

# Rhombix Technologies – DevOps Internship

**Candidate: Ali Shehzad**

## **Task 2 – Project 1 & Project 2**

### **Project 1: Continuous Integration using Jenkins**

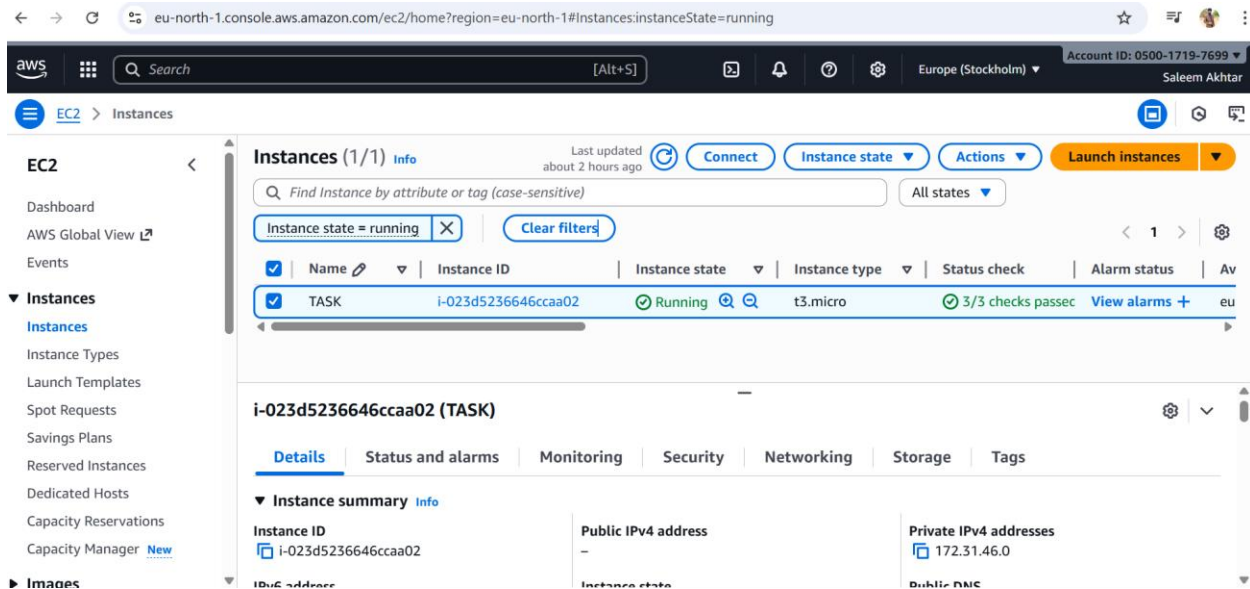
#### **1.Objective:**

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.

The screenshot shows the Jenkins dashboard at IP 16.171.18.172:8080. The 'Build Queue' section indicates 'No builds in the queue.' The 'Build Executor Status' section shows '(0 of 2 executors busy)'. A table displays the build history for 'task2'.

S	W	Name	Last Success	Last Failure	Last Duration
✓	☁	task2	1 min 1 sec #3	6 min 40 sec #1	1.2 sec

## 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.

The screenshot shows the Jenkins console output for 'task2 #2'. The output details the pipeline execution, including cloning the repository and fetching upstream changes.

```
Started by user Ali Shehzad
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/task2
[Pipeline] {
[Pipeline] stage
[Pipeline] { (CHECK CODE)
[Pipeline] git
The recommended git tool is: NONE
No credentials specified
Cloning the remote Git repository
Cloning repository https://github.com/alishehzad-943/rhombixtechnologies_tasks
> git init /var/lib/jenkins/workspace/task2 # timeout=10
Fetching upstream changes from https://github.com/alishehzad-943/rhombixtechnologies_tasks
> git --version # timeout=10
> git --version # 'git version 2.50.1'
> git fetch --tags --force --progress -- https://github.com/alishehzad-943/rhombixtechnologies_tasks
+refs/heads/*:refs/remotes/origin/* # timeout=10
> git config remote.origin.url https://github.com/alishehzad-943/rhombixtechnologies_tasks # timeout=10
```

### **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.

← → ↻ ⚠ Not secure 16.171.18.172:8081/# ☆ ⬇ 👤 ⋮

### Login with Social Media or Manually

Login with Facebook

Login with Twitter

Login with Google+

or

Login

[Sign up](#) [Forgot password?](#)

## Push to Docker Hub:

- Logged into Docker Hub using username (alishahzad943) and personal access token.
- Tagged and pushed the Docker image to Docker Hub for sharing and deployment.

← → ↻ 🌐 hub.docker.com/repositories/alishahzad943?\_gl=1\*y1uc36\*\_gcl\_au\*MTk1MzUwMTIwNy4xNzYxNDY4OTgw\*\_ga\*MTI2MDQ5NjQuMTc0OTgwODkwMw...\_g... ☆ ⬇ 👤 ⋮

hub [Explore](#) [My Hub](#) 🔍 Search Docker Hub CtrlK ? 🔔 🔄 ⋮ A

alishahzad943  
Docker Personal

Repositories

Hardened Images **NEW**

Collaborations

Settings

Default privacy

Notifications

Billing

Usage

Pulls

Storage

### Repositories

All repositories within the alishahzad943 namespace.

Create a repository

Name	Last Pushed	Contains	Visibility	Scout
alishahzad943/social-login-app	1 minute ago	IMAGE	Public	Inactive
alishahzad943/dth	4 months ago	IMAGE	Public	Inactive
alishahzad943/train1	4 months ago	IMAGE	Public	Inactive
alishahzad943/train	4 months ago	IMAGE	Public	Inactive
alishahzad943/paytm	4 months ago	IMAGE	Public	Inactive

### **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](#)