DEVOPS EXPERIMENTS STEPS

Experiment 1: Write a program to perform various GIT operations on local and Remote repositories using Git and GitHub.

Steps:

1. Open Gitbash from your location
2. Create a Folder in D drive “ACE” and create a text file name “HelloAce.txt”.
3. Perform the following commands:
   - git config --global user.name “abc”
   - git config --global user.email abc.@gmail.com
   - git config –global –list
4. Now move to that location where the folder is created (cd d:\ACE)
5. Create a Repostories and give the name “ACE” and keep it public
6. Perform the following commands:
   - git init
   - git add .
   - git status
   - git remote add origin https://github.com/Harshit07Kadam/ACE51.git (copy the link from the repository create and paste here.)
   - git remote show origin
   - git add HelloAce.txt
   - git commit -m “Hello This is a new program”
   - git push origin master
   - git pull orgin master
   - git pull origin https://github.com/Harshit07Kadam/ACE51.git (copy the link from the repository create and paste here.)
   - git pull https://github.com/Harshit07Kadam/ACE51.git (copy the link from the repository create and paste here.)
7. Now open the repository in GitHub and see the file is add with the text you enter above.

Experiment 2: Write a program to Create Branch on Git and perform Merge operation. (Virtual Lab Experiment)

Steps:

1. Open Github -> Login -> Create a repository with same name of folder.
2. Create a folder in D drive with same name as repository
3. Open gitbash from that folder and perform the following commands:

- git config --global user.name “abc”
- git config --global user.email abc.@gmail.com
- git config –global –list
- cd d\:folder_name
- git init
- git status
- git remote add origin https://github.com/Harshit07Kadam/ACE51.git (copy the link from the repository create and paste here.)
- git remote show origin

- git add Final.txt
- git commit -m “Hello this is a experiment 4 code”
- git status
- git branch branch1
- git checkout branch
- git push -u origin branch1
- git checkout master
- git merge branch1
- git push origin master
- git branch -a
- git branch -d branch1
- git branch -a

4. Check the Github the folder is added to the repository and check the branches the new branch is added.

Experiment 3: Write a program on Jenkins to understand Jenkins Master Slave architecture and perform simple java program to show the current system date and time on Jenkins.

Steps:

1. Download Jenkins.war file from the websites.
2. Open cmd from download and write this command: java -jar Jenkins.war
3. Let the command run and see for a password which has a number (eg. 4482379285074908) and copy that password
4. Open browser and type localhost:8080
5. When done paste the password and save as admin, install plugins and open jenkins.
6. Now click on “New Item” and give a name and click on next. Give a description.
7. Add Steps as Windows command bash and write this code => echo “Hello World:%date%:%time%”
8. Click on Save and Click on Build Now it will display as #1 with green tick.
9. Click on Console Output and the code will be display with current time and date.

Experiment 4: Do the method on Jenkins to integrate GitHub and Jenkins.

Steps:

1. Jenkins is already installed go to browser and type localhost:8080
2. Click on “New Item” and give a name and click on next. Click on Source Code Management which is on left side.
3. Click on Branch Specifier as */master or */main as per your github repository you have.
4. In Source code Management Select Git and give repository URL as per your repo. Keep credentials as none.
5. In Build Triggers as GitHub hook trigger for GITScm polling
6. Click on Save and Click on Build now and check the Console Output.

Experiment 5: Execute docker commands to print hello world and version and to manage images and interact with containers. (push and pull image).

Steps:

1. First go to browser, type “docker download” and open the first link -> from nav bar hover on developers -> click on documentation -> again in nav bar click on Manual -> on left side click on docker desktop drop down menu and click on install and select your OS -> for windows install the x86_64 version.
2. After downloading, install the exe file. Now again browser, type “Docker hub” and click on first link. Sign up and login. Whenever you do docker experiment keep the docker hub open with login.
3. Now Go to Docker desktop and “Run as administrator” click on yes and then open docker desktop. The docker engine will start. Now minimize the docker desktop and open cmd. Type the code “docker run hello-world” you will get a message having first line “Hello from docker”. The first part of this experiment is done.
4. For the second part, again open docker desktop, Login to docker hub. Create a repository in docker by click on create repository and give it a name and keep the repo private and create
5. Open the cmd , and type the command “docker login” this will load and display docker login successfully. If not it will ask username and password which is same of the docker hub one.
6. Type the command => docker images which will show existing images created on dockers.
7. Type the command => docker tag hello-world:latest harshit/private-repo1:tag1 (here docker tag => command to create a new tag, hello-world => is the name of docker images, :latest => specifies the tag if the image, harshit/private-repo1 => it is the name of the user/repo name, tag1 => it is the new tag name). It can be check by docker images command.
8. Type the command
   - docker push harshit/private-repo1
   - docker push harshit/private-repo1:tag1 (if successfully execute it will give message pushed)
   - docker rmi harshit/private-repo1:tag1
   - docker pull harshit.private-repo1:tag1

Experiment 6: Build an image for a sample Java application using Dockerfile.

Steps:

1. Create a folder on desktop and create two text files in that folder with names “HelloWorld.java” which is a java file and other “Dockerfile” with no extension ie) system file type. (For newer version, by giving no extension doesn’t work. So go to notepad and click on save as and write the file name in double quotes “” which will create a system file.)
2. Now write the following code in java file :

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

3. Same for docker file write the below code:

FROM OpenJDK ://

COPY ./usr/src/app/

WORKDIR /usr/src/app

RUN javac HelloWorld.java

CMD [“java”,”hello”]

4. Now open Docker desktop, docker hub and open cmd from the folder location. Type docker login.
5. Type the command => docker images
6. Type the command as it is => docker build -t java:dd . (if there is no error in the code the code will execute successfully)
7. Again type => docker images
8. And finally type the command => docker run java:dd (which will give the output as Hello World!!)

Experiment 7: Build an image for a sample Web application using Dockerfile.

1. Create a folder on desktop and create two text files in that folder with names “Index.html” which is a html file and other “Dockerfile” with no extension ie) system file type. (For newer version, by giving no extension doesn’t work. So go to notepad and click on save as and write the file name in double quotes “” which will create a system file.)
2. Now write the following code in html file :

```
<Doc type html >
<html>
<head>
        <title> WELCOME TO PAGE</title>
</head>
<body>
        <h1> WELCOME TO THE WEBPAGE</h1>
        <p>
                <img src
="https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.pexels.com%2Fsearch%2Fbeautiful%2520nature%2F&psig=AOvVaw10-3MgyxUr1neDe5F29ofx&ust=1728629068178000&source=images&cd=vfe&opi=89978449&ved=0CBQQjRxqFwoTCMCW0pabg4kDFQAAAAAdAAAAABAE">
        </p>
</body>

</html>
```

3. Write the below code in docker file:

```
FROM nginx:alpine
COPY . /usr/share/nginx/html
```

4. Now open Docker desktop, docker hub and open cmd from the folder location. Type docker login.

5. Type the command => docker images
6. Type the command as it is => docker build -t webform:ver . (if there is no error in the code the code will execute successfully)
7. Type the command => docker images
8. Type the command => docker run -d -p 8080:80 webform:ver
9. Now to see the code : curl localhost:8080
10. Go to browser and type localhost:8080 and the webpage will be displayed. Now if the port is busy change it to 8081, 8082 or more.

Experiment 8: Build an images for a sample Web Form using Dockerfile.

Steps:

1. Create a folder on desktop and create two text files in that folder with names “Index.html” which is a html file and other “Dockerfile” with no extension ie) system file type. (For newer version, by giving no extension doesn’t work. So go to notepad and click on save as and write the file name in double quotes “” which will create a system file.)
2. Now write the following code in html file :

```
<html lang="en">
<head>
 <meta name="description" content="">
   <title>Register</title>
</head>
<body>
   <h1>Fill the details to Register</h1>
     <div>
      <form>
         <label for="First Nmae">Enter First Name:</label>
         <input type="text" name="First">
         <br>
         <br>
         <label for="Last Nmae">Enter Last Name:</label>
         <input type="text" name="Last">
         <br>
         <br>
         <label for="DOB">Enter DOB</label>
         <input type="date" name="dob">
         <br>
         <br>
         <label for="Username">Enter Username:</label>
         <input type="text" name="username">
         <br>
         <br>
         <label for="Password">Enter Password:</label>
         <input type="password" name="password">
      </form>
```

```
    <button>Submit</button>
  </div>

</body>
</html>
```

3. And write the docker file code:
```
FROM nginx:alpine
COPY . /usr/share/nginx/html
```

4. Type the command => docker images
5. Type the command as it is => docker build -t webform:ver . (if there is no error in the code the code will execute successfully)
6. Type the command => docker images
7. Type the command => docker run -d -p 8080:80 webform:ver
8. Now to see the code : curl localhost:8080
9. Go to browser and type localhost:8080 and the webpage will be displayed. Now if the port is busy change it to 8081, 8082 or more.