🧩 **SET 4 – Independent Ubuntu Commands**

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

```
sudo apt update
sudo apt install git -y

mkdir DevOpsProject
cd DevOpsProject
git init

nano HelloWorld.java

git add HelloWorld.java
git commit -m "Initial commit - HelloWorld.java"
git branch -M main
git remote add origin https://github.com/vilas423/<repo-name>.git
git push -u origin main
```

Q2 – Create Maven Project & Build

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

java -version
mvn -v

mvn archetype:generate -DgroupId=com.example -DartifactId=myapp \
-DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false

cd myapp
mvn clean package
```

Q3 – Create Dockerfile & Build Docker Image

sudo apt update

sudo apt install docker.io -y

sudo systemctl start docker

sudo systemctl enable docker

cd ~/myapp

nano Dockerfile

FROM openjdk:17

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

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

sudo docker build -t myapp:latest .

sudo docker images

🧩 **SET 5 – Independent Ubuntu Commands**

Q1 – Create Jenkins Freestyle Project with Maven Build

sudo apt install git -y

sudo apt install maven -y

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

Open Jenkins → http://localhost:8080

- SCM → Git URL: https://github.com/vilas423/<repo-name>.git

- Build → Execute Shell: mvn clean package

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

git clone https://github.com/vilas423/<repo-name>.git

cd <repo-name>

echo "// Test webhook" >> HelloWorld.java

git add HelloWorld.java

git commit -m "Testing GitHub webhook"

git push

GitHub → Settings → Webhooks → Add:
Payload URL: http://<jenkins-server>:8080/github-webhook/
Content type: application/json
Enable “GitHub hook trigger for GITScm polling”.

Q3 – Run Docker Container and Verify Logs

sudo apt update

sudo apt install docker.io -y

sudo systemctl start docker

sudo systemctl enable docker

mkdir DockerTest && cd DockerTest

nano HelloWorld.java

mkdir -p src/main/java

mv HelloWorld.java src/main/java/

nano pom.xml # add minimal Maven config

sudo apt install maven -y

mvn clean package

nano Dockerfile

FROM openjdk:17

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

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

sudo docker build -t myapp:latest .

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

sudo docker ps

sudo docker logs mycontainer

🧩 **SET 6 – Independent Ubuntu Commands**

Q1 – Clone Repo, Modify Java File, Commit Push

sudo apt update

sudo apt install git -y

sudo apt install openjdk-17-jdk -y

java -version

git clone https://github.com/vilas423/<repo-name>.git

cd <repo-name>

nano HelloWorld.java

git add HelloWorld.java

git commit -m "Updated HelloWorld.java"

git push

Q2 – Jenkins Pipeline (Checkout → Build → Test)

sudo apt update

sudo apt install git openjdk-17-jdk maven -y

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
```

Jenkinsfile

```
pipeline {
  agent any
  stages {
    stage('Checkout') {
      steps { git 'https://github.com/vilas423/<repo-name>.git' }
    }
    stage('Build') {
      steps { sh 'mvn clean package' }
    }
    stage('Test') {
      steps { sh 'mvn test' }
    }
  }
}
```

Q3 – Build Docker Image for Java Project and Verify

```
sudo apt update
sudo apt install docker.io maven -y
sudo systemctl start docker
sudo systemctl enable docker

mkdir DockerJavaProject && cd DockerJavaProject
nano HelloWorld.java
# public class HelloWorld {
```

```
#   public static void main(String[] args) {
#     System.out.println("Hello Docker World!");
#   }
# }

mkdir -p src/main/java
mv HelloWorld.java src/main/java/
nano pom.xml
mvn clean package

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

sudo docker build -t myapp:latest .
sudo docker images
```

🧩 **SET 1**

Q1 – Install and Start Nginx (Ansible Playbook)

```
- name: Install and Start Nginx Web Server
  hosts: all
  become: yes
  tasks:
    - name: Update APT package index
      apt:
        update_cache: yes
    - name: Install Nginx
```

```
    apt:
      name: nginx
      state: present
  - name: Start and enable Nginx service
    service:
      name: nginx
      state: started
      enabled: yes
```

Run:

ansible-playbook -i inventory nginx_setup.yml

Q2 – Run Docker Container and Show Logs

docker images

docker run -d --name myapp-container myapp:latest

docker ps

docker logs myapp-container

Q3 – Manual Maven Build in Jenkins

- Jenkins → New Freestyle Project
- Add Git Repo https://github.com/vilas423/<repo-name>.git
- Build → Invoke top-level Maven targets → clean package

🧩 **SET 2**

Q1 – Install and Start Apache2 (Ansible Playbook)

```
- name: Install and Start Apache2 Web Server
  hosts: all
  become: yes
  tasks:
    - name: Update APT packages
      apt:
```

```
    update_cache: yes
  - name: Install Apache2
    apt:
      name: apache2
      state: present
  - name: Start and enable Apache2 service
    service:
      name: apache2
      state: started
      enabled: yes
```

Q2 – Dockerfile Build Image Push to Docker Hub

```
mkdir docker-exam && cd docker-exam
nano app.py
# print("Hello from Docker image!")

nano Dockerfile
# (add your Dockerfile)

docker build -t myimage:latest .
docker run --rm myimage:latest
docker tag myimage:latest vilas423/<repo-name>:latest
docker login
docker push vilas423/<repo-name>:latest
docker images | grep <repo-name>
```

Q3 – Git Version Control

```
git init
echo "Version 1" > file.txt
git add file.txt
```

```
git commit -m "Initial version"
echo "Version 2" > file.txt
git commit -am "Updated to version 2"
git remote add origin https://github.com/vilas423/<repo>.git
git push -u origin master
```

🧩 **SET 3**

Q1 – Install and Start HAProxy (Ansible Playbook)

```
- name: Install and Configure HAProxy
  hosts: localhost
  connection: local
  become: yes
  tasks:
    - name: Update APT package index
      apt:
        update_cache: yes
    - name: Install HAProxy
      apt:
        name: haproxy
        state: present
    - name: Enable and start HAProxy service
      service:
        name: haproxy
        state: started
        enabled: yes
```

Q2 – Dockerfile for Python Application

```
nano app.py
# print("Hello from Docker and Python!")
```

nano Dockerfile

(add Dockerfile content)

docker build -t hello-python .

docker run --rm hello-python

Q3 – Git Repository Setup

git init

echo "<h1>Hello GitHub</h1>" > index.html

git add index.html

git commit -m "Initial commit"

git remote add origin https://github.com/vilas423/<repo>.git

git branch -M main

git push -u origin main

Notes

- Localhost IP for Ansible/HAProxy tests → 127.0.0.1
- Add --ask-become-pass for Ansible sudo password.
- Jenkins initial admin password:
  sudo cat /var/lib/jenkins/secrets/initialAdminPassword