DevOps Practical Session

1. Terminal Session: Creating a "Hello World" Java Application

1. Check if Java is Installed

```
bash

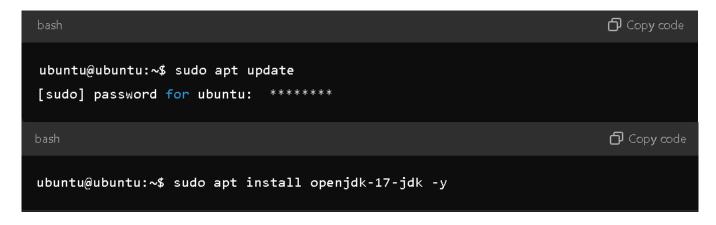
□ Copy code

ubuntu@ubuntu:~$ java -version
```

Output (if Java is not installed):

```
Command 'java' not found, but can be installed with:
sudo apt install default-jdk
```

2. Install Java



Output:

```
Setting up openjdk-17-jdk-headless ...
java 17 is now installed.
```

3. Create a Project Directory

```
bash

ubuntu@ubuntu:∼$ mkdir hello-world-java

ubuntu@ubuntu:∼$ cd hello-world-java
```

4. Create the Java File

```
bash

ubuntu@ubuntu:~/hello-world-java$ vim HelloWorld.java
```

In Vim (after pressing i to enter insert mode), type this:

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

Save and Exit:

Press Esc and type: wq

5. Compile the Java Code



Output (no errors):



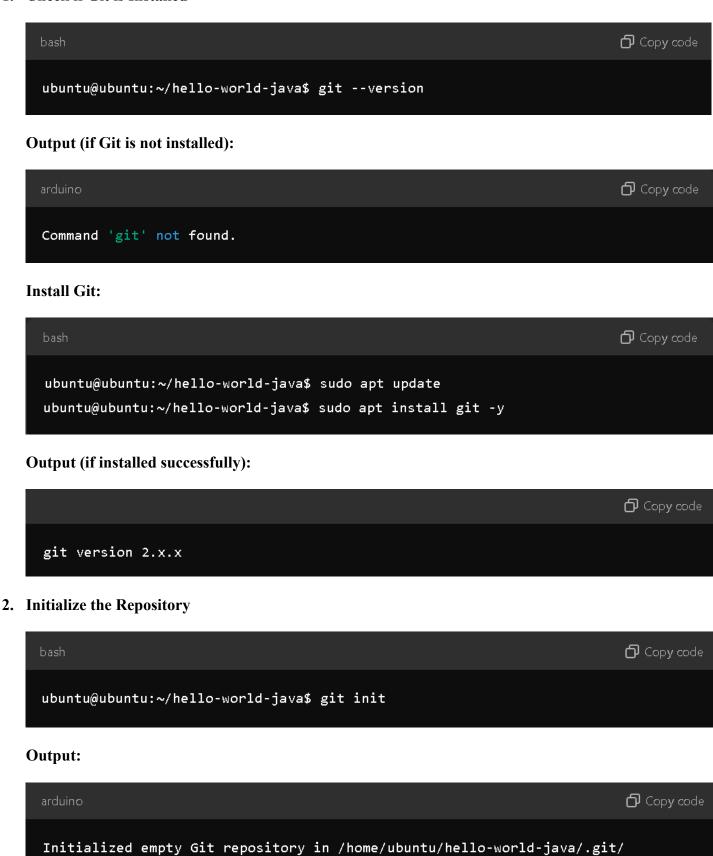
6. Run the Java Program





2. Terminal Session: Git Repository Initialization

1. Check if Git is Installed



3. Check Git Status

bash

ubuntu@ubuntu:∼/hello-world-java\$ git status

Output:

```
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
  HelloWorld.class
  HelloWorld.java

nothing added to commit but untracked files present (use "git add" to track)
```

4. Stage Files for the First Commit

```
bash

□ Copy code

ubuntu@ubuntu:~/hello-world-java$ git add .
```

Output (No message; files staged):

```
Csharp

[No output if successful]
```

Check Staged Files:



```
On branch master

No commits yet

Changes to be committed:

(use "git rm --cached <file>..." to unstage)

new file: HelloWorld.class
new file: HelloWorld.java
```

5. Make the Initial Commit

bash

ubuntu@ubuntu:~/hello-world-java\$ git commit -m "Initial commit: Added HelloWorld Java application"

Output:

```
[master (root-commit) eadbeef] Initial commit: Added HelloWorld Java application
2 files changed, 14 insertions(+)
create mode 100644 HelloWorld.class
create mode 100644 HelloWorld.java
```

3. Terminal Session: Pushing to GitHub

1. Create a Repository on GitHub

- Open GitHub in your browser: <u>GitHub</u>
- Log in and click on the **New Repository** button.
- Fill in:
 - Repository Name: hello-world-java
 - **Privacy Setting:** Choose Private or Public.
- Click Create Repository.
- You'll get a repository URL like: https://github.com/your-username/hello-world-java.git

2. Add Remote Repository

hash

ubuntu@ubuntu:~/hello-world-java\$ git remote add origin https://github.com/your-username/hello-world-java.git

Check Remote:

```
bash

ubuntu@ubuntu:~/hello-world-java$ git remote -v
```

```
origin https://github.com/your-username/hello-world-java.git (fetch)
origin https://github.com/your-username/hello-world-java.git (push)
```

3. Push to GitHub

```
bash

ubuntu@ubuntu:~/hello-world-java$ git branch -M main

bash

□ Copy code

ubuntu@ubuntu:~/hello-world-java$ git push -u origin main
```

Output:

```
Username for 'https://github.com': your-username
Password for 'https://your-username@github.com': *******

Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 240 bytes | 240.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/your-username/hello-world-java.git
  * [new branch] main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
```

4. Verify Your Code on GitHub

- Go to your GitHub repository in your browser.
- Your files (HelloWorld.java and HelloWorld.class) should now appear.

4. Terminal Session: Installing Jenkins

1. Update the System Packages

```
ubuntu@ubuntu:~$ sudo apt update -y && sudo apt upgrade -y
[sudo] password for ubuntu: *******
```

```
python

Hit:1 http://archive.ubuntu.com/ubuntu focal InRelease
... (other packages listed)
```

2. Install Java (Required for Jenkins)

Jenkins requires Java to run, so install OpenJDK:

```
bash

□ Copy code

ubuntu@ubuntu:~$ sudo apt install openjdk-11-jdk -y
```

Verify the Installation:

```
bash

ubuntu@ubuntu:~$ java -version
```

Output:

```
mathematica

Openjdk version "11.0.x"

OpenJDK Runtime Environment ...
```

3. Add Jenkins Repository Key

bash

ubuntu@ubuntu:~\$ sudo wget -O /usr/share/keyrings/jenkins-keyring.asc https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key

Output:

```
lua Copy code

--2024-12-27 00:00:00-- https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key

Resolving pkg.jenkins.io...
jenkins.io-2023.key saved.
```

4. Add Jenkins Repository

bash

ubuntu@ubuntu:~\$ echo "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] https://pkg.jenkins.io/debian-stable binary/" | sudo tee /etc/apt/sources.list.d/jenkins.list > /dev/null

5. Update Package Lists and Install Jenkins

```
bash
ubuntu@ubuntu:∼$ sudo apt update -y
```

Install Jenkins:

```
bash
ubuntu@ubuntu:∼$ sudo apt install jenkins -y
```

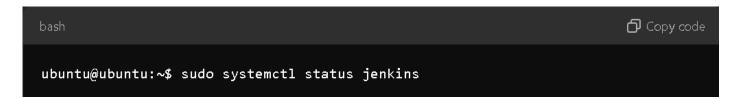
Output:

Setting up jenkins ...

6. Start and Enable Jenkins



Check Jenkins Status:



Output:

```
■ jenkins.service - Jenkins Continuous Integration Server

Active: active (running) since Fri YYYY-MM-DD HH:MM:SS
```

7. Access Jenkins

• Open a browser and go to:

```
arduino

Copy code

http://localhost:8080
```

- (Replace localhost with your server's IP if using a remote machine.)
- You will be prompted to unlock Jenkins.

8. Unlock Jenkins



9. Install Suggested Plugins

• Choose "Install suggested plugins" and wait for the installation to complete.

10. Create an Admin User

- Enter your preferred username, password, and email.
- Click Save and Finish.

5. Terminal Session: Configuring Jenkins Job

1. Access Jenkins Dashboard

- 1. Open a browser on the Ubuntu machine or connect remotely.
- 2. Go to:



2. Login

• Login with the credentials you set up earlier (admin username and password).

3. Create a New Job

- 1. Click on "New Item" from the Jenkins dashboard.
- 2. Enter a name for your job, such as HelloWorld-Java-Job.
- 3. Select "Freestyle project" and click OK.

4. Configure Job Details

- 1. Source Code Management (SCM):
- Choose Git.
- In the repository URL field, enter the URL of your GitHub repository:

```
arduino
☐ Copy code

https://github.com/your-username/hello-world-java.git
```

If authentication is required:

• Add Jenkins credentials for GitHub.

2. Build Triggers:

- Select **Poll SCM** if you want periodic builds.
- Set the schedule (e.g., every 5 minutes): H/5 * * * *

3. Add Build Steps:

- Under the **Build** section, choose **Execute Shell**.
- Add the following script:

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

5. Save and Build Job

- 1. Click **Save** to save the job configuration.
- 2. On the job's page, click Build Now.

6. Check Build Status

- 1. In the **Build History**, click the build number (e.g., #1).
- 2. Go to **Console Output** to view the results.

Sample Console Output (If Successful):

```
Building in workspace /var/lib/jenkins/workspace/HelloWorld-Java-Job
[INFO] Checking out repository...
[INFO] Running shell script
[INFO] Compiling HelloWorld.java
[INFO] Executing program:
Hello, World!

Finished: SUCCESS
```

6. Terminal Session: Configuring Jenkins Pipeline

1. Access Jenkins Dashboard

- Open your browser on the Ubuntu machine.
- Navigate to:

2. Create a New Pipeline Job

- From the Jenkins Dashboard, click New Item.
- Enter a name for your pipeline job, such as HelloWorld-Java-Pipeline.
- Select Pipeline and click OK.

3. Write the Pipeline Script

- Scroll down to the **Pipeline** section.
- Select **Pipeline script** and paste the following code:

```
pipeline {
  agent any
  stages {
     stage('Clone') {
       steps {
          git branch: 'main', url: 'https://github.com/your-username/hello-world-java.git '
     stage('Build') {
       steps {
          sh 'javac HelloWorld.java'
     stage('Run') {
       steps {
          sh 'java HelloWorld'
  }
  post {
       echo 'Build and execution successful!'
     failure {
       echo 'Build or execution failed!'
```

4. Save and Run the Pipeline

- Click Save.
- On the pipeline job's page, click Build Now.

5. Monitor Pipeline Progress

- View the pipeline's progress in **Build History** and check the **Console Output**.
- The output should show:
 - Git repository cloning.
 - Java code compilation.
 - Execution of java HelloWorld

Sample Console Output (If Successful):

[Pipeline] Start of Pipeline

[Pipeline] { (Clone)

Cloning repository...

Cloning branch 'main' of repository 'https://github.com/your-username/hello-world-java.git '

[Pipeline] { (Build)

Compiling HelloWorld.java...

[Pipeline] { (Run)

Executing program...

Hello, World!

[Pipeline] End of Pipeline Finished: SUCCESS