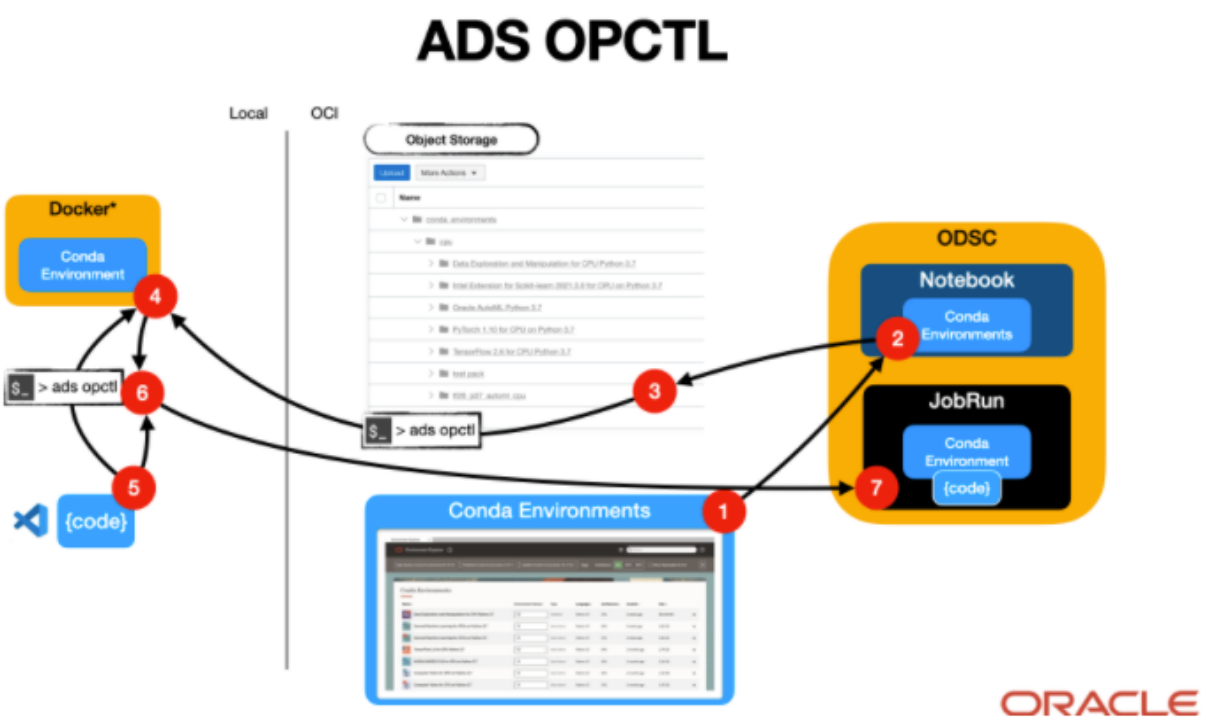
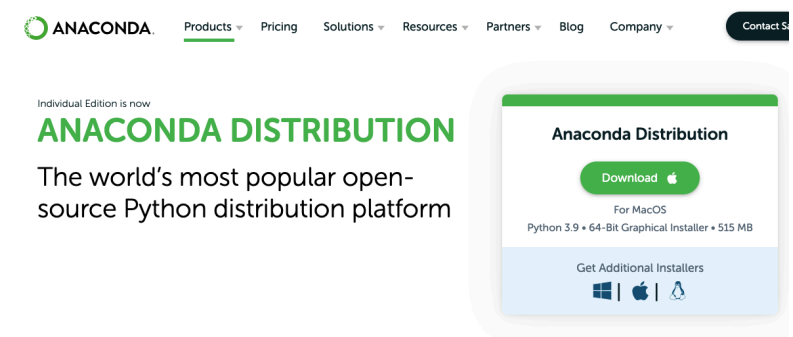
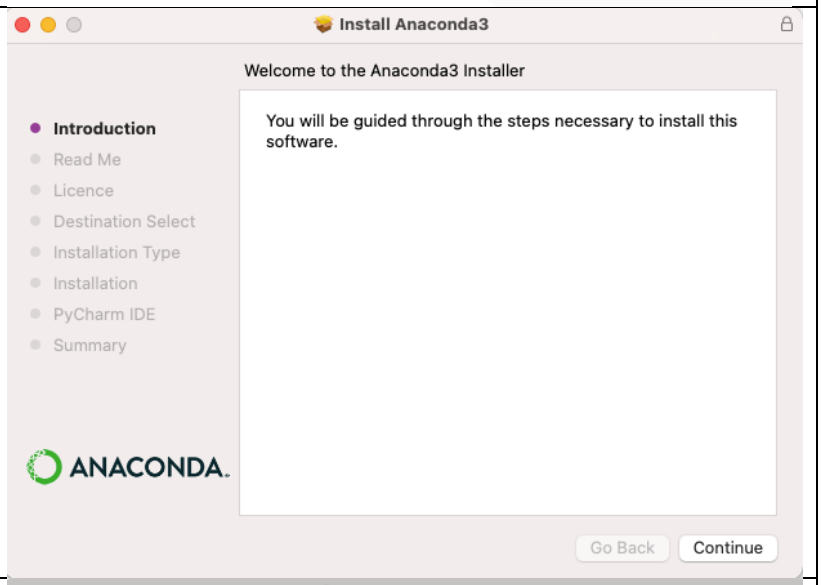
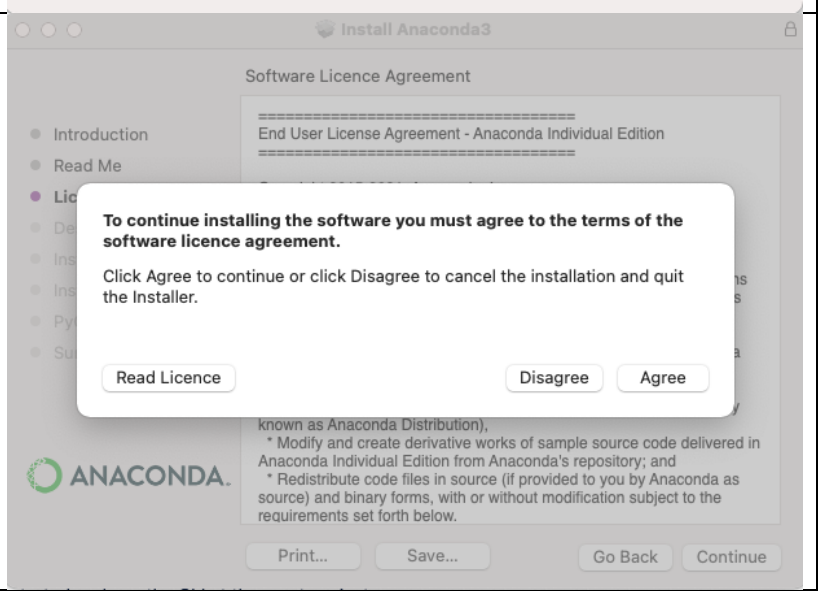
ADS OPTCL Jobs

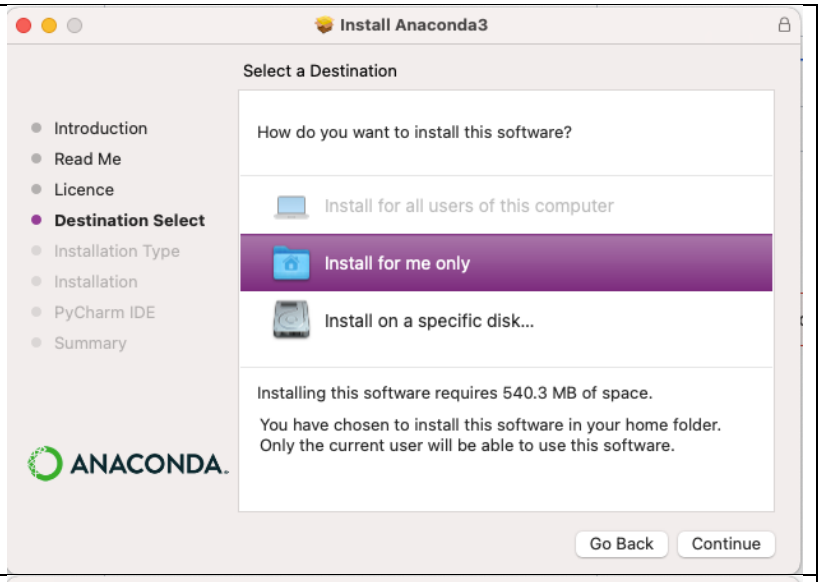
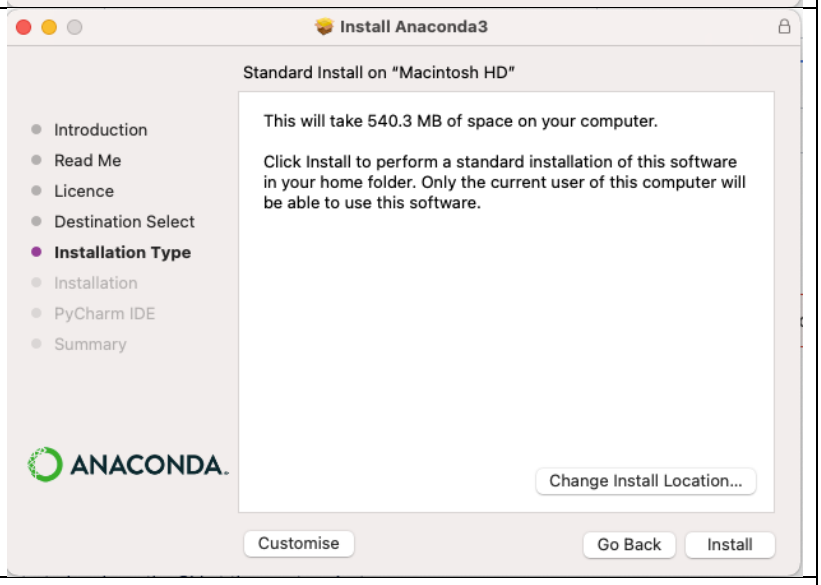
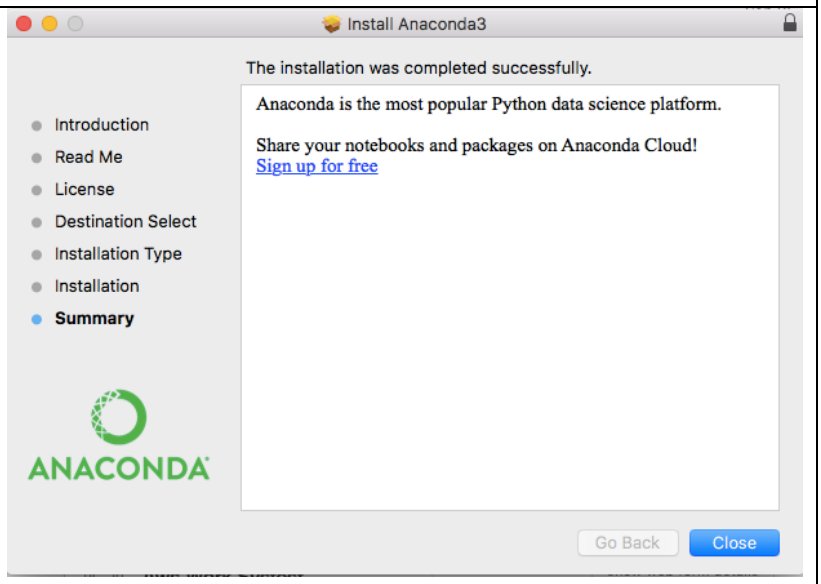
ADS Operator Control is a new CLI that allows to run Python Code on OCI essentially from everywhere, as well as test run code against ODSC Conda Environments outside of OCI. The main capabilities of the feature are:

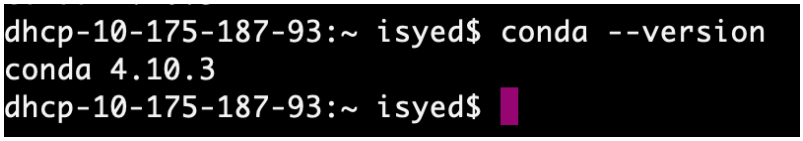
- Allows to test run code against the ODSC Conda Packages outside OCI
- Provides the ability to submit code to OCI Data Science Jobs with a single command line essentially from everywhere
- Streamlines the AI/ML End-to-End development process based on Oracle Data Science Cloud Service
- Helps manage machine learning environment dependencies and reproducibility

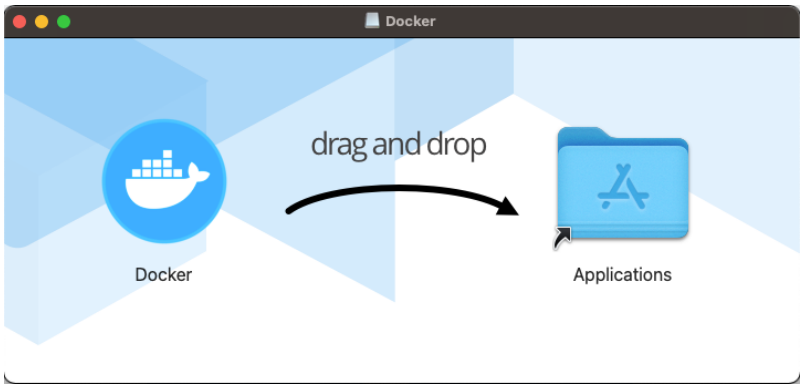
***Note** – This guide will cover implementation using MacOS.*

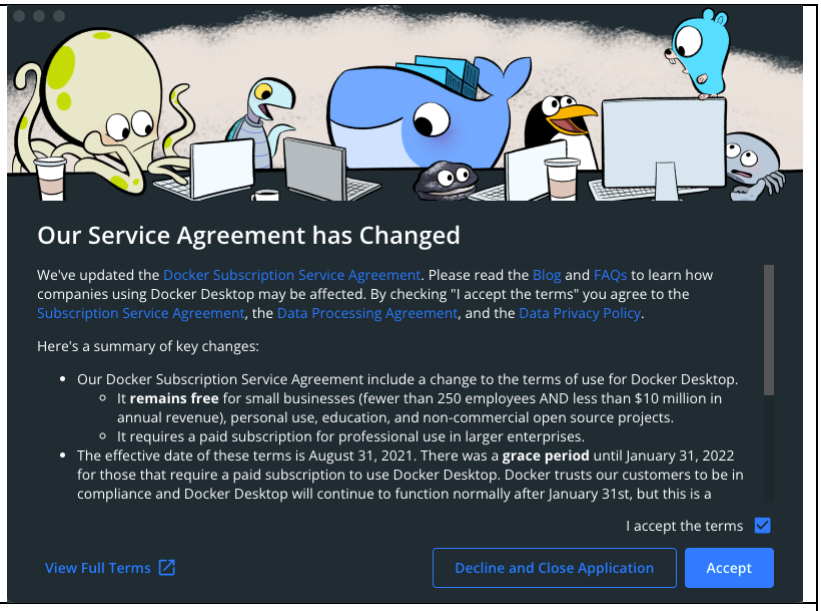
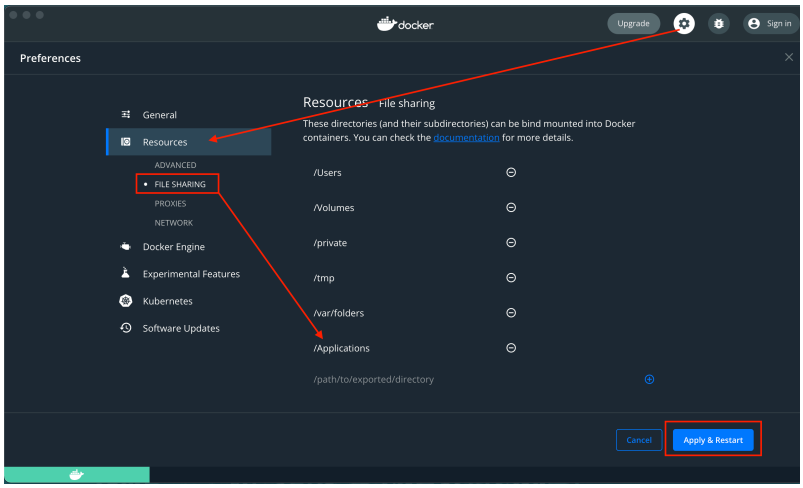


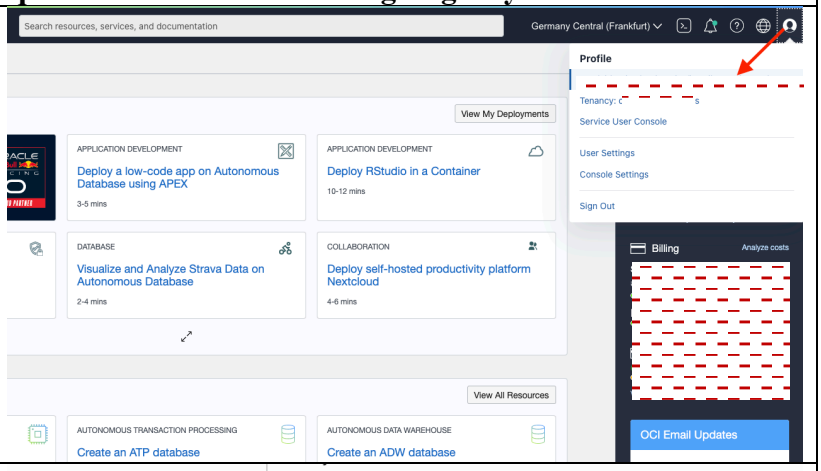
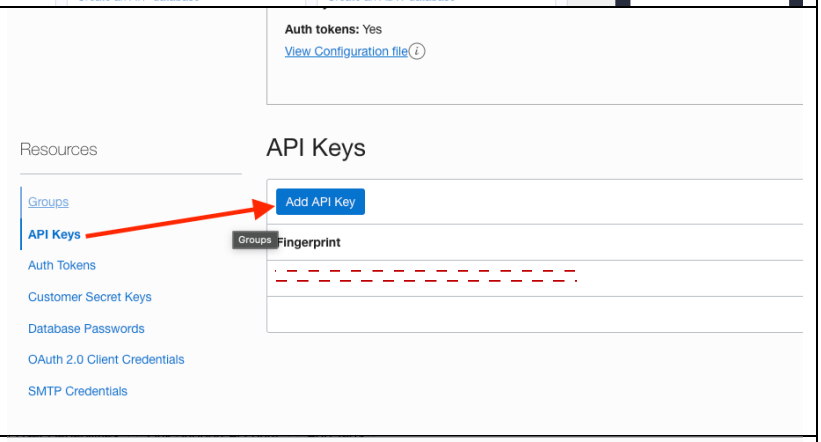
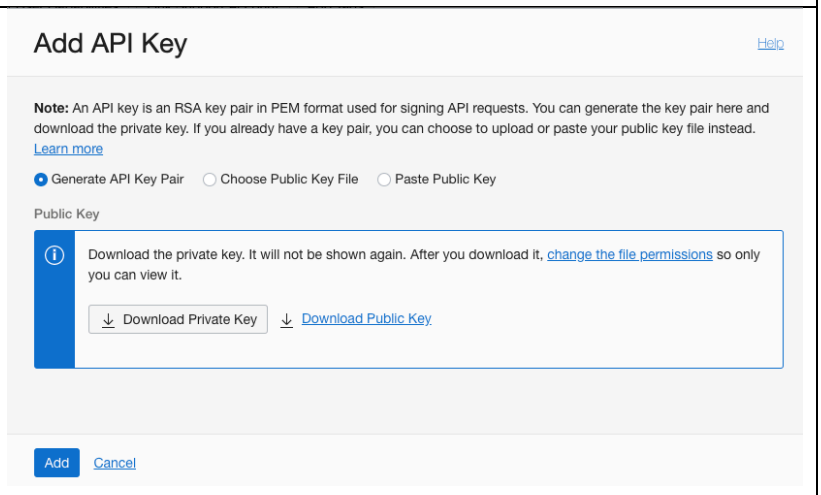
Pre-Requisite 1 – Install Conda	
Visit the Anaconda Website and download the Installer for your OS - https://www.anaconda.com/products/distribution	
Run the .pkg installer from your downloads folder. ***Double Click on File.*** ***Click Continue on Introduction.*** ***Click Continue on Read Me.***	
Accept Licence Agreement.	

Select Install for me only. ***Click Continue.***	
Click Install.	
You can ignore the installation of PyCharm IDE and Skip.	
Once Installed, ***Click Close.***	

You can verify installation of Anaconda, by opening a Terminal and Running: ***conda --version***	

Pre-Requisite 2 – Install Docker Desktop	
Visit the Docker Desktop Download Page and download the installer relevant to you OS: https://www.docker.com/products/docker-desktop/	
Once downloaded, double click on the ***Docker.dmg*** file downloaded. You can ***Drag and Drop*** the Docker Application Icon into you Applications Folder. This may take a couple minutes to execute the copy.	

Once copied across you can launch Docker. You will be asked to Accept the new Service Agreement.	
Creating, installing, and publishing conda packs involves mounting folders to docker images. Please make sure that you give docker desktop access to file sharing of the parent folder for conda. ***Visit Settings > Resources > File Sharing*** ***Add your parent folder for conda*** in my case /Applications**.** ***Click Apply & Restart.***	

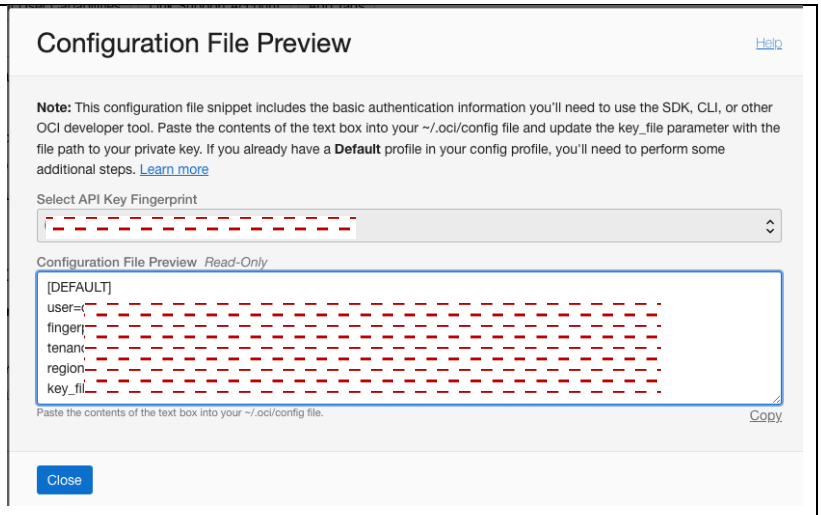
Pre-Requisite 3 – Generate API Signing Key	
Login to the OCI Console. Visit your Profile in the top right corner and select your username.	
Scroll down and select the API Keys Tab. Click on **Add API Key**.	
Select ***'Generate API Key Pair'*** ***Download*** both the ***Private and Public Key*** and keep them safe. Click on ***Add***.	

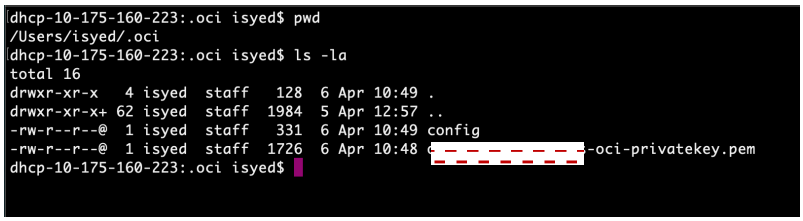
Copy and paste the content of the Configuration File Preview into a local file named '**config**'.

Make sure to update the **key_file** path to point to your private key file.

It is recommended to store both the config and private key file into the **~/.oci** directory.

These will be used to authenticate against OCI when downloading condas locally and submitting Data Science Jobs.



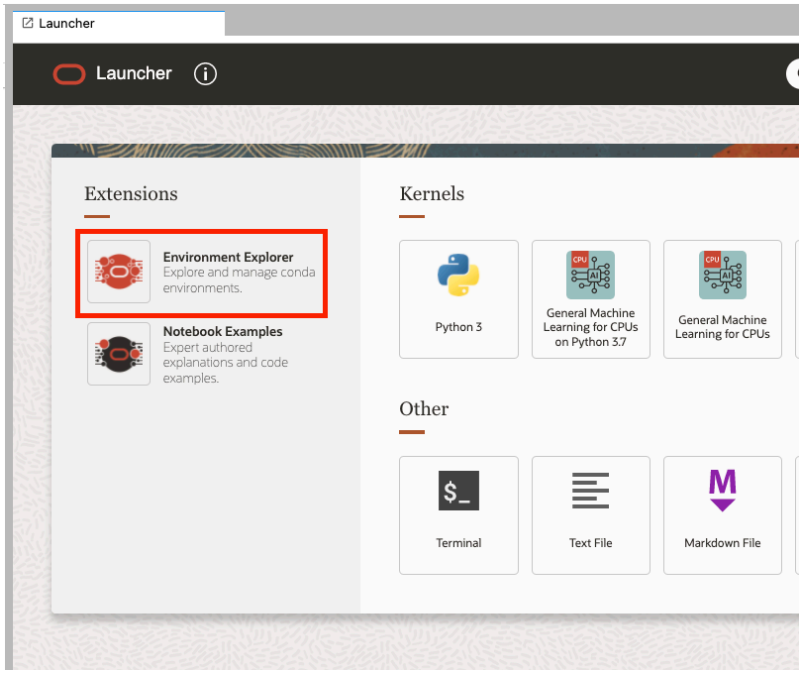


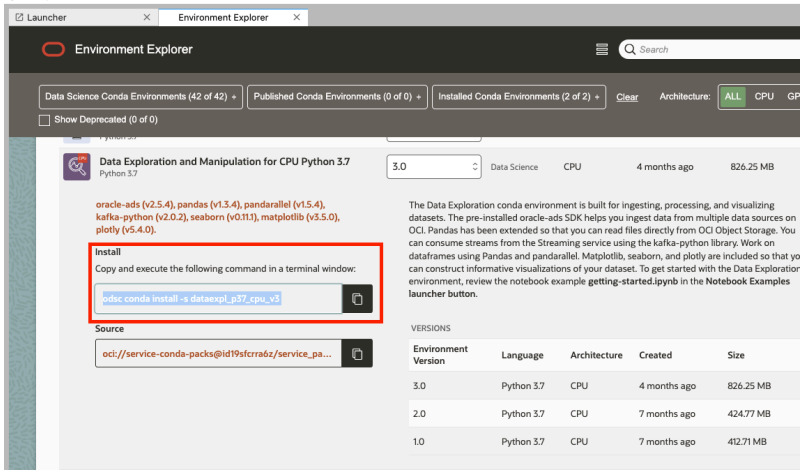
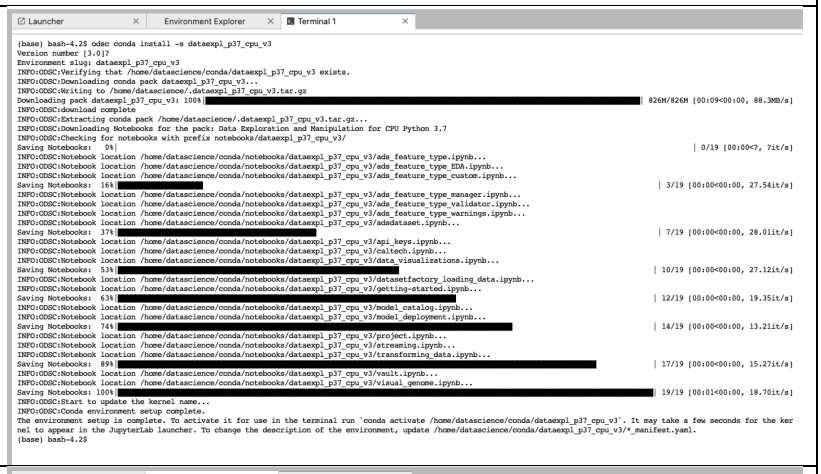
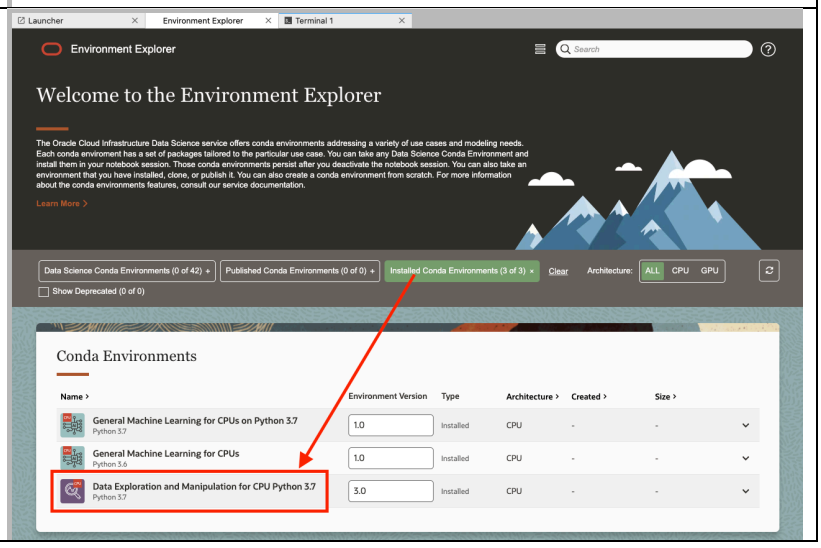
Publishing a OCI Data Science Conda to Object Storage

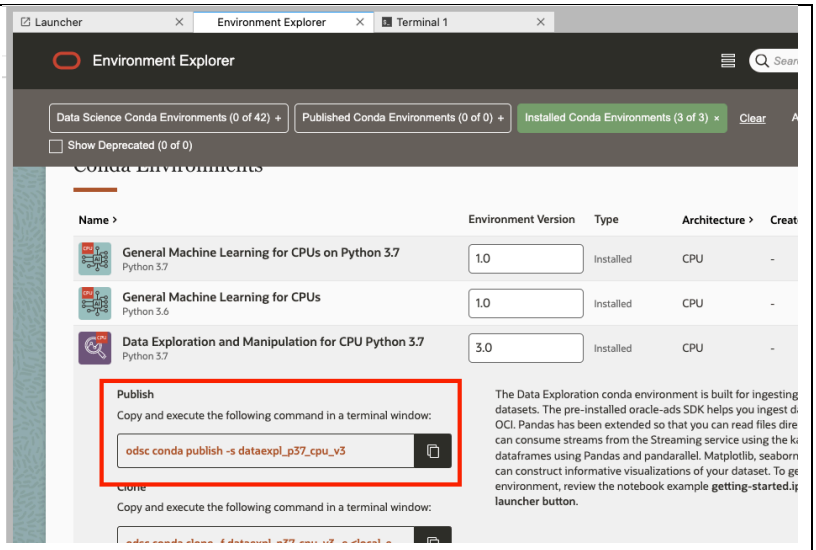
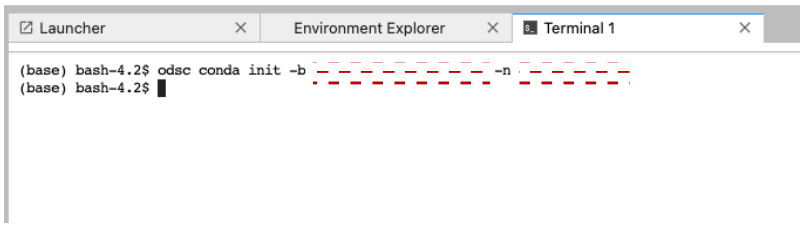
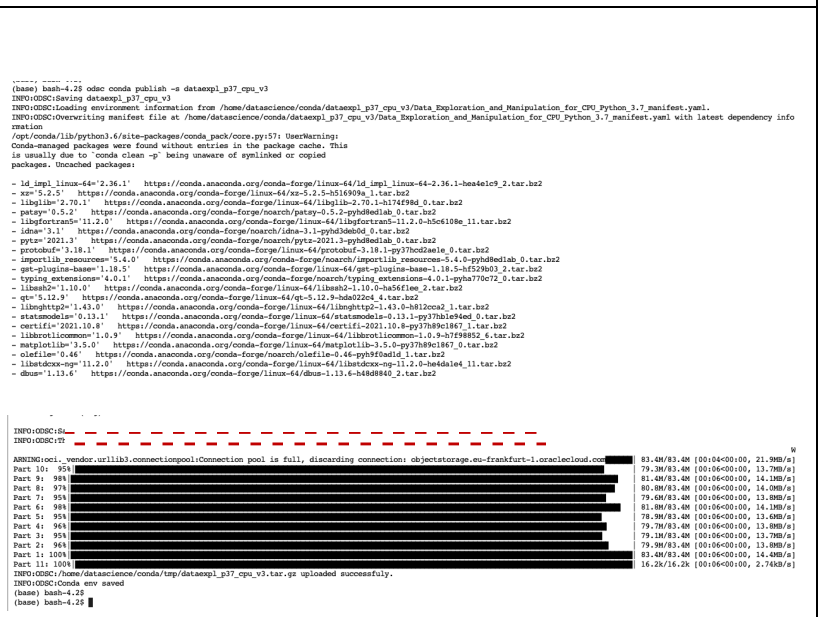
If we want to install our ADS Conda environments locally we must first publish them to the OCI Object Storage in order to then later download them locally.

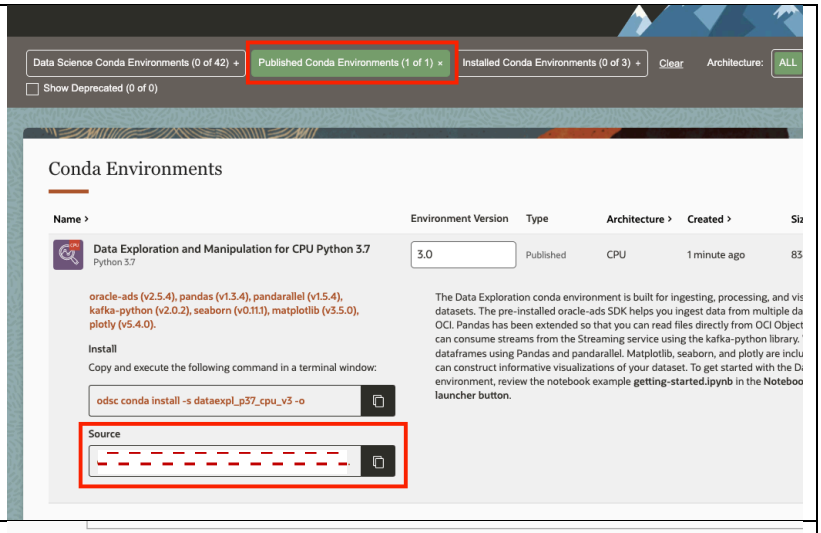
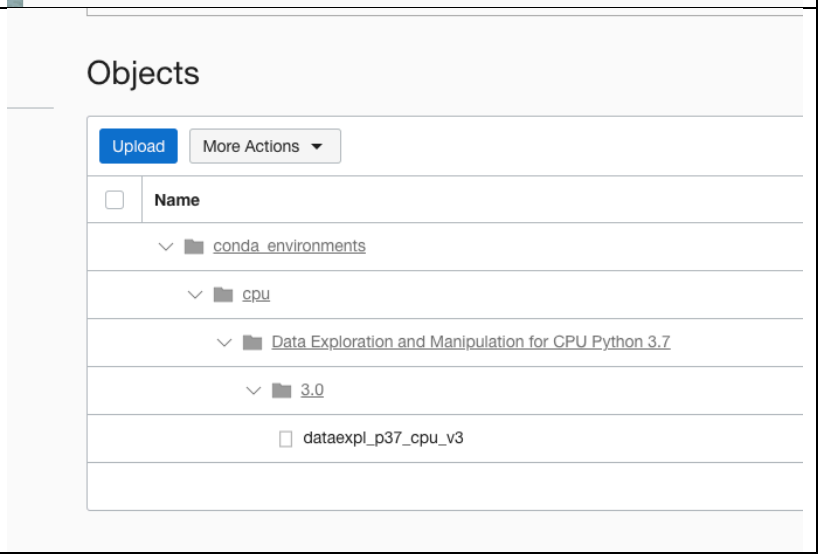
Open up your OCI Data Science Notebook Session and Navigate to the Launcher.

Select 'Environment Explorer'.

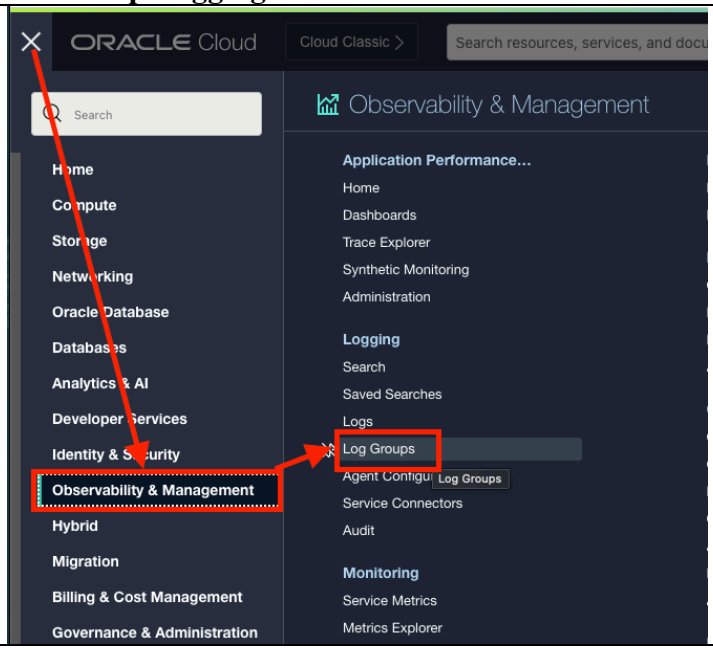


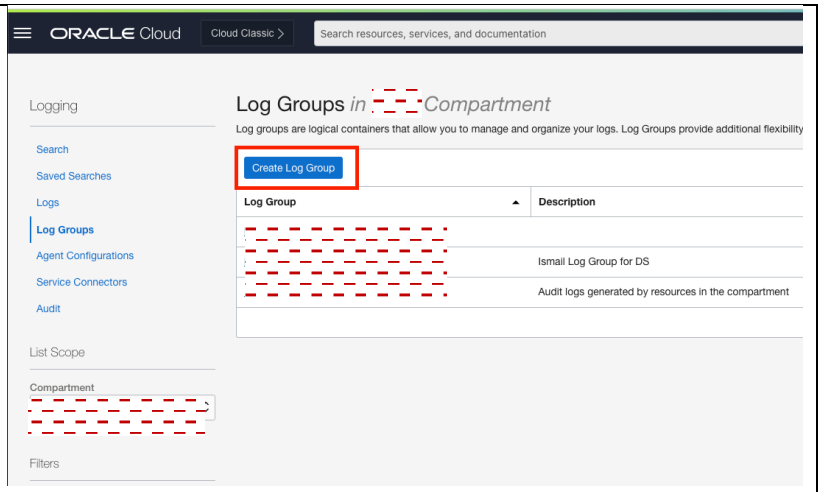
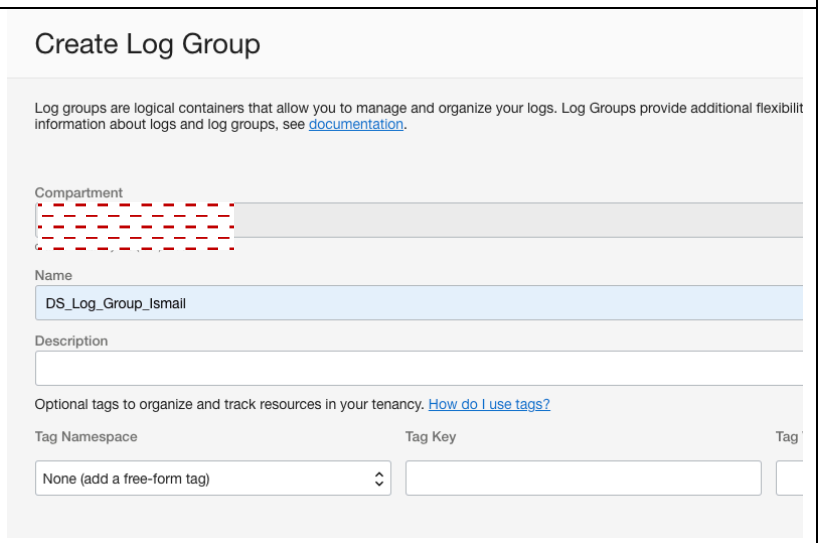
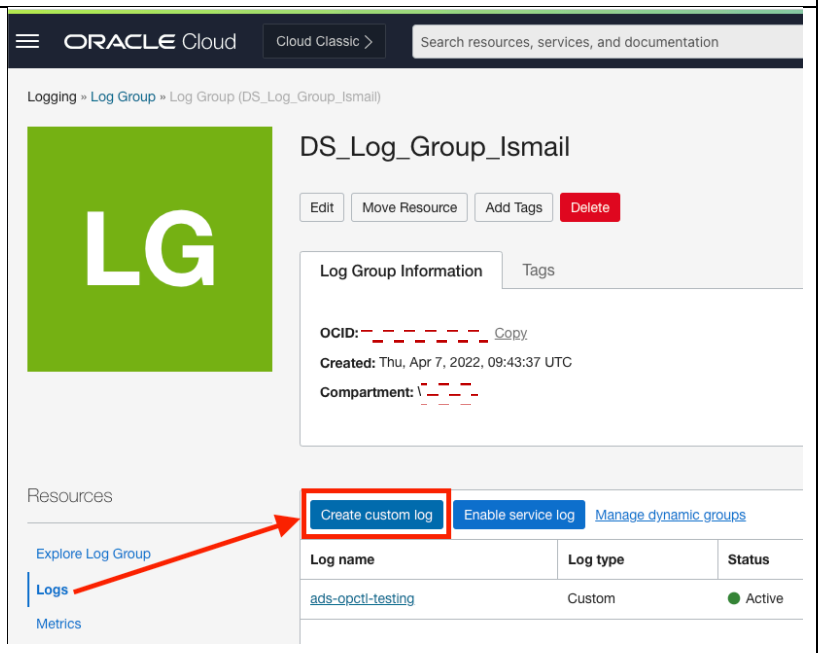
Navigate to a conda environment you would like to make available locally and ***copy*** the install command to first install it within the OCI Data Science Service. In my case I chose the ***'Data Exploration and Manipulation for CPU Python 3.7'*** conda.	
Open a Terminal window from the Launcher within the OCI Data Science Platform, **paste and run** the command from early. `odsc conda install -s dataexpl_p37_cpu_v3` This may take a bit of time to install all packages.	
Once installed, navigate back to the environment explorer, and ***click*** on the ***'Installed Conda Environments'*** Tab, you should now see your new conda listed.	

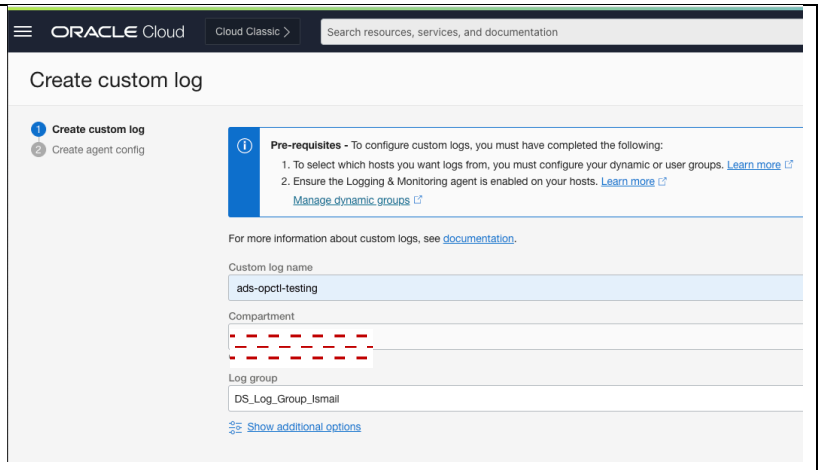
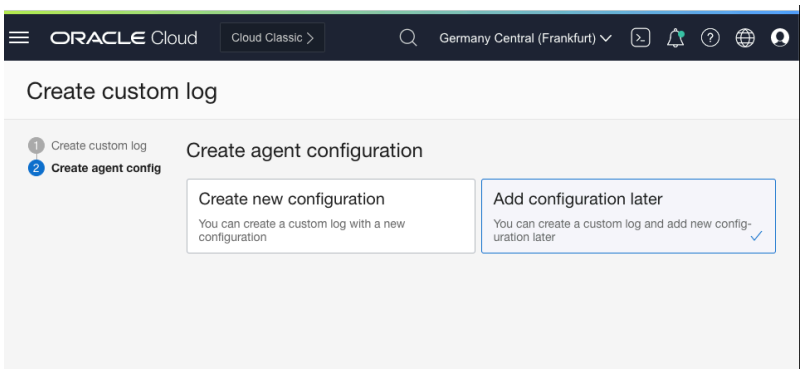
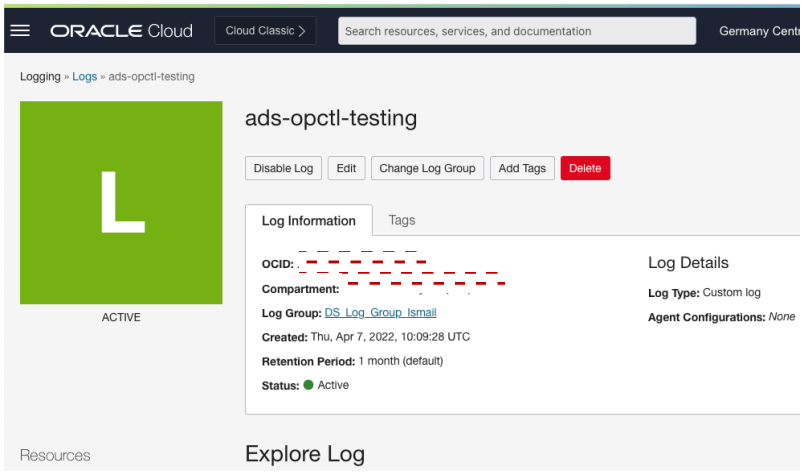
Select your newly installed conda and ***copy*** the ***Publish Command*** which we will use to push the conda environment to our Object Storage.	
Before we publish the conda we must first initialise our Object Storage within the Data Science Service. Within your Terminal in the Data Science Service ***enter*** and run the following: `odsc conda init -b <bucket_name> -n <bucket_name_space>`	
Once initialised, we can ***paste*** our ***publish conda command*** into the Terminal Window. `odsc conda publish -s dataexpl_p37_cpu_v3` Where -s is the conda slug name. This might take some time to save the content to OCI Object Storage	

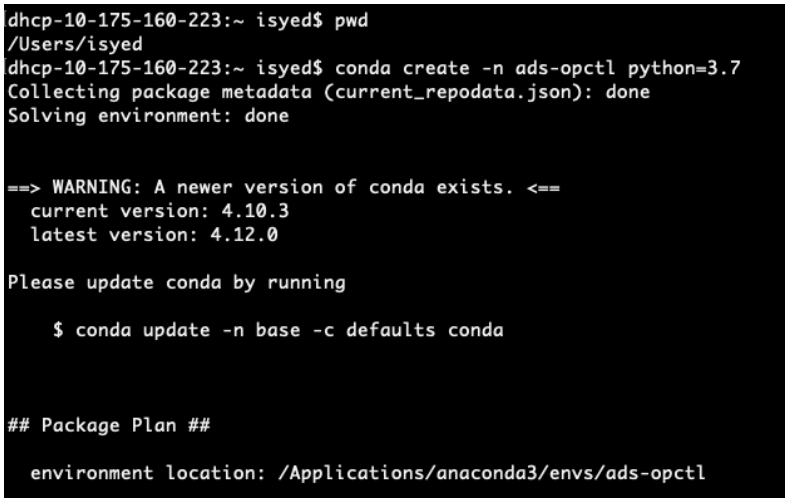
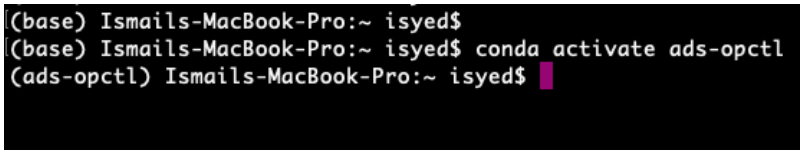
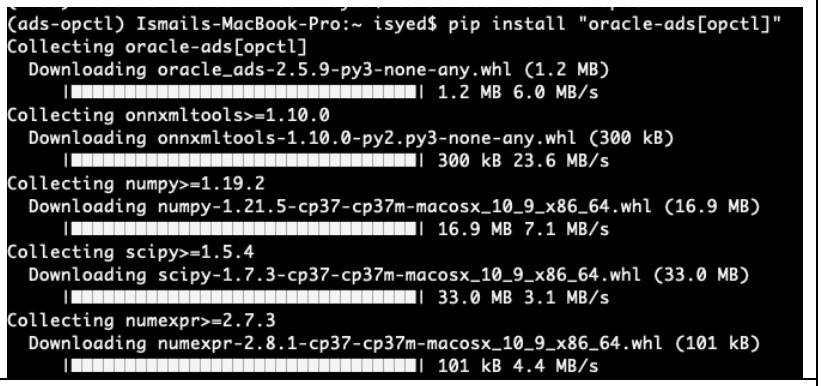
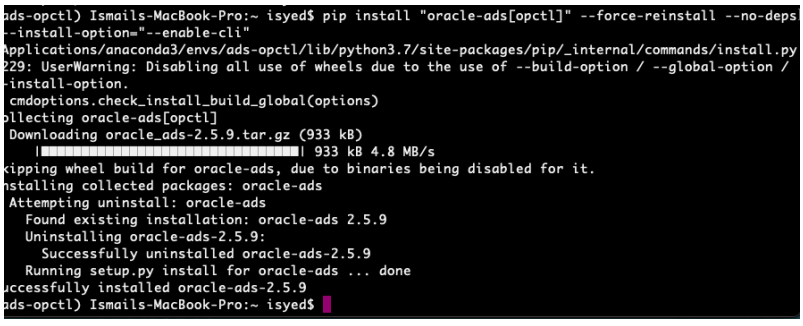
Once complete, we can re-visit the Environment Explorer and ***click*** on the ***'Published Conda Environments'*** Tab and view our published conda. We can also see the source is referenced to our Object Storage Location.	
Within the OCI Console, if we navigate to our Bucket that we initialised against the OCI Data Science Service, we should be able to see our conda environment saved.	

Set up Logging Service

When we come to later running jobs from our local machine within OCI Data Science Service, it is a good idea to set up logging in order to understand how the job has run. From the OCI Console ***visit Menu > Observability & Management > Log Groups***	

Click on ***'Create Log Group'*** to create a new log group to store our logs in.	
Enter a name for your Log Group and click ***'Create Log Group'*** down at the bottom.	
Once the Log Group is created ***make note*** of the ***Log Group OCID*** as we will need this later. Now, we will go ahead and create a custom log file to store our Job Run Outputs. Click on ***'Create Custom Log'***.	

Give the Log a ***Name***. ***Click 'Create Custom Log'*** at the bottom of the page.	
We will not add any custom configurations to our Log. ***Select 'Add Configuration Later'*** ***Click 'Create Custom Log'*** at the bottom of the page.	
Once created, ***make note*** of the ***Log OCID*** as we will need this later.	

Install Opctl	
Open a Terminal Window and create a new Conda Environment with at least Python 3.7. `conda create -n ads-opctl python=3.7` ***-n flag*** sets the name of the conda environment.	
We can now activate the conda. `conda activate ads-opctl`	
We will now install ADS with opctl within our conda environment. `pip install "oracle-ads[opctl]"` This will take a while to install all the packages.	
We will now enable the CLI. `pip install "oracle-ads[opctl]" --force-reinstall --no-deps --install-option="--enable-cli"`	

We can confirm the installation by running:

```
ads opctl -h
```

This will display the ads opctl help commands.

```
(ads-opctl) Ismails-MacBook-Pro:~ isyed$ ads opctl -h

^[[A
Usage: cli.py opctl [OPTIONS] COMMAND [ARGS]...

Options:
  -h, --help  Show this message and exit.

Commands:
  build-image
  cancel
  conda
  configure
  delete
  init-vscode
  publish-image
  run
  spark
  watch
(ads-opctl) Ismails-MacBook-Pro:~ isyed$
```

Next, we can set up the default ADS OPCTL config by running:

```
ads opctl configure
```

You will be prompted to enter the following info:

- OCI Config Path
- Default OCI Profile
- Conda Install Folder
- Conda Pack OS Prefix
- Compartment OCID
- Data Science Project OCID
- Data Science Subject OCID
- Job Run Shape
- Block Storage GB
- Log Group OCID
- Log OCID
- Docker Registry ID
- Conda Pack OS Prefix

Fill in the relevant information.

This will create two files in **/Users/<username>/.ads_ops**

```
(ads-opctl) Ismails-MacBook-Pro:~ isyed$ ads opctl configure
Folder to save ADS operators related configurations: [~/.ads_ops]:
OCI config path: [~/.oci/config]:
Default OCI profile: [DEFAULT]:
Conda pack install folder: [/Applications/anaconda3]:
Object storage Conda Env prefix, in the format oci://<bucket>@<namespace>/<path> []: oci://West_Isma
ilSyed_Conda@frqap2zhtzbe/
Configuration saved at /Users/isyed/.ads_ops/config.ini
==== Setting configuration for OCI Jobs ====
Do you want to set up or update OCI Jobs configuration? [Y/n]: Y
Do you want to set up or update for profile DEFAULT? [y/N]: y
Specify compartment_id: [redacted]
Specify project_id: [redacted]
Specify subnet_id: [redacted]
Specify shape_name: VM.Standard2.1
Specify block_storage_size_in_GBs: 50
Specify log_group_id []:
Specify log_id []:
Specify docker_registry []:
Specify conda_pack_os_prefix, in the format oci://<bucket>@<namespace>/<path> []: [redacted]
Configuration saved at /Users/isyed/.ads_ops/ml_job_config.ini
==== Setting configuration for OCI DataFlow ====
Do you want to set up or update OCI DataFlow configuration? [Y/n]: n
(ads-opctl) Ismails-MacBook-Pro:~ isyed$
```

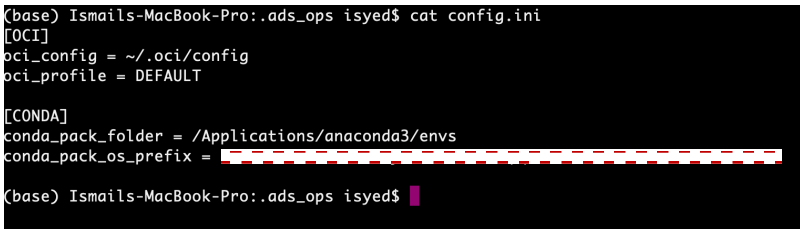
OCI Config Path = ~/.oci/config
Default OCI Profile = DEFAULT
Conda Install Folder = /Applications/anaconda/envs
Conda Pack OS Prefix = oci://<bucket_name>@<name_space>/<os_published_conda_folder>
Shape Name = VM.Standard2.1
Block Storage = 50

We can change directory to see the config files created.

```
cd /Users/<username>/.ads_ops
```

We can print out the first config file:

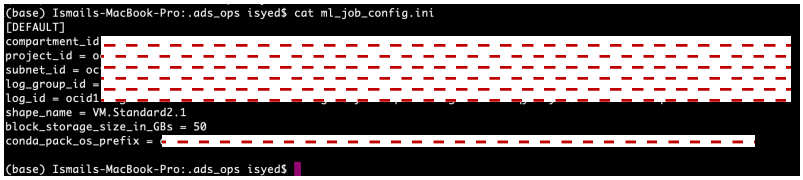
```
cat config.ini
```



We can print out the second config file:

```
cat ml_job_config.ini
```

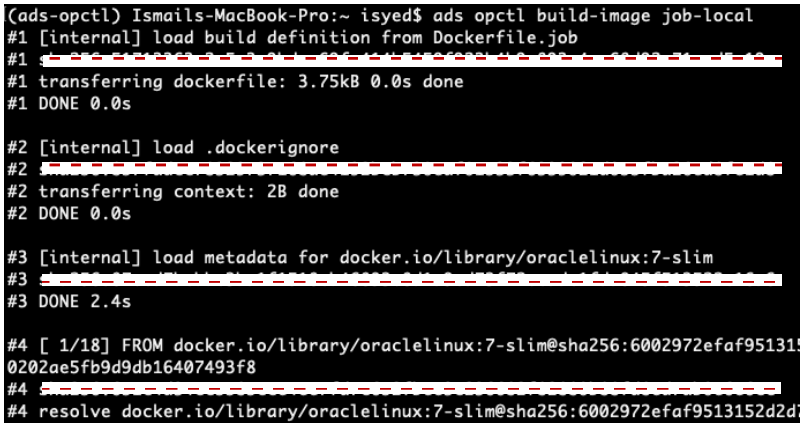
Remember you can update these files at any time, if any parameters change by using an editor such as vi.



For local development, including creating and publishing conda packs, a local docker image is necessary. The local docker image built by the following command approximately mirrors the actual Jobs image used by ML Jobs to run code with conda packs. The ***opctl*** commands later will run the docker image and execute the code within in the image and reference our local conda pack we install later.

Make sure your Docker Desktop Environment is up and running before executing the following:

```
ads opctl build-image job-local
```



This Docker build will take some time.

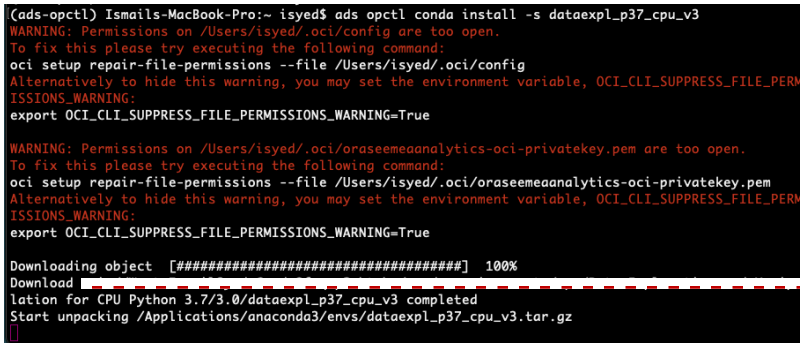
Once the docker build is complete we can now install our Published Conda Environments Locally.

```
ads opctl conda install -s dataexpl_p37_cpu_v3
```

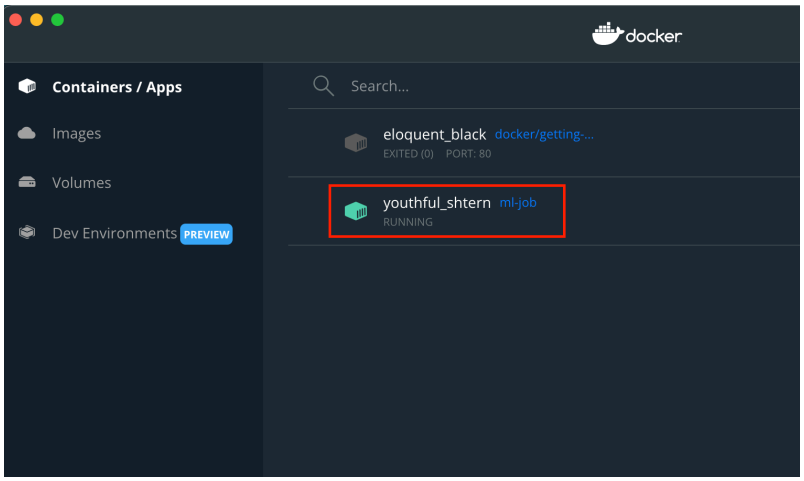
Where **-s** is the **slug name** of the conda environment published to the OCI Object Storage.

This command will use the parameters we previously defined in the oci config files, ***config.ini file and ml_jobs_config.ini file*****,** to reference the Object Storage, Authenticate, and where to save the Conda Env Locally.

This will take some time to download the conda and install the packages.



If you open your Docker Desktop while this is executing, you can see our ml-jobs Docker Image running in the background.



Once installed we can use the following command to list our local conda environments.

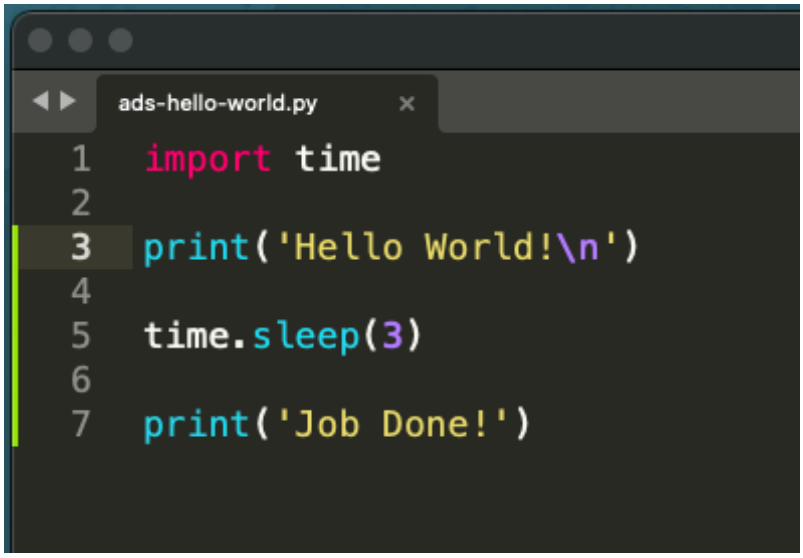
```
conda env list
```

```
(base) Ismails-MacBook-Pro:~ isyed$ conda env list
# conda environments:
#
base                  *  /Applications/anaconda3
ads-opctl                /Applications/anaconda3/envs/ads-opctl
dataexpl_p37_cpu_v3      /Applications/anaconda3/envs/dataexpl_p37_cpu_v3

(base) Ismails-MacBook-Pro:~ isyed$
```

Now everything is set up and configured, I can use an editor of my choice to write my program, referencing my newly installed conda environment.

For this demo, I have just used a text editor, nothing fancy to write a simple python program.



To run this locally I can use the following command:

```
ads opctl run -s <source-code-folder> -e <entry-script> -b local --conda-slug <slug-name> --cmd-args <-flag value> --env-var ENV_NAME=value
```

```
(ads-opctl) Ismails-MacBook-Pro:Downloads isyed$ ads opctl run -s ~/Downloads -e ads-hello-world.py
-b local --conda-slug dataexpl_p37_cpu_v3
Hello World!

Job Done!
(ads-opctl) Ismails-MacBook-Pro:Downloads isyed$
```

In my example I ran:

```
ads opctl run -s ~/Downloads -e ads-hello-world.py
-b local --conda-slug dataexpl_p37_cpu_v3
```

Notice the **flag -b** runs this code **locally** against our conda slug.

Alternatively, I can push this script to the OCI Data Science Service as a Job.

Notice the main difference is to change the **-b flag** from **local** to **job**.

We can also add an extra **flag --job-name <name>** to specify a friendly job name in the OCI Console.

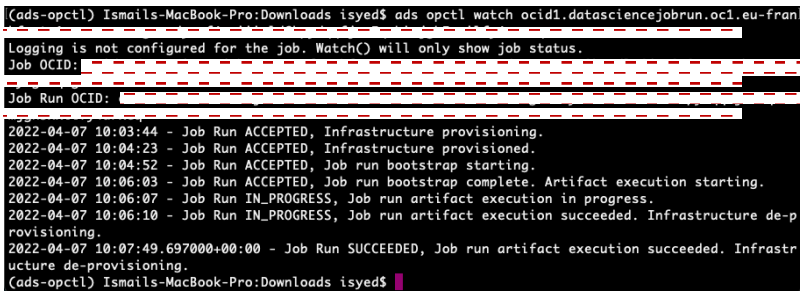
```
(ads-opctl) Ismails-MacBook-Pro:Downloads isyed$ ads opctl run -s ~/Downloads -e ads-hello-world.py
-b job --conda-slug dataexpl_p37_cpu_v3
JOB OCID: [redacted]
JOB RUN OCID: [redacted]
```

```
ads opctl run -s ~/Downloads -e ads-hello-world.py
-b job --conda-slug dataexpl_p37_cpu_v3 --job-name
test-conda-pack
```

While the Job is executing, we can run the following command to track the progress of the execution real time:

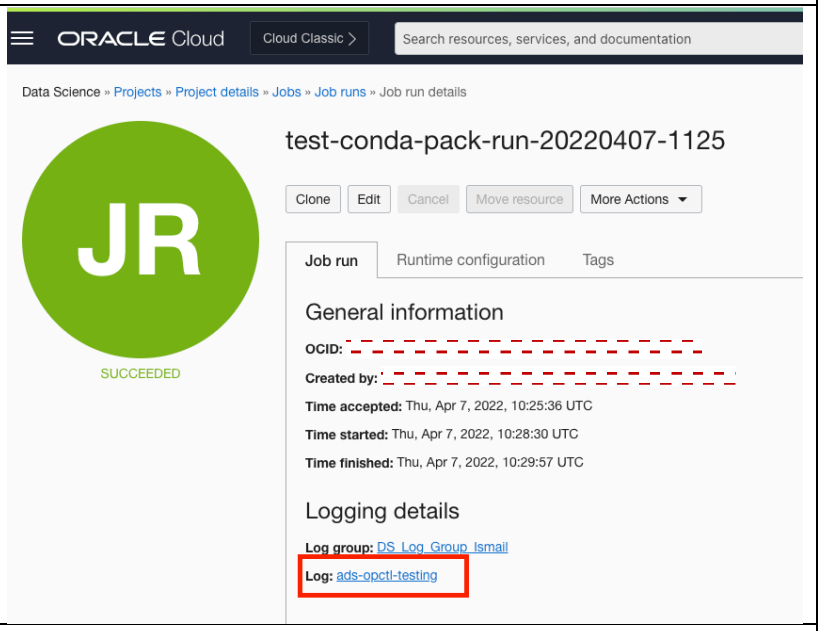
```
ads opctl watch <run-ocid>
```

The Run OCID will be displayed as an output of running the previous `ads opctl run` command.

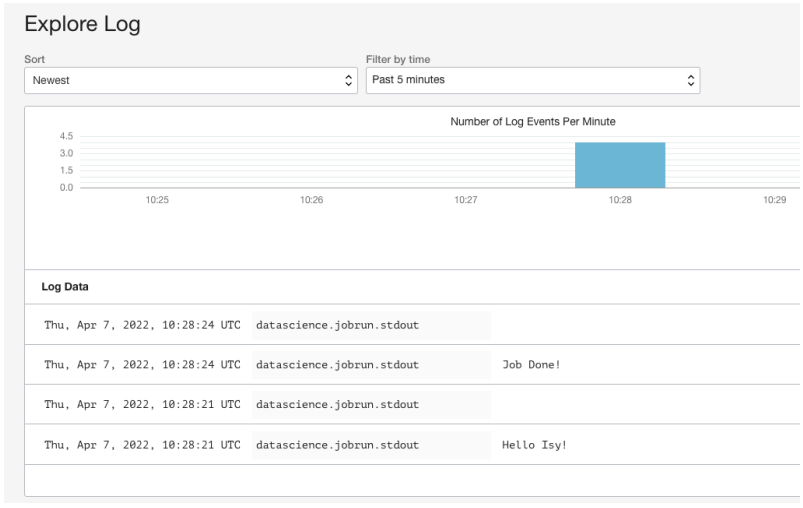


If we visit our Jobs Page under the OCI Data Science Service, we can view the Job that has just executed and succeeded.

As we configured a Logging Group and Log within our configuration files, we can use the hyperlink to visit the Log File.



Here we can see the outputs from the executed Python Script.



If you want to you can write scripts that take in command line arguments, and we can pass this into our ***ads opctl*** command when submitted a job.

ads-hello-world-args.py

```python
import time
import argparse

# Read in command line argument
parser = argparse.ArgumentParser()
parser.add_argument('-g', '--greeting', required=False, default='Mystery Person')
args = parser.parse_args()


print(f'Hello {args.greeting}!\n')

time.sleep(3)

print('Job Done!')
```

Here is an example of passing through some command line arguments:

```
ads opctl run -s ~/Downloads -e ads-hello-world-args.py -b job --conda-slug dataexpl_p37_cpu_v3 --cmd-args '-g Isy' --job-name test-conda-pack
```

```
(ads-opctl) Ismails-MacBook-Pro:~ isyed$ ads opctl run -s ~/Downloads -e ads-hello-world-args.py -b
job --conda-slug dataexpl_p37_cpu_v3 --cmd-args '-g Isy' --job-name test-conda-pack
JOB OCID: [redacted]

JOB RUN OCID: [redacted]

(ads-opctl) Ismails-MacBook-Pro:~ isyed$ ads opctl watch ocid1.datasciencejobrun.oc1.eu-frankfurt-1.
[redacted]
Job OCID: [redacted]
kybzsq4nkq
Job Run OCID: [redacted]

2022-04-07 10:26:04 - Job Run ACCEPTED, Infrastructure provisioning.
2022-04-07 10:27:03 - Job Run ACCEPTED, Infrastructure provisioned.
2022-04-07 10:27:13 - Job Run ACCEPTED, Job run bootstrap starting.
2022-04-07 10:28:24 - Job Run ACCEPTED, Job run bootstrap complete. Artifact execution starting.
2022-04-07 10:28:31 - Job Run IN_PROGRESS, Job run artifact execution in progress.
2022-04-07 10:28:21 -
2022-04-07 10:28:21 - Hello Isy!
2022-04-07 10:28:24 -
2022-04-07 10:28:24 - Job Done!
```

Here is also an example of a script reading in an environment variable.

ads-hello-world.py

```python
import os
import time

print('Hello World!\n')

time.sleep(3)

print(f'Hello {os.environ.get('TEST_NAME', 'UNKNOWN')}')

time.sleep(3)

print('Job Done!')
```