**MLOps Group 24**

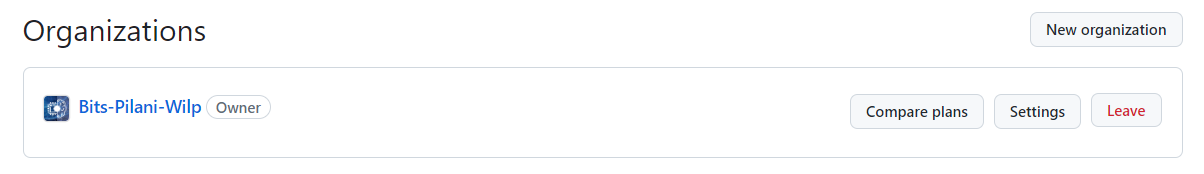
**Group Members**

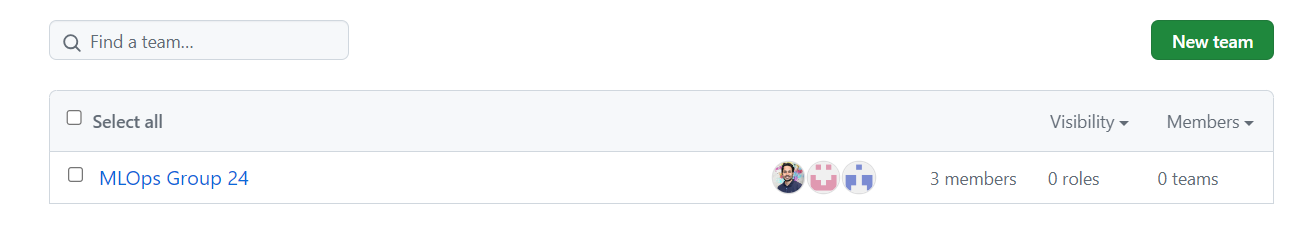
* Rajneesh Jha (2022ac05706)
* Sri Jayant (2022ac05045)
* Abhilesh Subhashrao Jawanjal (2022ac05604)
* Jeny Jasmine (2022ac05467)
* Mulla Akib Javed (2022ac05631)

**M1: MLOps Foundations**

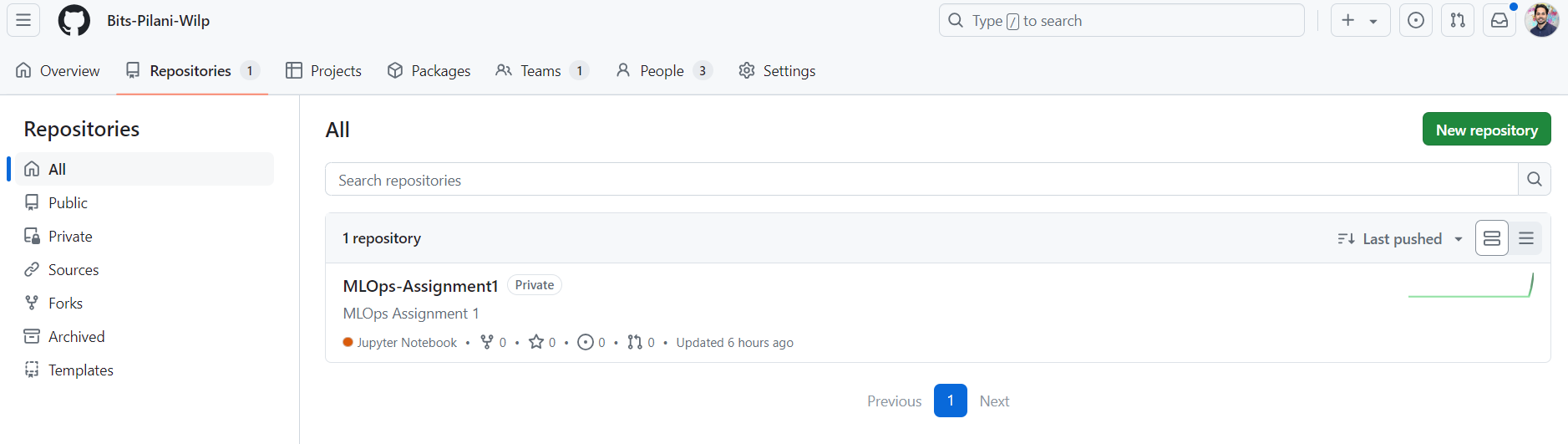
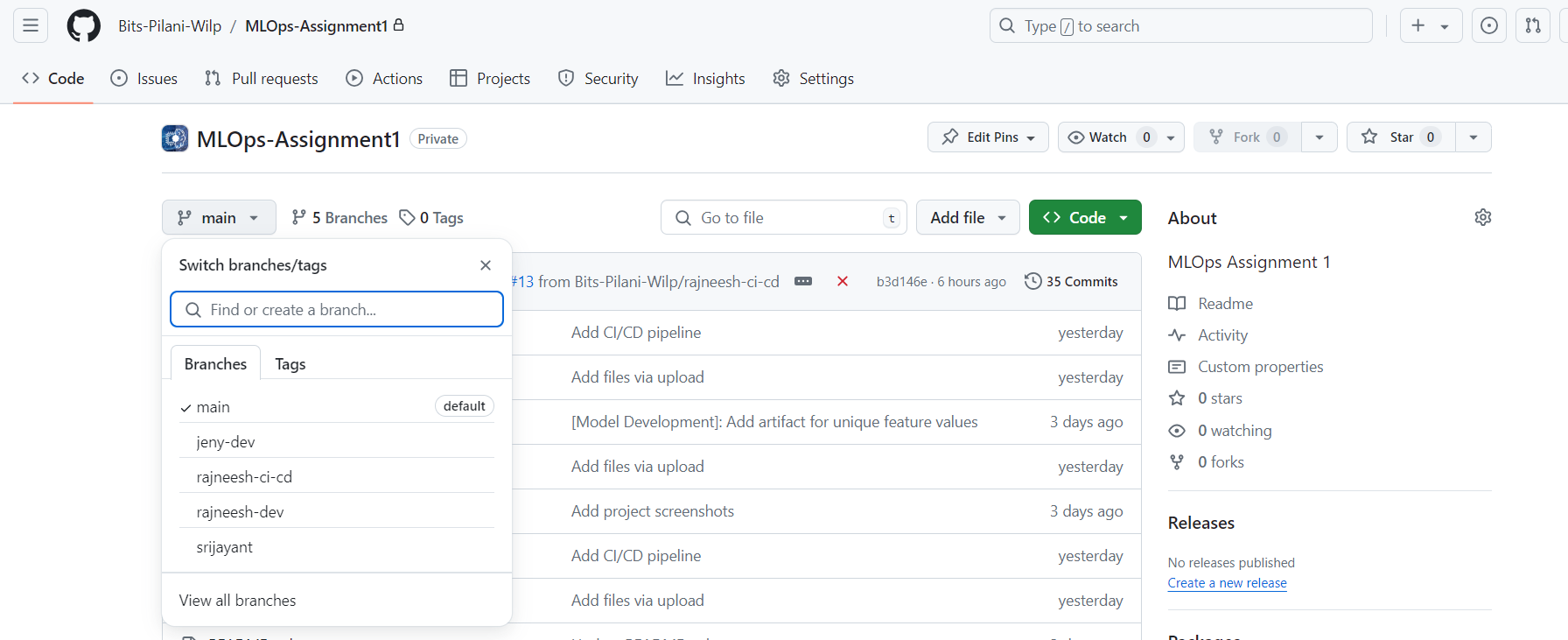
**1. GitHub Setup:**

* **Account and Organization Creation:** We initiated our project by creating a GitHub account and establishing an organization. This organization includes our entire team, fostering collaboration and centralized project management.



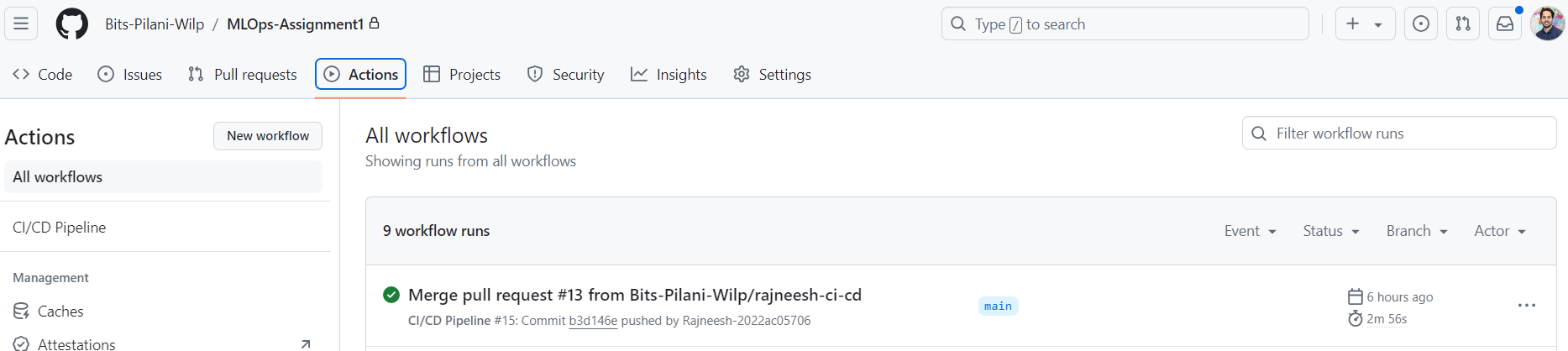


* **Repository Management:** The project repository was set up within the organization, with multiple branches to facilitate parallel development and seamless collaboration.



**2. CI/CD Pipeline Configuration:**

* **Pipeline Overview:** We implemented a CI/CD pipeline using GitHub Actions, defined in a YAML file within our repository. This pipeline is triggered automatically whenever code is pushed to the repository or a pull request is made.

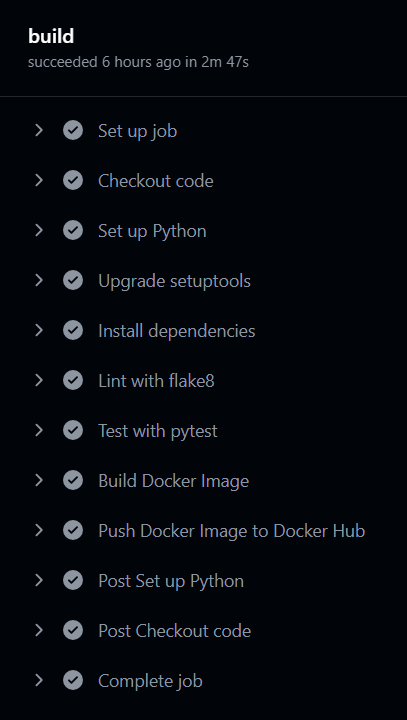


Repository Link :- [Link](https://github.com/Bits-Pilani-Wilp/MLOps-Assignment1/tree/main/.github/workflows)

Merge History :- [Link](https://github.com/Bits-Pilani-Wilp/MLOps-Assignment1/actions)

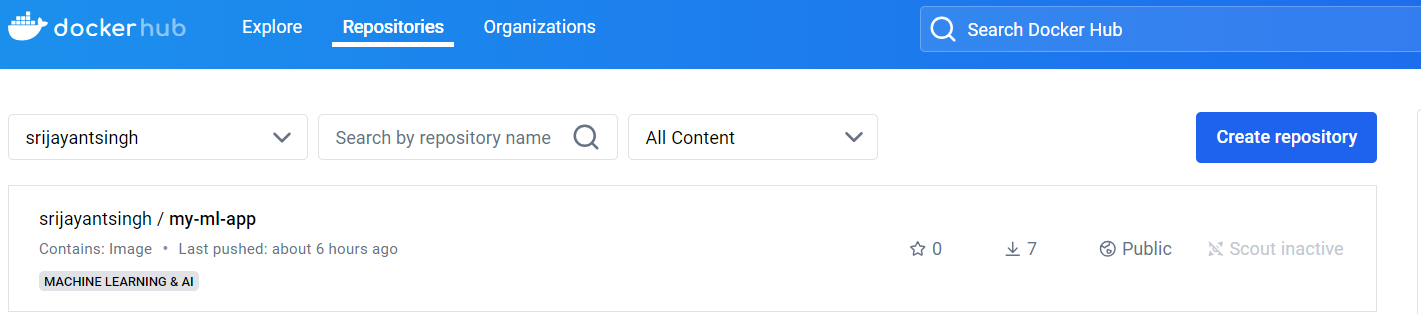
**3. Detailed Pipeline Breakdown:**

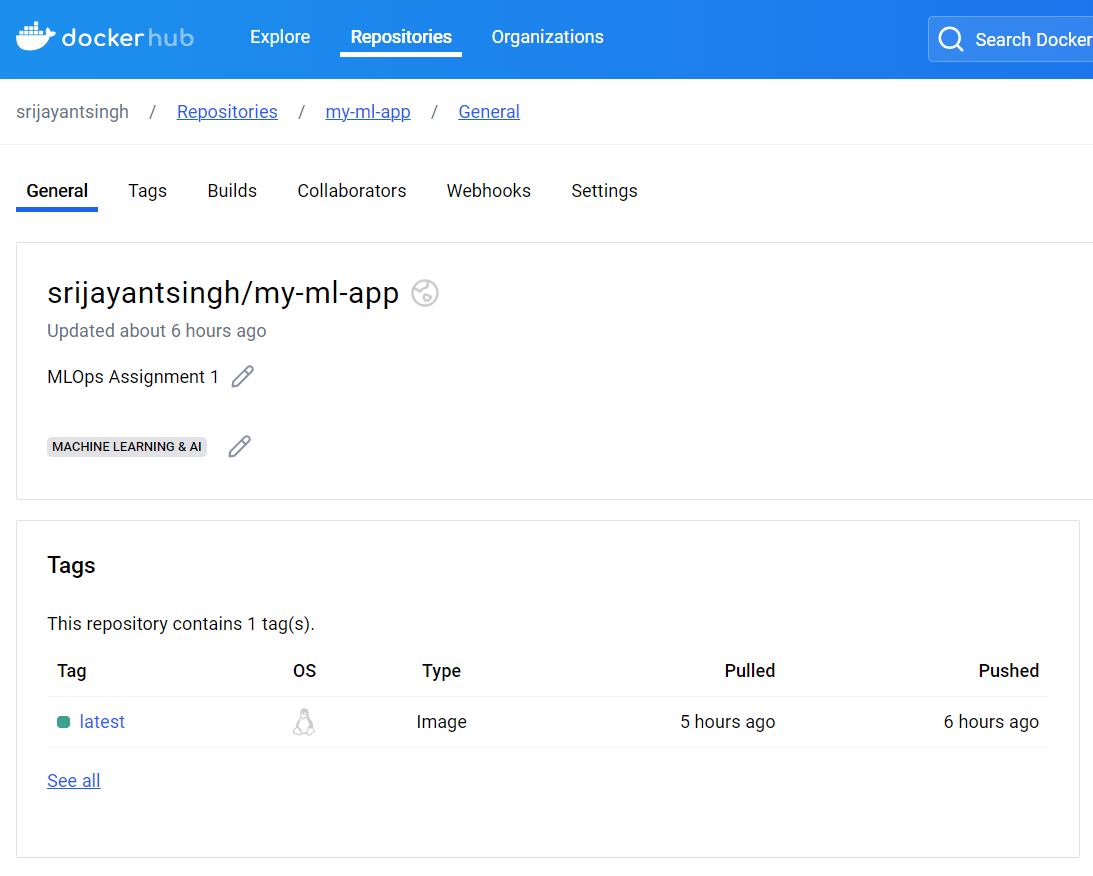
* **Triggering Events:** The pipeline is set to run on both push and pull\_request events, ensuring that our code is continuously tested and built.
* **Job Execution Environment:** The pipeline runs on a virtual machine provided by GitHub Actions, specifically using the latest version of Ubuntu.
* **Pipeline Steps:**
  + **Code Checkout:** The repository’s code is cloned into the virtual machine.
  + **Python Setup:** Python 3.11.5 is installed on the virtual machine.
  + **Dependency Management:** The setuptools package is upgraded, and a virtual environment (venv) is created to install the project dependencies from the requirements.txt file.
  + **Code Linting:** The code in the src and tests directories is linted using flake8 to ensure it adheres to coding standards and is free of basic errors.
  + **Unit Testing:** The pytest framework runs unit tests to verify the functionality of the code.
  + **Docker Image Build:** A Docker image, my-ml-app:latest, is built using the Dockerfile in the repository.
  + **Docker Hub Deployment:** The Docker image is tagged and pushed to Docker Hub, using credentials securely stored as GitHub secrets.



**4. Docker Image Management:**

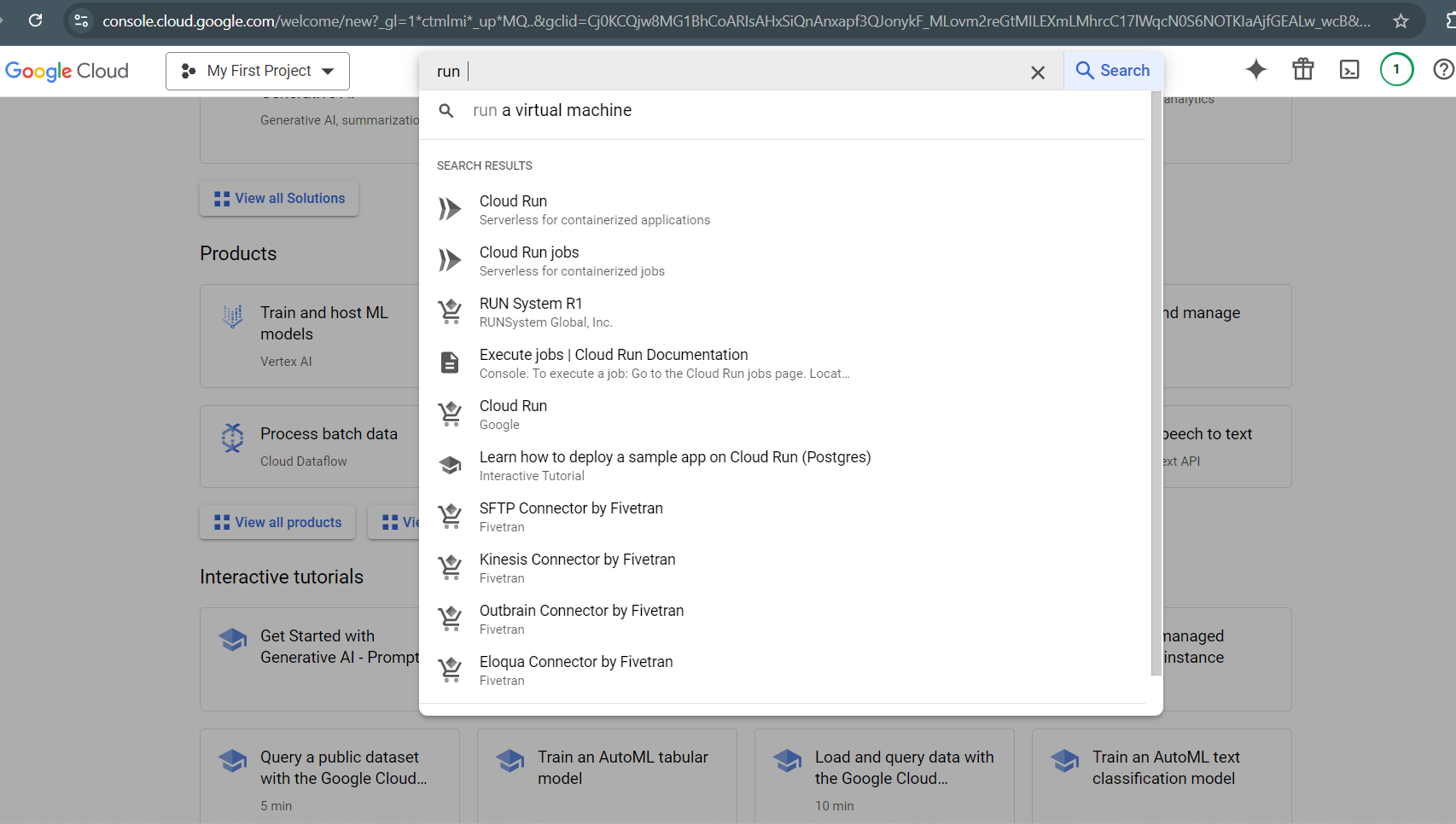
* The Docker image built through the CI/CD pipeline is pushed to Docker Hub, making it accessible for deployment across various platforms.

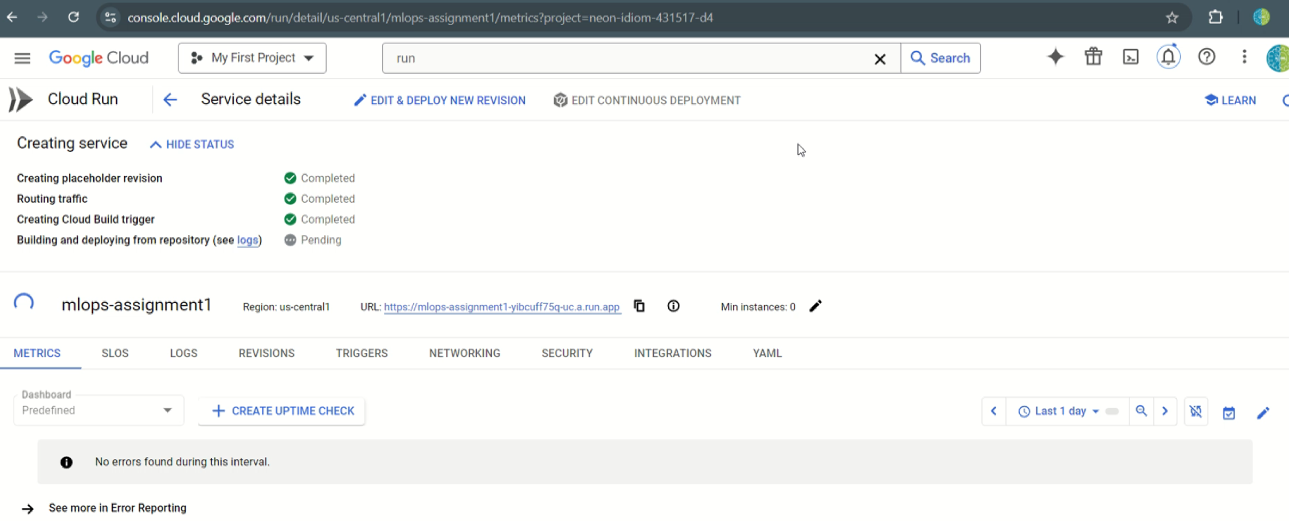


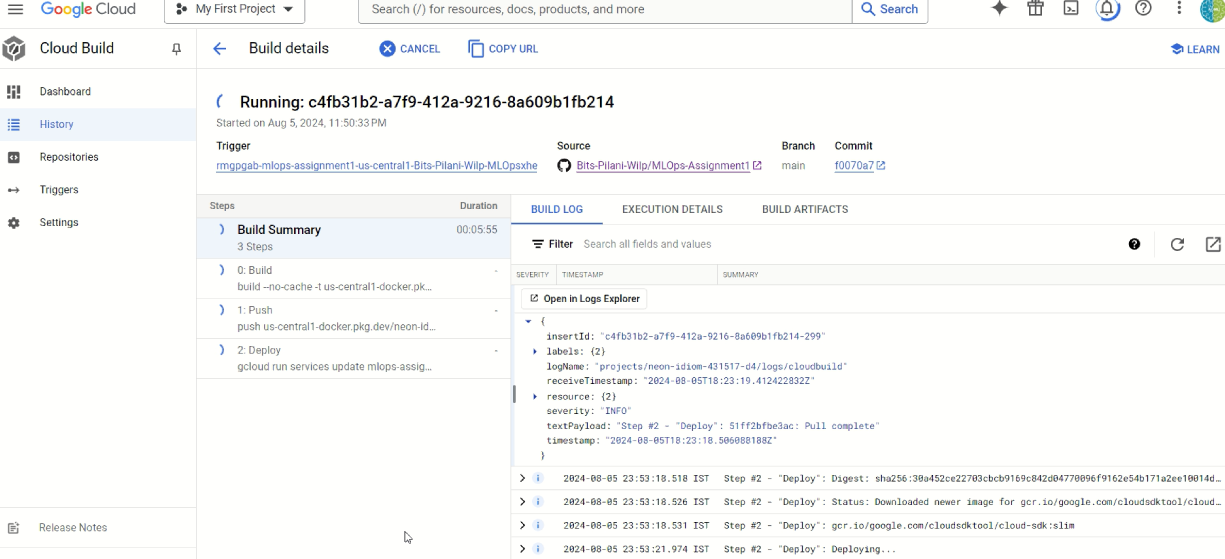


**5. Google Cloud Platform (GCP) Deployment:**

* **GCP Setup:** Before deploying the Docker image to GCP, necessary services were activated, and permissions were configured on GCP.







* **Cluster Creation:** A Kubernetes cluster was created on GCP to manage the deployment.
* **Image Deployment:** The Docker image from Docker Hub was pulled and deployed to GCP. Alternatively, the image could be deployed to GCP via Cloud Run, either by deploying a revision from the existing container image or by building a CI/CD pipeline directly within GCP.

