Lab Exercise 9- Continuous Integration & Delivery (CI/CD) for APIs Using GitHub Actions + JUnit + Docker Hub

1. Objective

In this lab you will:

1. Write and run automated JUnit tests for your API.
2. Build your Spring Boot application using Maven.
3. Build a Docker image of the API.
4. Push the Docker image to Docker Hub automatically through GitHub Actions.
5. Understand CI/CD workflow and artifacts.

2. Prerequisites

You must have:

- Your Spring Boot API (from Lab Exercise 4)
- JUnit tests placed under:

  src/test/java/com/example/simpleapi/
- A Docker Hub account
- GitHub repository containing your Spring Boot project
- Dockerfile in the project root

3. Add Dockerfile to Your Project

Create a file named:

```
Dockerfile
```

At project root (same level as pom.xml).

Add:

```
FROM eclipse-temurin:17-jre-jammy

ARG JAR_FILE=target/*.jar

COPY ${JAR_FILE} app.jar

EXPOSE 8080

ENTRYPOINT ["java", "-jar", "/app.jar"]
```

4. Add Junit Tests to Your Project

Example test files (from earlier lab):

```
src/test/java/com/example/simpleapi/UserServiceTest.java

src/test/java/com/example/simpleapi/UserControllerTest.java
```

Junit tests will run automatically in the CI pipeline.

6. Create GitHub Secrets

Go to:

```
GitHub → Repository → Settings → Secrets → Actions → New secret
```

Add these:

Secret Name	Value
DOCKERHUB_USERNAME	your Docker Hub username
DOCKERHUB_TOKEN	Docker Hub access token or password

6. Create GitHub Actions Workflow

Create the folder:

```
.github/workflows/
```

Inside create the file:

```
ci-cd.yml
```

Paste the full working workflow below.

Full CI/CD Pipeline YAML

JUnit + Docker Build + Docker Push

```
name: CI/CD - JUnit + Docker Build & Push

on:
  push:
    branches: ["main"]
  pull_request:
    branches: ["main"]

jobs:
  build-test:
    name: Run JUnit Tests & Build Jar
    runs-on: ubuntu-latest

    steps:
      - name: Checkout Repository
        uses: actions/checkout@v4

      - name: Set up Java 17
        uses: actions/setup-java@v4
        with:
          java-version: "17"
          distribution: "temurin"

      - name: Cache Maven dependencies
        uses: actions/cache@v4
        with:
          path: ~/.m2
          key: ${{ runner.os }}-m2-${{ hashFiles('**/pom.xml') }}
          restore-keys: |
```

```
      ${{ runner.os }}-m2

  - name: Run JUnit Tests
    run: mvn -B test

  - name: Build JAR Package
    run: mvn -B package -DskipTests

  - name: Upload JAR Artifact
    uses: actions/upload-artifact@v4
    with:
      name: springboot-jar
      path: target/*.jar
  - uses: mr-smithers-excellent/docker-build-push@v6
    name: Build & push Docker image
    with:
      image: hkshitesh/sebapi
      registry: docker.io
      username: ${{ secrets.DOCKER_USERNAME }}
      password: ${{ secrets.DOCKER_PASSWORD }}
```

7. How the CI/CD Pipeline Works

Step 1 — Checkout Repository

GitHub fetches your latest code.

Step 2 — JUnit Tests (CI)

mvn test runs:

- UserServiceTest
- UserControllerTest

If any test fails → pipeline stops.

Step 3 — Build JAR

mvn package produces:

```
target/simpleapi-0.0.1-SNAPSHOT.jar
```

Step 4 — Build Docker Image

Using your Dockerfile:

```
docker build -t username/simpleapi .
```

Step 5 — Push Docker Image (CD)

Two versions are pushed:

1. :latest
2. SHA tag (unique):

username/simpleapi:ab3490fa

8. Verify Docker Image on Docker Hub

After a successful run:

Go to:

```
Docker Hub → Repositories → your repo
```

You should see:

```
latest      3 minutes ago

ab3490f     3 minutes ago
```