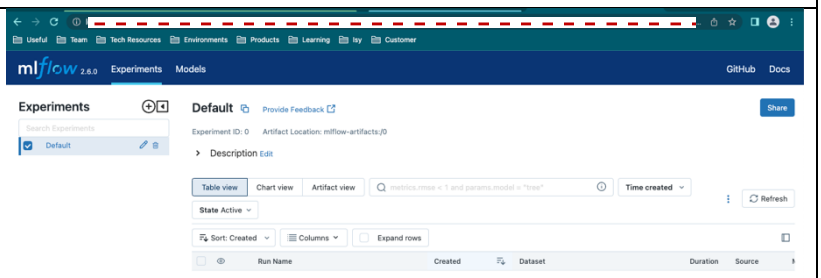
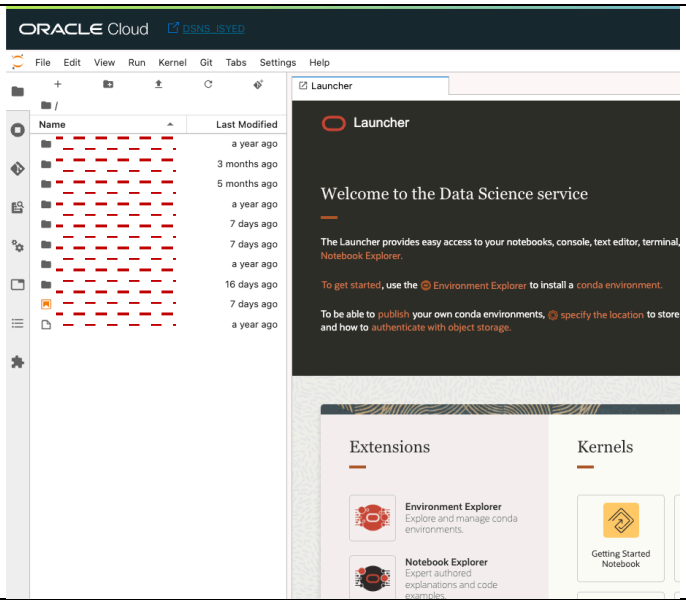
Give the Container a ***Name***. Select our ***Container Image*** from the Registry	
Enter the following Environment Variables: ***MLFLOW_HOST = 0.0.0.0*** ***MLFLOW_GUNICORN_OPTS = --log-level debug*** ***OCIFS_IAM_TYPE= resource_principal*** *Click **Next** and **Create**.*	
While the container instance is creating, we will open the correct ports on our Private Subnet Security List to allow us to connect. From *OCI Menu > Networking > Virtual Cloud Network.*	
Select your ***VCN***.	

Select your ***Private Subnet*** that has the Container Instance deployed in.	
Select the **Security List**.	
Add a new ***Ingress Rule.***	

Set: **Source:** 0.0.0.0/0 *(Access from anywhere)* **IP Protocol:** TCP **Source Port:** All **Destination Port:** 5000 **Click Create.** You can change this according to your network configurations.	
We can then go check our Container Instance is up and running.	
We will now create a Jump Server so that we can connect to our Private MLFlow Tracking Server. ***OCI Menu > Compute > Instances.***	

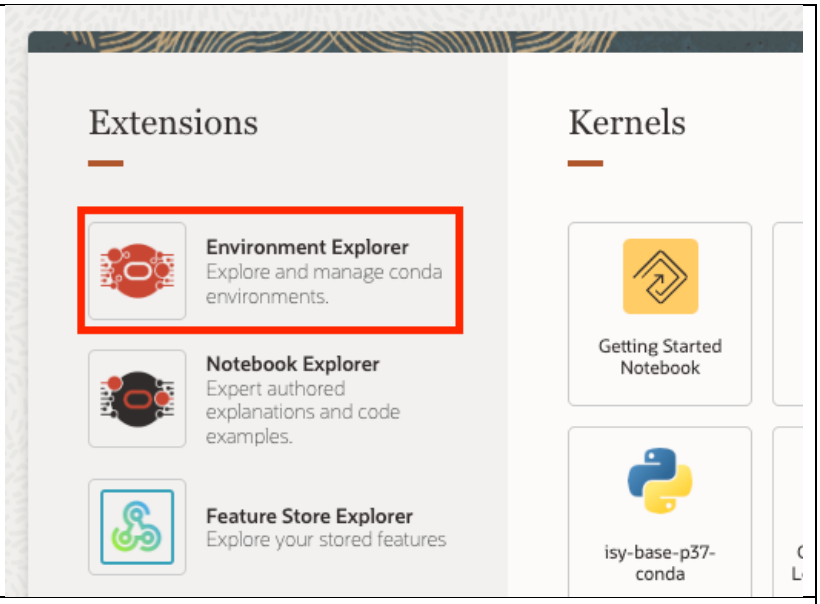
Create Instance.	
Select: • Name • Compartment • Availability Domain	
Select Image and ***Shape***. I have gone for Oracle Linux 8 and a small AMD Shape of 1 OCPU and 16GB RAM.	

Select the ***Public Subnet*** within the ***same VCN*** as our Tracking Server is Deployed.	
Ensure you have ***'Automatically assign public IPv4 address'*** selected.	
Generate and ***save*** a new pair of ***SSH Keys***. ***Click Create.***	

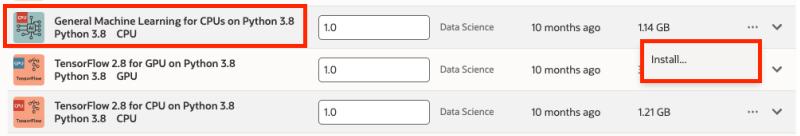
Once the instance is created make note of the ***Public IP address*** and ***Username***.	
We can now open up an ***SSH Tunnel*** and ***Port Forward*** from our Private Tracking Server to Local Laptop via the Jump Server. ***ssh –N –i <private–key> <user>@<bastion–ip> –L 5000:<tracking–server–ip>:5000***	
We can now open your local browser and visit ***localhost:5000***. This will open our MLFlow Tracking Server UI.	
We will now ***open*** our ***OCI Data Science Environment*** and push some experiment data to our tracking server.	

We will install the ***General Machine Learning for CPU on Python 3.8*** Conda Environment.

Click on ***Environment Explorer.***



Find the ***General Machine Learning for CPU on Python 3.8 Conda Environment, select the 3 dots and click Install***.

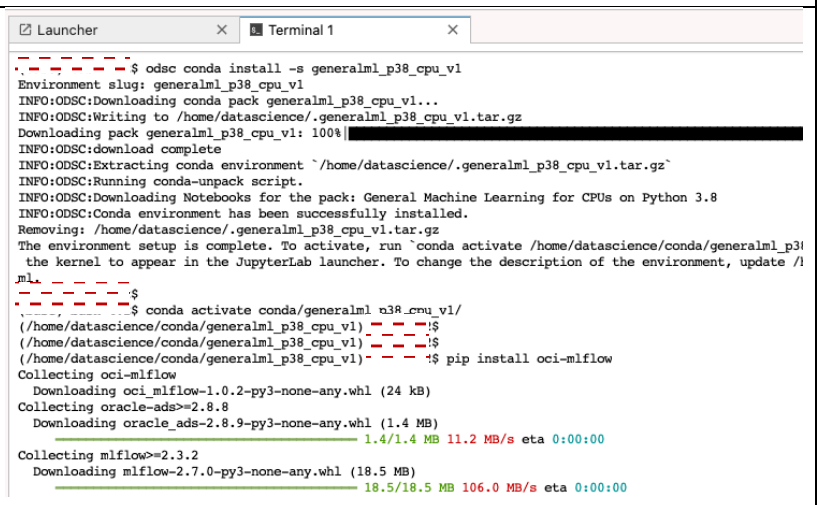


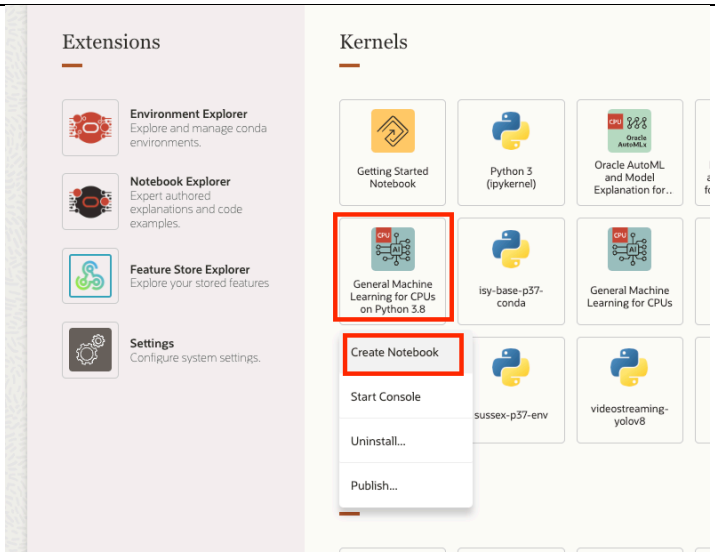
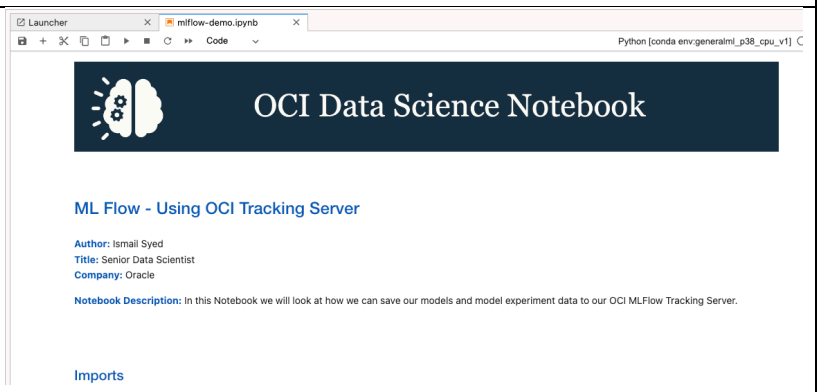
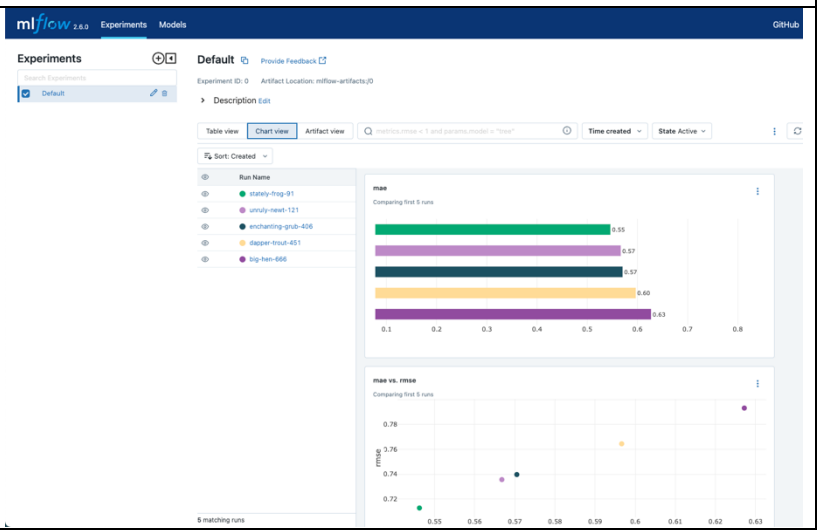
Once the Conda Environment is ***finished*** installing, let's activate it.

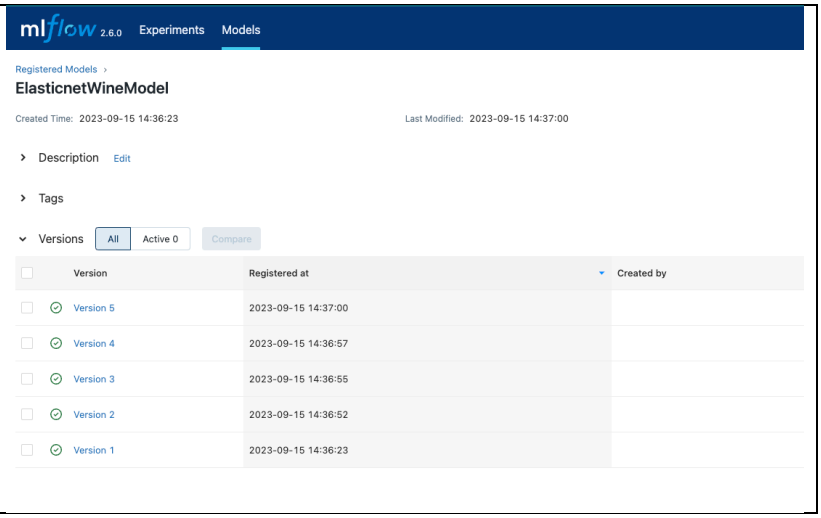
conda activate conda/generalml_p38_cpu_v1/

You can then install the oci-mflow library.

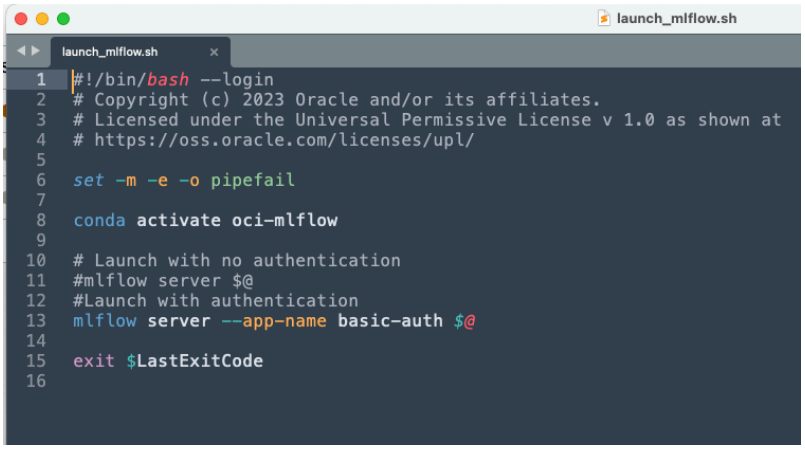
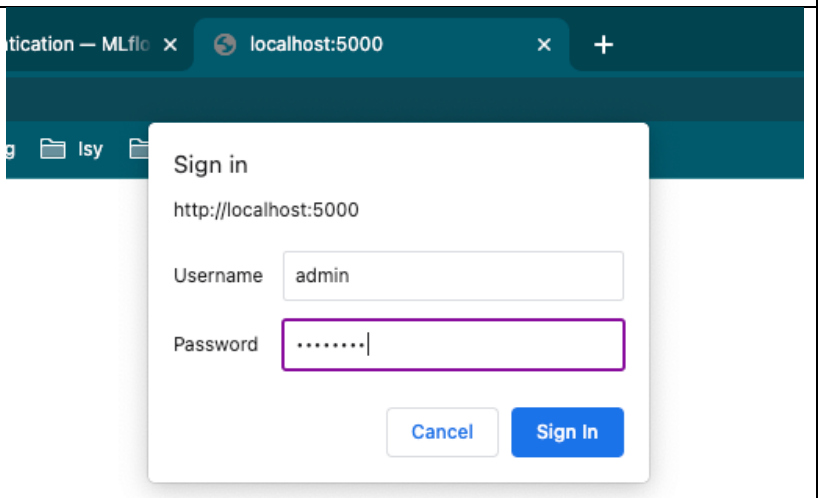
pip install oci-mlflow



Once the ***oci-mlflow*** library is installed, head to the ***Launcher*** and create a new Notebook from our new Conda Environment.	
Execute the ***mlflow-demo.ipynb*** Notebook. Notebook should be in the same directory as this guide. The goal of this Notebook is to have all the experiments written to the Tracking Server.	
We can then visit our MLFlow Tracking Server at ***localhost:5000*** to see all our experiments done.	

We can also see the different Versions of the Elastic Net Model we saved.	

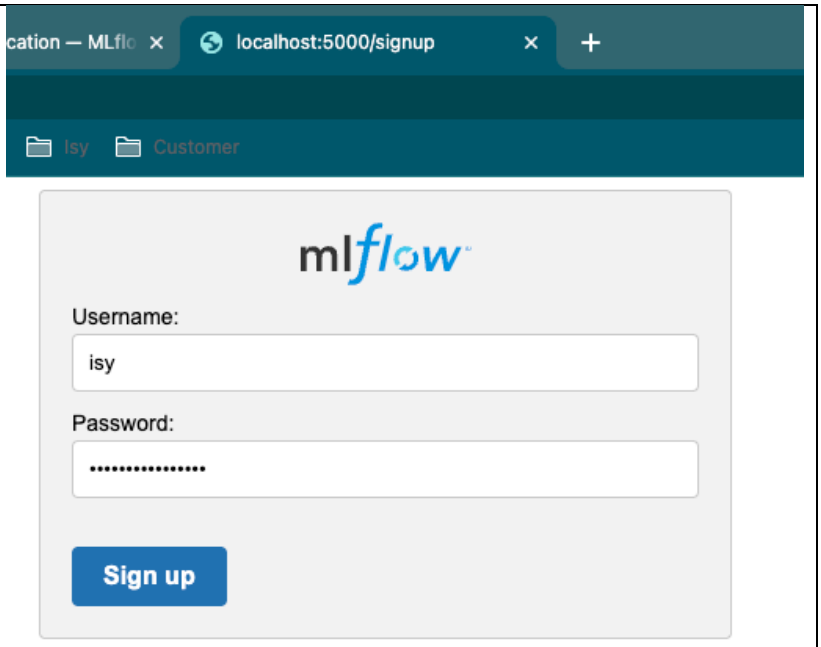
Optional: Setting up MLFlow Authentication

We will now look at how to set up authentication to the tracking server. ***This step will need to be done before building and pushing our MLFLow Image to the Oracle Container Registry.*** We will need to edit our Docker Image. Open the following file: ***oci-mlflow-main/container-image/run/ launch_mlflow.sh*** We can then replace the line ***mlflow server $@*** with ***mlflow server --app-name basic-auth $@***	
You will then have to follow the same steps above to Build, Push, and Create the Container Instance.	***You will then have to follow the same steps above to Build, Push, and Create the Container Instance.***
This time when the Container Instance is up and running and you visit the Tracker Server you will be prompted with a username and password. Default username is: ***admin*** Default password is: ***password***	

We can create a new user by visiting the URL ***localhost:5000/signup***

Let's create a new user called ***isy*** with password **********.

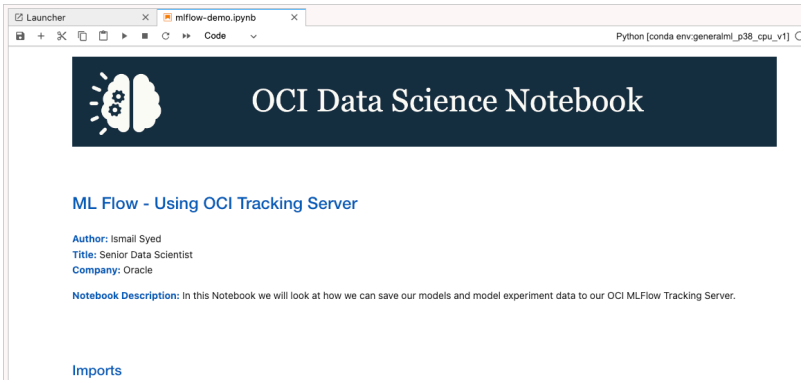
By default, this user only has Read Access. To grant more permissions you can do this using the Python API. Visit the MLFlow Docs.



Execute the **mlflow-auth-demo.ipynb** Notebook which has a few modifications to deal with authentication.

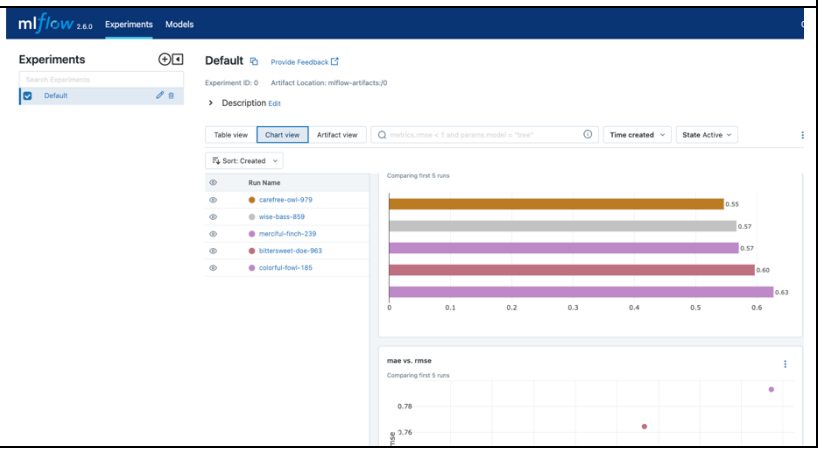
Notebook should be in the same directory as this guide.

The goal of this Notebook is to have all the experiments written to the Tracking Server which requires Authentication.

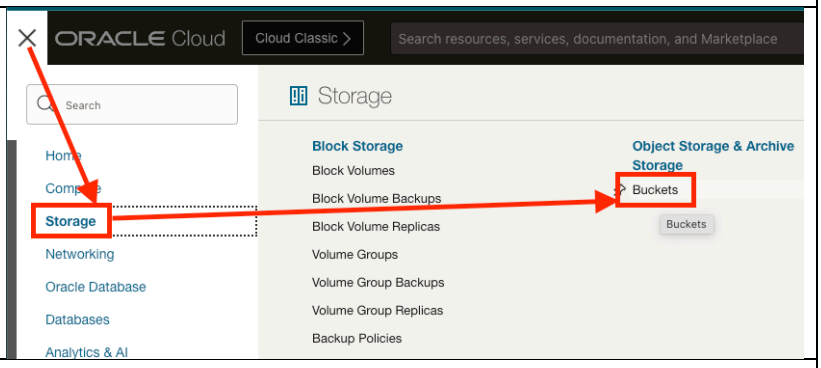
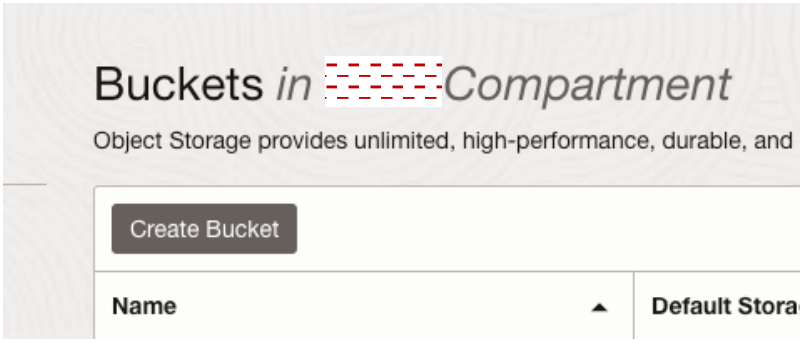
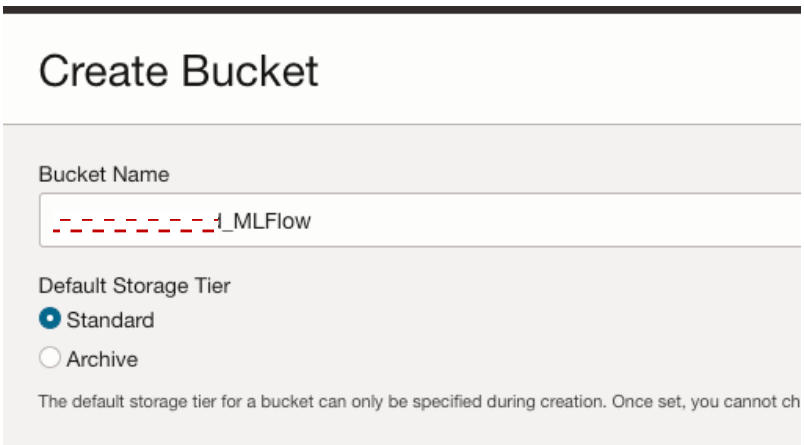


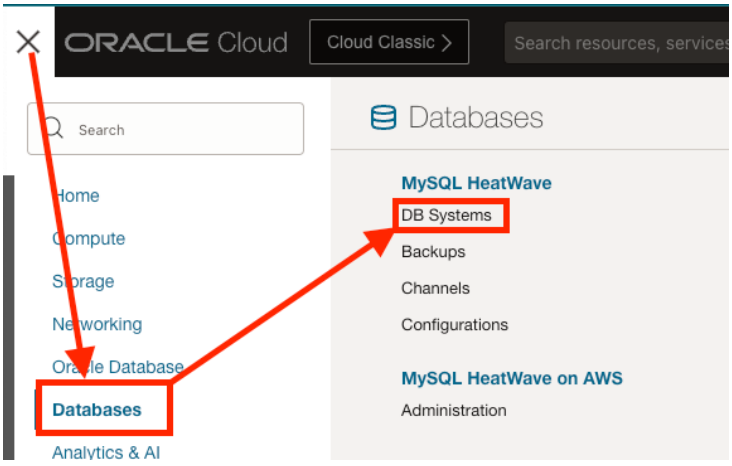
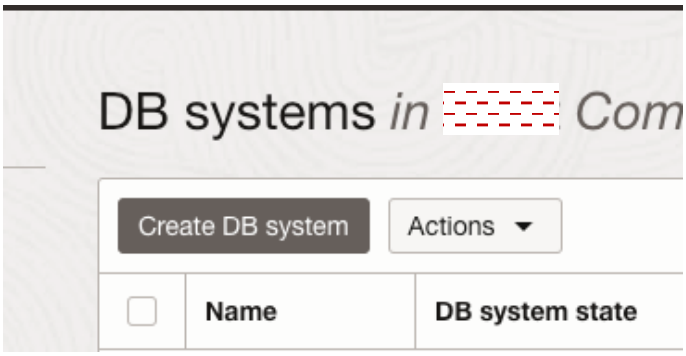
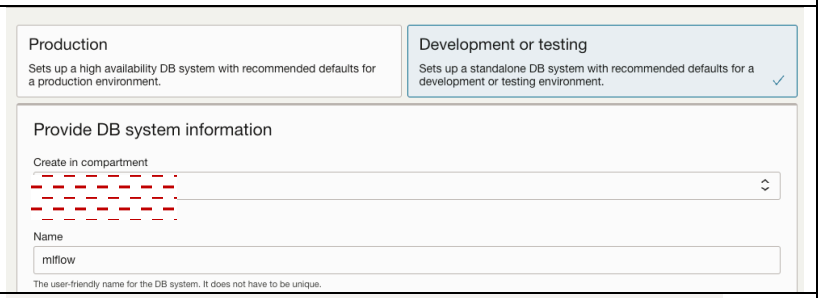
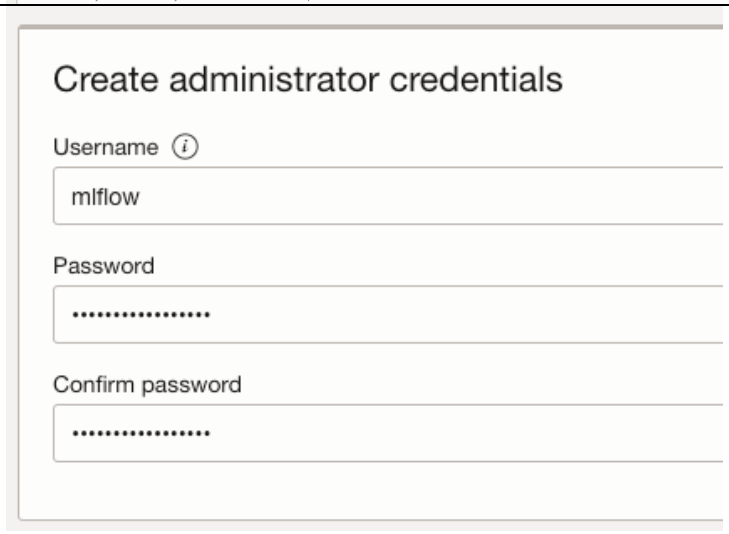
We can then visit our MLFlow Tracking Server at ***localhost:5000*** to see all our experiments done.

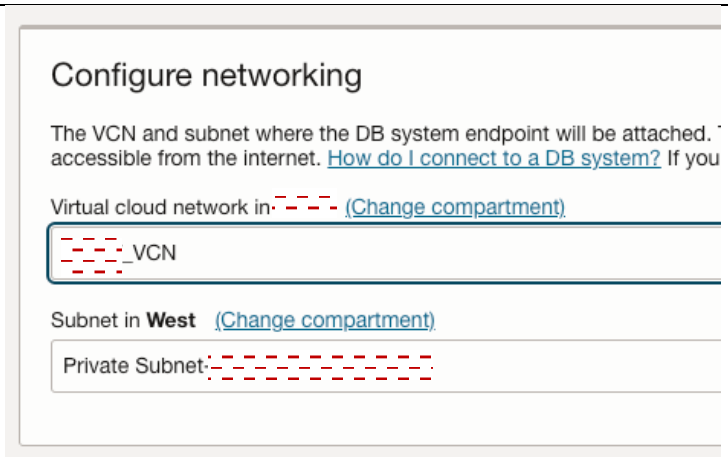
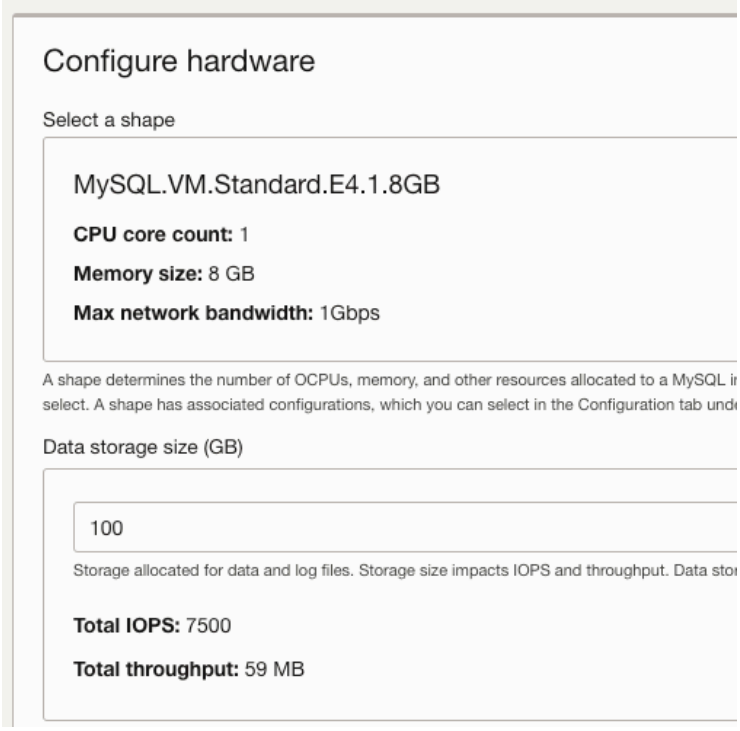
Remember to login using our username and password as defined in our Notebook. – In the notebook we will use the admin user.

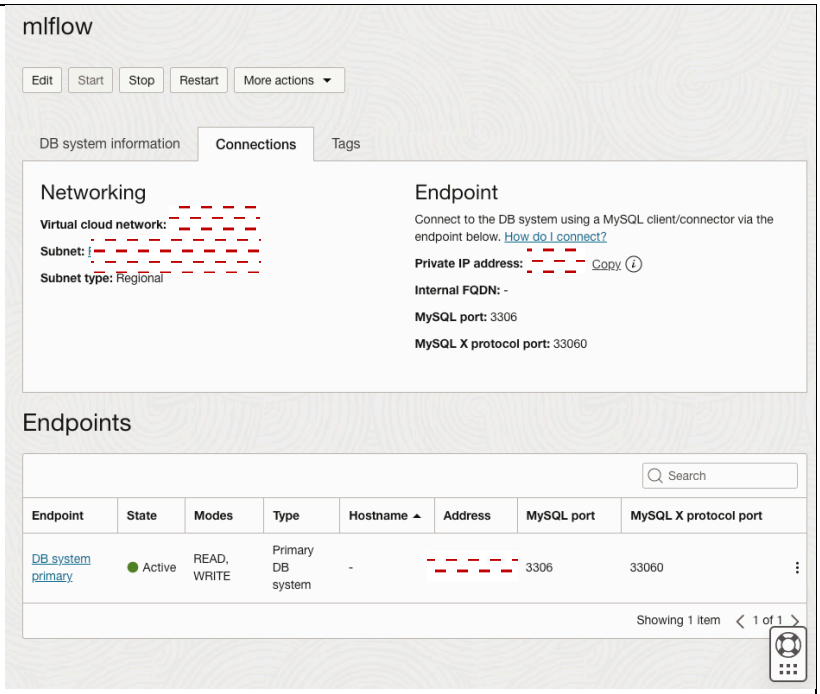
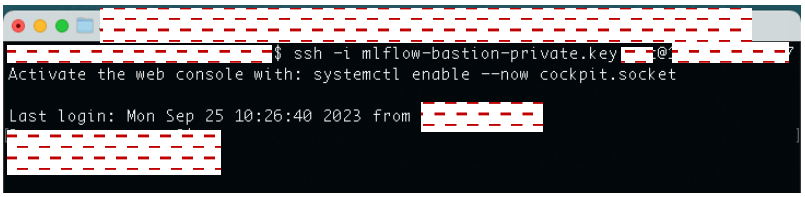
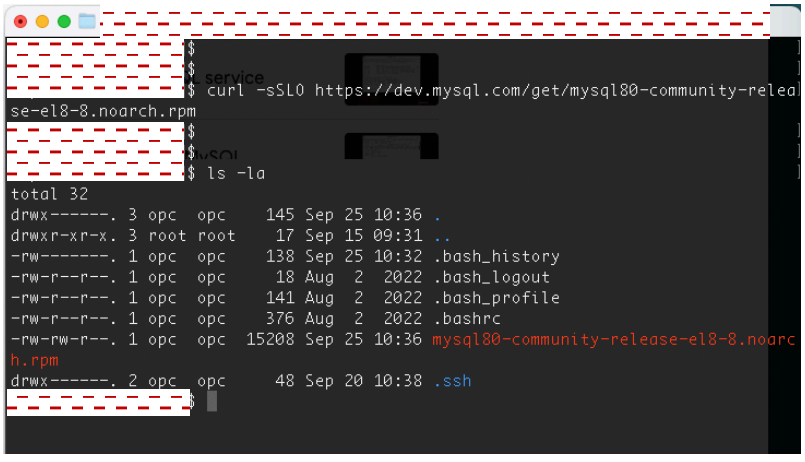


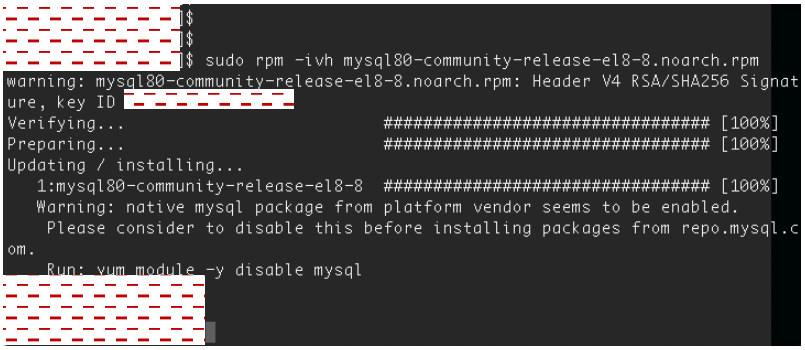
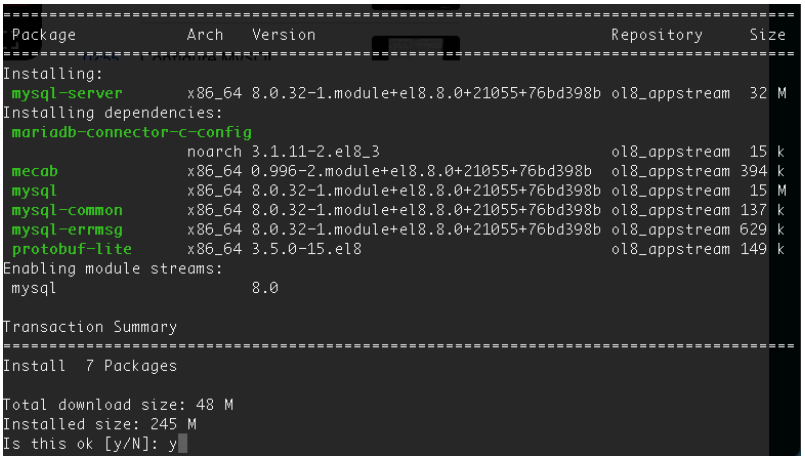
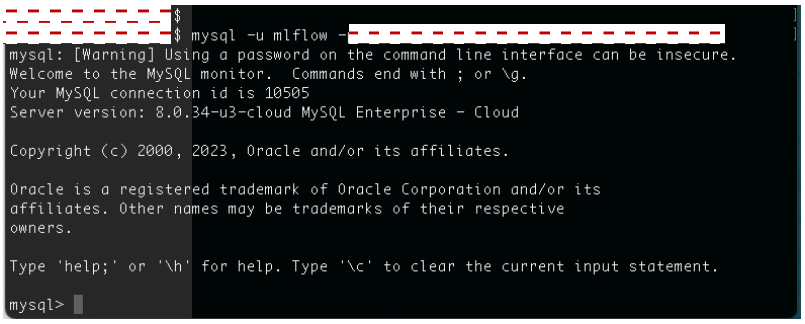
Approach 2: Store Artifacts in Object Storage and Experiments in MySQL

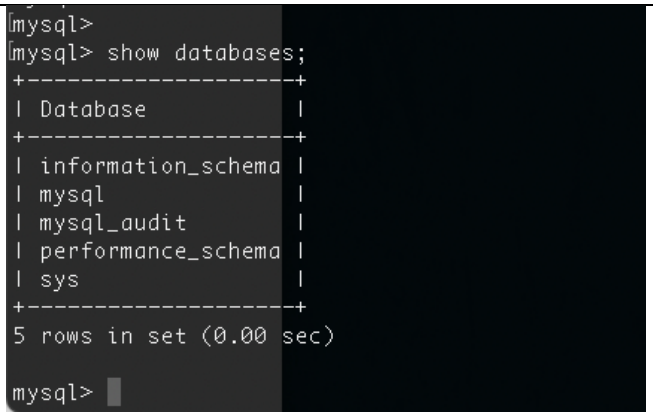
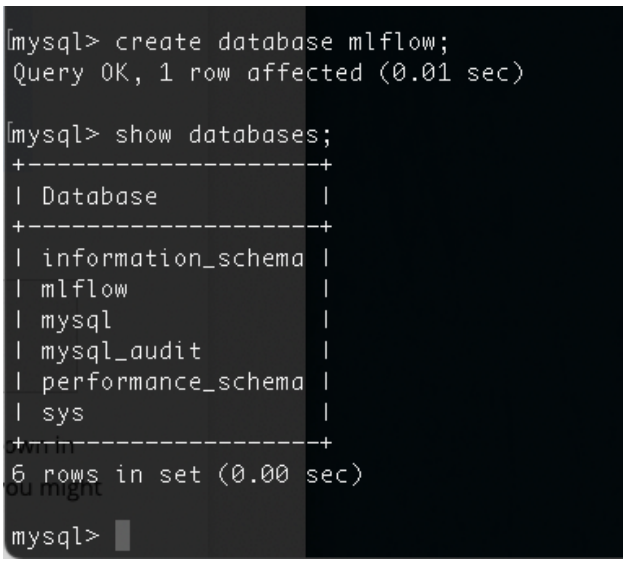
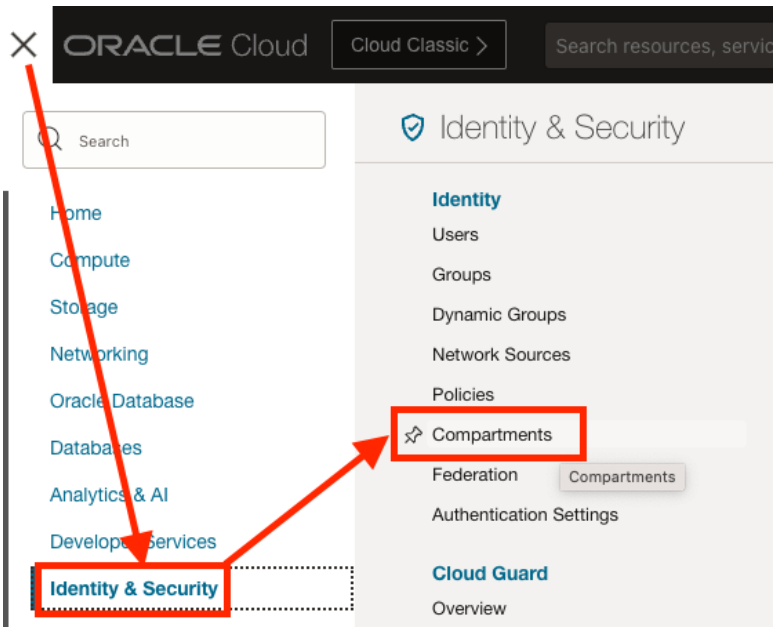
Step	Screenshot
Majority of the Steps are the same for utilising Object Storage and MySQL as storage options. In this Approach we will focus on setting up Object Storage, MySQL and Configuring a new Container Instance to utilise them.	
First let's create an Object Storage Bucket. Navigate to ***OCI Menu > Storage > Buckets.***	
Click ***Create Bucket.***	
Give your Bucket a ***Name*** and ***Click Create***.	

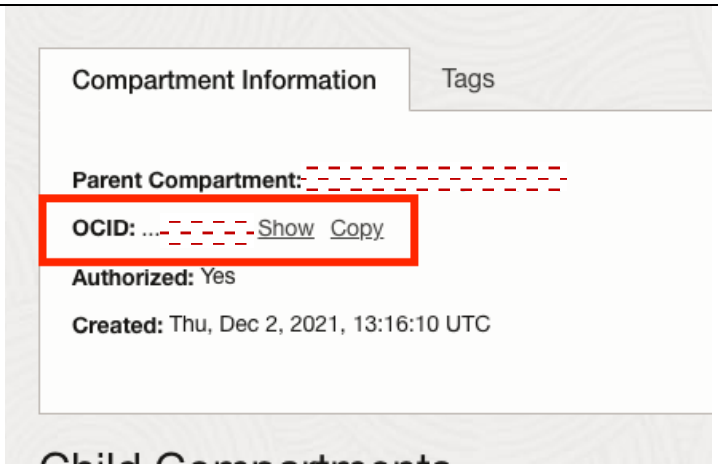
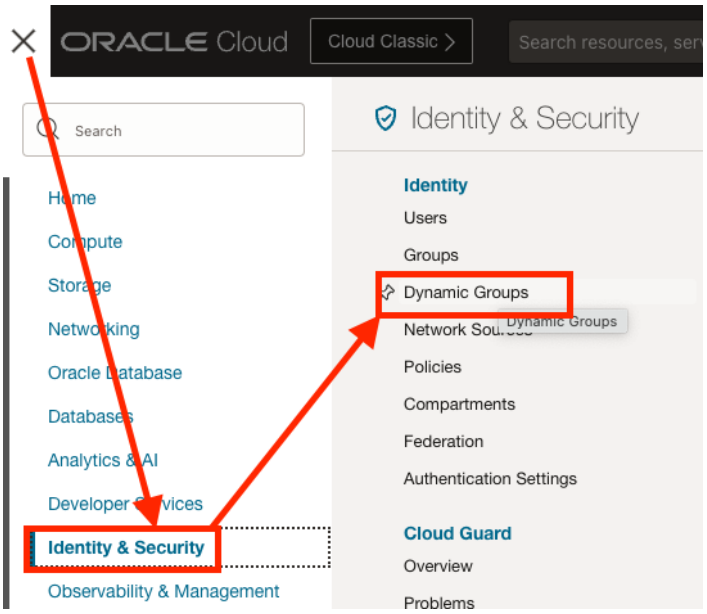
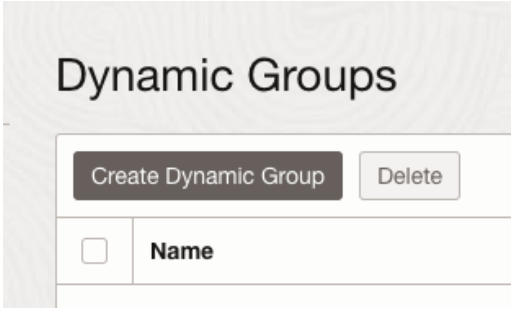
We will now create a MySQL Instance. Navigate ***OCI Menu > Databases > DB Systems.***	
Click ***Create DB System.***	
Select Type as ***Development or Testing.*** Give a name of ***mlflow.***	
Set ***Username*** and ***Password***. ***mlflow*** **********	

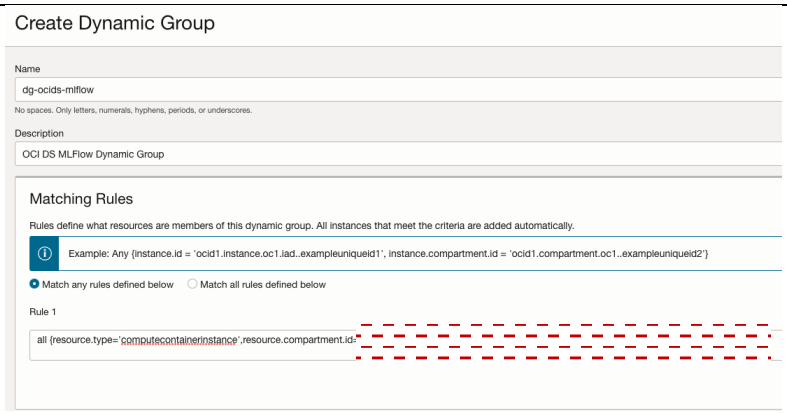
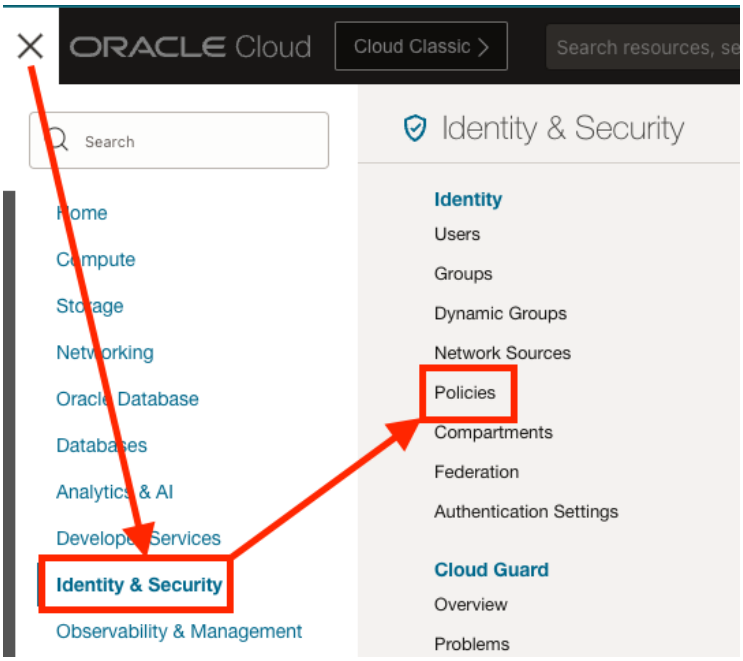
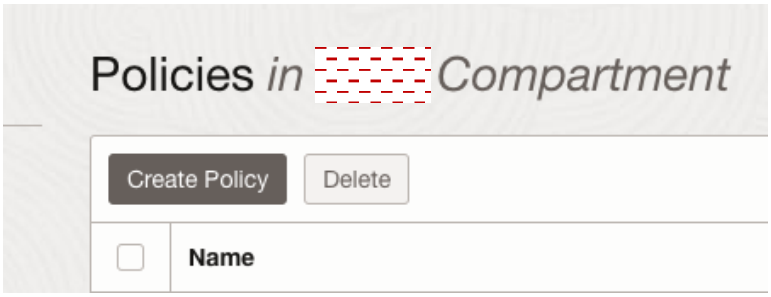
Select the ***VCN*** and ***Private Subnet*** of where our MLFlow Tracking Server will be located.	
Choose Hardware requirements. I have gone for a small shape with ***MySQL.VM.Standard.E4.1.8GB Compute 100GB Storage.*** Leave all the other settings as default. Click **Create**.	

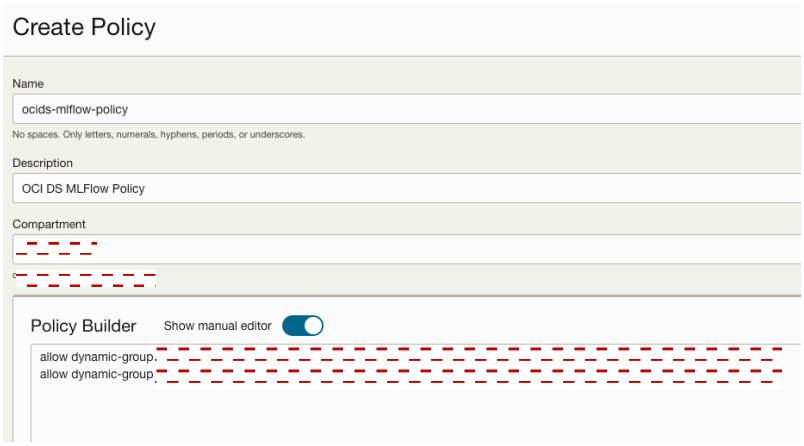
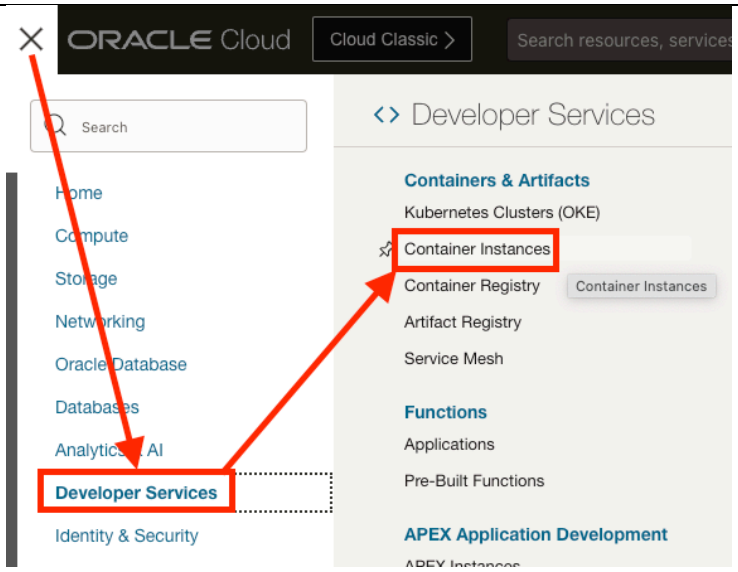
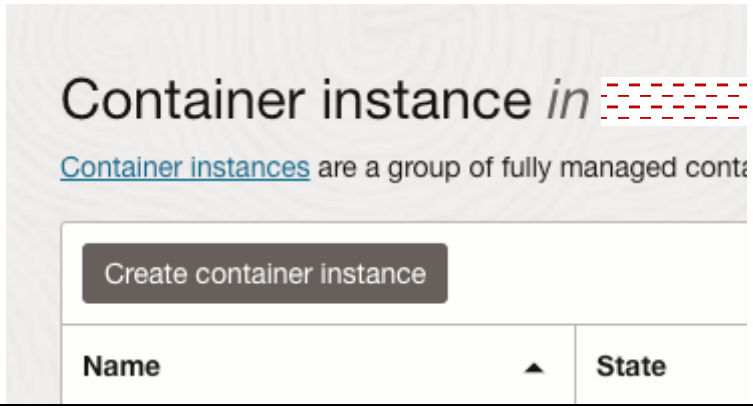
Once created, make note of the ***Username***, ***Password***, ***Private IP Address***, ***Port*** and ***db_name***.	
We will now login to our MYSQL DB and ***create a new Database*** to host our experiments. SSH into our BASTION server. ***ssh -i <private-key> opc@<host>***	
Download the ***MySQL RPM Package***. Make sure you use the correct version according to your environment – View Versions here: https://dev.mysql.com/downloads/repo/yum/ ***curl -sSLO https://dev.mysql.com/get/mysql80-community-release-el8-8.noarch.rpm***	

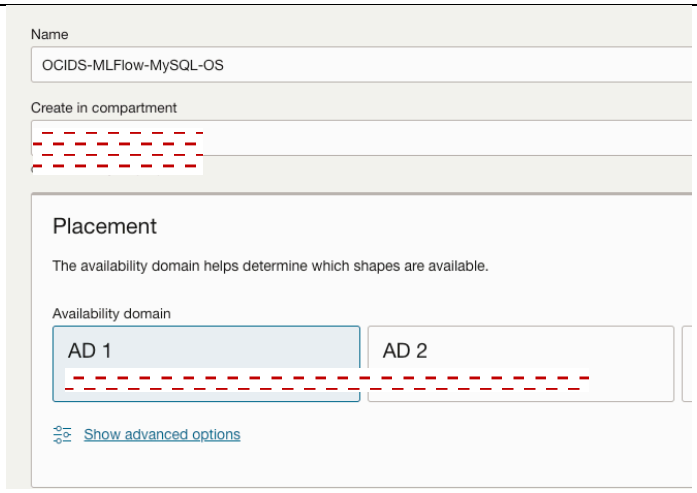
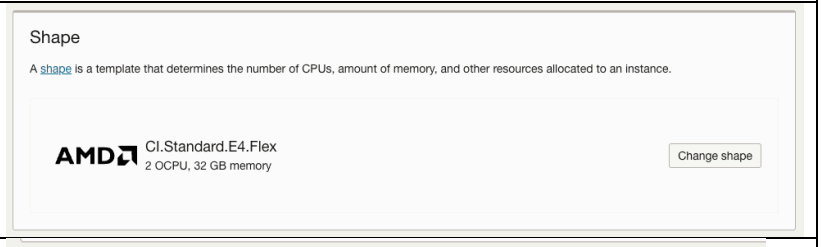
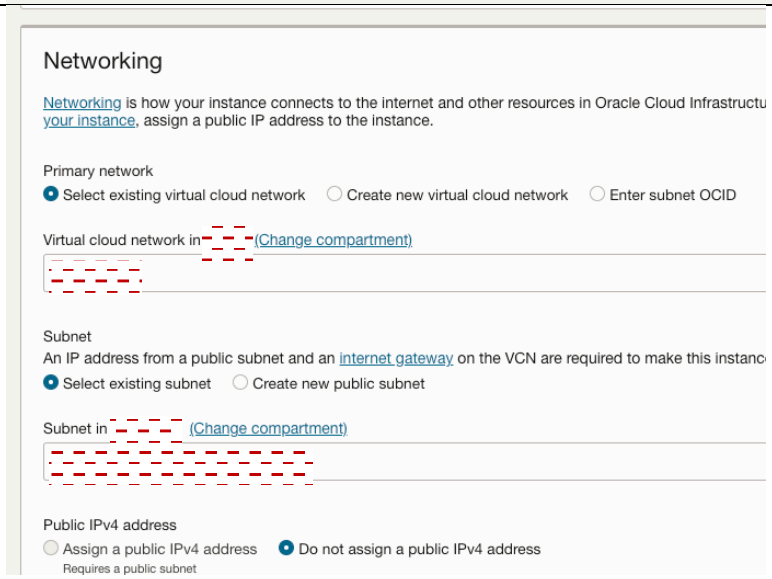
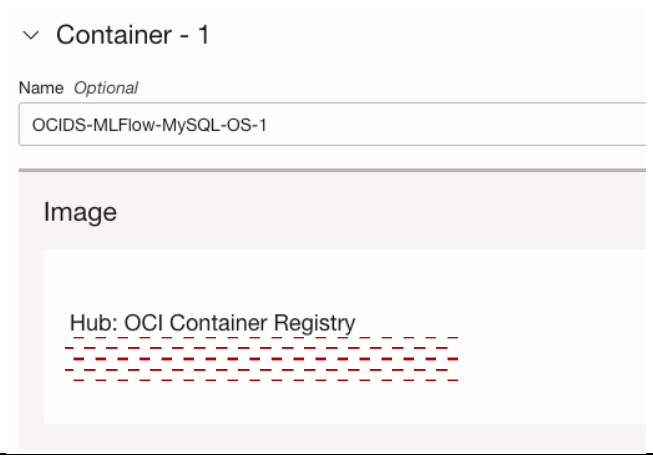
We will now install the yum repository. ***sudo rpm -ivh mysql80-community-release-el8-8.noarch.rpm***	
Now ***install*** the ***mysql server*** using: ***sudo yum install mysql-server***	
We can then login to our remote MySQL Server using: ***mysql -u <username> -p<password> -h <hostname> -P <port>***	

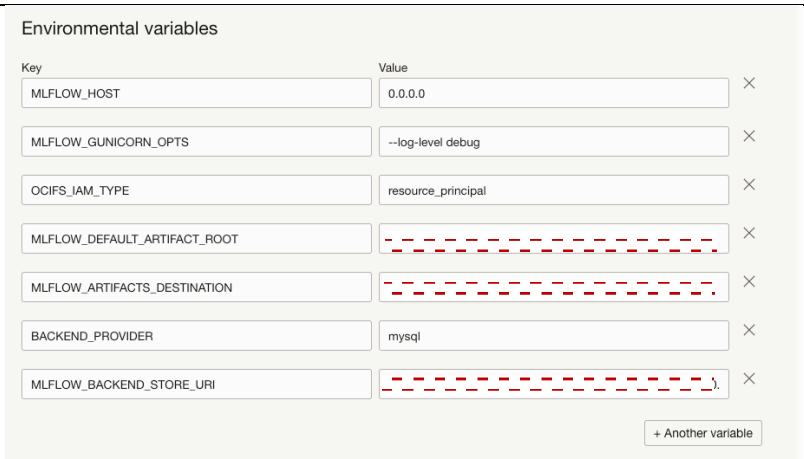
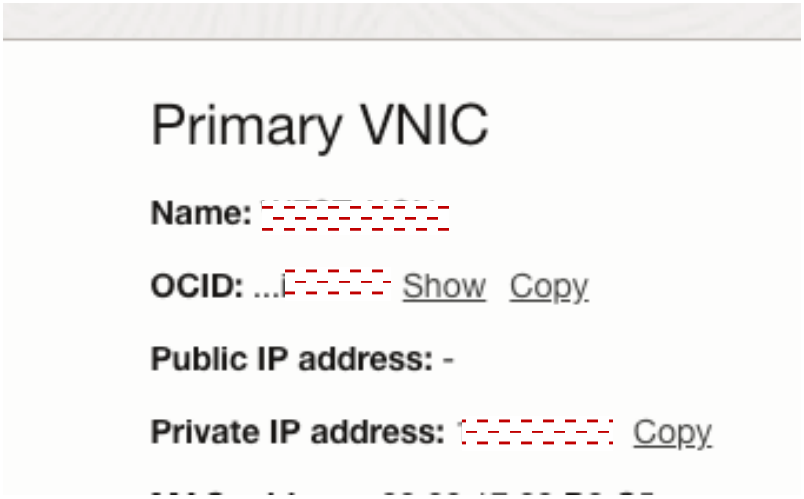
We can then list our Databases. ***show databases;***	
Let's create a new database called ***mlflow***. ***create database mlflow;*** Once created, we can exit our MySQL env and Bastion host.	
Now let's create a dynamic group to identify ***computecontainerinstances*** within our compartment. Navigate to ***OCI Menu > Identity and Security > Compartment.***	

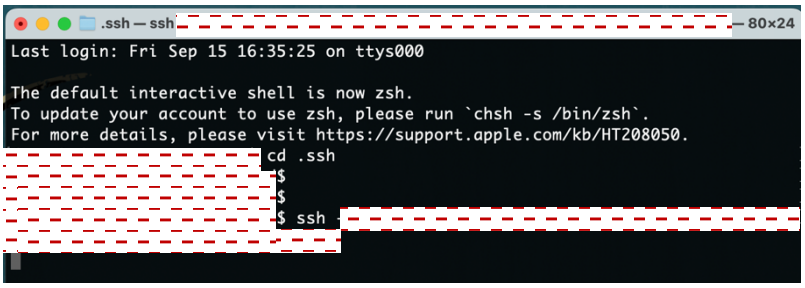
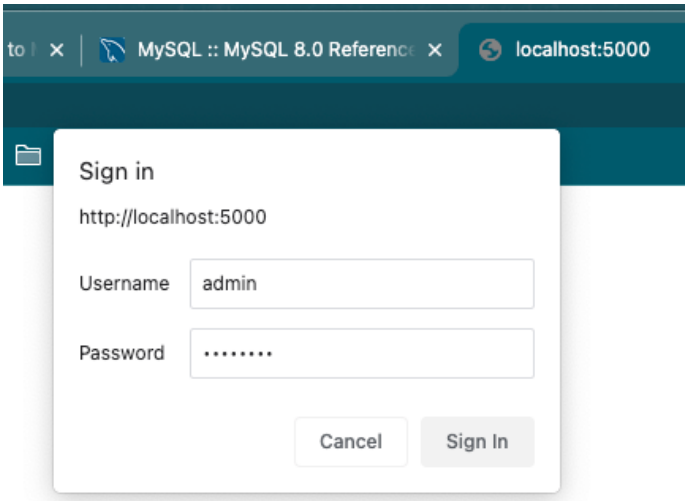
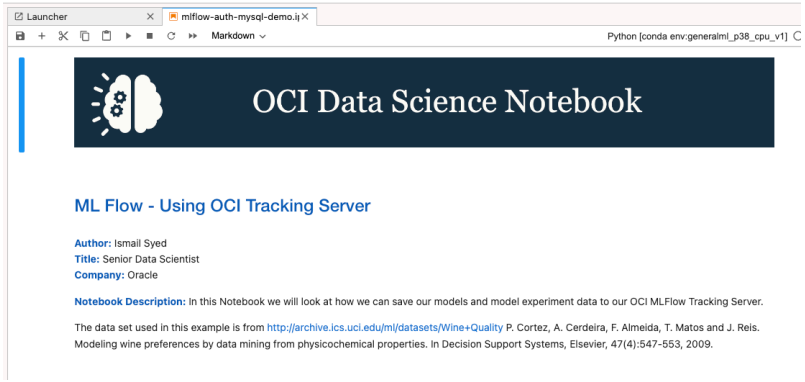
Select your ***compartment*** hosting your resources.	
Make note of your Comp***artment OCID***.	
Now we will navigate to ***OCI Menu > Identity & Security > Dynamic Groups.***	
Click ***Create Dynamic Group.***	

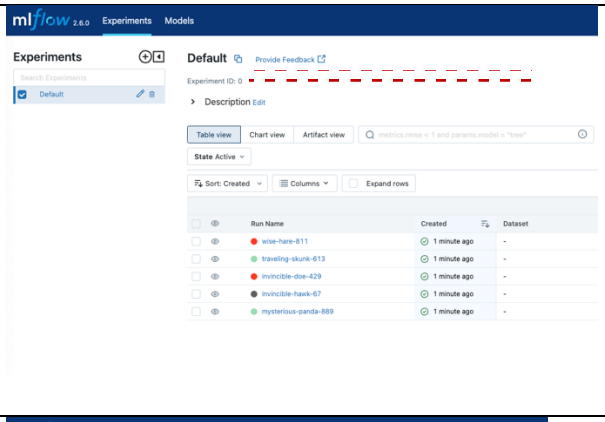
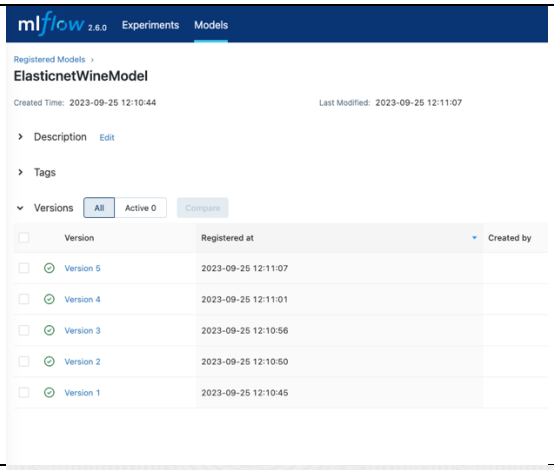
Give the Dynamic Group a Name, Description and Enter the Following Rule. **all {resource.type='computecontainerinstance',resource.compartment.id='<copartment-ocid>'}**	
We will now add some policies to grant the dynamic group access to other OCI resources. Navigate ***OCI Menu > Identity & Security > Policies.***	
Click ***Create Policy***.	

Give the Policy a ***Name***, ***Description*** and ***Enter*** the following rules and ***Click Create***. ***allow dynamic-group <your dynamic group> to manage objects in compartment <your_compartment_name> where all {target.bucket.name=<your_bucket_name>}*** ***allow dynamic-group container-instance-group to use secret-family in compartment <your_compartment_name>***	
We will now create a new Tracking Server Container Instance which is configured to use the Object Storage and MySQL DB to store Artifacts and Experiments. Navigate to ***OCI Menu > Developer Services > Container Instances.***	
Click ***Create Container Instance.***	

Give the container instance a ***Name*** and select the ***Availability Domain.***	
Select a Compute Shape. I have gone for 2 OCPU and 32 GB Memory.	
Select the ***VCN*** and ***Private Subnet*** we will be deploying our Tracking Server into. Best to select the same VCN and Private Subnet as our MySQL DB and OCI Data Science. **Click Next.**	
Give our Container Instance a ***Name*** and ***Select our Container Image*** from the Registry. I have selected my image with Authentication Enabled.	

Next, we will enter the following Environment Variables. This is where we will tell our Tracking Server the Location of our ***Object Storage*** and ***MySQL DB***. **Click Next** and then **Create**.	 ***MLFLOW_HOST***=0.0.0.0 ***MLFLOW_GUNICORN_OPTS***=--log-level debug ***OCIFS_IAM_TYPE***=resource_principal ***MLFLOW_DEFAULT_ARTIFACT_ROOT***=oci://<bucket_name>@<namespace>/ARTIFACTS/ ***MLFLOW_ARTIFACTS_DESTINATION***=oci://<bucket_name>@<namespace>/ARTIFACTS/ ***BACKEND_PROVIDER***=mysql ***MLFLOW_BACKEND_STORE_URI***=mysql+mysqlconnector://{mysql-user}:{mysql-passsword}@{mysql-host}:3306/mlflow
Once the instance is created make not of the ***Private IP address***.	

We can now open up an SSH Tunnel and Port Forward from our Private Tracking Server to Local Laptop via the Jump Server we created earlier in this documentation. ***ssh –N –i <bastion–private–key> opc@<bastion–host> –L 5000:<tracking–server–host>:5000***	
Using a local browser, we can visit the tracking server at ***localhost:5000.*** The Tracker Server will prompt you with a username and password. Default username is: ***admin*** Default password is: ***password***	
Execute the **mlflow-auth-mysql-demo.ipynb** Notebook. Notebook should be in the same directory as this guide. The goal of this Notebook is to have all the experiments written to the Tracking Server which requires Authentication.	

Once executed, we can return to our MLFlow Tracking Server and view the experiments that have been logged.	
We can also view the Models stored in the tracking server.	
Finally, let's check our Object storage Bucket and MySQL DB (mlflow) where we should see some stored entries and artifacts.