MLOps on AWS project for Topic Modeling using Gunicorn Flask

Business Overview

Machine learning operations are widely known as MLOps include various technologies, processes, and practices that automate deployment, monitoring, and management of machine learning models in production. Many organizations are turning towards machine learning and Artificial intelligence. MLOps advocates automation and monitoring at all steps of the ML system. In this project, we aim to provide hands-on experience in MLOps by using cloud computing. Amazon Web Services(AWS) is used as a cloud provider. We would advise you to have a basic understanding of NLP Project on LDA Topic Modelling Python using RACE Dataset before jumping into this project.

Aim

To provide an end-to-end machine learning development process to design, build and manage reproducible, testable, and evolvable machine learning models by using Amazon Web Services(AWS)

Tech Stack

➔ Language: Python
➔ Libraries: Flask, gunicorn, scipy, nltk, tqdm, numpy, joblib, pandas, scikit_learn, boto3
➔ Services: Flask, Docker, AWS, Gunicorn

Prerequisites

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

- Flask
- AWS s3
- Aws ECR
- AWS ECS
- AWS EC2 Load balancer
- AWS Code commit
- AWS Code Build
- AWS Code Deploy
- AWS Code Pipeline

Approach

- Step 1
    1. Create Code Commit Repository
       Name: topic_modeling
    2. Create one branch apart from the main branch
       Name: TestBranch
    3. Commit the Initial codes to the `TestBranch`

- Step 2
    1. Create ECR Repository
       RepoName: topic_modeling

- Step 3
    1. Convert the ML Application as Flask Application
    2. Dockerize the flask application
    3. Test the Docker image in your local

- Step 4
    1. Create code build project
       Name: topic_modeling-video-demo
    2. Create the buildspec.yaml and upload it to the code commit repo
    3. Test your code build project

- Step 5
    1. Create ECS cluster
    2. Push your image to the ECR repo
    3. Create a task definition using the ECR Repo image URL
    4. Test your container in the ECS.
    5. Create a load balancer in the EC2 service
    6. Create service in the ECS using the load balancer and also add the deployment strategy.
    7. The above step will create a code deploy the application

- Step 6
    1. Create appspec.yml file and create a new task definition in the ECS
    2. Deploy the new version in the ECS using this code deploy application

- Step 7
    1. Create taskdef.json using the task definition and make the IMAGE_NAME changes

2. Add the taskdef.json to the code commit repo
3. Create the code pipeline.
    a. source will be code commit
    b. build will be code deploy
    c. Deploy will AWS ECS blue/green if you selected blue/green strategy in step 5 else select AWS ECS
4. Test the code pipeline, by committing code changes in the code commit repo

Project Takeaways

1. MLOps Architecture overview
2. Understanding various services provided by AWS
3. How to create s3 Bucket
4. How to create a commit repository and commit code in it?
5. How to create an ECR repository?
6. Converting ML application to Flask application
7. How to deploy the application using the Gunicorn web server?
8. Building Docker image
9. How to create ECS Cluster?
10. Testing the docker container in ECS
11. How to create a load balancer in the EC2 service?
12. How to create and build a project in Code Build?
13. Understanding ECS Cluster Task Definition
14. How to create Code Pipeline?