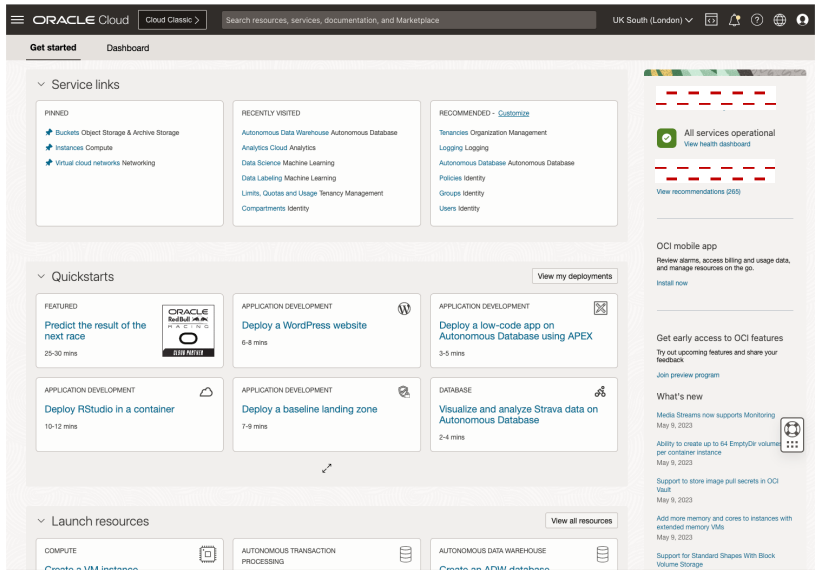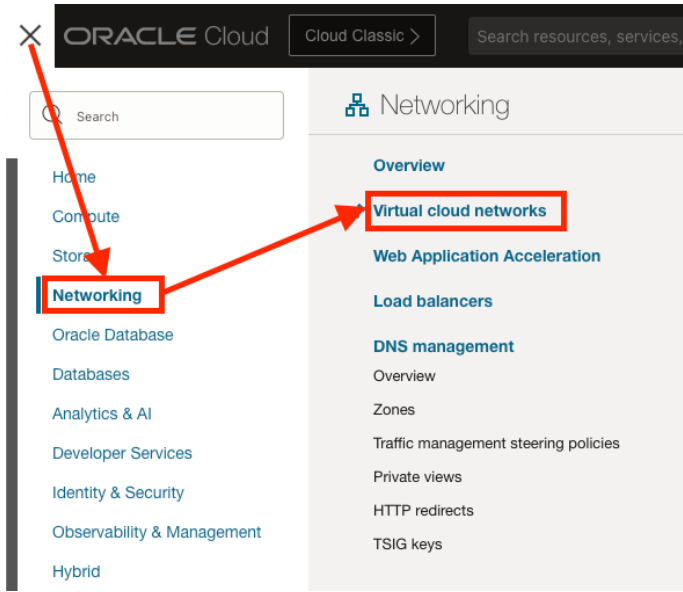# Provisioning OCI AI 'all-in-one' Data Science Image

Description: The purpose of this guide is to provision the **OCI AI 'all-in-one' Data Science** Marketplace Image within OCI and set up the Jupyter Server and an Anaconda Environment to execute your Data Science Notebooks.

Step	Screenshots
Login to the OCI Cloud Console at: cloud.oracle.com	
First, we will create a new Network for the Data Science VM to live within. Navigate from the **OCI Menu > Networking > Virtual Cloud Networks.**	

In this tutorial we will create a basic **Virtual Cloud Network** with a **Public Subnet** with **Internet Connectivity.** Click on **Start VCN Wizard.**	
Select **Create VCN with Internet Connectivity.** Select **Start VCN Wizard.**	
Enter **VCN Name.** Select **Compartment.** Select **VCN CIDR Block Range.**	

Enter CIDR Block Ranges for both your **Public and Private Subnet.** Click **Next.**	
Review the remaining default options. By default, you'll be able to SSH into the Public Network. Click **Create.** This will take a minute to Create.	
Once your VCN is created, we will provision our VM Image inside the Public Network. From the **OCI Menu > Compute > Instances.**	
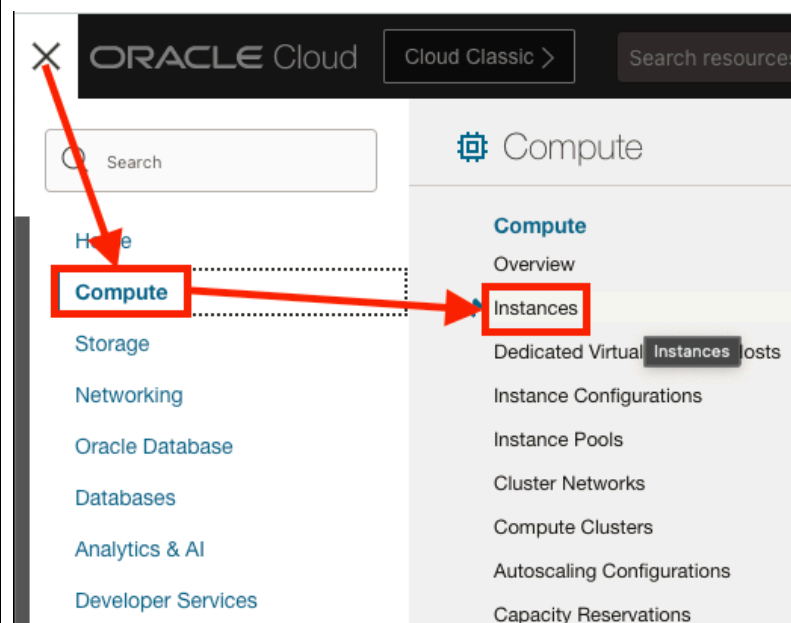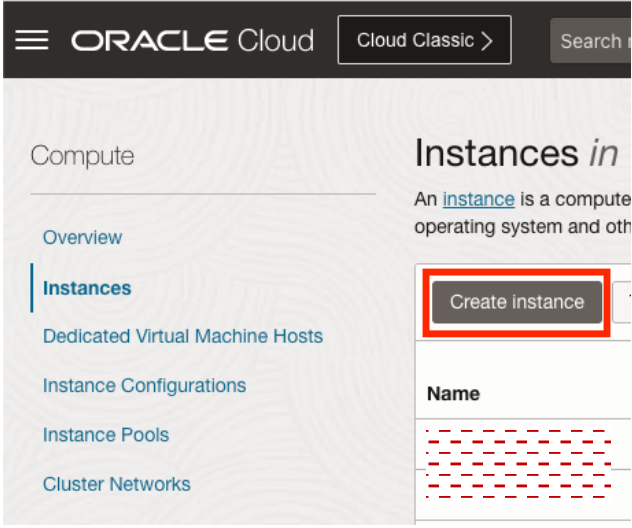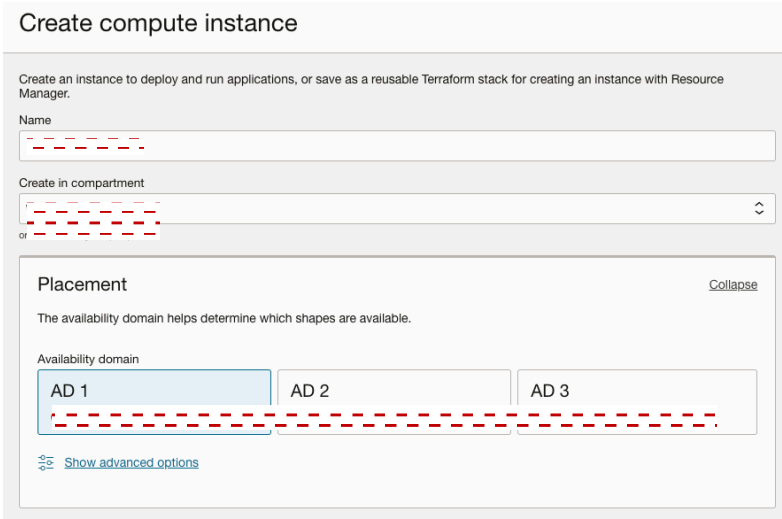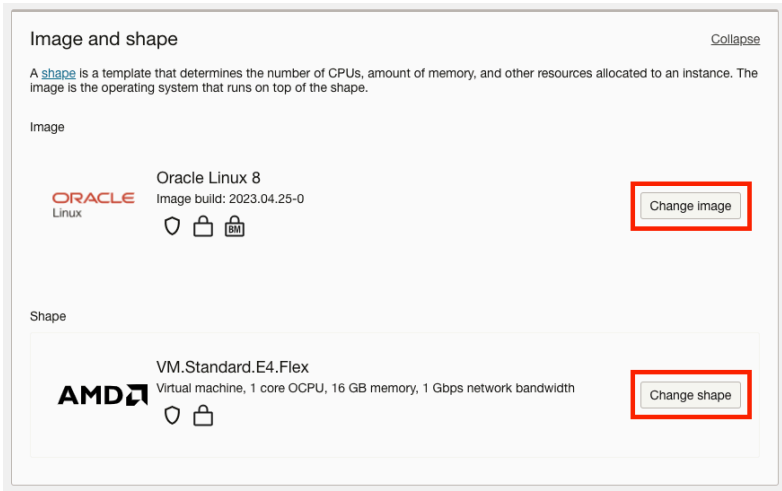
Click on **Create Instance.**	
Enter **Name.** Select **Compartment.** Choose **Availability Domain.**	
We will change both the Image and Shape. First the Image, **Click on Change Image.**	

Select **Marketplace.**

Select **AI 'all-in-one' Data Science Image Intel/AMD.**

You can select the GPU version if deploying to GPUs.

Read and Accept the Terms and Restrictions.

Click **Select Image.**

Then we will change the Shape.

Select **Virtual Machine.**

Select **AMD.**

Select Shape Name as **VM.Standard.E4.Flex**

Select OCPU as **4.**

Click **Select Shape.**

Now time to define the Network we created earlier. Choose **Select existing virtual cloud network.** Select the **VCN we created earlier.** Choose **Select existing subnet.** Select the **Public Subnet we created earlier.** Select **Assign a public IPv4 address.**	**Networking** Collapse Networking is how your instance connects to the internet and other resources in the Console. To make sure you can connect to your instance, assign a public IP address to the instance. Primary network ◉ Select existing virtual cloud network ○ Create new virtual cloud network ○ Enter subnet OCID Virtual cloud network in ___ (Change compartment) Subnet An IP address from a public subnet and an internet gateway on the VCN are required to make this instance accessible from the internet. ◉ Select existing subnet ○ Create new public subnet Subnet in ___ ⓘ (Change compartment) Public IPv4 address ◉ Assign a public IPv4 address ○ Do not assign a public IPv4 address ⓘ If you're not sure whether you need a public IP address, you can always assign one later. ⚙ Show advanced options
We will then generate a public/private key pair. Select **Generate a key pair for me.** Click **Save private key.** Click **Save public key.**	**Add SSH keys** Generate an SSH key pair to connect to the instance using a Secure Shell (SSH) connection, or upload a public key that you already have. ◉ Generate a key pair for me ○ Upload public key files (.pub) ○ Paste public keys ○ No SSH keys ⓘ Download the private key so that you can connect to the instance using SSH. It will not be shown again. ✓ Save private key ✓ Save public key
Leave all the other options as default. Click **Create.**	**Boot volume** A boot volume is a detachable device that contains the image used to boot the compute instance. ☐ Specify a custom boot volume size Volume performance varies with volume size. Default boot volume size: 128.0 GB. When you specify a custom boot volume size, service limits apply. ☐ Use in-transit encryption Encrypts data in transit between the instance, the boot volume, and the block volumes. ☐ Encrypt this volume with a key that you manage By default, Oracle manages the keys that encrypt this volume, but you can choose a key from a vault that you have access to if you want greater control over the key's lifecycle and how it's used. How do I manage my own encryption keys? **Live migration** ◉━━ The instance is live migrated to a healthy physical VM host without any disruption. Use events to track the progress. If live migration isn't successful, reboot migration is used. When disabled, a notification is sent for the maintenance event, and the instance is only live migrated if you do not proactively reboot the instance before the due date. ⚙ Show advanced options Create Save as stack Cancel

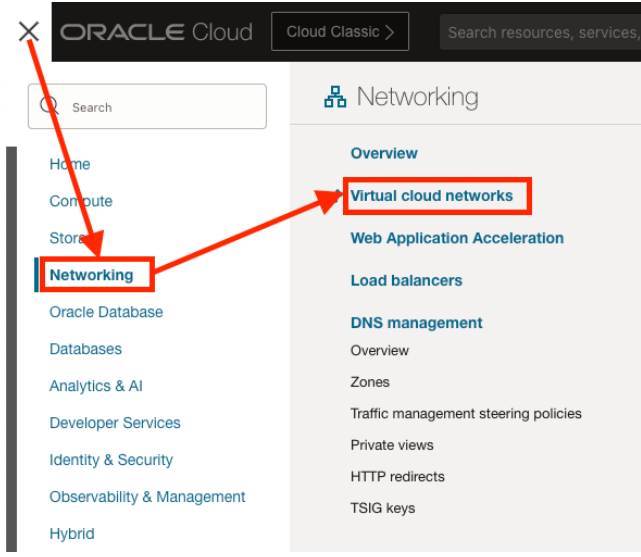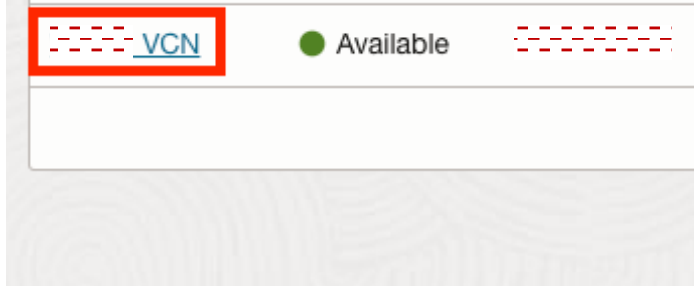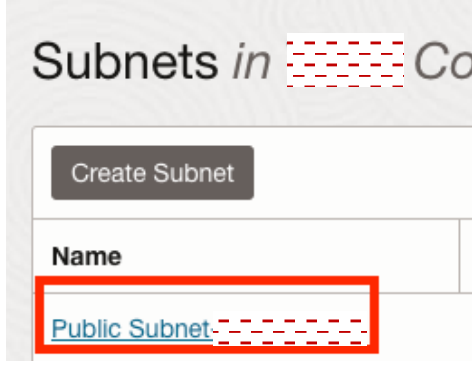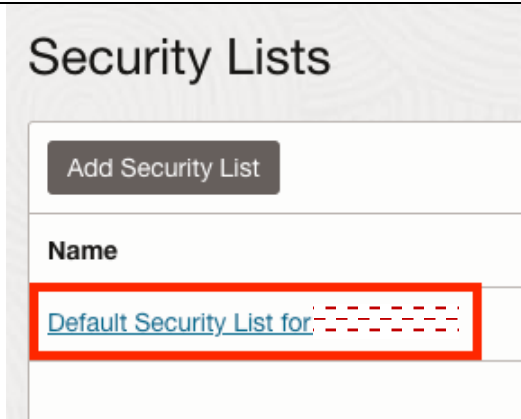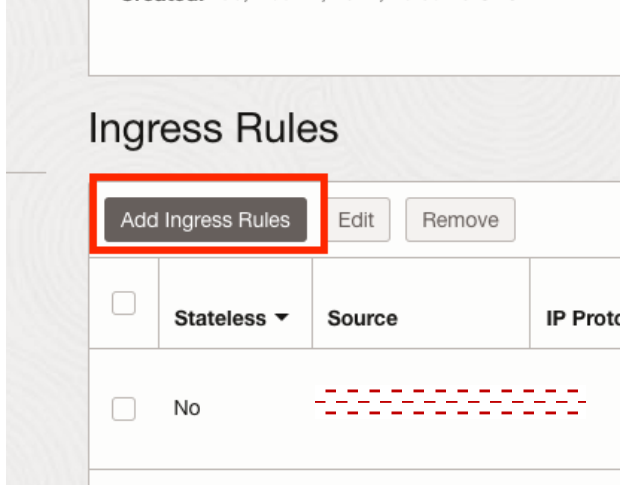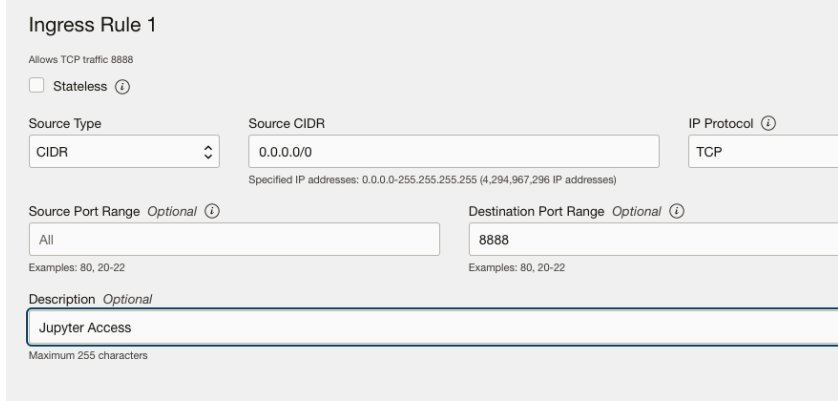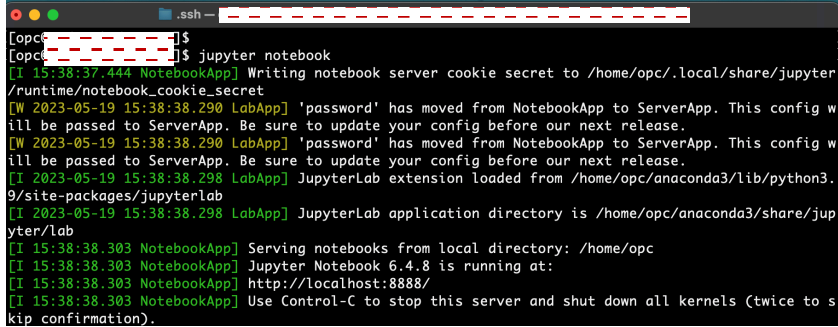|	
This will take a few minutes to provision. While it is provisioning, make note of a few of the details being displayed. **Public IP Address** **Username** You can also open up the usage instructions using the link above the Public IP Address - https://cloud.oracle.com/marketplace/application/134110504/usageInformation?region=eu-frankfurt-1 I will be referencing this when continuing.	**Instance access** You connect to a running Linux instance using a Secure Shell (SSH) connection. You'll need the private key from the SSH key pair that was used to create the instance. Usage information for this image Public IP address: _ _ _ _ _ _ _ _ _ _ Username: _ _ _ _
While the provisioning is taking place, I have opened a Terminal on my laptop and renamed and updated the private key permissions to 600. **chmod 600 <private-key-file>**	```(base) Ismails-MBP:Downloads isyed$ chmod 600 ___ private.key (base) Ismails-MBP:Downloads isyed$ ls -la total 32 drwx------@ 7 isyed staff 224 19 May 16:12 . drwxr-xr-x+ 68 isyed staff 2176 17 May 15:05 .. -rw-r--r--@ 1 isyed staff 6148 19 May 14:56 .DS_Store drwxr-xr-x 11 isyed staff 352 7 Apr 2022 .ipynb_checkpoints -rw------- 1 isyed staff 0 6 Jul 2019 .localized -rw-------@ 1 isyed staff 1675 19 May 16:08 ___ private.key -rw-r--r--@ 1 isyed staff 399 19 May 16:08 ___ public.pub (base) Ismails-MBP:Downloads isyed$```

If using a Mac, I would recommend moving your keys to the **~/.ssh** folder where all keys are stored. **mv <private-key-file> ~/.ssh** **mv <public-key-file> ~/.ssh** Navigate to your .ssh directory. **cd ~/.ssh**	```(base) Ismails-MBP:Downloads isyed$
(base) Ismails-MBP:Downloads isyed$
(base) Ismails-MBP:Downloads isyed$ mv ___private.key ~/.ssh
(base) Ismails-MBP:Downloads isyed$ mv ___public.pub ~/.ssh
(base) Ismails-MBP:Downloads isyed$
(base) Ismails-MBP:Downloads isyed$ cd ~/.ssh```
We will now test connecting into the VM. Let's SSH into the VM. **ssh -i <private-key> <user>@<public-ip>**	```(base) Ismails-MBP:.ssh isyed$
(base) Ismails-MBP:.ssh isyed$
(base) Ismails-MBP:.ssh isyed$ ssh -i _____
The authenticity of host _____.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added _____ to the list of known hosts.
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Thu Sep 1 21:05:13 2022 from _____
[opc@_____]$
[opc@_____]$
[opc@_____]$ pwd
/home/opc
[opc@_____]$``` |
| We will reset the Jupyter Notebook Password.

jupyter notebook password | ```[opc@_____]$
[opc@_____]$
[opc@_____]$ jupyter notebook password
Enter password:
Verify password:
[NotebookPasswordApp] Wrote hashed password to _____
[opc@_____]$
[opc@_____]$``` |

Now before connecting into the Jupyter Notebook, we will need to open **port 8888** on **our Subnet Security List.** Navigate back to the OCI Console, **OCI Menu > Networking > Virtual Cloud Network.**	
Select our **VCN.**	
Select our **Public Network.**	
Select our **Security List.**	

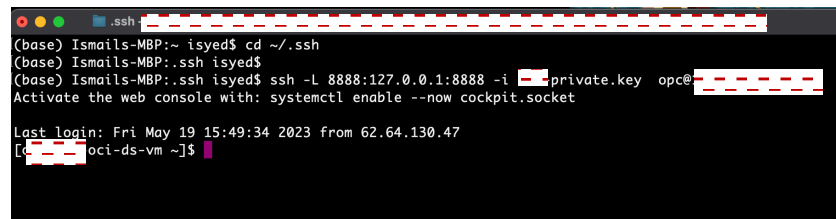|	
Click **Add Ingress Rule.**	
Add Source Type – **CIDR** Source CIDR – **0.0.0.0/0** (**access from anywhere, you can adjust as needed.**) IP Protocol – **TCP** Destination Range – **8888** Click **Add Ingress Rule.** This will allow us to connect to Port 8888 on the VM.	
We can now test creating an SSH Tunnel into the VM and access Jupyter Lab from our Local Browser. In your Terminal which is SSH'd into the VM, start the Jupyter Notebook Server. **jupyter notebook**	

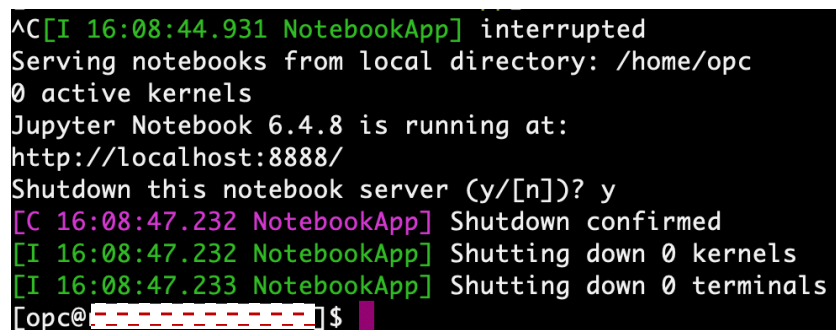|	
In another Terminal window navigate to the **~/.ssh** directory. **cd ~./ssh** Open up an SSH Tunnel to map the VM Host and Jupyter Port to the local host. **ssh -L 8888:127.0.0.1:8888 -i \<vm-private-key-file\> opc@\<vm-ip-address\>**	
Within a local browser visit the webpage: **http://localhost:8888/lab** When prompted with a password, enter the new password we created earlier. We are now into the Jupyter Environment.	
Once we can login, lets close down the Notebook Server (for now) within the SSH Session you have open. **Ctrl-C.**	
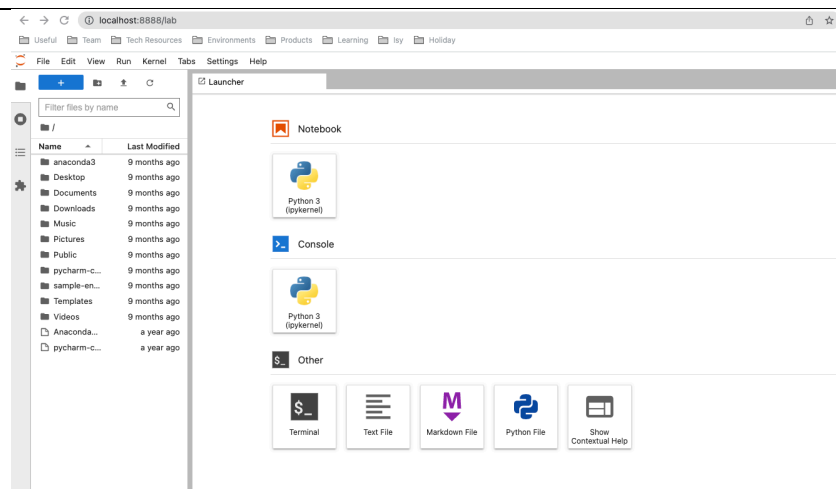
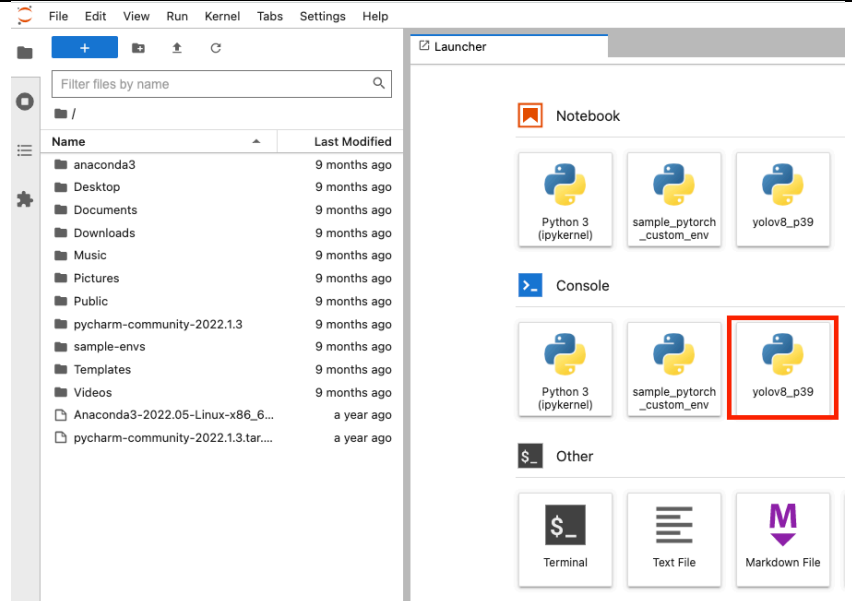|	
We will now **create** a **custom Conda Environment** to use within the Jupyter Notebook. First, we must initialise the terminal for conda. **conda init bash** Once this is done, we will have to **logout and log back into the VM via SSH.**	``` [opc@_____~]$ echo $SHELL /bin/bash [opc@_____~]$ conda init bash no change /home/opc/anaconda3/condabin/conda no change /home/opc/anaconda3/bin/conda no change /home/opc/anaconda3/bin/conda-env no change /home/opc/anaconda3/bin/activate no change /home/opc/anaconda3/bin/deactivate no change /home/opc/anaconda3/etc/profile.d/conda.sh no change /home/opc/anaconda3/etc/fish/conf.d/conda.fish no change /home/opc/anaconda3/shell/condabin/Conda.psm1 no change /home/opc/anaconda3/shell/condabin/conda-hook.ps1 no change /home/opc/anaconda3/lib/python3.9/site-packages/xontrib/conda.xsh no change /home/opc/anaconda3/etc/profile.d/conda.csh modified /home/opc/.bashrc ==> For changes to take effect, close and re-open your current shell. <== [opc@_____~]$ ▯ ```

Once logged back in we can **create a new python 3.9 environment.**
 	
conda create --name yolov8_p39 python=3.9
	``
	(base) [opc@_____~]$ conda create --name yolov8_p39 python=3.9
	Collecting package metadata (current_repodata.json): done
	Solving environment: done
	
	==> WARNING: A newer version of conda exists. <==
	  current version: 4.14.0
	  latest version: 23.3.1
	
	Please update conda by running
	
	    $ conda update -n base -c defaults conda
	
	## Package Plan ##
	
	  environment location: /home/opc/anaconda3/envs/yolov8_p39
	``

Once created, lets activate the conda. **conda activate yolov8_p39**	``` # # To activate this environment, use # # $ conda activate yolov8_p39 # # To deactivate an active environment, use # # $ conda deactivate Retrieving notices: ...working... done (base) [opc@_____]$ (base) [opc@_____]$ (base) [opc@_____vm_]$ conda activate yolov8_p39 (yolov8_p39) [opc@_____ ~]$ (yolov8_p39) [opc@_____ ~]$ ▮ ```

| We will now install two libraries to enable use within Jupyter Lab, **conda install ipykernel** **conda install nb_conda_kernels** | ```
[(yolov8_p39) [opc@_____ ~]$
[(yolov8_p39) [opc@_____ ~]$ conda install ipykernel
Collecting package metadata (current_repodata.json): done
Solving environment: done

==> WARNING: A newer version of conda exists. <==
   current version: 4.14.0
   latest version: 23.3.1

Please update conda by running

   $ conda update -n base -c defaults conda


## Package Plan ##

   environment location: /home/opc/anaconda3/envs/yolov8_p39
``` |
| Once done we will now register the conda environment for use within the Jupyter Lab Environment. **ipython kernel install -- user -- name=yolov8_p39** | ```
(yolov8_p39) [opc@_____ ~]$
(yolov8_p39) [opc@_____ ~]$
(yolov8_p39) [opc@_____ ~]$ ipython kernel install --user --name=yolov8_p39
Installed kernelspec yolov8_p39 in /home/opc/.local/share/jupyter/kernels/yolov8_p39
(yolov8_p39) [opc@_____ ~]$
(yolov8_p39) [opc@_____ ~]$
(yolov8_p39) [opc@_____ ~]$
``` |
| Switch back to the base conda environment. **conda activate base** | ```
Installed kernelspec yolov8_p39 in /home/opc/.local/share/
(yolov8_p39) [opc@_____ ~]$
(yolov8_p39) [opc@_____ ~]$
(yolov8_p39) [opc@_____ ~]$ conda activate base
(base) [opc@_____ ~]$
(base) [opc@_____ ~]$
(base) [opc@_____ ~]$
``` |
| Now let's start the Jupyter Notebook Session back up. **jupyter notebook** | ```
(base) [opc@_____ ~]$
(base) [opc@_____ ~]$ jupyter notebook
[W 2023-05-19 16:30:17.163 LabApp] 'password' has moved from NotebookApp to ServerApp. This config w
ill be passed to ServerApp. Be sure to update your config before our next release.
[W 2023-05-19 16:30:17.164 LabApp] 'password' has moved from NotebookApp to ServerApp. This config w
ill be passed to ServerApp. Be sure to update your config before our next release.
[I 2023-05-19 16:30:17.172 LabApp] JupyterLab extension loaded from /home/opc/anaconda3/lib/python3.
9/site-packages/jupyterlab
[I 2023-05-19 16:30:17.172 LabApp] JupyterLab application directory is /home/opc/anaconda3/share/jup
yter/lab
[I 16:30:17.177 NotebookApp] Serving notebooks from local directory: /home/opc
[I 16:30:17.177 NotebookApp] Jupyter Notebook 6.4.8 is running at:
[I 16:30:17.177 NotebookApp] http://localhost:8888/
[I 16:30:17.177 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to s
kip confirmation).
[W 16:30:17.180 NotebookApp] No web browser found: could not locate runnable browser.
``` |

In our local browser navigate to the Jupyter Lab homepage. http://localhost:8888/lab We can now see our new Conda environment ready for use.	

You can now create a new notebook and start running your code.