Rhombix Technologies – DevOps Internship

Candidate: Ali Shehzad

Task 2 – Project 1 & Project 2

Project 1: Continuous Integration using Jenkins

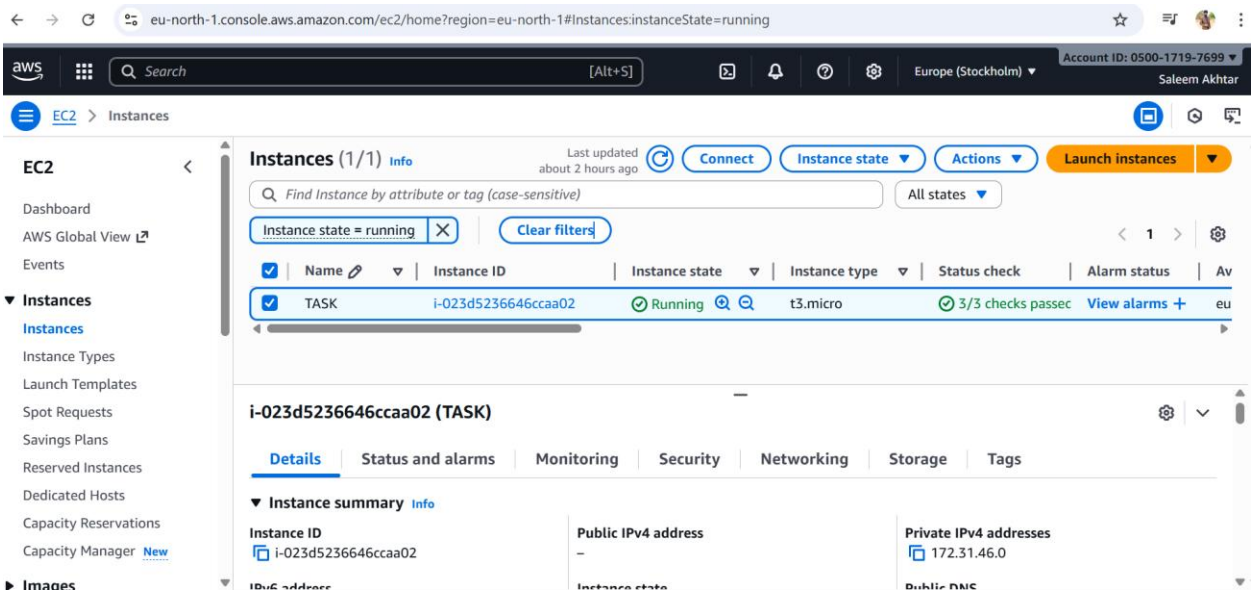
1.Objective:

In this task, you will learn the basics of Continuous Integration (CI) using Jenkins. You will set up Jenkins locally or on a cloud instance, create a Jenkins pipeline, and connect it with your GitHub repository. The goal is to automate code building and testing whenever developers push new changes. This task will teach you how CI helps teams detect issues early and improves collaboration between developers and operations

2.Process Overview:

EC2 Instance Setup:

- Launched an AWS EC2 instance to host Jenkins.
- Configured key-pair and connected to the instance using a terminal client.

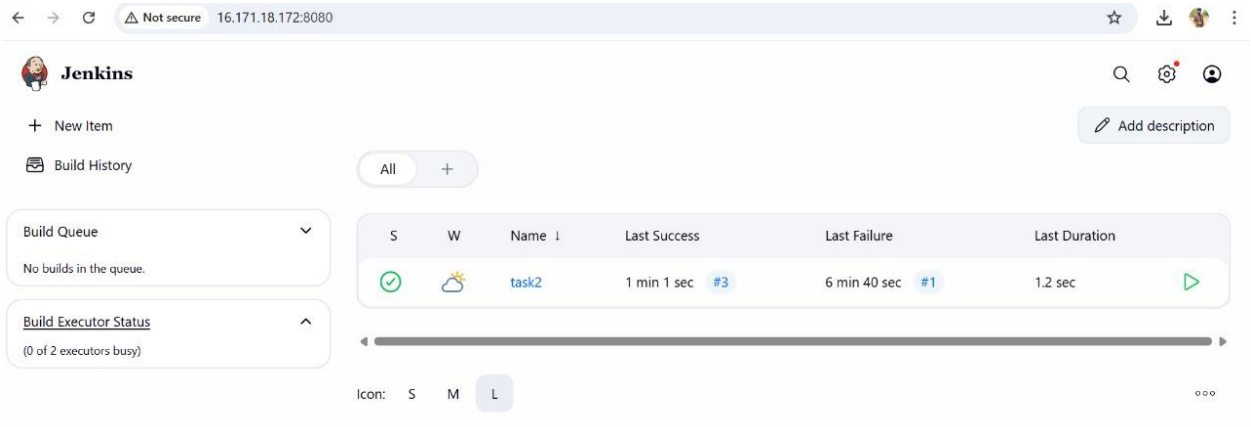


Jenkins Installation and Configuration:

- Installed Jenkins on the EC2 instance.
- Verified the service was running and accessed Jenkins dashboard via browser.
- Configured node monitoring to ensure disk space and resources did not interrupt builds.

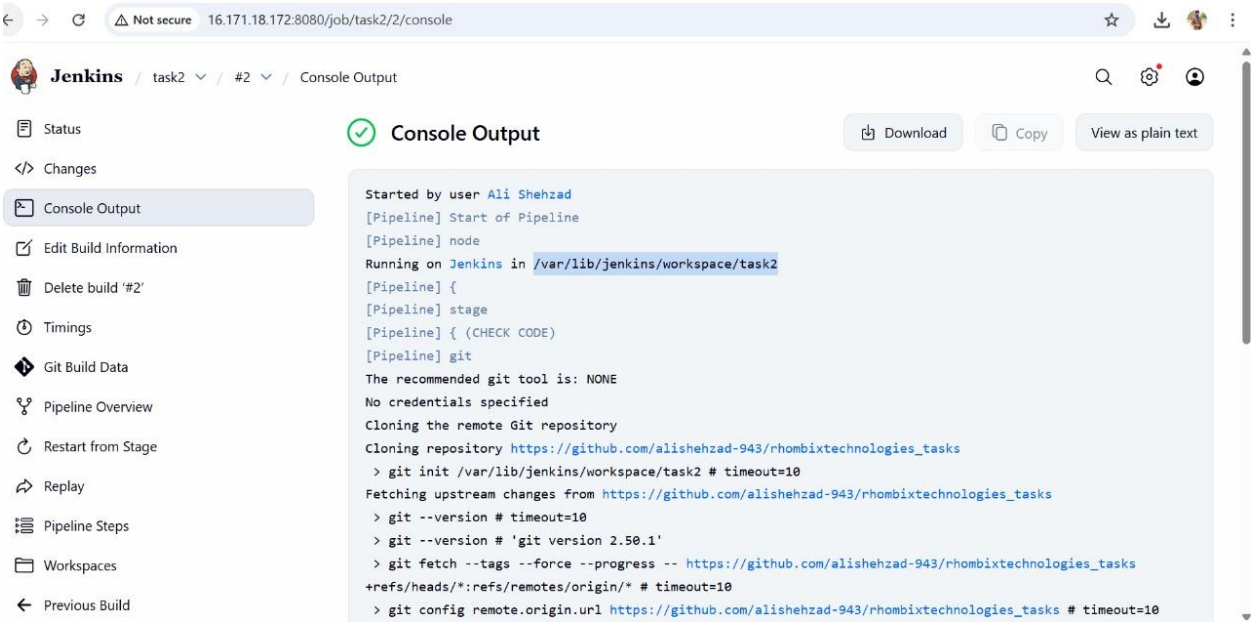
GitHub Integration:

- Connected Jenkins with the GitHub repository containing the sample project.
- Created a simple Jenkins pipeline to fetch the latest code automatically.



Webhook Configuration:

- Configured a GitHub webhook to notify Jenkins of push events.
- Verified that any code changes in GitHub automatically triggered the pipeline without manual intervention.



3.Outcome:

- Jenkins pipeline successfully integrated with GitHub.
- Automated builds were triggered on every code push.
- Demonstrated a working CI workflow with minimal manual effort, improving efficiency and collaboration.

Project 2: Containerization using Docker

1.Objective:

In this task, you will learn how to use Docker to package applications into containers. You will practice creating a Dockerfile, building Docker images, and running containers. You will also work on managing containers, pulling images from Docker Hub, and exposing containerized apps. This task will help you understand how containerization improves scalability, portability, and deployment speed in modern DevOps workflows

2.Process Overview:

Docker Installation:

- Installed Docker on the EC2 instance and verified the installation to ensure containerization capabilities were available.

Web Application Setup:

- Created a social login web page (index.html) sourced from the internet.
- Prepared the application folder for Docker packaging.

Dockerfile Creation:

- Developed a Dockerfile to build a container using Nginx as a lightweight web server.
- Ensured the Dockerfile included copying the web page and exposing the proper port.

Build Docker Image:

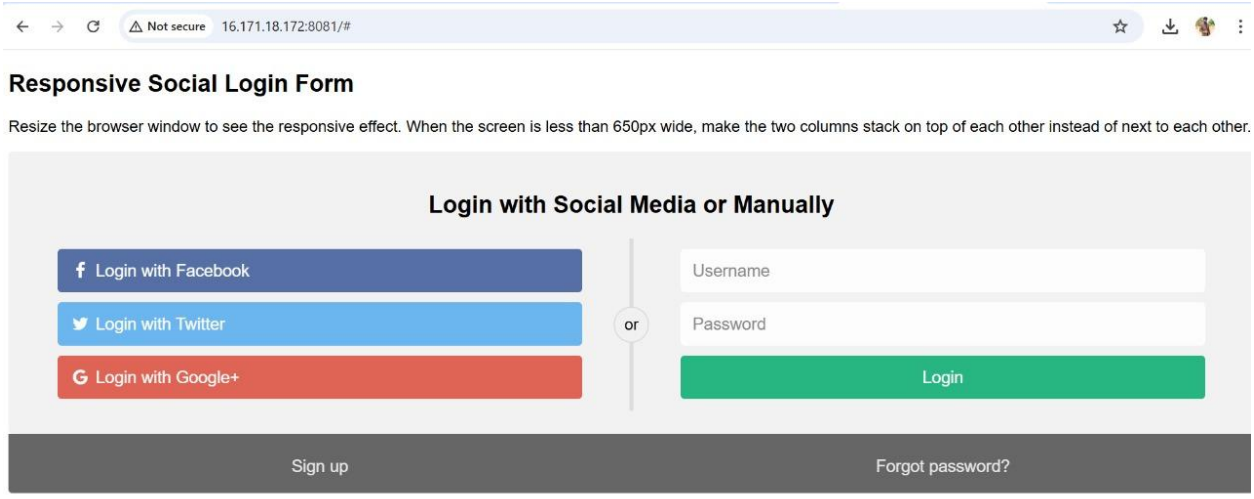
- Built a Docker image named social-login-app.

Run Docker Container:

- Deployed a container named cont1 mapped to host port 8081.
- Verified the container was running correctly using Docker management commands.

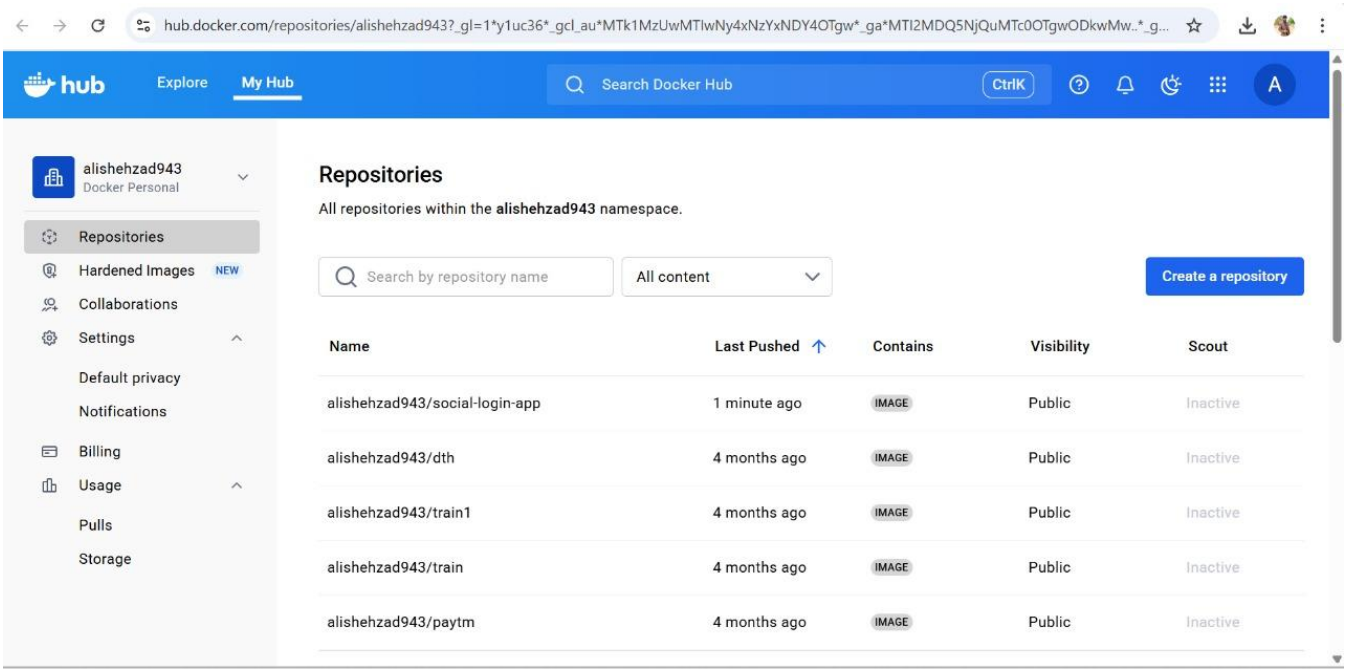
Browser Verification:

- Accessed the application in the browser via EC2 public IP and port 8081.
- Confirmed the social login page was displayed correctly, demonstrating successful containerization.



Push to Docker Hub:

- Logged into Docker Hub using username (alishehzad943) and personal access token.
- Tagged and pushed the Docker image to Docker Hub for sharing and deployment.



3.Outcome:

- Docker image built successfully, and container ran on port 8081.
- Application accessible via browser, confirming container functionality.
- Docker Hub image available for deployment or sharing.
- Demonstrated complete containerization workflow from local setup to cloud registry.

Task 2 -project1-history: Link

Task2-project2-history: Link

Complete Task2 Projects Vedio: Link