****

**MLOps |Spring 2025**

**Assignment 1**

**Title: Jenkins & Docker**

**Team Members:**

**Saaram Islam Cheema, 21i-1226**

**Umair Ahmed, 21i-1182**

**Submitted To:**

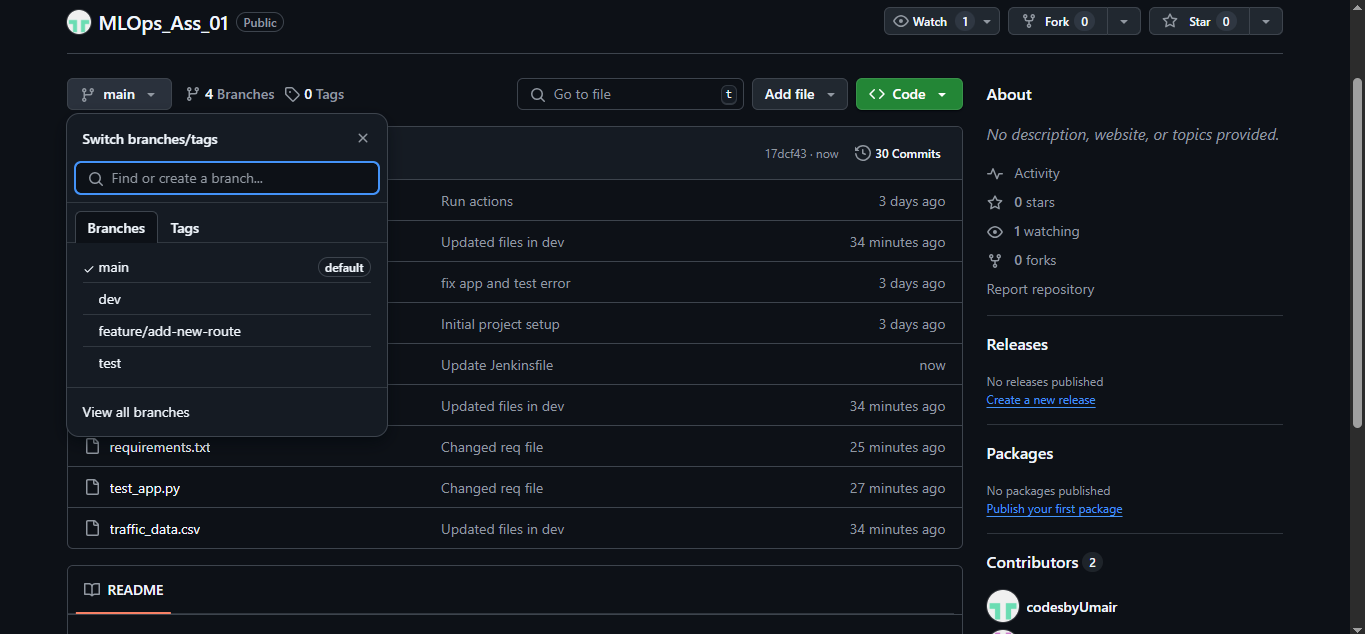
**Pir Samiullah Shah**

**Submission Date:**

**7 March 2025**

**Repository Structure and Branching Strategy**

A well-structured repository ensures maintainability and smooth collaboration among developers. The repository follows a branching strategy where the main branch is stable, while feature developments occur in separate branches. The dev branch is used for ongoing development, while the test branch is specifically for testing new features. Pull requests (PRs) are mandatory for merging changes, ensuring that every modification undergoes a review process.



**Code Quality Enforcement using Flake8**

Flake8 is employed to maintain Python code quality and enforce style guidelines. A GitHub Actions workflow is triggered on every pull request to the dev branch. The workflow sets up a Python environment, installs dependencies, and runs Flake8 to check for syntax errors and violations of the PEP 8 standard. If the code does not meet the required standards, the workflow fails, preventing poor-quality code from being merged.

**Unit Testing Workflow Implementation**

Automated unit testing is crucial to verify the correctness of the code. A GitHub Actions workflow is triggered on pull requests to the test branch. It ensures a clean environment by setting up Python, installing dependencies, and running tests using pytest. Any failure in unit tests prevents the pull request from being merged, ensuring that only functional and well-tested code is integrated into the main branch.

**Functional Jenkins & Docker Integration for Deployment**

Jenkins is integrated with Docker to automate the deployment process. The pipeline consists of stages such as checking out code, building a Docker image, and pushing it to Docker Hub. The process ensures that every successful build is stored in the Docker Hub repository with a unique version tag and a ‘latest’ tag. Jenkins notifies the admin via email regarding deployment success or failure. This workflow ensures a streamlined CI/CD process with minimal manual intervention.

**Proper Admin Notification Setup**

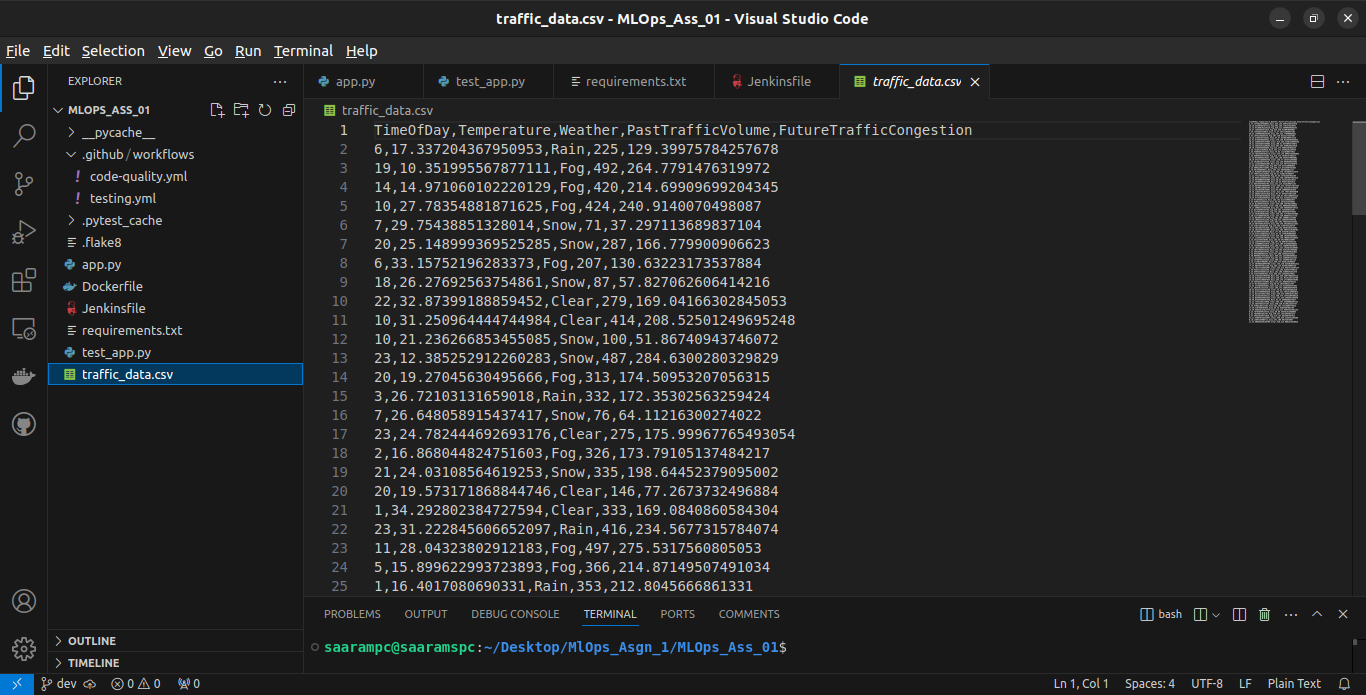
Email notifications are configured in Jenkins to alert administrators about deployment status. Upon successful deployment, an email containing the job details, build number, and Docker image tag is sent. In case of failure, the email contains a link to the Jenkins job logs for debugging. This proactive notification system ensures administrators are promptly informed of any issues or successful deployments.

**Dataset Description**

The dataset used for training the traffic congestion prediction model consists of synthetic data with random values for several key attributes:

* **Time of Day**: Represents the hour (0-23) when the data is recorded.
* **Temperature**: A numerical value between 10°C and 35°C.
* **Weather Condition**: Categorized as 'Clear', 'Rain', 'Fog', or 'Snow', encoded using one-hot encoding.
* **Past Traffic Volume**: A numerical value representing the historical traffic count.
* **Future Traffic Congestion**: The target variable predicted by the model.

Feature scaling is applied to numerical variables to balance the range of values. The dataset is then split into training and test sets, with 80% used for training and 20% for testing. A Linear Regression model is trained to predict future congestion based on past data, and its performance is evaluated using Mean Squared Error (MSE) and R² Score.

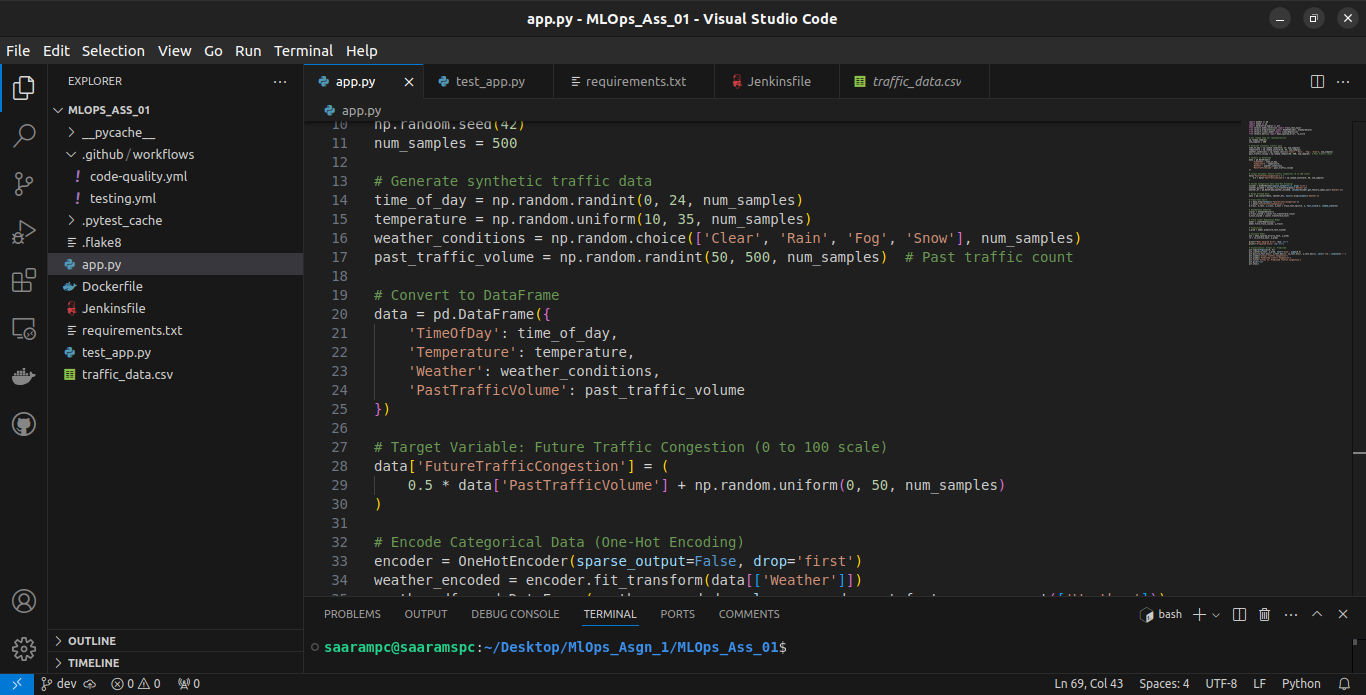


**Flask Application Description**

A Flask-based web application is implemented to serve the trained model as an API. The application allows users to send traffic data inputs, and the model returns congestion predictions. The Flask app follows a structured approach:

1. Loads the trained model and scaler upon startup.
2. Accepts HTTP POST requests with traffic-related data.
3. Preprocesses the input data, applying the same encoding and scaling techniques used during training.
4. Uses the trained Linear Regression model to generate predictions.
5. Returns the predicted traffic congestion as JSON.

The application ensures efficient deployment and real-time predictions, making it useful for traffic monitoring and planning systems.

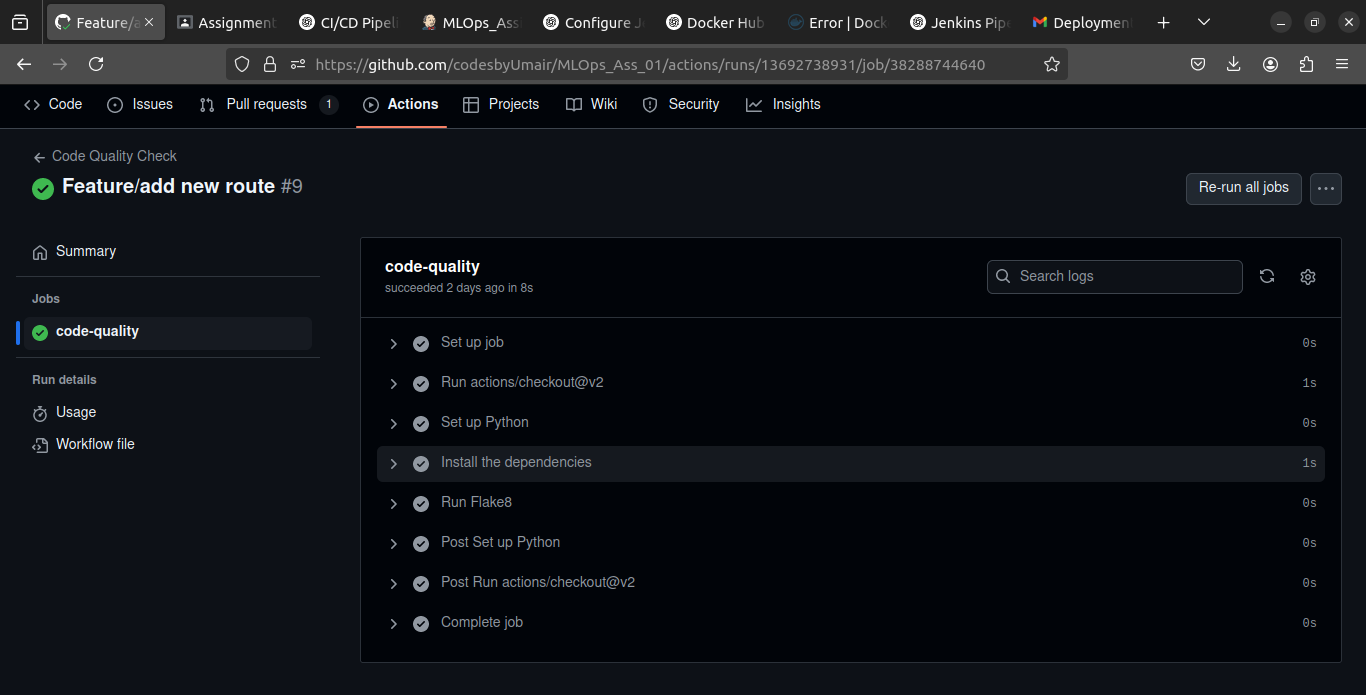
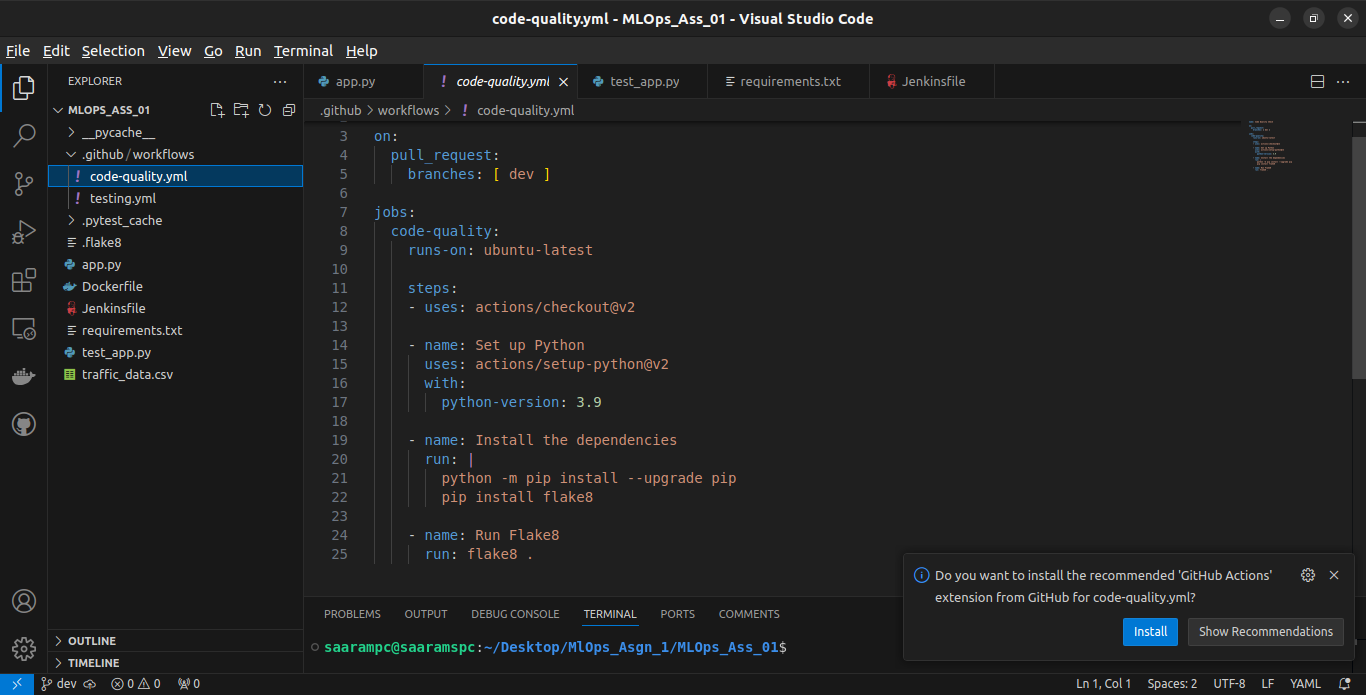


**Code-Quality Workflow Explanation**

The code-quality workflow ensures that every pull request to the dev branch undergoes a Flake8 check. The workflow consists of:

1. Checking out the repository.
2. Setting up Python 3.9.
3. Installing dependencies, including Flake8.
4. Running Flake8 on the entire codebase.
5. Failing the workflow if any linting errors are found.

By enforcing this workflow, the project maintains high coding standards, ensuring consistent readability and maintainability.

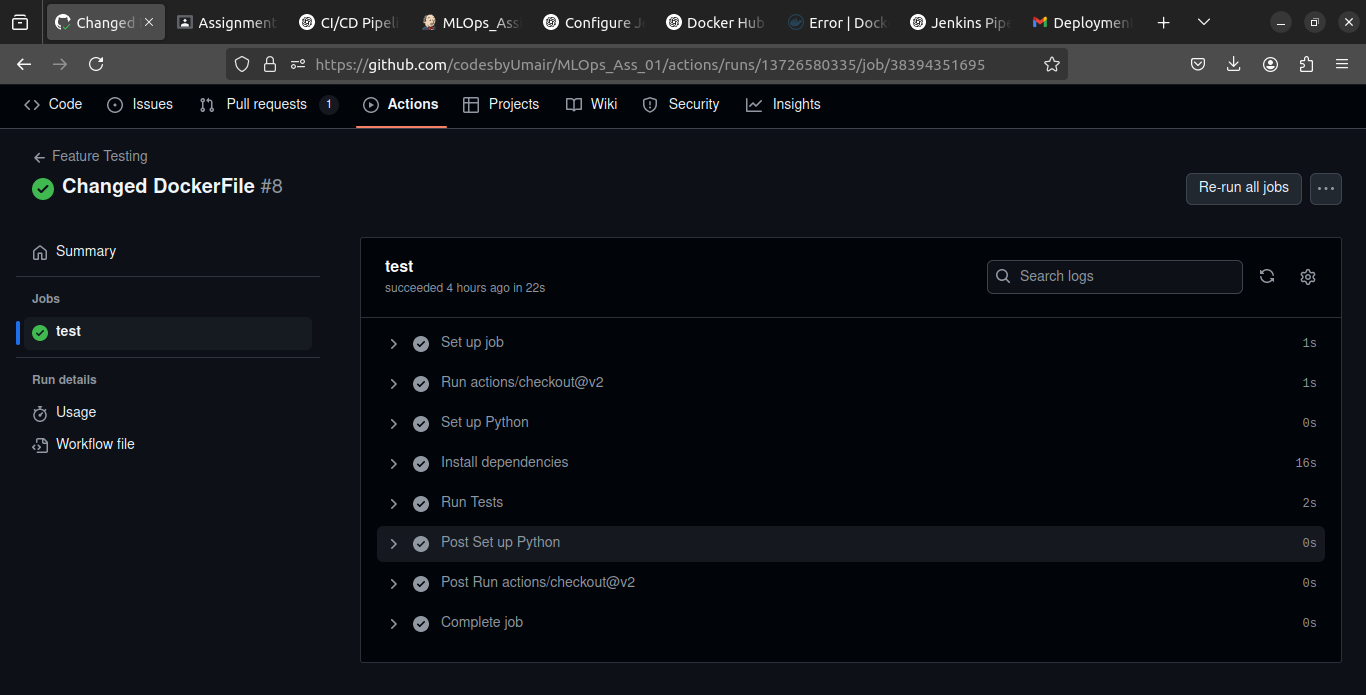
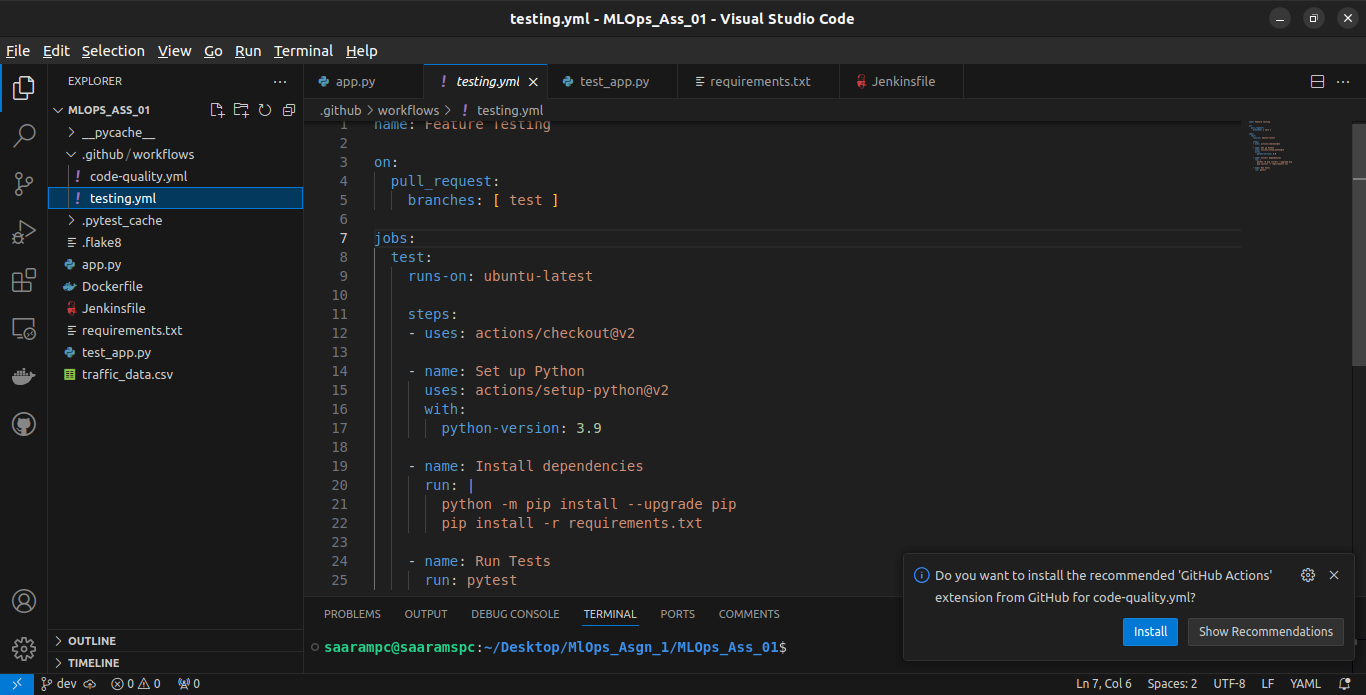


**Feature Testing Workflow Explanation**

Feature testing is performed via a GitHub Actions workflow on the test branch. The workflow includes:

1. Checking out the repository.
2. Setting up Python 3.9.
3. Installing dependencies from requirements.txt.
4. Running pytest to execute unit tests.

This process ensures that new features are thoroughly tested before being merged into the main codebase, reducing the risk of bugs and regressions.

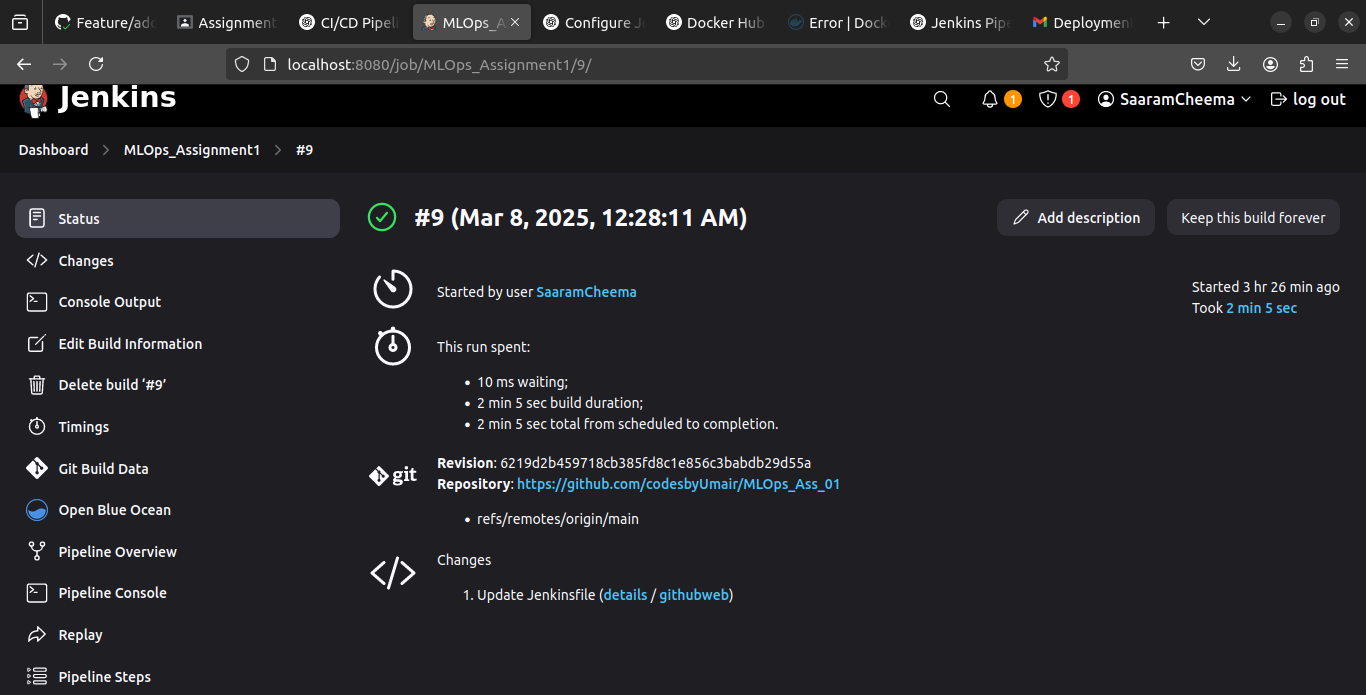


**Successful Jenkins Run**

The Jenkins pipeline automates the build and deployment process using Docker. The pipeline includes the following stages:

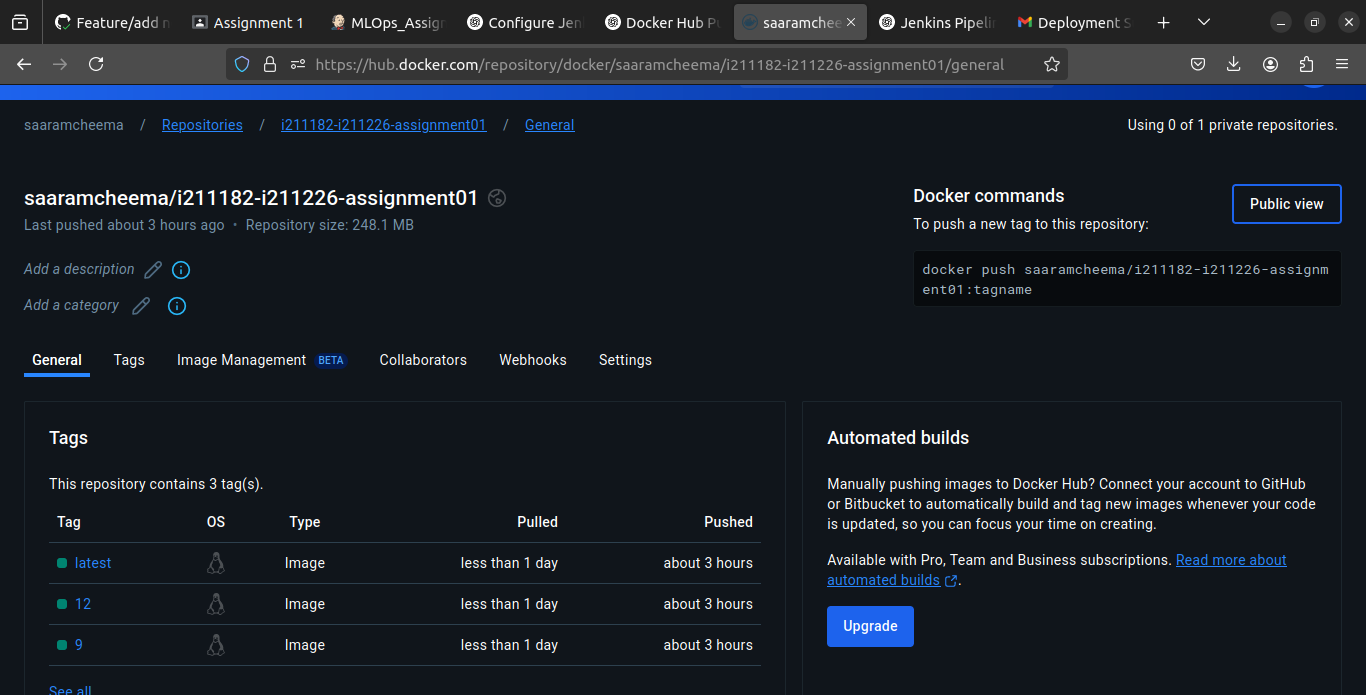
1. **Checkout**: Fetches the latest code from the repository.
2. **Build Docker Image**: Creates a Docker image using the source code and assigns it a unique tag based on the build number.
3. **Push to Docker Hub**: Authenticates with Docker Hub and pushes the newly built image.

If the pipeline succeeds, Jenkins sends a notification email to the admin, confirming the successful deployment.



**Successful Push of Docker Image to Docker Hub**

Upon successful execution of the Jenkins pipeline, the Docker image is pushed to the Docker Hub repository. The image is tagged with both the unique build number and the 'latest' tag, ensuring that the most recent version is always accessible for deployment. This automated process eliminates manual effort and ensures consistency across environments.



**Successful Mail to Admin**

As part of the Jenkins post-build actions, an email is sent to the administrator. The email includes:

* The job name and build number.
* The Docker image details.
* A link to the Jenkins job logs (in case of failure).

This notification system provides real-time updates, allowing the admin to take immediate action if required. The streamlined communication ensures that deployment activities are monitored effectively.

