SET 1 - Q1: Git Repository Setup

Commands:
mkdir MyJavaProject
cd MyJavaProject
echo 'public class HelloWorld { public static void main(String[] args) { System.out.println("Hello, World!"); } }' > HelloWorld.java
git init
git add HelloWorld.java
git commit -m "Initial commit - HelloWorld.java"
git remote add origin https://github.com//.git
git branch -M main
git push -u origin main

Expected Output:
- HelloWorld.java visible on GitHub with commit message 'Initial commit - HelloWorld.java'.

SET 1 - Q2: Maven Project Creation and Build

Commands:
sudo apt install maven -y
mvn archetype:generate -DgroupId=com.example -DartifactId=MyMavenProject -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false
cd MyMavenProject
mvn clean package

Expected Output:
- Build SUCCESS.
- JAR file generated in target/MyMavenProject-1.0-SNAPSHOT.jar

SET 1 - Q3: Dockerfile Creation

Dockerfile:
FROM openjdk:17-jdk-slim
COPY target/MyMavenProject-1.0-SNAPSHOT.jar app.jar
ENTRYPOINT ["java", "-jar", "app.jar"]

Commands:
sudo docker build -t my-maven-app .
sudo docker images

Expected Output:
Docker image visible in 'docker images' list.

SET 2 - Q1: Jenkins Freestyle Job with Maven

Steps:
1. Create new Freestyle Project in Jenkins.
2. Source Code Management → Git → Add repo URL.
3. Build → Add build step → Invoke Maven → Goals: clean package.
4. Save → Build Now.

Expected Output:
Build SUCCESS in Jenkins console log.

SET 2 - Q2: Jenkins Webhook Trigger

Steps:
1. In Jenkins job: Enable 'GitHub hook trigger for GITScm polling'.
2. In GitHub repo: Settings → Webhooks → Add webhook.
- Payload URL: http://:8080/github-webhook/
- Content type: application/json
- Select 'Just the push event'.
Expected Output:
Auto build trigger on every git push.

SET 2 - Q3: Run Docker Container

Commands:
sudo docker run -d --name my-running-app my-maven-app
sudo docker ps
sudo docker logs my-running-app

Expected Output:
Container running successfully showing 'Hello, World!'.

SET 3 - Q1: Clone, Modify, and Push Changes

Commands:
git clone https://github.com//.git
cd
echo 'public class HelloWorld { public static void main(String[] args) { System.out.println("Updated Hello, World!"); } }' > HelloWorld.java
git add HelloWorld.java
git commit -m "Updated HelloWorld.java message"
git push

Expected Output:
Updated code visible on GitHub.

SET 3 - Q2: Jenkins Pipeline Setup

Pipeline Script:
pipeline {
agent any
stages {
stage('Checkout') { steps { git 'https://github.com//.git' } }
stage('Build') { steps { sh 'mvn clean package' } }
stage('Test') { steps { sh 'echo "Running tests..."' } }
}
}

Expected Output:
Pipeline stages execute successfully with 'BUILD SUCCESS'.

SET 3 - Q3: Build Docker Image for Java Project

Commands:
sudo docker build -t my-java-app .
sudo docker images

Expected Output:
Image visible in docker images list.