

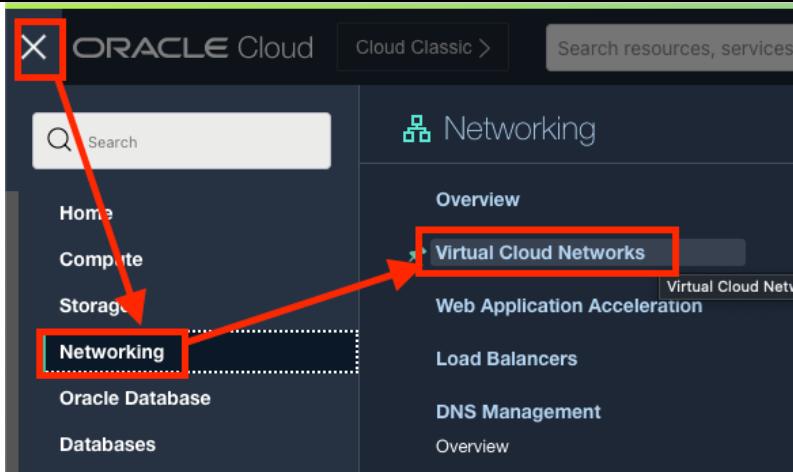
Distributed Training with OCI DS Jobs

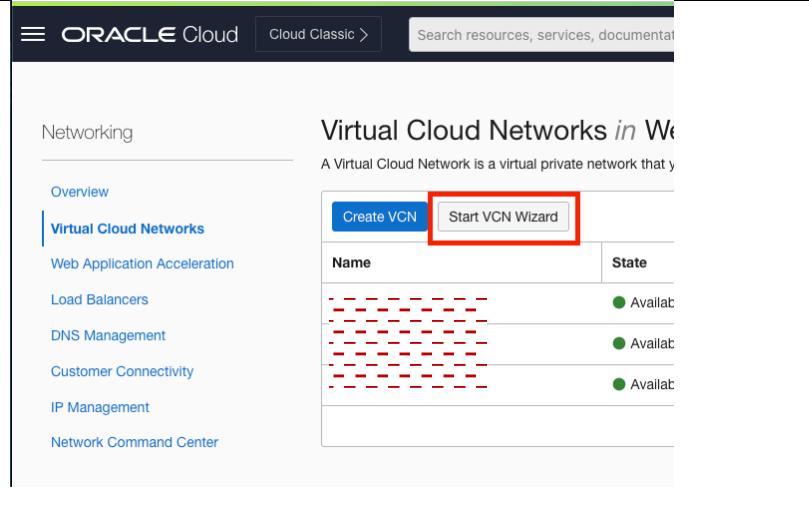
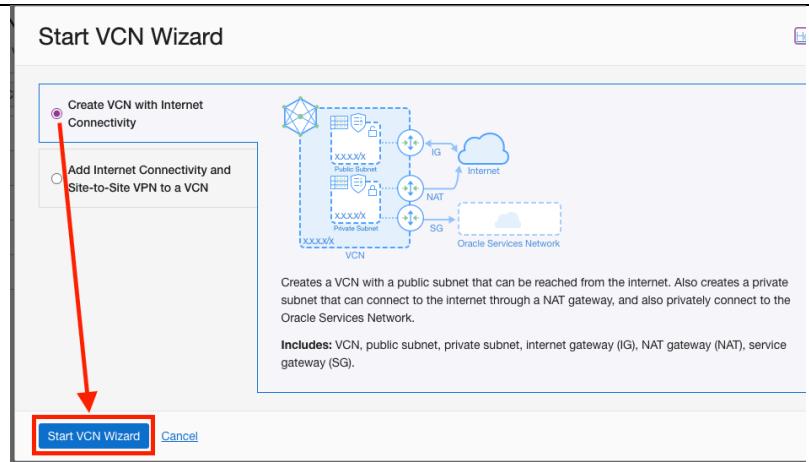
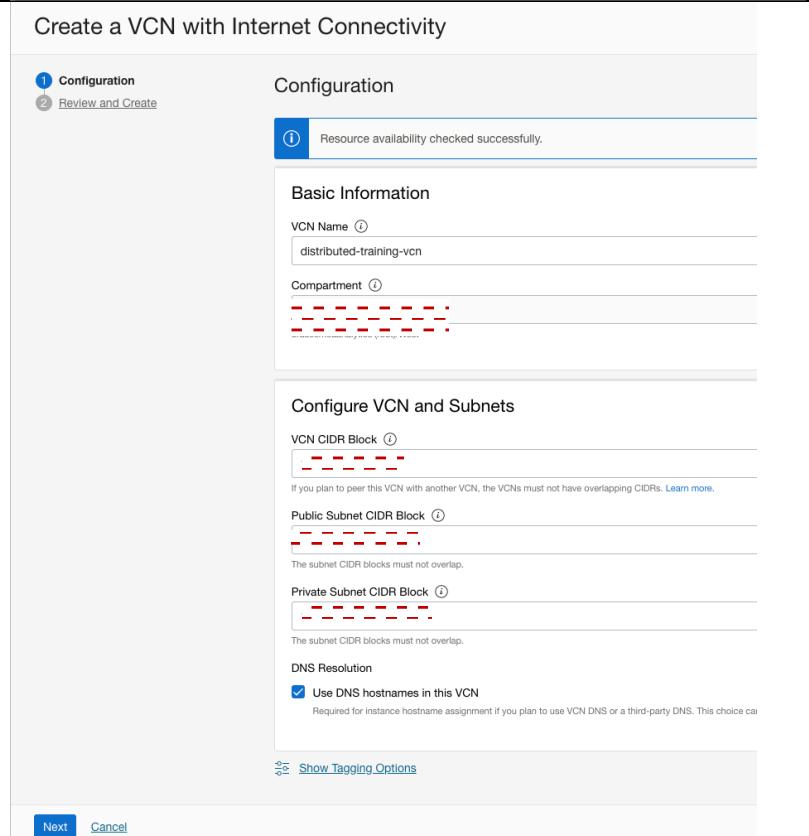
Referenced Documentation: https://github.com/oracle-samples/oci-data-science-ai-samples/tree/master/distributed_training/notebooks

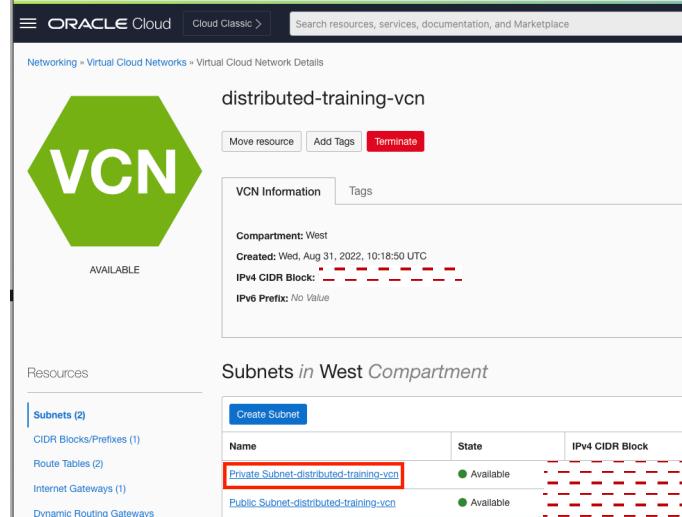
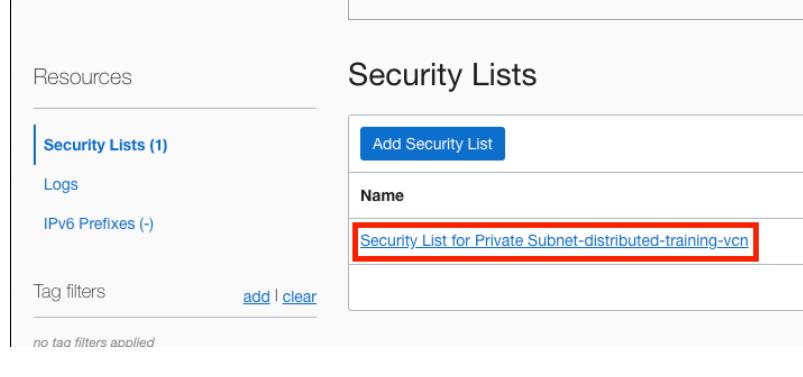
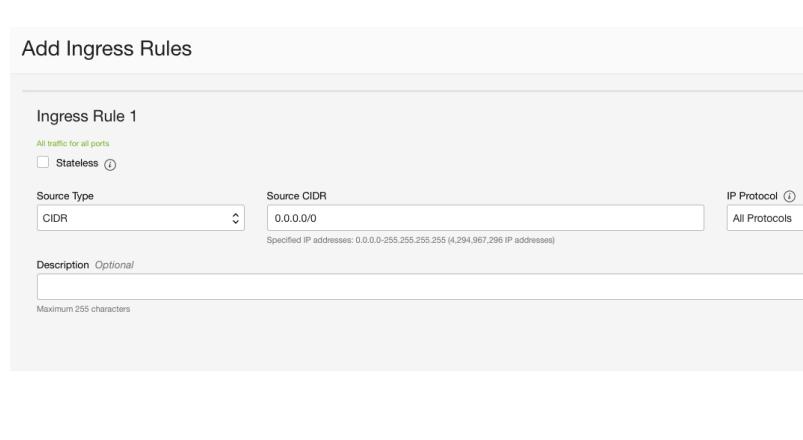
In this document, we will only be looking at the Horovod TensorFlow Distributed Training Example.

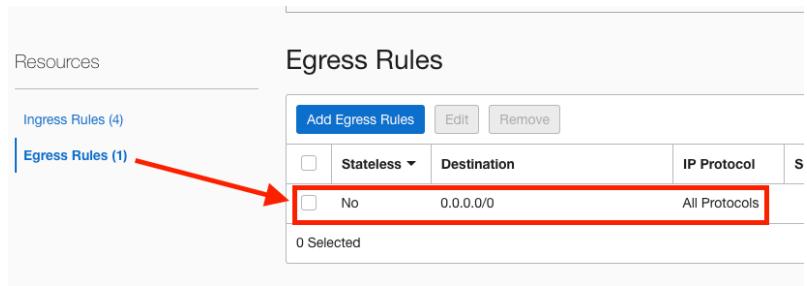
Here are the key pre-requisites that you would need to execute before you can proceed to building a distributed workload docker image and execute it using **odsc-jobs**.

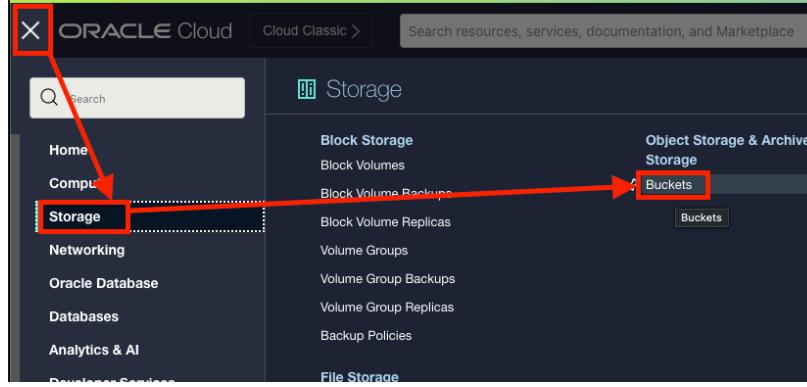
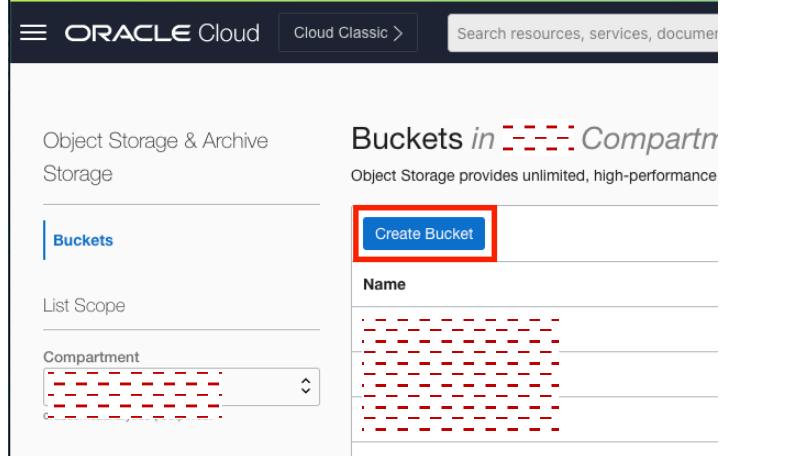
1. Configure Network. Required for the inter node communication.
2. OCI policies. Required for accessing OCI services by the distributed job.
3. Install ads-opctl. Required for packaging training script and launching odsc distributed job.

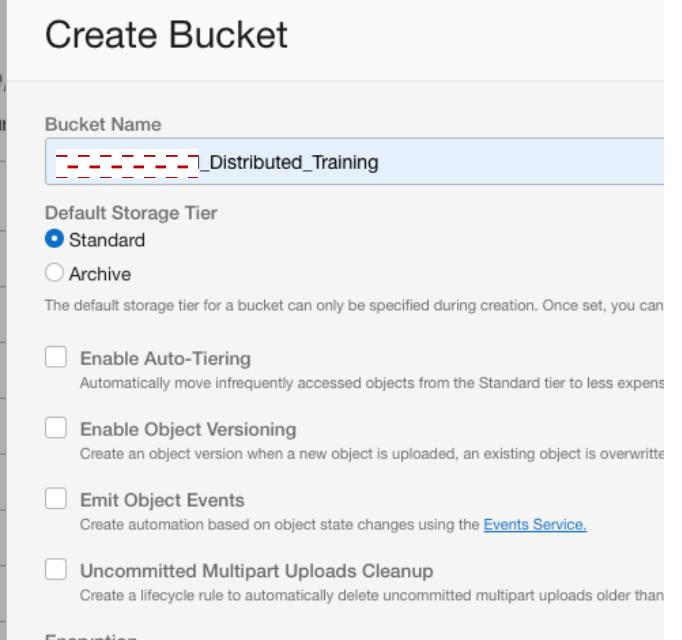
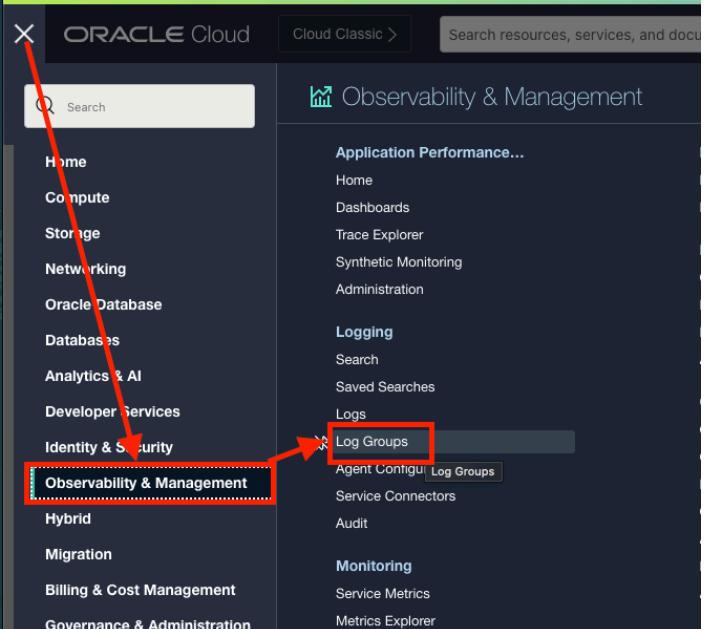
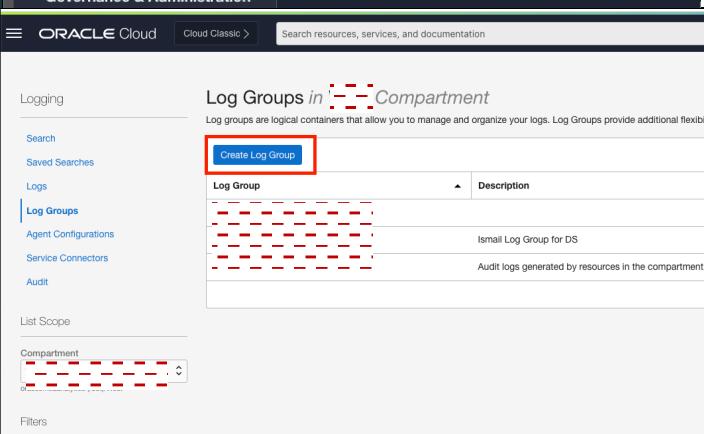
Networking Pre-Requisite	
Step	Screenshot
<p>Create a new VCN with a Private Subnet, that allows all traffic within the subnet.</p> <p>This will allow the nodes to all communicate with each other.</p> <p>OCI Console Menu > Networking > Virtual Cloud Networks.</p>	

<p>Click on <i>Start VCN Wizard</i>.</p>	
<p>Select <i>Create VCN with Internet Connectivity</i>.</p> <p>Click on <i>Start VCN Wizard</i>.</p>	
<p>Enter a <i>Name</i>.</p> <p>Leave defaults or enter CIDR Block ranges for the VCN, Public Subnet and Private Subnet.</p> <p>Click <i>Next</i>.</p> <p>Review Summary Page.</p> <p>Click <i>Create</i>.</p>	

<p>We need to now allow all communication between the nodes in the Subnet.</p> <p>Click on the <i>Private Subnet</i> that has just been created.</p>	 <p>The screenshot shows the Oracle Cloud interface for a Virtual Cloud Network named "distributed-training-vcn". It displays basic information like compartment (West), creation date (Wed, Aug 31, 2022, 10:18:50 UTC), and IPv4 CIDR Block (dashed red line). Below this, a table lists "Subnets in West Compartment" with two entries: "Private Subnet-distributed-training-vcn" and "Public Subnet-distributed-training-vcn", both marked as available.</p>
<p>Select the <i>Private Subnet Security List</i>.</p>	 <p>The screenshot shows the Oracle Cloud interface for "Security Lists". A new security list is being created with the name "Security List for Private Subnet-distributed-training-vcn", highlighted with a red box.</p>
<p>We will add a new Ingress Rule to Allow communication between all IPs and Ports within the Subnet.</p> <p>Click <i>Add Ingress Rule</i>.</p>	 <p>The screenshot shows the Oracle Cloud interface for "Ingress Rules". The "Add Ingress Rules" button is highlighted with a red box. The table below shows three existing rules, each with a "Source" column containing a dashed red line.</p>
<p>Add a Rule with Source CIDR as 0.0.0.0/0 (This can also be locked down to the Subnet itself, however for simplicity we will leave it as such).</p> <p>Set IP Protocol as <i>All Protocols</i>.</p> <p>Click <i>Add Ingress Rules</i>.</p>	 <p>The screenshot shows the "Add Ingress Rules" configuration page. It includes fields for "Source Type" (CIDR), "Source CIDR" (0.0.0.0/0), and "IP Protocol" (All Protocols). The "Description" field is optional and empty.</p>

<p>Double Check the Egress Rules to ensure the same Rule is set up here (Destination 0.0.0.0/0 All Protocols).</p> <p>This should have been created by default anyhow.</p>	 <p>The screenshot shows the 'Egress Rules' section of the Oracle Cloud interface. It displays one rule: 'No' for Stateless, '0.0.0.0/0' for Destination, and 'All Protocols' for IP Protocol. A red arrow points from the text in the left column to this rule.</p>
--	--

Steps	Screenshot
<p>We will create an Object Storage Bucket as a Staging area and to store outputs in when the job runs.</p> <p>OCI Menu > Storage Buckets.</p>	 <p>The screenshot shows the OCI Cloud Classic navigation bar with 'Storage' selected. Below it, the 'Storage' service page is shown with 'Buckets' highlighted. Red arrows point from the 'Storage' link in the menu and the 'Buckets' link in the service page to their respective locations.</p>
<p>Click Create Bucket</p>	 <p>The screenshot shows the 'Create Bucket' form. It includes fields for 'Name' (with a placeholder 'my-new-bucket') and 'Compartment' (with a dropdown showing a compartment name). A large blue 'Create Bucket' button is prominently displayed at the top right of the form.</p>

<p>Enter a Bucket Name and Click on Create.</p>	 <p>Create Bucket</p> <p>Bucket Name: <input type="text" value="Ismail_Distributed_Training"/></p> <p>Default Storage Tier: <input checked="" type="radio"/> Standard <input type="radio"/> Archive</p> <p>The default storage tier for a bucket can only be specified during creation. Once set, you can change it later.</p> <p><input type="checkbox"/> Enable Auto-Tiering Automatically move infrequently accessed objects from the Standard tier to less expensive tiers.</p> <p><input type="checkbox"/> Enable Object Versioning Create an object version when a new object is uploaded, an existing object is overwritten.</p> <p><input type="checkbox"/> Emit Object Events Create automation based on object state changes using the Events Service.</p> <p><input type="checkbox"/> Uncommitted Multipart Uploads Cleanup Create a lifecycle rule to automatically delete uncommitted multipart uploads older than 1 day.</p>				
<p>When we come to later running jobs within OCI Data Science Service, it is a good idea to set up logging in order to understand how the job has run.</p> <p>From the OCI Console visit Menu > Observability & Management > Log Groups</p>	 <p>ORACLE Cloud</p> <p>Observability & Management</p> <ul style="list-style-type: none"> Application Performance... Home Dashboards Trace Explorer Synthetic Monitoring Administration Logging Search Saved Searches Logs Log Groups Agent Config Service Connectors Audit Monitoring Service Metrics Metrics Explorer <p>Observability & Management</p>				
<p>Click on 'Create Log Group' to create a new log group to store our logs in.</p>	 <p>Log Groups in Ismail Compartment</p> <p>Log groups are logical containers that allow you to manage and organize your logs. Log Groups provide additional flexibility.</p> <p>Create Log Group</p> <table border="1"> <thead> <tr> <th>Log Group</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>Ismail Log Group for DS</td> <td>Audit logs generated by resources in the compartment</td> </tr> </tbody> </table>	Log Group	Description	Ismail Log Group for DS	Audit logs generated by resources in the compartment
Log Group	Description				
Ismail Log Group for DS	Audit logs generated by resources in the compartment				

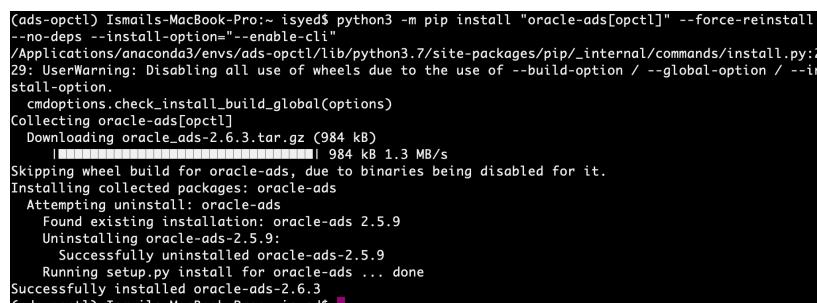
<p>Enter a name for your Log Group and click 'Create Log Group' down at the bottom.</p>	
<p>We will create multiple policies to allow users and our Data Science Dynamic Group access to Networking Components, Data Science Components, OCI Container Registry, Logs, Metrics and Object Storage.</p> <p>OCI Menu > Identity & Security > Policies.</p>	
<p>Click on Create Policy.</p>	

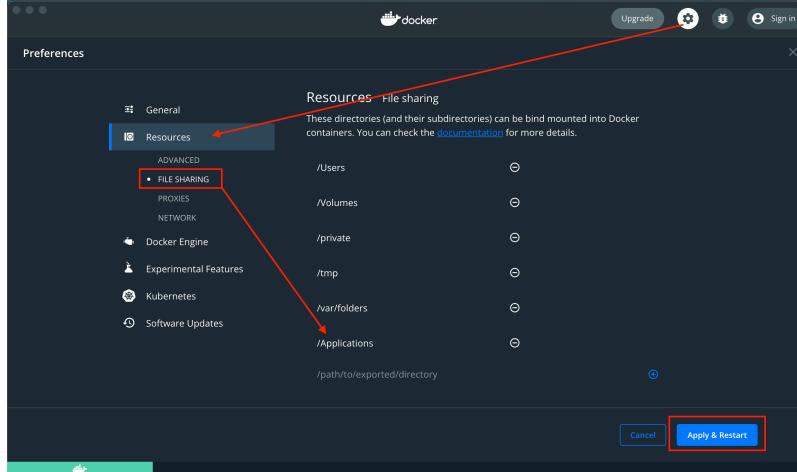
Give the Policy a **Name**, **Description** and enter the following policies, under **Manual Editor**.

NOTE: This assumes you have already created a **Group** of Users, have a **Dynamic Group** set up for your Data Science Resources

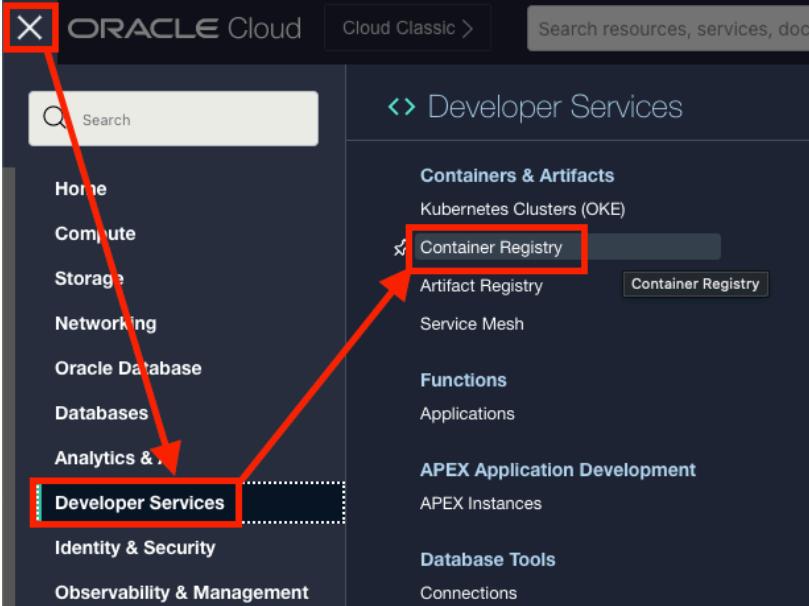
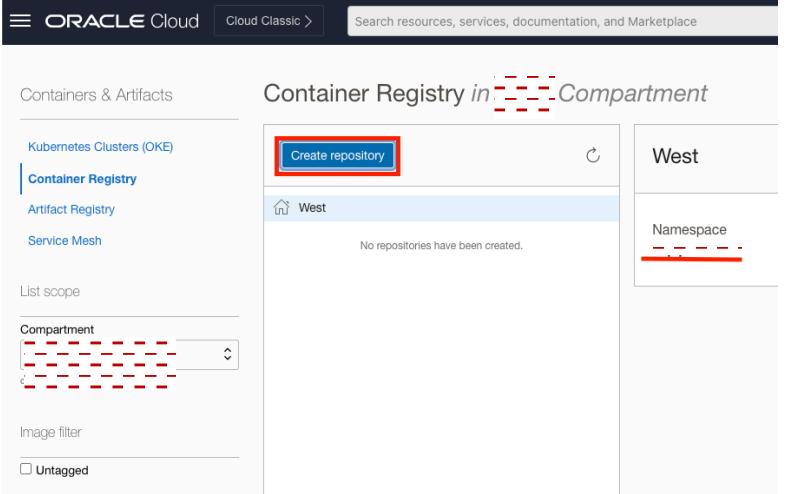
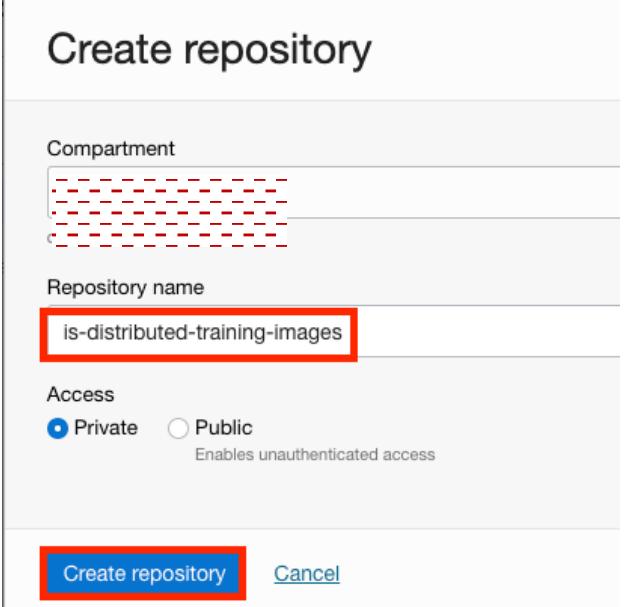
Click **Create**.

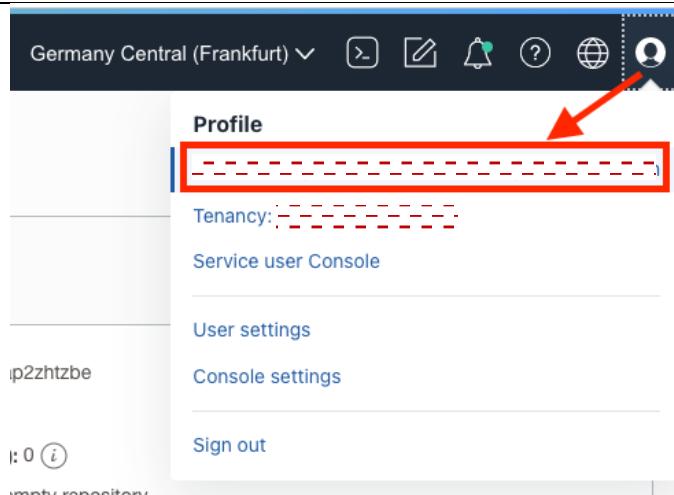
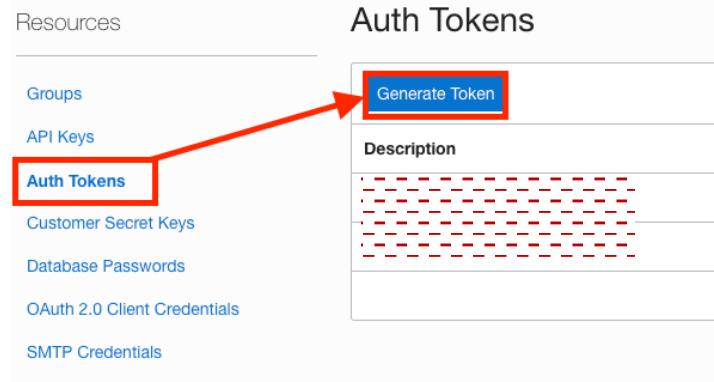
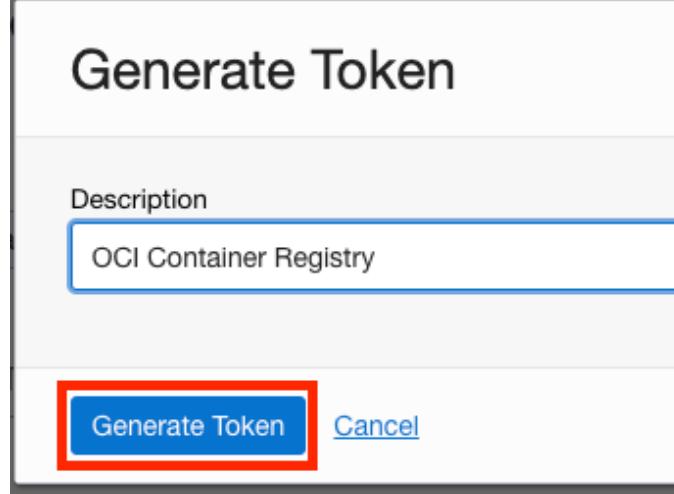
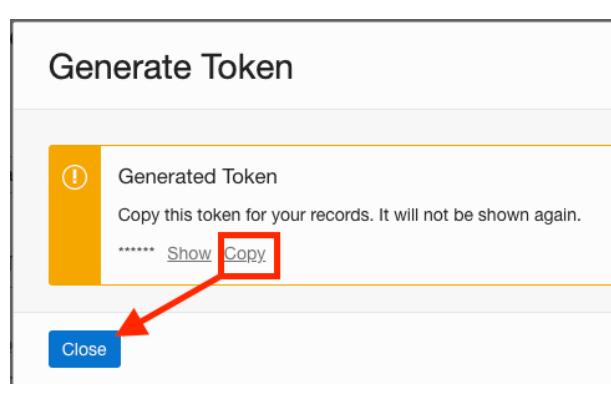
```
Allow group <group> to manage repos in compartment <compartment>
Allow group <group> to manage data-science-jobs in compartment <compartment>
Allow group <group> to manage data-science-job-runs in compartment <compartment>
Allow group <group> to use virtual-network-family in compartment <compartment>
Allow group <group> to manage log-groups in compartment <compartment>
Allow group <group> to use logging-family in compartment <compartment>
Allow group <group> to use metrics in compartment <compartment>
Allow dynamic-group <dynamic-group> to read repos in compartment <compartment>
Allow dynamic-group <dynamic-group> to use data-science-family in compartment <compartment>
Allow dynamic-group <dynamic-group> to use virtual-network-family in compartment <compartment>
Allow dynamic-group <dynamic-group> to use log-groups in compartment <compartment>
Allow dynamic-group <dynamic-group> to use logging-family in compartment <compartment>
Allow group <group> to manage objects in compartment <compartment> where all
{target.bucket.name=<object_storage_bucket_name>}
Allow dynamic-group <dynamic-group> to manage objects in compartment <compartment> where all
{target.bucket.name=<object_storage_bucket_name>}
Allow dynamic-group <dynamic-group> to manage buckets in compartment <compartment> where all
{target.bucket.name=<object_storage_bucket_name>}
```

Install ads-opctl and Docker	
Steps	Screenshot
<p>This assumes you already have Python3 installed locally on your machine.</p> <p>I will install ADS within a Python3.7 Conda Environment I have already created.</p> <pre>python3 -m pip install "oracle-ads[opctl]"</pre> <pre>python3 -m pip install "oracle-ads[opctl]" --force-reinstall --no-deps --install-option="--enable-cli"</pre>	 <p>Need to run both commands on the left.</p>
<p>Download Docker and Run Installation from:</p> <p>https://docs.docker.com/get-docker</p>	

<p>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.</p> <p>Visit Settings > Resources > File Sharing</p> <p>Add your parent folder for conda in my case /Applications.</p> <p>Click Apply & Restart.</p>	 <p>This is more a pre-requisite for Jobs and does not have to be done, but it is good practise.</p>
<p>Using the Terminal, navigate to a folder where you wish to download the related docker artifacts for distributed TF training.</p> <p>cd ~/Distributed_Training</p> <p>Using the ads-opctl, download and initialize the relevant distributed training framework.</p> <p>ads opctl distributed-training init --framework horovod-tensorflow --version v1</p>	<pre>(ads-opctl) Ismails-MacBook-Pro:~ isyed\$ ads opctl distributed-training init --framework horovod-tensorflow --version v1 Downloading from https://raw.githubusercontent.com/oracle-samples/oci-data-science-ai-samples/master/distributed_training/artifacts/horovod_tensorflow_v1.tar.gz to .tmp_horovod_tensorflow_v1.tar.gz x horovod/v1/cluster_runner_horovod.py x horovod/v1/discover_workers.py x horovod/v1/horovod_provider.py x horovod/v1/ip_helper.py x horovod/v1/run_horovod.py x horovod/v1/discover-hosts.sh x horovod/v1/run.sh x horovod/v1/start_worker.sh x horovod/v1/sync.sh x horovod/v1/docker/tensorflow.cpu.Dockerfile x horovod/v1/docker/tensorflow.gpu.Dockerfile x horovod/v1/conda-tensorflow-cpu.yaml x horovod/v1/conda-tensorflow-gpu.yaml x horovod/v1/README.md Check oci_dist_training_artifacts/horovod_tensorflow/v1/README.md for build instructions (ads-opctl) Ismails-MacBook-Pro:~ isyed\$</pre>

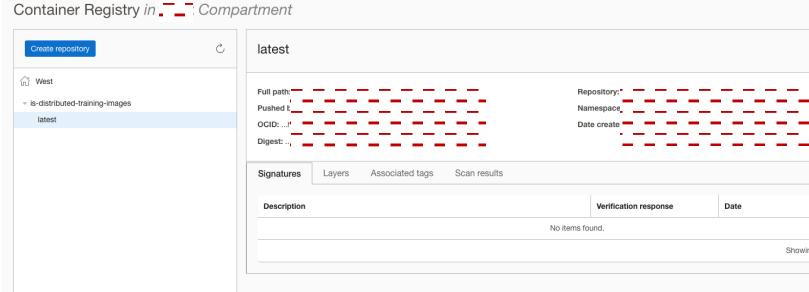
Prepare Docker Image	
Steps	Screenshots
<p>All the docker image related artifacts are located under: <i>oci_dist_training_artifacts/horovod/v1/</i></p> <p>Horovod provides support for Pytorch and Tensorflow. Within these frameworks, there are two separate docker files, for cpu and gpu.</p> <p>Choose the docker file and conda environment files based on whether you are going to use Pytorch or Tensorflow with either cpu or gpu.</p> <p>The instruction assumes that you are running this within the folder where you ran <i>ads opctl distributed-training init -framework horovod <pytorch/tensorflow></i></p>	<p>Continued...</p> <p>Upload your Tensorflow training script (<i>train.py</i>) within your current directory on your local machine as this will then be copied into the <i>/code</i> folder inside docker image once built.</p> <p>Note: Whenever you change the code, you have to build, tag and push the image to repo. If you change the tag, it needs to be updated inside the cluster definition yaml later on.</p> <p>The required python dependencies are provided inside the conda environment file <i>oci_dist_training_artifacts/horovod/v1/conda-<pytorch/tensorflow>-<cpu/gpu>.yaml</i>. If your code requires additional dependency, update this file.</p> <p>Note: While updating <i>conda-<pytorch/tensorflow>-<cpu/gpu>.yaml</i> do not remove the existing libraries. You can append to the list.</p>

<p>We will now create an OCI Container Registry to store our Docker Image in when we push from our local repo to the cloud.</p> <p>This pushed Docker Image will be used in the Job Run.</p> <p>OCI Menu > Developer Services > Container Registry.</p>	
<p>Make note of the Namespace to your right, you will need this later.</p> <p>Click on Create Repository.</p>	
<p>Give it a Name and Click Create Repository.</p>	

We then need to Generate 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.	

<p>Ensure your local Docker installation is running, before doing the following tasks.</p> <p>In a terminal window on the client machine running Docker, log in to Container Registry by entering <i>docker login <region-key>.ocir.io</i>, where <region-key> corresponds to the key for the Container Registry Region you're using.</p>	<pre>(ads-opctl) Ismails-MacBook-Pro:Distributed-Training isyed\$ (ads-opctl) Ismails-MacBook-Pro:Distributed-Training isyed\$ (ads-opctl) Ismails-MacBook-Pro:Distributed-Training isyed\$ docker login fra.ocir.io Username: [REDACTED] Password: Login Succeeded (ads-opctl) Ismails-MacBook-Pro:Distributed-Training isyed\$ </pre> <p><i>docker login fra.ocir.io for Frankfurt region</i></p>
<p>When prompted for a username, enter your username in the format <tenancy-namespace>/<username></p> <p>If your tenancy is federated with Oracle IDCS, use the format <tenancy-namespace>/oracleidentitycloudservice/<username></p>	<pre>(ads-opctl) Ismails-MacBook-Pro:Distributed-Training isyed\$ (ads-opctl) Ismails-MacBook-Pro:Distributed-Training isyed\$ (ads-opctl) Ismails-MacBook-Pro:Distributed-Training isyed\$ docker login fra.ocir.io Username: [REDACTED] Password: Login Succeeded (ads-opctl) Ismails-MacBook-Pro:Distributed-Training isyed\$ </pre>
<p>When prompted for a password, enter the Auth Token you copied earlier.</p>	<pre>(ads-opctl) Ismails-MacBook-Pro:Distributed-Training isyed\$ (ads-opctl) Ismails-MacBook-Pro:Distributed-Training isyed\$ (ads-opctl) Ismails-MacBook-Pro:Distributed-Training isyed\$ docker login fra.ocir.io Username: [REDACTED] Password: Login Succeeded (ads-opctl) Ismails-MacBook-Pro:Distributed-Training isyed\$ </pre>



<p>Once complete, we can list our Docker Images</p> <p>docker image ls</p>	<pre>(ads-opctl) Ismails-MacBook-Pro:Distributed-Training isyed\$ docker image ls REPOSITORY TAG IMAGE ID CREATED SIZE ml-job latest 8ecc585f431c 4 months ago 3.97GB 3.45GB 3.97GB docker/getting-started latest bd9a9f733898 6 months ago 28.8MB (ads-opctl) Ismails-MacBook-Pro:Distributed-Training isyed\$</pre>
<p>Push the local Docker Image to the OCI Registry.</p> <p>docker push \$IMAGE_NAME:\$TAG</p> <p>Where IMAGE_NAME refers to the repo you want to push to.</p>	<pre>(ads-opctl) Ismails-MacBook-Pro:Distributed-Training isyed\$ docker push \$IMAGE_NAME:\$TAG The push refers to repository [REDACTED] f10ad6ffc383: Pushed 0956d49a7d8b: Pushed 356bf6e79263: Pushed 28be39c2b950: Pushed 1d1400de82a3: Pushed e842e591c85b: Pushed 5f70bf18a086: Pushed 9d9aa9ec7949: Pushed 7d8dc3457555: Pushed 03567b043911: Pushed 1859e1e14339: Pushing 598.2MB/1.943GB 18e7ff736de1d: Pushed b200ee6d7ee0: Pushed 99217ceef20d: Pushing 286.1MB/748.7MB 068e4eac9ed1: Pushing 246MB/414.6MB c6b466cc9552: Pushed</pre>
<p>We can then view our Image in the OCI Registry.</p>	 <p>The screenshot shows the OCI Container Registry interface. It displays a list of tags for a repository named 'is-distributed-training-images'. The 'latest' tag is selected. The interface includes fields for 'Full path', 'Pushed', 'OCIID', and 'Digest'. Below the table, there are tabs for 'Signatures', 'Layers', 'Associated tags', and 'Scan results'. A note at the bottom states 'No items found.'</p>
<p>Before triggering the job run, you can test the docker image and verify the training code, dependencies etc. You can do this using a local stand-alone docker run</p> <p>For a stand-alone docker run, start a docker container with bash entry point.</p> <p>docker run --rm -it --privileged --entrypoint bash \$IMAGE_NAME:\$TAG</p>	<pre>(ads-opctl) Ismails-MacBook-Pro:Distributed-Training isyed\$ docker run --rm -it --privileged --entrypoint bash \$IMAGE_NAME:\$TAG int bash \$IMAGE_NAME:\$TAG (env) [root@6ec1b2e06e72 horovod]# (env) [root@6ec1b2e06e72 horovod]# pwd /etc/dataservice/horovod (env) [root@6ec1b2e06e72 horovod]# ls cluster_runner_horovod.py discover-hosts.sh horovod_provider.py run.sh start_worker.sh conda-tensorflow-cpu.yaml discover_workers.py ip_helper.py run_horovod.py sync.sh (env) [root@6ec1b2e06e72 horovod]#</pre>

You have now entered your docker container.

Now test your training script with the following command.

```
horovodrun -np 2 -H localhost:2 python /code/train.py --data-dir /code/data/mnist.npz
```

```
(env) [root@6ec1b2e06e72 ~]# horovodrun -np 2 -H localhost:2 python /code/train.py --data-dir /code/data/mnist.npz
[1,1]<stdout>:Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
[1,0]<stdout>:Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
[1,0]<stdout>: 5537792/11490434 [=====.....] - ETA: 4s[1,1]<stdout>:11272192/11490434
[1,1]<stdout>:11493376/11490434 [=====.....] - 4s 0us/step<stdout>:11493376/11490434
[1,0]<stdout>: 6s 1us/step - loss: 2.3076 - accuracy: 0.1016
[1,1]<stdout>: 6s 6s/step - loss: 2.3012 - accuracy: 0.1094[1,0]<stdout>:11493376/11490434
[1,0]<stdout>: 4s 4s/step - loss: 2.3012 - accuracy: 0.1094[1,0]<stdout>:11493376/11490434
[1,1]<stdout>:Epoch 1/2
[1,1]<stdout>:Epoch 1/2
[1,1]<stdout>:246/250 [=====.....] - ETA: 2s - loss: 0.2128 - accuracy: 0.9343[1,0]<stdout>:246/250
[1,0]<stdout>:250/250 [=====.....] - 146s 585ms/step - loss: 0.2128 - accuracy: 0.9343[1,0]<stdout>:246/250
[1,1]<stdout>:250/250 [=====.....] - 146s 585ms/step - loss: 0.2126 - accuracy: 0.9344
[1,1]<stdout>:Epoch 2/2
[1,0]<stdout>:Epoch 2/2
247/250 [=====.....] - ETA: 1s - loss: 0.0801 - accuracy: 0.9763[1,0]<stdout>:247/250
[1,0]<stdout>:250/250 [=====.....] - 143s 573ms/step - loss: 0.0796 - accuracy: 0.9759[stdout>:::
[1,1]<stdout>:250/250 [=====.....] - 143s 573ms/step - loss: 0.0801 - accuracy: 0.9762
[env] [root@6ec1b2e06e72 ~]#
```

Once done, exit the container with the exit command.

exit

```
(env) [root@6ec1b2e06e72 ~]#
(env) [root@6ec1b2e06e72 ~]#
(env) [root@6ec1b2e06e72 ~]# exit
exit
(ads-opctl) Ismails-MacBook-Pro:Distributed-Training isyed$
```

Create a **train.yaml** local file which will contain the job definition.

Edit the file and adjust parameters as necessary.
E.g. **Project ID**, **Compartment**, **Log Group**, **Docker Image**, etc...

An example file is contained in the same directory as this Guide.



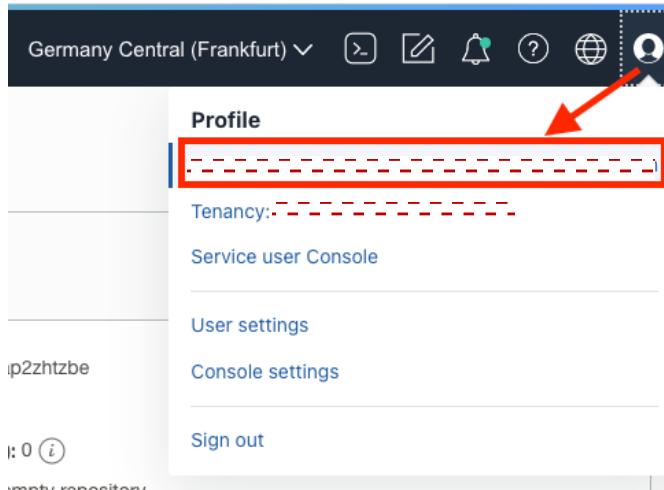
This will print/validate the Job and Job run configuration without launching the actual job.

ads opctl run -f train.yaml --dry-run

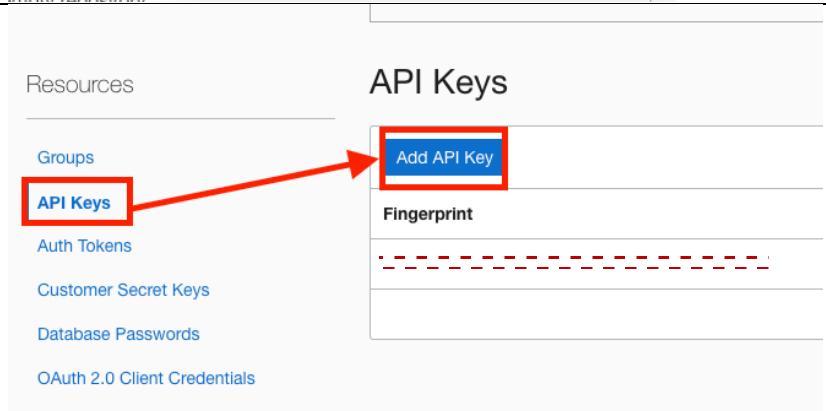
```
(ads-opctl) Ismails-MacBook-Pro:Distributed-Training isyed$ ads opctl run -f train.yaml --dry-run
-----
Creating Job with payload:
kind: job
spec:
  infrastructure:
    kind: infrastructure
    spec:
      blockStorageSize: 50
      compartmentId: -----
      displayname: horovod_tf
      jobInfrastructureType: ME_STANDALONE
      jobType: DEFAULT
      logGroupId: -----
      langVfp6q
      projectId: -----
      shapeName: VM.Standard2.4
      subnetId: -----
  xzucla
    type: dataScienceJob
    name: horovod_tf
    runtime:
      kind: runtime
      spec:
        entrypoint: null
        env:
          - name: WORKER_PORT
            value: '12345'
          - name: START_TIMEOUT
            value: '600'
          - name: ENABLE_TIMELINE
            value: '0'
          - name: SYNC_ARTIFACTS
```

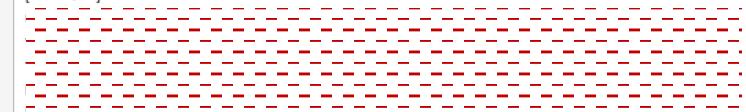
Before you submit the job to OCI DS Jobs, you need to add your Private Key File and Config file.

Click on your **Profile** then **Username**



Navigate to **API Keys > Add API Key**



<p>Generate API Key Pair.</p> <p>Download Private Key.</p> <p>Click Add.</p>	<h3>Add API Key</h3> <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 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> <p> Download the private key. It will not be shown again. After you download it, change the file permissions so o</p> <p>Download Private Key Download Public Key</p> <p>Add Cancel</p>
<p>Copy the config file and save locally along with the private key to <code>~/.oci</code>.</p> <p>Ensure to update the path to the private key in the config file.</p>	<h3>Configuration File Preview</h3> <p>Note: This configuration file snippet includes the basic authentication information you'll need to use the SDK, 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. If you have a Default profile in your config profile, you'll need to perform some additional steps. Learn more</p> <p>Select API Key Fingerprint </p> <p>Configuration File Preview <i>Read-only</i></p> <p>[DEFAULT] </p> <p>Paste the contents of the text box into your <code>~/.oci/config</code> file.</p> <p>Close</p>

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

`ads opctl configure`

The above is normally done for submitting standard jobs so not all info is needed.

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.

```
(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> []:
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/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$
```

This will create two files in `/User/<username>/.ads_ops`

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`

```
(base) Ismails-MacBook-Pro:.ads_ops isyed$ cat config.ini
[OCI]
oci_config = ~/.oci/config
oci_profile = DEFAULT

[CONDA]
conda_pack_folder = /Applications/anaconda3/envs
conda_pack_os_prefix = []
```

We can now submit the training job to OCI DS Jobs and we can also save the output to **info.yaml**.

You could use this yaml for checking the runtime details of the cluster - scheduler ip

```
ads opctl run -f train.yaml | tee info.yaml
```

```
(ads-opctl) Ismails-MacBook-Pro:Distributed-Training isyed$ ads opctl run -f train.yaml | tee info.yaml
[REDACTED]
jobId: [REDACTED]
odfnq
mainJobRunId: [REDACTED]
workDir: [REDACTED]
workerJobRunIds:
- [REDACTED]
l3q

# ★ To stream the logs of the main job run:
# $ ads opctl watch [REDACTED]
```

This command will stream the log from logging infrastructure that you provided while defining the cluster inside train.yaml in the example above.

```
ads opctl watch <job runid>
```

The Job Status and Any Outputs will be displayed as the Job Runs.

```
(ads-opctl) Ismails-MacBook-Pro:Distributed-Training isyed$ ads opctl run -f train.yaml | tee info.yaml
[REDACTED]
jobId: [REDACTED]
odfnq
mainJobRunId: [REDACTED]
workDir: [REDACTED]
workerJobRunIds:
- [REDACTED]
l3q

# ★ To stream the logs of the main job run:
# $ ads opctl watch [REDACTED]

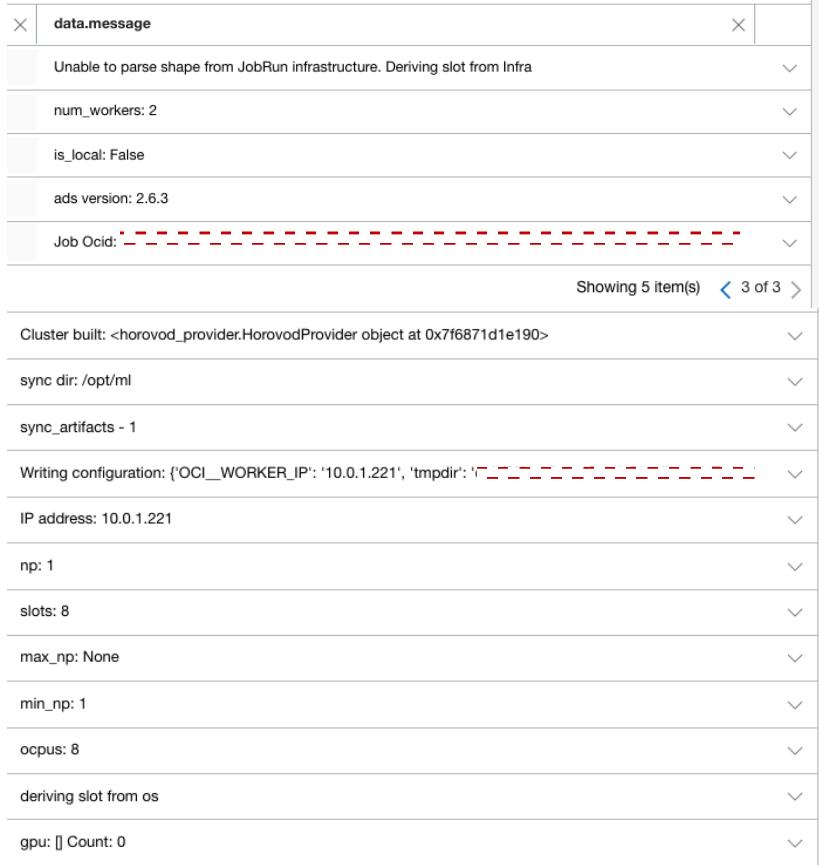
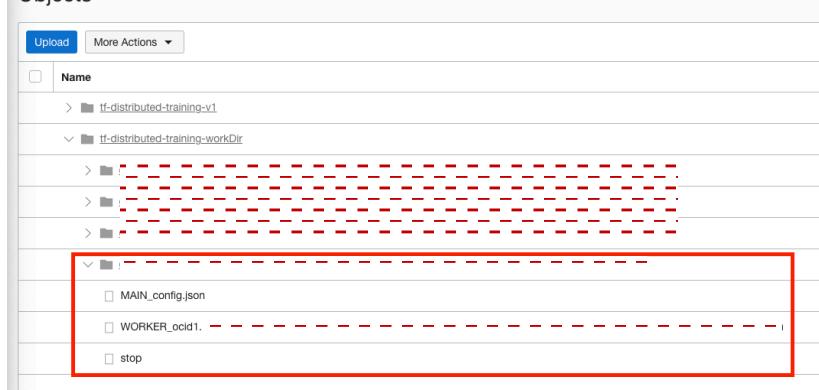
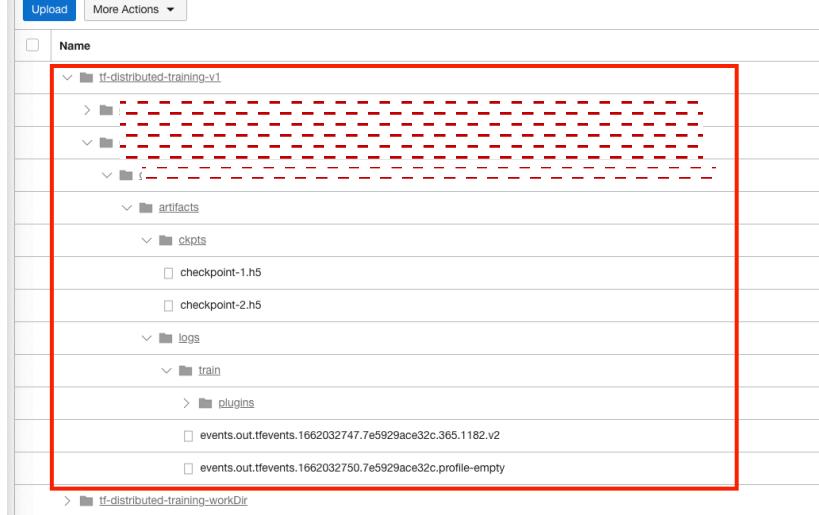
[REDACTED] 09-01 12:00:49 - [11]<stdout>:Epoch 2/2
2022-09-01 12:00:49 - [7]<stdout>:Epoch 2/2
2022-09-01 12:00:49 - [6]<stdout>:Epoch 2/2
2022-09-01 12:00:49 - [3]<stdout>:Epoch 2/2
2022-09-01 12:00:49 - [8]<stdout>:Epoch 2/2
2022-09-01 12:00:49 - [13]<stdout>:Epoch 2/2
2022-09-01 12:00:49 - [9]<stdout>:Epoch 2/2
2022-09-01 12:00:49 - [0]<stdout>:Epoch 2/2
[5]<stdout>: 4/31 [=>.....] - ETA: 25s - loss: 0.3176 - accuracy: 0...623
[3]<stdout>: 0/31 [=====>.....] - ETA: 20s - loss: 0.2926 - accuracy: 0.91096
nothing to sync in /opt/ml/-----] - ETA: 16s - loss: 0.3043 - accuracy: 0.9124
[12]<stdout>: 18/31 [=====>.....] - ETA: 1... - loss: 0.2647 - accuracy: 0.9162033
2022-09-01 12:01:56.411000+00:00 - Job Run SUCCEEDED, Job run artifact execution in progress.
(ads-opctl) Ismails-MacBook-Pro:Distributed-Training isyed$
```

Check runtime configurations

```
ads opctl distributed-training show-config -f info.yaml
```

```
(ads-opctl) Ismails-MacBook-Pro:Distributed-Training isyed$ 
(ads-opctl) Ismails-MacBook-Pro:Distributed-Training isyed$ ads opctl distributed-training show-config -f info.yaml
Main Info:
OCI__MAIN_IP: [REDACTED]
OCI__SLOTS: '8'
OCI__WORKER_IP: [REDACTED]
tmpdir: [REDACTED]
.datasci: [REDACTED]
```

<p>Have a look in the OCI console to see the jobs succeeded.</p> <p>OCI Menu > Analytics & AI > Data Science > Your Project > Jobs > Select your Distributed Job.</p>	
<p>We can see the Job Runs for both the Scheduler and Worker Nodes.</p> <p>Both have Succeeded.</p> <p>Let's Click on the Worker Node – worker-0-0</p>	<p>Job runs in [REDACTED] Compartment</p>
<p>Let's view the Log created within OCI Logging.</p>	

<p>We can see information such as:</p> <ul style="list-style-type: none"> • ADS Version • Num Nodes • Num OCPU • Location of Config Files 	 <pre> data.message ----- Unable to parse shape from JobRun infrastructure. Deriving slot from Infra num_workers: 2 is_local: False ads version: 2.6.3 Job Ocid: [REDACTED] ----- Showing 5 item(s) < 3 of 3 > Cluster built: <horovod_provider.HorovodProvider object at 0x7f6871d1e190> sync dir: /opt/ml sync_artifacts - 1 Writing configuration: {'OCI_WORKER_IP': '10.0.1.221', 'tmpdir': [REDACTED]} IP address: 10.0.1.221 np: 1 slots: 8 max_np: None min_np: 1 ocpus: 8 deriving slot from os gpu: [] Count: 0 </pre>																																
<p>We can also go view our Work Directory within Object Storage where the Config Files are: These can be downloaded.</p>	<p>Objects</p>  <table border="1"> <thead> <tr> <th>Upload</th> <th>More Actions</th> </tr> </thead> <tbody> <tr> <td><input type="checkbox"/></td> <td>Name</td> </tr> <tr> <td colspan="2">> tf-distributed-training-v1</td> </tr> <tr> <td colspan="2"> < tf-distributed-training-workDir</td> </tr> <tr> <td colspan="2"> > [REDACTED]</td> </tr> <tr> <td colspan="2"> > :</td> </tr> <tr> <td colspan="2"> MAIN_config.json</td> </tr> <tr> <td colspan="2"> WORKER_ocid1.</td> </tr> <tr> <td colspan="2"> stop</td> </tr> </tbody> </table>	Upload	More Actions	<input type="checkbox"/>	Name	> tf-distributed-training-v1		< tf-distributed-training-workDir		> [REDACTED]		> [REDACTED]		> [REDACTED]		> [REDACTED]		> :		MAIN_config.json		WORKER_ocid1.		stop									
Upload	More Actions																																
<input type="checkbox"/>	Name																																
> tf-distributed-training-v1																																	
< tf-distributed-training-workDir																																	
> [REDACTED]																																	
> [REDACTED]																																	
> [REDACTED]																																	
> [REDACTED]																																	
> :																																	
MAIN_config.json																																	
WORKER_ocid1.																																	
stop																																	
<p>We can check our Output Logs and Model Checkpoints also stored within our Object Storage.</p>	 <table border="1"> <thead> <tr> <th>Upload</th> <th>More Actions</th> </tr> </thead> <tbody> <tr> <td><input type="checkbox"/></td> <td>Name</td> </tr> <tr> <td colspan="2">> tf-distributed-training-v1</td> </tr> <tr> <td colspan="2"> < tf-distributed-training-workDir</td> </tr> <tr> <td colspan="2"> > [REDACTED]</td> </tr> <tr> <td colspan="2"> > [REDACTED]</td> </tr> <tr> <td colspan="2"> > artifacts</td> </tr> <tr> <td colspan="2"> < ckpts</td> </tr> <tr> <td colspan="2"> checkpoint-1.h5</td> </tr> <tr> <td colspan="2"> checkpoint-2.h5</td> </tr> <tr> <td colspan="2"> < logs</td> </tr> <tr> <td colspan="2"> < train</td> </tr> <tr> <td colspan="2"> > plugins</td> </tr> <tr> <td colspan="2"> events.out.tfevents.1662032747.7e5929ace32c.365.1182.v2</td> </tr> <tr> <td colspan="2"> events.out.tfevents.1662032750.7e5929ace32c.profile-empty</td> </tr> <tr> <td colspan="2">> tf-distributed-training-workDir</td> </tr> </tbody> </table>	Upload	More Actions	<input type="checkbox"/>	Name	> tf-distributed-training-v1		< tf-distributed-training-workDir		> [REDACTED]		> [REDACTED]		> artifacts		< ckpts		checkpoint-1.h5		checkpoint-2.h5		< logs		< train		> plugins		events.out.tfevents.1662032747.7e5929ace32c.365.1182.v2		events.out.tfevents.1662032750.7e5929ace32c.profile-empty		> tf-distributed-training-workDir	
Upload	More Actions																																
<input type="checkbox"/>	Name																																
> tf-distributed-training-v1																																	
< tf-distributed-training-workDir																																	
> [REDACTED]																																	
> [REDACTED]																																	
> artifacts																																	
< ckpts																																	
checkpoint-1.h5																																	
checkpoint-2.h5																																	
< logs																																	
< train																																	
> plugins																																	
events.out.tfevents.1662032747.7e5929ace32c.365.1182.v2																																	
events.out.tfevents.1662032750.7e5929ace32c.profile-empty																																	
> tf-distributed-training-workDir																																	

