■ BEGINNER'S DEVOPS PRACTICALS GUIDE (Set 1 → Set 3)

■ SET 1 – Basic Setup and Coding

■ Q1 – Create and Run a Java Program, Upload to GitHub

1. Open Terminal → Ctrl + Alt + T
2. Install Java:

```
sudo apt update
sudo apt install default-jdk -y
```

3. Check version: java -version
4. Create folder:

```
mkdir devpractice && cd devpractice
```

5. Create file:

```
nano HelloWorld.java
```

Paste:

```
public class HelloWorld {
public static void main(String[] args) {
System.out.println("Hello, world!");
}
}
```

Save → Ctrl+O, Enter | Exit → Ctrl+X

6. Run program:

```
javac HelloWorld.java
java HelloWorld
```

■ Output: Hello, world!

7. Install Git: sudo apt install git -y
8. Configure Git:

```
git config --global user.name "Your Name"
git config --global user.email "you@example.com"
```

9. Initialize Git and commit:

```
git init
git add HelloWorld.java
git commit -m "First Java program"
```

10. Go to GitHub → create new repo → copy repo URL
11. Link and push:

```
git branch -M main
git remote add origin https://github.com//myapp-example.git
git push -u origin main
```

■ Q2 – Create a Maven Project

1. Install Maven: sudo apt install maven -y
2. Create project:

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

3. Go into project:

cd myapp

4. Build project:

mvn clean package

5. Run:

java -cp target/myapp-1.0-SNAPSHOT.jar com.example.App

■ Output: Hello World!

■ Q3 – Create and Run a Dockerfile

1. Create Dockerfile:

nano Dockerfile

Paste:

FROM eclipse-temurin:17-jre

WORKDIR /app

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

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

2. Build image:

docker build -t myapp:1.0 .

3. Run container:

docker run --rm myapp:1.0

■ Output appears inside Docker container.

■■ SET 2 – Jenkins (Automation)

■ Q1 – Run Jenkins and Build Project

1. Start Jenkins in Docker:

docker run -d --name jenkins -p 8080:8080 -p 50000:50000 -v jenkins_home:/var/jenkins_home
jenkins/jenkins:lts

2. Open browser → http://localhost:8080

3. Get password:

docker exec jenkins cat /var/jenkins_home/secrets/initialAdminPassword

4. Install plugins → Create job (Freestyle Project)

5. Add GitHub repo URL → Build → Goal: clean package

6. Click Build Now → ■ BUILD SUCCESS

■ Q2 – Setup GitHub Webhook

1. In Jenkins → Configure job → Enable “GitHub hook trigger”

2. In GitHub → Settings → Webhooks → Add:

URL: http://:8080/github-webhook/

Event: Push event only

3. Push new code → Jenkins builds automatically.

■ Q3 – Run Docker Container and View Logs

```
docker run -d --name myapp -p 8080:8080 myapp:1.0
docker ps
docker logs myapp
docker stop myapp
```

■■ SET 3 – Pipelines and Version Control Flow

■ Q1 – Clone, Edit, Commit and Push Changes

1. cd ~/devpractice
2. git clone https://github.com//myapp-example.git
3. cd myapp-example
4. git checkout -b change-message
5. nano HelloWorld.java → change message → save
6. git add .
7. git commit -m "Changed message"
8. git push -u origin change-message
9. Go to GitHub → Create Pull Request → Merge.

■ Q2 – Create Jenkins Pipeline

1. nano Jenkinsfile

Paste:

```
pipeline {
agent any
tools { maven 'Maven' }
stages {
stage('Checkout') { steps { checkout scm } }
stage('Build') { steps { sh 'mvn clean package' } }
stage('Test') { steps { sh 'mvn test' } }
stage('Archive') { steps { archiveArtifacts 'target/*.jar' } }
}
}
```

2. Commit and push:

```
git add Jenkinsfile
git commit -m "Add Jenkins pipeline"
git push
```

3. In Jenkins → New Item → Pipeline → Script from SCM → Add Git repo → Build Now

----

■ Q3 – Add Docker Build Stage

Add below last stage in Jenkinsfile:

```
stage('Docker Build') {
steps {
```

```
sh 'docker build -t myapp:latest .'
}
}
```

Commit and push → Run pipeline again →

Check image:

```
docker images | grep myapp
```

■ Summary of Tools

Java → Write and run programs

Git → Save and track changes

GitHub → Upload code online

Maven → Build Java apps

Docker → Create containers

Jenkins → Automate builds

Webhook → Trigger builds on push

Pipeline → Complete CI/CD automation

■ END OF GUIDE – YOU DID IT!