SET 1 – Independent Ubuntu Commands

Q1 – Create Git Repo, Add Java File, Commit & Push

```
# 1. Update system and install Git
sudo apt update
sudo apt install git -y

# 2. Create project folder and initialize Git
mkdir DevOpsProject
cd DevOpsProject
git init

# 3. Create HelloWorld.java
nano HelloWorld.java

# 4. Add, commit and push to GitHub
git add HelloWorld.java
git commit -m "Initial commit - HelloWorld.java"
git branch -M main
git remote add origin https://github.com/<username>/<repo-name>.git
git push -u origin main
```

Q2 – Create Maven Project & Build

```
# 1. Update system and install Java & Maven
sudo apt update
sudo apt install openjdk-17-jdk maven -y

# 2. Check versions
```

```
java -version
mvn -v

# 3. Create Maven project
mvn archetype:generate -DgroupId=com.example -DartifactId=myapp \
-DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false

# 4. Go to project folder
cd myapp

# 5. Build project
mvn clean package
```

Q3 – Create Dockerfile & Build Docker Image

```
# 1. Update system and install Docker
sudo apt update
sudo apt install docker.io -y
sudo systemctl start docker
sudo systemctl enable docker

# 2. Navigate to Maven project folder
cd ~/myapp

# 3. Create Dockerfile
nano Dockerfile
# Paste:
# FROM openjdk:17
```

# COPY target/myapp-1.0-SNAPSHOT.jar app.jar

# ENTRYPOINT ["java","-jar","app.jar"]

# 4. Build Docker image

sudo docker build -t myapp:latest .

# 5. Verify image

sudo docker images

SET 2 – Independent Ubuntu Commands

Q1 – Create Jenkins Freestyle Project with Maven Build

# 3. Install Git

sudo apt install git -y

# 4. Install Maven

sudo apt install maven -y

# 5. Install Jenkins

wget -q -O - https://pkg.jenkins.io/debian/jenkins.io.key | sudo apt-key add -

sudo sh -c 'echo deb http://pkg.jenkins.io/debian/ binary/ > /etc/apt/sources.list.d/jenkins.list'

sudo apt update

sudo apt install jenkins -y

sudo systemctl start jenkins

sudo systemctl enable jenkins

# 6. Open Jenkins in browser: http://localhost:8080

# 7. Create Freestyle Project → Configure

#   - Source Code Management: Git → URL: https://github.com/<username>/<repo-name>.git

#   - Build → Execute Shell: mvn clean package

# 8. Save and Build → Check console output

Q2 – Configure Jenkins to Trigger Build on GitHub (Webhook)

```
# 2. Install Jenkins


# 3. Clone GitHub repo (optional, for test commit)
git clone https://github.com/<username>/<repo-name>.git
cd <repo-name>


# 4. Make a change
echo "// Test webhook" >> HelloWorld.java


# 5. Add, commit & push
git add HelloWorld.java
git commit -m "Testing GitHub webhook"
git push


# 6. In GitHub → Settings → Webhooks → Add webhook
#    Payload URL: http://<jenkins-server>:8080/github-webhook/
#    Content type: application/json
# 7. In Jenkins Freestyle job → Build Triggers → GitHub hook trigger for GITScm polling
# 8. Save job → Push to GitHub → Job triggers automatically
```

Q3 – Run Docker Container and Verify Logs

# 1. Update system and install Docker

sudo apt update

sudo apt install docker.io -y

sudo systemctl start docker

sudo systemctl enable docker

# 2. Create sample Maven Java project (if Docker image not already built)

mkdir DockerTest

cd DockerTest

# 3. Create HelloWorld.java

nano HelloWorld.java

# 4. Create Maven structure

mkdir -p src/main/java

mv HelloWorld.java src/main/java/

nano pom.xml

# Paste minimal Maven configuration

# 5. Build jar

sudo apt install maven -y

mvn clean package

# 6. Create Dockerfile

```
nano Dockerfile

# Paste:

# FROM openjdk:17

# COPY target/myapp-1.0-SNAPSHOT.jar app.jar

# ENTRYPOINT ["java","-jar","app.jar"]


# 7. Build Docker image

sudo docker build -t myapp:latest .


# 8. Run container

sudo docker run -d --name mycontainer myapp:latest


# 9. Verify running container

sudo docker ps


# 10. Check container logs

sudo docker logs mycontainer
```

SET 3 – Independent Ubuntu Commands

Q1 – Clone GitHub Repo, Modify Java File, Commit Push

```
# 1. Update system & install Git

sudo apt update

sudo apt install git -y


# 2. Install Java (for compiling Java code if needed)
```

```
sudo apt install openjdk-17-jdk -y

java -version


# 3. Clone GitHub repository

git clone https://github.com/<username>/<repo-name>.git

cd <repo-name>


# 4. Modify Java file

nano HelloWorld.java

# e.g., change message in System.out.println


# 5. Add, commit and push changes

git add HelloWorld.java

git commit -m "Updated HelloWorld.java"

git push
```

Q2 – Create Jenkins Pipeline (Checkout → Build → Test)

```
# 1. Update system

sudo apt update


# 2. Install Git, Java, Maven

sudo apt install git -y

sudo apt install openjdk-17-jdk -y

sudo apt install maven -y


# 3. Install Jenkins

wget -q -O - https://pkg.jenkins.io/debian/jenkins.io.key | sudo apt-key add -
```

```
sudo sh -c 'echo deb http://pkg.jenkins.io/debian/ binary/ >
/etc/apt/sources.list.d/jenkins.list'

sudo apt update

sudo apt install jenkins -y

sudo systemctl start jenkins

sudo systemctl enable jenkins
```

# 4. Create Jenkins Pipeline Job → Add pipeline script:

# Jenkinsfile content:

```
pipeline {
  agent any
  stages {
    stage('Checkout') {
      steps {
        git 'https://github.com/<username>/<repo-name>.git'
      }
    }
    stage('Build') {
      steps {
        sh 'mvn clean package'
      }
    }
    stage('Test') {
      steps {
        sh 'mvn test'
      }
    }
```

```
   }
}
```

```
# 5. Save and Build → check console output for all stages
```

Q3 – Build Docker Image for Java Project and Verify

```
# 1. Update system & install Docker
sudo apt update
sudo apt install docker.io -y
sudo systemctl start docker
sudo systemctl enable docker

# 2. Create new project folder
mkdir DockerJavaProject
cd DockerJavaProject

# 3. Create HelloWorld.java
nano HelloWorld.java
# Paste:
# public class HelloWorld {
#    public static void main(String[] args) {
#       System.out.println("Hello Docker World!");
#    }
# }

# 4. Create Maven structure
mkdir -p src/main/java
```

```
mv HelloWorld.java src/main/java/

# 5. Create minimal pom.xml
nano pom.xml
# Paste minimal Maven project configuration

# 6. Install Maven and build project
sudo apt install maven -y
mvn clean package

# 7. Create Dockerfile
nano Dockerfile
# Paste:
# FROM openjdk:17
# COPY target/myapp-1.0-SNAPSHOT.jar app.jar
# ENTRYPOINT ["java","-jar","app.jar"]

# 8. Build Docker image
sudo docker build -t myapp:latest .

# 9. Verify Docker image
sudo docker images
```