

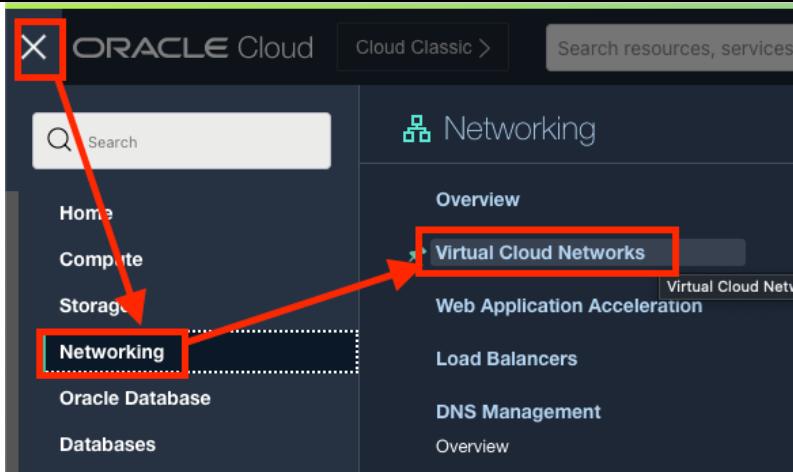
## Distributed Training with OCI DS Jobs

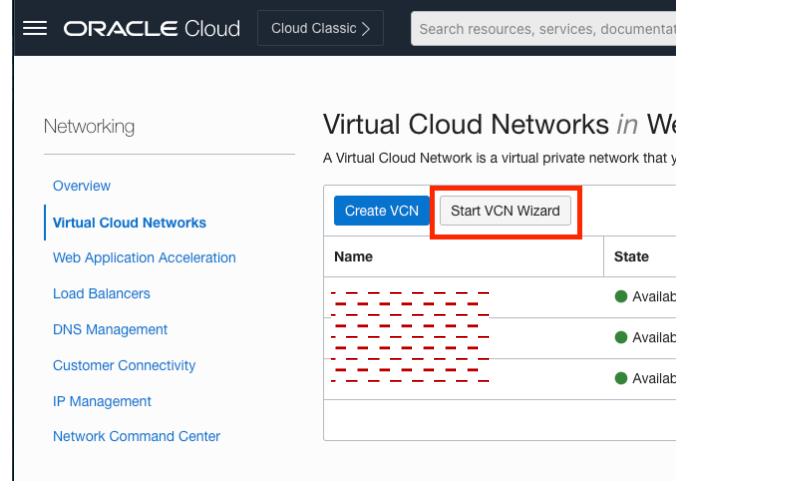
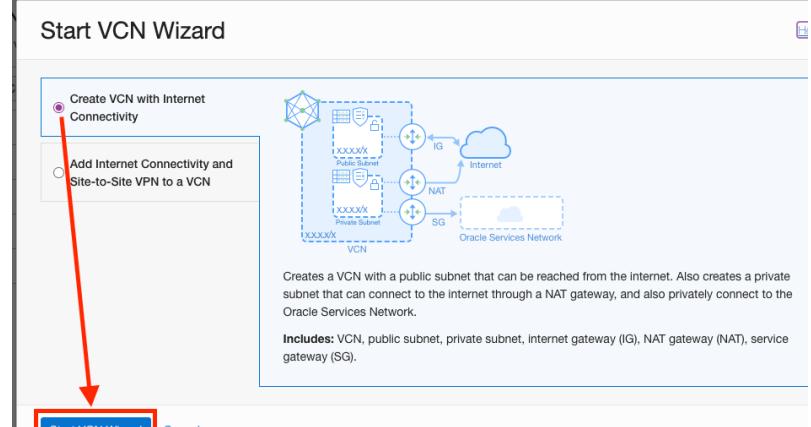
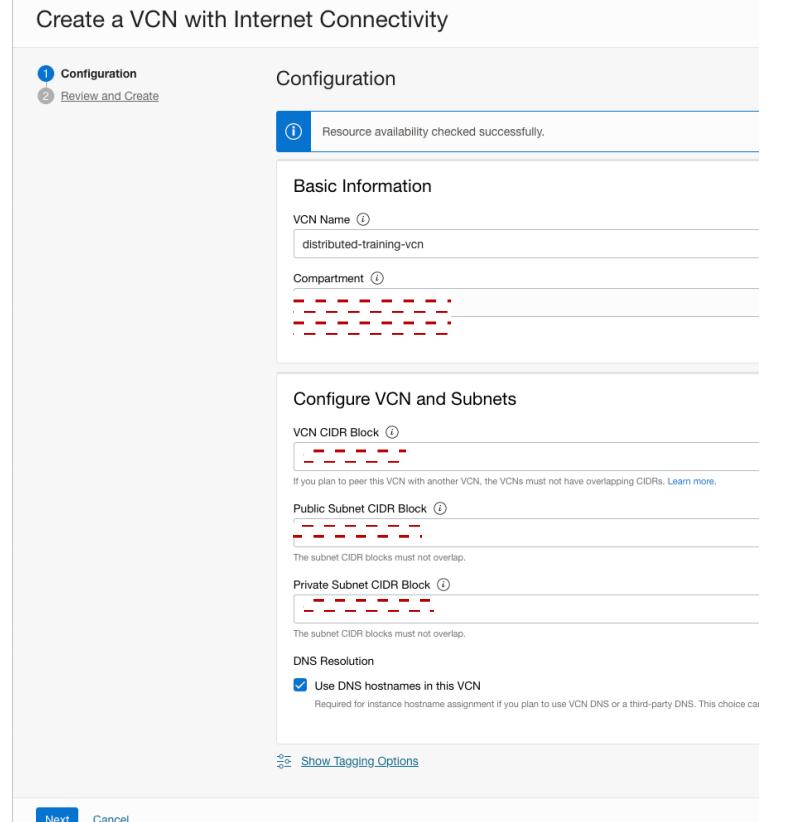
**Referenced Documentation:** [https://github.com/oracle-samples/oci-data-science-ai-samples/tree/master/distributed\\_training/notebooks](https://github.com/oracle-samples/oci-data-science-ai-samples/tree/master/distributed_training/notebooks)

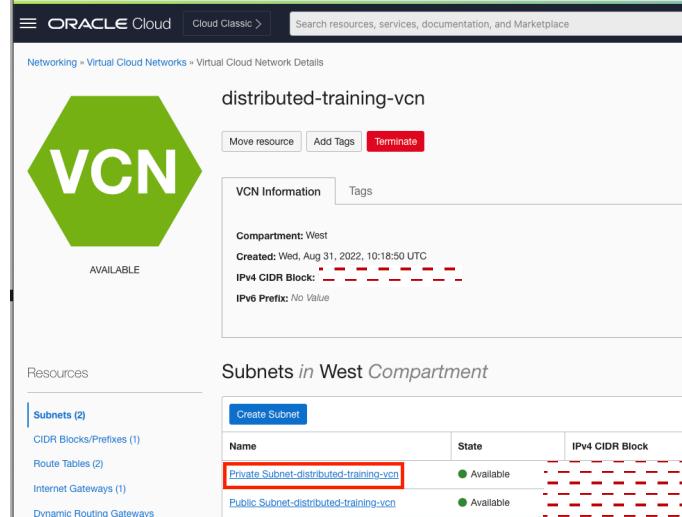
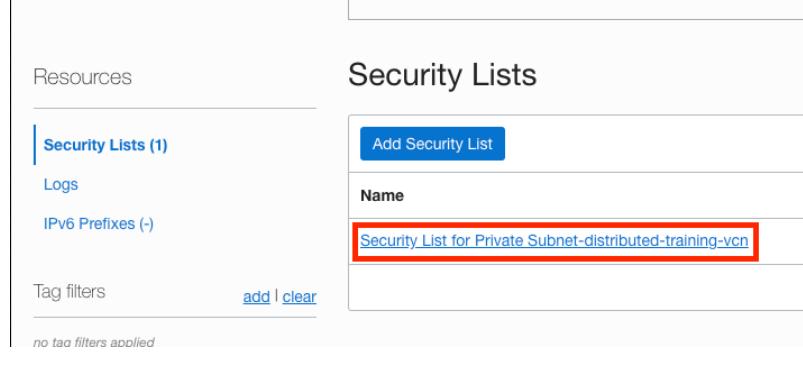
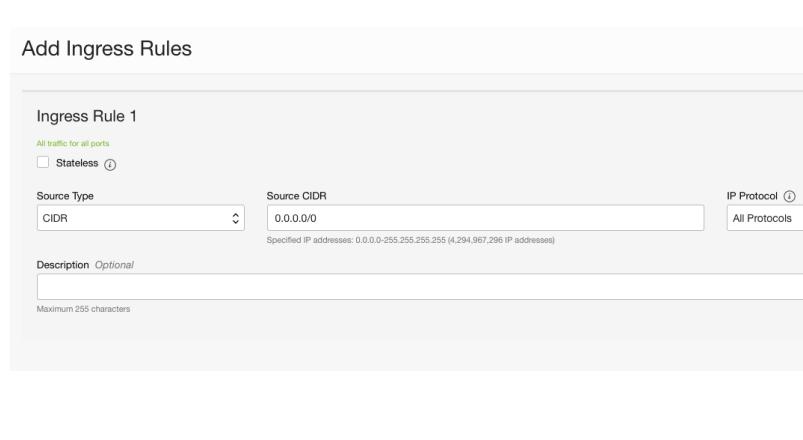
In this document, we will only be looking at the Horovod TensorFlow Distributed Training Example.

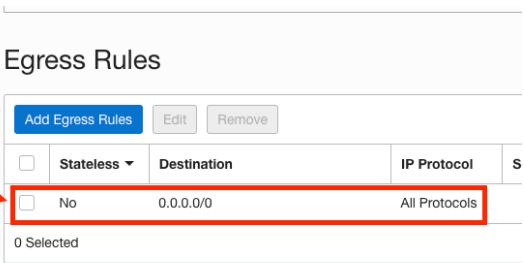
Here are the key pre-requisites that you would need to execute before you can proceed to building a distributed workload docker image and execute it using **odsc-jobs**.

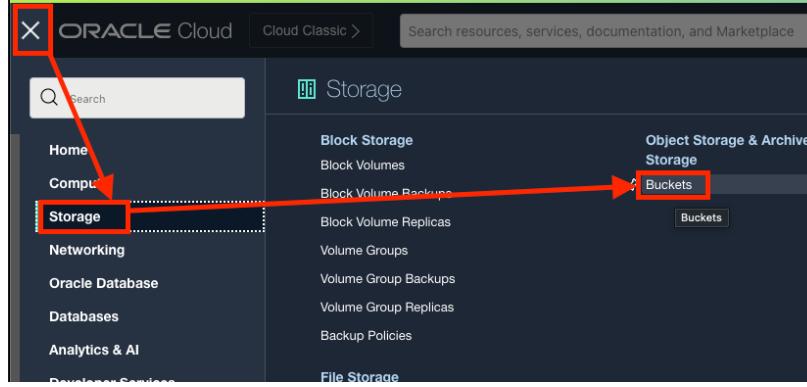
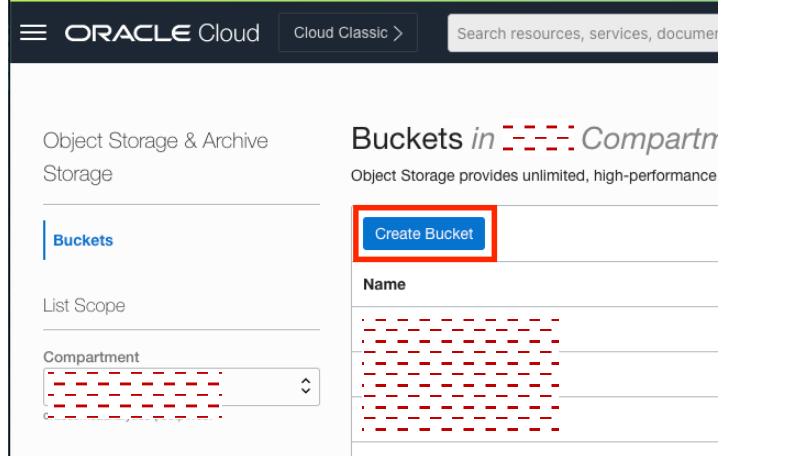
1. Configure Network. Required for the inter node communication.
2. OCI policies. Required for accessing OCI services by the distributed job.
3. Install ads-opctl. Required for packaging training script and launching odsc distributed job.

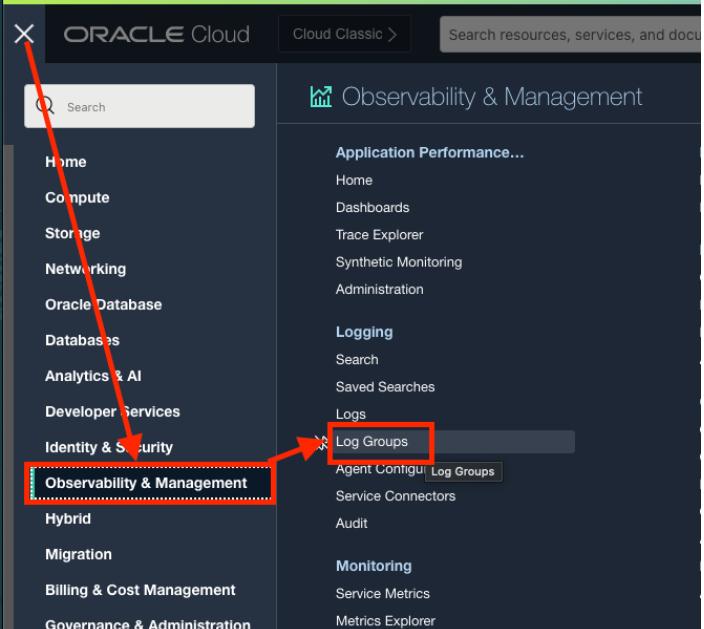
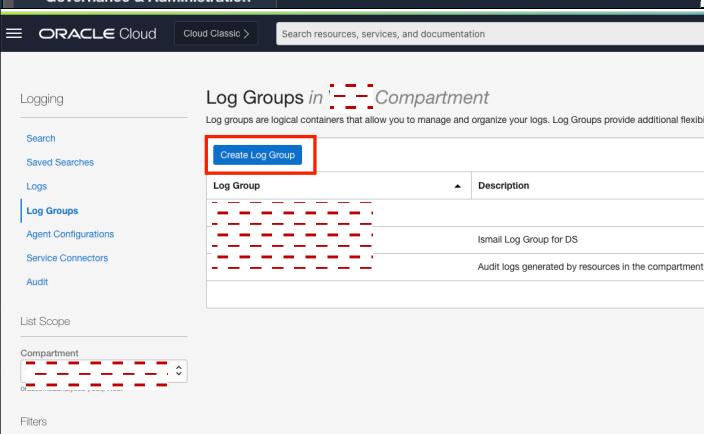
Networking Pre-Requisite	
Step	Screenshot
<p>Create a new VCN with a Private Subnet, that allows all traffic within the subnet.</p> <p>This will allow the nodes to all communicate with each other.</p> <p><b>OCI Console Menu &gt; Networking &gt; Virtual Cloud Networks.</b></p>	

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

<p>We need to now allow all communication between the nodes in the Subnet.</p> <p><b>Click on the <i>Private Subnet</i> that has just been created.</b></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 two subnets: "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><b>Click <i>Add Ingress Rule</i>.</b></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 it shows three existing rules, each with a checkbox, a stateless dropdown set to "No", and a source column with a dashed red line.</p>
<p>Add a Rule with <b>Source CIDR as 0.0.0.0/0</b> (This can also be locked down to the Subnet itself, however for simplicity we will leave it as such).</p> <p><b>Set IP Protocol as <i>All Protocols</i>.</b></p> <p><b>Click <i>Add Ingress Rules</i>.</b></p>	 <p>The screenshot shows the "Add Ingress Rules" dialog. It includes fields for "Source CIDR" (set to 0.0.0.0/0) and "IP Protocol" (set to All Protocols). Other fields include "Stateless" (unchecked), "Description" (Optional), and "Maximum 255 characters".</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>	 <table border="1"> <thead> <tr> <th></th> <th>Stateless</th> <th>Destination</th> <th>IP Protocol</th> </tr> </thead> <tbody> <tr> <td><input type="checkbox"/></td> <td>No</td> <td>0.0.0.0/0</td> <td>All Protocols</td> </tr> </tbody> </table> <p>0 Selected</p>		Stateless	Destination	IP Protocol	<input type="checkbox"/>	No	0.0.0.0/0	All Protocols
	Stateless	Destination	IP Protocol						
<input type="checkbox"/>	No	0.0.0.0/0	All Protocols						

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><b>OCI Menu &gt; Storage Buckets.</b></p>	
<p>Click <b>Create Bucket</b></p>	

<p>Enter a <b>Bucket Name</b> and Click on <b>Create</b>.</p>	<h2>Create Bucket</h2> <p>Bucket Name ---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</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 <a href="#">Events Service</a>.</p> <p><input type="checkbox"/> Uncommitted Multipart Uploads Cleanup Create a lifecycle rule to automatically delete uncommitted multipart uploads older than</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 <b>visit Menu &gt; Observability &amp; Management &gt; Log Groups</b></p>	 <p>A screenshot of the Oracle Cloud Observability &amp; Management interface. The left sidebar shows various service links like Home, Compute, Storage, Networking, Oracle Database, Databases, Analytics &amp; AI, Developer Services, Identity &amp; Security, and a red-highlighted 'Observability &amp; Management' link. The main content area is titled 'Observability &amp; Management' and contains sections for Application Performance Monitoring, Logging, and Monitoring. A red arrow points from the 'Observability &amp; Management' link in the sidebar to the 'Logs' section in the main content area, which is also highlighted with a red box.</p>
<p><b>Click on 'Create Log Group'</b> to create a new log group to store our logs in.</p>	 <p>A screenshot of the 'Log Groups' creation page. The top header says 'Log Groups in --- Compartment'. Below it is a 'Create Log Group' button. A table lists existing log groups: 'Ismail Log Group for DS' (Description) and 'Audit logs generated by resources in the compartment' (Description). The 'Compartment' dropdown is set to the same red-highlighted compartment as the previous screenshot. The left sidebar includes 'Logs', 'Log Groups' (which is red-highlighted), 'Agent Configurations', 'Service Connectors', and 'Audit'.</p>

<p><b>Enter</b> a name for your Log Group and click '<b>Create Log Group</b>' 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><b>OCI Menu &gt; Identity &amp; Security &gt; Policies.</b></p>	
<p>Click on <b>Create Policy</b>.</p>	

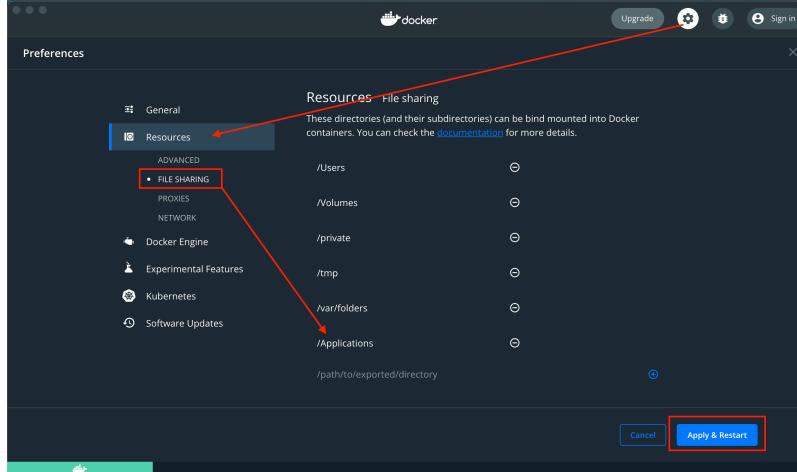
Give the Policy a **Name**, **Description** and enter the following policies, under **Manual Editor**.

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

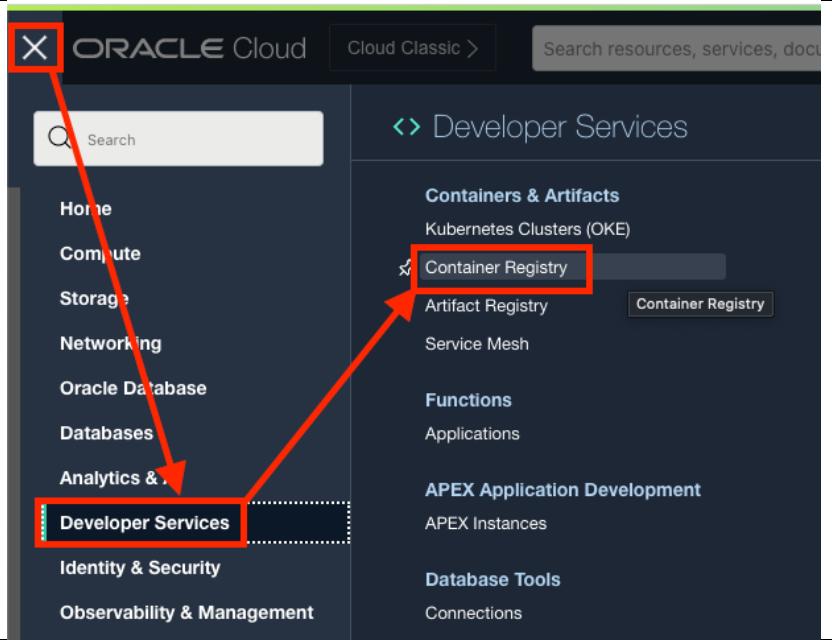
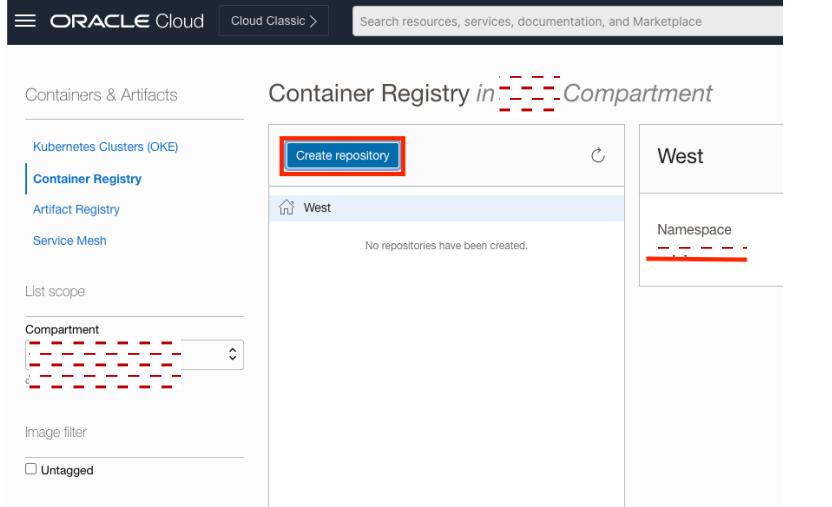
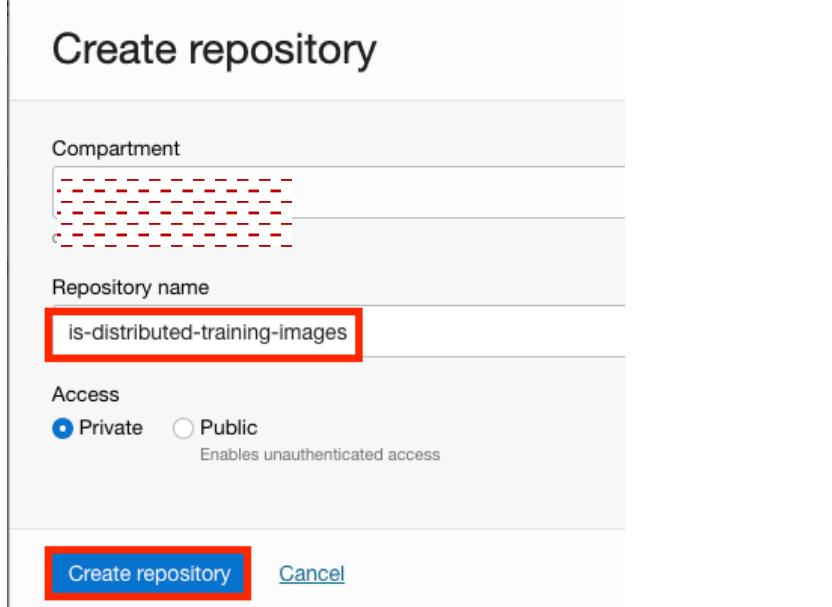
Click **Create**.

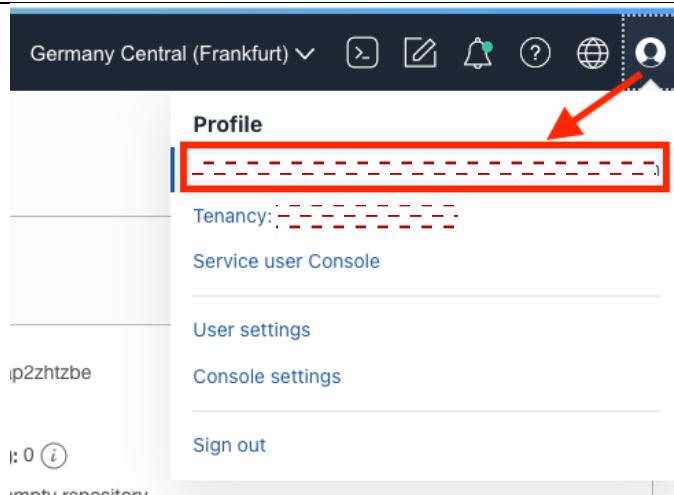
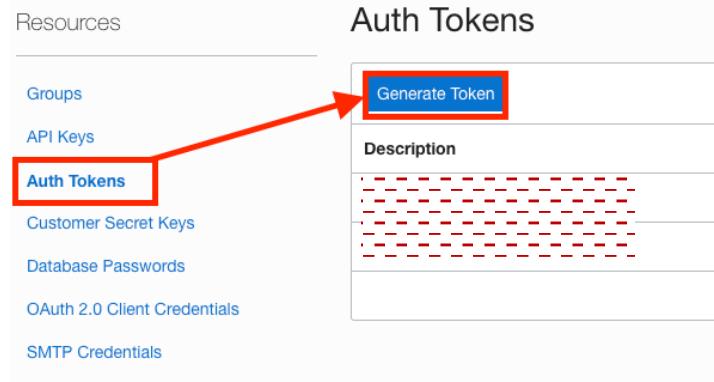
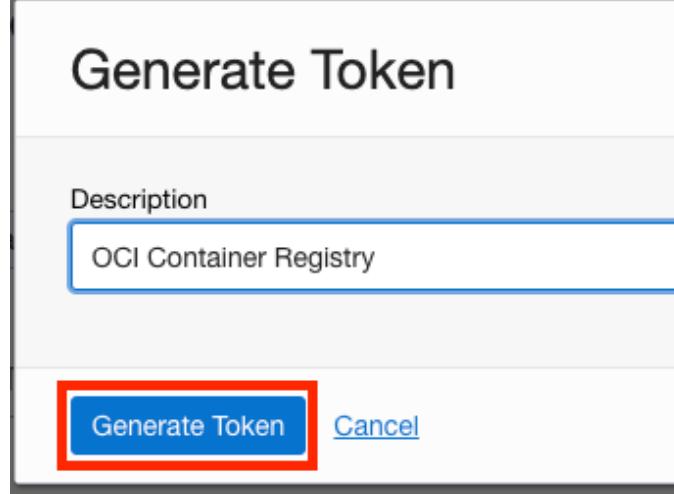
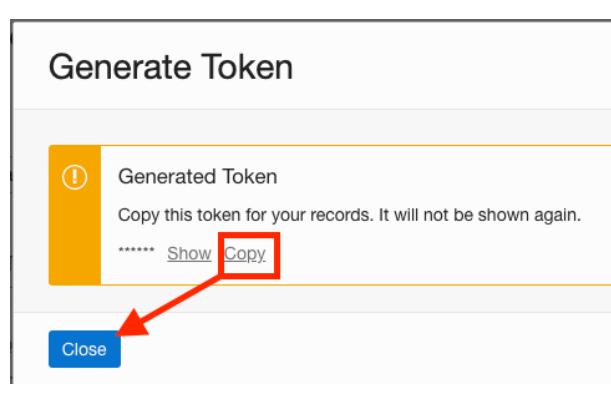
```
Allow group <group> to manage repos in compartment <compartment>
Allow group <group> to manage data-science-jobs in compartment <compartment>
Allow group <group> to manage data-science-job-runs in compartment <compartment>
Allow group <group> to use virtual-network-family in compartment <compartment>
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>}
```

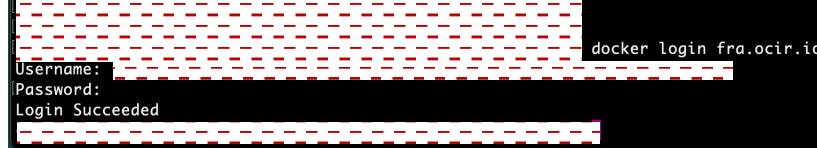
Install ads-opctl and Docker	
Steps	Screenshot
<p>This assumes you already have Python3 installed locally on your machine.</p> <p>I will install ADS within a Python3.7 Conda Environment I have already created.</p> <pre>python3 -m pip install "oracle-ads[opctl]"</pre> <pre>python3 -m pip install "oracle-ads[opctl]" --force-reinstall --no-deps --install-option="--enable-cli"</pre>	<pre>\$ python3 -m pip install "oracle-ads[opctl]" --force-reinstall --no-deps --install-option="--enable-cli" ... Successfully installed oracle-ads-2.6.3</pre> <p>Need to run both commands on the left.</p>
<p><b>Download Docker and Run Installation from:</b></p> <p><a href="https://docs.docker.com/get-docker">https://docs.docker.com/get-docker</a></p>	<p>The screenshot shows the Docker documentation website. The main heading is 'Get Docker'. It includes a note about updating to Docker Desktop terms and a brief description of Docker as an open platform for development, shipping, and running applications. Below this, there are three sections for Docker Desktop: 'Docker Desktop for Mac' (with an Apple icon), 'Docker Desktop for Windows' (with a Windows icon), and 'Docker Desktop for Linux' (with a Linux icon). Each section has a brief description and a link to download.</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><b>Visit Settings &gt; Resources &gt; File Sharing</b></p> <p><b>Add your parent folder for conda</b> in my case /Applications.</p> <p><b>Click Apply &amp; Restart.</b></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><b>cd</b> <b>~/Distributed_Training</b></p> <p>Using the ads-opctl, download and initialize the relevant distributed training framework.</p> <p><b>ads opctl distributed-training init --framework horovod-tensorflow --version v1</b></p>	<pre>\$ 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</pre>

Prepare Docker Image	
Steps	Screenshots
<p>All the docker image related artifacts are located under: <b><i>oci_dist_training_artifacts/horovod/v1/</i></b></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 <b><i>ads opctl distributed-training init -framework horovod &lt;pytorch/tensorflow&gt;</i></b></p>	<p>Continued...</p> <p>Upload your Tensorflow training script (<b><i>train.py</i></b>) within your current directory on your local machine as this will then be copied into the <b><i>/code</i></b> folder inside docker image once built.</p> <p><b>Note:</b> 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 <b><i>oci_dist_training_artifacts/horovod/v1/conda-&lt;pytorch/tensorflow&gt;-&lt;cpu/gpu&gt;.yaml</i></b>. If your code requires additional dependency, update this file.</p> <p><b>Note:</b> While updating <b><i>conda-&lt;pytorch/tensorflow&gt;-&lt;cpu/gpu&gt;.yaml</i></b> 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><b>OCI Menu &gt; Developer Services &gt; Container Registry.</b></p>	
<p><b>Make note of the Namespace to your right, you will need this later.</b></p> <p>Click on <b>Create Repository</b>.</p>	
<p>Give it a <b>Name</b> and Click <b>Create Repository</b>.</p>	

We then need to Generate Auth Token for Authentication from our Local Docker to Container Registry.  Click on your <b>Profile Icon</b> then your <b>Username</b> .	
Navigate to <b>Auth Tokens</b> and Click on <b>Generate Token</b> .	
Give it a <b>Description</b> and Click on <b>Generate Token</b> .	
Copy the <b>Token</b> 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 <b><i>docker login &lt;region-key&gt;.ocir.io</i></b>, where <b>&lt;region-key&gt;</b> corresponds to the key for the Container Registry Region you're using.</p>	 <p><b><i>docker login fra.ocir.io</i></b></p> <p>Username: [REDACTED] Password: [REDACTED] Login Succeeded</p> <p><b><i>docker login fra.ocir.io for Frankfurt region</i></b></p>
<p>When prompted for a username, enter your username in the format <b>&lt;tenancy-namespace&gt;/&lt;username&gt;</b></p> <p>If your tenancy is federated with Oracle IDCS, use the format <b>&lt;tenancy-namespace&gt;/oracleiden.titycloudservice/&lt;username&gt;</b></p>	 <p><b><i>docker login fra.ocir.io</i></b></p> <p>Username: [REDACTED] Password: [REDACTED] Login Succeeded</p>
<p>When prompted for a password, <b>enter the Auth Token</b> you copied earlier.</p>	 <p><b><i>docker login fra.ocir.io</i></b></p> <p>Username: [REDACTED] Password: [REDACTED] Login Succeeded</p>

We will set some environment variables for the Docker Image Name and Tag

```
export
IMAGE_NAME=fra.ocir.io/<name-space>/<repo-name>

export TAG=latest
```

Format of the image name must be **<region-key>.ocir.io/<tenancy-namespace>/<repo-name>:<tag>** where:

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

We can then build the Docker Image:

```
docker build -t
$IMAGE_NAME:$TAG \
-f
oci_dist_training_artifacts/horovod/v1/docker/tensorflow.cpu.Dockerfile
```

Include the (dot) at the end of the command.

Ensure we are pointing to the correct Docker File dependant of CPU or GPU

This will take a while to build.

```
$ docker build -t $IMAGE_NAME:$TAG -f oci_dist_training_artifacts/horovod/v1/docker/tensorflow.cpu.Dockerfile
[+] Building 799.2s (21/21) FINISHED
--> [internal] load build definition from tensorflow.cpu.Dockerfile
--> [internal] load .dockerrcignore
--> [internal] load context: 2B
--> [internal] load metadata for docker.io/library/oraclelinux:8
--> [ 1/16] FROM docker.io/library/oraclelinux:@sha256:56ecbe6dd70c1cd66706ea3317c08bee7084ae5f4
--> [ 2/16] RUN yum -y update && yum install -y gcc-c++ cmake && ssh-keygen -AqN
--> [ 3/16] RUN curl -Lo /tmp/miniconda-installer.sh https://repo.anaconda.com/miniconda/Mini
--> [ 4/16] RUN mkdir -p /etc/datasciencce/horovod
--> [ 5/16] COPY oci_dist_training_artifacts/horovod/v1/conda-tensorflow-cpu.yaml /etc/datascie
--> [ 6/16] RUN conda env create -f /etc/datasciencce/horovod/conda-tensorflow-cpu.yaml &&
--> [ 7/16] RUN conda init bash && echo "conda activate env" >> /root/.bashrc
--> [ 8/16] RUN HOROVOD_WITH_GLOO=1 HOROVOD_WITH_MPI=1 HOROVOD_WITH_TENSORFLOW=1 pip install
--> [ 9/16] RUN mkdir -p /etc/datasciencce/horovod
--> [10/16] WORKDIR /etc/datasciencce/horovod
--> [11/16] COPY oci_dist_training_artifacts/horovod/v1/*.py ..
--> [12/16] COPY oci_dist_training_artifacts/horovod/v1/*.sh ..
--> [13/16] RUN chmod u+x *.sh
--> [14/16] RUN mkdir -p /code
--> [15/16] COPY . /code
--> [16/16] RUN mkdir /opt/ml
--> exporting to image
--> exporting layers
--> V
--> F
Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
```

<p>Once complete, we can <b>list</b> our Docker Images</p> <p><b>docker image ls</b></p>	<table border="1"> <thead> <tr> <th>REPOSITORY</th> <th>SIZE</th> <th>TAG</th> <th>IMAGE ID</th> <th>CREATED</th> </tr> </thead> <tbody> <tr> <td>ml-job</td> <td>3.45GB</td> <td>latest</td> <td>[REDACTED]</td> <td>About a minute ago</td> </tr> <tr> <td>3.97GB</td> <td>3.97GB</td> <td>latest</td> <td>[REDACTED]</td> <td>4 months ago</td> </tr> <tr> <td>docker/getting-started</td> <td>28.8MB</td> <td>latest</td> <td>[REDACTED]</td> <td>6 months ago</td> </tr> </tbody> </table>	REPOSITORY	SIZE	TAG	IMAGE ID	CREATED	ml-job	3.45GB	latest	[REDACTED]	About a minute ago	3.97GB	3.97GB	latest	[REDACTED]	4 months ago	docker/getting-started	28.8MB	latest	[REDACTED]	6 months ago
REPOSITORY	SIZE	TAG	IMAGE ID	CREATED																	
ml-job	3.45GB	latest	[REDACTED]	About a minute ago																	
3.97GB	3.97GB	latest	[REDACTED]	4 months ago																	
docker/getting-started	28.8MB	latest	[REDACTED]	6 months ago																	
<p>Push the local Docker Image to the OCI Registry.</p> <p><b>docker push \$IMAGE_NAME:\$TAG</b></p> <p>Where <b>IMAGE_NAME</b> refers to the repo you want to push to.</p>	<pre>\$ docker push \$IMAGE_NAME:\$TAG The push refers to repository [REDACTED] Pushed:  Pushed:  Pushed:  Pushed:  Pushed:  Pushed:  Pushed:  Pushed:  Pushed:  Pushing 598.2MB/1.943GB Pushed:  Pushed:  Pushed:  Pushing 286.1MB/748.7MB Pushing 246MB/414.6MB Pushed: </pre>																				
<p>We can then view our Image in the OCI Registry.</p>	<p>Container Registry in [REDACTED] Compartment</p> <table border="1"> <thead> <tr> <th>Create repository</th> <th>latest</th> </tr> </thead> <tbody> <tr> <td>West</td> <td>Full path: [REDACTED]</td> </tr> <tr> <td>+ is-distributed-training-images</td> <td>Pushed: [REDACTED]</td> </tr> <tr> <td>latest</td> <td>OCIID: [REDACTED]</td> </tr> <tr> <td></td> <td>Digest: [REDACTED]</td> </tr> <tr> <td></td> <td>Repository: [REDACTED]</td> </tr> <tr> <td></td> <td>Namespace: [REDACTED]</td> </tr> <tr> <td></td> <td>Date create: [REDACTED]</td> </tr> </tbody> </table> <p>Signatures Layers Associated tags Scan results</p> <p>Description Verification response Date</p> <p>No items found.</p> <p>Showing</p>	Create repository	latest	West	Full path: [REDACTED]	+ is-distributed-training-images	Pushed: [REDACTED]	latest	OCIID: [REDACTED]		Digest: [REDACTED]		Repository: [REDACTED]		Namespace: [REDACTED]		Date create: [REDACTED]				
Create repository	latest																				
West	Full path: [REDACTED]																				
+ is-distributed-training-images	Pushed: [REDACTED]																				
latest	OCIID: [REDACTED]																				
	Digest: [REDACTED]																				
	Repository: [REDACTED]																				
	Namespace: [REDACTED]																				
	Date create: [REDACTED]																				
<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><b>docker run --rm -it --privileged --entrypoint bash \$IMAGE_NAME:\$TAG</b></p>	<pre>int bash \$IMAGE_NAME:\$TAG [REDACTED] \$ pwd /etc/dataservice/horovod \$ ls cluster_runner_horovod.py discover-hosts.sh horovod_provider.py run.sh start_worker.sh sync.sh conda-tensorflow-cpu.yaml discover_workers.py ip_helper.py run_horovod.py </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
```

```
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>:
[1,0]<stdout>:11493376/11490434 [=====] - 6s 1us/step - loss: 2.3076 - accuracy: 0.1016
[1,1] [=====] - 4s 4s/step - loss: 2.3012 - accuracy: 0.1094[1,0]<stdout>:
[1,0]<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>:
[1,0]<stdout>:250/250 [=====] - 146s 585ms/step - loss: 0.2128 - accuracy: 0.9344[1,0]<stdout>:::
[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 [=====] - 143s 573ms/step - loss: 0.0796 - accuracy: 0.9759[1,0]<stdout>:::
[1,1]<stdout>:250/250 [=====] - 143s 573ms/step - loss: 0.0801 - accuracy: 0.9762
```

Once done, exit the container with the exit command.

```
exit
```

```
#  
#  
# exit  
exit
```

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.



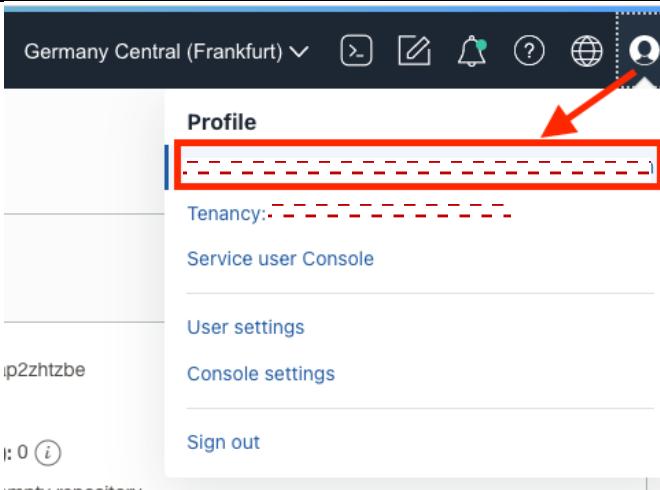
This will print/validate the Job and Job run configuration without launching the actual job.

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

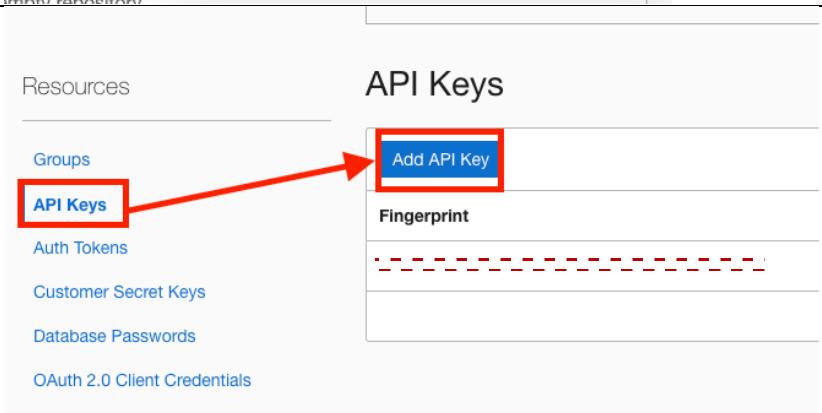
```
$ ads opctl run -f train.yaml --dry-run
--Entering dryrun mode--
Creating Job with payload:
kind: job
spec:
  infrastructure:
    kind: infrastructure
    spec:
      blockStorageSize: 50
      compartmentId: [REDACTED]
  sq
  displayName: horovod_tf
  jobInfrastructureType: ME_STANDALONE
  jobType: DEFAULT
  logGroupId: [REDACTED]
  langVfp6q
  projectId: [REDACTED]
  shapeName: VM.Standard2.4
  subnetId: [REDACTED]
  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
```

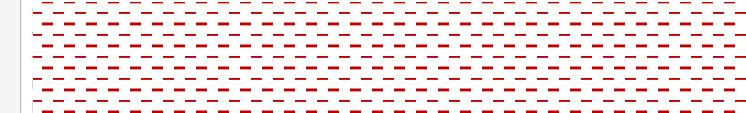
Before you submit the job to OCI DS Jobs, you need to add your Private Key File and Config file.

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



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



<p><b>Generate API Key Pair.</b></p> <p><b>Download Private Key.</b></p> <p><b>Click Add.</b></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. <a href="#">Learn more</a></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, <a href="#">change the file permissions</a> so o</p> <p><a href="#">Download Private Key</a>   <a href="#">Download Public Key</a></p> <p><a href="#">Add</a>   <a href="#">Cancel</a></p>
<p><b>Copy the config file and save locally along with the private key to <code>~/.oci</code>.</b></p> <p><b>Ensure to update the path to the private key in the config file.</b></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 <b>Default</b> profile in your config profile, you'll need to perform some additional steps. <a href="#">Learn more</a></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><a href="#">Close</a></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 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 [~/.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> []:
Configuration saved at [~/.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
```

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
```

```
[OCI]
oci_config = ~/.oci/config
oci_profile = DEFAULT

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

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 run -f train.yaml | tee info.yaml
[...]
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 run -f train.yaml | tee info.yaml
[...]
jobId: [REDACTED]
odfnq
mainJobRunId: [REDACTED]
workDir: [REDACTED]
workerJobRunIds:
- [REDACTED]
l3q

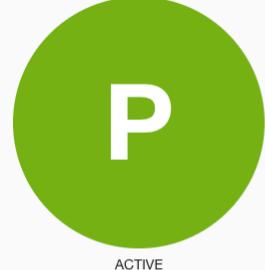
# ★ To stream the logs of the main job run:
# $ ads opctl watch [REDACTED]

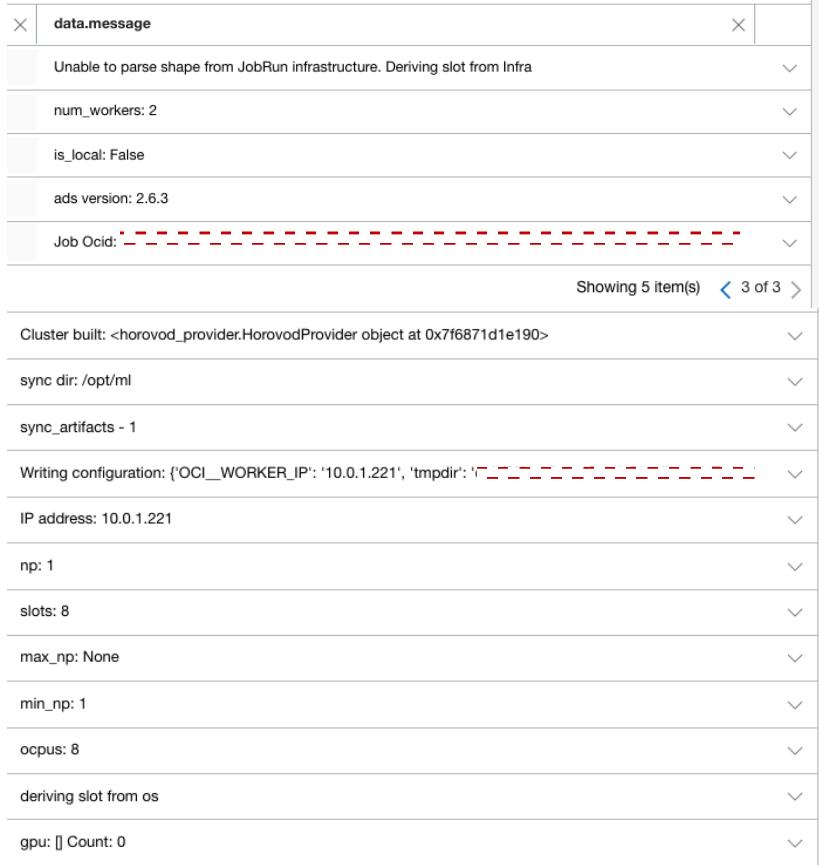
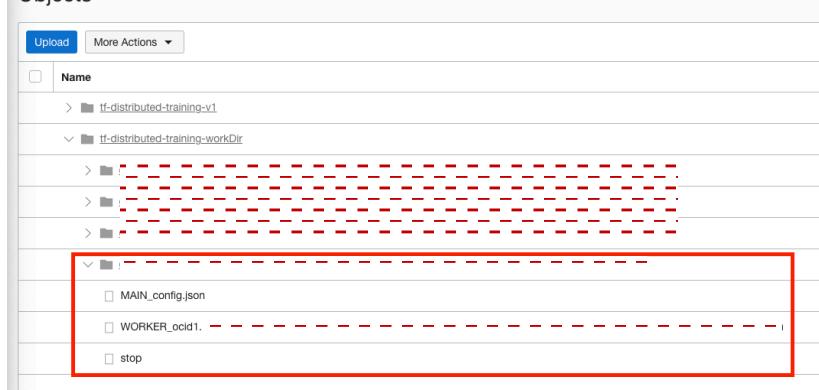
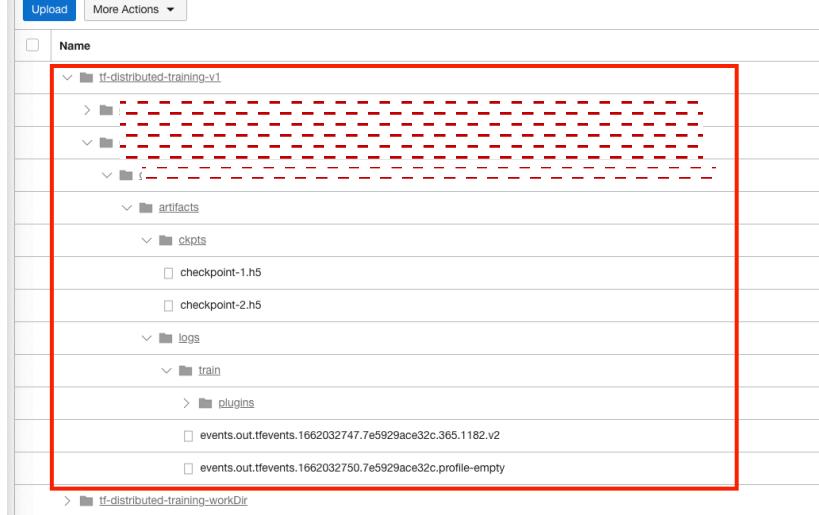
2022-09-01 12:00:49 - [11]<stdout>:Epoch 2/2
2022-09-01 12:00:49 - [7]<stdout>:Epoch 2/2
2022-09-01 12:00:49 - [6]<stdout>:Epoch 2/2
2022-09-01 12:00:49 - [3]<stdout>:Epoch 2/2
2022-09-01 12:00:49 - [8]<stdout>:Epoch 2/2
2022-09-01 12:00:49 - [13]<stdout>:Epoch 2/2
2022-09-01 12:00:49 - [9]<stdout>:Epoch 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>: 10/31 [=====>.....] - ETA: 20s - loss: 0.2926 - accuracy: 0.91096
nothing to sync in /opt/ml/mnt-----] - 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.
```

Check runtime configurations

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

```
$ 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><b>OCI Menu &gt; Analytics &amp; AI &gt; Data Science &gt; Your Project &gt; Jobs &gt; Select your Distributed Job.</b></p>	 <p>Data Science » Projects » Project details » Jobs</p> <p>IS</p> <p>Edit Move resource Add tags Delete</p> <p>Project information Tags</p> <p>Description: [REDACTED] Created by: [REDACTED]</p> <p>Resources</p> <ul style="list-style-type: none"> <li>Notebook sessions</li> <li><b>Jobs</b> (highlighted with a red box)</li> <li>Models</li> <li>Model deployments</li> </ul> <p>Jobs in IS Compartment</p> <table border="1"> <thead> <tr> <th>Name</th> <th>State</th> </tr> </thead> <tbody> <tr> <td>horovod_tf_run_4</td> <td>Active</td> </tr> <tr> <td>horovod_tf_run_3</td> <td>Active</td> </tr> <tr> <td>horovod_tf_run_2</td> <td>Active</td> </tr> </tbody> </table>	Name	State	horovod_tf_run_4	Active	horovod_tf_run_3	Active	horovod_tf_run_2	Active
Name	State								
horovod_tf_run_4	Active								
horovod_tf_run_3	Active								
horovod_tf_run_2	Active								
<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 IS Compartment</p> <table border="1"> <thead> <tr> <th>Name</th> <th>State</th> </tr> </thead> <tbody> <tr> <td>worker-0-0</td> <td>Succeeded</td> </tr> <tr> <td>worker-main</td> <td>Succeeded</td> </tr> </tbody> </table>	Name	State	worker-0-0	Succeeded	worker-main	Succeeded		
Name	State								
worker-0-0	Succeeded								
worker-main	Succeeded								
<p>Let's view the Log created within OCI Logging.</p>	<p>Data Science » Projects » Project details » Jobs » Job runs » Job run details</p> <p>worker-0-0</p> <p>Clone Edit Cancel Move resource More actions ▾</p> <p>Job run Runtime configuration Tags</p> <p>General information</p> <p>OCID: [REDACTED] Creator: [REDACTED]</p> <p>Time accepted: Thu, Sep 1, 2022, 11:40:59 UTC Time started: Thu, Sep 1, 2022, 11:44:16 UTC Time finished: Thu, Sep 1, 2022, 11:48:08 UTC</p> <p>Logging details</p> <p>Log group: DS_Log_Group_Ismail Log: [REDACTED]</p>								

<p>We can see information such as:</p> <ul style="list-style-type: none"> <li>• ADS Version</li> <li>• Num Nodes</li> <li>• Num OCPU</li> <li>• Location of Config Files</li> </ul>	 <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) &lt; 3 of 3 &gt;  Cluster built: &lt;horovod_provider.HorovodProvider object at 0x7f6871d1e190&gt; 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">&gt; tf-distributed-training-v1</td> </tr> <tr> <td colspan="2">  &lt; tf-distributed-training-workDir</td> </tr> <tr> <td colspan="2">    &gt; [REDACTED]</td> </tr> <tr> <td colspan="2">    &gt; :</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">&gt; tf-distributed-training-v1</td> </tr> <tr> <td colspan="2">  &lt; tf-distributed-training-workDir</td> </tr> <tr> <td colspan="2">    &gt; [REDACTED]</td> </tr> <tr> <td colspan="2">    &gt; [REDACTED]</td> </tr> <tr> <td colspan="2">    &gt; artifacts</td> </tr> <tr> <td colspan="2">      &lt; 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">      &lt; logs</td> </tr> <tr> <td colspan="2">      &lt; train</td> </tr> <tr> <td colspan="2">      &gt; 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">&gt; 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																																	

