**Jenkins Setup**

1. Java 17 download
2. Copy path location
3. Edit environment variables, add the java path to system variables path (as a new path file)
4. Check version -> java –version
5. Download jenkins stable version
6. Localhost 8080
7. Search localhost:8080
8. Password is found at Jenkins in programdata
9. Install suggested plugins
10. Username- Amish

Password- Amish1801

Full name- Amish Talekar

url- <http://localhost:8080/>

**Exp1** Demonstrate use of git commands to push and pull 3 html files from Github account

**Push**

1. Create a folder and add 3 blank files
2. Git init
3. Git status
4. Git add .
5. Git status
6. git commit -m "first commit"
7. add creds
8. git commit -m "first commit"
9. git branch -M main
10. git push -u origin main

**Pull**

1. git pull origin main

**Exp2**(Add 2 using Java in jenkins)

1. Open vscode
2. Save the file name as AdditionOfTwoNumbers.java
3. Create a code-

public class AdditionOfTwoNumbers {

public static void main(String[] args) {

int num1 = 5;

int num2 = 7;

int sum = num1 + num2;

System.out.println("The sum of " + num1 + " and " + num2 + " is: " + sum);

}

}

1. Create new
2. Advanced-> use custom workspace-> paste java file location
3. Build steps-> windows batch command-> code is below

javac AdditionOfTwoNumbers.java

java add

**Exp7** Create an image of php project and push on Dockerhub repository.

1. Download docker-> create acc
2. Create 2 folder SimplePHPAPP and docker file
3. Php file code-

<html>

<body>

<h1><?php echo "Hello, Docker!"; ?></h1>

</body>

</html>

1. Docker file code-

FROM php:7.4-cli

COPY ./ ./

EXPOSE 3000

CMD ["php", "-S", "0.0.0.0:3000"]

1. Run new terminal
2. Enter code

docker build . -t Amish1801/simplephpapp

1. Then,

docker run --name simple-php-app -p 3002:3000 Amish1801/simplephpapp

1. Search in google - <http://localhost:3002/>

**Exp9** Demonstrate use of docker commands to pull ubuntu official image, create a file and push updated image on Dockerhub.

1. docker pull ubuntu
2. docker run -it --name my-ubuntu ubuntu
3. echo "Hello, Docker!" > example.txt
4. echo "Additional text" >> example.txt
5. exit
6. docker commit my-ubuntu my-ubuntu-modified
7. create a repository in docker website
8. docker tag my-ubuntu-modified Amish1801/dockerexp7:latest
9. docker login
10. docker push Amish1801/dockerexp7:latest

**Exp6** Write a Selenium script to perform automated testing

1. install python
2. install selenium library

cmd code-> pip install selenium

1. pip install selenium
2. vscode->

from selenium import webdriver

from selenium.webdriver.common.keys import Keys

from selenium.webdriver.common.by import By

import time

driver = webdriver.Chrome()

driver.get("http://www.google.com")

search\_box = driver.find\_element(By.NAME, "q")

search\_box.send\_keys("javatpoint")

search\_box.send\_keys(Keys.ENTER)

time.sleep(30)

driver.close()

print("Sample test Successful")

**Exp3** Demonstrate Continuous Integration process in Jenkins. Build a java program every 2 minutes (addition of two numbers) residing in Github repository.

**Install necessary plugins:**

* Go to Jenkins dashboard.
* Navigate to "Manage Jenkins" > "Manage Plugins".
* Install the following plugins:
  + GitHub plugin: For integrating Jenkins with GitHub.
  + JDK Tool Plugin: For configuring JDK installations in Jenkins.
  + Pipeline Plugin: For defining Jenkins pipelines in code (optional, but useful for more advanced CI/CD workflows).

**3. Configure Jenkins to connect to your GitHub repository:**

* Go to Jenkins dashboard.
* Navigate to "Manage Jenkins" > "Configure System".
* Scroll down to the "GitHub" section.
* Add your GitHub credentials.
* Save the configuration.

**4. Create a Jenkins job to build the Java program:**

* Click on "New Item" on the Jenkins dashboard.
* Enter a name for your job (e.g., "Java Program Build").
* Select "Freestyle project" and click "OK".
* Under the "Source Code Management" section, select "Git".
* Enter the URL of your GitHub repository.
* Specify the branch (e.g., **main** or **master**).
* Scroll down to the "Build" section.
* Add a build step to compile your Java program. For example, you can use an "Execute shell" build step and run **javac YourJavaFile.java**.
* Save the job configuration.

**5. Schedule the job to run every 2 minutes:**

* Open the job configuration page.
* Scroll down to the "Build Triggers" section.
* Check the "Build periodically" option.
* Enter the cron expression **\*/2 \* \* \* \*** to run the job every 2 minutes.
* Save the configuration.