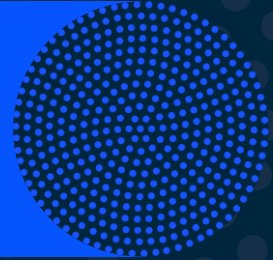


Docker  
with  
run:ai  
(and Jupyter)

# Docker intro

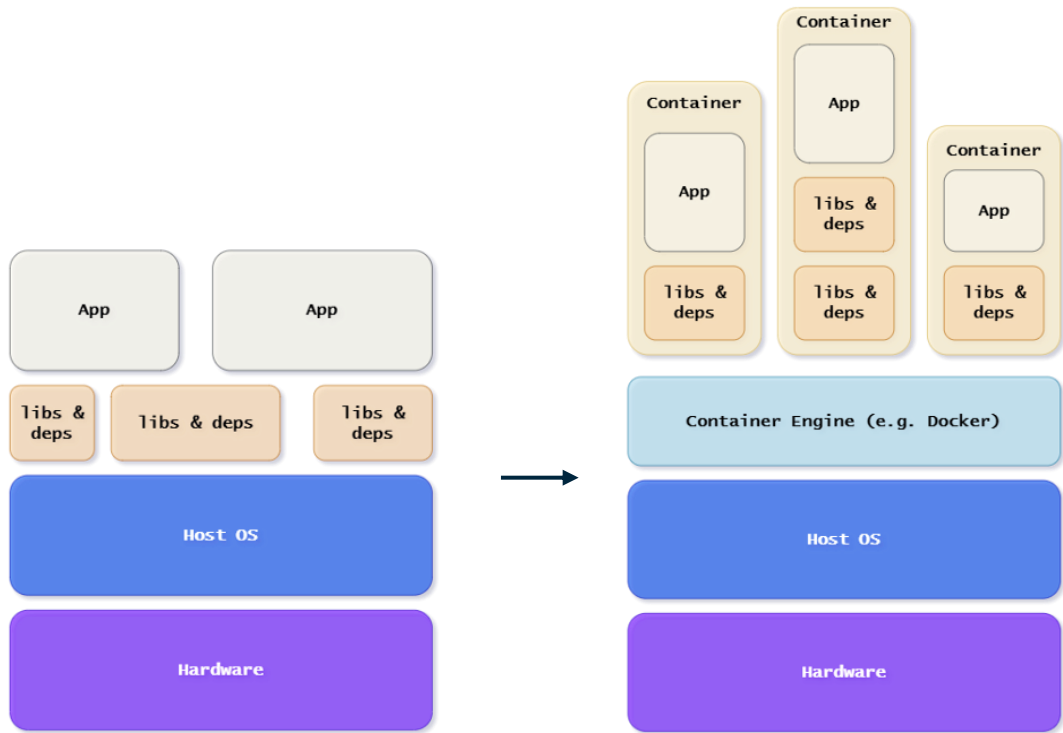


# What is Docker?

Docker is a way to create “images.”

Images 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



Image 1

## User 2 Code

Requires

- Cudatoolkit=11.7
- Python=3.9



Image 2

## User 3 Code

Requires

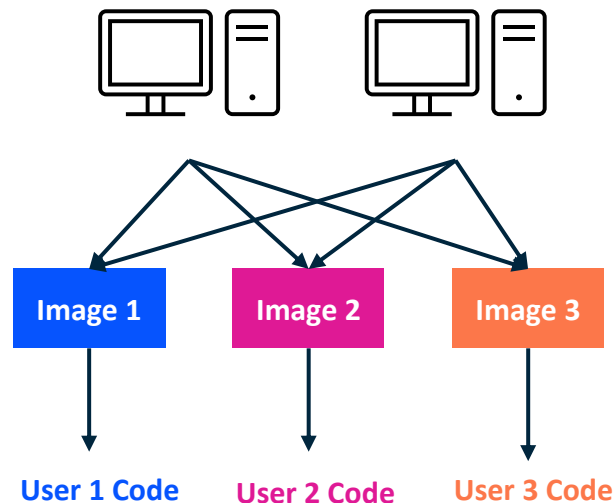
- Cudatoolkit=11.5
- Python=3.8



Image 3

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?

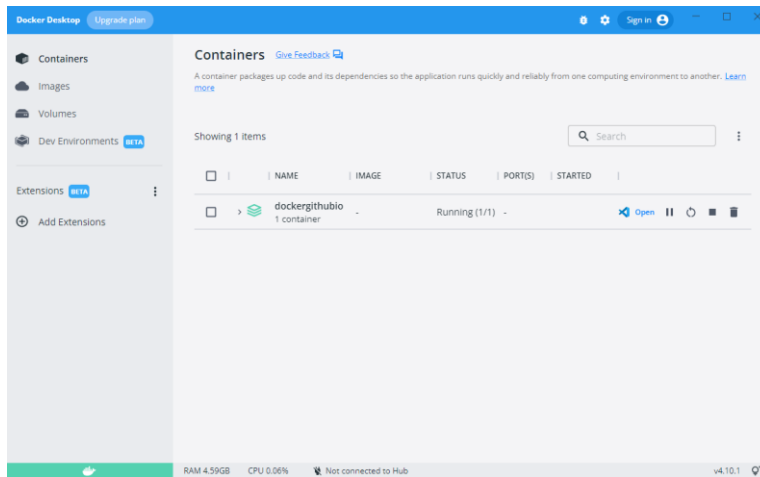
An image repository

I use Docker hub.



Docker software

I use Docker Desktop.



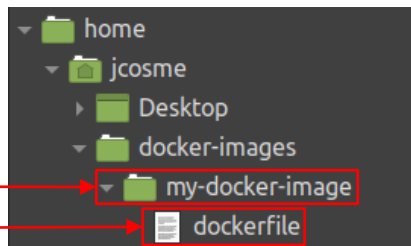
<https://docs.docker.com/desktop/install/linux-install/>

# 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} .`

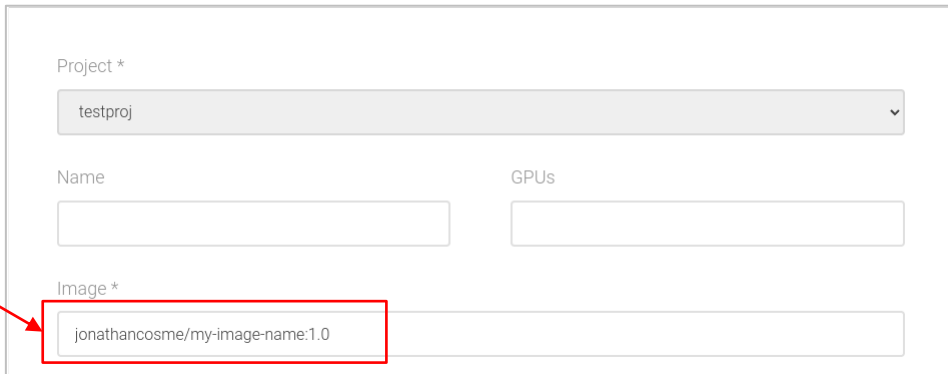
(this is what my command be) → `docker build -t jonathancosme/my-docker-image:1.0 .`

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



The screenshot shows a web form for submitting a job. It includes a 'Project \*' dropdown menu with 'testproj' selected. Below this are 'Name' and 'GPUs' input fields. The 'Image \*' field is highlighted with a red rectangle, and a red arrow points from the text 'Enter the image name in the Image field of the UI' to this field. The text 'jonathancosme/my-image-name:1.0' is entered in the 'Image \*' field.

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
1 FROM condaforge/mambaforge:4.13.0-1
2
3 RUN apt-get install --yes --no-install-recommends openssh-server
4
5 RUN mamba install --yes --name base --channel conda-forge tensorflow
6
7 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/>

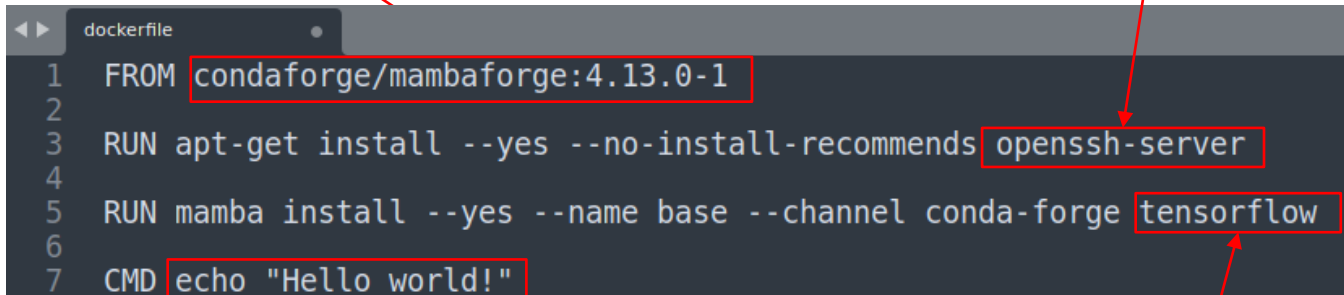


# What is a “dockerfile?”

In this particular image,

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

2. Install openssh-server



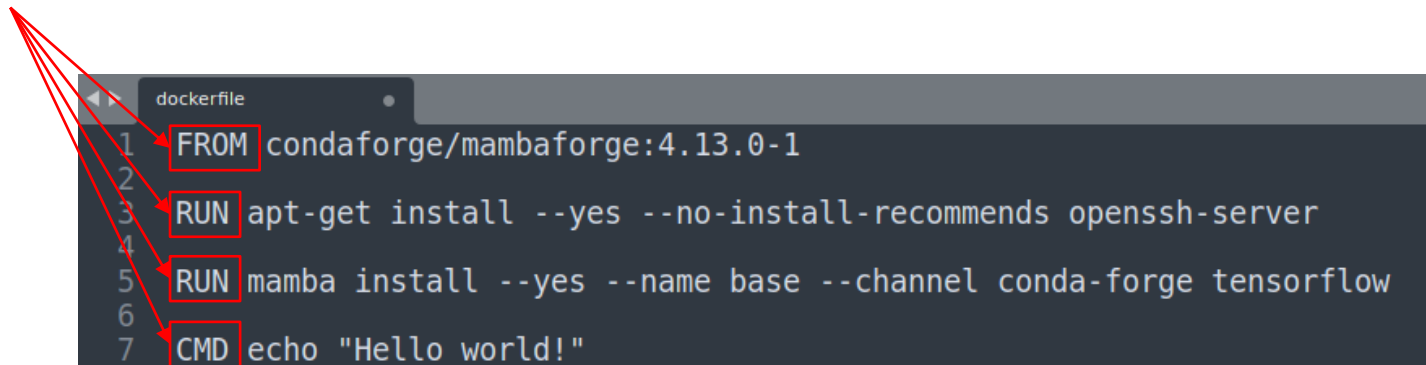
```
1 FROM condaforge/mambaforge:4.13.0-1
2
3 RUN apt-get install --yes --no-install-recommends openssh-server
4
5 RUN mamba install --yes --name base --channel conda-forge tensorflow
6
7 CMD echo "Hello world!"
```

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
1 FROM condaforge/mambaforge:4.13.0-1
2
3 RUN apt-get install --yes --no-install-recommends openssh-server
4
5 RUN mamba install --yes --name base --channel conda-forge tensorflow
6
7 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
1 FROM condaforge/mambaforge:4.13.0-1
2
3 RUN mamba install -y tensorflow
4
5 RUN mamba install -y pandas
6
7 RUN mamba install -y scikit-learn
8
9 RUN mamba install -y matplotlib
10
11 CMD echo "Hello world!"
```



```
dockerfile
1 FROM condaforge/mambaforge:4.13.0-1
2
3 RUN mamba install -y tensorflow \
4     pandas \
5     scikit-learn \
6     matplotlib
7
8 CMD echo "Hello world!"
```

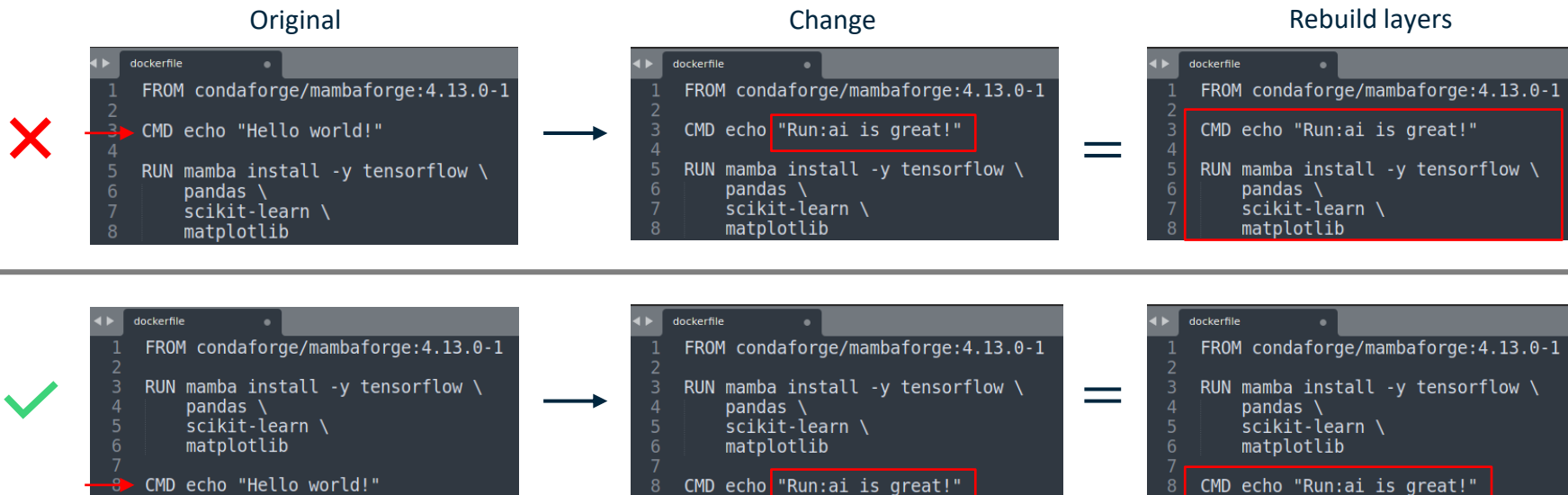
When you create a **layer**, it **should accomplish a specific task**, rather than installing one software package

# About “layers”

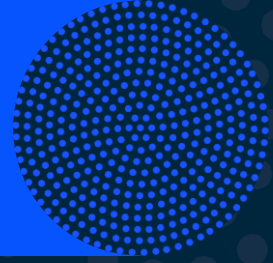
When building images, **docker will “cache” layers that have not changed**; this **makes building faster**.

**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**



Docker images with **Run:ai**  
(using Jupyter example)

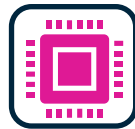


# Key concepts

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



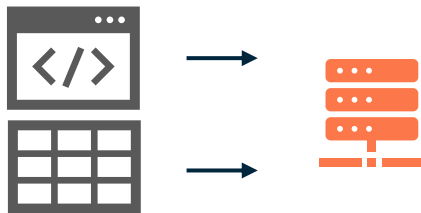
2. Run:ai will take care of the **hardware**



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

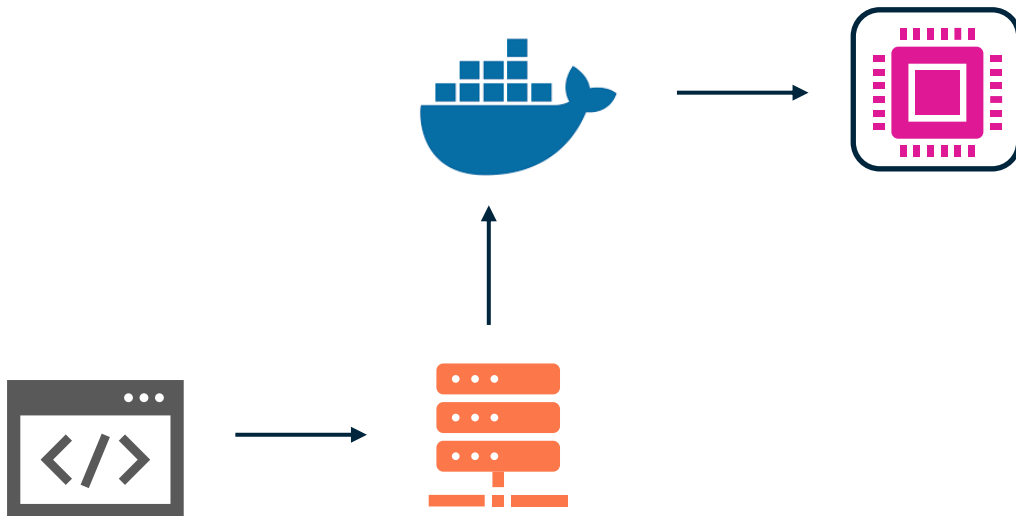


4. Your **code** and **data** will be kept on the **NFS**



# Key concepts

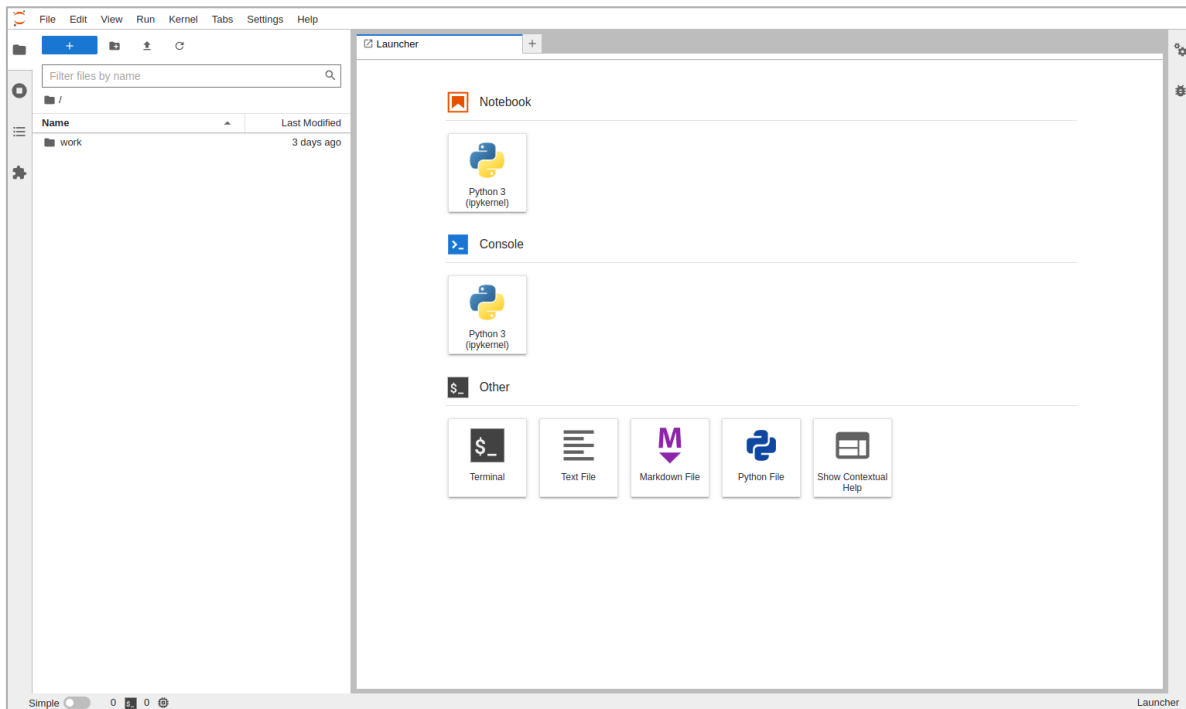
The idea is that you will **mount your NFS**, then run a script that is stored on the NFS



# 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

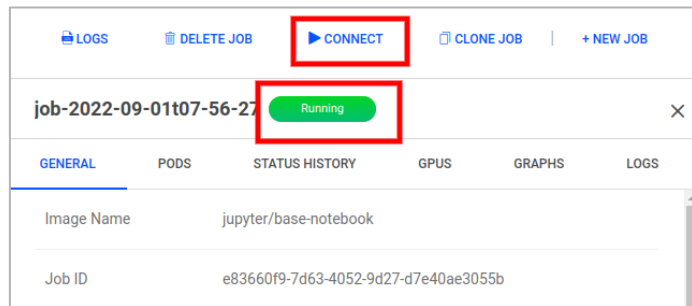
On the New Job UI:

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

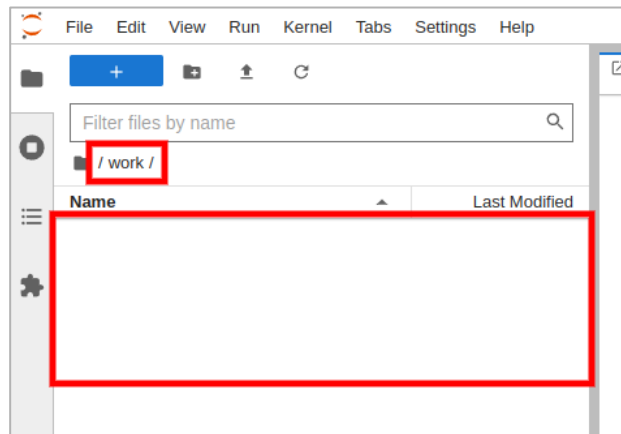
The screenshot displays the 'New Job' interface on the Run:ai platform. At the top, there are two tabs: 'INTERACTIVE' (highlighted with a red box) and 'TRAINING'. Below the tabs, the 'Project' dropdown is set to 'testproj'. The 'Name' and 'GPUs' fields are empty. The 'Image' dropdown is set to 'jupyter/base-notebook' (highlighted with a red box). The 'Distributed Training (MPI)' toggle is turned off. The 'Jupyter Notebook' toggle is turned on (highlighted with a red box). The 'Jupyter Notebook Password' field is empty.

# 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**.

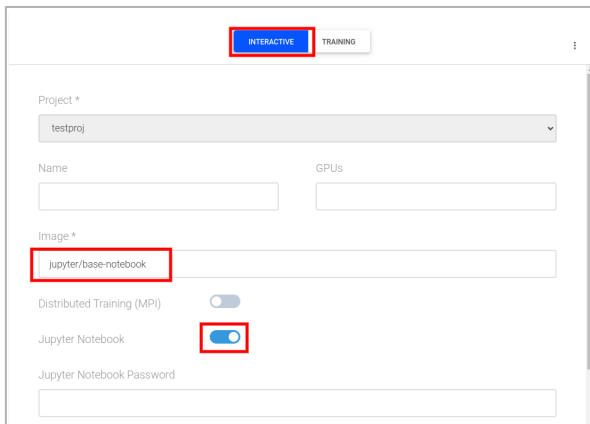


# Jupyter on Run:ai with mounted NFS

We'll start with the same options as before...

On the New Job UI:

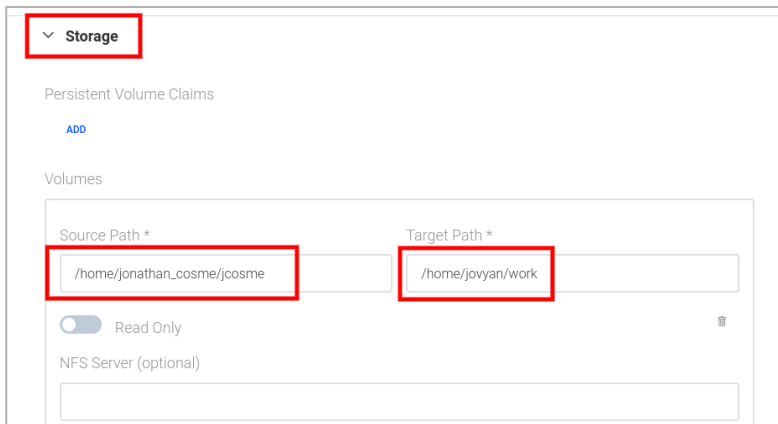
1. Select **Interactive**
2. Specify Image as **jupyter/base-notebook**
3. Toggle the **Jupyter Notebook** option



The screenshot shows the 'New Job' UI with the 'INTERACTIVE' tab selected. The 'Project' dropdown is set to 'testproj'. The 'Image' field contains 'jupyter/base-notebook'. The 'Jupyter Notebook' toggle is turned on. The 'Distributed Training (MPI)' toggle is turned off. The 'Jupyter Notebook Password' field is empty.

... but this time, we'll mount our NFS folder to the “work” directory.

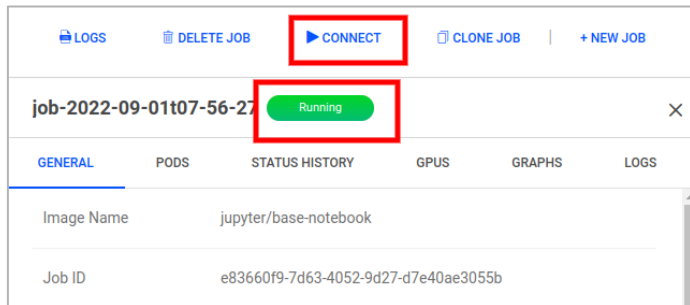
4. Under Storage, select **Add volume**
5. Input your **NFS directory** into the Source Path
6. Input **/home/jovyan/work** into the Target Path



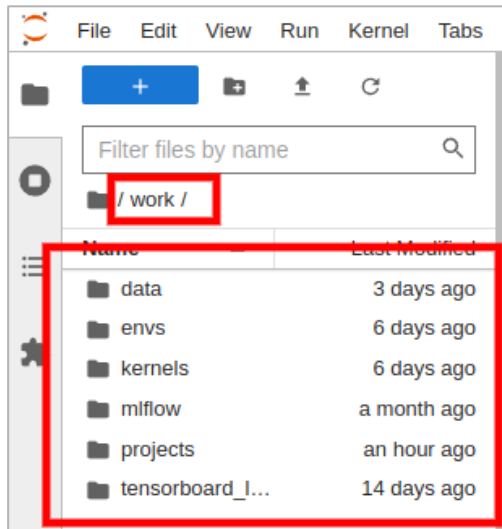
The screenshot shows the 'Storage' configuration section. The 'Storage' dropdown is expanded. The 'ADD' button is visible. The 'Volumes' section contains two input fields: 'Source Path' with the value '/home/jonathan\_cosme/jcosme' and 'Target Path' with the value '/home/jovyan/work'. The 'Read Only' toggle is turned off. The 'NFS Server (optional)' field is empty.

# Jupyter on Run:ai with mounted NFS

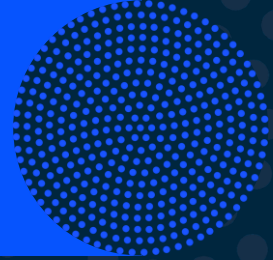
After the job status changes to **Running**, you can **connect** to the Jupyter Lab UI.



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

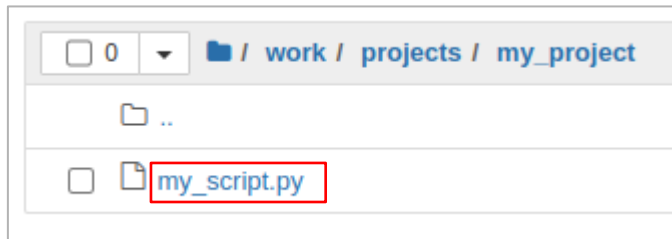


## Docker with non-interactive jobs

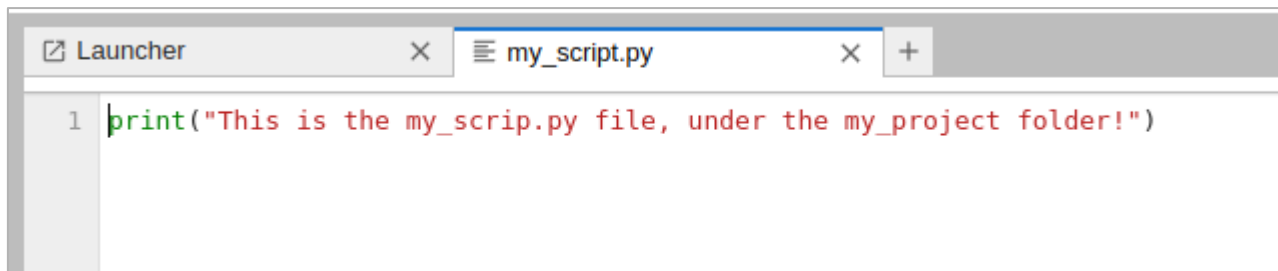


# Running a non-interactive job

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

The screenshot shows the 'Storage' configuration section. A red box labeled '3' highlights the 'Storage' header. Below it, the 'Persistent Volume Claims' section has an 'ADD' button. The 'Volumes' section contains two input fields: 'Source Path \*' with the value '/home/jonathan\_cosme/jcosme' and 'Target Path \*' with the value '/home/jovyan/work'. Both fields are highlighted with red boxes. Below these fields are a 'Read Only' toggle switch and an 'NFS Server (optional)' text input field.

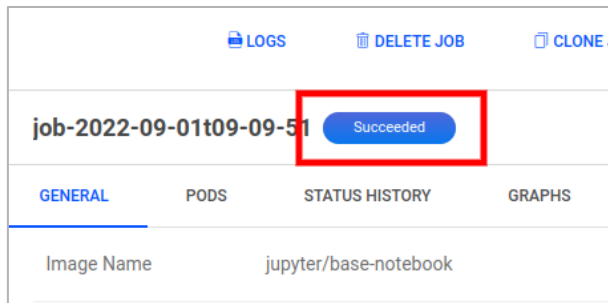
In Command, under Container Definition, input:

**conda run -n base python /home/jovyan/work/projects/my\_project/my\_script.py**

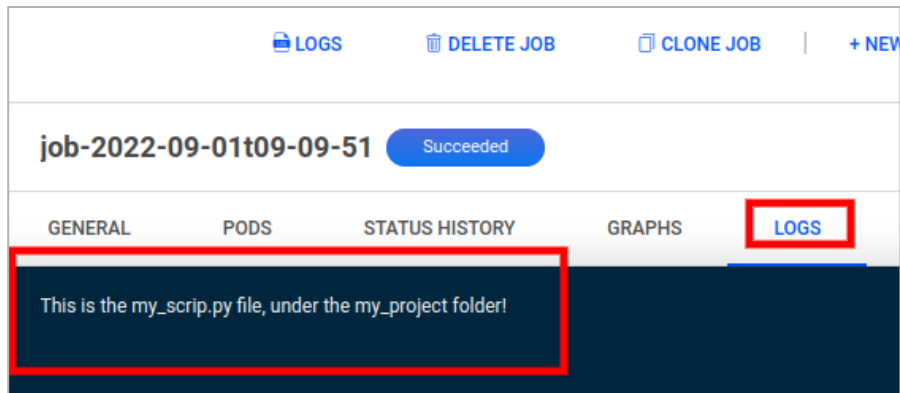
The screenshot shows the 'Container Definition' section. At the top, there are two tabs: 'INTERACTIVE' and 'TRAINING'. The 'TRAINING' tab is selected and highlighted with a red box labeled '1'. Below the tabs, the 'Image \*' field contains the value 'jupyter/base-notebook' and is highlighted with a red box labeled '2'. The 'Distributed Training (MPI)' toggle switch is turned off. The 'Resource Allocation' section is expanded. The 'Container Definition' section is expanded, and the 'Command' field contains the command 'conda run -n base python /home/jovyan/work/projects/my\_project/my\_script.py', which is highlighted with a red box labeled '4'.

# Running a non-interactive job

Wait for the job status to change to Succeeded

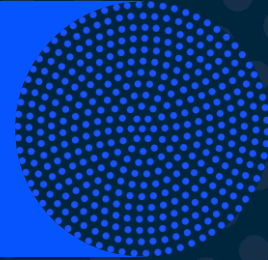


In the Logs tab, you should see the output of my\_script.py





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!

