run:ai

Docker with run:ai (and Jupyter)

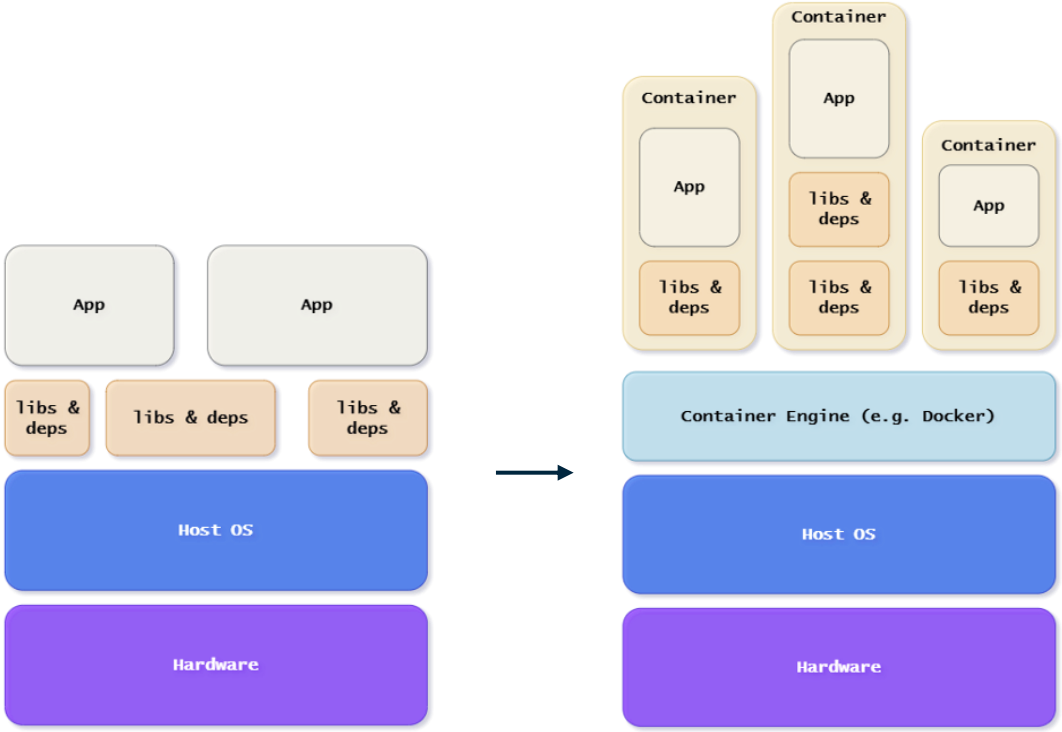
Docker intro

What is Docker?

Docker is a way to create “images.”

Image have an OS (usually linux) and software installed on them

Images can be run on any hardware as “containers”



Why use Docker?

User 1 Code

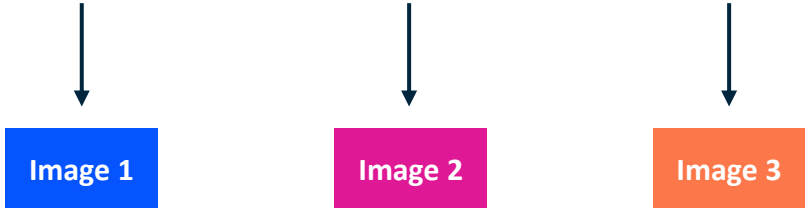
Requires
- Cudatoolkit=**11.2**
- Python=**3.7**

User 2 Code

Requires
- Cudatoolkit=**11.7**
- Python=**3.9**

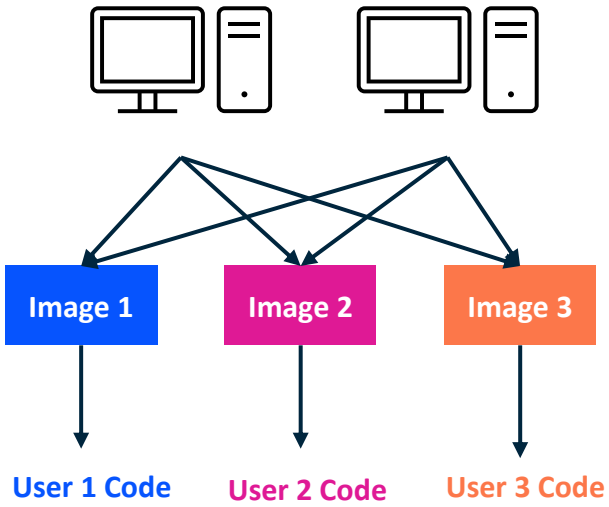
User 3 Code

Requires
- Cudatoolkit=**11.5**
- Python=**3.8**



Each user can create an image with the software they need installed on it…

… and simply run the image inside a container on any hardware with docker engine running.



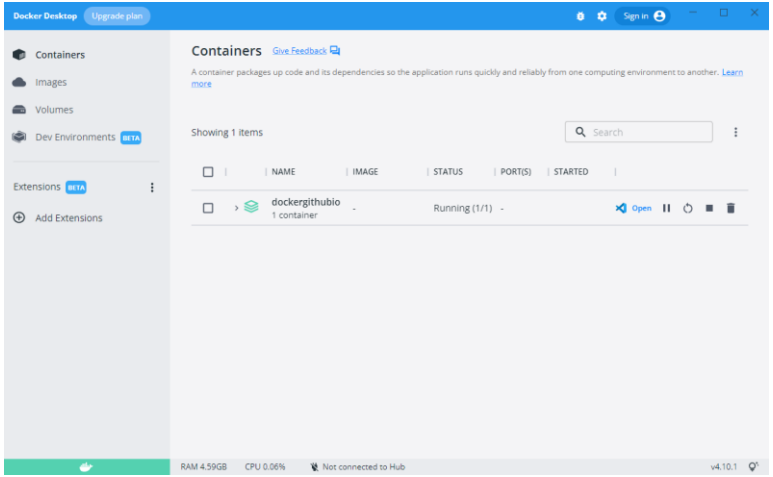
What do I need to use docker?

An image repository

I use Docker hub.

Docker software

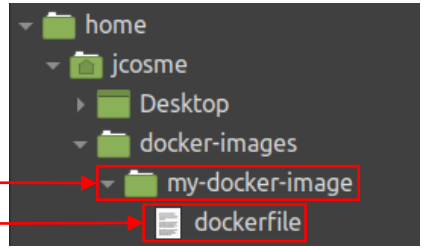
I use Docker Desktop.



https://docs.docker.com/desktop/install/linux-install/

How do I make a Docker image?

1. Make sure Docker desktop is running



2. You'll need a dedicated folder for your image
3. You'll need a file called "dockerfile" inside the folder

```
jcosme@jane:~/docker-images/my-docker-image$ pwd
/home/jcosme/docker-images/my-docker-image
jcosme@jane:~/docker-images/my-docker-image$ ls
dockerfile
jcosme@jane:~/docker-images/my-docker-image$
```

4. Navigate to the dedicated image folder in a terminal
5. Make sure the dockerfile is in the folder

6. Run this command to build

```
docker build -t {your-docker-hub-username}/{your-image-name}:{version-number} .
```

(this is what my command be)

```
docker build -t jonathancosme/my-docker-image:1.0 .
```

7. Run this to push to the repo

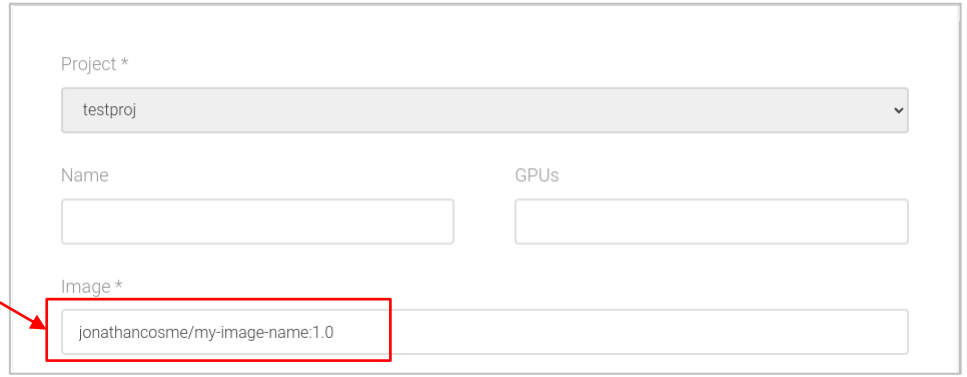
```
docker image push {your-docker-hub-username}/{your-image-name}:{version-number}
```

(this is what my command be)

```
docker image push jonathancosme/my-docker-image:1.0
```

How I use my Docker image with run:ai?

Enter the image name in the Image field of the UI



Use the **-i** flag on the CLI to specify the image

```
runai submit -i jonathancosme/my-image-name:1.0 {other arguments}
```

What is a “dockerfile?”

A “dockerfile” is a simple text file that tells docker how to build your image.

dockerfile

```
FROM condaforge/mambaforge:4.13.0-1

RUN apt-get install --yes --no-install-recommends openssh-server

RUN mamba install --yes --name base --channel conda-forge tensorflow

CMD echo "Hello world!"
```

There are **3 basic commands** to use

- **FROM** - specifies the base image to use (a **starting point**)
- **RUN** - specifies a command to run in a terminal. This is used **to install software**.
- **CMD** - specified the **default command** for the image **to run**, if no command is specified.

(yes, these are case-sensitive, and must always be in caps)

A full list and explanation of the dockerfile commands can be found here:
https://docs.docker.com/engine/reference/builder/

What is a “dockerfile?”

In this particular image,

1. We start with an image with conda and mamba preinstalled

2. Install openssh-server

dockerfile

```
FROM condaforge/mambaforge:4.13.0-1

RUN apt-get install --yes --no-install-recommends openssh-server

RUN mamba install --yes --name base --channel conda-forge tensorflow

CMD echo "Hello world!"
```

4. Tell the image to print “Hello world!” if no command is given

3. Use mamba (version of conda written in C++, so it’s much faster) to install tensorflow

About “layers”

Every time a dockerfile command
is used, a “layer” is created

dockerfile

```
FROM condaforge/mambaforge:4.13.0-1

RUN apt-get install --yes --no-install-recommends openssh-server

RUN mamba install --yes --name base --channel conda-forge tensorflow

CMD echo "Hello world!"
```

This image has **at least 4 layers**.
Keep in mind that the **base image** (FROM command) **also has layers**, so our layers will be built on top of these.
Essentially, **we** will **have number-of-layers-in-base-image + 3** layers.

About “layers”

All else equal, **more layers make the image larger**.
As a general rule, we **want images to be small**.
Pay attention to how you create your layers

dockerfile

```
FROM condaforge/mambaforge:4.13.0-1

RUN mamba install -y tensorflow

RUN mamba install -y pandas

RUN mamba install -y scikit-learn

RUN mamba install -y matplotlib

CMD echo "Hello world!"
```

dockerfile

```
FROM condaforge/mambaforge:4.13.0-1

RUN mamba install -y tensorflow \
    pandas \
    scikit-learn \
    matplotlib

CMD echo "Hello world!"
```

When you create **a layer**, it **should accomplish** a **specific task**, rather than installing one software package

About “layers”

When building images, **docker will “cache” layers that have not changed**; this **makes building faster**.
If a **layer** is **changed**, **all layers after** it will need to **be re-built**.
You want to structure your layers so that **the most-often change layers are at bottom**

Original

```
dockerfile
FROM condaforge/mambaforge:4.13.0-1

CMD echo "Hello world!"

RUN mamba install -y tensorflow \
    pandas \
    scikit-learn \
    matplotlib
```

Change

```
dockerfile
FROM condaforge/mambaforge:4.13.0-1

CMD echo "Run:ai is great!"

RUN mamba install -y tensorflow \
    pandas \
    scikit-learn \
    matplotlib
```

Rebuild layers

```
dockerfile
FROM condaforge/mambaforge:4.13.0-1

CMD echo "Run:ai is great!"

RUN mamba install -y tensorflow \
    pandas \
    scikit-learn \
    matplotlib
```

```
dockerfile
FROM condaforge/mambaforge:4.13.0-1

RUN mamba install -y tensorflow \
    pandas \
    scikit-learn \
    matplotlib

CMD echo "Hello world!"
```

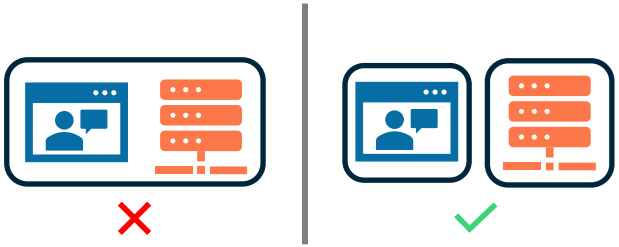
```
dockerfile
FROM condaforge/mambaforge:4.13.0-1

RUN mamba install -y tensorflow \
    pandas \
    scikit-learn \
    matplotlib

CMD echo "Run:ai is great!"
```

```
dockerfile
FROM condaforge/mambaforge:4.13.0-1

RUN mamba install -y tensorflow \
    pandas \
    scikit-learn \
    matplotlib

CMD echo "Run:ai is great!"
```

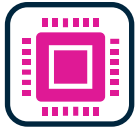
Docker images with Run:ai
(using Jupyter example)

Key concepts

1. Think of your **software** and your **storage** as **separate**

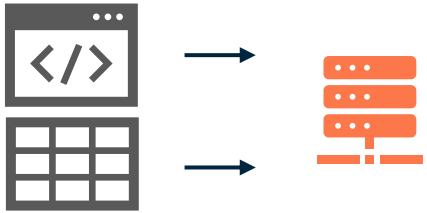


2. Run:ai will take care of the **hardware**



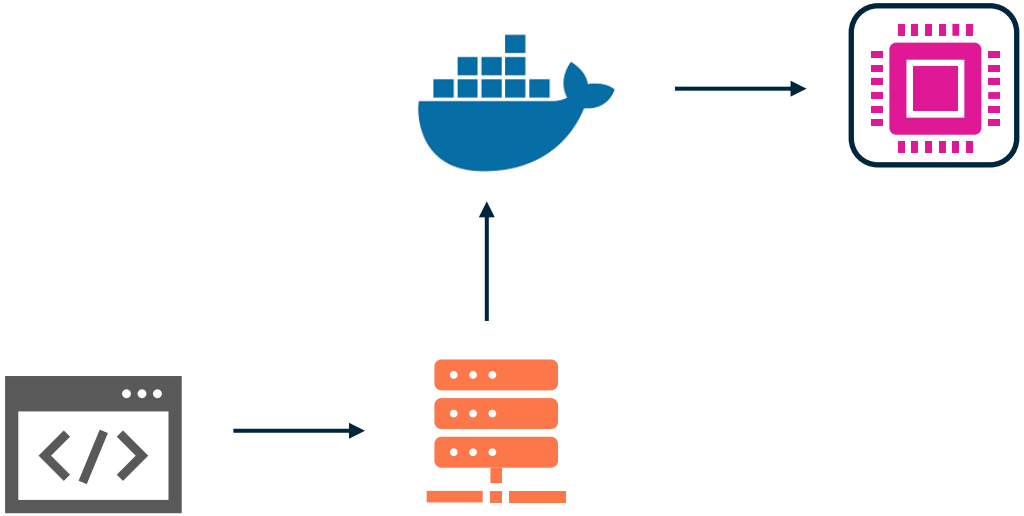
3. Your **software** will be put in **containers** (via Docker)

4. Your **code** and **data** will be kept on the **NFS**



Key concepts

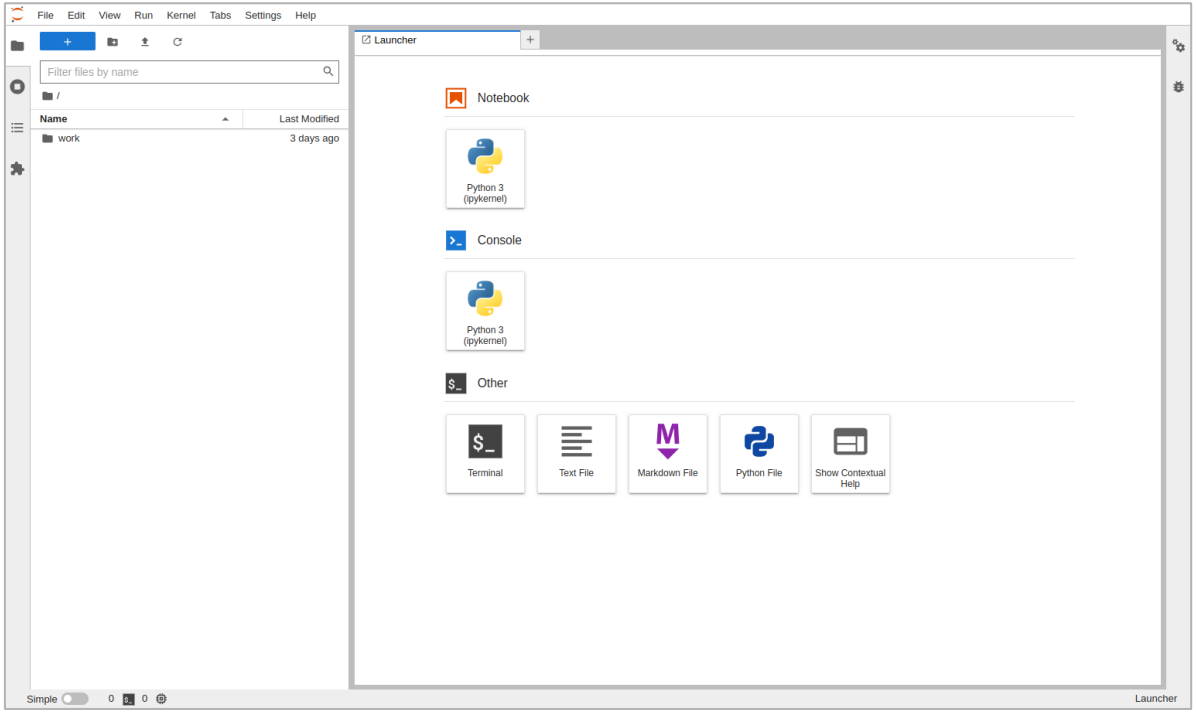
The idea is that you will ***mount* your NFS**, then run a script that is stored on the NFS



Jupyter

Jupyter Lab is a tool commonly used by data scientist.

For our example, we will use an official image which can be pulled with:
jupyter/base-notebook

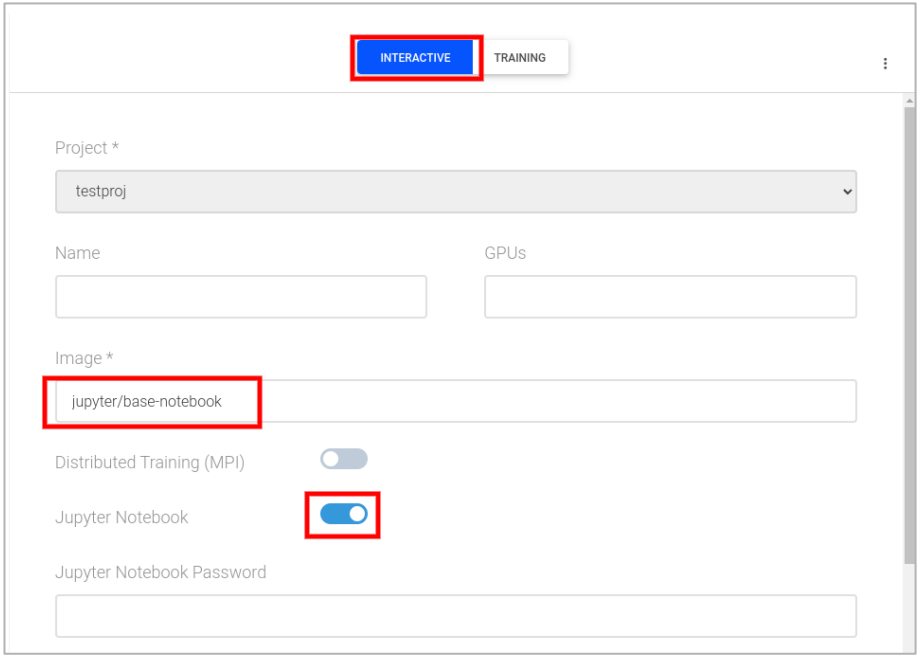


Jupyter on Run:ai

On the New Job UI:

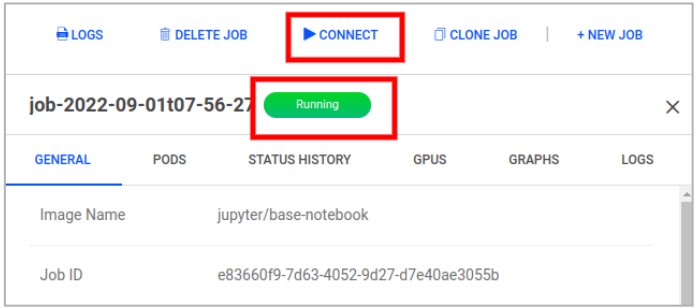
1. Select **Interactive**
2. Specify Image as **jupyter/base-notebook**
3. Toggle the **Jupyter Notebook** option

Submit the job

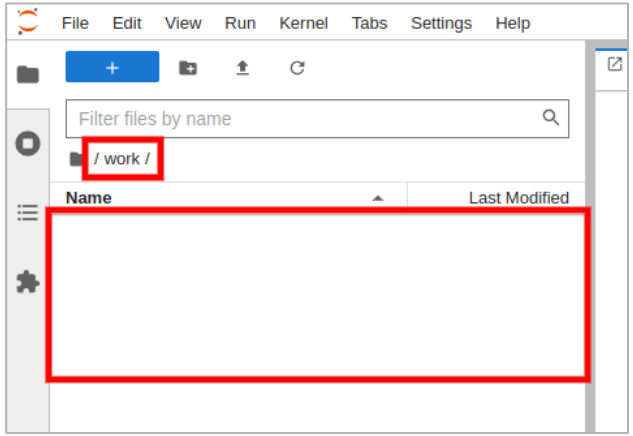


Jupyter on Run:ai

After the job **status changes to Running**, you can **connect** to the Jupyter Lab UI.



If you navigate to the **work folder**, you'll notice it's **empty**.



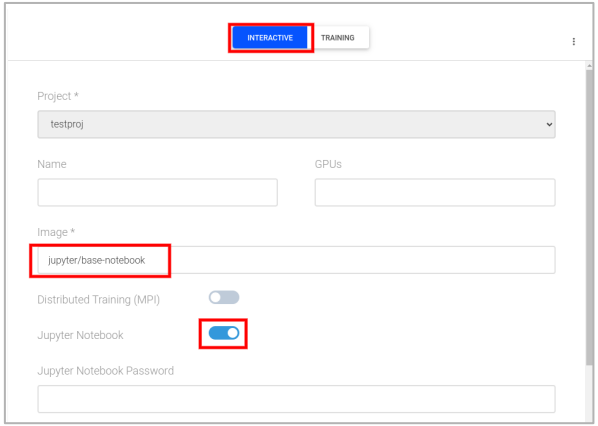
Jupyter on Run:ai **with mounted NFS**

We'll **start with the same** options as before…

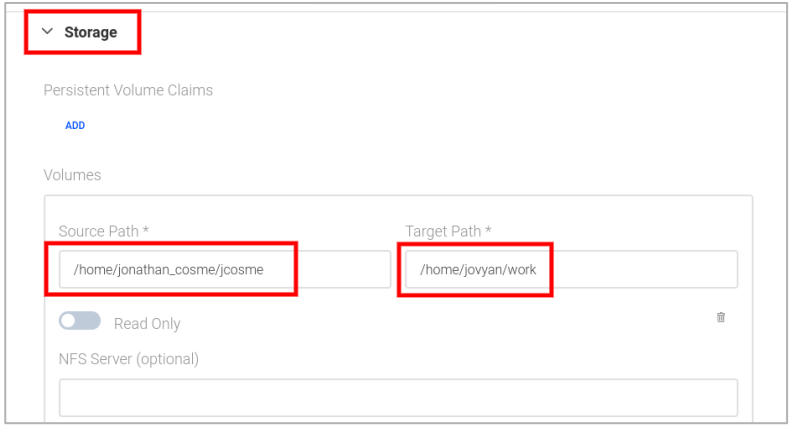
… but this time, we'll **mount** our NFS folder **to the "work" directory**.

On the New Job UI:

1. Select **Interactive**
2. Specify Image as **jupyter/base-notebook**
3. Toggle the **Jupyter Notebook** option

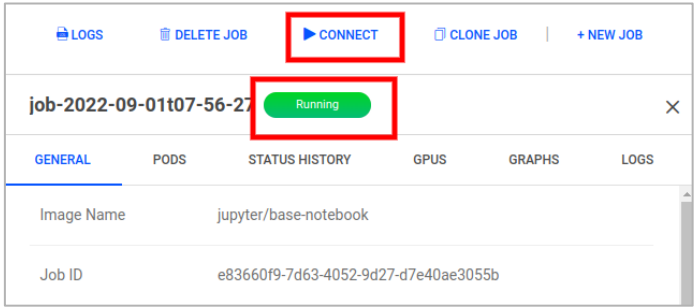


4. Under Storage, select **Add volume**
5. **Input** your **NFS directory** into the Source Path
6. **Input /home/jovyan/work** into the Target Path

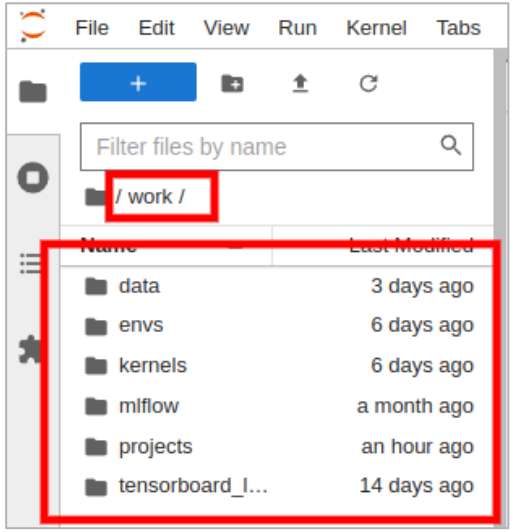


Jupyter on Run:ai with mounted NFS

After the job **status changes to Running**, you can **connect** to the Jupyter Lab UI.



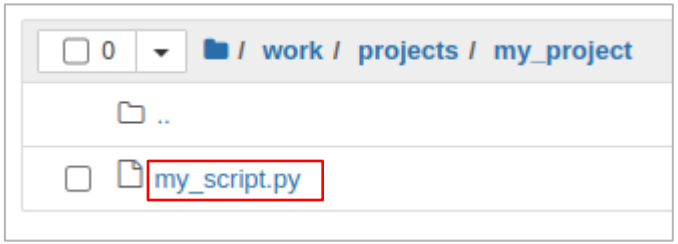
If you navigate to the **work folder**, you'll notice that all **our files and folder are there**



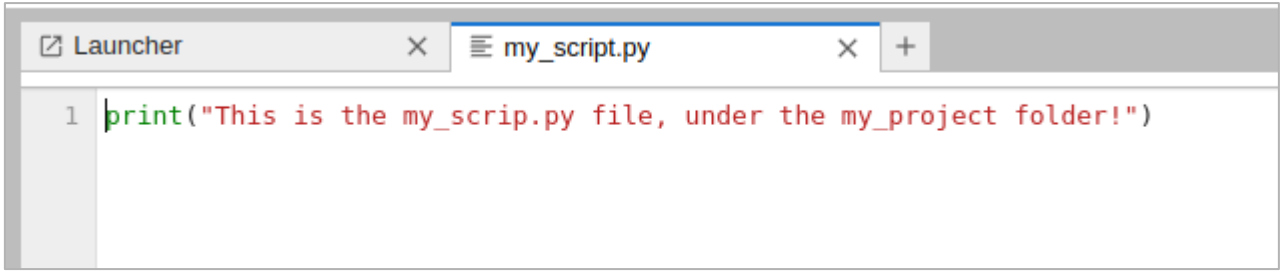
Docker with non-interactive jobs

Running a non-interactive job

Lets say that I had a script called **my_script.py**



This is the **content** of the python script:

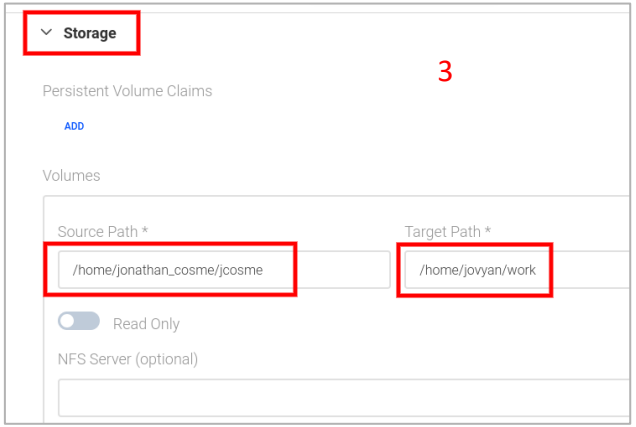


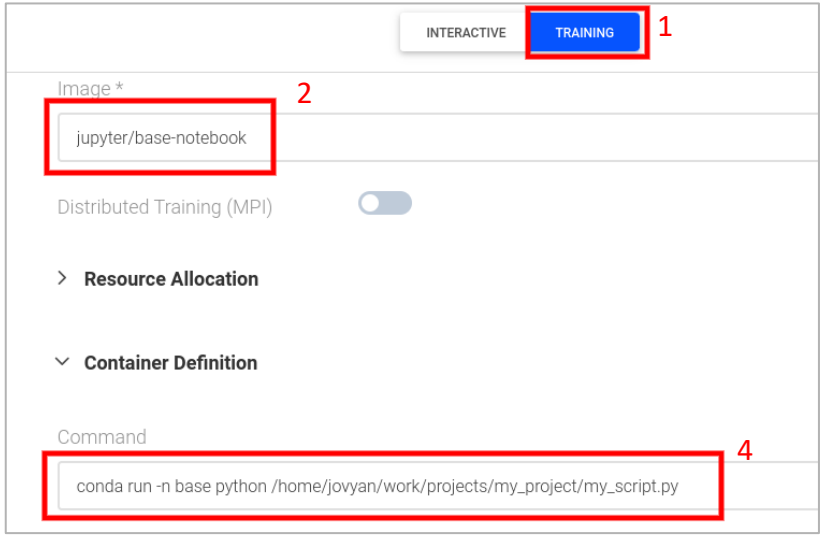
Running a non-interactive job

In order to run the script using the image:

1. Select **Training** type job
2. Specify the image as **jupyter/base-notebook**
3. **Mount** your **NFS** volume
4. **Add a Command** under Container Definition

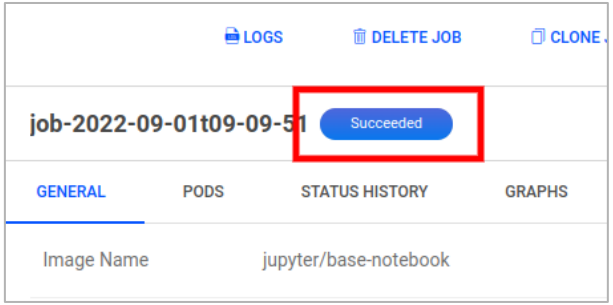
In Command, under Container Definition, input:
conda run -n base python /home/jovyan/work/projects/my_project/my_script.py



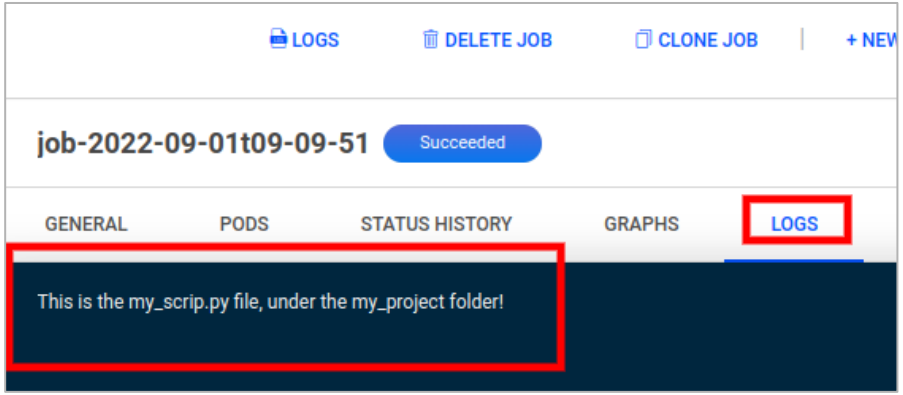


Running a non-interactive job

Wait for the job status to change to Succeeded



In the Logs tab, you should see the output of my_script.py



Closing notes

Closing notes

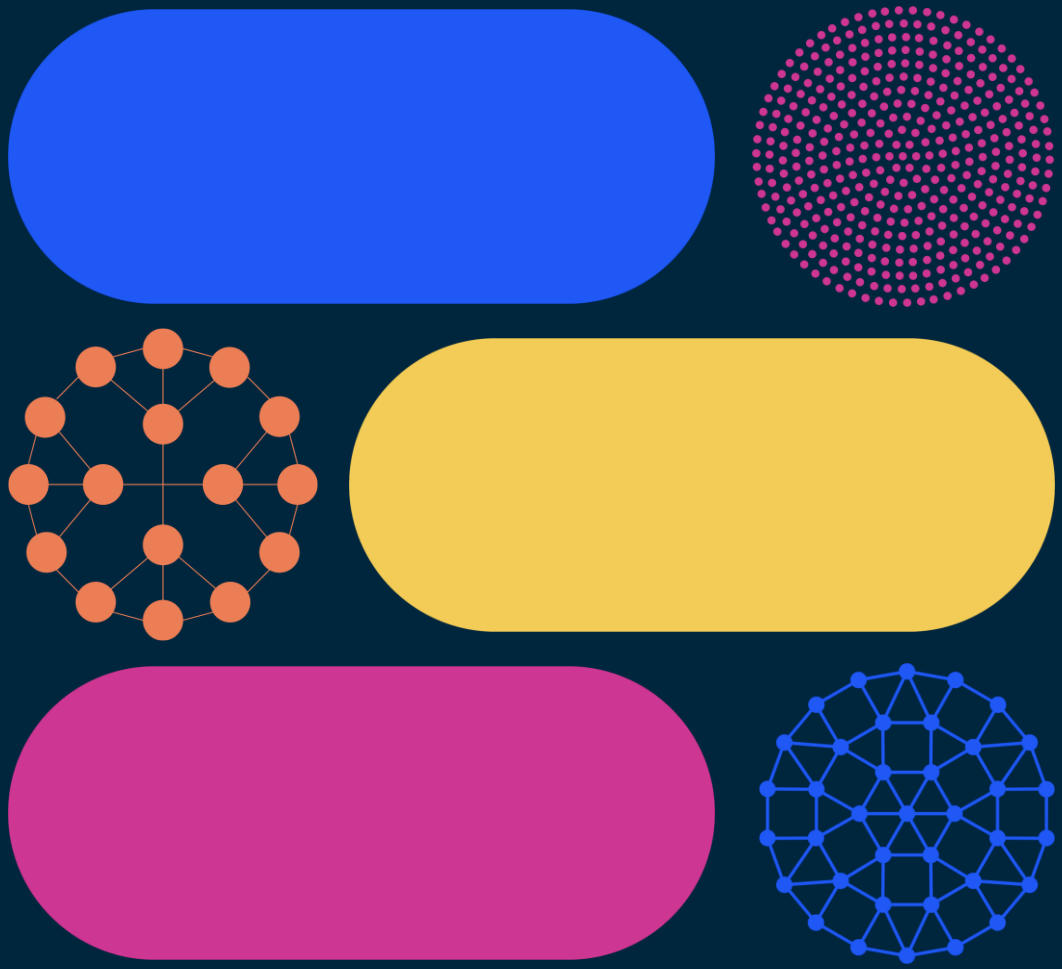
Some general Docker tips

- Use an official Docker image as a base image
  - They are more stable
- Use a specific version (the more specific the better)
  - If no version is specified Docker will just take the latest version, which could break your image
- If starting with a raw OS image, selected an image with a lean distribution.
  - For example, use Alpine over Ubuntu. Ubuntu contains a lot of software you won't need.

BE AWARE

- ONLY THE CONTENT OF THE FOLDER CONTAINING YOUR NFS MOUNT WILL BE SAVED/PERSIST
  - This means that ALL OTHER FILES that were created during the job session will be gone forever, once the job finishes
- ANY SOFTWARE CHANGES (using apt/apt-get install or conda/mamba install/create) made while the job is running WILL NOT BE SAVED
  - If you want to make permanent software changes to the image, you will need to update the dockerfile, re-build it, and re-push it to Docker hub.

Thank you!



info@run.ai | www.run.ai