Name: Sameer Shah

Department: Cyber Security

Div: BE-15

Roll No: 33

Subject: DSO

Experiment No. – 1				
Date of Performance:	15/6/2024			
Date of Submission:	22/6/2024			
Program Execution/ formation/ correction/ ethical practices (06)	Timely Submission (01)	Viva (03)	Experiment Total (10)	Sign with Date

Experiment No. 1

Aim: To Understand Version Control System / Source Code Management, install git and to perform various GIT operations on local remote repositories.

Lab Outcome: CSL701.1 Understand the concepts of distributed version control using GIT and GITHUB

Theory:

Definition of Git

Git is an open-source version control system for projects of all sizes, ensuring quick and efficient collaboration among developers. It is used to track changes and coordinate work within teams, allowing teamwork in the same workspace.

Git forms the basis of services like GitHub and GitLab, although it can be used independently. It is usable both privately and publicly.

Created in 2005 by Linus Torvalds for the Linux Kernel, Git is vital for DevOps and distributed version control. It is user-friendly, high-performance, and surpasses other tools like Subversion, CVS, and ClearCase.

Features of Git

1. **Open Source**:

Git is an open-source tool. It is released under the GPL (General Public License).

2. **Scalable**:

Git is scalable, which means when the number of users increases, the Git can easily handle such situations.

3. **Distributed**:

One of Git's great features is that it is distributed. Distributed means that instead of switching the project to another machine, we can create a "clone “of the entire repository.

Also, instead of just having one central repository that you send changes to, every user has their own repository that contains the entire commit history of the project.

We do not need to connect to the remote repository; the change is just stored on our local repository. If necessary, we can push these changes to a remote repository.

4. **Security**

Git is secure, using SHA-1 to uniquely identify objects in its repository.

Files and commits are verified during checkout using checksums.

Commit IDs depend on the entire development history, enhancing security.

Once published, old versions cannot be altered, maintaining integrity.

Output:

Installation of Git:

Go to the website and download the ‘git’ file according to your system configuration.

Operations on GIT

1. Config

```
$ git config --global user.name "black-knight2"
```

2. Git init & add

```
$ git init
Initialized empty Git repository in F:/CYSE/DevSecOps/GIT/.git/


$ git add button.html
```

3. Diff

```
$ git diff
diff --git a/para.html b/para.html
index f50048f..5f8040f 100644
--- a/para.html
+++ b/para.html
@@ -2,7 +2,8 @@
 <html>
 <body>

-<p>This is a paragraph.</p>
+<p>My first paragraph.</p>
+
 <p>This is another paragraph.</p>

 </body>
```

4. Status

```
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   button.html
        new file:   para.html

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   para.html
```

5. Remote

```
$ git remote add Testing https://github.com/black-knight2/Testing.git
```

6. Commit

```
$ git commit -m "first commit"
[master (root-commit) d919bff] first commit
 2 files changed, 21 insertions(+)
 create mode 100644 button.html
 create mode 100644 para.html
```

7. Branch & checkout

```
$ git branch -M main
```

```
$ git checkout main
Already on 'main'
M       para.html
```

8. Push

Push the file from remote server

```
$ git remote add origin https://github.com/black-knight2/Testing.git
git branch -M main
git push -u origin main
info: please complete authentication in your browser...
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 438 bytes | 438.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/black-knight2/Testing.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

9. Log

```
$ git log
commit d919bff72330077bf53a6fab15f080b53f1bf3de (HEAD -> master)
Author: black-knight2 <2002hetchheda@gmail.com>
Date:   Wed Aug 16 16:21:47 2023 +0530

    first commit
```

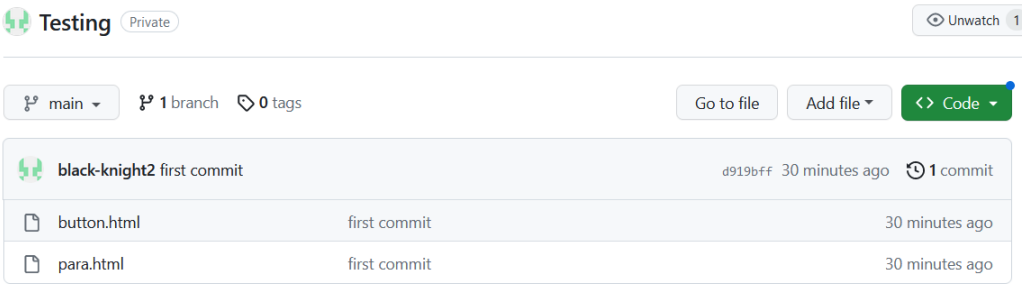
10. Reset

```
$ git reset --hard
HEAD is now at d919bff first commit
```

11. Stash

```
$ git stash
Saved working directory and index state WIP on master: d919bff first commit
```

GitHub repository



Conclusion:

The Version Control System was understood in this experiment. Git was installed and Git bash was exercised. A GitHub account was created, and a repository was made. Difference between Git and GitHub was made clear.

Subject: DSO

Experiment No. – 2				
Date of Performance:	22/7/2024			
Date of Submission:	29/7/2024			
Program Execution/ formation/ correction/ ethical practices (06)	Timely Submission (01)	Viva (03)	Experiment Total (10)	Sign with Date

Experiment No. 2

Aim: To implement version control using GitHub to sync local git repositories and perform various related operations

Lab Outcome: CSL701.1 Understand the concepts of distributed version control using GIT and GITHUB

Theory:

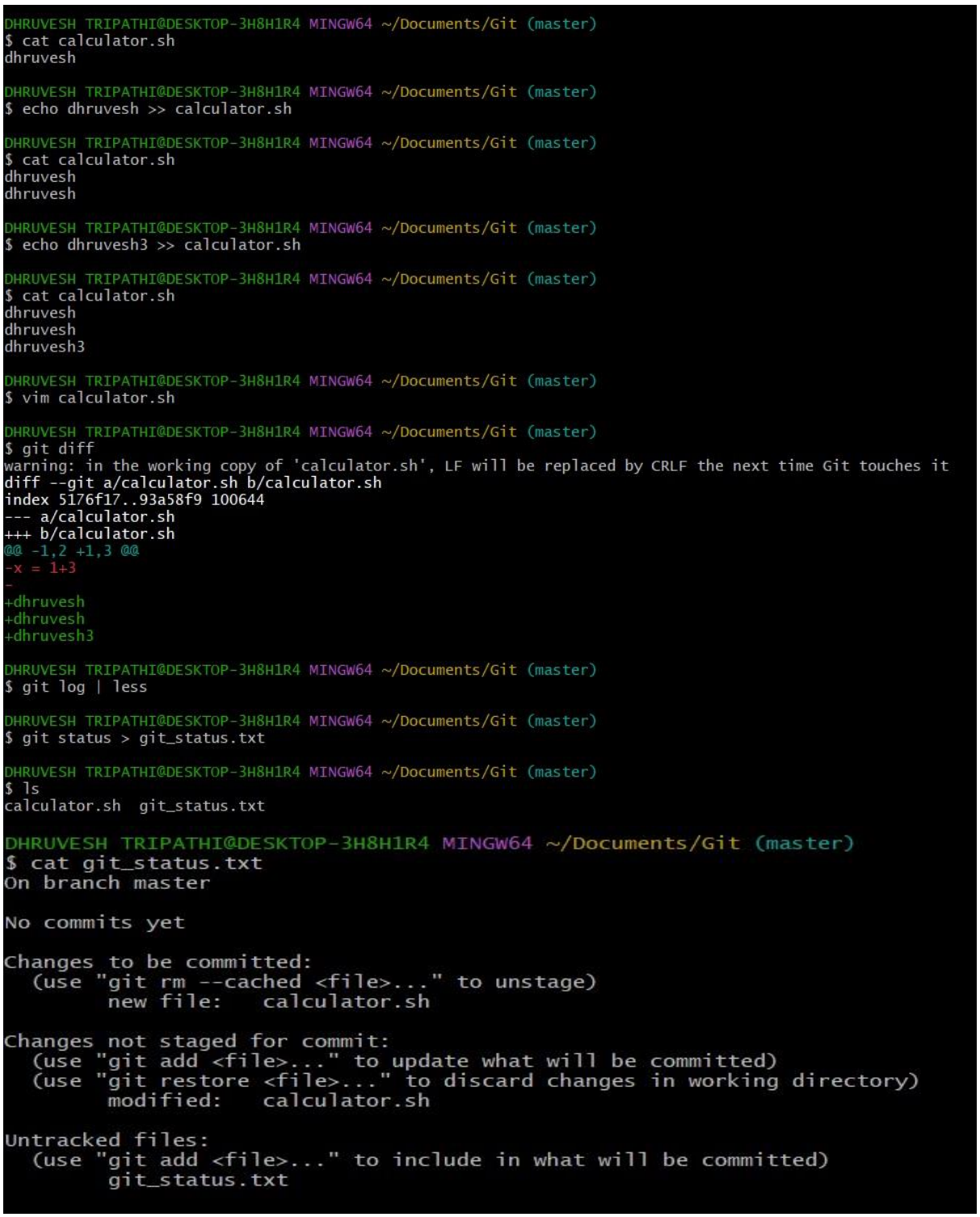
Git init
Create a local repository**:** $ git init
Git clone
Make a local copy of the server repository: $ git clone

Git diff
Track the changes that have not been staged: $ git diff
Track the changes that have staged but not committed: $ git diff --staged
Track the changes after committing a file: $ git diff HEAD
Track the changes between two commits: $ git diff
Git Diff Branches: $ git diff < branch 2>
Git status
Display the state of the working directory and the staging area: $ git status
Git show
Shows objects: $ git show

Output:
Installation of Git:
Go to the website and download the 'git' file according to your system configuration.

```
DHRUVESH TRIPATHI@DESKTOP-3H8H1R4 MINGW64 ~/Documents/Git (master)
$ git init
Initialized empty Git repository in C:/Users/DHRUVESH TRIPATHI/Documents/Git/.git/

DHRUVESH TRIPATHI@DESKTOP-3H8H1R4 MINGW64 ~/Documents/Git (master)
$ ls
\calculator.sh

DHRUVESH TRIPATHI@DESKTOP-3H8H1R4 MINGW64 ~/Documents/Git (master)
$ ls -a
./  ../  .git/  calculator.sh

DHRUVESH TRIPATHI@DESKTOP-3H8H1R4 MINGW64 ~/Documents/Git (master)
$ ls .git/hooks/pre-commit
ls: cannot access '.git/hooks/pre-commit': No such file or directory

DHRUVESH TRIPATHI@DESKTOP-3H8H1R4 MINGW64 ~/Documents/Git (master)
$ ls  .git/hooks/pre-commit
ls: cannot access '.git/hooks/pre-commit': No such file or directory

DHRUVESH TRIPATHI@DESKTOP-3H8H1R4 MINGW64 ~/Documents/Git (master)
$ ls -lrta
total 44
drwxr-xr-x 1 DHRUVESH TRIPATHI 197121 0 Oct 20 12:28 ../
drwxr-xr-x 1 DHRUVESH TRIPATHI 197121 0 Oct 20 12:28 .git/
-rw-r--r-- 1 DHRUVESH TRIPATHI 197121 0 Oct 20 12:29 calculator.sh
drwxr-xr-x 1 DHRUVESH TRIPATHI 197121 0 Oct 20 12:29 ./

DHRUVESH TRIPATHI@DESKTOP-3H8H1R4 MINGW64 ~/Documents/Git (master)
$ git status
On branch master

No commits yet

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

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

DHRUVESH TRIPATHI@DESKTOP-3H8H1R4 MINGW64 ~/Documents/Git (master)
$ ls
calculator.sh

DHRUVESH TRIPATHI@DESKTOP-3H8H1R4 MINGW64 ~/Documents/Git (master)
$ git add calculator.sh

DHRUVESH TRIPATHI@DESKTOP-3H8H1R4 MINGW64 ~/Documents/Git (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   calculator.sh
```

```
DHRUVESH TRIPATHI@DESKTOP-3H8H1R4 MINGW64 ~/Documents/Git (master)
$ vim calculator.sh
DHRUVESH TRIPATHI@DESKTOP-3H8H1R4 MINGW64 ~/Documents/Git (master)
$ :wq is for saving and exit from vim
bash: :wq: command not found

DHRUVESH TRIPATHI@DESKTOP-3H8H1R4 MINGW64 ~/Documents/Git (master)
$ :q! is to exit without saving
bash: :q!: command not found

DHRUVESH TRIPATHI@DESKTOP-3H8H1R4 MINGW64 ~/Documents/Git (master)
$ git diff
warning: in the working copy of 'calculator.sh', LF will be replaced by CRLF the next time Git touches it
diff --git a/calculator.sh b/calculator.sh
index e69de29..5176f17 100644
--- a/calculator.sh
+++ b/calculator.sh
@@ -0,0 +1,2 @@
+x = 1+3
+

DHRUVESH TRIPATHI@DESKTOP-3H8H1R4 MINGW64 ~/Documents/Git (master)
$ git diff
warning: in the working copy of 'calculator.sh', LF will be replaced by CRLF the next time Git touches it
diff --git a/calculator.sh b/calculator.sh
index e69de29..5176f17 100644
--- a/calculator.sh
+++ b/calculator.sh
@@ -0,0 +1,2 @@
+x = 1+3
+

DHRUVESH TRIPATHI@DESKTOP-3H8H1R4 MINGW64 ~/Documents/Git (master)
$ git add calculator.sh
warning: in the working copy of 'calculator.sh', LF will be replaced by CRLF the next time Git touches it

DHRUVESH TRIPATHI@DESKTOP-3H8H1R4 MINGW64 ~/Documents/Git (master)
$ git checkout calculator.sh
Updated 0 paths from the index

DHRUVESH TRIPATHI@DESKTOP-3H8H1R4 MINGW64 ~/Documents/Git (master)
$ git checkout cal
error: pathspec 'cal' did not match any file(s) known to git

DHRUVESH TRIPATHI@DESKTOP-3H8H1R4 MINGW64 ~/Documents/Git (master)
$ ls
calculator.sh

DHRUVESH TRIPATHI@DESKTOP-3H8H1R4 MINGW64 ~/Documents/Git (master)
$ cat calculator.sh
x = 1+3


DHRUVESH TRIPATHI@DESKTOP-3H8H1R4 MINGW64 ~/Documents/Git (master)
$ echo dhruvesh > calculator.sh
```



Conclusion:

Hence, I have learned and executed various git commands to perform task like creating directory, uploading data to directory, fetching data from directory etc.

Name: Sameer Shah

Department: Cyber Security

Div: BE-15

Roll No: 33

Subject: DSO

Experiment No. – 3				
Date of Performance:	29/7/2024			
Date of Submission:	05/8/2024			
Program Execution/ formation/ correction/ ethical practices (06)	Timely Submission (01)	Viva (03)	Experiment Total (10)	Sign with Date

Experiment No. 3

Aim: To implement Jenkins pipeline using scripted/declarative pipeline.

Lab Outcome: CSL701.2 Apply Jenkins to Build, Deploy and Test the Software Applications

Theory:
Jenkins: Jenkins is an open-source automation server that aids in automating various aspects of the software development lifecycle. It facilitates building, testing, and deploying software projects.

Why Use Jenkins: Jenkins streamlines development processes, enhances collaboration, and automates repetitive tasks. It offers extensibility through plugins and supports continuous integration and delivery.

Features: Jenkins provides an intuitive web interface, supports a wide range of plugins, and offers robust integration with version control systems, testing frameworks, and deployment tools.

Pipeline: A pipeline in Jenkins is a set of steps that define how software is built, tested, and deployed. It provides a structured approach to automating the entire delivery process.

Steps to Create a Scripted Pipeline:

1. **Install Jenkins:** Set up Jenkins on your server.

2. **Create a New Item:** In Jenkins, create a new "Pipeline" project.

3. **Define Scripted Pipeline:** In the project configuration, select "Pipeline script" and write your scripted pipeline code.

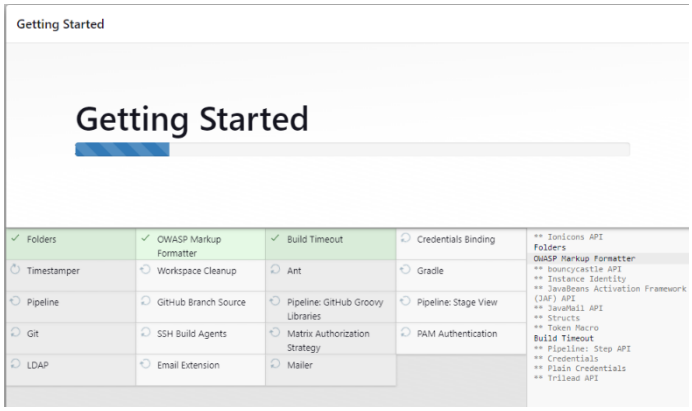
4. **Define Stages:** Define stages for building, testing, and deployment using the stage directive.

5. **Add Steps:** Within each stage, use steps like sh for shell commands, git for version control operations, etc.

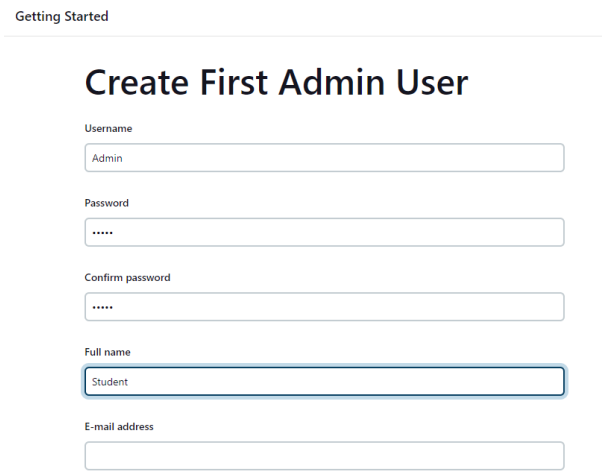
6. **Configure Post-Build Actions:** Set up post-build actions such as notifications, reports, or deployment triggers.

7. **Save and Run:** Save your pipeline configuration and run it to observe the flow and outcome.

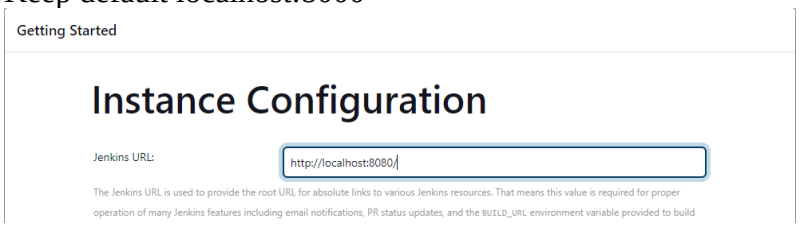
Output:
Launch the Jekins



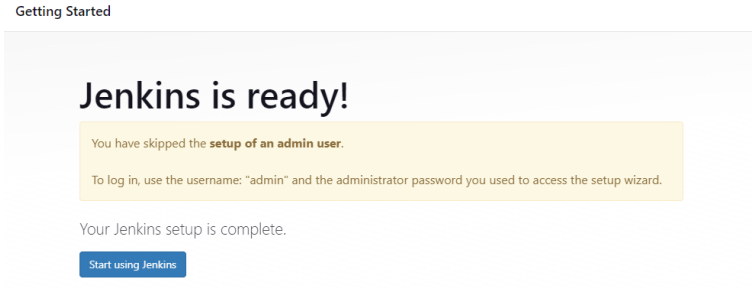
Create an Account



Configure it
Keep default localhost:8000



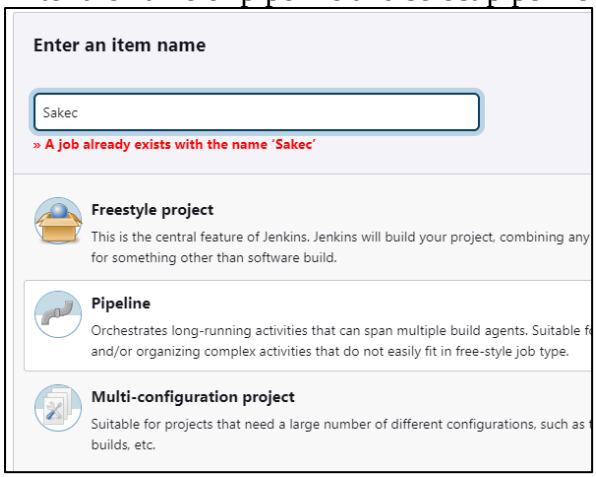
Completed the installation



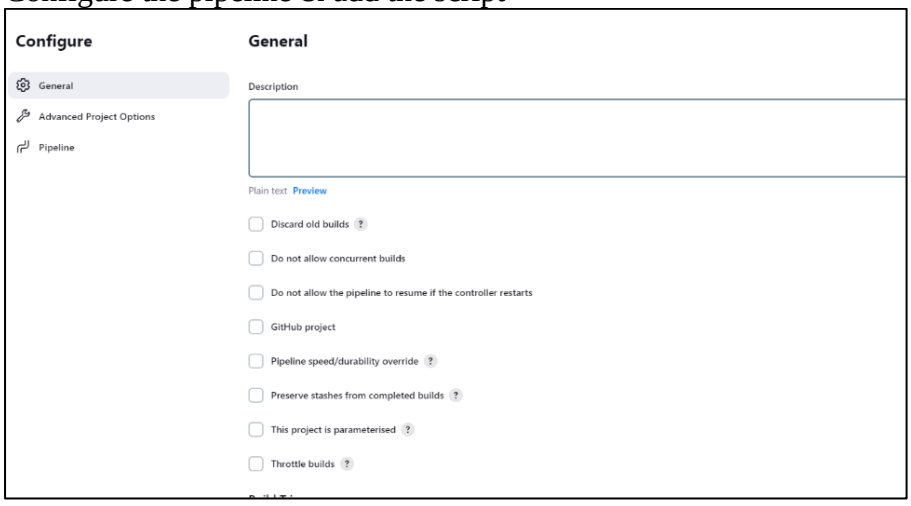
Sign into the account

Creating a pipeline

Enter the name of pipeline and select pipeline from below option

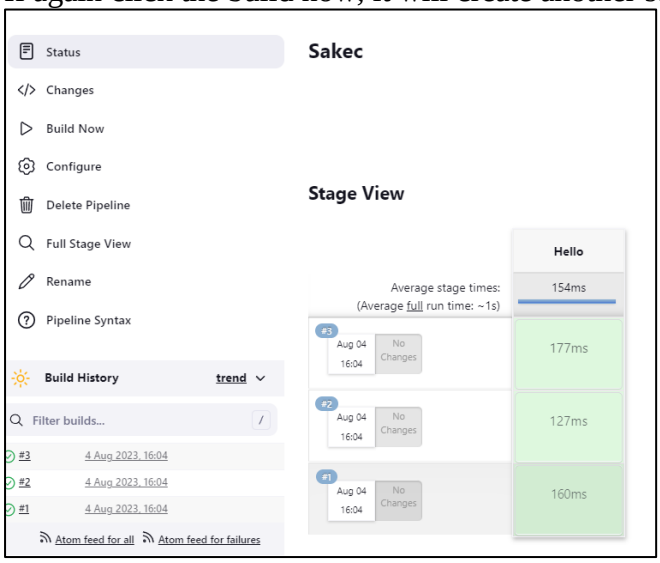


Configure the pipeline & add the script

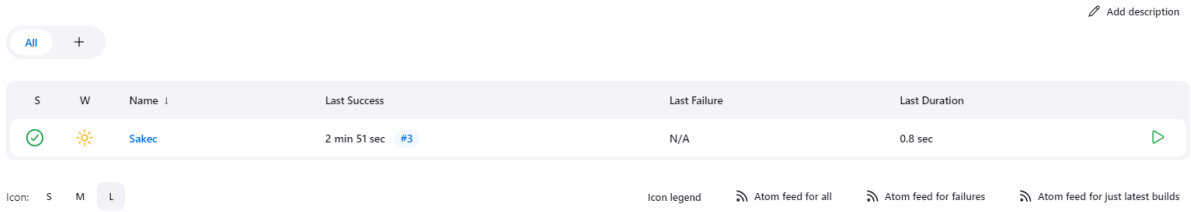


Apply the script and save it

After saving go to build now option which will build the stage
If again click the build now, it will create another stage on it

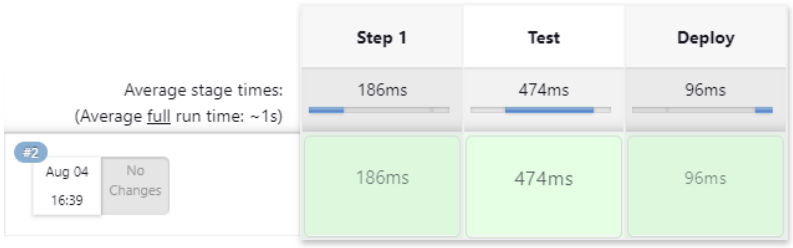


Dashboard

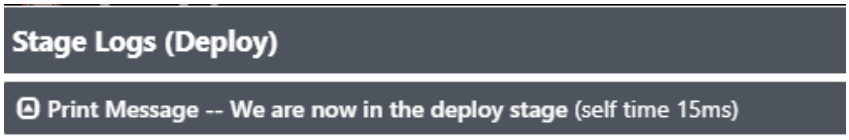


Now use another script to create pipeline

Stage View



See the logs of Deploy




```
We are now in the deploy stage
```

```
C:\ProgramData\Jenkins\.jenkins\workspace\Student>java --version
java 17.0.8 2023-07-18 LTS
Java(TM) SE Runtime Environment (build 17.0.8+9-LTS-211)
Java HotSpot(TM) 64-Bit Server VM (build 17.0.8+9-LTS-211, mixed mode, sharing)
```

Conclusion:

Implementing a Jenkins scripted pipeline empowers software teams to automate development workflows efficiently. It enhances collaboration, accelerates delivery, and ensures consistent software quality.

Name: Sameer Shah

Department: Cyber Security

Div: BE-15

Roll No: 33

Subject: DSO

Experiment No. – 4				
Date of Performance:	05/08/2024			
Date of Submission:	12/08/2024			
Program Execution/ formation/ correction/ ethical practices (06)	Timely Submission (01)	Viva (03)	Experiment Total (10)	Sign with Date

Experiment No. 4

Aim: To Setup and Run Selenium Tests in Jenkins Using Maven.

Lab Outcome: CSL701.2 Apply Jenkins to Build, Deploy and Test the Software Applications

Theory:
Selenium is a widely used tool for automating web applications, and Jenkins is a powerful automation server. Integrating Selenium tests with Jenkins using Maven streamlines the testing process.

Selenium: It's an open-source framework for automating web browsers, allowing testers to perform functional and regression testing on web applications.

Jenkins: An automation server that facilitates continuous integration and continuous delivery. It's used to automate building, testing, and deployment of software projects.

Maven: A build automation and project management tool. It simplifies project setup, handling dependencies, and building Java projects.

Steps to Set Up and Run Selenium Tests in Jenkins Using Maven:

Step 1: Install Jenkins
Download and install Jenkins on your server.
Set up and configure Jenkins according to your requirements.

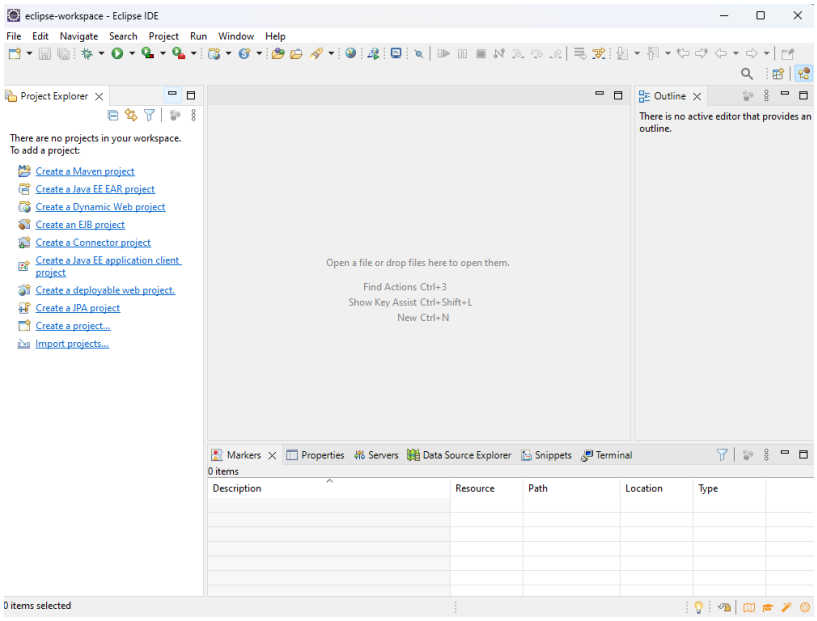
Step 2: Create a Jenkins Job
Create a new Jenkins job (Freestyle project).
Configure the job to fetch your Selenium test project from a version control repository.

Step 3: Install and Configure Maven
Ensure Maven is installed on your Jenkins server.
Configure Maven settings and paths in Jenkins.

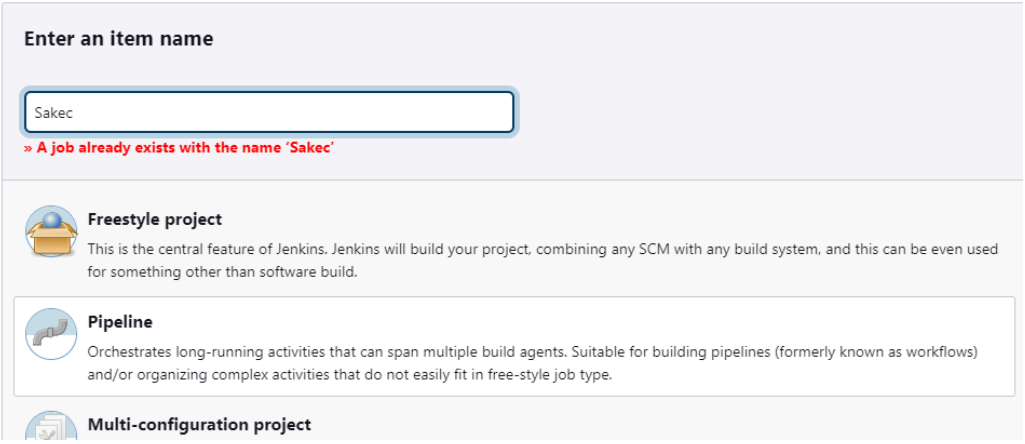
Step 4: Build Selenium Tests Using Maven
In the Jenkins job configuration, add a build step to execute Maven commands.
Use Maven commands like clean test to build and run Selenium tests.

Step 5: View Test Results
Configure the Jenkins job to generate test reports.
View test reports within Jenkins to analyze test results.

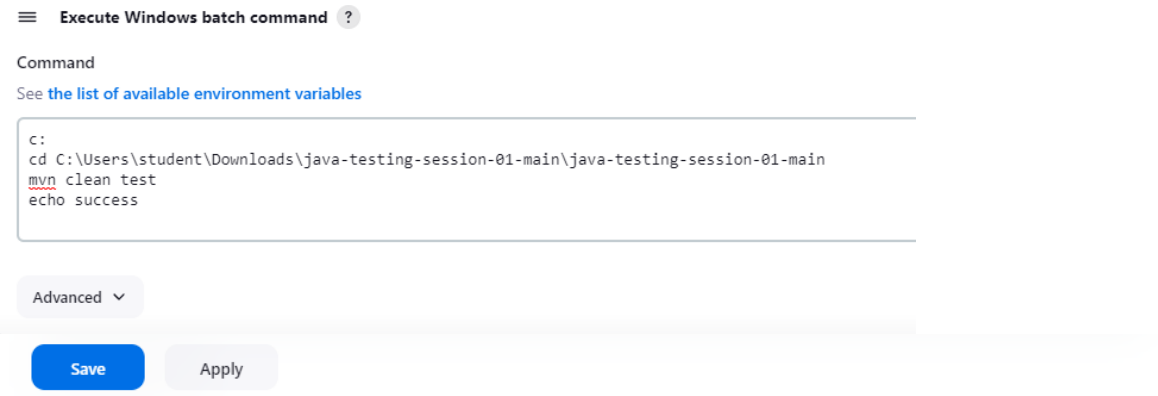
Output:
Installed Apache Maven.
Installed Eclipse for Maven project development.
Added a Maven project in Eclipse.



Created a Freestyle project in Jenkins.
Configured the Jenkins project with appropriate settings.



In the build steps, added the path to your Maven project and the command **mvn clean test**, and saved the configuration.



Used the "Build Now" option in Jenkins to initiate the build process.
Observed output for both success and failure cases.

Console Output

```
Started by user admin
Running as SYSTEM
Building in workspace C:\ProgramData\Jenkins\.jenkins\workspace\SeleniumTest1
Finished: SUCCESS
```

Console Output

```
Started by user admin
Running as SYSTEM
Building in workspace C:\ProgramData\Jenkins\.jenkins\workspace\SeleniumTest1
[SeleniumTest1] $ cmd /c call C:\WINDOWS\TEMP\jenkins4375057127926646358.bat

C:\ProgramData\Jenkins\.jenkins\workspace\SeleniumTest1>c:

C:\ProgramData\Jenkins\.jenkins\workspace\SeleniumTest1>cd C:\Users\student\Downloads\java-testing-session-01-main\java-

C:\Users\student\Downloads\java-testing-session-01-main\java-testing-session-01-main>mvn clean test
'mvn' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\student\Downloads\java-testing-session-01-main\java-testing-session-01-main>echo successful
successful

C:\Users\student\Downloads\java-testing-session-01-main\java-testing-session-01-main>exit 9009
Build step 'Execute Windows batch command' marked build as failure
Finished: FAILURE
```

Manually ran the command mvn clean test in the command prompt and received a successful result.

```
INFO] --- surefire:3.1.2:test (default-test) @ java-testing-collections ---
INFO] Using auto detected provider org.apache.maven.surefire.junit4.JUnit4Provider
INFO]
INFO] -------------------------------------------------------
INFO]  T E S T S
INFO] -------------------------------------------------------
INFO] Running es.seresco.cursojee.StackTest
INFO] Tests run: 6, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.041 s -- in es.seresco.curs
INFO]
INFO] Results:
INFO]
INFO] Tests run: 6, Failures: 0, Errors: 0, Skipped: 0
INFO]
INFO] ------------------------------------------------------------------------
INFO] Reactor Summary for java-testing-session-01 0.0.1:
INFO]
INFO] java-testing-session-01 ........................... SUCCESS [  0.666 s]
INFO] java-testing-calculadora .......................... SUCCESS [  3.600 s]
INFO] java-testing-collections .......................... SUCCESS [  1.244 s]
INFO] ------------------------------------------------------------------------
INFO] BUILD SUCCESS
INFO] ------------------------------------------------------------------------
INFO] Total time:  5.602 s
INFO] Finished at: 2023-08-11T16:23:42+05:30
INFO] ------------------------------------------------------------------------
S C:\Users\student\Downloads\java-testing-session-01-main\java-testing-session-01-main>
```

Conclusion:

Successfully configured Jenkins to seamlessly run Selenium tests using Maven, enhancing the testing process. This integration streamlines test automation and supports efficient continuous integration, enabling robust and reliable web application testing.

Name: Sameer Shah

Department: Cyber Security

Div: BE-15

Roll No: 33

Subject: DSO

Experiment No. – 5				
Date of Performance:	12/08/2024			
Date of Submission:	19/08/2024			
Program Execution/ formation/ correction/ ethical practices (06)	Timely Submission (01)	Viva (03)	Experiment Total (10)	Sign with Date

Experiment No. 5

Aim: Use Jenkins to deploy and run test cases for Java/Web applications using Selenium/JUni..

Lab Outcome: Analyze & Illustrate the Containerization of OS images and deployment of applications over Docker

Theory:

Docker:
Docker is a platform that enables you to develop, ship, and run applications within containers, offering a lightweight and consistent environment that can host various applications and even different operating systems.

Why Use Docker:
Docker simplifies application deployment by packaging software with all its dependencies into a single container. It provides consistency, isolation, and portability across various environments.

Features:
Containerization: Encapsulates applications and their dependencies.
Portability: Containers run consistently across various environments.
Efficiency: Shares host OS resources, minimizing overhead.
Isolation: Ensures applications don't interfere with each other.
Scalability: Easily scale applications by launching multiple containers.
Version Control: Containers can be versioned for reproducibility.

Why Use WSL:
WSL integration enhances performance and compatibility by providing a Linux-compatible environment on Windows.
It allows you to work with Docker commands directly from a Linux terminal, improving the Docker experience on Windows.

Steps to Create a Container:

Step 1: Install Docker Desktop
Download and install Docker Desktop from the official Docker website.
Follow the installation instructions for your operating system.

Step 2: Setup WSL
1. Install and Update WSL:
Open the command prompt (CMD) or PowerShell as an administrator.
Run the following command to install WSL 2: **wsl --install**
After the installation, reboot your system.
Once your system reboots, open the command prompt (CMD) or WSL terminal again.
2. Update WSL:
To ensure you have the latest version of WSL, run: **wsl –update**

3. Enable WSL Integration:
Open Docker Desktop settings.
Navigate to the "Resources" section and select "WSL Integration."
Enable the WSL integration for the desired WSL distribution.

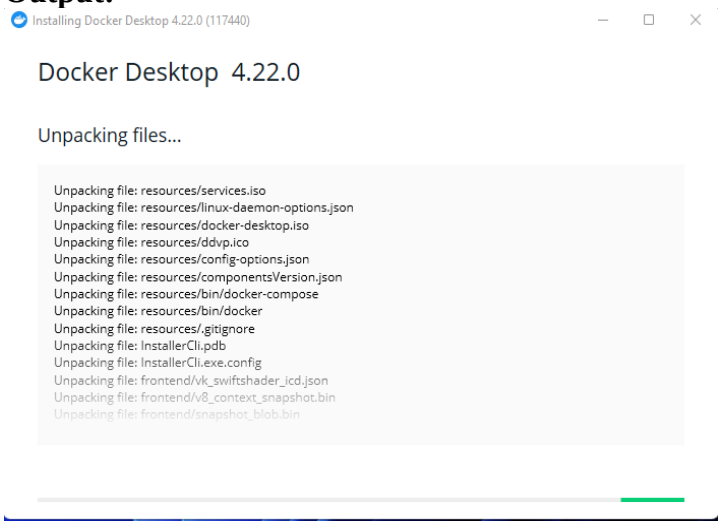
Step 3: Pull Image
Open the command prompt (CMD) or WSL terminal.
Run the following command to pull the "alpine" image from Docker Hub: docker pull alpine

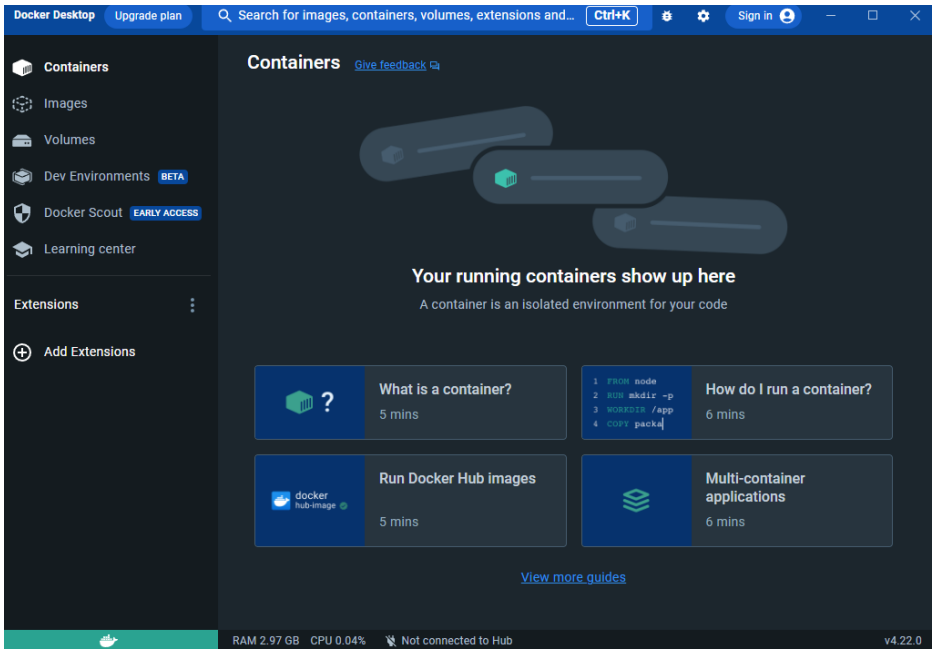
Step 4: Check Images
To verify that the "alpine" image has been downloaded, run: **docker images**

Step 5: Run Container
Create and enter an "alpine" container using the following command: **docker run -it alpine**
The -it flag allows you to interact with the container using the terminal.
This command starts a new instance of the "alpine" image and opens a terminal within the container.

Step 6: Explore the Container
You'll now be inside the "alpine" container's terminal.
You can explore the container, run commands, and interact with the isolated environment.
For example, you can try running commands like ls, echo, etc.

Output:

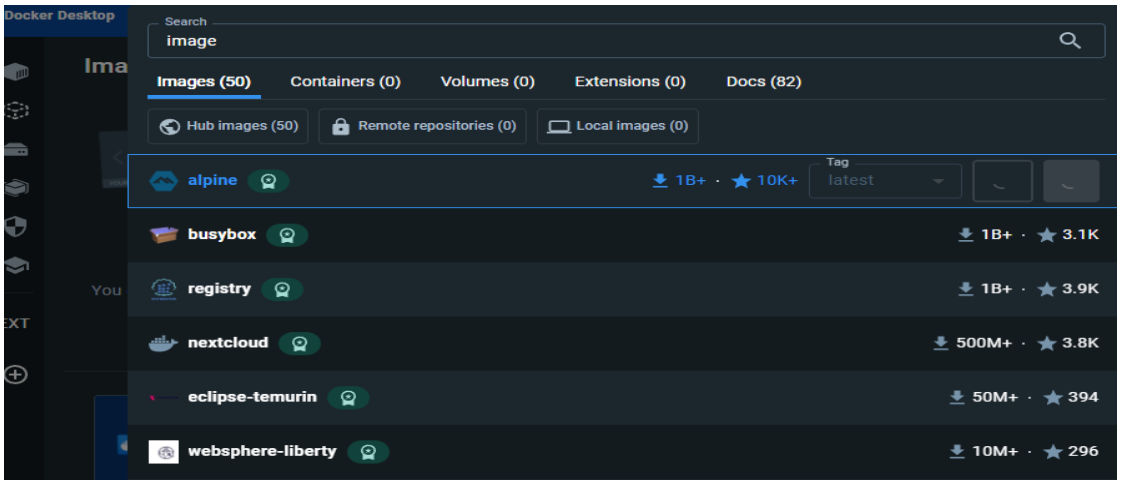


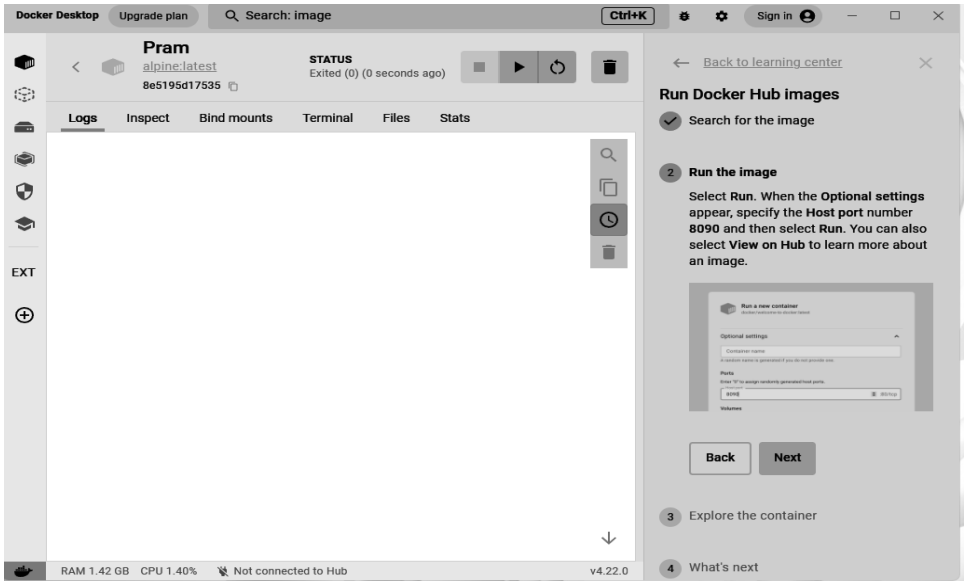



```
C:\Users\student>wsl --install
The requested operation requires elevation.
Installing: Windows Subsystem for Linux
Windows Subsystem for Linux has been installed.
Installing: Ubuntu
Ubuntu has been installed.
The requested operation is successful. Changes will not be effective until the system is rebooted.
```

```
C:\Users\student>wsl --version
WSL version: 1.2.5.0
Kernel version: 5.15.90.1
WSLg version: 1.0.51
MSRDC version: 1.2.3770
Direct3D version: 1.608.2-61064218
DXCore version: 10.0.25131.1002-220531-1700.rs-onecore-base2-hyp
Windows version: 10.0.22000.1455
```

```
C:\Users\Administrator>wsl --update
Installing: Windows Subsystem for Linux
Windows Subsystem for Linux has been installed.
```






```
C:\Users\Administrator>docker --version
Docker version 24.0.5, build ced0996

C:\Users\Administrator>docker images
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE

C:\Users\Administrator>docker pull alpine
Using default tag: latest
latest: Pulling from library/alpine
7264a8db6415: Pull complete
Digest: sha256:7144f7bab3d4c2648d7e59409f15ec52a18006a128c733fcff20d3a4a54ba44a
Status: Downloaded newer image for alpine:latest
docker.io/library/alpine:latest

What's Next?
  View summary of image vulnerabilities and recommendations → docker scout quickview alpine
```

```
C:\Users\Administrator>docker images
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
alpine       latest    7e01a0d0a1dc   10 days ago   7.34MB

C:\Users\Administrator>
```

```
C:\Users\Administrator>docker run -it alpine
/ # ls
bin    dev    etc    home   lib    media  mnt    opt    proc   root
/ # pwd
/
/ # cd home
/home #
```

Conclusion:

Leveraging Docker to run containers for different applications and OS offers a flexible and efficient way to manage and deploy software, enhancing development workflows and cross-platform compatibility.

Name: Sameer Shah

Department: Cyber Security

Div: BE-15

Roll No: 33

Subject: DSO

Experiment No. – 6				
Date of Performance:	19/08/2024			
Date of Submission:	26/08/2024			
Program Execution/ formation/ correction/ ethical practices (06)	Timely Submission (01)	Viva (03)	Experiment Total (10)	Sign with Date

Experiment No. 6

Aim: To create a custom Docker image using Docker files and upload it to the Docker hub.

Lab Outcome: Analyze & Illustrate the Containerization of OS images and deployment of applications over Docker

Theory:

Docker:
Docker is a platform that enables you to develop, ship, and run applications within containers, offering a lightweight and consistent environment that can host various applications and even different operating systems.

Features:
Containerization: Encapsulates applications and their dependencies.
Portability: Containers run consistently across various environments.
Efficiency: Shares host OS resources, minimizing overhead.
Isolation: Ensures applications don't interfere with each other.
Scalability: Easily scale applications by launching multiple containers.
Version Control: Containers can be versioned for reproducibility.

Container:
A container is a lightweight, standalone, and executable package that includes everything needed to run a piece of software, including the code, runtime, libraries, and system tools. Containers provide consistency and portability, making it easy to deploy applications across different environments.

Steps Followed:
1. **Install Docker:**
    - Download and install Docker Desktop from the official Docker website based on your operating system.
    - Follow the installation instructions to set up Docker.

2. **Download a Project from GitHub:**
    - Choose a project from GitHub that contains a Dockerfile, which defines the container environment.

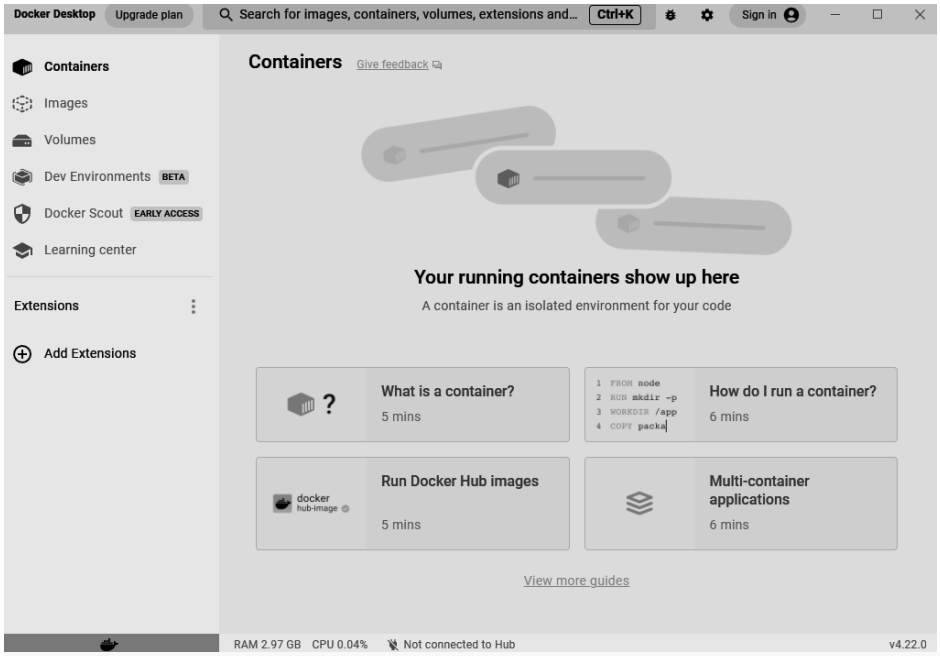
3. **Build a Docker Image:**
    - Open a terminal and navigate to the directory containing the Dockerfile.
    - Run the command to build a Docker image, e.g., **docker build -t welcome-to-docker.**
    - This command creates an image based on the instructions in the Dockerfile.

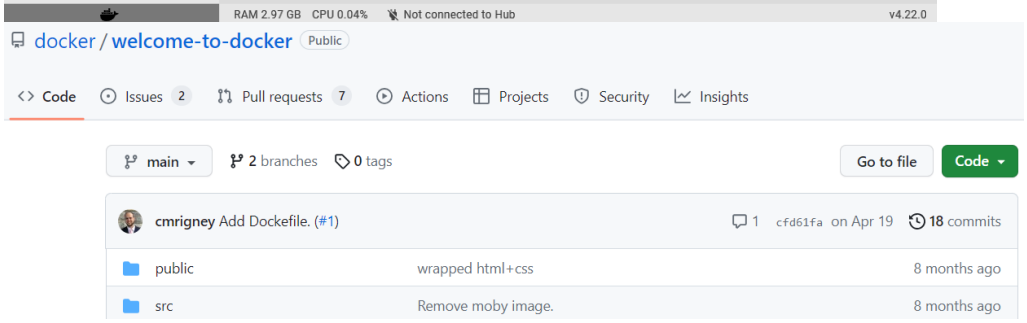
4. **Check the Built Image:**
    - Go to the Docker Desktop's Image tab to verify that the image has been successfully built.
5. **Run the Container:**
    - Start a container based on the image by specifying a container name and port.
    - Runs the application in the container and maps port 3000 on your local machine to port 80 in the container.

6. **Access the Application:**
    - Open a web browser and go to **localhost:3000**. You should see your application running in the container.

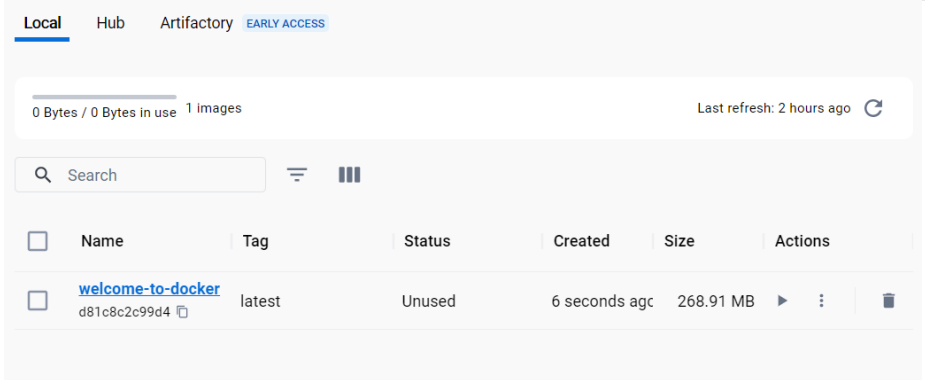
7. **Remove the Container:**
    - Once you're done, you can stop and remove the container by going to the Docker Desktop's Container tab.

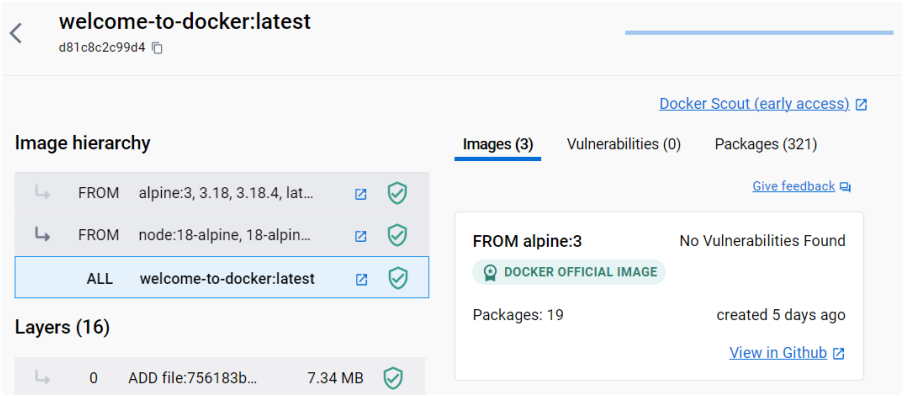
Output:

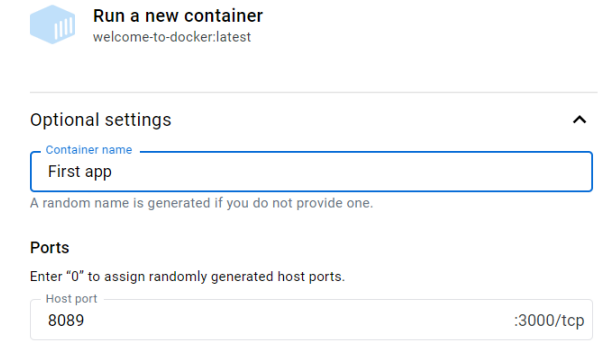


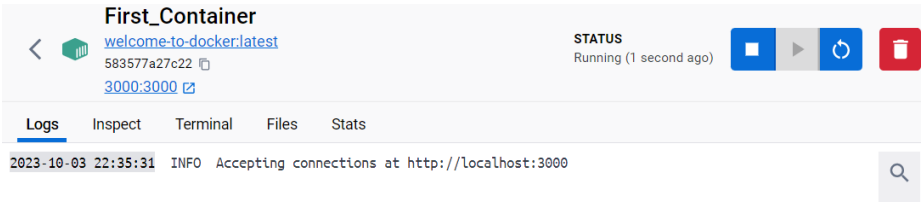


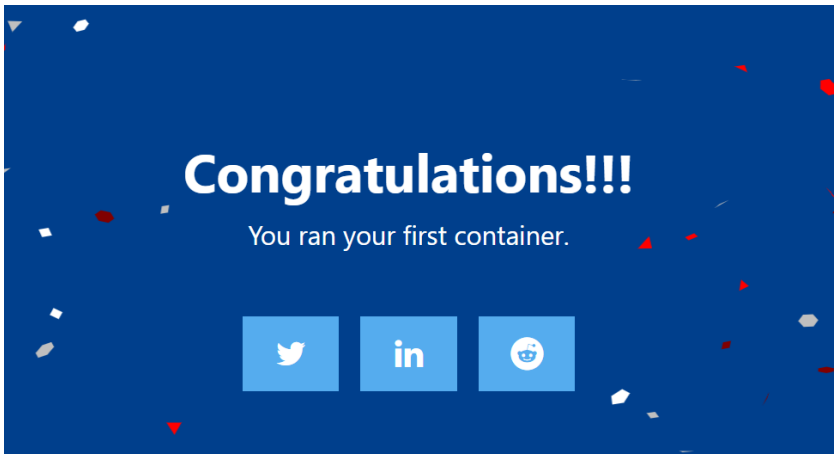

```
PS F:\CYSE\DevSecOps\Terraform> cd F:\CYSE\DevSecOps\Docker\welcome-to-docker-main
PS F:\CYSE\DevSecOps\Docker\welcome-to-docker-main> docker build -t welcome-to-docker .
[+] Building 4.6s (2/3)
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 647B
 => [internal] load .dockerignore
 => => transferring context: 52B
 => [internal] load metadata for docker.io/library/node:18-alpine
```

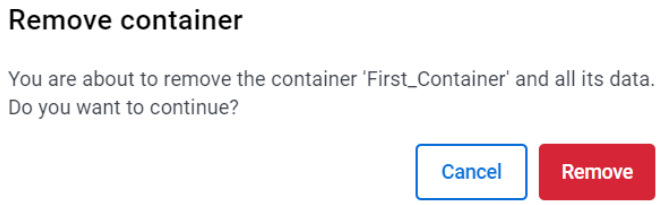



welcome-to-docker:latest
d81c8c2c99d4
Docker Scout (early access)
Image hierarchy
Images (3)
Vulnerabilities (0)
Packages (321)
FROM alpine:3, 3.18, 3.18.4, lat...
FROM node:18-alpine, 18-alpin...
ALL welcome-to-docker:latest
Give feedback
FROM alpine:3
No Vulnerabilities Found
DOCKER OFFICIAL IMAGE
Packages: 19
created 5 days ago
View in Github
Layers (16)
0 ADD file:756183b... 7.34 MB


Run a new container
welcome-to-docker:latest
Optional settings
Container name
First app
A random name is generated if you do not provide one.
Ports
Enter "0" to assign randomly generated host ports.
Host port
8089
:3000/tcp


First_Container
welcome-to-docker:latest
583577a27c22
3000:3000
STATUS
Running (1 second ago)
Logs
Inspect
Terminal
Files
Stats
2023-10-03 22:35:31 INFO Accepting connections at http://localhost:3000


Congratulations!!!
You ran your first container.


Remove container
You are about to remove the container 'First_Container' and all its data.
Do you want to continue?
Cancel
Remove

Conclusion:

In this experiment, we successfully utilized Docker to deploy containers running various applications and operating systems. Docker's containerization technology proved invaluable in achieving consistency and portability across different environments. By following the steps outlined, we gained practical experience in efficiently managing and running applications within containers, highlighting the advantages of this approach in modern software development and deployment.

Subject: DSO

Experiment No. – 7				
Date of Performance:	26/08/2024			
Date of Submission:	02/09/2024			
Program Execution/ formation/ correction/ ethical practices (06)	Timely Submission (01)	Viva (03)	Experiment Total (10)	Sign with Date

Experiment No. 7

Aim: To implement Application and code security testing using snyk.

Lab Outcome: Use Sonarqube and snyk to perform code quality checks and Threat Dragon to create threat models to identify threats in the system

Theory:

What is Snyk:

Snyk is a developer-first security platform that helps developers find and fix vulnerabilities in their code and open-source dependencies early in the development process. It scans code and dependencies for known vulnerabilities and offers remediation guidance. It provides tools for both application security (AppSec) and open-source security.

Why Use Snyk:

1. **Vulnerability Detection:** Snyk scans your code and dependencies to identify known vulnerabilities in libraries and frameworks.
2. **Continuous Monitoring:** It offers continuous monitoring, alerting you to new vulnerabilities as they are discovered.
3. **Integration:** Snyk integrates seamlessly into the development workflow, including CI/CD pipelines.
4. **Developer-Friendly:** Snyk is developer-friendly, providing actionable insights and fixes for vulnerabilities.
5. **Open-Source Security:** It helps secure open-source libraries, which are commonly used in software development.

Steps to Implement Security Testing Using Snyk:

1. **Go to Snyk Website:**
   Visit the Snyk website (https://snyk.io/).

2. **Create an Account:**
   Sign up for a Snyk account if you don't already have one.

3. **Choose Integration Method:**
   Decide how you want to integrate Snyk into your development workflow, such as using the Snyk CLI or integrations with your CI/CD pipeline.

4. **Install Snyk CLI:**
   If you choose to use the Snyk CLI, follow the installation instructions for your platform. For Windows, you can use the following command:
   curl https://static.snyk.io/cli/latest/snyk-win.exe -o snyk.exe

5. **Authenticate Your Machine:**
   Run the command **snyk auth** to authenticate your machine with your Snyk account.
6. **Scan for Security Issues:**

Before scanning, navigate to your project's directory.
Ensure Snyk Code is enabled in the Snyk settings.
Run the following command to scan your code for vulnerabilities:
snyk code test --org=04c00119-817c-4791-bc5a-a814134efa86

Output:

```
C:\Users\student>curl https://static.snyk.io/cli/latest/snyk-win.exe -o snyk.exe
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 83.0M  100 83.0M    0     0  27.5M      0  0:00:03  0:00:03 --:--:-- 27.5M

C:\Users\student>
```

Authenticated

Your account has been authenticated. Snyk is now ready to be used.

Authenticate your machine to associate it with your Snyk Account

```
snyk auth                                                    Copy
```

`snyk auth` Opens a browser window with prompts to log in to your Snyk account and authenticate your machine. No repository permissions are needed at this stage.

Thank you for authenticating your machine. You can now start scanning for issues.

```
Testing C:\Users\student\Downloads\projects-main ...

 ✗ [Medium] Cross-site Scripting (XSS)
   Path: script.js, line 27
   Info: Unsanitized input from data from a remote resource flows into innerHTML, where it is used to
 dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Script
ing attack (DOMXSS).

 ✗ [Medium] Cross-site Scripting (XSS)
   Path: script.js, line 85
   Info: Unsanitized input from data from a remote resource flows into innerHTML, where it is used to
 dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Script
ing attack (DOMXSS).

 ✗ [Medium] Cross-site Scripting (XSS)
   Path: script.js, line 90
   Info: Unsanitized input from data from a remote resource flows into innerHTML, where it is used to
 dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Script
ing attack (DOMXSS).

 ✗ [Medium] Cross-site Scripting (XSS)
   Path: script.js, line 95
   Info: Unsanitized input from data from a remote resource flows into innerHTML, where it is used to
 dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Script
ing attack (DOMXSS).
```

```
✓ Test completed

Organization:      04c00119-817c-4791-bc5a-a814134efa86
Test type:         Static code analysis
Project path:      C:\Users\student\Downloads\projects-main

Summary:

  7 Code issues found
  7 [Medium]


PS C:\Users\student\Downloads\projects-main>
```

Conclusion:

Implementing security testing using Snyk is crucial for identifying and fixing vulnerabilities in your applications and code early in the development process. By integrating Snyk into your workflow, you can enhance the security of your software and reduce the risk of deploying insecure code.

Name: Sameer Shah

Department: Cyber Security

Div: BE-15

Roll No: 33

Subject: DSO

Experiment No. – 8				
Date of Performance:	02/08/2024			
Date of Submission:	09/09/2024			
Program Execution/ formation/ correction/ ethical practices (06)	Timely Submission (01)	Viva (03)	Experiment Total (10)	Sign with Date

Experiment No. 7

Aim: To implement Application and code security testing using snyk.

Lab Outcome: Use Sonarqube and snyk to perform code quality checks and Threat Dragon to create threat models to identify threats in the system

Theory:

What is Snyk:

Snyk is a developer-first security platform that helps developers find and fix vulnerabilities in their code and open-source dependencies early in the development process. It scans code and dependencies for known vulnerabilities and offers remediation guidance. It provides tools for both application security (AppSec) and open-source security.

Why Use Snyk:

1. **Vulnerability Detection:** Snyk scans your code and dependencies to identify known vulnerabilities in libraries and frameworks.
2. **Continuous Monitoring:** It offers continuous monitoring, alerting you to new vulnerabilities as they are discovered.
3. **Integration:** Snyk integrates seamlessly into the development workflow, including CI/CD pipelines.
4. **Developer-Friendly:** Snyk is developer-friendly, providing actionable insights and fixes for vulnerabilities.
5. **Open-Source Security:** It helps secure open-source libraries, which are commonly used in software development.

Steps to Implement Security Testing Using Snyk:

1. **Go to Snyk Website:**
   Visit the Snyk website (https://snyk.io/).

2. **Create an Account:**
   Sign up for a Snyk account if you don't already have one.

3. **Choose Integration Method:**
   Decide how you want to integrate Snyk into your development workflow, such as using the Snyk CLI or integrations with your CI/CD pipeline.

4. **Install Snyk CLI:**
   If you choose to use the Snyk CLI, follow the installation instructions for your platform.
   For Windows, you can use the following command:
   curl https://static.snyk.io/cli/latest/snyk-win.exe -o snyk.exe

5. **Authenticate Your Machine:**
   Run the command **snyk auth** to authenticate your machine with your Snyk account.
6. **Scan for Security Issues:**
   Before scanning, navigate to your project's directory.
   Ensure Snyk Code is enabled in the Snyk settings.
   Run the following command to scan your code for vulnerabilities:
   snyk code test --org=04c00119-817c-4791-bc5a-a814134efa86

Output:

```
C:\Users\student>curl https://static.snyk.io/cli/latest/snyk-win.exe -o snyk.exe
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 83.0M  100 83.0M    0     0  27.5M      0  0:00:03  0:00:03 --:--:-- 27.5M

C:\Users\student>
```

Authenticated

Your account has been authenticated. Snyk is now ready to be used.

Authenticate your machine to associate it with your Snyk Account

```
snyk auth                                                    Copy
```

`snyk auth` Opens a browser window with prompts to log in to your Snyk account and authenticate your machine. No repository permissions are needed at this stage.

Thank you for authenticating your machine. You can now start scanning for issues.

```
Testing C:\Users\student\Downloads\projects-main ...

 ✗ [Medium] Cross-site Scripting (XSS)
   Path: script.js, line 27
   Info: Unsanitized input from data from a remote resource flows into innerHTML, where it is used to
 dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Script
ing attack (DOMXSS).

 ✗ [Medium] Cross-site Scripting (XSS)
   Path: script.js, line 85
   Info: Unsanitized input from data from a remote resource flows into innerHTML, where it is used to
 dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Script
ing attack (DOMXSS).

 ✗ [Medium] Cross-site Scripting (XSS)
   Path: script.js, line 90
   Info: Unsanitized input from data from a remote resource flows into innerHTML, where it is used to
 dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Script
ing attack (DOMXSS).

 ✗ [Medium] Cross-site Scripting (XSS)
   Path: script.js, line 95
   Info: Unsanitized input from data from a remote resource flows into innerHTML, where it is used to
 dynamically construct the HTML page on client side. This may result in a DOM Based Cross-Site Script
ing attack (DOMXSS).
```

Conclusion:

Implementing security testing using Snyk is crucial for identifying and fixing vulnerabilities in your applications and code early in the development process. By integrating Snyk into your workflow, you can enhance the security of your software and reduce the risk of deploying insecure code.

Name: Sameer Shah

Department: Cyber Security

Div: BE-15

Roll No: 33

Subject: DSO

Experiment No. – 9				
Date of Performance:	09/09/2024			
Date of Submission:	16/09/2024			
Program Execution/ formation/ correction/ ethical practices (06)	Timely Submission (01)	Viva (03)	Experiment Total (10)	Sign with Date

Experiment No. 9

Aim: To implement threat models to identify threats in the system using Threat Dragon.

Lab Outcome: Use Sonarqube and snyk to perform code quality checks and Threat Dragon to create threat models to identify threats in the system

Theory:

Threat Dragon: Threat Dragon is an open-source threat modeling tool used to identify and mitigate security threats in software systems. Threat modeling is a systematic approach to identifying potential security risks and vulnerabilities in a software system during the design phase. Threat Dragon helps teams visualize and analyze these threats, allowing for better risk management and security enhancement.

Why Use Threat Dragon:

- **Security Analysis:** Threat Dragon allows you to systematically identify and assess security threats and vulnerabilities.
- **Collaboration:** It facilitates collaboration among team members by providing a central platform for threat modeling.
- **Visual Representation:** Threat models are often represented as diagrams, making it easier to understand and communicate security risks.

Steps to Implement Threat Models Using Threat Dragon:

1. **Access Threat Dragon:**
   Go to the Threat Dragon website or install the Threat Dragon application locally if available.

2. **Create a New Project:**
   Start by creating a new project, giving it a meaningful name like "Demo Threat Model."
   Fill in project details such as title, owner, system description, contributors, and reviewers.

3. **Create Diagram:**
   Using Threat Dragon's diagramming tools, create a flowchart or diagram that represents the system's architecture and data flows.

4. **Identify Threats:**
   Collaborate with your team to identify potential threats and vulnerabilities within the system based on the diagram and system description.

5. **Rate and Prioritize Threats:**
   Rate the identified threats based on severity and impact.
   Prioritize them to address the most critical threats first.

6. **Provide Mitigation Strategies:**
   For each threat, define mitigation strategies or security controls that can be implemented to mitigate the risk.

Output:

OWASP Threat Dragon

OWASP Threat Dragon is a free, open-source, cross-platform application for creating threat models. Use it to draw threat modeling diagrams and to identify threats for your system. With an emphasis on flexibility and simplicity it is easily accessible for all types of users.

Login with GitHub | Login to Local Session

Editing: Demo Threat Model

Title

Demo Threat Model

Owner

Het Chheda

High level system description

A sample model of a web application, with a queue-decoupled background process.

Contributors

Aditya Patel × Start typing to add a contributor

Diagrams

Main Request Data Flow

Demo Threat Model

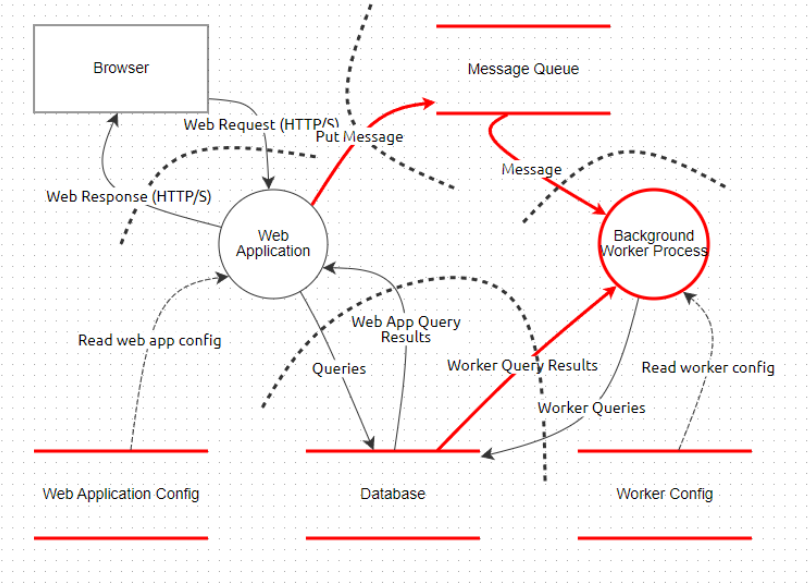
Owner:
Het Chheda

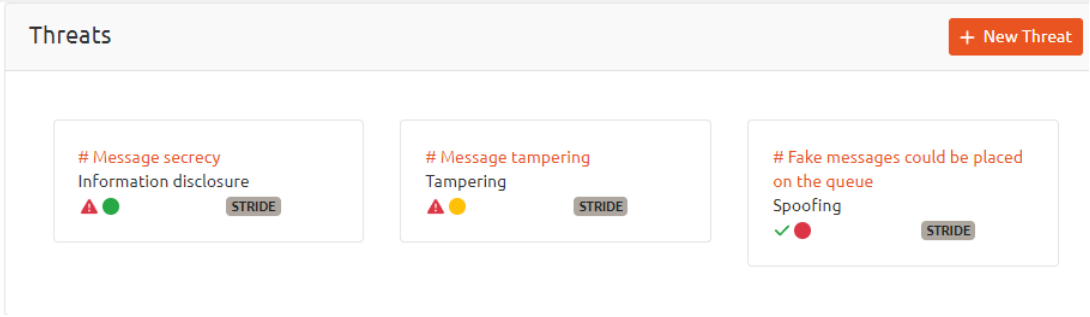
Reviewer:

Contributors:
Aditya Patel

High level system description

A sample model of a web application, with a queue-decoupled background process.





Conclusion:

Implementing threat models with Threat Dragon is a proactive approach to enhancing software security. It allows teams to visualize, identify, prioritize, and mitigate potential threats early in the development process, resulting in a more secure and robust software system.

Name: Kanav Tikone Department: Cyber Security

Div: BE-15 Roll No: 44

Subject: DSO

Experiment No. – 10				
Date of Performance:	16/09/2024			
Date of Submission:	07/10/2024			
Program Execution/ formation/ correction/ ethical practices (06)	Timely Submission (01)	Viva (03)	Experiment Total (10)	Sign with Date

Experiment No. 10

Aim: To implement terraform script for deploying compute/Storage/network infrastructure on the public cloud platform (GCP / AWS / Azure).

Lab Outcome: Implement Terraform scripts to manage VMs on a cloud.

Theory:

Terraform: Terraform is an open-source infrastructure as code (IAC) tool by HashiCorp that allows you to define and provision infrastructure using a declarative configuration language. It provides a consistent way to create, modify, and manage infrastructure across various cloud providers and on-premises environments.

Why Use Terraform:

- **Infrastructure as Code:** Terraform allows you to define your infrastructure as code, making it version able, maintainable, and reproducible.
- **Multi-Cloud Support:** It supports multiple cloud providers, enabling you to manage infrastructure consistently across AWS, Azure, Google Cloud, and others.
- **Resource Management:** Terraform can manage a wide range of resources, including virtual machines, networks, storage, and more.
- **Modularity:** It encourages modular code and reusability, making it easier to manage complex infrastructure.

Why Terraform with Cloud Platforms:

- Using Terraform with cloud platforms like AWS provides a structured and automated approach to provisioning and managing resources.
- It helps ensure infrastructure consistency and scalability.
- Terraform's modular approach allows you to manage complex cloud environments efficiently.

AWS (Amazon Web Services): AWS is a leading cloud services provider offering a wide range of cloud computing services, including computing power, storage, databases, machine learning, analytics, and more. It's widely used for hosting applications, websites, and managing various cloud resources.

Steps Followed:

1. **Installation of Terraform:**
    - Download Terraform from the HashiCorp website.
    - Add the Terraform binary to your system's PATH.
    - Verify the installation using **terraform --version**.

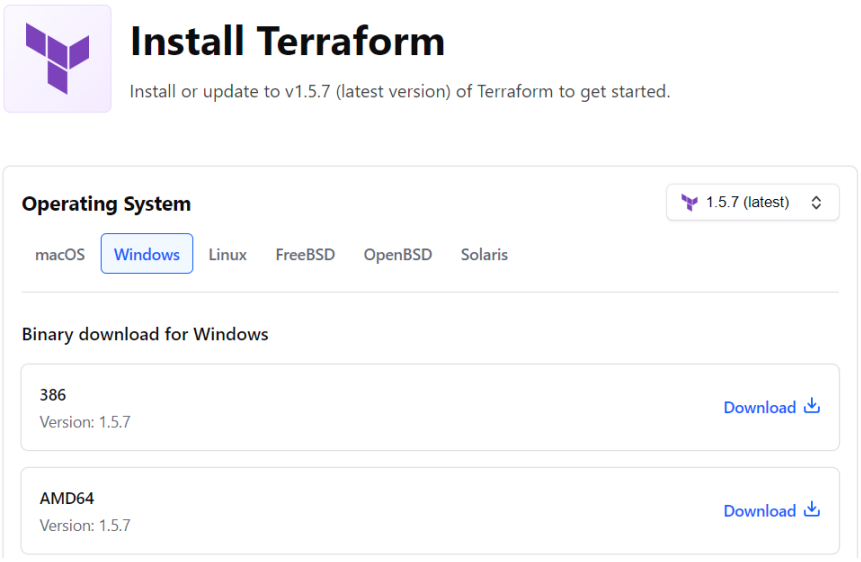
2. **AWS Account Setup:**
    - Sign up for an AWS account as the root user.
    - Provide personal and payment details to complete the registration.

3. **Add IAM User:**
    - Log in to the AWS console as the root user.
    - Go to the Identity and Access Management (IAM) service and create a new IAM user.
    - Configure permissions

- Login as IAM user
- Create an access key for the IAM user.
- Save the access key ID and secret access key for later use.

4. **Create Instance Using Terraform Script:**
    - Install Visual Studio Code or any other code editor.
    - Create a folder for your Terraform project.
    - Create a **main.tf** file and write a Terraform script to define your infrastructure resources (e.g., EC2 instance, VPC, security groups).
    - Run the following Terraform commands:
        - **terraform init**: Initializes the project and downloads necessary providers.
        - **terraform plan**: Shows the execution plan without making changes.
        - **terraform apply**: Creates the AWS resources based on your Terraform configuration.
        - **terraform destroy**: Destroys the created resources when no longer needed.

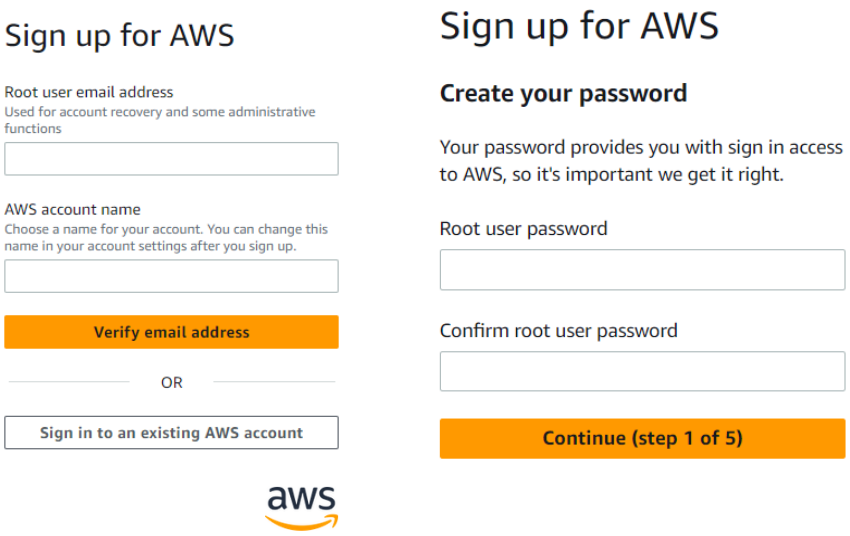
Output:
Installation of Terraform:

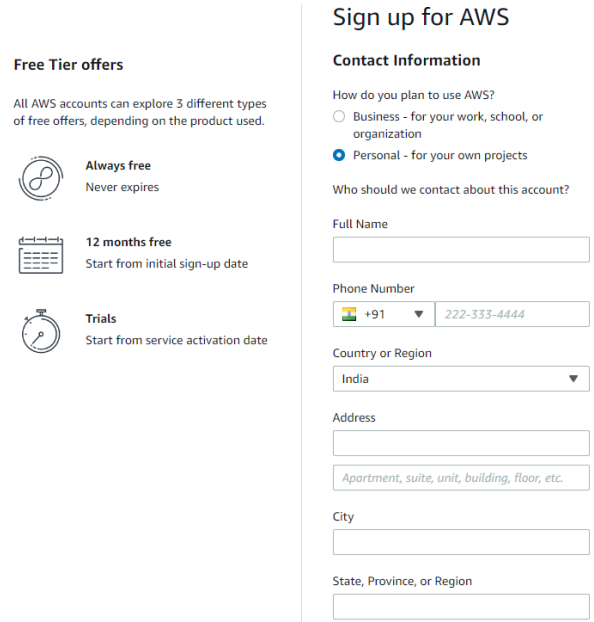


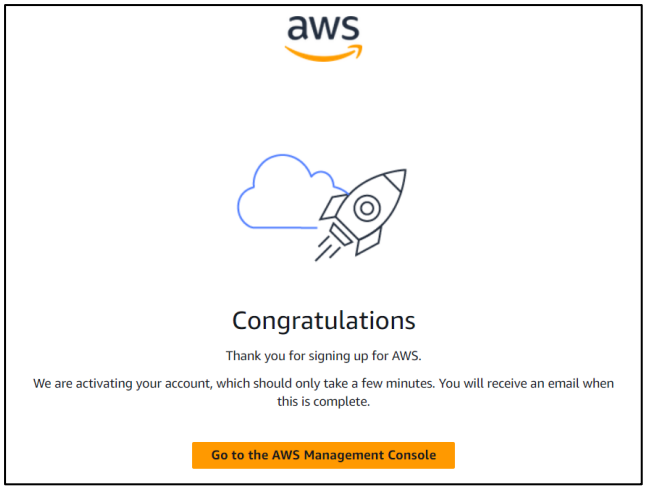

```
PS F:\CYSE\DevSecOps\Terraform> terraform --version
Terraform v1.5.7
on windows_amd64
PS F:\CYSE\DevSecOps\Terraform>
```

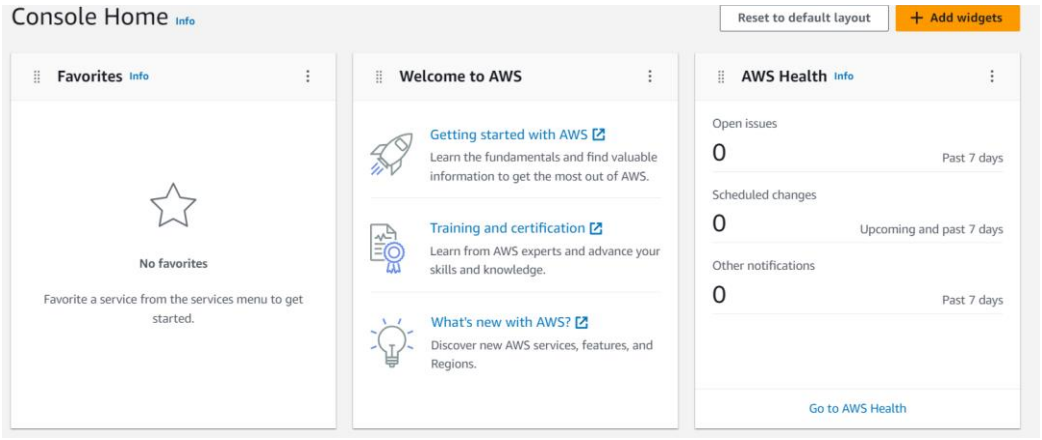
```
PS F:\CYSE\DevSecOps\Terraform> terraform
Usage: terraform [global options] <subcommand> [args]

The available commands for execution are listed below.
The primary workflow commands are given first, followed by
less common or more advanced commands.

Main commands:
  init          Prepare your working directory for other commands
  validate      Check whether the configuration is valid
  plan          Show changes required by the current configuration
  apply         Create or update infrastructure
  destroy       Destroy previously-created infrastructure

All other commands:
  console       Try Terraform expressions at an interactive command prompt
  fmt           Reformat your configuration in the standard style
  force-unlock  Release a stuck lock on the current workspace
  get           Install or upgrade remote Terraform modules
  graph         Generate a Graphviz graph of the steps in an operation
```

AWS Account Setup:

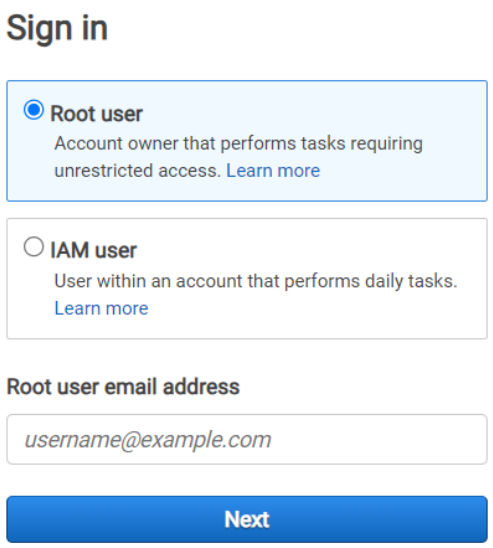


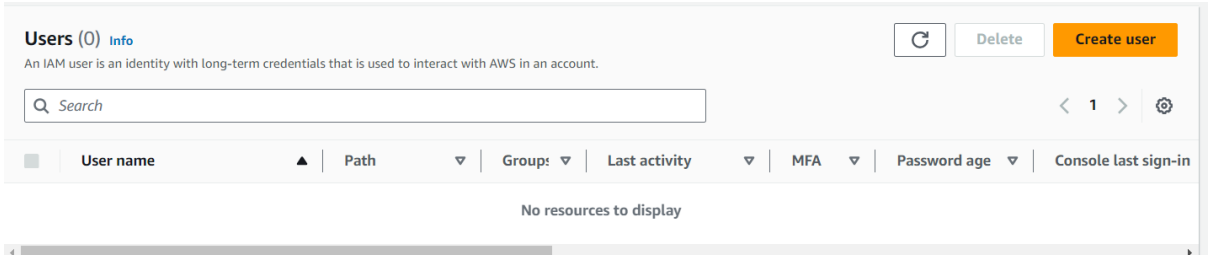


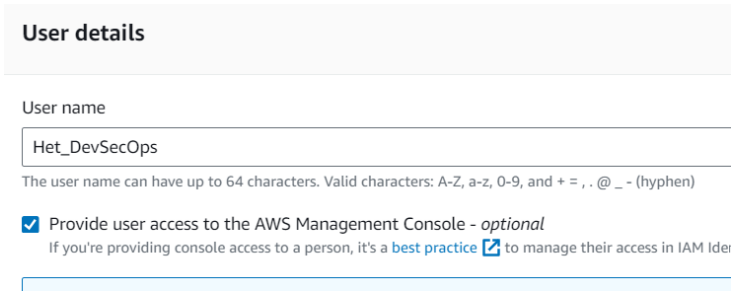

aws
Congratulations
Thank you for signing up for AWS.
We are activating your account, which should only take a few minutes. You will receive an email when this is complete.
Go to the AWS Management Console

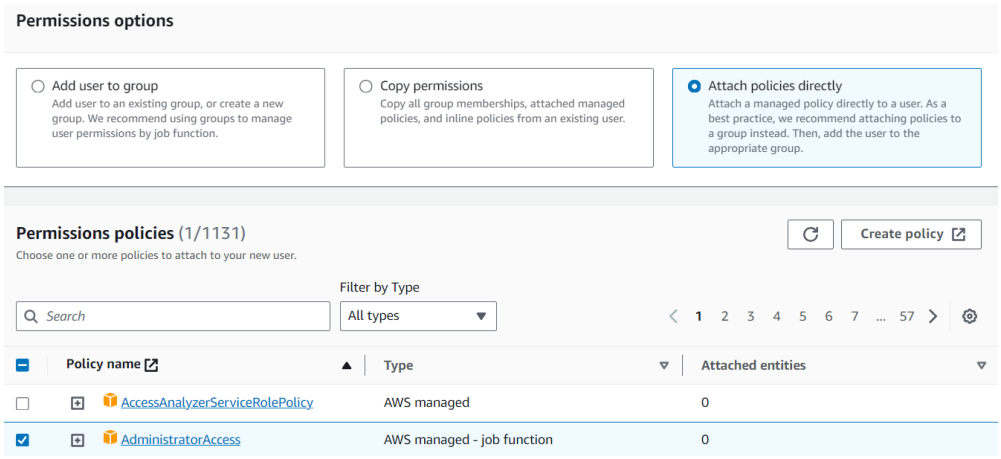

Console Home Info
Reset to default layout
+ Add widgets
Favorites Info
No favorites
Favorite a service from the services menu to get started.
Welcome to AWS
Getting started with AWS
Learn the fundamentals and find valuable information to get the most out of AWS.
Training and certification
Learn from AWS experts and advance your skills and knowledge.
What's new with AWS?
Discover new AWS services, features, and Regions.
AWS Health Info
Open issues
0
Past 7 days
Scheduled changes
0
Upcoming and past 7 days
Other notifications
0
Past 7 days
Go to AWS Health

Add IAM User:

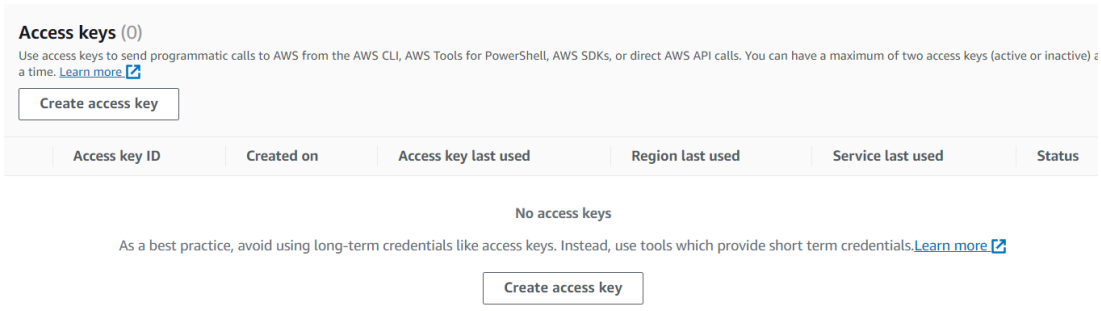

Sign in
Root user
Account owner that performs tasks requiring unrestricted access. Learn more
IAM user
User within an account that performs daily tasks.
Learn more
Root user email address
username@example.com
Next


Users (0) Info
An IAM user is an identity with long-term credentials that is used to interact with AWS in an account.
Delete
Create user
Search
1
User name
Path
Groups
Last activity
MFA
Password age
Console last sign-in
No resources to display


User details
User name
Het_DevSecOps
The user name can have up to 64 characters. Valid characters: A-Z, a-z, 0-9, and + = , . @ _ - (hyphen)
Provide user access to the AWS Management Console - optional
If you're providing console access to a person, it's a best practice to manage their access in IAM Ide


Permissions options
Add user to group
Add user to an existing group, or create a new group. We recommend using groups to manage user permissions by job function.
Copy permissions
Copy all group memberships, attached managed policies, and inline policies from an existing user.
Attach policies directly
Attach a managed policy directly to a user. As a best practice, we recommend attaching policies to a group instead. Then, add the user to the appropriate group.
Permissions policies (1/1131)
Choose one or more policies to attach to your new user.
Create policy
Filter by Type
Search
All types
1 2 3 4 5 6 7 ... 57
Policy name
Type
Attached entities
AccessAnalyzerServiceRolePolicy
AWS managed
0
AdministratorAccess
AWS managed - job function
0


Sign in as IAM user
IAM user name
Het_DevSecOps
Password
Remember this account
Sign in


Access keys (0)
Use access keys to send programmatic calls to AWS from the AWS CLI, AWS Tools for PowerShell, AWS SDKs, or direct AWS API calls. You can have a maximum of two access keys (active or inactive) a
a time. Learn more
Create access key
Access key ID
Created on
Access key last used
Region last used
Service last used
Status
No access keys
As a best practice, avoid using long-term credentials like access keys. Instead, use tools which provide short term credentials. Learn more
Create access key

Create Instance Using Terraform Script:

Terraform > main.tf > ...

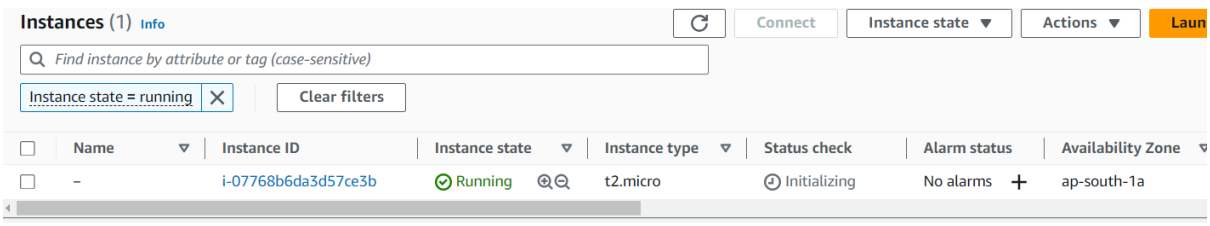
```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 5.0"
    }
  }
}

provider "aws" {
  region  = "ap-south-1"
  access_key = "..................."
  secret_key = "..................."
}

resource "aws_instance" "Windows_Terraformr" {
  ami           = "ami-08abb3eeacc61972d"
  instance_type = "t2.micro"
}

```

```
PS F:\CYSE\DevSecOps\Terraform> terraform init

Initializing the backend...

Initializing provider plugins...
- Finding hashicorp/aws versions matching "~> 5.0"...
- Installing hashicorp/aws v5.19.0...
- Installed hashicorp/aws v5.19.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

```
PS F:\CYSE\DevSecOps\Terraform> terraform plan

Terraform used the selected providers to generate the following ex
  + create

Terraform will perform the following actions:

  # aws_instance.Windows_Terraformr will be created
  + resource "aws_instance" "Windows_Terraformr" {
      + ami                                  = "ami-08abb3eeacc619
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = (known after apply)
      + cpu_core_count                       = (known after apply)
      + cpu_threads_per_core                 = (known after apply)
      + disable_api_stop                     = (known after apply)
      + disable_api_termination              = (known after apply)
      + ebs_optimized                        = (known after apply)
      + get_password_data                    = false
      + host_id                              = (known after apply)
      + host_resource_group_arn              = (known after apply)
```

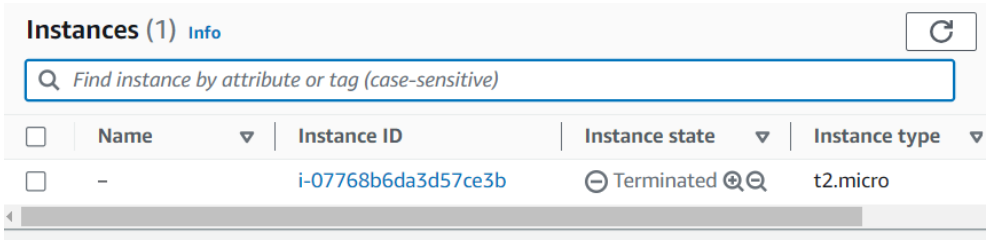
```
PS F:\CYSE\DevSecOps\Terraform> terraform apply

Terraform used the selected providers to generate the following exec
  + create

Terraform will perform the following actions:

  # aws_instance.Windows_Terraformr will be created
  + resource "aws_instance" "Windows_Terraformr" {
      + ami                                  = "ami-08abb3eeacc61972
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = (known after apply)
      + cpu_core_count                       = (known after apply)
      + cpu_threads_per_core                 = (known after apply)
      + disable_api_stop                     = (known after apply)
      + disable_api_termination              = (known after apply)
      + ebs_optimized                        = (known after apply)
      + get_password_data                    = false
      + host_id                              = (known after apply)
      + host_resource_group_arn              = (known after apply)
```

```
Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_instance.Windows_Terraformr: Creating...
aws_instance.Windows_Terraformr: Still creating... [10s elapsed]
aws_instance.Windows_Terraformr: Still creating... [20s elapsed]
aws_instance.Windows_Terraformr: Still creating... [30s elapsed]
aws_instance.Windows_Terraformr: Still creating... [40s elapsed]
aws_instance.Windows_Terraformr: Creation complete after 42s [id=

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
```



```
PS F:\CYSE\DevSecOps\Terraform> terraform destroy
aws_instance.Windows_Terraformr: Refreshing state.

Terraform used the selected providers to generate
  - destroy

Terraform will perform the following actions:

  # aws_instance.Windows_Terraformr will be destro
  - resource "aws_instance" "Windows_Terraformr" {
```

```
Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastru
  There is no undo. Only 'yes' will be accepted to

  Enter a value: yes

aws_instance.Windows_Terraformr: Destroying... [id=
aws_instance.Windows_Terraformr: Still destroying..
aws_instance.Windows_Terraformr: Still destroying..
aws_instance.Windows_Terraformr: Still destroying..
aws_instance.Windows_Terraformr: Destruction comple

Destroy complete! Resources: 1 destroyed.
PS F:\CYSE\DevSecOps\Terraform>
```



Conclusion:

In this experiment, we successfully utilized Terraform to deploy an AWS EC2 instance using infrastructure-as-code (IAC) scripts. Terraform's declarative approach allowed us to define and provision cloud resources with precision and repeatability. This streamlined method enhances the management of cloud infrastructure, making it easier to scale, version, and maintain AWS resources.