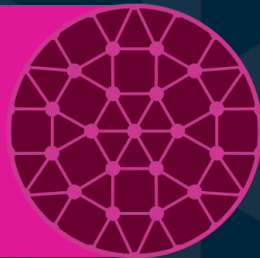


mlflow
with
run:ai

mlflow overview



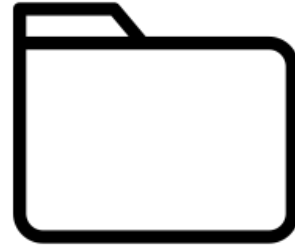
What is needed for mlflow to run?

- 1) A **Database** to store information related to experiment runs
- 2) An **Artifacts folder** to store objects related to the runs

Database



Artifacts folder



How does mlflow work?

1) First we start the server with a CLI command

running either of these commands will automatically create a database in the local directory, if one doesn't exist.

```
mlflow ui
```

(or)

```
mlflow server
```

we can also choose to specify the location of the database, and artifact folder, as well as the host IP, and port.

```
mlflow server \  
  --backend-store-uri=sqlite:///abs/path/to/db/mlflow.db \  
  --default-artifact-root=/abs/path/to/artifacts \  
  --host=0.0.0.0 \  
  --port=5000
```

How does mlflow work?

2) call mlflow commands within the python script

you'll want to import mlflow, then set the tracking uri so that mlflow will save everything to the database and artifact folder.

Then you'll want to start your run, and at the end, you'll want to end the run.

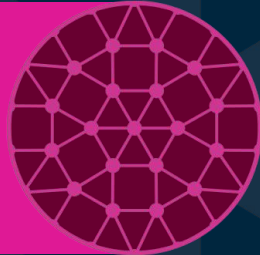
```
import mlflow

mlflow.set_tracking_uri('0.0.0.0:5000')
mlflow.start_run()

"""
your code here
"""

mlflow.end_run()
```

mlflow with run:ai



What is needed for mlflow to run on **run:ai**?

1) A **persistent** directory to keep

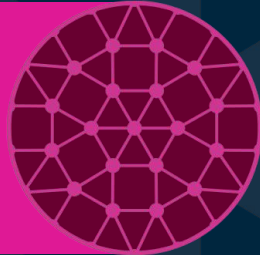
- mlflow database
- mlflow artifacts folder

2) A **docker image** with the following installed

- mlflow
- jupyterlab*
- jupyter-server-proxy*

*needed in order to access the mlflow UI

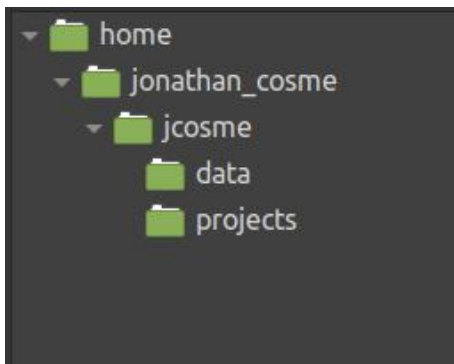
creating persistent directory



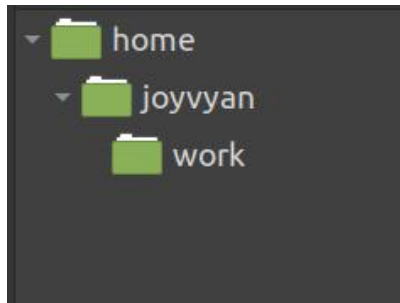
Before we start

We need to create an 'mlflow' folder on our NFS.

This is what our NFS folder structure looks like now



This is the default folder structure for our jupyter lab image

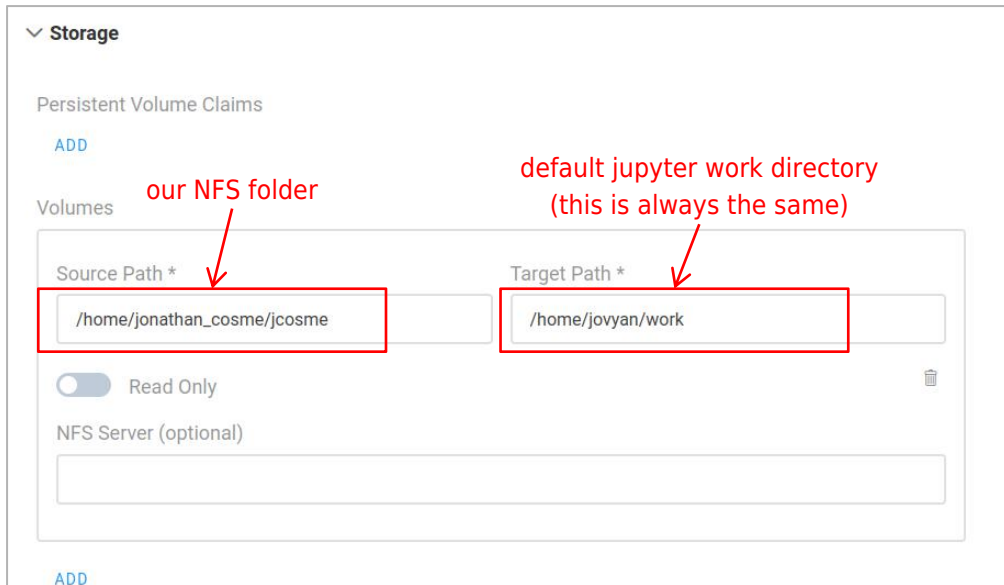


Before we start

We need to create an 'mlflow' folder on our NFS.

For *our example*:

Whenever we create a job on run:ai, we *must always* mount our NFS to the default jupyter work directory



The screenshot shows the 'Storage' configuration section in the run:ai interface. Under 'Persistent Volume Claims', there is an 'ADD' button. Below this, the 'Volumes' section contains a configuration card. The card has two input fields: 'Source Path *' and 'Target Path *'. The 'Source Path *' field contains the text '/home/jonathan_cosme/jcosme' and is highlighted with a red box. A red arrow points from the text 'our NFS folder' to this box. The 'Target Path *' field contains the text '/home/jovyan/work' and is also highlighted with a red box. A red arrow points from the text 'default jupyter work directory (this is always the same)' to this box. Below the input fields, there is a 'Read Only' toggle switch which is currently turned off. At the bottom of the configuration card is an 'NFS Server (optional)' text input field. Below the entire configuration card is another 'ADD' button.

Storage

Persistent Volume Claims

ADD

Volumes

Source Path *

/home/jonathan_cosme/jcosme

our NFS folder

Target Path *

/home/jovyan/work

default jupyter work directory
(this is always the same)

Read Only

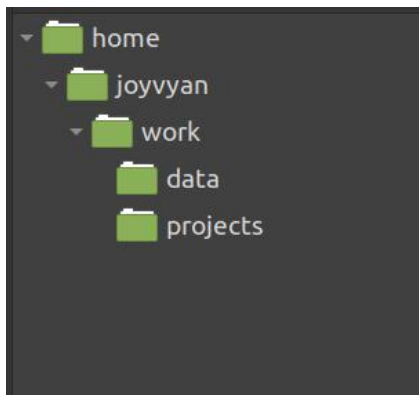
NFS Server (optional)

ADD

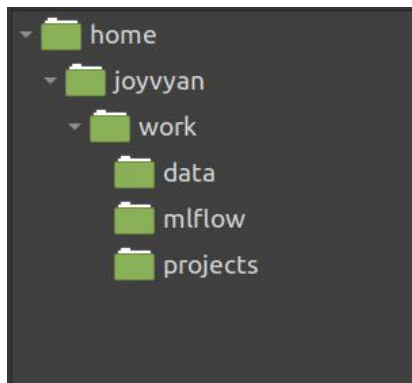
Before we start

We need to create an 'mlflow' folder on our NFS.

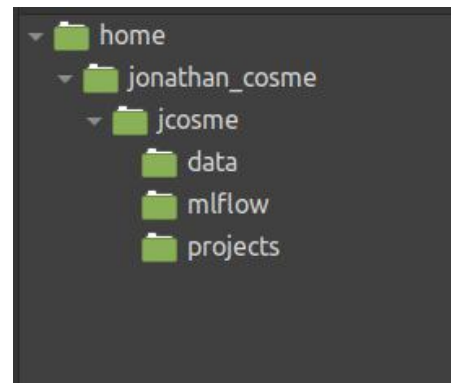
1. After we mount our NFS volume, our work directory will look like this:



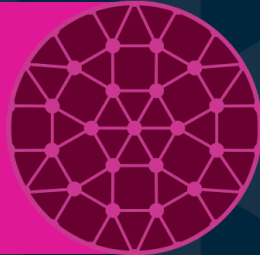
2. Using Jupyter Lab, we create an 'mlflow' folder within the work directory:



3. This will cause our NFS directory to automatically look like this:



docker image



Docker used in our example

The docker image we will use is:

jonathancosme/mlflow-ui

This is what is in the dockerfile:

```
FROM jonathancosme/root-jpy-prox
```

← notebook with jupyter-server-proxy already installed

```
RUN mamba install -c conda-forge mlflow -y && \
    mamba clean --all -f -y
```

← install mlflow

```
RUN sudo apt-get update && \
    sudo apt-get install -y git
```

← install git (needed by mlflow)

```
COPY jupyter_server_config.py /etc/jupyter/
```

← copy new config file, with mlflow UI access configured

Docker used in our example

The docker image we will use is:

jonathancosme/mlflow-ui

in order to access the mlflow UI, we need to **add this entry to the jupyter_server_config.py** file, and replace the existing file in the image

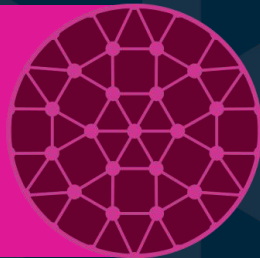
```
c.ServerProxy.servers = {
  'mlflow-server': {
    'command': [
      'mlflow', 'server',
      '--backend-store-uri=sqlite:///home/jovyan/work/mlflow/mlflow.db',
      '--default-artifact-root=/home/jovyan/work/mlflow/artifacts',
      '--host=0.0.0.0',
      '--port=5000',
      '--serve-artifacts',
    ],
    'timeout': 30,
    'launcher_entry': {
      'title': 'mlflow'
    }
  },
  'port': 5000
}
```

we specify our database and artifact folder locations (this is why we must always mount our NFS directory to the default jupyter work directory)

we must make sure to start the server on this IP and port

This tells jupyter to forward port 5000 to the URL

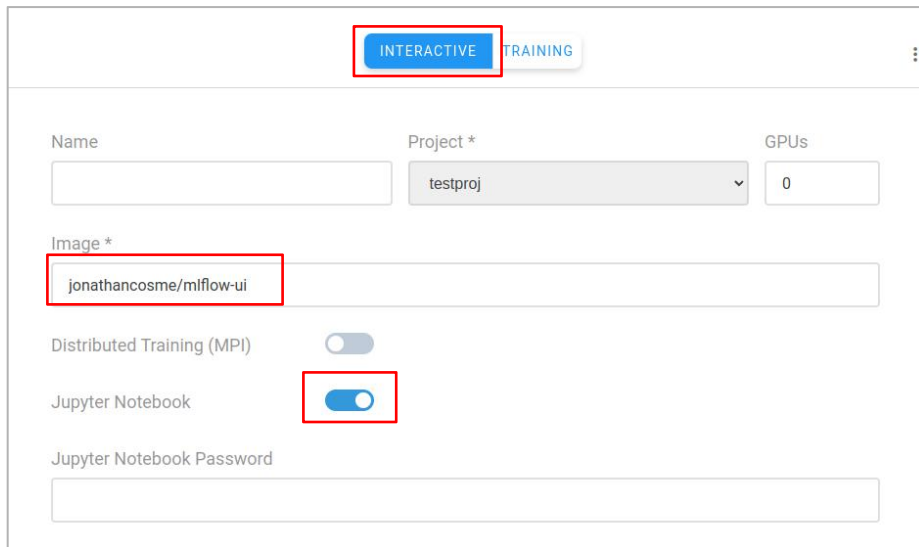
Accessing the mlflow UI



Access mlflow UI

Create a jupyter interactive job with:

- image jonathancosme/mlflow-ui
- mounted NFS folder (with 'mlflow' folder) in default jupyter work directory



The screenshot shows the 'INTERACTIVE' tab selected in the RunAI job configuration. The 'Image' field is set to 'jonathancosme/mlflow-ui'. The 'Jupyter Notebook' toggle is turned on. The 'Project' is set to 'testproj' and 'GPUs' is set to 0.

INTERACTIVE TRAINING

Name Project * GPUs

testproj 0

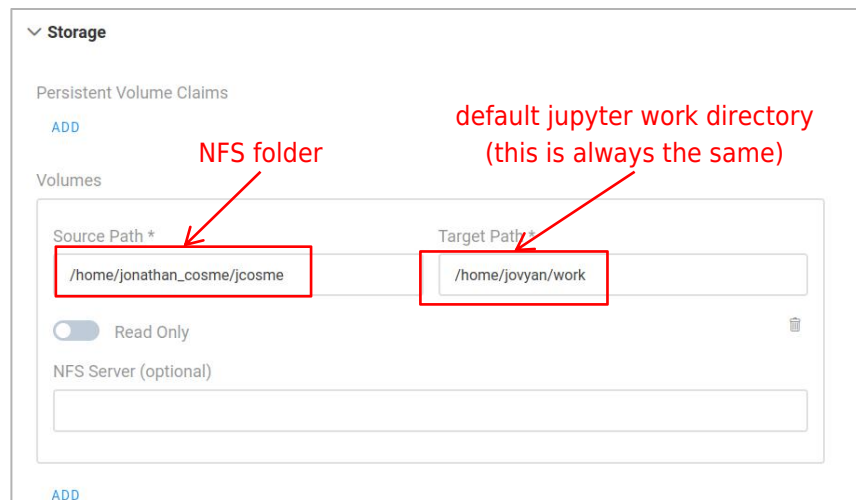
Image *

jonathancosme/mlflow-ui

Distributed Training (MPI) ☐

Jupyter Notebook ☒

Jupyter Notebook Password



The screenshot shows the 'Storage' section with 'Volumes' configured. The 'Source Path' is '/home/jonathan_cosme/jcosme' and the 'Target Path' is '/home/jovyan/work'. Red arrows point to these paths with labels: 'NFS folder' for the source path and 'default jupyter work directory (this is always the same)' for the target path.

Storage

Persistent Volume Claims

ADD

Volumes

Source Path * Target Path

/home/jonathan_cosme/jcosme /home/jovyan/work

☒ Read Only

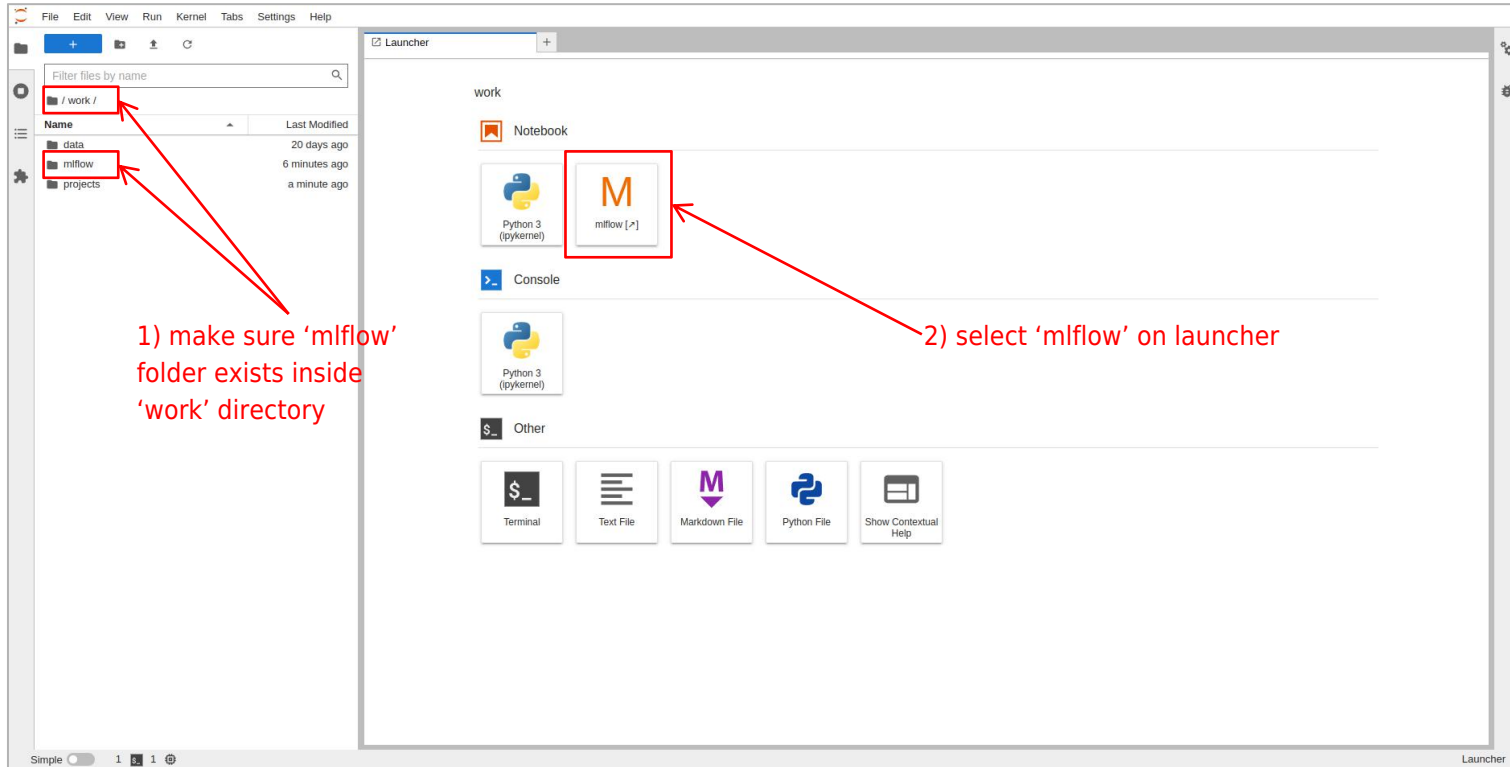
NFS Server (optional)

ADD

NFS folder

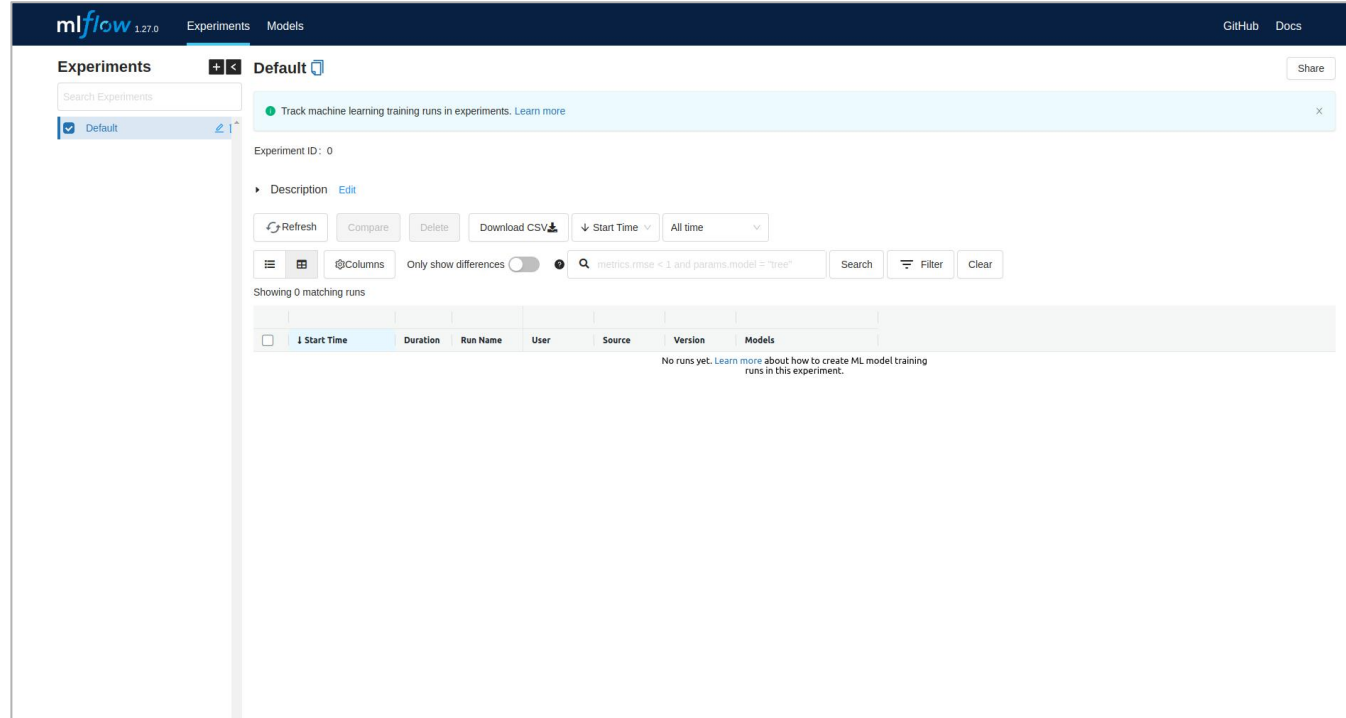
default jupyter work directory (this is always the same)

Access mlflow UI

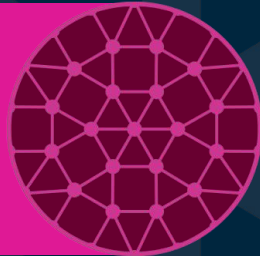


Access mlflow UI

A new tab should appear with the mlflow UI



running mlflow experiments with run:ai



Python scripts

in your script, you should first launch the mlflow server

```
import mlflow
import os
import time
from subprocess import Popen

# first we start the mlflow server
mlflow_command = 'mlflow server --backend-store-uri=sqlite:///home/jovyan/work/mlflow/mlflow.db --default-artifact-root=/home/jovyan/work/mlflow/artifacts --host=0.0.0.0 --port=5000 --serve-artifacts'
print(f'running command: {mlflow_command}')
proc = Popen([mlflow_command], shell=True,
             stdin=None, stdout=None, stderr=None, close_fds=True)
print(f'waiting 5 seconds...')
time.sleep(5) # we wait 5 seconds to give the server time to start

# create the tracking uri string
uri = 'http://0.0.0.0:5000' # in our example, this is always the same
project_name = 'test_project'
run_name = 'run_1'

mlflow.set_tracking_uri(uri)
mlflow.set_experiment(experiment_name=project_name)
mlflow.start_run(run_name=run_name)

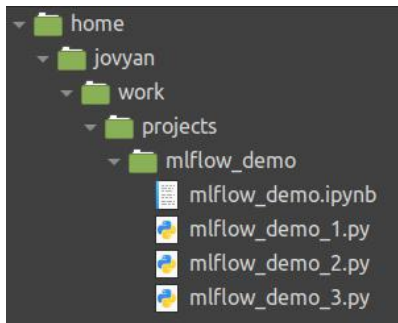
"""
your code here
"""

mlflow.end_run()
```

Then be sure to set the tracking uri

CLI submission

Our example scripts are located here:



so our CLI command would look like this:

```
runai submit \
  --project testproj \
  --gpu 0 \
  --job-name-prefix mlflow-demo \
  --image jonathancosme/mlflow-ui \
  --volume /home/jonathan_cosme/jcosme:/home/jovyan/work \
  -- python work/projects/mlflow_demo/mlflow_demo_1.py
```

make sure you use the mlflow-ui
docker image

make sure you mount the NFS to the
work directory

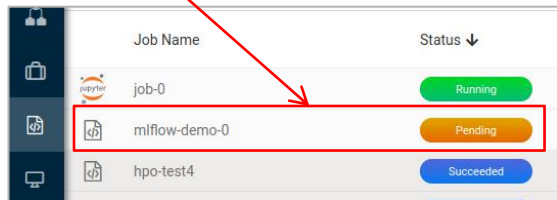
The code to run the job must specify the
python script with relative location

Example job submission

1) submit CLI command

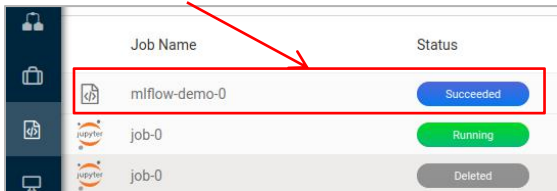
```
jcosme@Jane:~$ runai submit \  
> --project testproj \  
> --gpu 0 \  
> --job-name-prefix mlflow-demo \  
> --image jonathancosme/mlflow-ui \  
> --volume /home/jonathan cosme/jcosme:/home/jovyan/work \  
> -- python work/projects/mlflow demo/mlflow demo 1.py
```

2) a new job should appear



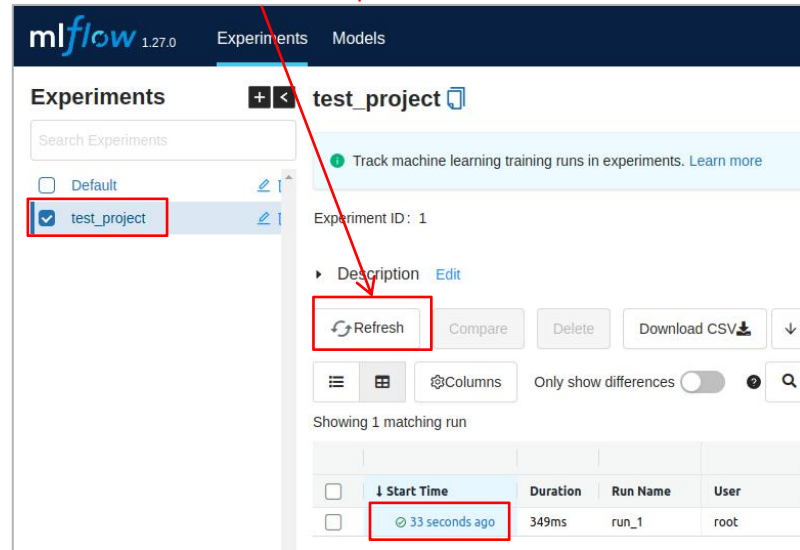
Job Name	Status
job-0	Running
mlflow-demo-0	Pending
hpo-test4	Succeeded

3) wait for job to finish



Job Name	Status
mlflow-demo-0	Succeeded
job-0	Running
job-0	Deleted

4) refresh mlflow UI to see updates



mlflow 1.27.0 Experiments Models

Experiments

Search Experiments

☐ Default ☒ test_project

Experiment ID: 1

Description Edit

Refresh Compare Delete Download CSV

Columns Only show differences

Showing 1 matching run

	Start Time	Duration	Run Name	User
<input type="checkbox"/>	33 seconds ago	349ms	run_1	root

Thank you!

