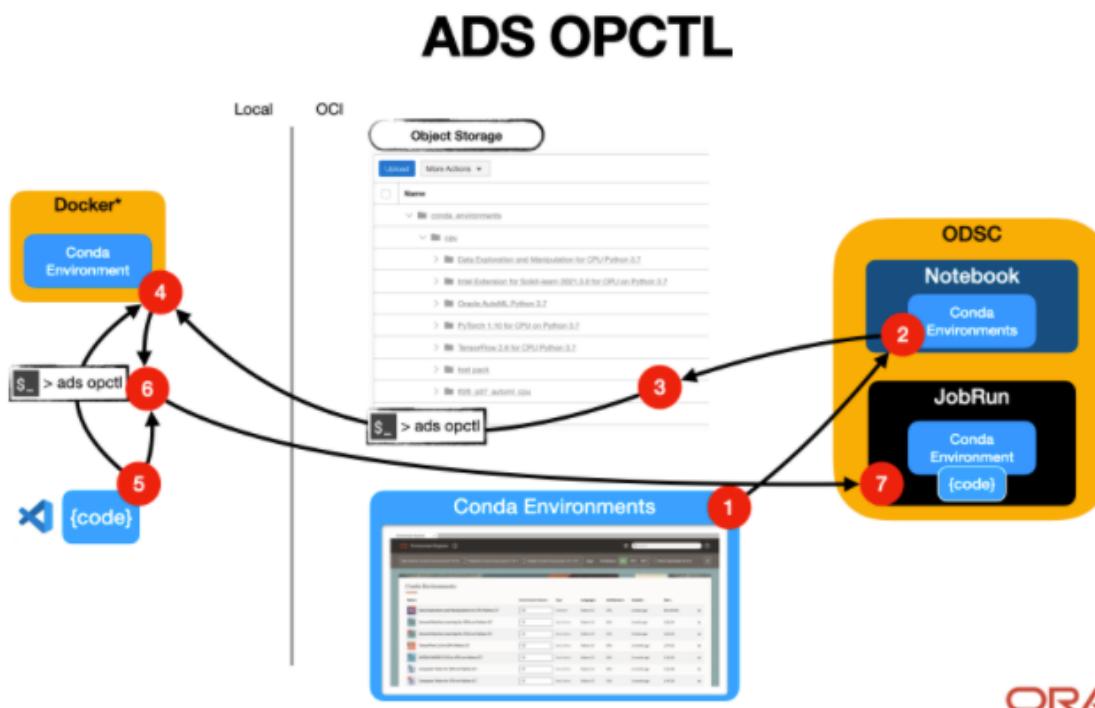


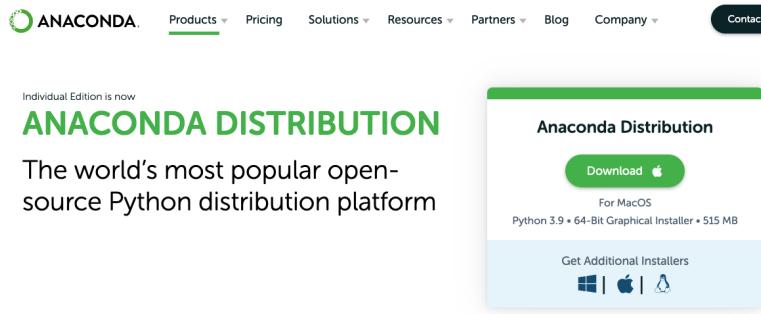
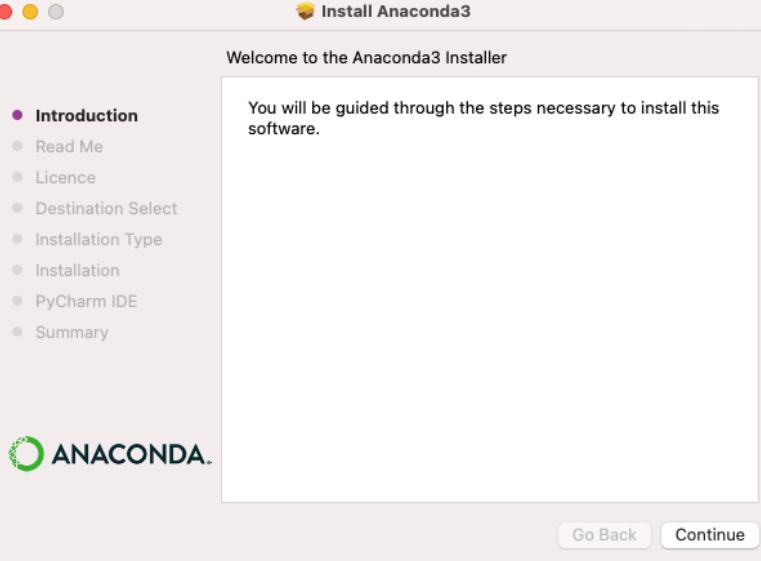
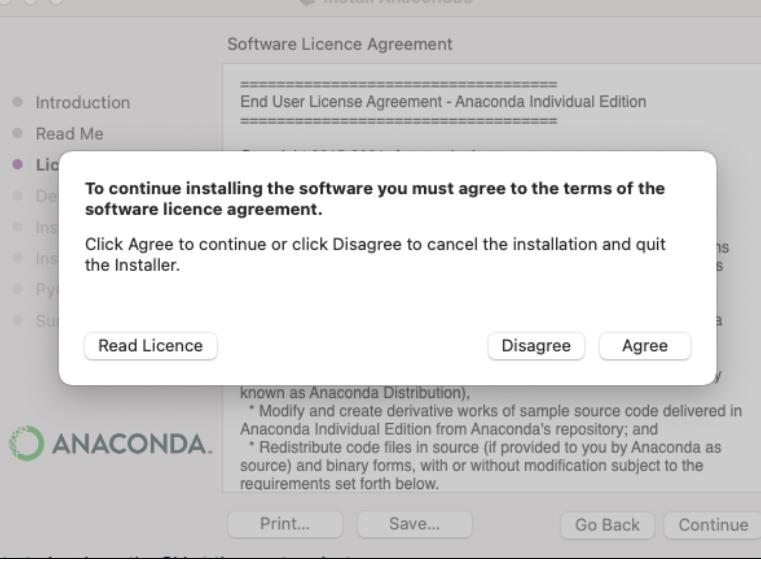
ADS OPTCL Jobs

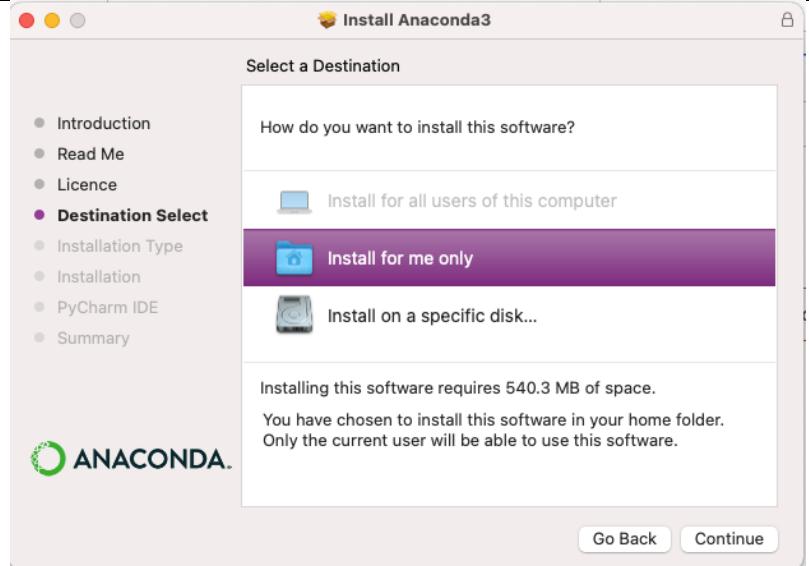
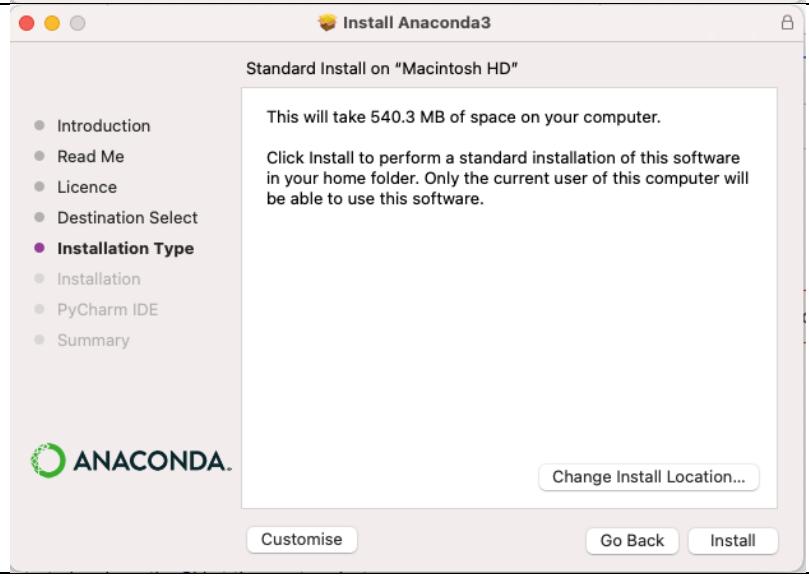
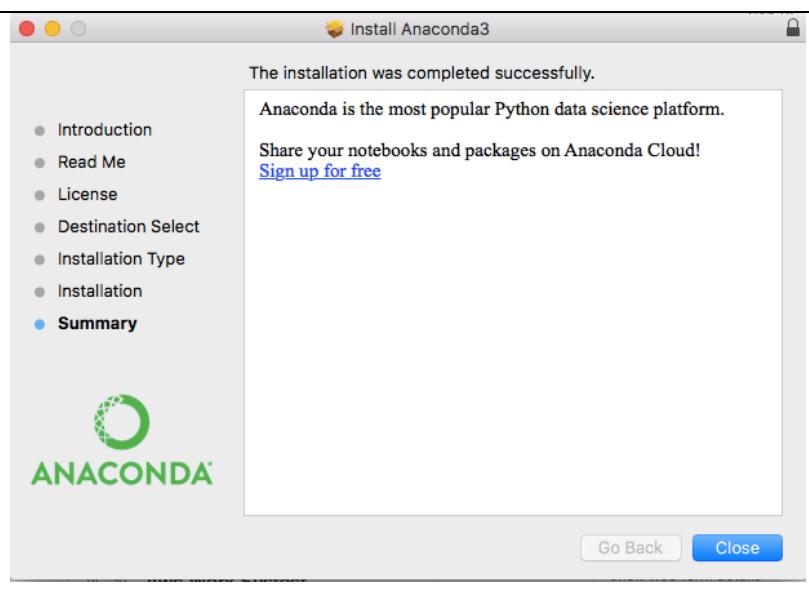
ADS Operator Control is a new CLI that allows to run Python Code on OCI essentially from everywhere, as well as test run code against ODSC Conda Environments outside of OCI. The main capabilities of the feature are:

- 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	 <p>The website shows the Anaconda Distribution logo and a prominent green "Download" button for Mac OS. Below it, there's a link to "Get Additional Installers" with icons for Windows, Mac, and Linux.</p>
Run the .pkg installer from your downloads folder. Double Click on File. Click Continue on Introduction. Click Continue on Read Me.	 <p>The installer window title is "Install Anaconda3". It says "Welcome to the Anaconda3 Installer" and "You will be guided through the steps necessary to install this software." On the left, a navigation menu lists steps: Introduction (selected), Read Me, Licence, Destination Select, Installation Type, Installation, PyCharm IDE, and Summary. At the bottom are "Go Back" and "Continue" buttons.</p>
Accept Licence Agreement.	 <p>The installer window title is "Install Anaconda3". It says "Software Licence Agreement" and displays the "End User License Agreement - Anaconda Individual Edition". A message box in the center states: "To continue installing the software you must agree to the terms of the software licence agreement." It also says "Click Agree to continue or click Disagree to cancel the installation and quit the Installer." Below the message are "Read Licence", "Disagree", and "Agree" buttons. At the bottom are "Print...", "Save...", "Go Back", and "Continue" buttons.</p>

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

You can verify installation of Anaconda, by opening a Terminal and Running:

conda --version

```
$ conda --version  
conda 4.10.3  
$
```

Visit the Docker Desktop Download Page and download the installer relevant to your OS:
<https://www.docker.com/products/docker-desktop/>

Docker Desktop

Install Docker Desktop – the fastest way to containerize applications.

Mac with Intel Chip

Mac with Apple Chip

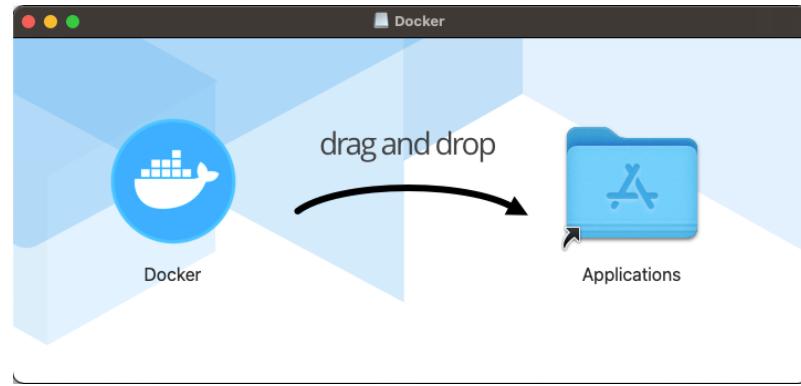
MOST COMMON

Also available for Windows and Linux

Once downloaded, double click on the **Docker.dmg** file downloaded.

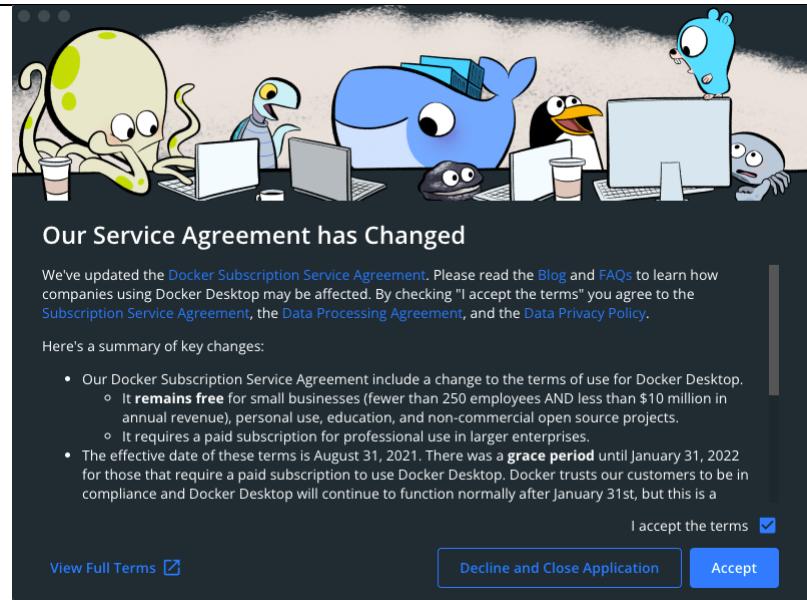
You can **Drag and Drop** the Docker Application Icon into your Applications Folder.

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.

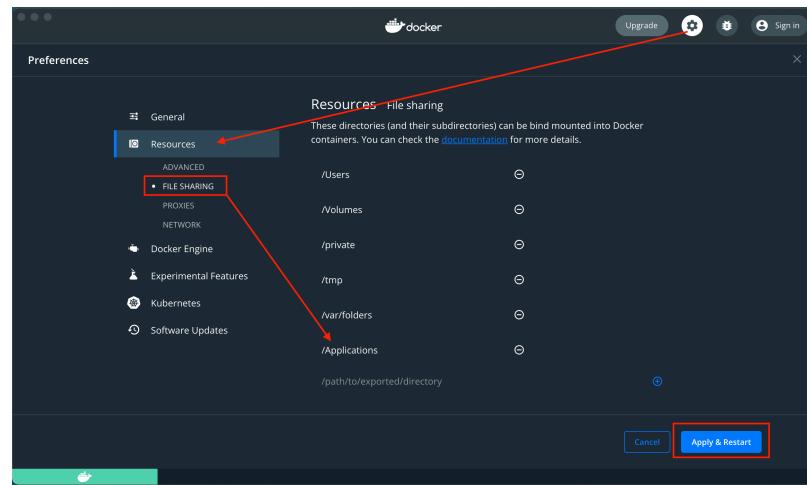


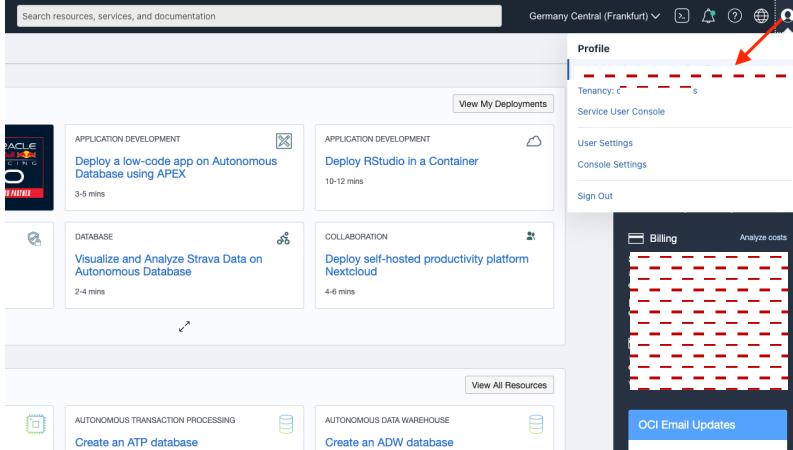
Creating, installing, and publishing conda packs involves mounting folders to docker images. Please make sure that you give docker desktop access to file sharing of the parent folder for conda.

Visit Settings > Resources > File Sharing

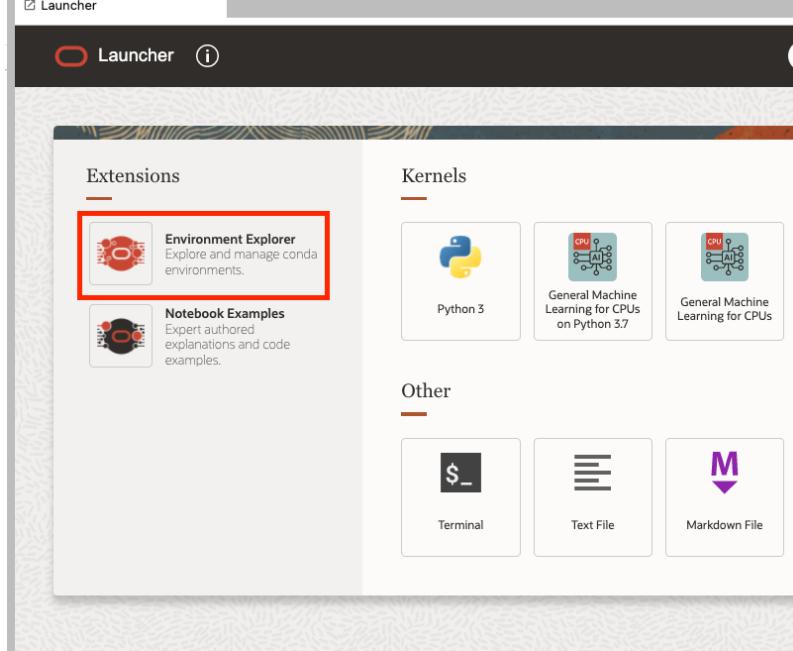
Add your parent folder for conda in my case /Applications.

Click Apply & Restart.



Pre-Requisite 3 – Generate API Signing Key					
<p>Login to the OCI Console.</p> <p>Visit your Profile in the top right corner and select your username.</p>					
<p>Scroll down and select the API Keys Tab.</p> <p>Click on Add API Key.</p>	<p>Auth tokens: Yes View Configuration file</p> <table> <thead> <tr> <th>Resources</th> <th>API Keys</th> </tr> </thead> <tbody> <tr> <td> Groups API Keys → Auth Tokens Customer Secret Keys Database Passwords OAuth 2.0 Client Credentials SMTP Credentials </td> <td> Add API Key → Groups Fingerprint </td> </tr> </tbody> </table>	Resources	API Keys	Groups API Keys → Auth Tokens Customer Secret Keys Database Passwords OAuth 2.0 Client Credentials SMTP Credentials	Add API Key → Groups Fingerprint
Resources	API Keys				
Groups API Keys → Auth Tokens Customer Secret Keys Database Passwords OAuth 2.0 Client Credentials SMTP Credentials	Add API Key → Groups Fingerprint				
<p>Select 'Generate API Key Pair'</p> <p>Download both the Private and Public Key and keep them safe.</p> <p>Click on Add.</p>	<p>Add API Key Help</p> <p>Note: An API key is an RSA key pair in PEM format used for signing API requests. You can generate the key pair here and download the private key. If you already have a key pair, you can choose to upload or paste your public key file instead. Learn more</p> <p><input checked="" type="radio"/> Generate API Key Pair <input type="radio"/> Choose Public Key File <input type="radio"/> Paste Public Key</p> <p>Public Key</p> <div style="border: 1px solid #ccc; padding: 5px;"> <p>(i) Download the private key. It will not be shown again. After you download it, change the file permissions so only you can view it.</p> <p>Download Private Key Download Public Key</p> </div> <p>Add Cancel</p>				

<p>Copy and paste the content of the Configuration File Preview into a local file named 'config'.</p> <p>Make sure to update the key_file path to point to your private key file.</p> <p>It is recommended to store both the config and private key file into the <code>~/.oci</code> directory.</p> <p>These will be used to authenticate against OCI when downloading condas locally and submitting Data Science Jobs.</p>	<h3>Configuration File Preview</h3> <p>Help</p> <p>Note: This configuration file snippet includes the basic authentication information you'll need to use the SDK, CLI, or other OCI developer tool. Paste the contents of the text box into your <code>~/.oci/config</code> file and update the <code>key_file</code> parameter with the file path to your private key. If you already have a Default profile in your config profile, you'll need to perform some additional steps. Learn more</p> <p>Select API Key Fingerprint <input type="text" value="-----"/></p> <p>Configuration File Preview Read-Only</p> <pre>[DEFAULT] user=- finger= tenant= region= key_file=-----</pre> <p>Paste the contents of the text box into your <code>~/.oci/config</code> file. Copy</p> <p>Close</p> <pre>\$ pwd /oci \$ ls -la total 16 drwxr-xr-x 4 staff 128 6 Apr 10:49 . drwxr-xr-x+ 62 staff 1984 5 Apr 12:57 .. -rw-r--r--@ 1 staff 331 6 Apr 10:49 config -rw-r--r--@ 1 staff 1726 6 Apr 10:48 -----oci-privatekey.pem</pre>
--	---

<h3>Publishing a OCI Data Science Conda to Object Storage</h3>	
<p>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.</p> <p>Open up your OCI Data Science Notebook Session and Navigate to the Launcher.</p> <p>Select 'Environment Explorer'.</p>	 <p>The screenshot shows the OCI Data Science Notebook Launcher interface. The 'Extensions' section is active, displaying two items: 'Environment Explorer' (selected) and 'Notebook Examples'. The 'Kernels' section shows icons for Python 3, General Machine Learning for CPUs on Python 3.7, and General Machine Learning for CPUs. The 'Other' section shows icons for Terminal, Text File, and Markdown File.</p>

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.

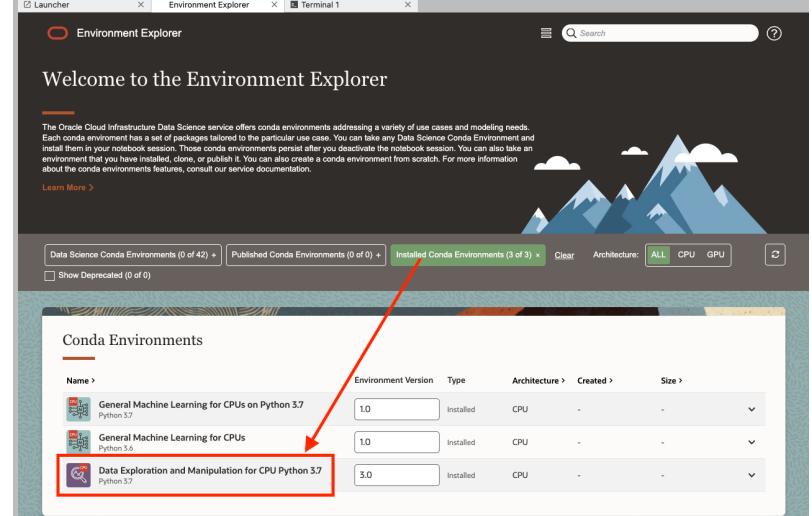
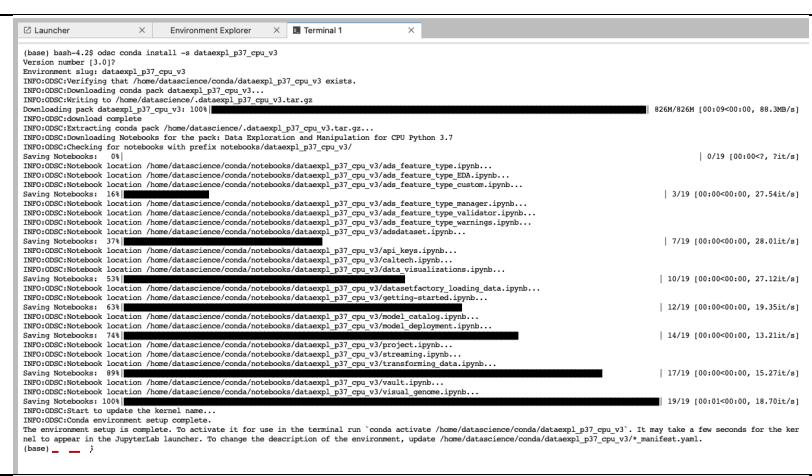
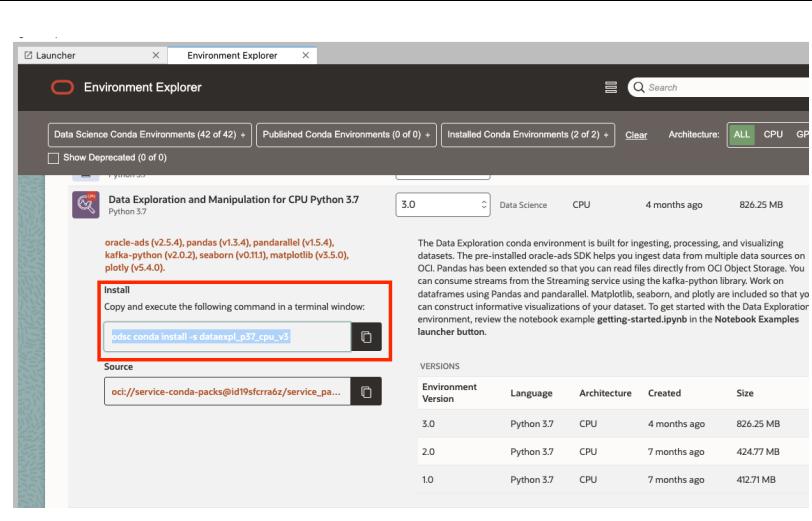
In my case I chose the '*Data Exploration and Manipulation for CPU Python 3.7*' conda.

Open a Terminal window from the Launcher within the OCI Data Science Platform, **paste and run** the command from early.

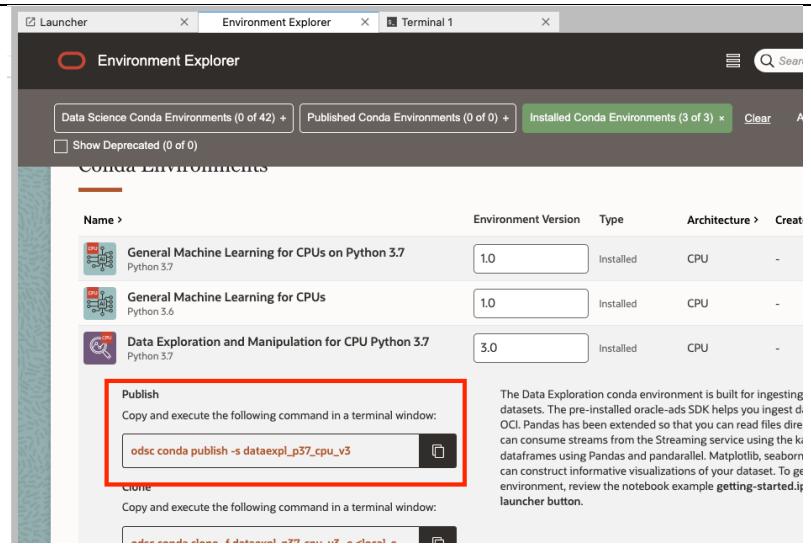
```
odsc conda install -s  
dataexpl_p37_cpu_v3
```

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:

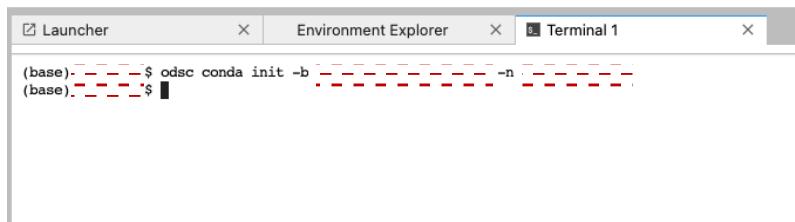
```
odsc conda init -b
<bucket_name> -n
<bucket_name_space>
```

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



```
(base) [REDACTED] odsc conda publish -s dataexpl_p37_cpu_v3
INFO:ODSC:Loading manifest information from /home/datascience/conda/condaexpl_p37_cpu_v3/Data_Exploration_and_Manipulation_for_CPU_Python_3.7.manifest.yaml.
INFO:ODSC:Overwriting manifest file at /home/datascience/conda/condaexpl_p37_cpu_v3/Data_Exploration_and_Manipulation_for_CPU_Python_3.7.manifest.yaml with latest dependency information
(base) [REDACTED] python /opt/odsc/lib/python3.6/site-packages/conda/pack/cone.py:57: UserWarning:
Conda-managed packages were found without entries in the package cache. This is usually due to 'conda clean -p' being unaware of symlinked or copied packages. Uncached packages:
- id_iml_linux-64_2.36.1 https://conda.anaconda.org/conda-forge/linux-64/x-5.0.3-b318999_1.tar.bz2
- id_ml_2.70.1 https://conda.anaconda.org/conda-forge/linux-64/l19p12z-70.1-517498d_0.tar.bz2
- patay_0.5.2 https://conda.anaconda.org/conda-forge/noarch/patay-0.5.2-pyhd8ed1ab_0.tar.bz2
- libzfpformat_1.12.2 https://conda.anaconda.org/conda-forge/noarch/libzfpformat-1.12.2.0-h5c6108e_11.tar.bz2
- id_1 https://conda.anaconda.org/conda-forge/noarch/1de-1.1-pyhd8ed1ab_0.tar.bz2
- ptyz_2021.3 https://conda.anaconda.org/conda-forge/noarch/ptyz-2021.3-pyhd8ed1ab_0.tar.bz2
- protocol_1 https://conda.anaconda.org/conda-forge/linux-64/protocol-3.18.1-pyhd8ed1ab_0.tar.bz2
- impala_resources_1 https://conda.anaconda.org/conda-forge/noarch/impala-resources-5.4-pyhd8ed1ab_0.tar.bz2
- pytz_2021.3 https://conda.anaconda.org/conda-forge/linux-64/pytz-2021.3-pyhd8ed1ab_0.tar.bz2
- typing_extensions_4.4.1 https://conda.anaconda.org/conda-forge/search/typing_extensions-4.4.1-pyha770c72_0.tar.bz2
- libzfpformat_1.12.2 https://conda.anaconda.org/conda-forge/noarch/libzfpformat-1.12.2.0-h5c6108e_11.tar.bz2
- olefile_0.46 https://conda.anaconda.org/conda-forge/linux-64/qt/ qt-12.9-hbd022d4_4.tar.bz2
- libhttp2_1.43.0 https://conda.anaconda.org/conda-forge/linux-64/libhttp2-1.43.0-h79891867_1.tar.bz2
- fastavif_2021.10.8 https://conda.anaconda.org/conda-forge/linux-64/fastavif-2021.10.8-py779891867_1.tar.bz2
- libssl1.1.1 https://conda.anaconda.org/conda-forge/linux-64/libssl1.1.1-h79891867_1.tar.bz2
- libcurl10_1.0.0 https://conda.anaconda.org/conda-forge/linux-64/libcurl10-1.0.0-h79891867_1.tar.bz2
- olefile_0.46 https://conda.anaconda.org/conda-forge/linux-64/qt/olefile-0.46-pyhd8ed1ab_1.tar.bz2
- libstdcxx-ng_11.2.0 https://conda.anaconda.org/conda-forge/linux-64/libstdcxx-ng-11.2.0-hed4dale4_11.tar.bz2
- dbus_1.13.6 https://conda.anaconda.org/conda-forge/linux-64/dbus-1.13.6-h44b0894_1.tar.bz2
```

```
[INFO:ODSC:0]
[INFO:ODSC:0]
[INFO:ODSC:0] INFO:ODSC:0:2021-11-11T11:22:45.123Z:connectionpool:Connection pool is full, discarding connection: objectstorage.eu-frankfurt-1.oraclecloud.com
Part 10: 951 [REDACTED] 21.7MB/s
Part 9: 981 [REDACTED] 14.1MB/s
Part 8: 951 [REDACTED] 13.9MB/s
Part 7: 951 [REDACTED] 13.9MB/s
Part 6: 981 [REDACTED] 14.1MB/s
Part 5: 951 [REDACTED] 13.9MB/s
Part 4: 951 [REDACTED] 13.9MB/s
Part 3: 951 [REDACTED] 13.9MB/s
Part 2: 981 [REDACTED] 14.1MB/s
Part 1: 951 [REDACTED] 13.9MB/s
Part 11: 100% [REDACTED] 2.74kB/s
[INFO:ODSC:0] /home/datascience/conda/tmp/dataexpl_p37_cpu_v3.tar.gz uploaded successfully.
[INFO:ODSC:0] Installed conda environment saved
(base) [REDACTED]
(base) [REDACTED]
```



Once complete, we can revisit 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.

The screenshot shows the 'Published Conda Environments' tab selected in the top navigation bar. A single environment named 'Data Exploration and Manipulation for CPU Python 3.7' is listed. The 'Source' field is highlighted with a red box, showing a dashed URL.

Name	Environment Version	Type	Architecture	Created	Size
Data Exploration and Manipulation for CPU Python 3.7	3.0	Published	CPU	1 minute ago	83

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.

Objects

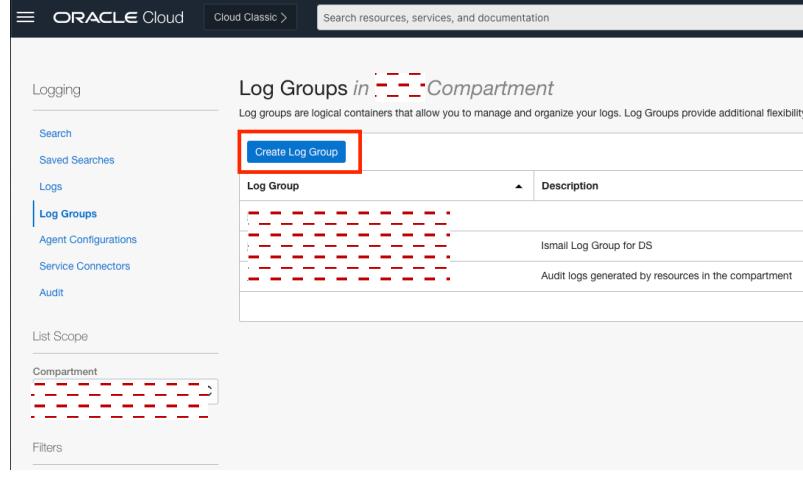
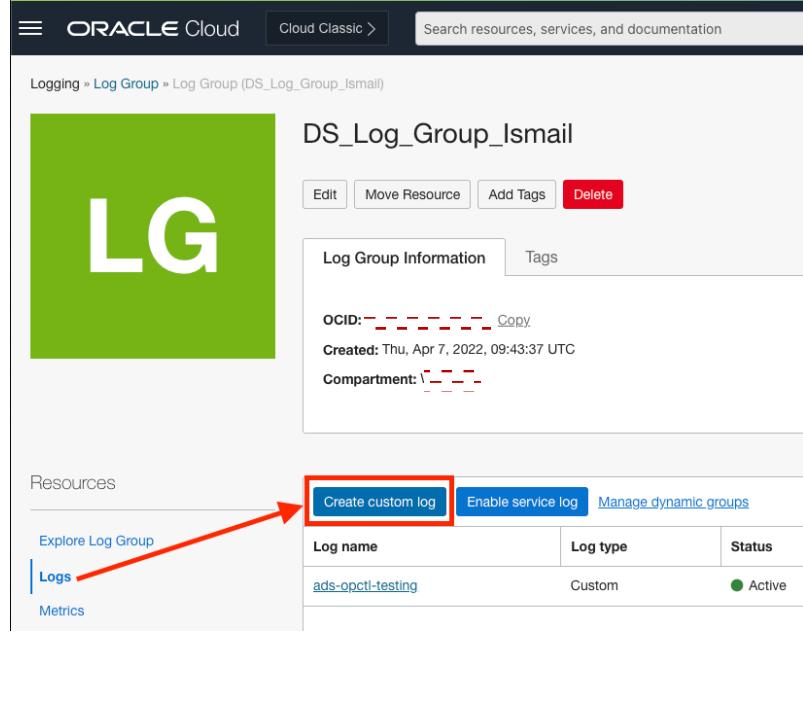
The screenshot shows the 'Objects' page with a tree view of saved files. The structure is: conda_environments/cpu/Data Exploration and Manipulation for CPU Python 3.7/3.0/dataexpl_p37_cpu_v3. The 'dataexpl_p37_cpu_v3' file is highlighted with a red box.

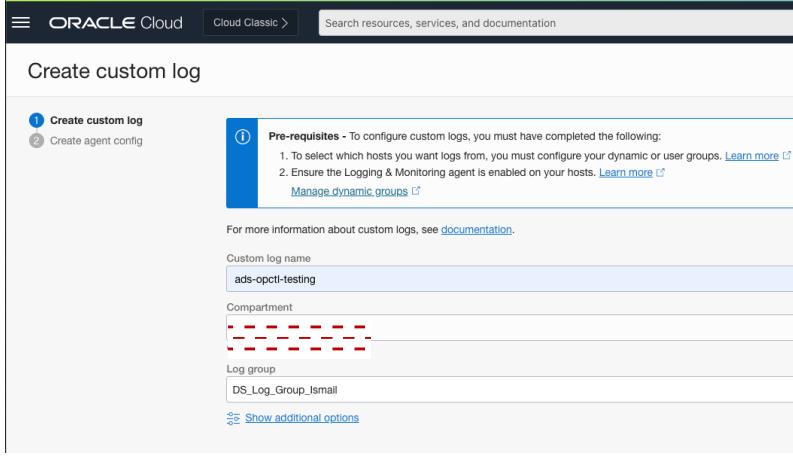
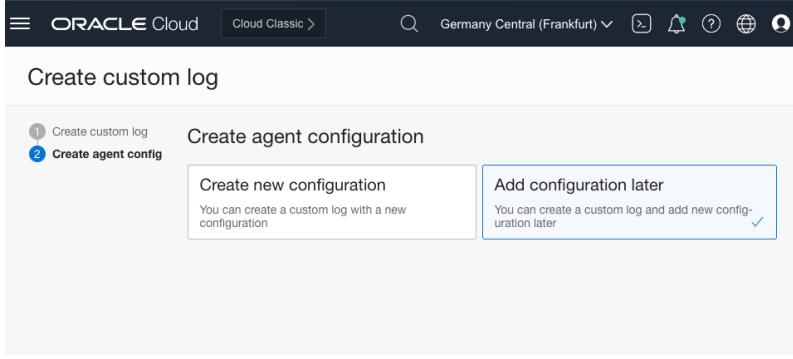
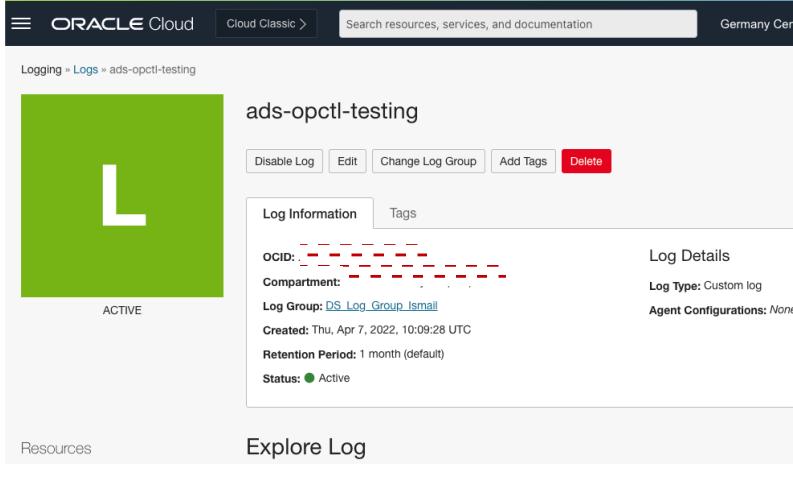
When we come to later running jobs from our local machine within OCI Data Science Service, it is a good idea to set up logging in order to understand how the job has run.

From the OCI Console visit **Menu > Observability & Management > Log Groups**

Set up Logging Service

The screenshot shows the OCI Cloud console with the 'Observability & Management' menu item highlighted with a red box. Within the 'Logs' section, the 'Log Groups' item is also highlighted with a red box.

<p>Click on '<i>Create Log Group</i>' to create a new log group to store our logs in.</p>	
<p>Enter a name for your Log Group and click '<i>Create Log Group</i>' down at the bottom.</p>	<p>Create Log Group</p> <p>Log groups are logical containers that allow you to manage and organize your logs. Log Groups provide additional flexibility information about logs and log groups, see documentation.</p> <p>Compartment: [redacted]</p> <p>Name: DS_Log_Group_Ismail</p> <p>Description:</p> <p>Optional tags to organize and track resources in your tenancy. How do I use tags?</p> <p>Tag Namespace: None (add a free-form tag) Tag Key: [redacted] Tag Value: [redacted]</p>
<p>Once the Log Group is created make note of the Log Group OCID as we will need this later.</p> <p>Now, we will go ahead and create a custom log file to store our Job Run Outputs.</p> <p>Click on '<i>Create Custom Log</i>'.</p>	

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

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.

```
$ conda create -n ads-opctl python=3.7
Collecting package metadata (current_repodata.json): done
Solving environment: done

==> WARNING: A newer version of conda exists. <==
    current version: 4.10.3
    latest version: 4.12.0

Please update conda by running

$ conda update -n base -c defaults conda

## Package Plan ##

environment location: /Applications/anaconda3/envs/ads-opctl
```

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.

```
$ pip install "oracle-ads[opctl]"
Collecting oracle-ads[opctl]
  Downloading oracle_ads-2.5.9-py3-none-any.whl (1.2 MB)
    |████████████████████████████████| 1.2 MB 6.0 MB/s
Collecting onnxmtools>=1.10.0
  Downloading onnxmtools-1.10.0-py2.py3-none-any.whl (300 kB)
    |████████████████████████████████| 300 kB 23.6 MB/s
Collecting numpy>=1.19.2
  Downloading numpy-1.21.5-cp37-cp37m-macosx_10_9_x86_64.whl (16.9 MB)
    |████████████████████████████████| 16.9 MB 7.1 MB/s
Collecting scipy>=1.5.4
  Downloading scipy-1.7.3-cp37-cp37m-macosx_10_9_x86_64.whl (33.0 MB)
    |████████████████████████████████| 33.0 MB 3.1 MB/s
Collecting numexpr>=2.7.3
  Downloading numexpr-2.8.1-cp37-cp37m-macosx_10_9_x86_64.whl (101 kB)
    |████████████████████████████████| 101 kB 4.4 MB/s
```

We will now enable the CLI.

```
pip install
"oracle-ads[opctl]"
--force-reinstall
--no-deps
--install-option=
"--enable-cli"
```

```
$ pip install "oracle-ads[opctl]" --force-reinstall --no-deps
--install-option="--enable-cli"
Applications/anaconda3/envs/ads-opctl/lib/python3.7/site-packages/pip/_internal/commands/install.py:229: UserWarning: Disabling all use of wheels due to the use of --build-option / --global-option /
--install-option.
cmdoptions.check_install_build_global(options)
Collecting oracle-ads[opctl]
  Downloading oracle_ads-2.5.9.tar.gz (933 kB)
    |████████████████████████████████| 933 kB 4.8 MB/s
  Skipping wheel build for oracle-ads, due to binaries being disabled for it.
  Installing collected packages: oracle-ads
    Attempting uninstall: oracle-ads
      Found existing installation: oracle-ads 2.5.9
      Uninstalling oracle-ads-2.5.9:
        Successfully uninstalled oracle-ads-2.5.9
      Running setup.py install for oracle-ads ... done
      Successfully installed oracle-ads-2.5.9
$
```



We can confirm the installation by running:

```
ads opctl -h
```

This will display the ads opctl help commands.

```
$ 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
$
```

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 configure
Folder to save ADS operators related configurations: [/Users/isyed/.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> []:
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: []
Specify project_id: []
Specify subnet_id: []
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> []:
Configuration saved at /Users/isyed/.ads_ops/config.ini
==== Setting configuration for OCI DataFlow ====
Do you want to set up or update OCI DataFlow configuration? [Y/n]: n
$
```

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
```

```
$ cat config.ini  
[OCI]  
oci_config = ~/.oci/config  
oci_profile = DEFAULT  
  
[CONDA]  
conda_pack_folder = /Applications/anaconda3/envs  
conda_pack_os_prefix =  
  
$
```

We can print out the second config file:

```
cat ml_job_config.ini
```

```
$ cat ml_job_config.ini  
[DEFAULT]  
compartment_id = ocid1.compartment.oc1..  
project_id = ocid1.project.oc1..  
subnet_id = ocid1.subnet.oc1..  
log_group_id =  
log_id = ocid1.log.oc1..  
shape_name = VM.Standard2ze.1  
block_storage_size_in_GBs = 50  
conda_pack_os_prefix =
```

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
```

```
$ ads opctl build-image job-local  
#1 [internal] load build definition from Dockerfile.job  
#1 ...  
#1 transferring dockerfile: 3.75kB 0.0s done  
#1 DONE 0.0s  
  
#2 [internal] load .dockerignore  
#2 ...  
#2 transferring context: 2B done  
#2 DONE 0.0s  
  
#3 [internal] load metadata for docker.io/library/oraclelinux:7-slim  
#3 ...  
#3 DONE 2.4s  
  
#4 [ 1/18] FROM docker.io/library/oraclelinux:7  
#4 ...  
#4 resolve docker.io/library/oraclelinux
```

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
```

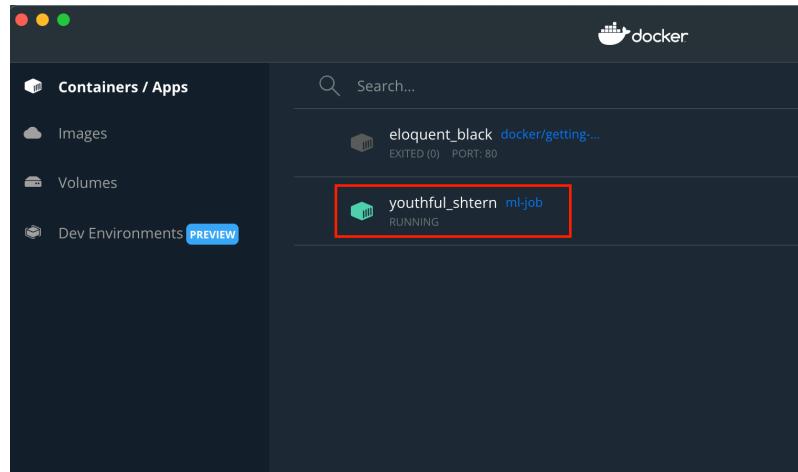
Where **-s** is the **slug name** of the conda environment published to the OCI Object Storage.

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.

```
$ ads opctl conda install -s dataexpl_p37_cpu_v3
export OCI_CLI_SUPPRESS_FILE_PERMISSIONS_WARNING=True
Downloading object [#####] 100%
Download [#####] completed
Start unpacking /Applications/anaconda3/envs/dataexpl_p37_cpu_v3.tar.gz
```

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
```

```
$ 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
```

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.

```
ads-hello-world.py
1 import time
2
3 print('Hello World!\n')
4
5 time.sleep(3)
6
7 print('Job Done!')
```

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 <-fFlag value> --env-var ENV_NAME=value
```

```
$ ads opctl run -s ~/Downloads -e ads-hello-world.py
Hello World!
Job Done!
```

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 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.

```
$ ads opctl watch ocid1.datasciencjobrun.oc1.eu-fran
Logging is not configured for the job. Watch() will only show job status.
Job OCID: [REDACTED]
Job Run OCID: [REDACTED]

2022-04-07 10:03:44 - Job Run ACCEPTED, Infrastructure provisioning.
2022-04-07 10:04:23 - Job Run ACCEPTED, Infrastructure provisioned.
2022-04-07 10:04:52 - Job Run ACCEPTED, Job run bootstrap starting.
2022-04-07 10:06:03 - Job Run ACCEPTED, Job run bootstrap complete. Artifact execution starting.
2022-04-07 10:06:07 - Job Run IN_PROGRESS, Job run artifact execution in progress.
2022-04-07 10:06:10 - Job Run IN_PROGRESS, Job run artifact execution succeeded. Infrastructure de-provisioning.
2022-04-07 10:07:49.697000+00:00 - Job Run SUCCEEDED, Job run artifact execution succeeded. Infrastructure de-provisioning.

$
```

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.

test-conda-pack-run-20220407-1125

Clone	Edit	Cancel	Move resource	More Actions ▾
Job run Runtime configuration Tags				
General information				
OCID: [REDACTED]	Created by: [REDACTED]			
Time accepted: Thu, Apr 7, 2022, 10:25:36 UTC	Time started: Thu, Apr 7, 2022, 10:28:30 UTC			
Time finished: Thu, Apr 7, 2022, 10:29:57 UTC				
Logging details				
Log group: DS_Log_Group_Ismail	Log: ads-opctl-testing			

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

Explore Log

Sort: Newest Filter by time: Past 5 minutes

Number of Log Events Per Minute

Log Data

Thu, Apr 7, 2022, 10:28:24 UTC	datascience.jobrun.stdout	Job Done!
Thu, Apr 7, 2022, 10:28:21 UTC	datascience.jobrun.stdout	Hello Isy!

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.py ads-hello-world-args.py
1 import time
2 import argparse
3
4 # Read in command line argument
5 parser = argparse.ArgumentParser()
6 parser.add_argument('-g', '--greeting', required=False, default='Mystery Person')
7 args = parser.parse_args()
8
9
10 print(f'Hello {args.greeting}!\n')
11
12 time.sleep(3)
13
14 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 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 watch ocid1.datasciencjobrun.oc1.eu-frankfurt-1.[REDACTED]
Job OCID: [REDACTED]
kybzsq4nkq
Job Run OCID: [REDACTED]
[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
1 import os
2 import time
3
4 print('Hello World!\n')
5
6 time.sleep(3)
7
8 print(f'Hello {os.environ.get("TEST_NAME", "UNKNOWN")}')
9
10 time.sleep(3)
11
12 print('Job Done!')
```

