Table of Contents

[1. Introduction](#__RefHeading___Toc332212250) 2

[2.](#__RefHeading___Toc332212251) About MLFlow

[2.1.](#__RefHeading___Toc332212252) Quick start with MLFlow..................................................................................................

[2.2. Inter Function/Component Movement Diagram 5](#__RefHeading___Toc332212253)

[2.4. MLFlow Locally View 5](#__RefHeading___Toc332212256)

[3. MLFlow Cloud services 6](#__RefHeading___Toc332212261)

[3.1. EC2 Instance 6](#__RefHeading___Toc332212262)

[3.2. S3 bucket for Storage Aritifacts 6](#__RefHeading___Toc332212263)

[3.3. System Performance, Resilience, and Scalability 6](#__RefHeading___Toc332212264)

[4. Data 6](#__RefHeading___Toc332212265)

ML FLOW

Documentation

# What is MLflow

*“MLflow is an open-source platform to manage the****ML lifecycle,****including****experimentation****,****reproducibility****,****deployment****, and a****central model registry.****” — mlflow.org*

Put it simply, MLflow is a package that you can install in your python environment to:

* Perform experiment tracking
* Package data science code in a reproducible way
* Deploy models
* Manage models from development to production

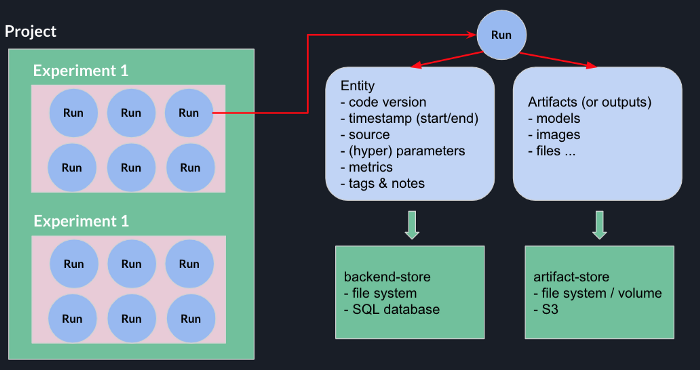
MLflow integrates with any machine learning or deep learning framework such as Scikit-learn, TensorFlow, PyTorch, h2o.ai, XGBoost, etc.

Besides, it’s also cloud-agnostic. You can run it ****everywhere****: AWS, Google Cloud Platform or Azure Machine Learning. We’ll see in this post how to set it up on AWS. Wait till the end to see how it’s done

# Quickstart using MLFlow tracking

MLflow tracking is a component that’ll help you log your machine learning experiments very easily.

It’s organized into ****experiments**** and each experiment is split into ****runs.****

********

An example of an experiment could be “Training a binary classifier”. In that case, each run corresponds to a single model fit.

While fitting different models on different runs, you can use MLflow to track ****a lot**** of data:

* ****parameters:****what you use to tune your models (e.g. n\_estimators, max\_depth, epochs, kernel\_size, dropout, batch\_size, etc.)
* the****metrics****of your models (loss, AUC, MAE, MSE; F1 score, accuracy, R-squared (r2))
* ****data****: the different versions of the data your model used in each run.
* ****saved models:****binary outputs (think pickle files) related to each run
* ****other outputs****such as images, CSV, text, HTML that results from your code
* ****source****: the script/notebook filename responsible for the run + the git commit
* ****tags and comments:**** individual or team annotations

# ****How do I use MLflow in my code?****

Quite easy.

* Start by installing mlflow using pip :

****pip install mlflow****

* Import mlflow and use the set\_tracking\_uri method to set the path to where MLflow will store the results of each run
* Call the create\_experiment method with an appropriate name



To be able to log your parameters, metrics and save your models on each run, you’ll have to wrap your code inside the mlflow.start\_runcontext by specifying the experiment\_id as argument.

Inside this context, MLflow creates a run with a unique id ****(run\_id).****This run will wait for any information to be tracked.

* Saving a parameter can be done using the mlflow.log\_param method
* Saving a metric can be done using the mlflow.log\_metric method
* Saving a (scikit-learn) model as artifcat can be done in a simple way by calling the mlflow.sklearn.log\_model



# ****Ok, What happens now? Where are the results?****

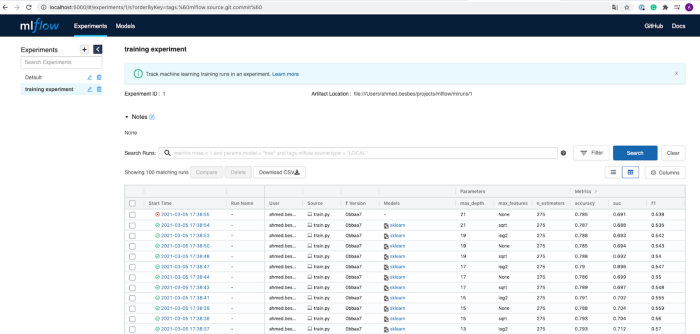
MLflow provides a great UI to visualize the results of each run as well as compare runs between each other.

To start the UI, run the following command at the same location of the mlruns directory:

****mlflow ui****

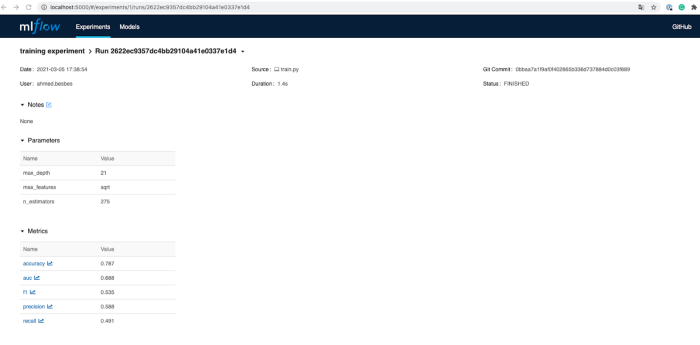
This will start a local server at port 5000.

On the left side, you’ll see the experiments. On the right side, you’ll see the runs. The table is interactive, you can sort the runs, search them by specifying queries and even compare them (in a second view).

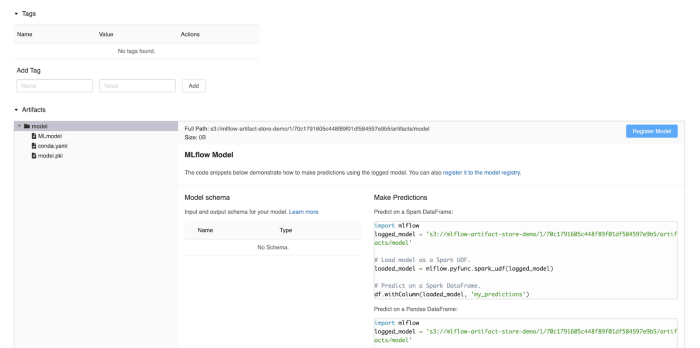


Once you click on a run, you are redirected to another page with more details.

Metrics, parameters, duration, status, git commit:



as well as model output and artifacts:



# ****Set up a tracking server on AWS****

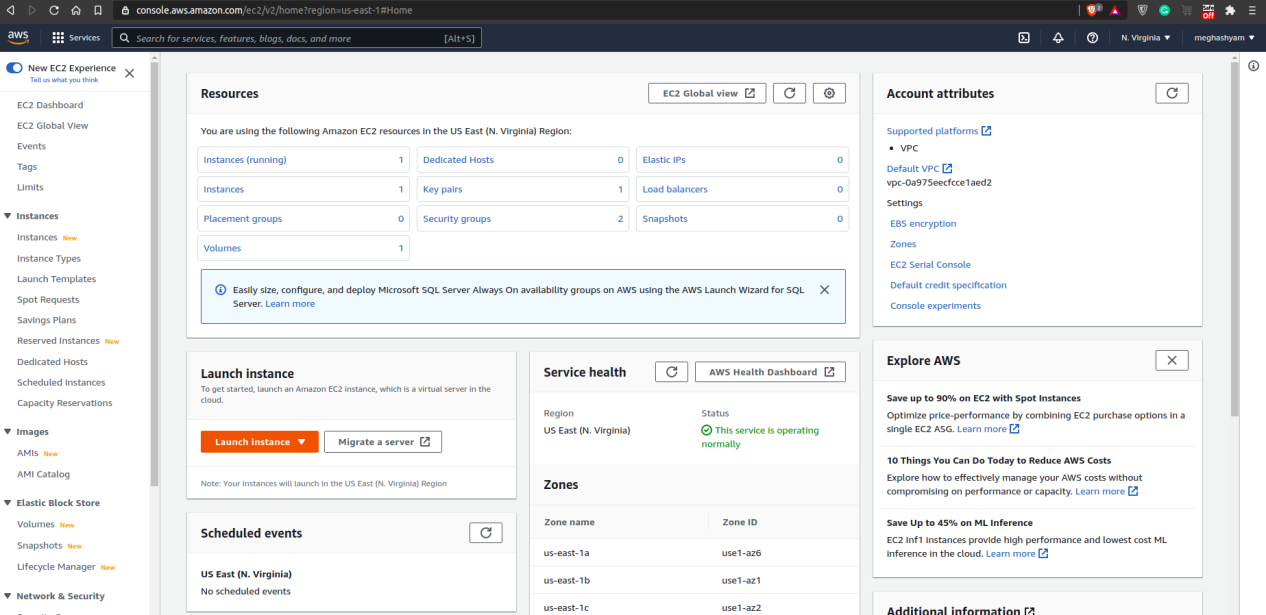
So far, we’ve used MLflow locally. This is not ideal if we want to collaborate with other colleagues and track experiments in a team.

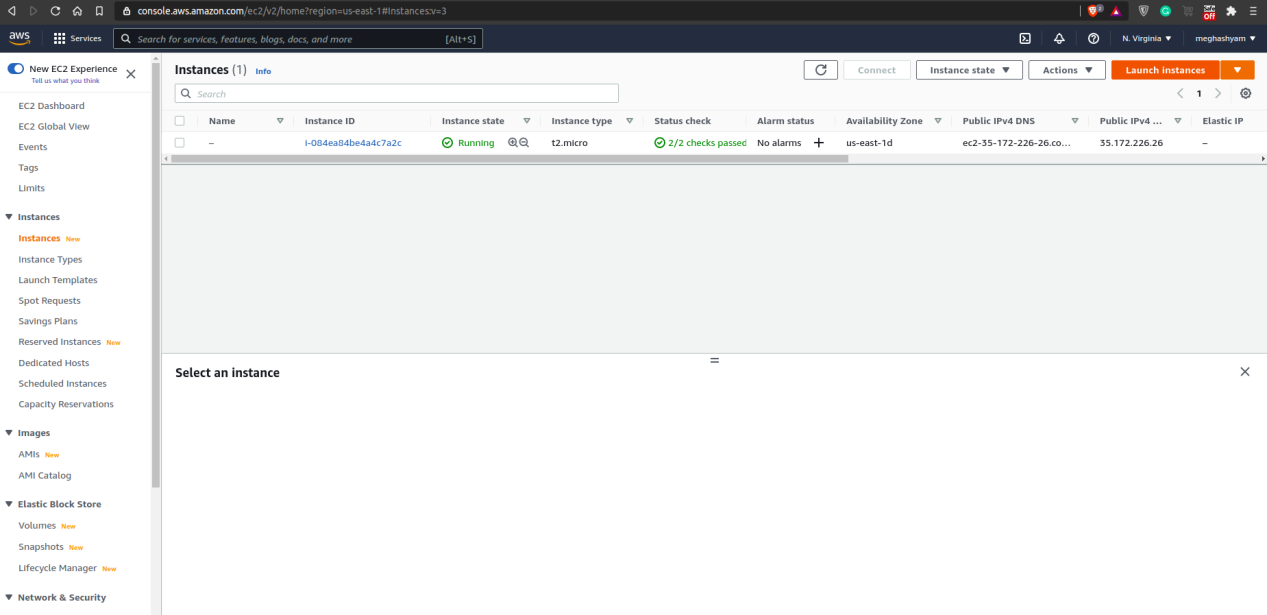
Fortunately, setting up a remote MLflow tracking server is quite easy. Let’s see how this done.

****In this section I’ll be using AWS, so make sure you have an account if you want to reproduce these steps.****

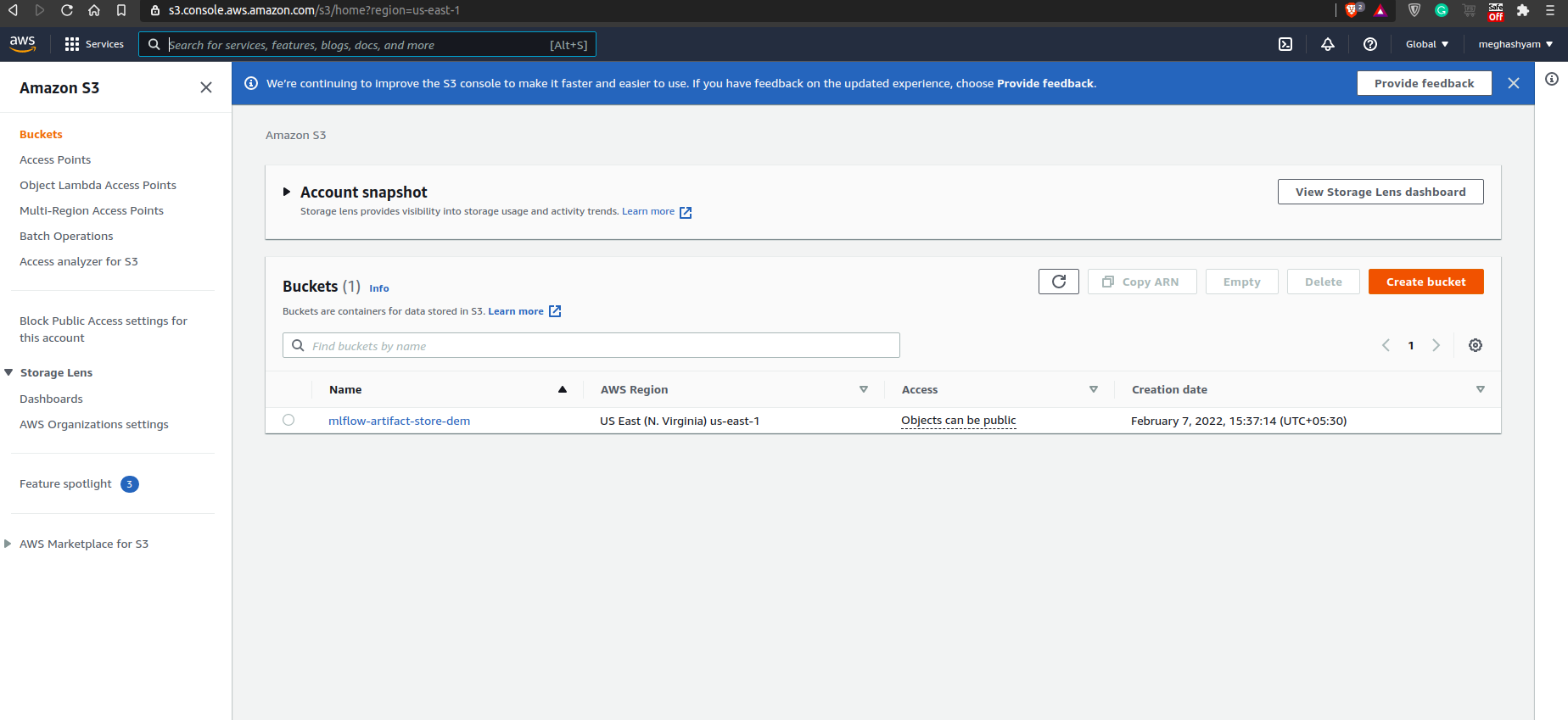
****1 — Set up a remote EC2 machine with MLflow****

* Create an IAM user. Grab the Access key ID and Secret access key credentials and store them somewhere safe. We’ll need them later.





* With this same user, create an s3 bucket to store future artifacts: give this bucket a name. Mine is mlflow-artifact-store-dem but you cannot pick it. Note that you don’t need custom configuration for your bucket (it doesn’t need to be public for instance)



* Launch an EC2 instance: it doesn’t have to be big. At2.micro eligible to free tier does perfectly the job
* Configure the security group of this instance to accept inbound HTTP traffic on port 5000 and any IP address so that the MLflow server is accessible from the outside
* Connect to your instance using SSH and run the following commands to install pip, pipenv and mlflow

# install pip  
sudo apt update  
sudo apt install python3-pip# install   
sudo pip3 install pipenv  
sudo pip3 install virtualenv  
  
export PATH=$PATH:/home/[your\_user]/.local/bin/# install mlflow, awscli and boto3  
pipenv install mlflow  
pipenv install awscli  
pipenv install boto3

* On the EC2 machine, configure AWS with the user’s credentials so that the tracking server can have access to s3 and display the artifacts on the UI.  
  Enter aws configure then follow the instructions to enter the credentials
* Start an MLflow server on the EC2 instance by defining the host as 0.0.0.0 and the --default-artifact-root as the S3 bucket

****mlflow server -h 0.0.0.0 --default-artifact-root s3://mlflow-artifact-store-demo****

Now your EC2 machine is properly configured.

****2 — Set up your environment****

To allow MLflow to push runs from your local environment to EC2 and S3, you’ll have to:

* Install boto3 locally via pip install boto3
* Set the AWS credentials as environment variables so that MLflow has the right permissions of S3 read and write access

****export AWS\_ACCESS\_KEY\_ID=<your-aws-access-key-id>  
export AWS\_SECRET\_ACCESS\_KEY = <your-aws-secret-access-key>****

* change the tracking URI in your code to the ****HTTP://<public-dns>:5000****

Now everything should be set up correctly.

If you execute your code again, the runs will no longer be saved locally but on the remote tracking server, which you can check by visiting the URL.

