Aalto University
School of Science

STUDY LOG 4 (Project Report)

Author:

Omer Ahmed Khan

802062

omer.khan@aalto.fi

CS-E4002 - Special Course in Computer Science: Advanced Topics in Systems for Big Data and Machine Learning

SPRING 2020

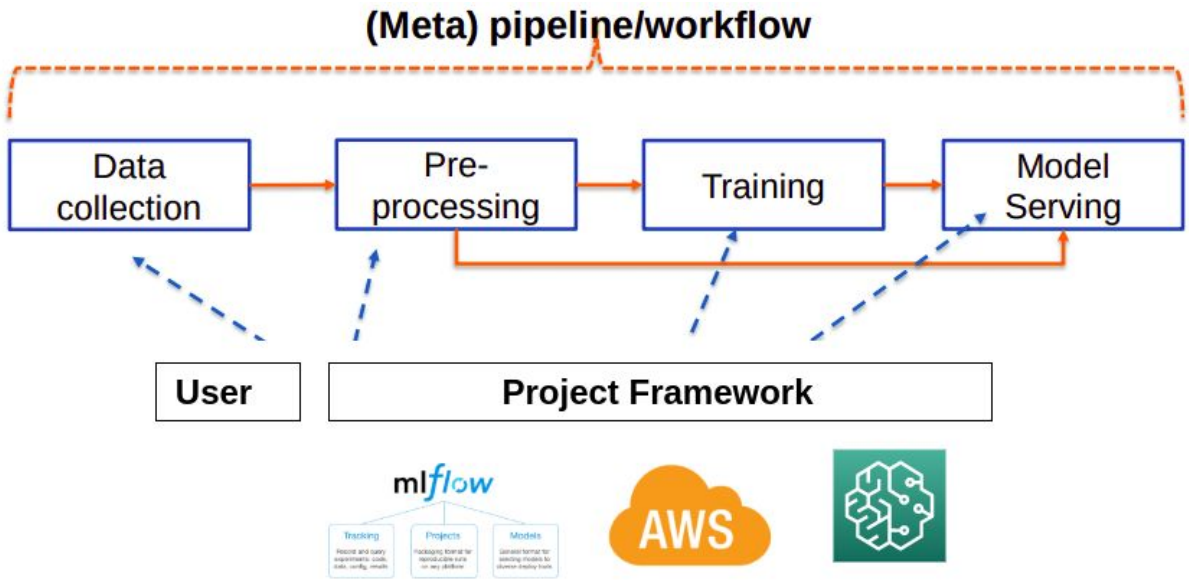
Modeling Automation - From Human to Machine

The main concept of this project to enable the customer/user to experiment and deploy his Machine Learning Models to cloud which in the end result provide highly scalable endpoints which can be consumed as per customer application. This project is merely a platform which can be used for multi-users with no restrictions of which dataset or Machine learning algorithms to be used. The experimentation and data will be provided by the customer which will be validated by his own rules on our platform and then provide experiments results, which then be published ton demand to serve scalable solution.

From the course study, a different concept was utilized to model this solution. From the most famous R3E (Reliability, Resilience, Robustness & Elasticity), I emphasize on Elasticity which enables scaling and then robustness which allows users to experiment with rapidness. Moreover, from the F3 (Frameworks), I used the open-source framework MLFlow, which is contributed and used by many professionals. From this, I focus on two concepts, or rather two methodologies to provide a final solution

- Orchestration to enable Experince Management through MLflow
- Model serving by deploying the final model to cloud to provide Elasticity

A higher level of flow diagram could be seen as below:



To understand this, we can easily able to see that there are two contributors to this system. One is the user itself who want to get services and Framework itself. The responsibility is divided as follow:

USER:

- Should create a script to do his experimentation. This script should follow some guideline provided by Framework. It can include some config file for data validation and code structure.
- Execute all experimentation on provided MLFlow tracking URI

FRAMEWORK:

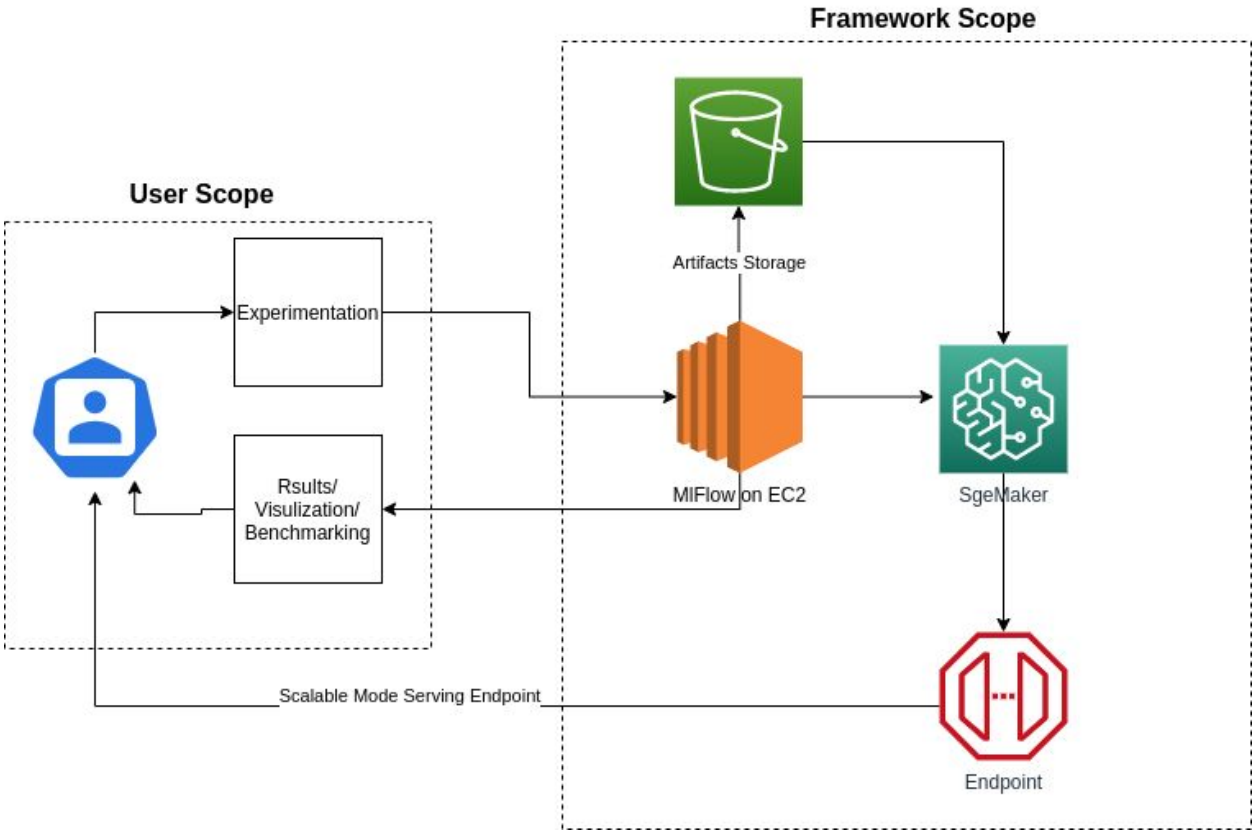
- Should execute experiment provided by users
- Provide Experiment Management View (Provided by MLflow)
- Store Artifacts to specific locations on the cloud
- Deploy Model to Sagemaker to provide a scalable endpoint to serve the model

Architecture:

There are two scopes for an end to end solution, one is what the user gets and on is what and how the framework provides services. The framework works over AWS cloud services. The MLFlow is deployed over EC2 where the experimentation management will occur. The user needs to use the Tracking URI to perform his experimentation. The artefacts produced by this experimentation will store to S3 buckets which then utilize to deploy the final solution to Sagemaker as a docker image.

The user can see the results of his experimentation on MLFlow UI which is deployed over EC2, and can have a visualization of his results and experiment management. On his wish then he can issue a request to deploy the final solution as an endpoint service point, which converts models as a server. Which means the model will transform to an HTTP request which is provided with a data sample can return the predicted result.

The Architecture is defined in a way that it can provide high Elasticity. The ML FLow is deployed over EC2 behind a load balancer which will autoscale its instance with the load. Due to which the Mlflow instance will be scale with high throughput. Moreover, the artefacts are stored in AWS scalable storage S3 which is handled by AWS itself with a prominent feature of elasticity. Furthermore, to do ML model serving we are using AWS Sage maker which is again a scalable service which deploys an image of ML model and then AWS handles its elasticity itself.



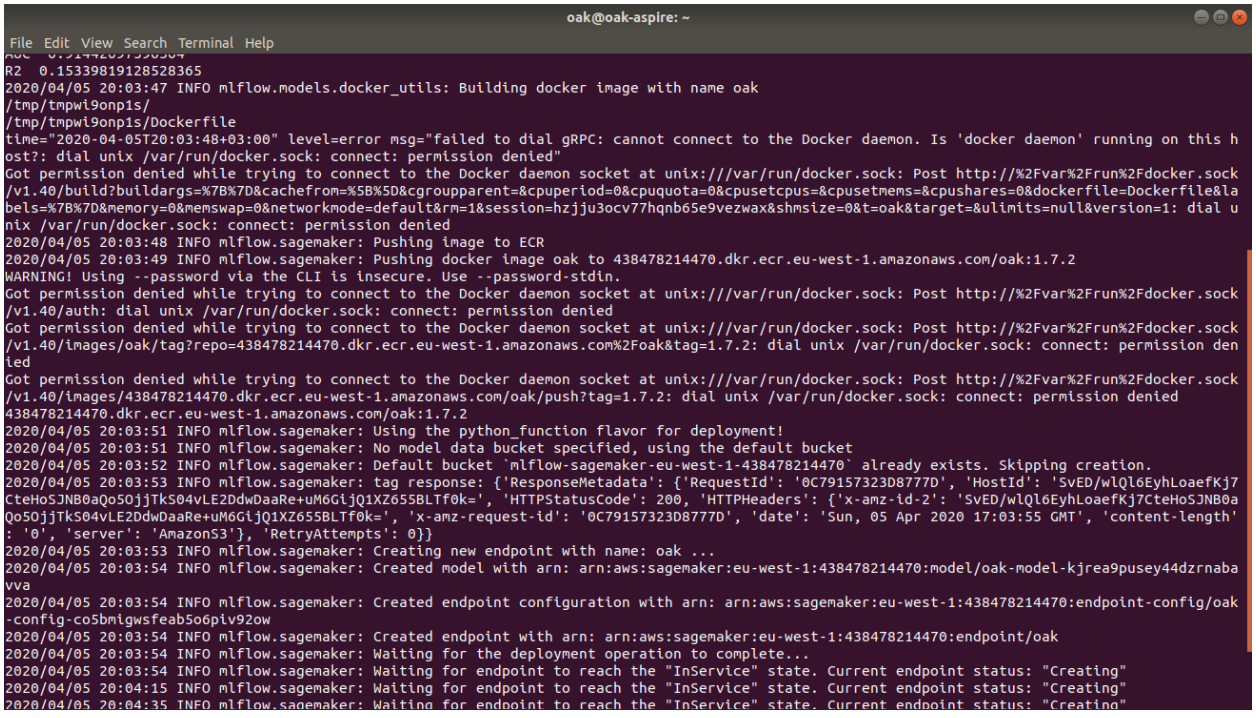
Use Case/Story:

The User from XYZ industry wants to evaluate its dagta and wants to do predictions on its dataset. But it needs a solution through which he can evaluate its experimentation and then finally want to deploy it with elasticity. However, he may not have the ability to tackle all the DEV-OPS stuff. Which means he needs someone or something to help him to do all this work for him. This platform is best fit for such a scenario. The user can perform all his logics with his dataset in a privacy manner as the system doesn't require a dataset to upload and finally deploy it in a scalable way.

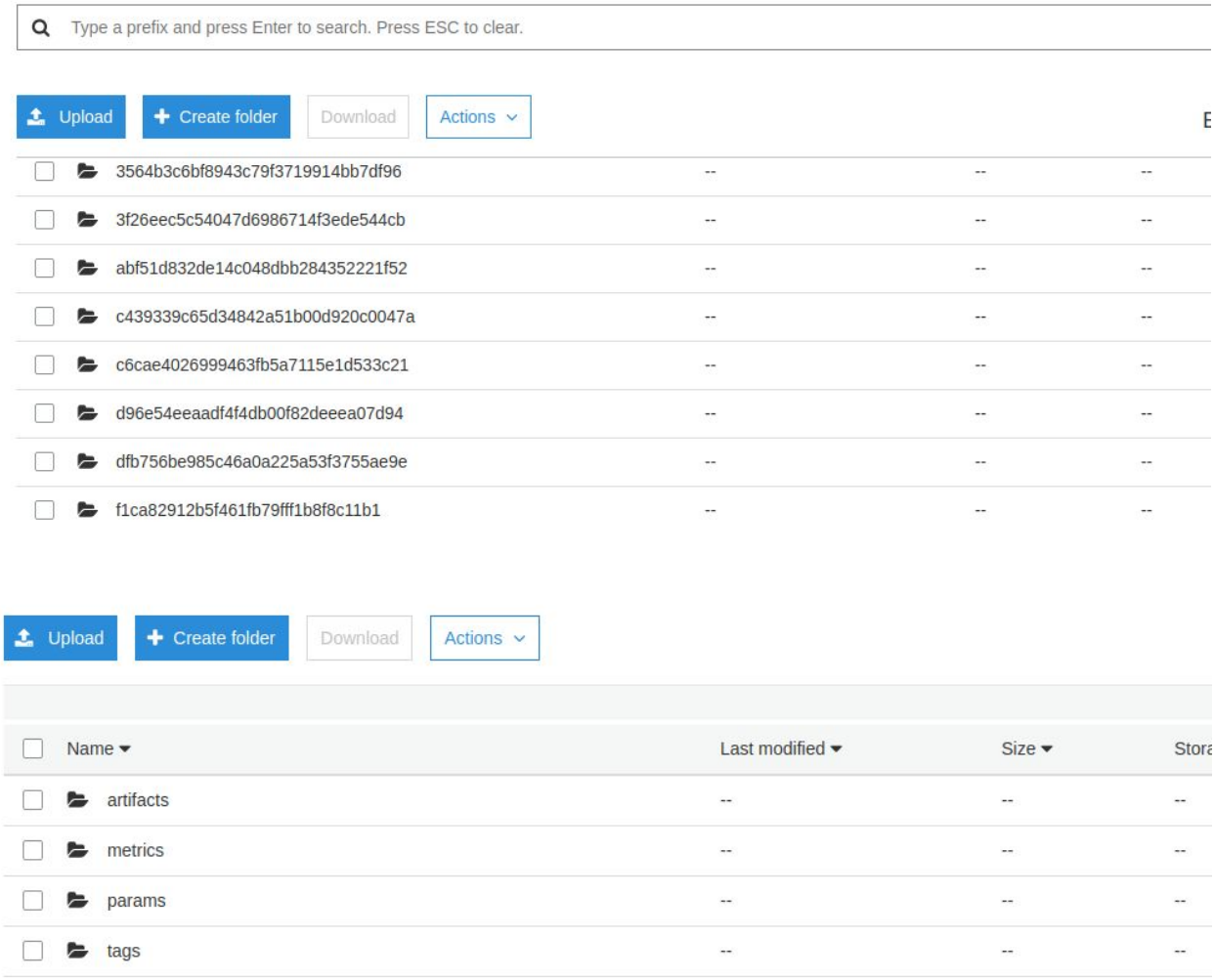
Now let's say, A data scientist at XYZ company wants to predict/forecast the sales and he has all the data he needs. But he needs to compare his experimentation and not only that he needs to deploy the final iterations and make it scalable to be used in production. The platform can provide a platform of MLflow to do all the experimentation while all its data remains on its premises. All the experimentation is stored on my platform and then on just a single API call it can deploy it to the cloud and provide the endpoint to be used by User in any way.

Results and Monitoring

There are certain monitoring and logging done in a framework. Most of the monitoring done by AWS CloudWatch or simple AWS modules consoles, which can provide us with all the traffics insight and the resource allocation and utilization. One of our critical deploying script which deploys on-demand ML model to Sagemaker store logs as follows:

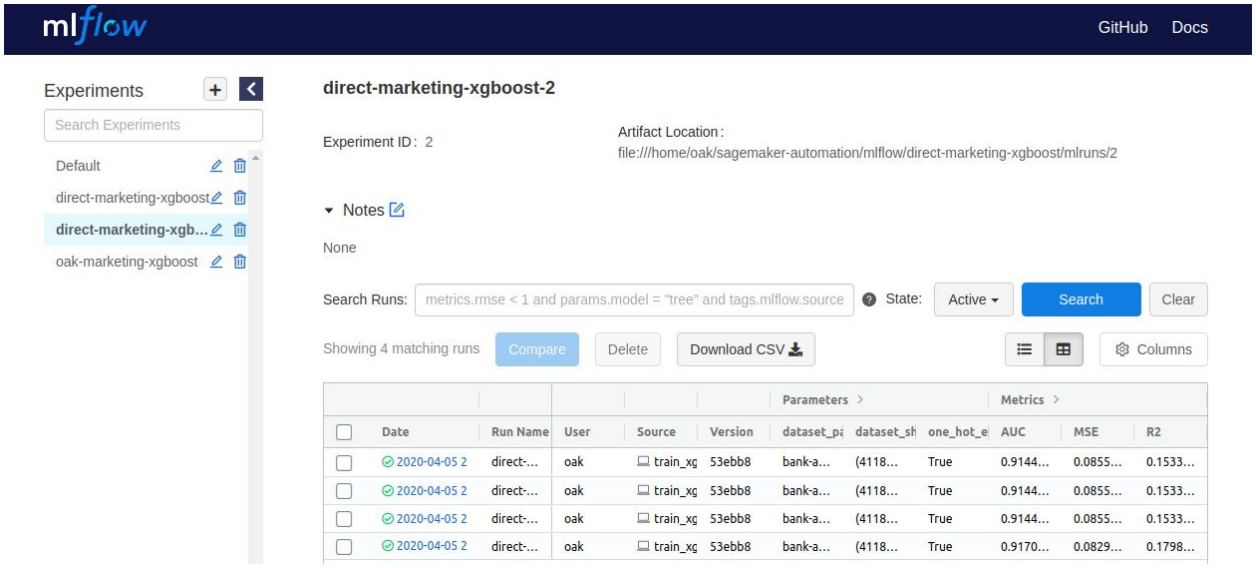


This is minimal logs which are created IF the image is already created otherwise the process takes more time and all the dependencies logs are generated. In deploying process certain things take parts like we need to have artefacts store in a bucket to have proper versioning and scalability. The artefacts in a bucket look like below:



All the experimentation are secured and scale with s3, which then utilize by MLflow framework from our EC2 instance to do experiment management and visualization of results and by our Sagemaker to create a Docker container and then deploy it with automated fashion to generate us with endpoint yet to be utilized by the user.

It's up to the user of how and what he wants to see on Mlflow application, as we are not concerned with what he codes or use, we just show him of the results he logs into ML flow framework, some of the snippets from different results are as follow from Mlflow Experiment management platform.


mlflow
GitHub
Docs
Experiments
Search Experiments
Default
direct-marketing-xgboost
direct-marketing-xgb...
oak-marketing-xgboost
direct-marketing-xgboost-2
Experiment ID: 2
Artifact Location:
file:///home/oak/sagemaker-automation/mlflow/direct-marketing-xgboost/mlruns/2
Notes
None
Search Runs:
metrics.rmse < 1 and params.model = "tree" and tags.mlflow.source
State:
Active
Search
Clear
Showing 4 matching runs
Compare
Delete
Download CSV
Columns
Parameters
Metrics
Date
Run Name
User
Source
Version
dataset_pa
dataset_sh
one_hot_e
AUC
MSE
R2
2020-04-05 2
direct-...
oak
train_xg
53ebb8
bank-a...
(4118...
True
0.9144...
0.0855...
0.1533...
2020-04-05 2
direct-...
oak
train_xg
53ebb8
bank-a...
(4118...
True
0.9144...
0.0855...
0.1533...
2020-04-05 2
direct-...
oak
train_xg
53ebb8
bank-a...
(4118...
True
0.9144...
0.0855...
0.1533...
2020-04-05 2
direct-...
oak
train_xg
53ebb8
bank-a...
(4118...
True
0.9170...
0.0829...
0.1798...

mlflow GitHub Docs

oak-marketing-xgboost > direct-marketing-xgboost-basic

Date: 2020-04-05 20:51:19 Source: train_xgboost.py Git Commit: 53ebb851b747d70e3ed676e94b2a5cc584053f1a

User: oak Duration: 3.5s Status: FINISHED

▾ Notes

None

▾ Parameters

Name	Value
dataset_path	bank-additional/bank-additional-full.csv
dataset_shape	(41188, 64)
one_hot_encoding	True
parameter	1
random_state	123
test_size	0.2

▾ Metrics

Name	Value
AUC	0.914
MSE	0.086
R2	0.153
metric	2

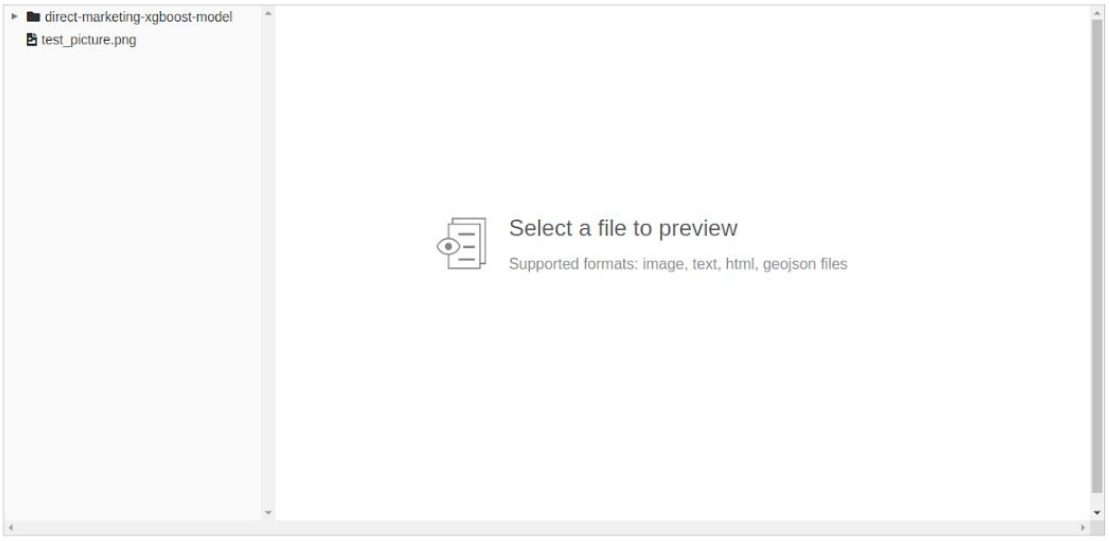
▾ Tags

Name	Value	Actions
	No tags found.	

Add Tag

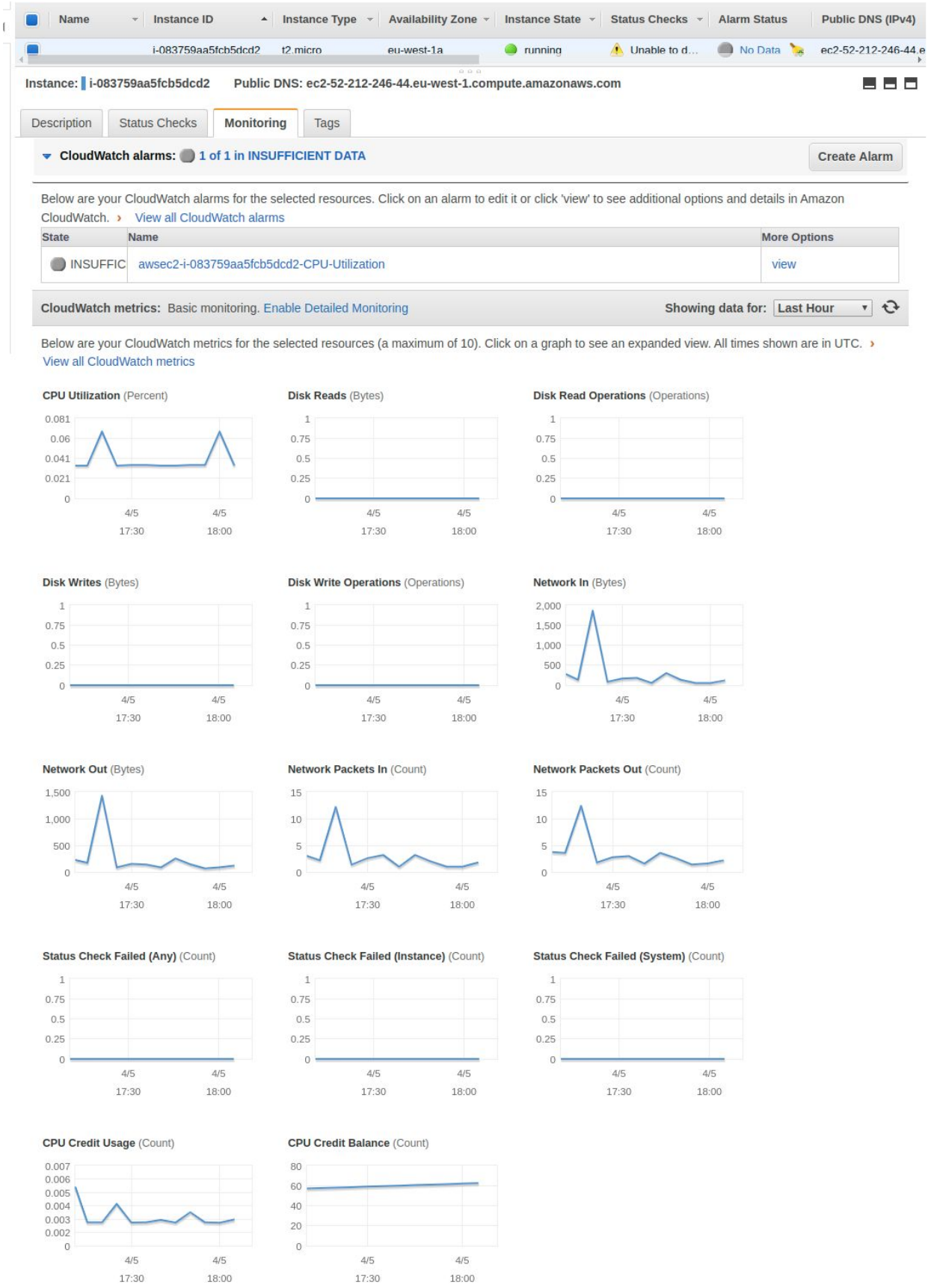
Name Value Add

▾ Artifacts



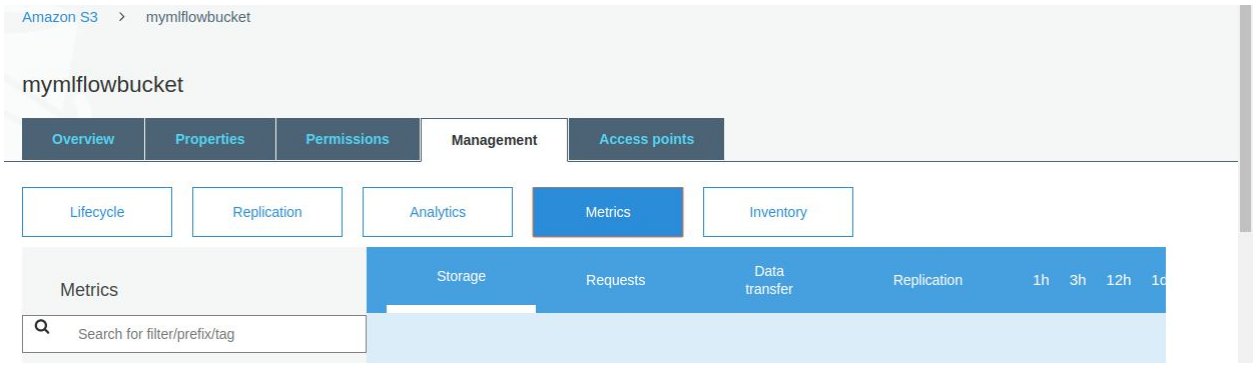
As you can see, a user can publish its experiment and can easily monitor. This feature, however, belongs to MlFlow framework and my project is just utilizing it. But my project allows users to store all its experimentation on my platform which can be reused in future in anyways. These experiments can log metrics and graphs which can be used to compare the result and to make the decision of which model needs to be deployed further to create an endpoint for related applications.

This was, however, is a story from a user perspective. The monitoring done from the platform is different and a customer/user is of no concern to them. However, as a platform engineer, we need to monitor them and put alerts on them to identify pressing or critical bugs and issues which can bring our system down. These measures can cover yet again in another **R3E** of being robust in approach and detect any resource or technical issues. Following are the resulting monitoring from the AWS platform which is tightly coupled with our platform modules.


Name
Instance ID
Instance Type
Availability Zone
Instance State
Status Checks
Alarm Status
Public DNS (IPv4)
i-083759aa5fcb5dcd2
t2.micro
eu-west-1a
running
Unable to d…
No Data
ec2-52-212-246-44.e
Instance: i-083759aa5fcb5dcd2
Public DNS: ec2-52-212-246-44.eu-west-1.compute.amazonaws.com
Description
Status Checks
Monitoring
Tags
CloudWatch alarms: 1 of 1 in INSUFFICIENT DATA
Create Alarm
Below are your CloudWatch alarms for the selected resources. Click on an alarm to edit it or click 'view' to see additional options and details in Amazon CloudWatch. › View all CloudWatch alarms
State
Name
More Options
INSUFFIC
awsec2-i-083759aa5fcb5dcd2-CPU-Utilization
view
CloudWatch metrics: Basic monitoring. Enable Detailed Monitoring
Showing data for: Last Hour
Below are your CloudWatch metrics for the selected resources (a maximum of 10). Click on a graph to see an expanded view. All times shown are in UTC. ›
View all CloudWatch metrics
CPU Utilization (Percent)
Disk Reads (Bytes)
Disk Read Operations (Operations)
Disk Writes (Bytes)
Disk Write Operations (Operations)
Network In (Bytes)
Network Out (Bytes)
Network Packets In (Count)
Network Packets Out (Count)
Status Check Failed (Any) (Count)
Status Check Failed (Instance) (Count)
Status Check Failed (System) (Count)
CPU Credit Usage (Count)
CPU Credit Balance (Count)
4/5
17:30
18:00

The above image shows the resource utilization on our MlFlow EC2 instance which is our main point of contact to a customer. The Mlflow Tracking server and a small Flask server is built on this EC2 which serve Customer requests either of experimentation or on-demand ML model deployment. This instance is behind the load balancer to avoid any jam from the load and to maintain availability. Furthermore, we also need to monitor Sagemaker endpoint, below images can describe how it utilizes resources. However, this management is solely handled by AWS so we do not need to apply any load balancer or anything but we need to monitor and apply appropriate alarms to avoid any breakage. (PLEASE REFER IMAGE AFTER THIS PAGE)

Another thing we could monitor but it asked for more money so I didn't enable it. It can help us to monitor our S3 storage which needs to be controlled at some point as we need to limit our Customer on experimentation or charge them for extra experimentation artefacts storage.



oak

Delete

Endpoint settings

Name
oak

ARN
arn:aws:sagemaker:eu-west-1:438478214470:endpoint/oak

Status
InService

Creation time
Sun Apr 05 2020 20:03:55 GMT+0300 (Eastern European Summer Time)

Last updated
Sun Apr 05 2020 20:09:03 GMT+0300 (Eastern European Summer Time)

URL
https://runtime.sagemaker.eu-west-1.amazonaws.com/endpoints/oak/invocations
Learn more about the API

Data capture settings

Enable data capture
No

Current sampling percentage (%)
-

S3 location to store data collected

Data capture status
-

Monitor

Access Jupyter Notebook logs for debugging and progress reporting. Learn more

View invocation metrics | View instance metrics | View logs

Add to dashboard | 1h | 3h | 12h | 1d | 3d | 1w

Lesson Learn:

This course has taught us many things which are closely related to what is needed in the industry. From the origin concept of R3E (Reliability, Resilience, Robustness & Elasticity) to orchestration and ML modeling. With this project I try to apply each concept I learn, which can be linked as follows:

- R3E - In this project I focus on Elasticity and Robustness by using AWS and MLflow framework with each other which allow robustness from Mlflow and Scalability from AWS
- Benchmarking & Monitoring - With Mlflow a user can benchmark its experimentation while monitoring all its past experimentation and with the AWS monitoring system a cloud provider (e.g me) can monitor and set alarms on any critical bottleneck modules.
- Coordination of Big Data/ML Tasks - This is my favorite part, it really helps to build my project setup and can connect different pieces together. It also allow us as a student to understand ML serving and deployment on more clear way

Conclusion:

Overall my platform is a layer over MLFlow framework which enables users to do experimentation and deploy their perfected model over AWS Sagemaker making it an Elastic and robust platform to use. MLflow itself is just an open-source project which doesn't provide elasticity thus my project is to make the use of the framework to make it scalable and making it much easier to use by the different business applications. The concept of this project comes with the previous course Big Data Platform in which we are not concerned with the data but to provide a platform which enables different customers and solutions to be built and deployed over it.

Self-Grading: 5/5