Instruction

Part1 DevOps

Step1: Create Github

1. Create a Github Repository https://github.com/ritabanchowdhury319/REPA-web.git
2. Clone it to your computer
   a. git clone https://github.com/ritabanchowdhury319/REPA-web.git

Step 2: Containerize the app and run

1. Make an application called hello.js

```
const http = require('http');

const hostname = '0.0.0.0';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

2. Make a file called Dockerfile
3. Put following code in Dockerfile :

```
FROM node:14
COPY . .
EXPOSE 3000
CMD [ "node", "hello.js" ]
```

4. Now go back to terminal and type :

```
docker build -t ritaban/hello-world .
```

5. Then from terminal type :

```
docker run -p 3000:3000 hello-world
```

Go to http://0.0.0.0:3000/ to verify the application is running.

Step 3: Deploy using Kubernetes

1. Login to docker repository : `docker login`
2. Push the image in repository : `docker push ritaban/hello-world`
3. minikube start
4. Make a file hello.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hello-app-deployment
  labels:
    app: hello-app
spec:
  replicas: 1
  selector:
    matchLabels:
      app: hello-app
  template:
    metadata:
      labels:
        app: hello-app
    spec:
      nodeSelector:
        type: backend
      containers:
        - name: hello-app
          image: ritaban/hello-world
          ports:
            - containerPort: 3000
```

5. Give following command: `kubectl apply -f hello.yaml`
6. Then : `kubectl get deployments`
7. Then: `kubectl expose deployment hello-app-deployment —type=NodePort`
8. `minikube service hello-app-deployment`

Part 2 MLOps

Step1: Create Github

3. Create a Github Repository https://github.com/ritabanchowdhury319/REPA-mlops.git
4. Clone it to your computer
    a. git clone https://github.com/ritabanchowdhury319/REPA-mlops.git

Step 2: Make Component

6. Make an application called component.py

```
import argparse
import os
from pathlib import Path
import pickle

import pandas as pd
import numpy as np

# for plotting


# to build the models
from sklearn.linear_model import Lasso
from sklearn.feature_selection import SelectFromModel

# to visualise al the columns in the dataframe
pd.pandas.set_option('display.max_columns', None)



def read_data(input1_path):
    csv_data = pd.read_csv(input1_path, error_bad_lines=False)
    return csv_data


# read data
X_train = pd.read_csv('/components/feature-engineering/xtrain.csv')
X_test = pd.read_csv('/components/feature-engineering/xtrain.csv')

X_train.head()

# remove duplicate rows
print(X_train.head())
# capture the target (remember that the target is log transformed)
y_train = X_train['SalePrice']
y_test = X_test['SalePrice']

# drop unnecessary variables from our training and testing sets
X_train.drop(['Id', 'SalePrice'], axis=1, inplace=True)
X_test.drop(['Id', 'SalePrice'], axis=1, inplace=True)
sel_ = SelectFromModel(Lasso(alpha=0.005, random_state=0))

# train Lasso model and select features
sel_.fit(X_train, y_train)
```

```
print(sel_.get_support())
selected_feats = X_train.columns[(sel_.get_support())]

# let's print some stats
print('total features: {}'.format((X_train.shape[1])))
print('selected features: {}'.format(len(selected_feats)))
print('features with coefficients shrank to zero: {}'.format(
    np.sum(sel_.estimator_.coef_ == 0)))


print(selected_feats)
selected_feats = X_train.columns[(sel_.estimator_.coef_ != 0).ravel().tolist()]
pd.Series(selected_feats).to_csv('../selected_features.csv', index=False)
```

7. Make a file called Dockerfile
8. Put following code in Dockerfile :

```
FROM python
RUN python3 -m pip install pandas
RUN python3 -m pip install sklearn
COPY ./src/component.py /components/feature-engineering/src/component.py
COPY xtrain.csv /components/feature-engineering/xtrain.csv
COPY xtest.csv /components/feature-engineering/xtest.csv
RUN python3 /components/feature-engineering/src/component.py
```

9. Now go back to terminal and type :

```
docker build -t ritaban/feature-engineering .
```

Step 3: Deploy using Kubeflow

9. Login to docker repository : `docker login`
10. Push the image in repository : `docker push ritaban/feature-engineering`
11. Make a file component.yaml

```
name: feature-engineering
description: Performs the IOB preprocessing.
implementation:
  container:
    image: docker.io/ritaban/feature-engineering:latest
    command: [
      python3, /components/feature-engineering/src/component.py,
    ]
```

12. Go to Kubeflow User Interface : vagrant up
13. Make a pipeline in notebook
14. Run Pipeline

Requirements:

- Node js.
- Docker Desktop
- MiniKF
- MiniKube
- Personal Docker Registry