Module 4: Mini Project-1 Part-B

Bike Rental Prediction

Continuous Deployment

For this project, we will build a GitHub Actions workflow to automate the steps: model training, testing, package building, api dockerizing, docker image pushing, and deployment of api to Amazon EC2 for the bike rental count prediction model. Please refer to Module 4 - AST 3 for this mini-project.

Part B [Mini-project Session - 3 August 2024]

Step 1: Ensure to go through the previous mini-project [Bike Rental Prediction - Continuous Integration & Continuous Delivery]

Step 2: Download and understand project folder in your local system:

2.1 Download the given project folder '*bikeshare_project*' on to your system
2.2 Open it in VS Code, and understand the project structure and code organization

Step 3: Create an EC2 instance (1 point)

3.1 Navigate to EC2 dashboard on your AWS Management Console
3.2 Launch a new EC2 instance
3.3 Choose an instance type that is eligible for the free-tier
3.4 Connect to the instance using EC2 instance connect

Step 4: Add port to the security group associated with the instance (1 point)

4.1 Access the security group in the Security settings of the instance
4.2 Add a new inbound rule to allow incoming traffic on the port where the API will be hosted

Step 5: On your GitHub account, create a new repository, & add DockerHub credentials within its secrets:

5.1 Create a new repository to store files related to this mini-project
5.2 Get the access token from your DockerHub account settings
5.3 Add the docker username and access token to the Secrets of this repository

Step 6: Clone the remote GitHub repository in your system & add project files:

6.1 Clone the remote repository in your system
6.2 Add files from the downloaded "*bikeshare_project*" folder to this cloned repository
6.3 Finally, push the changes into the remote GitHub repository

Step 7: Update the virtual machine (EC2 instance) & configure it as a self-hosted GitHub Runner (3 points)

7.1 Update different dependencies of the virtual machine

7.2 Install docker on the virtual machine

7.3 Go to the settings on the GitHub repository and create a self-hosted runner

7.4 Configure this virtual machine/EC2 instance as a self-hosted runner

Step 8: Create a GitHub Actions workflow to define continuous deployment on EC2 instance and confirm that the container is running: (3 points)

8.1 Create a GitHub Actions workflow to automate the deployment of the API over EC2 instance
8.2 Include jobs for *train, test, build, push-image,* and *deploy*
8.3 Within the *deploy* job workflow, make sure to include steps to delete the old API docker container before the run of new one
8.4 Execute the *deploy* job with self-hosted runner

Step 9: Access the application using public IP address of the EC2 Instance (2 points)

9.1 Visit the public IP at the exposed port in your browser
9.2 Make a prediction over the API
9.3 Make some changes to the application UI on GitHub and see if the changes are reflecting on the public IP after the completion of ci/cd workflow

Step 10: Cleanup

10.1 Once done, terminate the EC2 instance, & delete the security group to prevent unnecessary charges, or consumption of free-tier resources