Build Classification Algorithms [Banking] Deployment

Business Objective

MLOps is a means of continuous delivery and deployment of a machine learning model. Practicing MLOps means that you advocate for automation and monitoring at all steps of ML system construction, including integration, testing, releasing, deployment, and infrastructure management.

In this project, we will be deploying the machine learning application for the Build Classification Algorithms for Digital Transformation [Banking]. Hence it is advised to go through this project first before. This project uses Amazon EKS (cloud platform), Amazon EC2, Elastic Load Balancing, etc services. Amazon EKS is a fully managed service that makes it easy to deploy, manage, and scale containerized applications using Kubernetes on AWS.

Aim

Deploy a machine learning model to identify the potential borrower customers to support focused marketing and deploy them through a cloud provider (AWS)

Tech stack

- Language - Python
- Services - AWS EKS, ECR, Load balancer, code commit, code deploy, code pipeline

Prerequisites

It is advisable to have a basic knowledge of the following services to get an understanding of the project.

- Flask
- Aws ECR
- AWS ECS
- AWS EC2 Load balancer
- AWS Code commit
- AWS Code Build
- AWS Code Deploy
- AWS Code Pipeline

Approach

A. Cluster creation steps

1. Create an EKS cluster master node.
2. Create Node groups in the EKS cluster
3. Create OIDC connection in the AWS Identity provider with EKS cluster
4. Install Kubernetes dashboard for monitoring
5. Install AWS Load balancer controller

B. Create an ECR repository for the docker image

C. EKS Yamls for our application

1. Create deployment yaml
2. create service yaml
3. create ingress yaml

D. Code Pipeline for EKS

1. Create 2 code commit repositories one for flask ml application and another one for EKS yamls
2. Create 2 code build projects one for flask ml application and another one for EKS yamls
3. Create 2 code pipelines one for flask ml application and another one for EKS yamls

E. In the EKS yaml code pipeline, have both ECR repo and code commit repo as the source

F. Push ML application into EKS

1. Create Flask application for our ML Application
2. Convert that Flask application into a docker Application
3. Test the docker application in your local
4. Push the Flask code to code commit, check the pipeline runs without any issues.
5. Once the Flask code pipeline ran successfully, check whether it triggers the second EKS yaml code pipeline.
6. If the second pipeline is triggered, check the API response for the update values.

Files Required

1. Banking-classification folder
    I. Flask application folder: contains the modular code along with the app.py file
    - Input: CallCenterData.xlsx
    - MLPipeline: this folder contains all the functions put into different python files
    - Notebook: time series gaussian model ipynb file
    - Output: gaussian model saved in a pickle format
    - App.py: flask app configuration
    - Engine.py: File where the MLPipeline files are called
    - requirements.txt: essential libraries with their versions

    II. Dockerfile: docker image

2. Banking-classification-eks folder
    Yaml files for -
    - Aws-auth
    - Buildpsec
    - Deployment
    - Ingress
    - Namespace
    - Service

Project Takeaways

1. Understanding various services provided by AWS
2. Why is AWS security groups necessary?
3. What is Elastic Kubernetes Service (EKS)?
4. Creating AWS EKS clusters
5. Creating EKS worker nodes
6. How to connect EKS with kubectl
7. Kubernetes dashboard for monitoring
8. How to create a load balancer in the EC2 service?
9. Create code build repositories.
10. How to create and build a project in Code Build?
11. How to create Code Pipeline?
12. Create an ECR repository for the docker image
13. Creating yamls for deployment, service, and ingress
14. Creating code pipeline for EKS
15. Create a Flask application for ML application
16. Converting the Flask application into a docker application
17. Testing the docker container in local