<u>MLflow</u>

<u>Introduction to Experiment Tracking and Model Management</u>

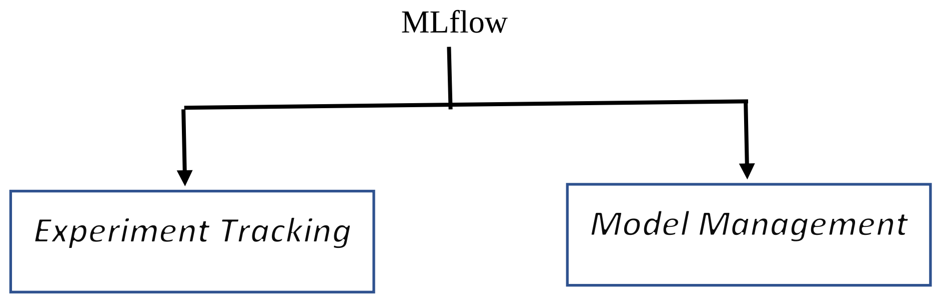
MLflow is an open-source platform to manage the ML lifecycle, including experimentation, reproducibility, deployment, and a central model registry. MLflow currently offers four components:

- **MLflow Tracking-**Record and query experiments: code, data, config, and results
- **MLflow projects-**Package data science code in a format to reproduce runs on any platform
- **MLflow models-** Deploy Machine learning models in diverse serving environments
- **Model Registry-** Store, annotate, discover, and manage models in a central repository

MLflow: Tracking of Experiments, logging or recording all the Experiments.
Login is important for Organization of Production pipeline properly.
MLflow interface is helping to track for-

- What kind of Algorithm
- What kind of Hyper parameters
- What kind of scores we get for the model.

For the production we choose best one based on the above data.



Terminologies:

1. Experiment -Each trial of some Experiment
2. Run
3. Metadata -All Information related to an Experiment Run (i.e. Tags, Parameters, Data, Train-Test size, Algorithms, Metrics)
4. Artifacts -Output files associated with experiment runs (i.e. Pickle files)

Why Track?

Organization is much easier
Optimization is much more meaningful

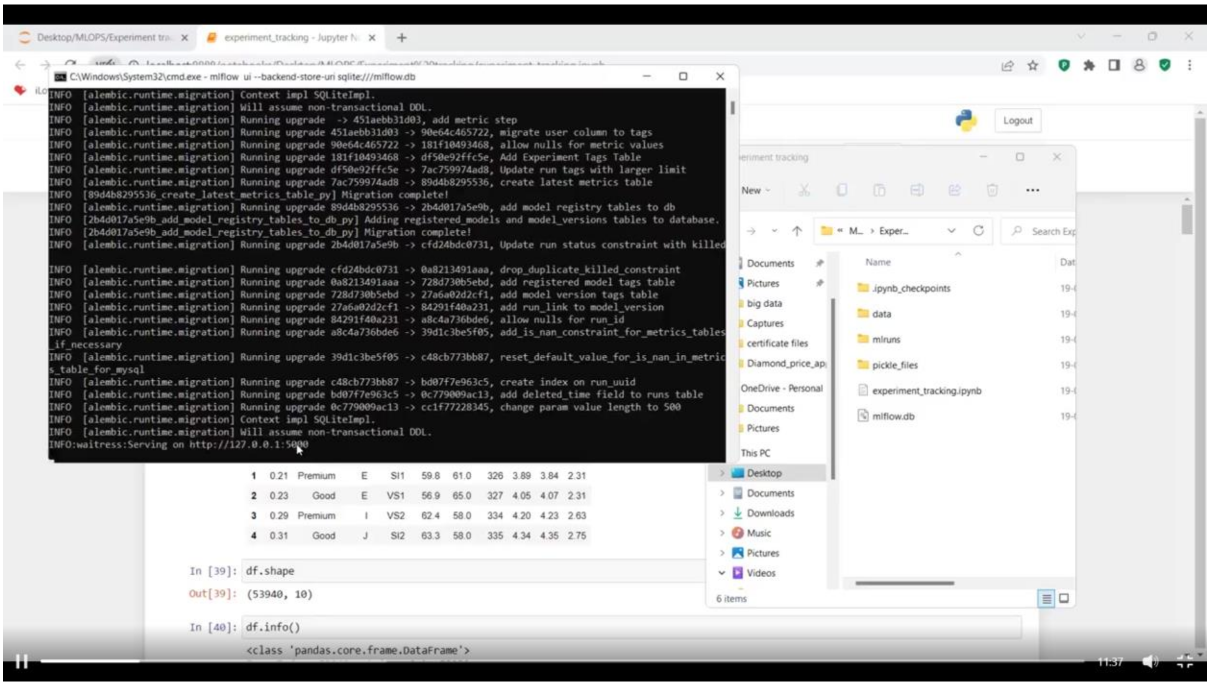
Reproducibility of Experiment run

Run below mentioned commands to install mlflow on your system:

pip install mlflow

mlflow ui --backend-store-uri sqlite:///mlflow.db

After executing above command in cmd we get user interface http://127.0.0.1:5000



Step 1 - Import MLFlow

```
import mlflow
```

Step 2 - Set the tracker and experiment

```
mlflow.set_tracking_uri(DATABASE_URI)
mlflow.set_experiment("EXPERIMENT_NAME")
```

Step 3 - Start a experiment run

```
with mlflow.start_run():
```

Step 4 - Logging the metadata

```
mlflow.set_tag(KEY, VALUE)
mlflow.log_param(KEY, VALUE) mlflow.log_metric(KEY, VALUE)
```

Step 5 - Logging the model and other files (2 ways)

Way 1 - `mlflow.<FRAMEWORK>.log_model(MODEL_OBJECT, artifact_path="PATH")`

Way 2 - `mlflow.log_artifact(LOCAL_PATH, artifact_path="PATH")`

Running the Experiment

```python
import mlflow
```

```python
mlflow.set_tracking_uri("sqlite:///mlflow.db")

mlflow.set_experiment("Diamond Price Prediction")
```

```
2022/09/19 13:28:55 INFO mlflow.tracking.fluent: Experiment with name 'Diamond Price Prediction' does not exist
<Experiment: artifact_location='./mlruns/1', experiment_id='1', lifecycle_stage='active', name='Diamond Price F
```

```python
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR
from sklearn.ensemble import GradientBoostingRegressor

from sklearn import metrics
```

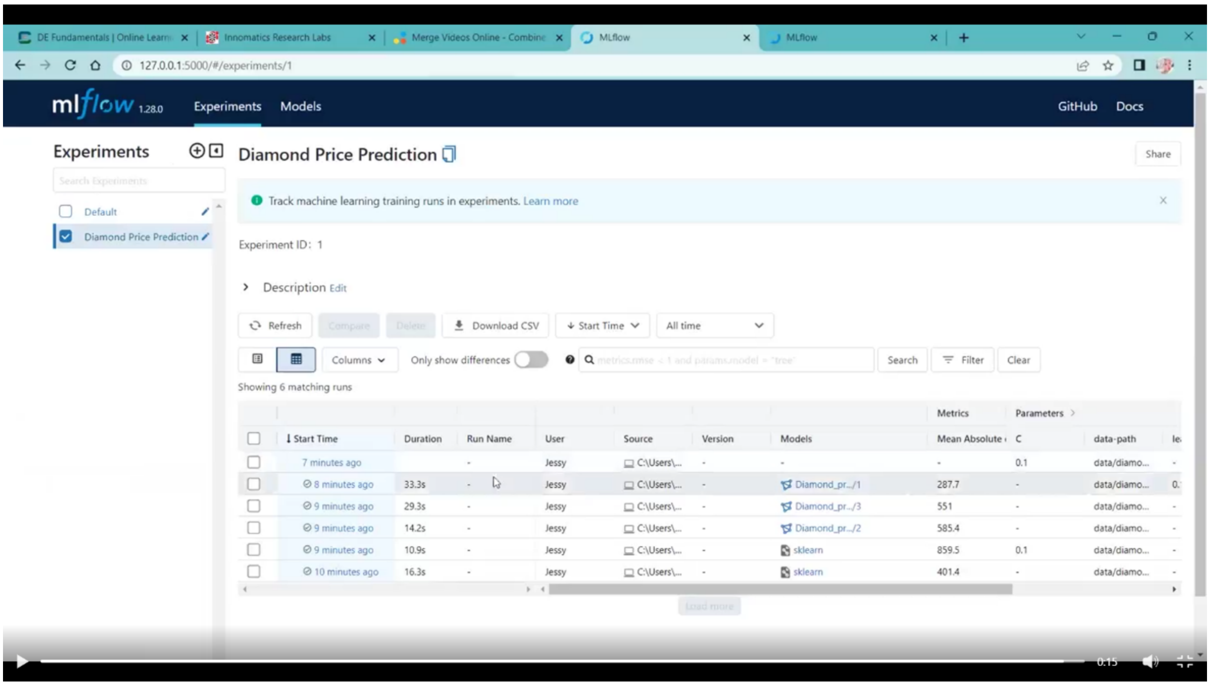
```python
from pickle import dump
dump(enc,open('pickle_files/Ordinal_Encoder.pkl','wb'))
dump(scaler, open('pickle_files/Standard_Scaler.pkl', 'wb'))
```

Experiment 1 - Training KNN Regressor

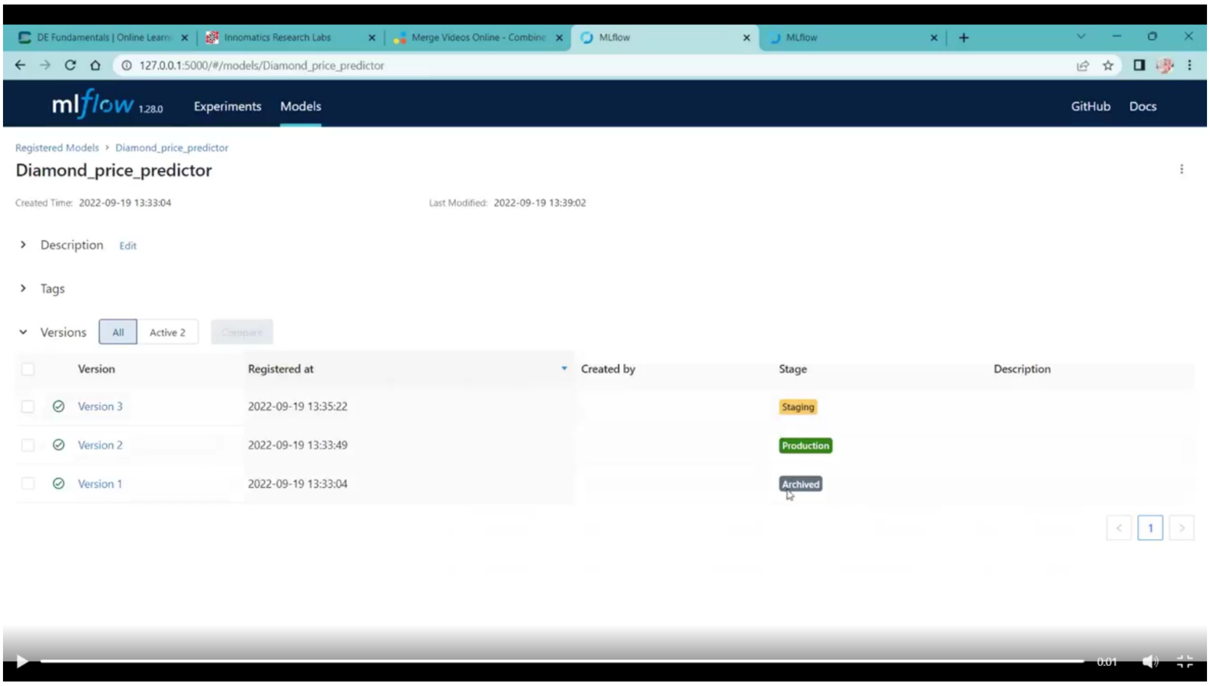
```python
with mlflow.start_run():
    mlflow.set_tag("dev", "Veena Grace")
    mlflow.set_tag("algo", "KNN")
    # log the data for each run using log_param, log_metric, log_model
    mlflow.log_param("data-path", "data/diamonds.csv")
    k = 3
    mlflow.log_param("n_neighbors", k)
    knn_regressor = KNeighborsRegressor(n_neighbors=k)
    knn_regressor.fit(X_train_rescaled, y_train)
    y_test_pred = knn_regressor.predict(X_test_rescaled)
    MAE = metrics.mean_absolute_error(y_test, y_test_pred)
    mlflow.log_metric("Mean Absolute error",MAE)
    mlflow.sklearn.log_model(knn_regressor, artifact_path="models")
    mlflow.log_artifact("pickle_files/Standard_Scaler.pkl")
    mlflow.log_artifact("pickle_files/Ordinal_Encoder.pkl")
```

After Executing the above code and open the URL to get the-

Mlflow Interface For Tracking Experiments



MLFlow Interface for Model Management



<u>PREFECT</u>

<u>ORCHESTRATE ML PIPELINES</u>

Managing Machine Learning Workflows using the tool Prefect 2.0

Why Prefect?

- Python-based open source tool
- Manage ML Pipelines
- Schedule and Monitor the flow
- Gives observability into failures
- Native dask integration for scaling (Dask is used for parallel computing)

Creating And Activating A Virtual Environment

In order to install prefect, create a virtual environment:

$ python -m venv mlops
Enter the Virtual Environment using below mentioned command:

$.\mlops\Scripts\activate

Installing Prefect 2.0

$ pip install prefect
OR if you have Prefect 1, upgrade to Prefect 2 using this command:

$ pip install -U prefect
OR to install a specific version:

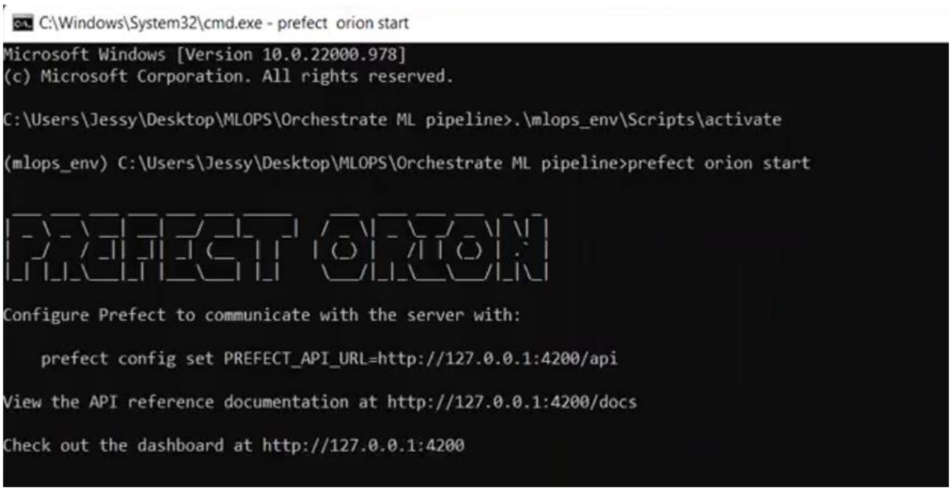
$ pip install prefect==2.4

Check Prefect Version

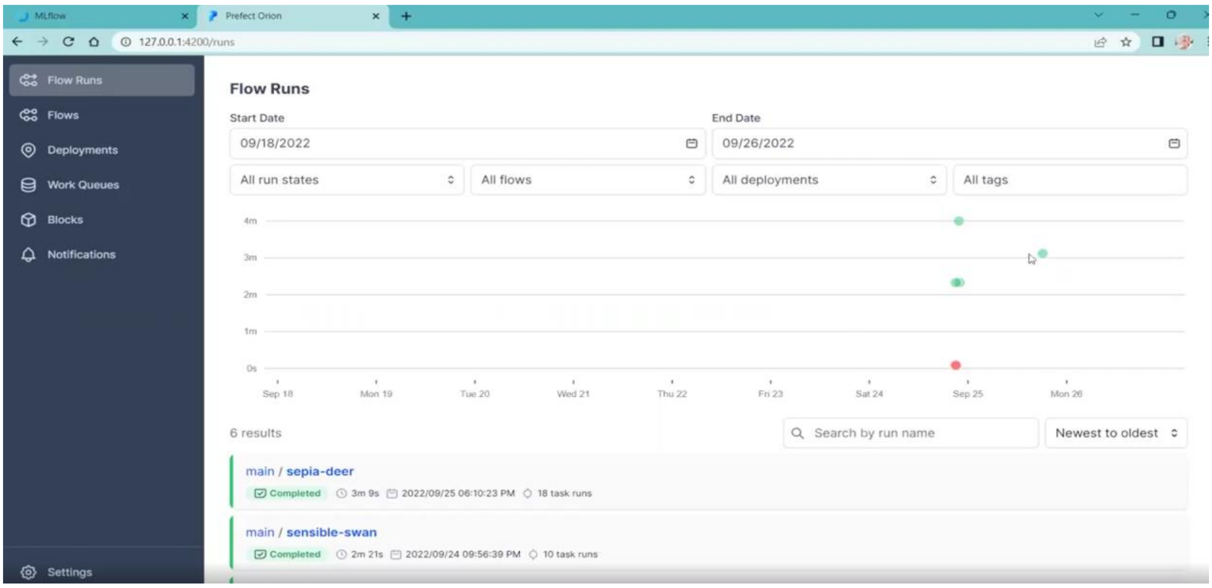
$ prefect version

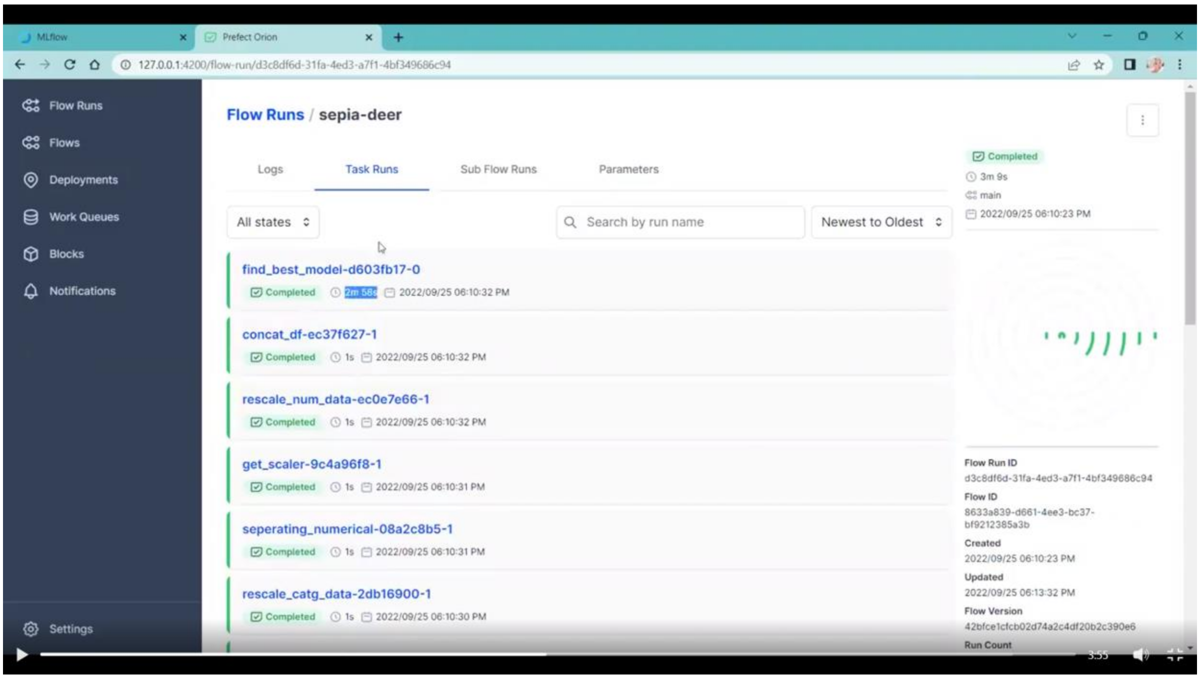
Running Prefect Dashboard

```
$ prefect orion start
```

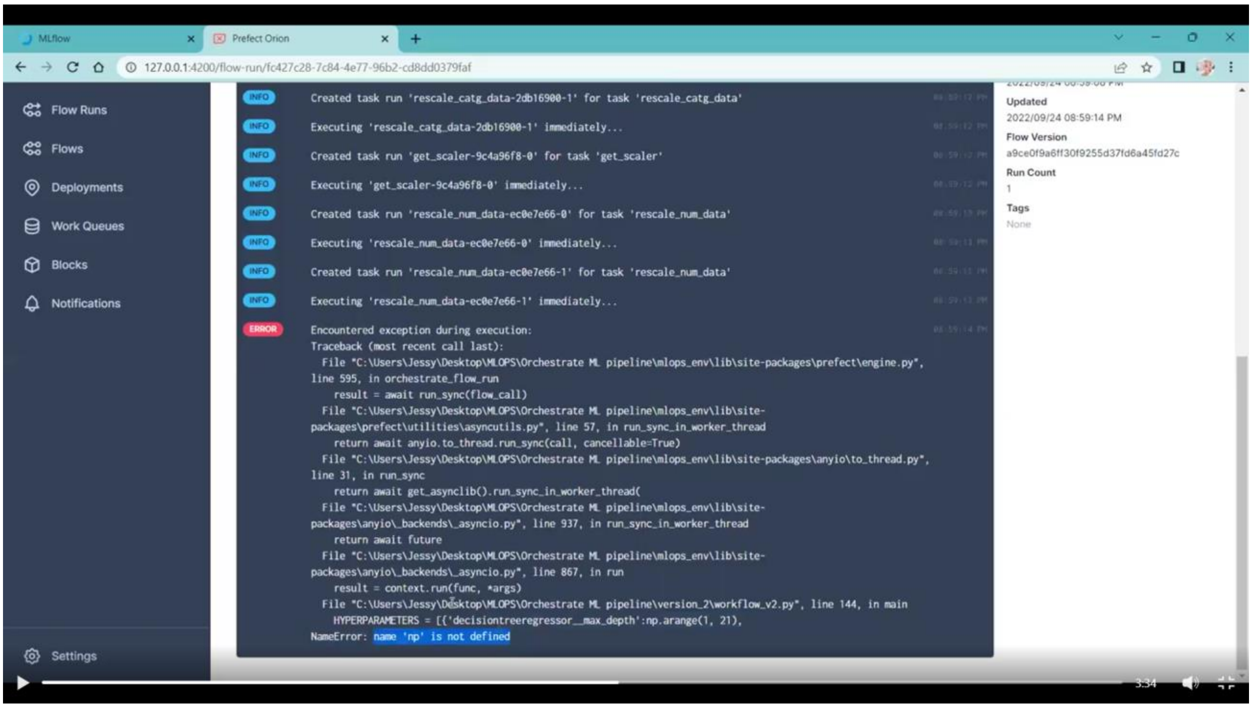


After starting the dashboard at http://127.0.0.1:4200 the below screen is opened.

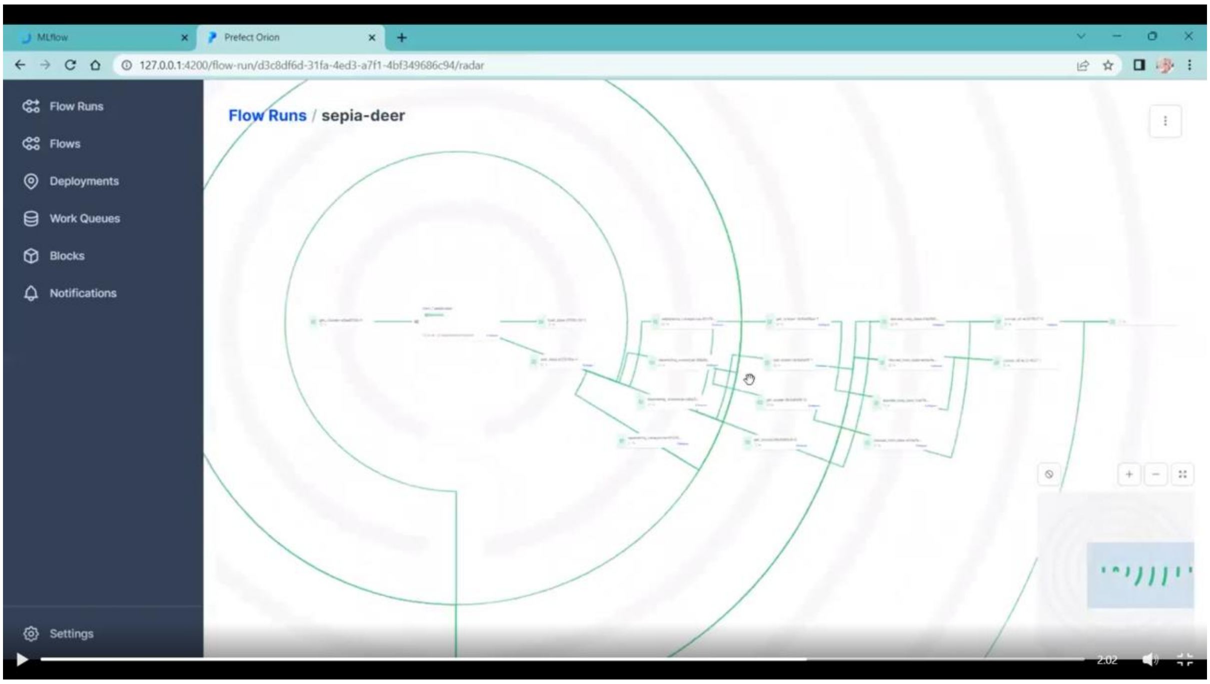




By using this prefect we can also detect the errors in a detailed manner.

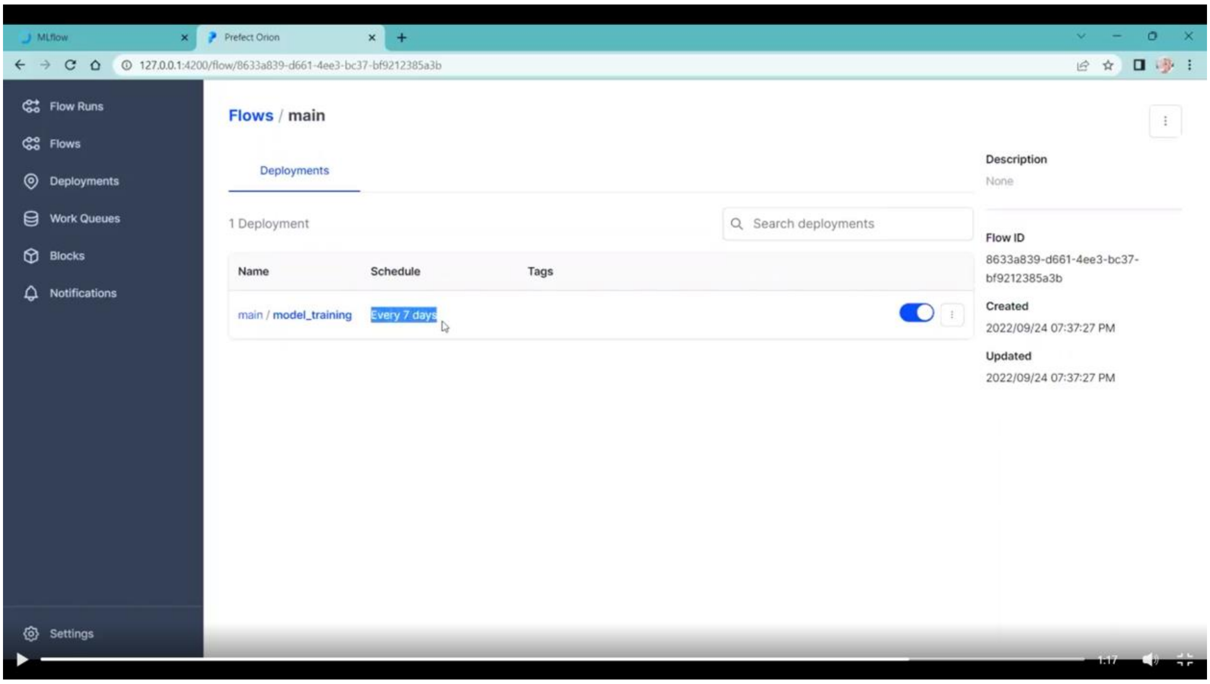


Flow runs displayed as



Deployment of Prefect Flow

- work_queue_name is used to submit the deployment to the a specific work queue.
- You don't need to create a work queue before using the work queue. A work queue will be created if it doesn't exist.



Running an Agent

$ prefect agent start --work-queue "ml"