Lab Manual

for

DevOps LAB

(CM605PC)

III B. TECH II SEMESTER (R20 AUTONOMOUS)

DEPARTMENT OF CSE

(ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING)

ACE

Engineering College

An Autonomous Institution

Ghatkesar, Hyderabad - 501 301, Telangana.

Approved by AICTE & Affiliated to JNTUH

NBA Accredited B.Tech Courses, Accorded NACC A-Grade with 3.20 CGPA

DEVOPS LAB

B.Tech. III Year II Semester

Course Code	Category	Hours/Week			Credits	Maximum Marks		
		L	T	P	C	CIA	SEE	Total
		0	0	3	1.5	30	70	100
Contact Classes: 45	Tutorial Classes: 15	Practical Classes: Nil				Total Classes: 60		

Course Objectives:

1. Describe the agile relationship between development and IT operations.
2. Understand the skill sets and high-functioning teams involved in
3. DevOps and related methods to reach a continuous delivery capability
4. Implement automated system update and DevOps lifecycle

List of Experiments:

1. Write code for a simple user registration form for an event.
2. Explore Git and GitHub commands.
3. Practice Source code management on GitHub. Experiment with the source code written inexercise 1.
4. Jenkins installation and setup, explore the environment.
5. Demonstrate continuous integration and development using Jenkins.
6. Explore Docker commands for content management.
7. Develop a simple containerized application using Docker.
8. Integrate Kubernetes and Docker
9. Automate the process of running containerized application developed in exercise 7 usingKubernetes.
10. Install and Explore Selenium for automated testing.
11. Write a simple program in JavaScript and perform testing using Selenium.
12. Develop test cases for the above containerized application using selenium.

Text Books:

1. Joakim Verona. Practical Devops, Second Edition. Ingram short title; 2nd edition (2018).ISBN-10: 1788392574
2. Deepak Gaikwad, Viral Thakkar. DevOps Tools from Practitioner's Viewpoint. Wiley publications. ISBN: 9788126579952

REFERENCE BOOK:

1. Len Bass, Ingo Weber, Liming Zhu. DevOps: A Software Architect's Perspective. AddisonWesley
2. Edureka DevOps Full Course - https://youtu.be/S_0q75eD8Yc

Lab01 :

Writecodeforasimpleuserregistration formforanevent.

```
<divclass="container">
   <h1>EventRegistrationForm</h1>
   <form name="Registration" class="registartion-
form"onsubmit="returnformValidation()">
      <table>
         <tr>
            <td><labelfor="name">Name:</label></td>
            <td><inputtype="text"name="name"id="name"placeholder=
"yourname"></td>
         </tr>
         <tr>
            <td><labelfor="email">Email:</label></td>
            <td><inputtype="text"name="email"id="email"placeholder="youremail"><
/td>
         </tr>
         <tr>
            <td><labelfor="password">Password:</label></td>
            <td><input type="password"
name="password"id="password"></td>
         </tr>
         <tr>
            <td><labelfor="phoneNumber">PhoneNumber:</label></td>
            <td><inputtype="number"name="phoneNumber"id="phoneNumber"></td>
         </tr>
         <tr>
            <td><labelfor="gender">Gender:</label></td>
            <td>Male:<inputtype="radio"name="gender"value="male">
               Female: <input type="radio"
name="gender"value="female">
               Other: <input type="radio"
name="gender"value="other"></td>
         </tr>
         <tr>
            <td><labelfor="language">language</label></td>
            <td>
               <selectname="language"id="language">
```

```
                <optionvalue="">Selectlanguage</option>
                <optionvalue="English">English</option>
                <optionvalue="Spanish">Spanish</option>
                <optionvalue="Hindi">Hindi</option>
                <optionvalue="Arabic">Arabic</option>
                <optionvalue="Russian">Russian</option>
              </select>
            </td>
          </tr>
          <tr>
            <td><labelfor="zipcode">ZipCode:</label></td>
            <td><input type="number"
name="zipcode"id="zipcode"></td>
          </tr>
          <tr>
            <td><labelfor="about">About:</label></td>
            <td><textareaname="about"id="about"placeholder="Writeaboutyourself..."
></textarea></td>
          </tr>
          <tr>
            <td colspan="2"><input type="submit"
class="submit"value="Register"/></td>
          </tr>
        </table>
    </form>
</div>
```

Event Registration Form

Name: your name

Email: your email

Password:

Phone Number:

Gender: Male: Female: Other:

language Select language

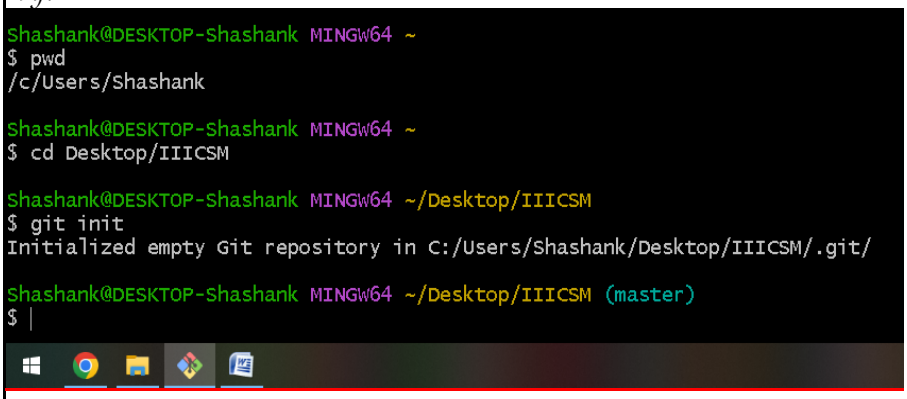
Zip Code:

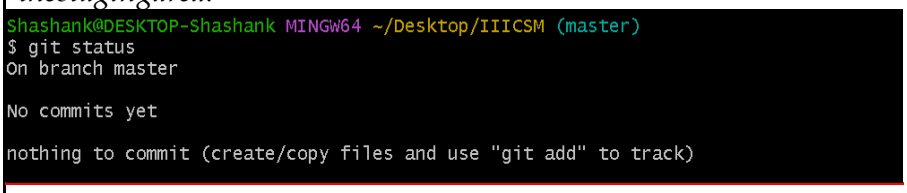
About: Write about yourself...

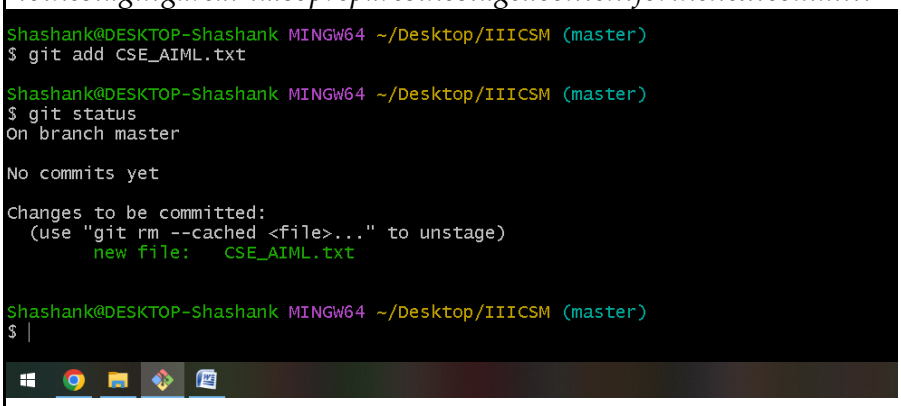
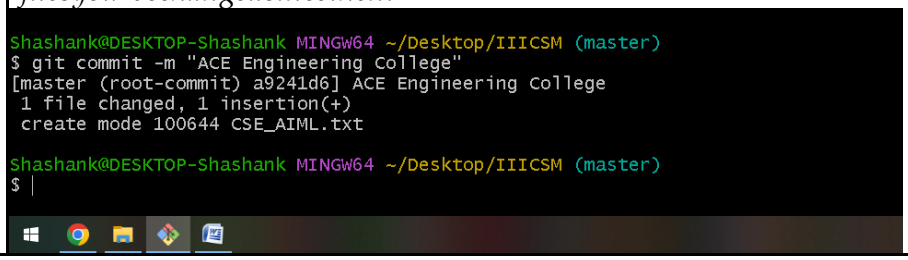
Register

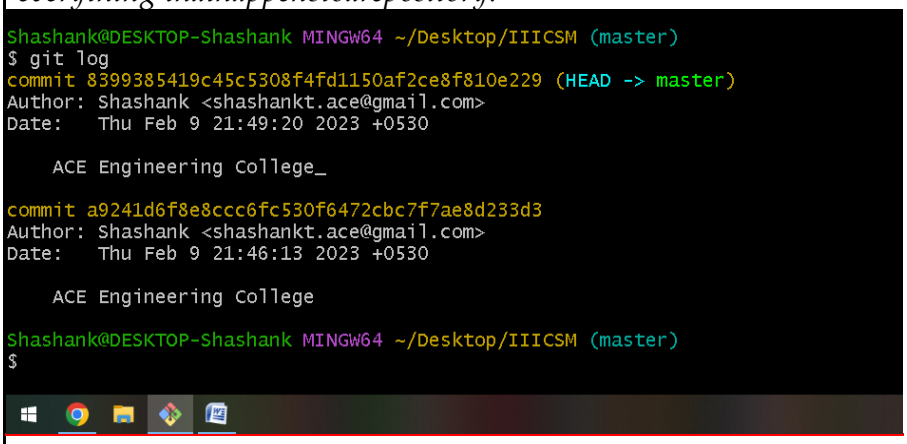
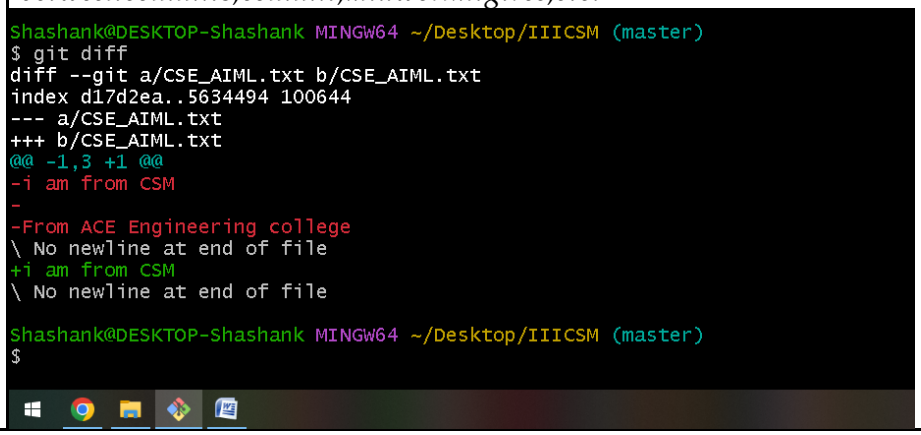
Lab02 :
ExploreGitandGithubcommands.

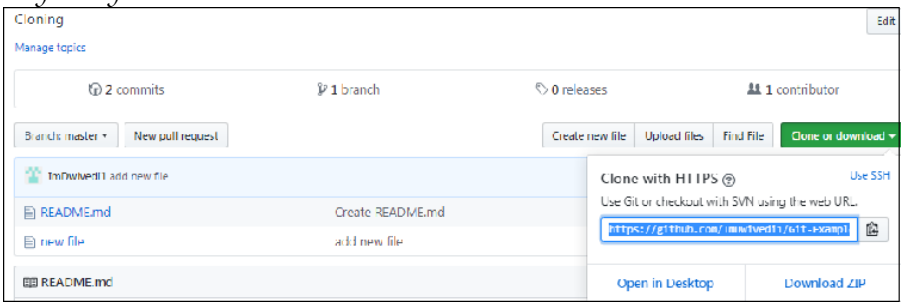
GitEnvironmentSetup-GitConfigcommand	*Git supports a command called git config that lets you get and setconfigurationvariablesthatcontrolallfacetsofhowGitlooksandoperates .ItisusedtosetGitconfigurationvaluesonaglobalorlocal projectlevel. Setting user.name and user.email are the necessary configuration options asyournameandemailwill showupinyourcommitmessages.*
	`$ git config --global user.name "Himanshu Dubey"`
	`$ git config --global user.email "himanshudubey481@gmail.com"`

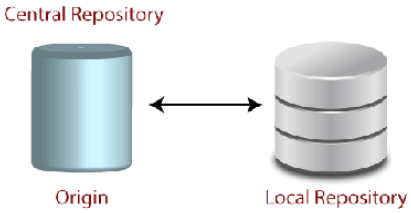
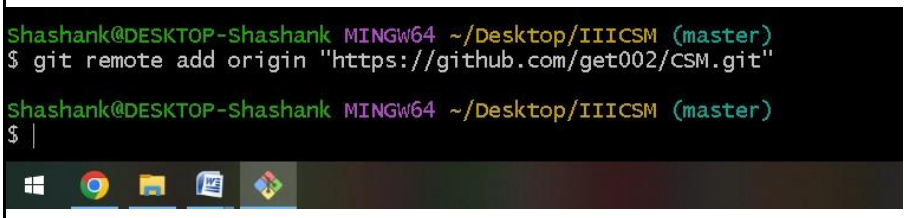
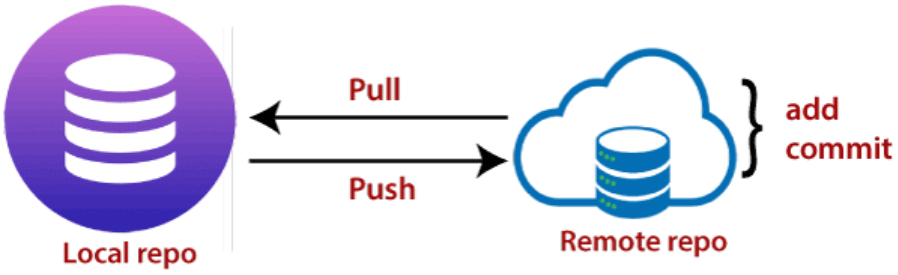
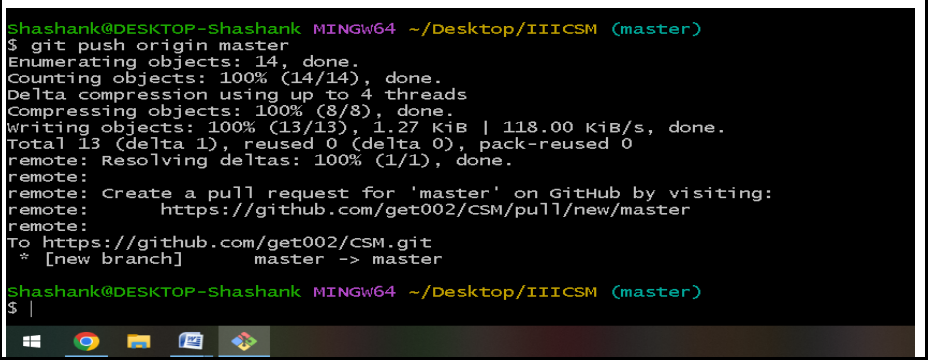
Gitinitcommand	*Thiscommandisusedtocreatealocalrepository. Thegitinitcommandisthefirstcommandthatyouwill runonGit.Thegitinitcommandisusedtocreateanewblankreposito ry.* 

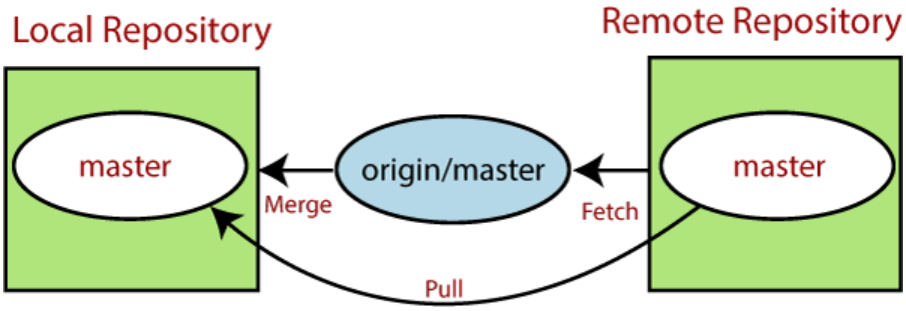
Git statuscommand	*Thestatuscommandisusedtodisplaythestateoftheworkingdirectoryandthestagingarea.* 

Gitaddcommand	*This commandis usedtoaddone ormore filestostaging(Index) area.The git add command is used to add file contents to the Index (StagingArea). This command updates the current content of the working tree tothestagingarea.Italsopreparesthestagedcontentforthenextcommit.* 
Gitcommitcommand	*It is used to record the changes in the repository. It is the next commandafterthegitadd.Thiscommandcommitsany filesaddedwithgitaddintherepositoryandalso commitsany filesyou'vechangedsincethen.* 

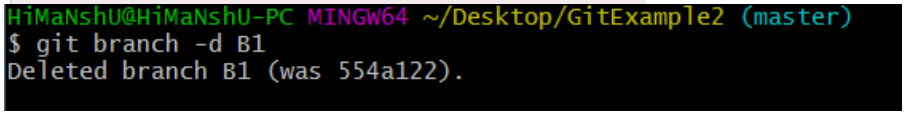
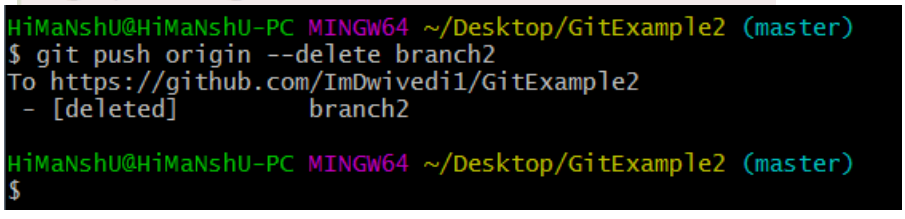
Gitlogcommand	*Thiscommandisusedtocheckthecommithistory.* *Git log is a utility tool to review and read a history of everything thathappenstoarepository.* 
GitDiffcommand	*Itcomparesthedifferentversionsofdatasources.* *Git diff is a command-line utility. It's a multi use Git command. When it isexecuted,itrunsadifffunctiononGitdatasources.Thesedatasourcescanbe files, branches, commits, and more. It is used to show changes betweencommits,commit,andworkingtree,etc.* 

GitClone	*In Git, cloning is the act of making a copy of any target repository.Thetargetrepository canberemoteorlocal. Youcancloneyourrepositoryfromtheremoterepositorytocreatealocalcopy onyoursystem.*  $ git clone https://github.com/ImDwivedi1/Git-Example.git

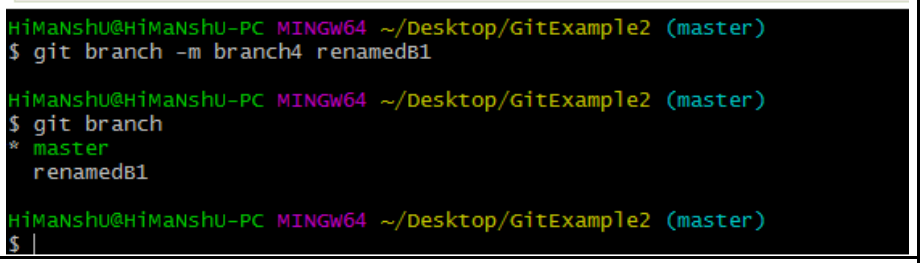
GitOriginMaster	*In Git, The term origin is referred to the remote repository where you wanttopublishyourcommits. Thedefaultremoterepositoryiscalledorigin,althoughyoucanworkwithsev eralremoteshavingadifferentnameatthesametime. The origin is a short name for the remote repository that a project wasinitiallybeingcloned.ItisusedinplaceoftheoriginalrepositoryURL.T hus,itmakesreferencingmucheasier.*  
GitPushCommand	*The push term refers to upload local repository content to a remoterepository.Pushingisanactoftransfercommitsfromyourlocalrepo sitoryto a remote repository. Pushing is capable of overwriting changes; cautionshouldbetaken whenpushing.*  

GitPullCommand	*The term pull is used to receive data from GitHub. It fetches and mergeschanges from the remote server to your working directory. The git pullcommandisused topull arepository.*
	 
	`Shashank@DESKTOP-Shashank MINGW64 ~/Desktop/IIICSM (master)`
	`$ git pull origin main`
	`From https://github.com/get002/CSM`
	`* branch main -> FETCH_HEAD`
	`fatal: refusing to merge unrelated histories`
	
	`Shashank@DESKTOP-Shashank MINGW64 ~/Desktop/IIICSM (master)`
	`$ git pull origin main --allow-unrelated-histories`
	`From https://github.com/get002/CSM`
	`* branch main -> FETCH_HEAD`
	`Merge made by the 'ort' strategy.`
	`README.md | 1 +`
	`1 file changed, 1 insertion(+)`
	`create mode 100644 README.md`
	
	`Shashank@DESKTOP-Shashank MINGW64 ~/Desktop/IIICSM (master)`
	`$`

GitFetchc ommand	*Git"fetch"Downloadscommits,objectsandrefsfromanotherrepository.It fetches branches and tags from one or more repositories. It holdsrepositories along with the objects that are necessary to complete theirhistoriestokeepupdatedremote-trackingbranches.*
	`HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/Git-Example (master)`
	`$ git fetch https://github.com/ImDwivedi1/Git-Example.git`
	`warning: no common commits`
	`remote: Enumerating objects: 6, done.`
	`remote: Counting objects: 100% (6/6), done.`
	`remote: Compressing objects: 100% (4/4), done.`
	`remote: Total 6 (delta 0), reused 0 (delta 0), pack-reused 0`
	`Unpacking objects: 100% (6/6), done.`
	`From https://github.com/ImDwivedi1/Git-Example`

OperationsonBranches-CreateBranch	*You can create a new branch with the help of the git branch command. Thiscommandwill beusedas:* `$ git branch <branch name>` 
ListBranch	*YoucanListalloftheavailablebranchesinyourrepositorybyusingthefollowingcommand.* *Either we can use git branch - list or git branch command to list theavailablebranchesintherepository.* `$ git branch --list` 
OperationsonBranches-DeleteBranch	*You can delete the specified branch. It is a safe operation. In this command,Gitpreventsyoufromdeletingthebranchifithasunmergedchanges.* `$ git branch -d <branch name>` 
OperationsonBranches-DeleteaRemote Branch	*Youcandeletearemotebranchfrom Gitdesktopapplication.* `$ git push origin -delete <branch name>` 

Date:

OperationsonBranches - SwitchBranch	*Gitallowsyoutoswitchbetween thebrancheswithoutmakingacommit.Youcanswitchbetweentwobranchesswiththegitcheckoutcommand.* $ git branch -m <old branch name> <new branch name> 

Lab03 :

PracticesourcecodemanagementonGitHub.Experimentwiththesourcecodewritteninexercise1.

Check the statusoflocalrepositoryusinggitstatuscommand,

```
Shashank@DESKTOP-Shashank MINGW64 ~/Desktop/WebProject (master)
$ git status
On branch master

No commits yet

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

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

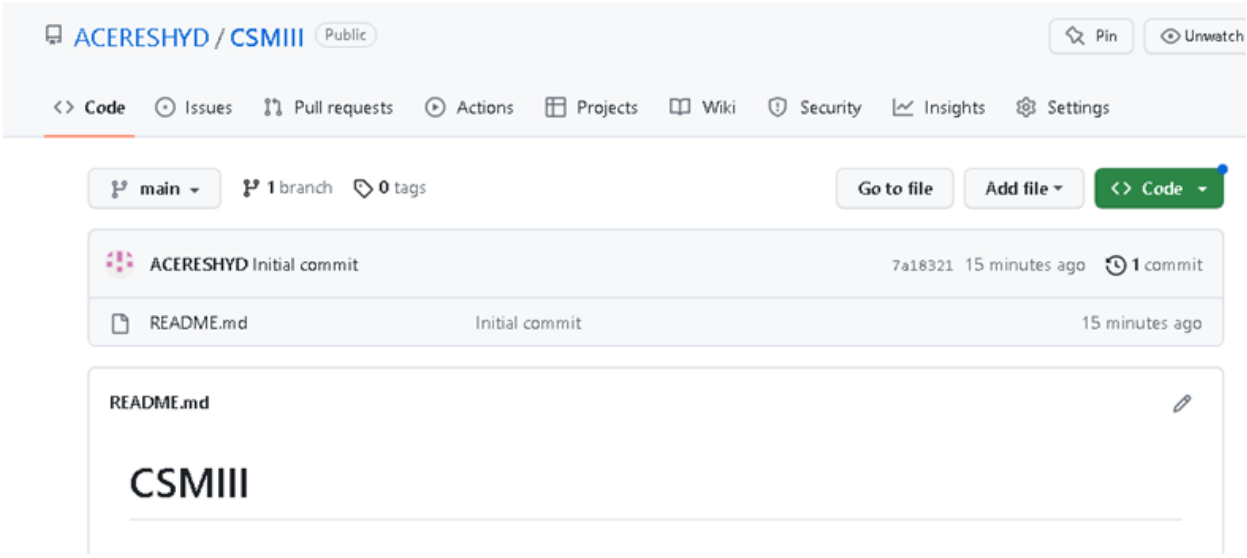
Thereisafileindex.htmlatworkingarea, Usegitaddcommandtoaddonstagingarea. Thencommitthe changesonlocalrepositoryusinggitcommitcommand.

```
Shashank@DESKTOP-Shashank MINGW64 ~/Desktop/WebProject (master)
$ git add index.html

Shashank@DESKTOP-Shashank MINGW64 ~/Desktop/WebProject (master)
$ git commit -m "index file"
[master (root-commit) dab7375] index file
 1 file changed, 92 insertions(+)
 create mode 100644 index.html
```

```
Shashank@DESKTOP-Shashank MINGW64 ~/Desktop/WebProject (master)
$ git log
commit dab7375858be19f5a9c3b7e35a230bdbe344a9b6 (HEAD -> master)
Author: Shashank <shashankt.ace@gmail.com>
Date:   Fri Mar 31 18:39:32 2023 +0530

    index file

Shashank@DESKTOP-Shashank MINGW64 ~/Desktop/WebProject (master)
$ git status
On branch master
nothing to commit, working tree clean
```

Allchangesupdatedonlocalrepository.

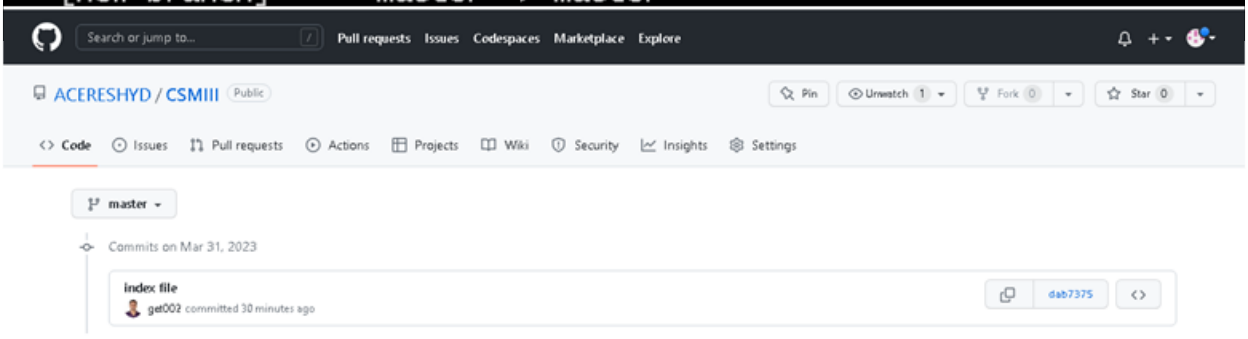
Nowtopushthesechangesonremoterepository,firstcreatearemoterepository.




```
Shashank@DESKTOP-Shashank MINGW64 ~/Desktop/WebProject (master)
$ git remote add origin git@github.com:ACERESHYD/CSMIII.git
```

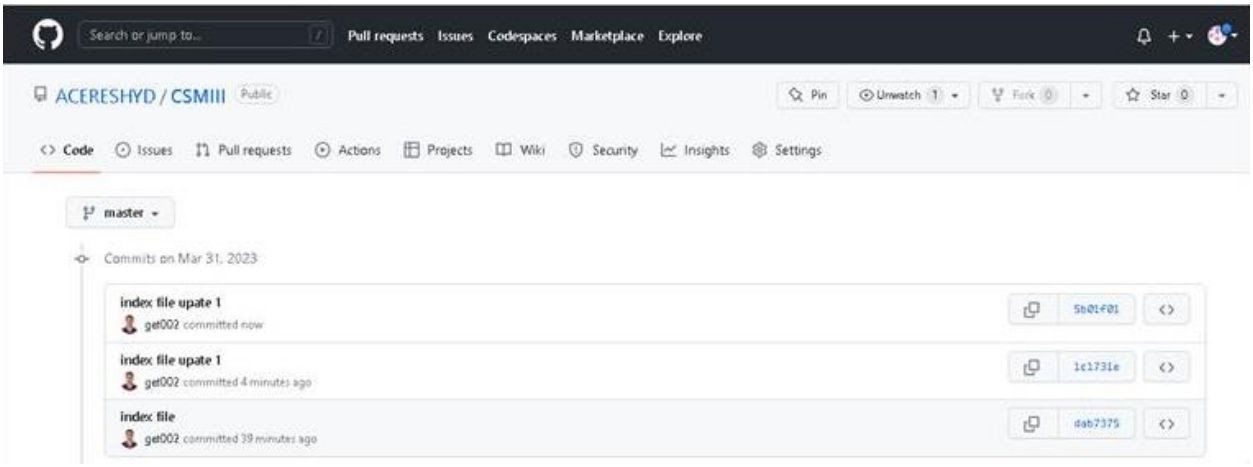
Nowpushthechangesfromlocalrepositoryto remoterepositoryusing gitpushoriginmaster

```
Shashank@DESKTOP-Shashank MINGW64 ~/Desktop/WebProject (master)
$ git push origin master
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), 1018 bytes | 169.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:      https://github.com/ACERESHYD/CSMIII/pull/new/master
remote:
To github.com:ACERESHYD/CSMIII.git
 * [new branch]      master -> master
```



Task : Makes atleast 6 commits on your remote repository from git bash and alsofromgithub topractisesourcecodemanagement.

```
Shashank@DESKTOP-Shashank MINGW64 ~/Desktop/WebProject (master)
$ git add index.html

Shashank@DESKTOP-Shashank MINGW64 ~/Desktop/WebProject (master)
$ git commit -m "index file upate 1"
[master 1c1731e] index file upate 1
 1 file changed, 1 insertion(+), 1 deletion(-)

Shashank@DESKTOP-Shashank MINGW64 ~/Desktop/WebProject (master)
$ git push origin master
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 282 bytes | 94.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:ACERESHYD/CSMIII.git
   dab7375..1c1731e  master -> master

Shashank@DESKTOP-Shashank MINGW64 ~/Desktop/WebProject (master)
$ git log
commit 1c1731eef8150d21459a893f660aa0e603d9db7b (HEAD -> master, origin/master)
Author: Shashank <shashankt.ace@gmail.com>
Date:   Fri Mar 31 19:14:24 2023 +0530

    index file upate 1

commit dab7375858be19f5a9c3b7e35a230bdbe344a9b6
Author: Shashank <shashankt.ace@gmail.com>
Date:   Fri Mar 31 18:39:32 2023 +0530

    index file
```



Here**HEAD**alwaysrepresentcurrentcommitcontent.Nowcomparing latestcommitwiththepreviouscommitusinggitdiffcommand

```
Shashank@DESKTOP-Shashank MINGW64 ~/Desktop/WebProject (master)
$ git diff 5b01f017 1c1731ee
diff --git a/index.html b/index.html
index 377aeea..f76e7b0 100644
--- a/index.html
+++ b/index.html
@@ -6,7 +6,6 @@
 <h2 align="center"> An Autonomous institute</h2>
 <hr/>
 <title>Student Registration Form</title>
-<hr/>
 <style type="text/css">
 th {
```

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

index file upate 1

master

Browse files

get002 committed 12 minutes ago

1 parent 1c1731e commit 5b01f01

Showing 1 changed file with 1 addition and 0 deletions.

Split Unified

1 index.html

```
@@ -6,6 +6,7 @@ <h1 align="center">ACE Engineering College </h1>
6  6   <h2 align="center"> An Autonomous institute</h2>
7  7   <hr/>
8  8   <title>Student Registration Form</title>
   9 + <hr/>
9  10  <style type="text/css">
10 11  th {
11 12          color: #FFF;
```

0 comments on commit 5b01f01

Lock conversation

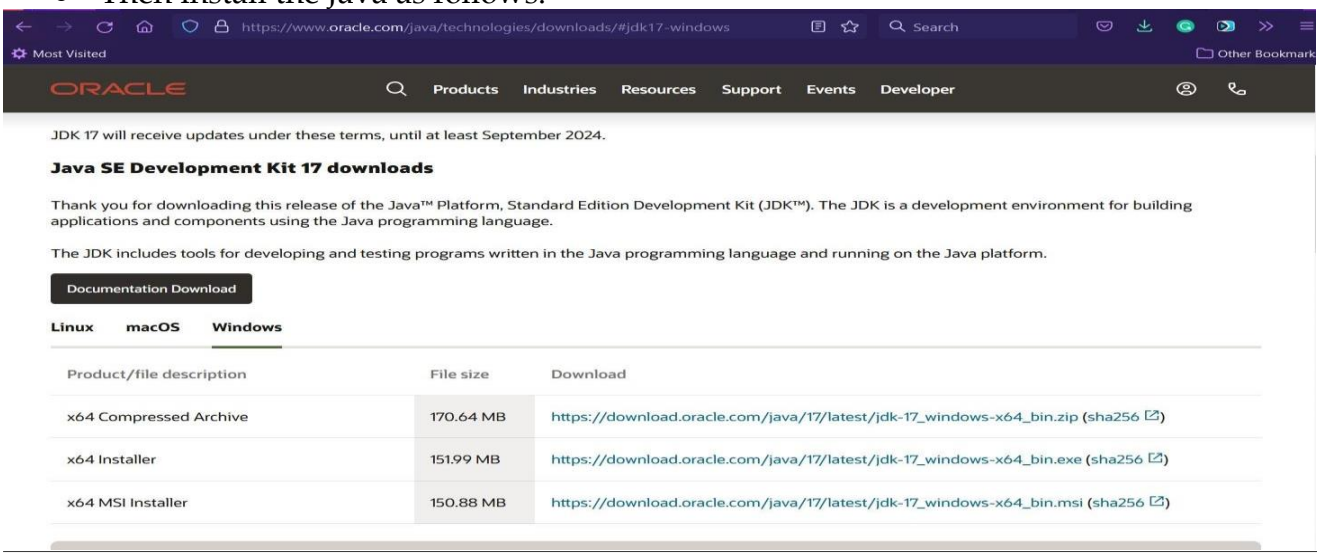
Lab04:
Jenkins installation and setup, explore the environment.

- Jenkins is an open source automation tool written in Java programming language.
- Jenkins is free and allows continuous integration.
- What is continuous integration?
    - Continuous integration refers to the build and unit testing stages of the software release process. Every revision that is committed triggers an automated build and test. With continuous delivery, code changes are automatically built, tested, and prepared for a release to production.
- Jenkins is a server based application and requires a web server like Apache Tomcat.
- Jenkins builds and tests our software projects continuously.
    - This is the main reason for Jenkins to become so popular, continuously monitoring of repeated tasks which arise during the development of a project.
    - Example, if your team is developing a project, Jenkins will continuously test your project builds and show you the errors in early stages of your development.
- Benefits of using Jenkins CI
    - **Reduced Development Cycle –**Since every commit is built and tested, it allows releasing new features to the user faster and with fewer errors.
    - **Shorter Time to Integrate Code –** Before the use of Jenkins CI, integration of code was done manually, thus taking a few days. In some cases, it might happen that the code is not running, and it is hard to debug as it might have gone through various commits in the repository. Integrating code after every commit ensures that the functionality is not broken after a commit.
    - **Faster Feedback Loops –** Developers get feedback and improve the code whenever a test breaks during a commit. Otherwise, debugging the issue can be very difficult; given teams would not be sure which commit resulted in the bug.
    - **Automated Workflow –** Teams should not worry about running a manual test for each commit. The Jenkins CI pipeline checks the latest code and builds the code along with the tests. The test can deploy the project in a specific environment if it is green. It can notify the developer by breaking the build.

Jenkinsinstallation →

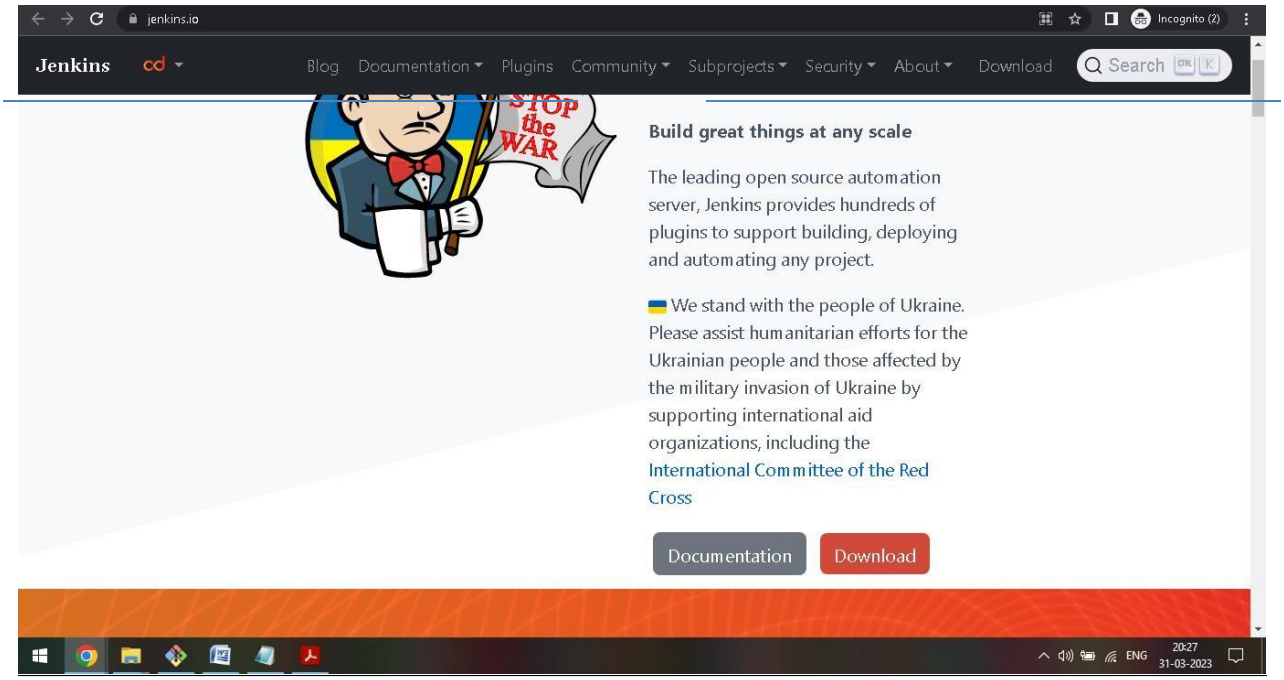
InstallJavaVersion8

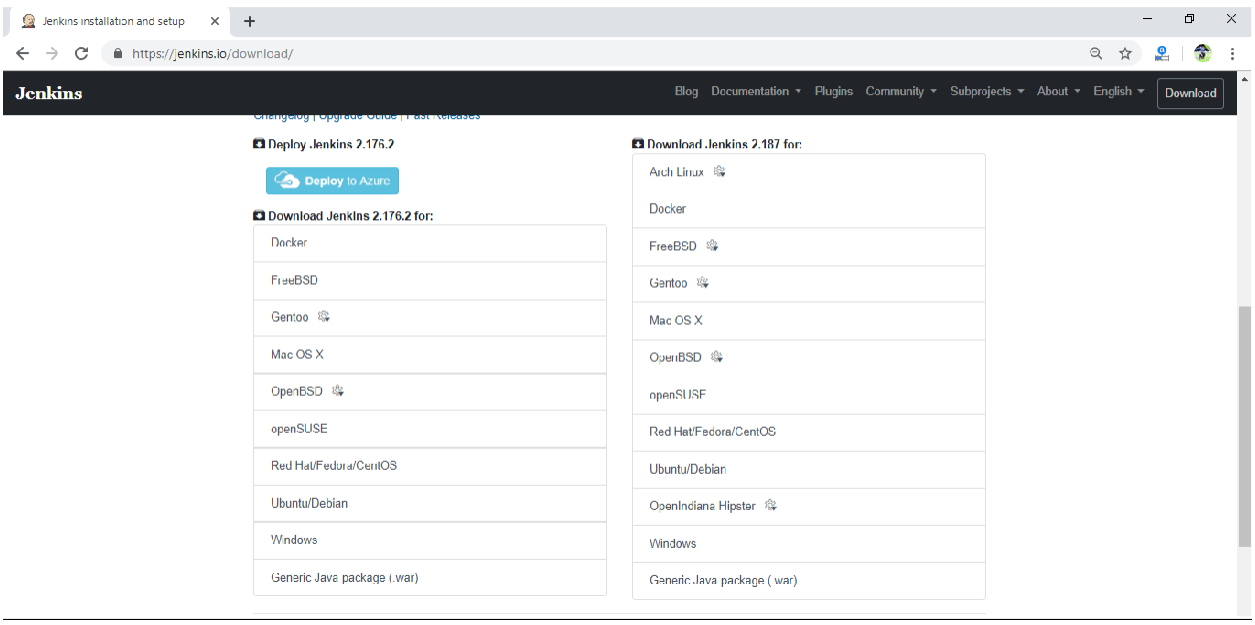
- Since Jenkins is a Java based application, therefore Java is a must.
- Download java8 from the below link:
  https://www.oracle.com/java/technologies/downloads/#java8
- Then install the Java as follows:



DownloadJenkinswarFile

Download from the following link https://www.jenkins.io/



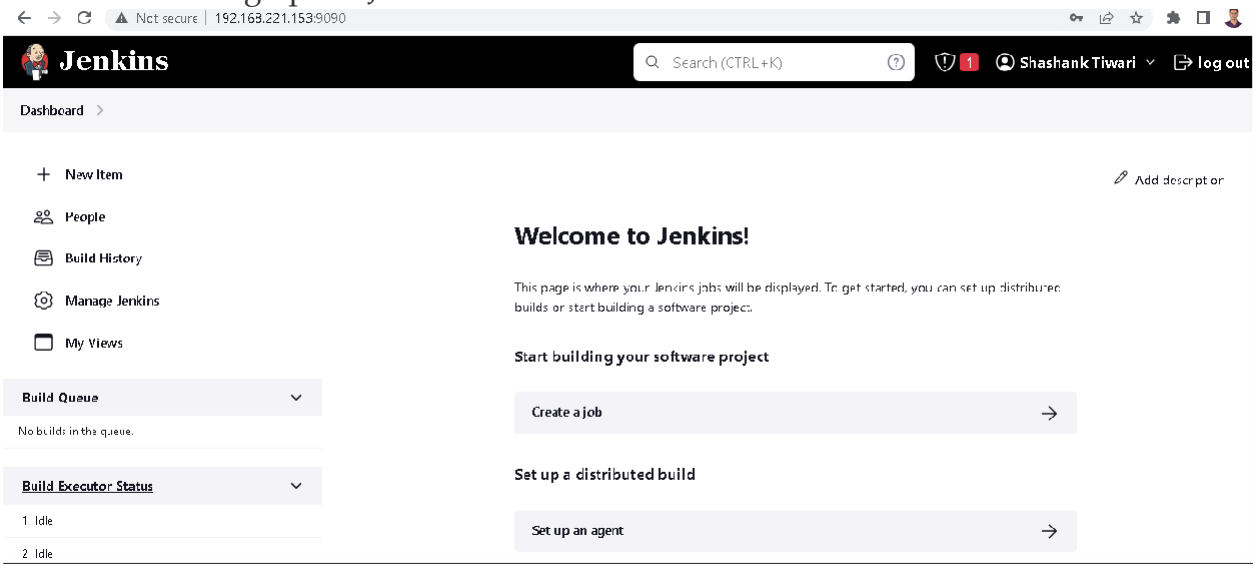


Accessing Jenkins

Now you can access the Jenkins. Open your browser and type the following url on your browser:

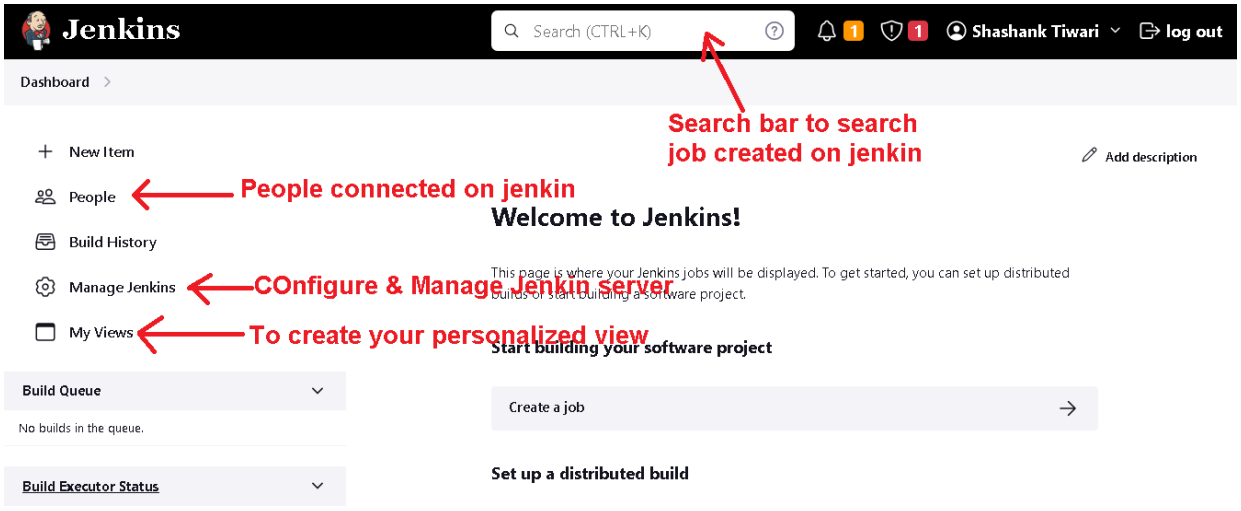
http://localhost:8080

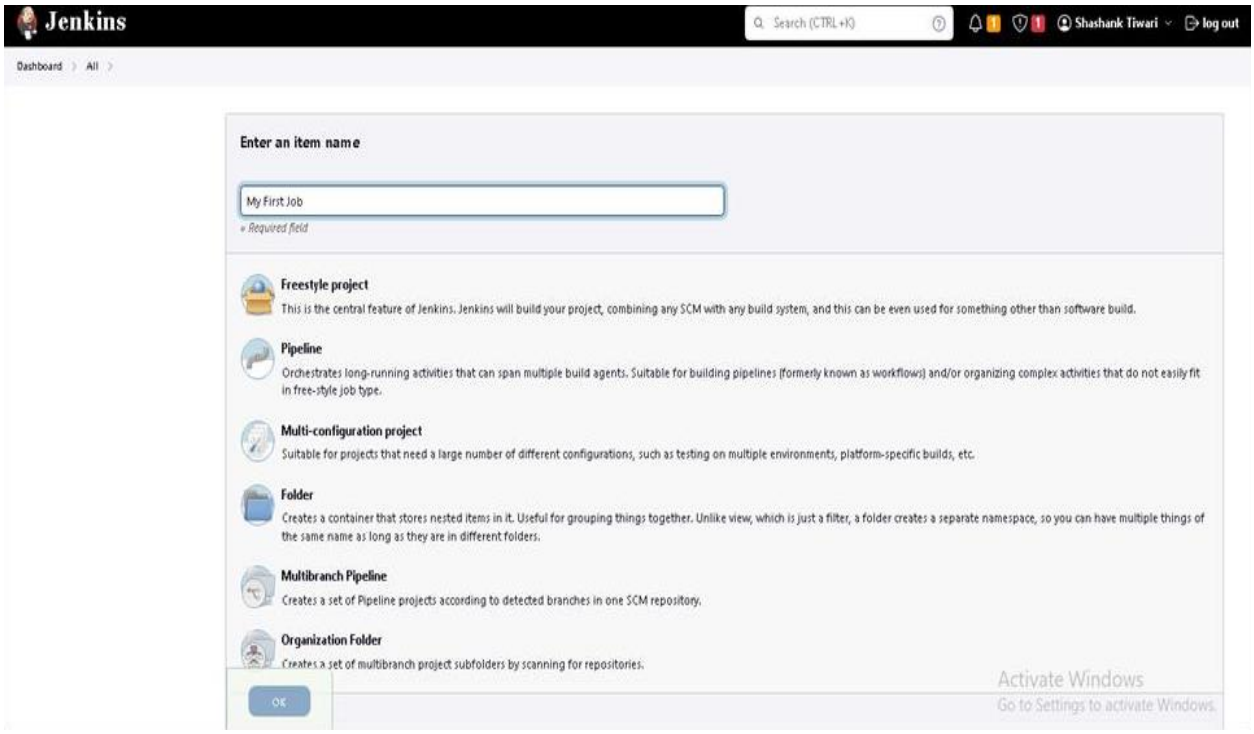
This url will bring up the Jenkins dashboard.



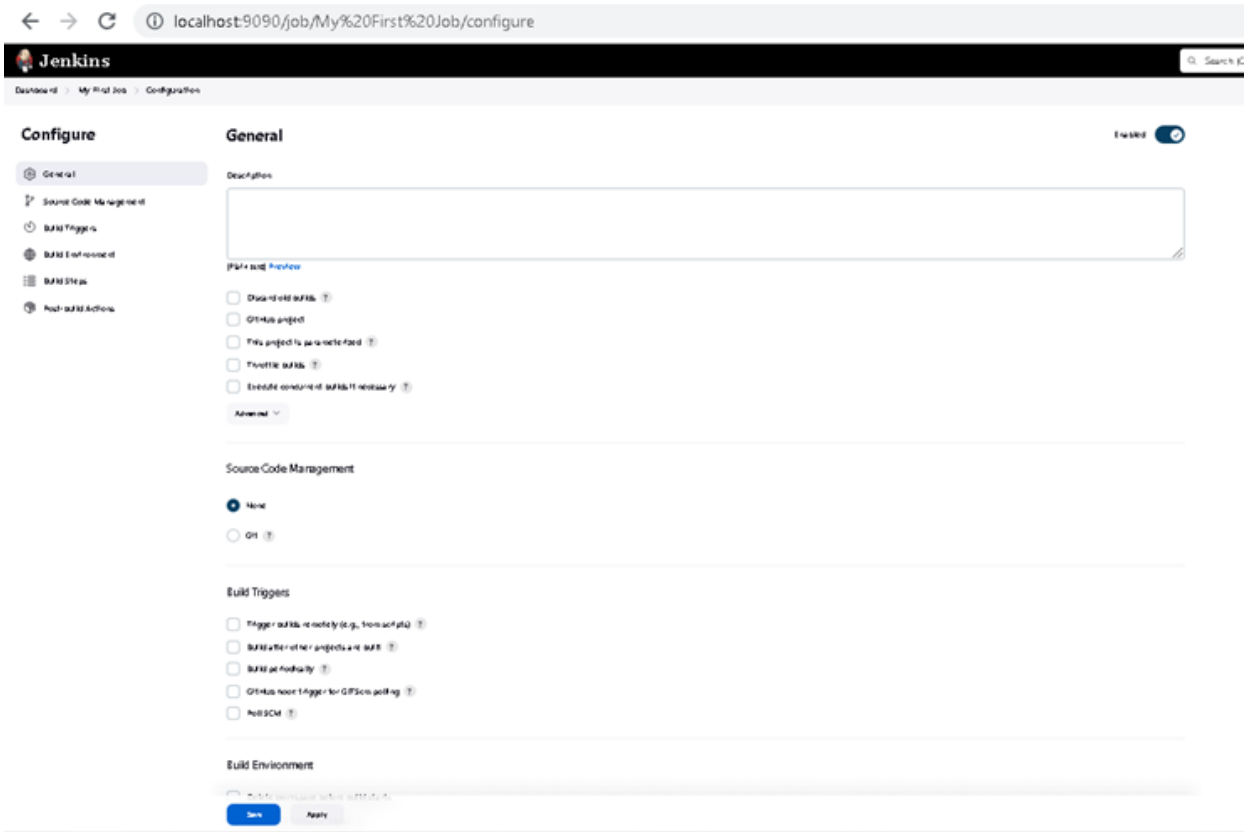
Exploring the Jenkin environment

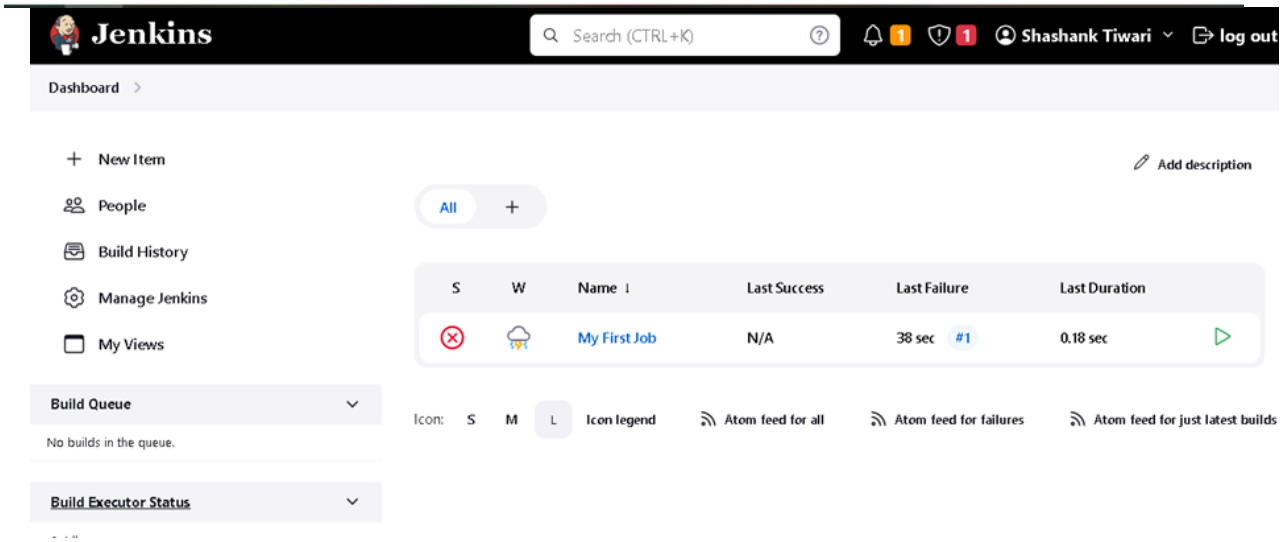
- Log into your jenkin server





Next, define the parameter with the job

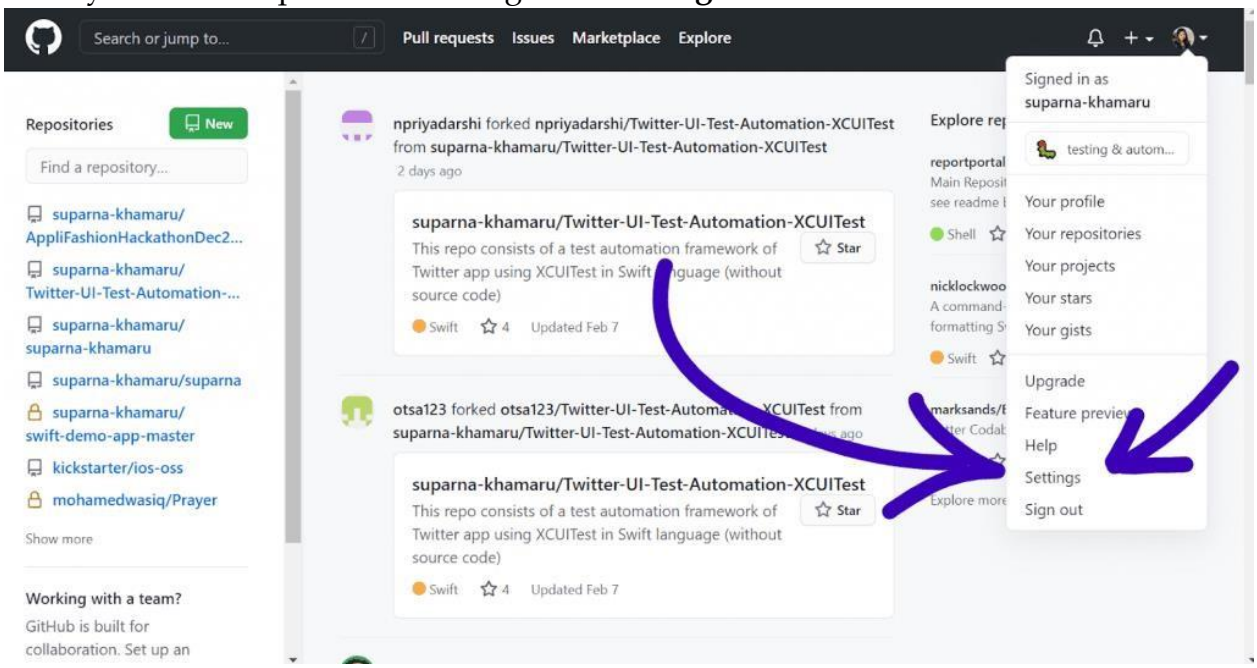

localhost:9090/job/My%20First%20Job/configure
Jenkins
Configure
General
Source Code Management
Build Triggers
Build Environment


Jenkins
Search (CTRL+K)
Shashank Tiwari
log out
Dashboard
New Item
People
Build History
Manage Jenkins
My Views
Add description
All
S
W
Name
Last Success
Last Failure
Last Duration
My First Job
N/A
38 sec
#1
0.18 sec
Build Queue
No builds in the queue.
Build Executor Status
Icon: S M L
Icon legend
Atom feed for all
Atom feed for failures
Atom feed for just latest builds

Lab05:

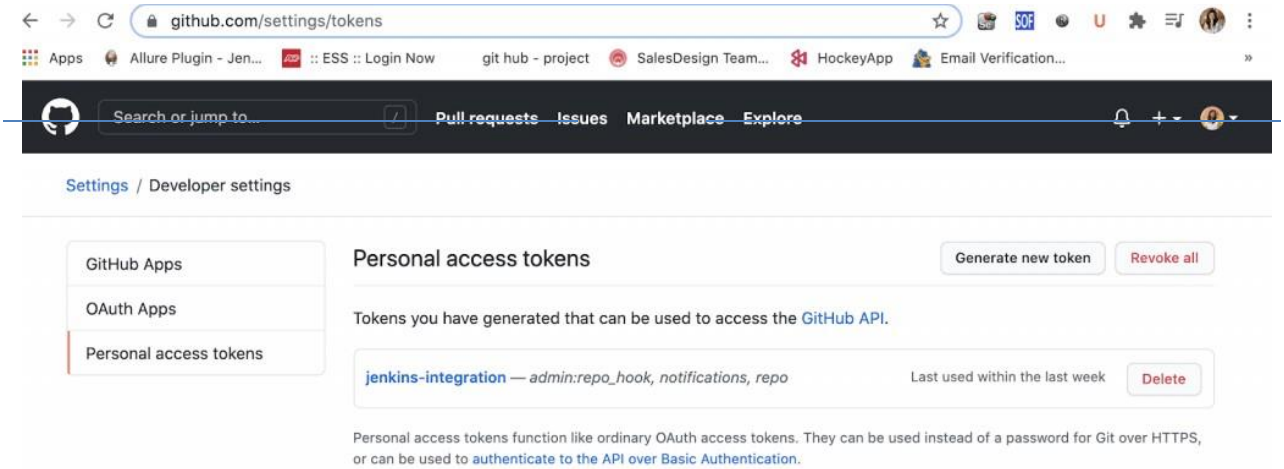
Demonstrate continuous integration and development using jenkins.

Go to your Github profile and navigate to **Settings**.



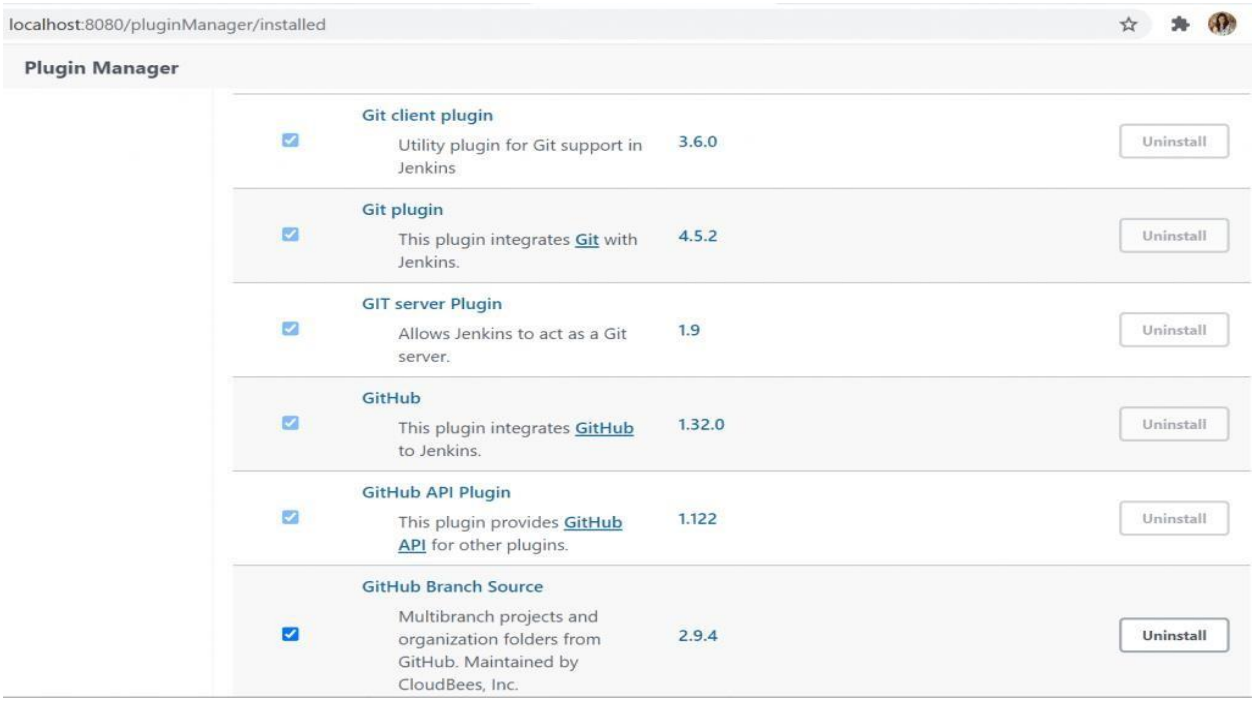
In the settings screen, click on the "**Developer settings**" menu and click on "**Personal access tokens**."

In the "**Personal access tokens**" tab, click on the "**Generate new token**" button and provide necessary details as desired (an example is provided in the below figure), and click on the "**Generate token**" button.



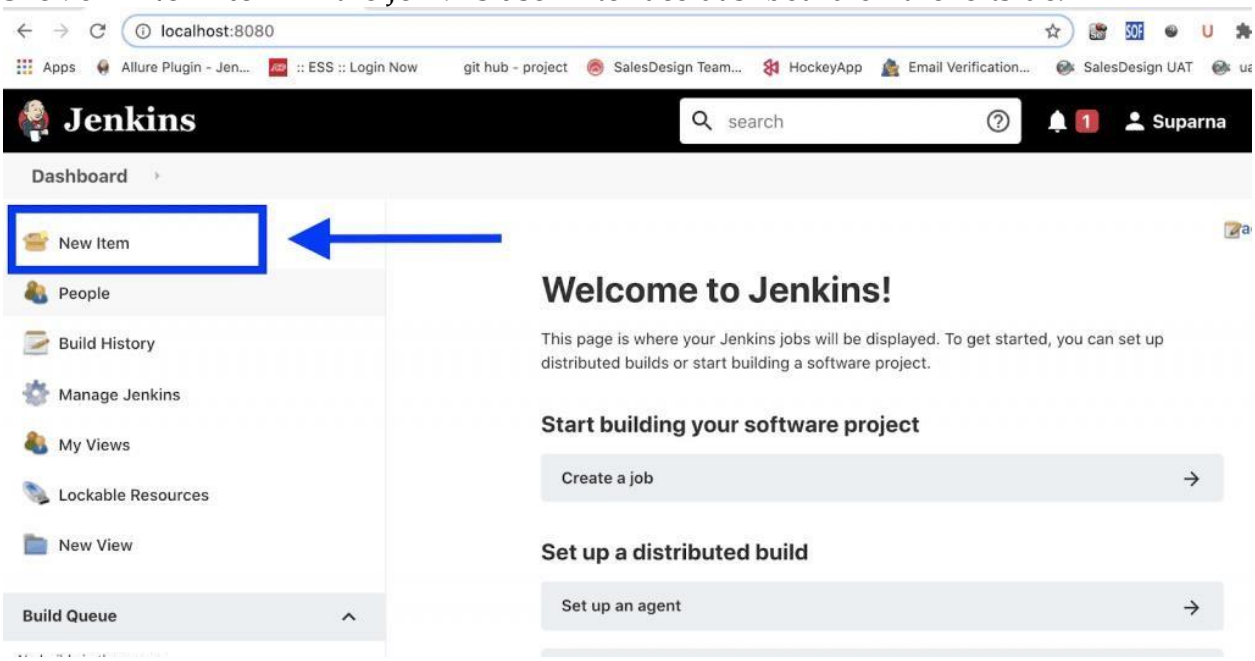
Configuring Jenkins for Git Hub Integration

Steps to follow - Go to : Jenkins Dashboard->Manage Jenkins->Manage Plugins->'Available' tab->Enter Git in search bar and filter - > Install required plugin



Creating Your First Jenkins Job Integrated Into a TestProject

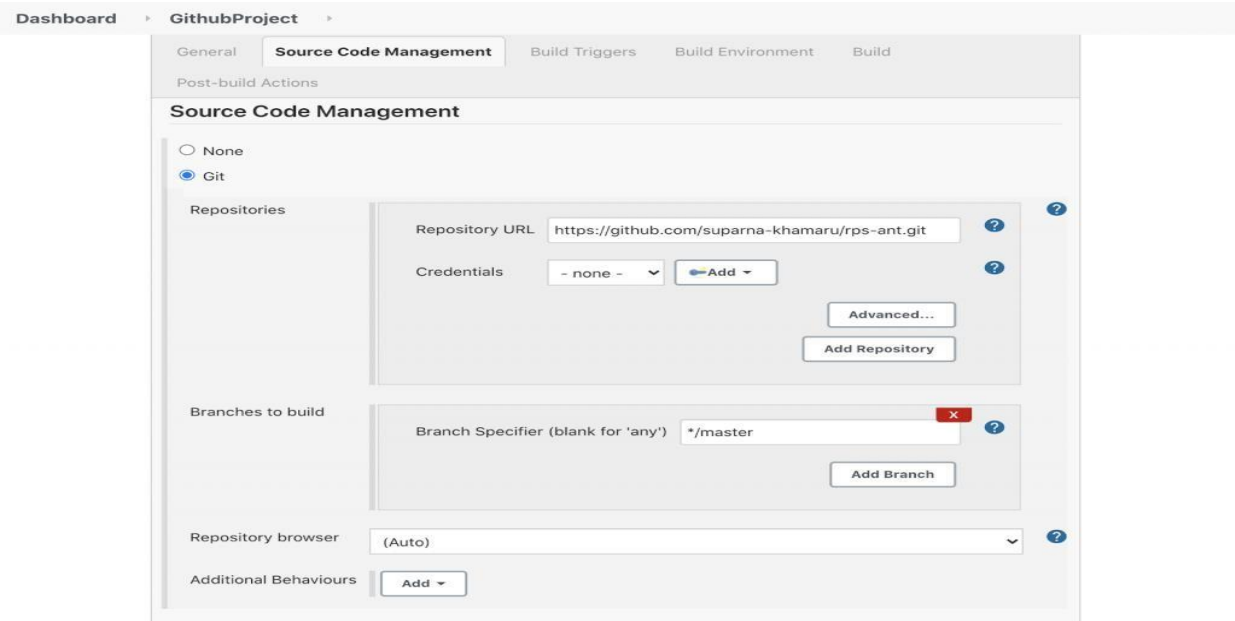
Click on "**New Item**"in the Jenkins user interface dashboard on the leftside.



Enter a project name in the requested textbox field and select the **Freestyle project.**

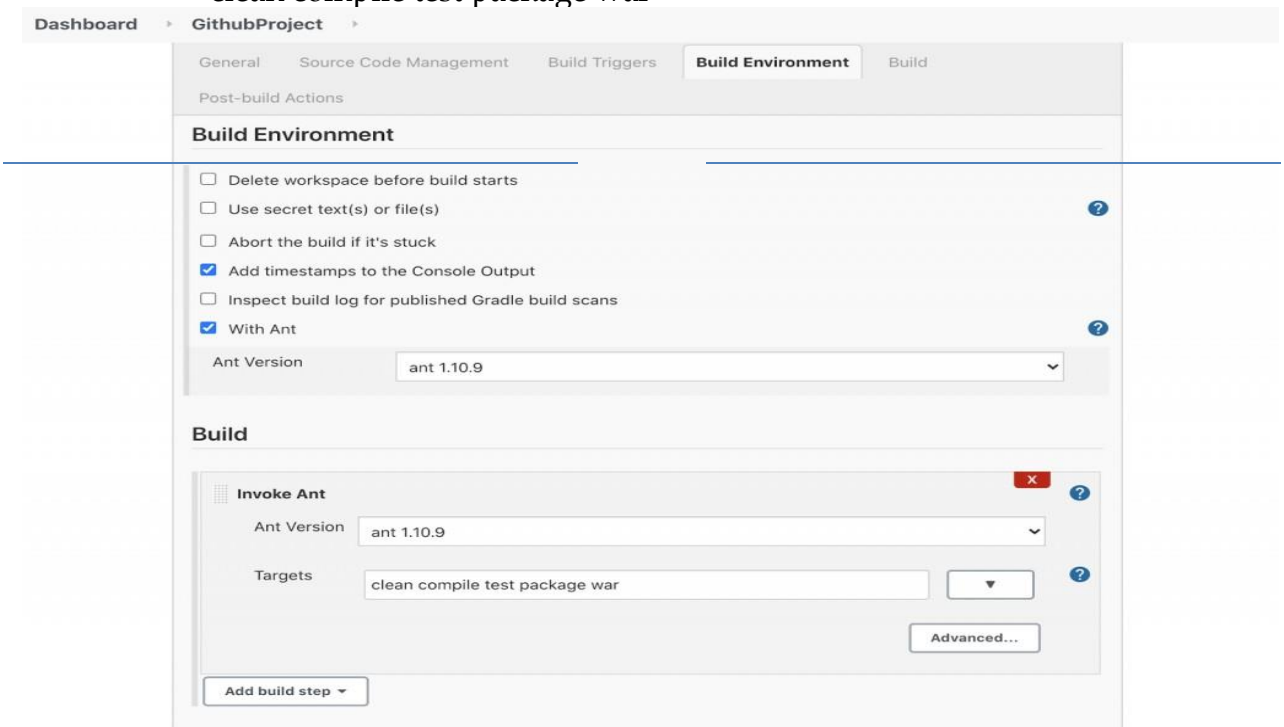
In the **General** section, check the field "**Git Hub project**" and enter a valid Ant project as given below.

Then, go to the "**Source Code Management**" section and enter the same link as above with an extra.git extension at the end of the URL, as shown in the below screen.

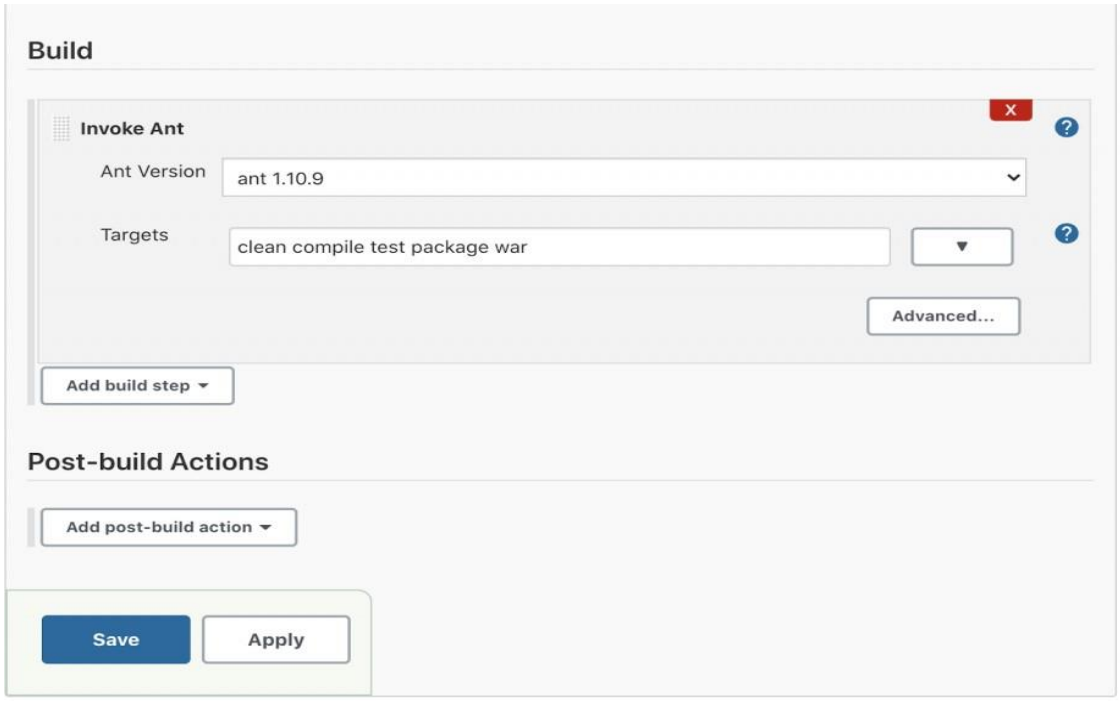


Go to the "**Build Environment**" section and add the following steps shown in the image below. Make sure to add the command in the Targets field under the Ant Build step:

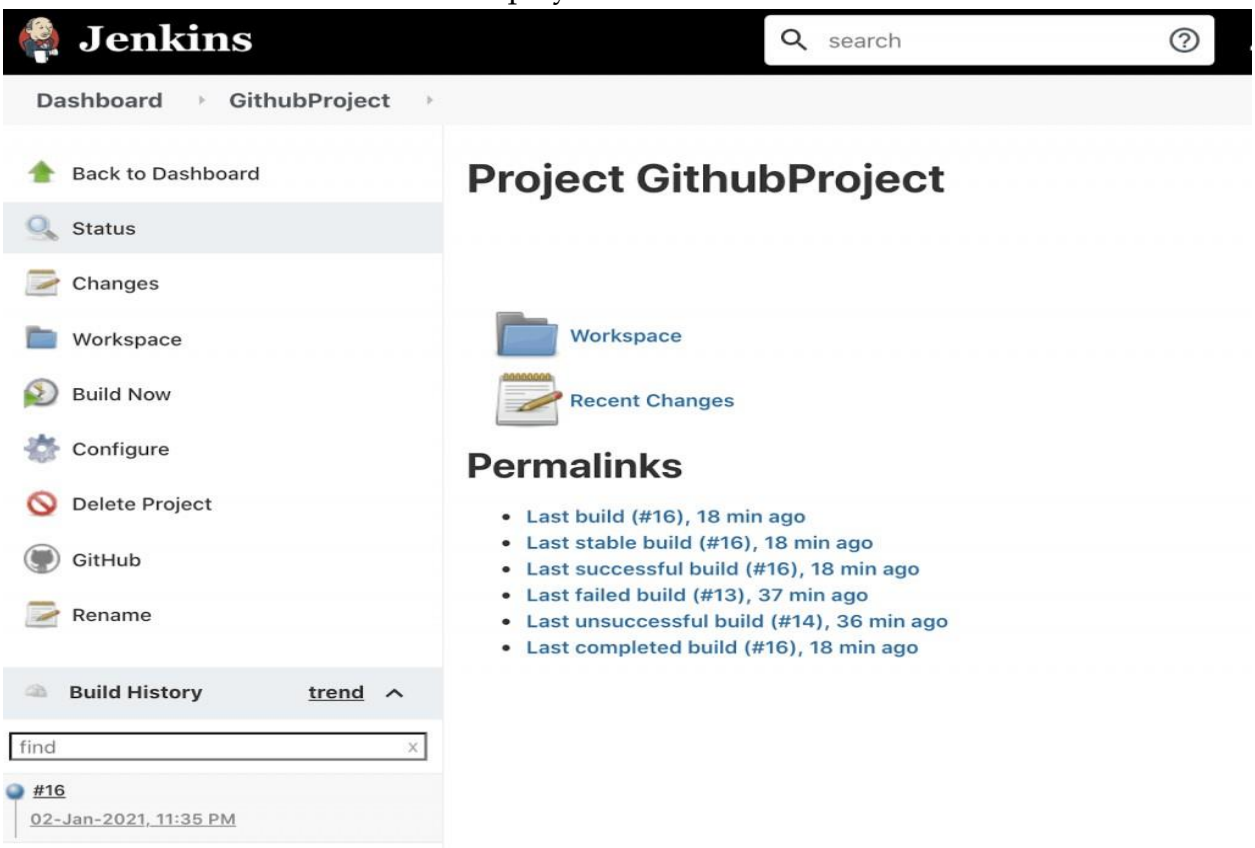
clean compile test package war

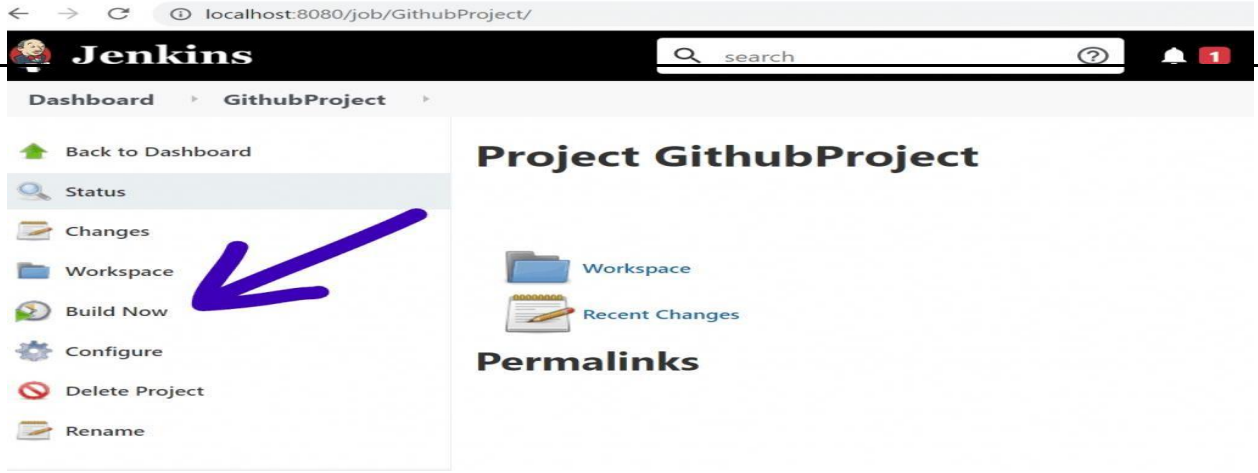


Click on the"**Apply**"button and then click on the"**Save**"button.

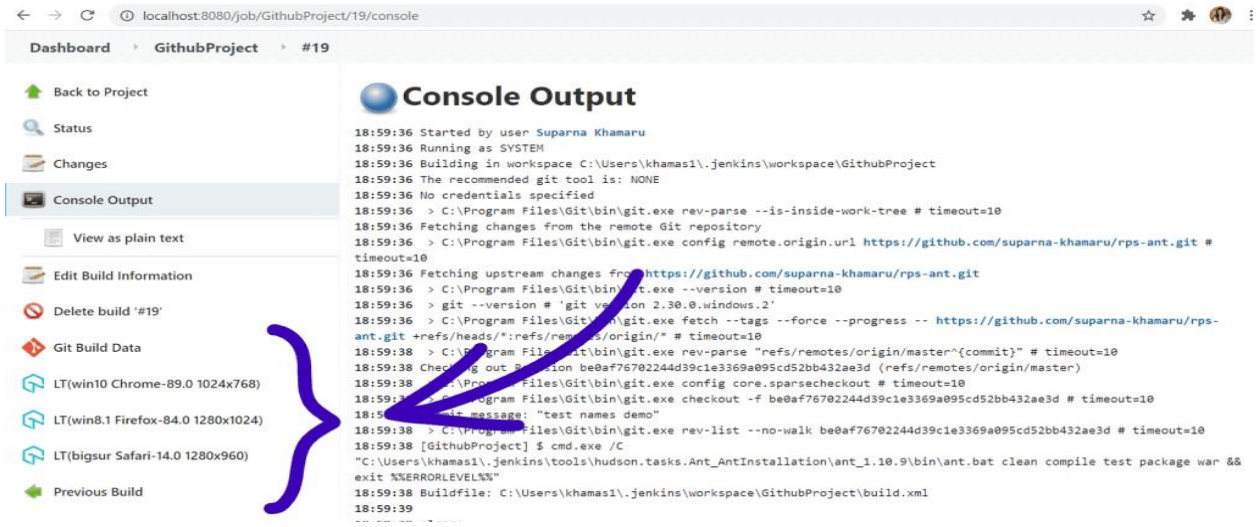


Once the new Jenkins configurations are successfully saved, the user shall be navigated back to the Project dashboard screen, where the user now clicks on the "**Build Now**" menu on the left-handside of the displayed dashboard screen.





Click on the "**Build Now**" button and allow the scheduled build to complete.



Lab06:
Explore Docker commands for content management.

Docker:

- Docker is a platform that allows us to package our applications into deployable executables — called containers, with all its necessary OS libraries and dependencies.
- Docker is an open-source software platform.
- It is a containerization platform that allows developers to create, package, run and deploy their applications on any operating system or cloud platform.
- With Docker, developers can quickly and easily create, test, and deploy applications without having to worry about compatibility issues or hardware requirements.

Docker Container:

- Docker container is an isolated environment that is running instance of Docker images.

Docker Image:

- A Docker image is a lightweight, standalone, and executable package that contains everything needed to run an application, including the code, runtime environment, libraries, and system tools.

Docker Hub:

- Docker Hub is a cloud-based repository service provided by Docker for storing and sharing Docker images.
- Docker registry is of 2 type: public and private repositories.
- Public repositories are freely accessible to anyone, while private repositories require authentication and access control.

Dockerfile:

- A Dockerfile is a text document that contains all the commands and Docker can build images automatically by reading the instructions from a Dockerfile.

Docker Commands:

1. ### docker --version
   - ➔ ***To get the information of currently installed version of docker***
   - ➔

```
C:\Shashank>docker --version
Docker version 26.1.1, build 4cf5afa
```

2. ### docker run
   - ➔ ***Docker run command is used to start a new container based on a Docker image.***
   - ➔ ***Docker first check whether image exists on your local system or not. If not, docker will pull the image from the docker registry.***
   - ➔ ***Docker will download the image file to your local system and stores it in a local repository.***

docker run -p 8085:8085 mysql

```
C:\Shashank>docker run -p 8085:8085 mysql
Unable to find image 'mysql:latest' locally
latest: Pulling from library/mysql
d9a40b27c30f: Downloading [================>                                  ]  16.25MB/48.99MB
d948328c7651: Download complete
c1e267313ede: Download complete
7478f013875a: Downloading [==================================>                ]  4.451MB/6.712MB
9221a2250289: Download complete
d1f57baa52d5: Download complete
35c3d30e8624: Waiting
```

3. ## docker images

➔ ***Docker images command will list all of the images that are currently stored on your local system***

```
C:\Shashank>docker images
REPOSITORY    TAG                 IMAGE ID       CREATED         SIZ
mysql         latest              5cde95de907d   9 days ago      586
tomcat        jre21               214192ee3a5a   2 weeks ago     299
nginx         latest              e0c9858e10ed   2 weeks ago     188
docker        latest              1acd5db2357a   5 weeks ago     366
postgres      alpine3.20          f1bf85e5d935   5 weeks ago     242
nginx         stable-alpine-perl  27f43866f4ac   6 weeks ago     85.
mysql         oraclelinux9        fcd86ff8ce8c   2 months ago    578
hello-world   latest              d2c94e258dcb   14 months ago   13.
```

➔

4. ## docker rmi imageName

➔ ***Removes/ un-tags images from the host machine (not from a registry). You cannot remove an image of a running container.***

```
C:\Shashank>docker rmi hello-world
Untagged: hello-world:latest
Untagged: hello-world@sha256:94323f3e5e09a8b9515d74337010375a456c909543e1ff1538f5
Deleted: sha256:d2c94e258dcb3c5ac2798d32e1249e42ef01cba4841c2234249495f87264ac5a
Deleted: sha256:ac28800ec8bb38d5c35b49d45a6ac4777544941199075dff8c4eb63e093aa81e
```

➔

5. ## docker container ls

➔ ***Lists all running containers on your system***

```
C:\Shashank>docker container ls
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
```

➔

6. ## docker container ls -a

➔ ***shows all the running and exited containers***

```
C:\Shashank>docker container ls -a
CONTAINER ID   IMAGE                     COMMAND                  CREATED         STATUS
                        NAMES
9e9bac7a9eef   mysql                     "docker-entrypoint.s…"   3 minutes ago   Exited (1) 3 minutes ago
                        reverent_carson
e759b4555768   postgres:alpine3.20       "docker-entrypoint.s…"   6 minutes ago   Exited (1) 5 minutes ago
                        sad_kalam
b629d0884dd9   nginx                     "/docker-entrypoint.…"   13 days ago     Exited (0) 13 days ago
                        vigorous_robinson
18e1c885a2b4   mysql:oraclelinux9        "docker-entrypoint.s…"   2 weeks ago     Exited (1) 2 weeks ago
                        strange_wu
04894a0a7d55   nginx:stable-alpine-perl  "/docker-entrypoint.…"   2 weeks ago     Exited (255) 13 days ago
0.0.0:8086->8086/tcp    kind_saha
26427b3c58bf   nginx:stable-alpine-perl  "/docker-entrypoint.…"   4 weeks ago     Exited (0) 4 weeks ago
                        pedantic_aryabhata
9247cb958ae8   postgres:alpine3.20       "docker-entrypoint.s…"   4 weeks ago     Exited (1) 4 weeks ago
                        nifty_feynman
```

➔

7. ## docker image pull <image name> →

fetches and downloads an image from the docker registry/docker Hub without starting a container. docker Hub contains many pre-built images.

```
C:\Shashank>docker image pull hello-world
Using default tag: latest
latest: Pulling from library/hello-world
c1ec31eb5944: Pull complete
Digest: sha256:1408fec50309afee38f3535383f5b09419e6dc0925bc69891e79d84cc4cdcec6
Status: Downloaded newer image for hello-world:latest
docker.io/library/hello-world:latest
```

8. ## docker images

➔ ***Lists all the images that are locally stored with the Docker engine.***

```
C:\Shashank>docker images
REPOSITORY    TAG                 IMAGE ID       CREATED         SI
mysql         latest              5cde95de907d   9 days ago      58
tomcat        jre21               214192ee3a5a   2 weeks ago     29
nginx         latest              e0c9858e10ed   2 weeks ago     18
docker        latest              1acd5db2357a   5 weeks ago     36
postgres      alpine3.20          f1bf85e5d935   5 weeks ago     24
nginx         stable-alpine-perl  27f43866f4ac   6 weeks ago     85
mysql         oraclelinux9        fcd86ff8ce8c   2 months ago    57
hello-world   latest              d2c94e258dcb   14 months ago   13
```

➔

9. ## docker start [container_id]

➔ ***Starts a docker container that has been stopped.***

```
C:\Shashank>docker start b629d0884dd9
b629d0884dd9
```

➔

10. ## docker restart [container_id]

➔ ***Restarts a container that's running or stopped.***

```
C:\Shashank>docker restart b629d0884dd9
b629d0884dd9
```

➔

11. ## docker rm <container id>

➔ ***remove the Docker containers manually.***

```
C:\Shashank>docker rm 9e9bac7a9eef
9e9bac7a9eef
```

➔

Lab07:

Develop a simple containerized application using Docker.

First create a JavaScript file with a single line of code. It is just a print statement on the console.

```
console.log('HelloDocker!');
```

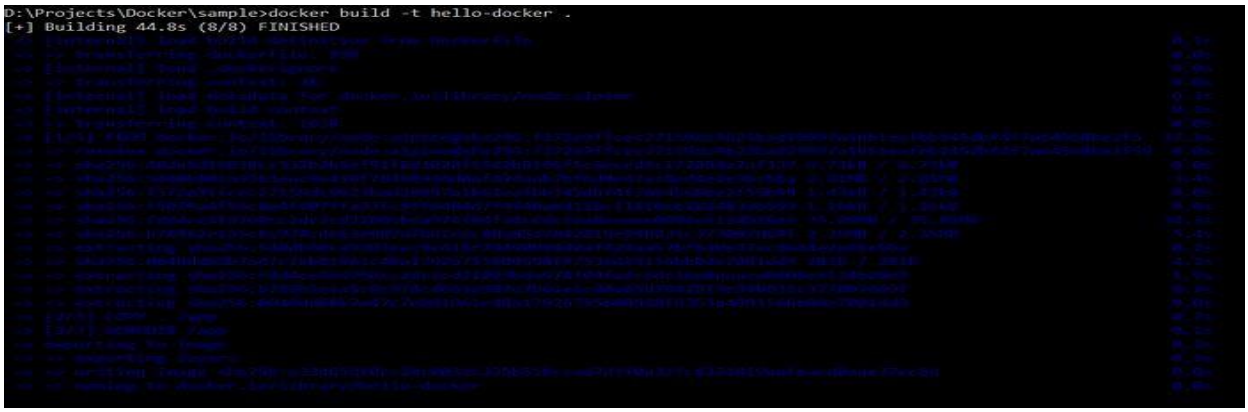
Creating the Docker file is start with the base image. This base image has a bunch of files. So, we take those files and add additional customization to them. I used Node image which builds on top of the Linux image. You can find any official images from the docker hub.

```
FROM
node:alpine
        COPY./app
        WORKDIR
        /app
        CMDnodeapp.j
        s
```

FROM—Define the base image
COPY— Copy files from the current directory to /app directory in the image WORKDIR— Set the current working directory in the image
CMD—Write the command that should be executed (Here, I execute the node command) Then we need to execute a command as below. It will build the Docker package for our application.

```
Docker build –t hello -docker.
```

The –t is a tag to identify the image. Then, the hello-docker is a specific name that can be used to access it. Finally,this command mentions the Docker file directory (.means current directory).



Then, I can see all the images on my computer with this command.

docker images

```
D:\Projects\Docker\sample>docker images
REPOSITORY                TAG       IMAGE ID       CREATED              SIZE
hello-docker              latest    e33465df0cc2   About a minute ago   113MB
docker/getting-started    latest    3ba8f2ff0727   2 months ago         27.9MB
```

Now,we can run this image on any computer that running Docker. Let's run it with this command.

docker run hello-docker

```
D:\Projects\Docker\sample>docker run hello-docker
Hello Docker!
```

Lab 08:

Integrate Kubernetes and Docker.

What is Kubernetes?

Kubernetes is an open-source orchestration tool that allows you to run and manage your container based workloads. Kubernetes K8s was Developed by google and was later donated to the Cloud Native Computing Foundation. Kubernetes helps you manage hundreds and thousands of containerized applications in different deployable environments be it physical machines, virtual machines, cloud or even hybrid environments!

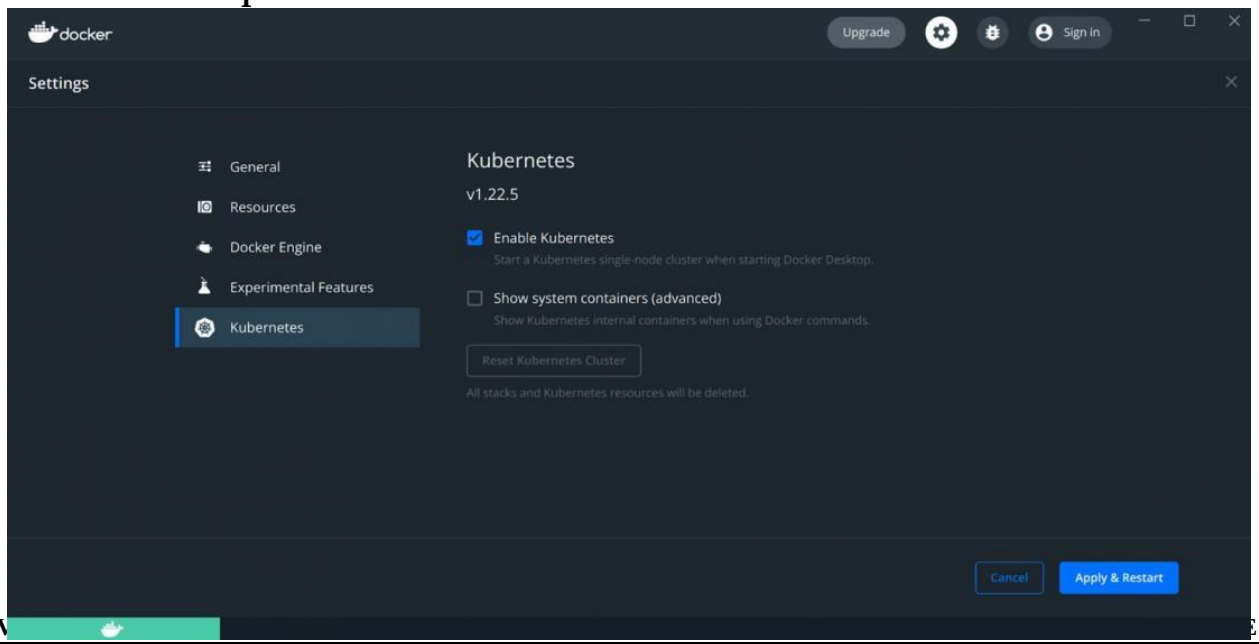
How Does Kubernetes Work?

Kubernetes was made to solve the problems of the Monolith approach of application deployment used to face. This is where the Microservice type of approach comes into the picture. To get to know why Microservice is the business as of today, have a look at Monolithic vs Microservices. Kubernetes and the Microservice approach has made the lives of so many tech-giants easier!

How Does Kubernetes Work with Docker?

You'd have figured it by now that Docker helps to "create" containers whereas K8s enables you to "manage" them at runtime. Docker is used in packing and hip your application. In the same way, K8s empowers you to deploy and scale your application. Kubernetes comes into the picture only when there are more containers that need to be managed. While the big tech-giants are in the race to adapt to Kubernetes, small startups preferably wouldn't be needing K8s for managing their applications. But, as the companies grow, their infrastructure needs will rise; hence, the number of containers will increase, which can be difficult to manage.

Kubernetes setup →



Dockerfile to Create a HelloWorld Container Image

A manifest, called a Dockerfile, describes how the image and its parts are to run in a container deployed on a host. To make the relationship between the Dockerfile and the image concrete, here'sanexample of a Dockerfile that creates a "HelloWorld" app from scratch:

```
FROM
scratchCOPY
hello
```

When you give this Dockerfile to a local instance of Docker by using the docker build command,

it creates a container image with the "HelloWorld"app installed init.

Creating aKubernetesDeployment for HelloWorld

Define a deployment manifest, commonly done with a YAML or JSON file, to tell

```
# Hello World Deployment YAML
apiVersion:apps/v1
kind: Deployment
metadata:
  name: helloworld
spec:
  selector:
       match Labels:
       app: helloworld
  template:
       meta data:
       labels:
       app: helloworld
       spec:
       containers:
       -name: helloworldimage:
       boskey/helloworld
       resources:
       limits:
             memory:
             "128Mi"cpu:"500
```

Kubernetes how to run "HelloWorld"based on the container image:

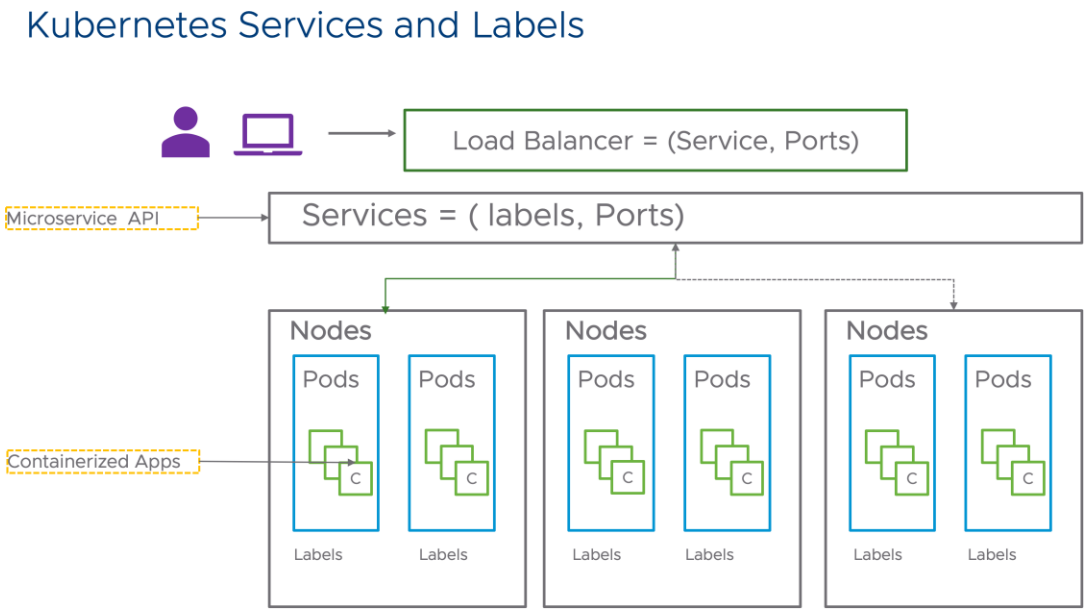
To deploy the application on a Kubernetes cluster, you can submit a YAML file using Command similar to the following.

Kubectlapply - fhttps://yourdomain.ext/application/helloworld.yaml—record

Creating a Kubernetes Service

The container is now deployed to Kubernetes but there is no way to communicate with it, the next step is to turn the deployment into a Service by establishing communication.

In Kubernetes, a Service is an abstraction which defines a logical set of pods and a policy by which to access them. This guide demonstrates a basic method of providing services to pods.



Hello World service definition

A corresponding service definition for the earlier "Hello World" deployment manifest is shown below. Notice line 5 onward. With the selector as "app: hello world" the service will forward traffic coming to port 80 on the cluster network to pods that match this label.

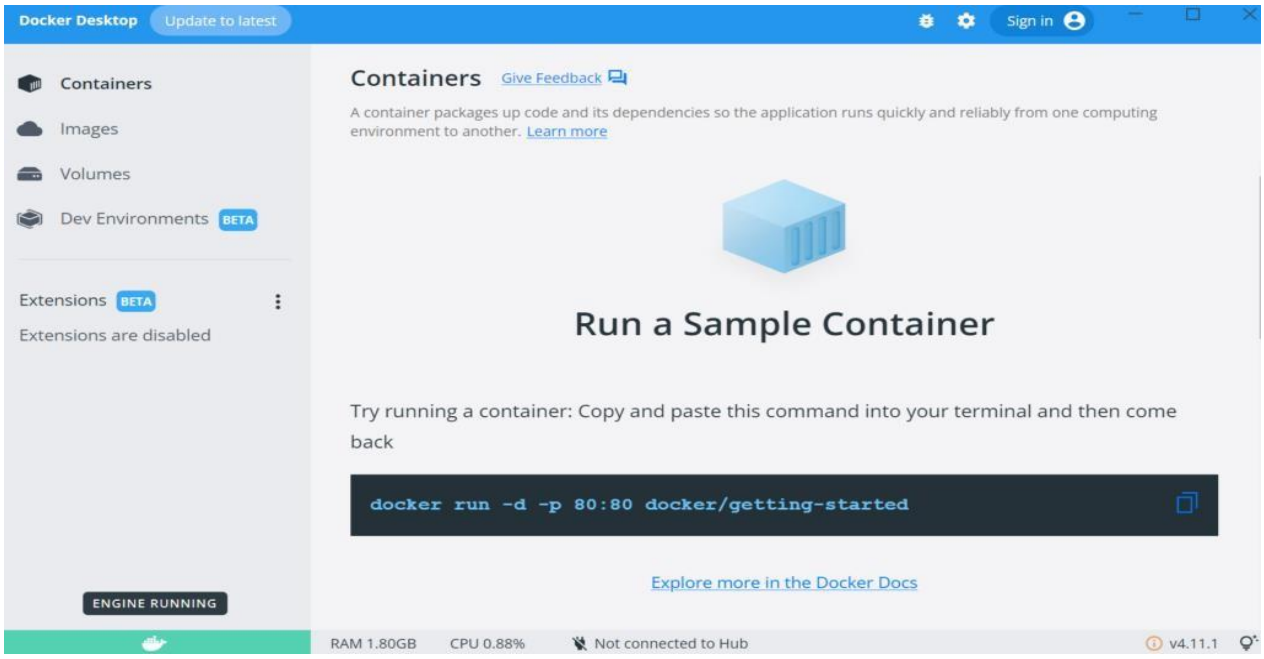
```
apiVersion:
v1kind:
Servicemetadat
a:
  name:
helloworldspec:
  selector:
      app:
  helloworldports:
  -port:80
```

Lab09:

Automate the process of running containerized application developed in exercise 7 using Kubernetes.

To get a Kubernetes Cluster up and running, we will use the following tools:

- DockerEngine: For creating, building, and deploying application images.
- A Docker Hub Account for deploying Docker images to the Docker Hub registry.
- Minikube: Minikube is Kubernetes tool. It allows you to run Kubernetes locally on your computer. It runs as a single-node Kubernetes cluster with in your local computer, making it easy to develop the Kubernetes app.
- Kubectl: Kubectl is a Kubernetes command-line (CLI) tool. It allows you to run commands related to deploying, inspecting, and managing cluster resources against Kubernetes.
- Node.js runtime: For creating aNode.js app.



```
* docker "minikube" container is missing, will recreate.
* Creating docker container (CPUs=2, Memory=2200MB) ...
* Preparing Kubernetes v1.24.3 on Docker 20.10.17 ...
  - Generating certificates and keys ...
  - Booting up control plane ...
  - Configuring RBAC rules ...
* Verifying Kubernetes components...
! Executing "docker container inspect minikube --format={{.State.Status}}"
took an unusually long time: 4.3711286s
  - Using image gcr.io/k8s-minikube/storage-provisioner:v5
* Restarting the docker service may improve performance.
  - Using image kubernetesui/metrics-scraper:v1.0.8
  - Using image kubernetesui/dashboard:v2.6.0
* Enabled addons: default-storageclass, storage-provisioner, dashboard
* Done! kubectl is now configured to use "minikube" cluster and "default" n
amespace by default
```

Once you have the PATH ready, run the following command to check if your set Kubectl is ready to execute Kubernetes commands:

Kubectl cluster-info

```
C:\Users\kim>kubectl cluster-info
Kubernetes control plane is running at https://127.0.0.1:8547
CoreDNS is running at https://127.0.0.1:8547/api/v1/namespaces/kube-system
/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info
dump'.
```

Kubernetes allows you to manage large applications. Like wise, Docker can deploy this application to a Kubernetes cluster seamlessly. For this guide, we will just create a simple Node.js app to demonstrate how you can get your application running on a Kubernetes cluster.

If you have an application reader, you can skip this step and use the application along side this guide.

First, navigate to your project directory and initialize Node.js using the following

command:

npm init -y

Just like any application, you go ahead and add dependencies/libraries to run your application. In this case, we create a minimal Node.js app. Therefore, we will use the Express library to create a simple Node.js server. Install Express as follows:

npm install express

Then create an index.js file and create the server as

follows:

```
const express = require('express');
const app=express();
const port= 3000;app.get('/',(req,res)=>{
res.send('Hello Kubernetes!!!');
});
app.listen(port,()=>{
console.log(`Server started on port ${port}`);
});
```

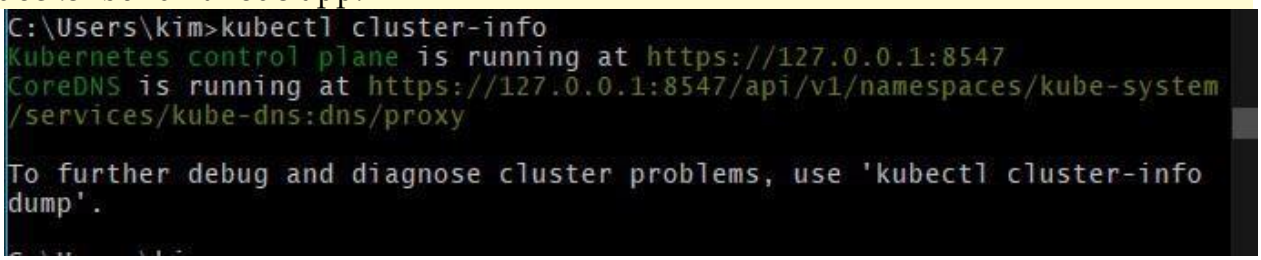
Then test the application by running the following:

node index.js

Finally, open http://localhost:3000 on the browser to test the application:

Hello Kubernetes!!!

docker build –t node-app.



Now run acontainer from this image and see that it works as

expected:

docker run –it –p 3000:3000 node-app

If you open http://localhost:3000, you should be served the same application we run locally. However,this time the application will be loaded from a Docker container.

Hello Kubernetes!!!

To deploy this application to Kubernetes, we first need to push the image to a Docker Hub registry. To do so, log into your DockerHub account using a terminal by running:

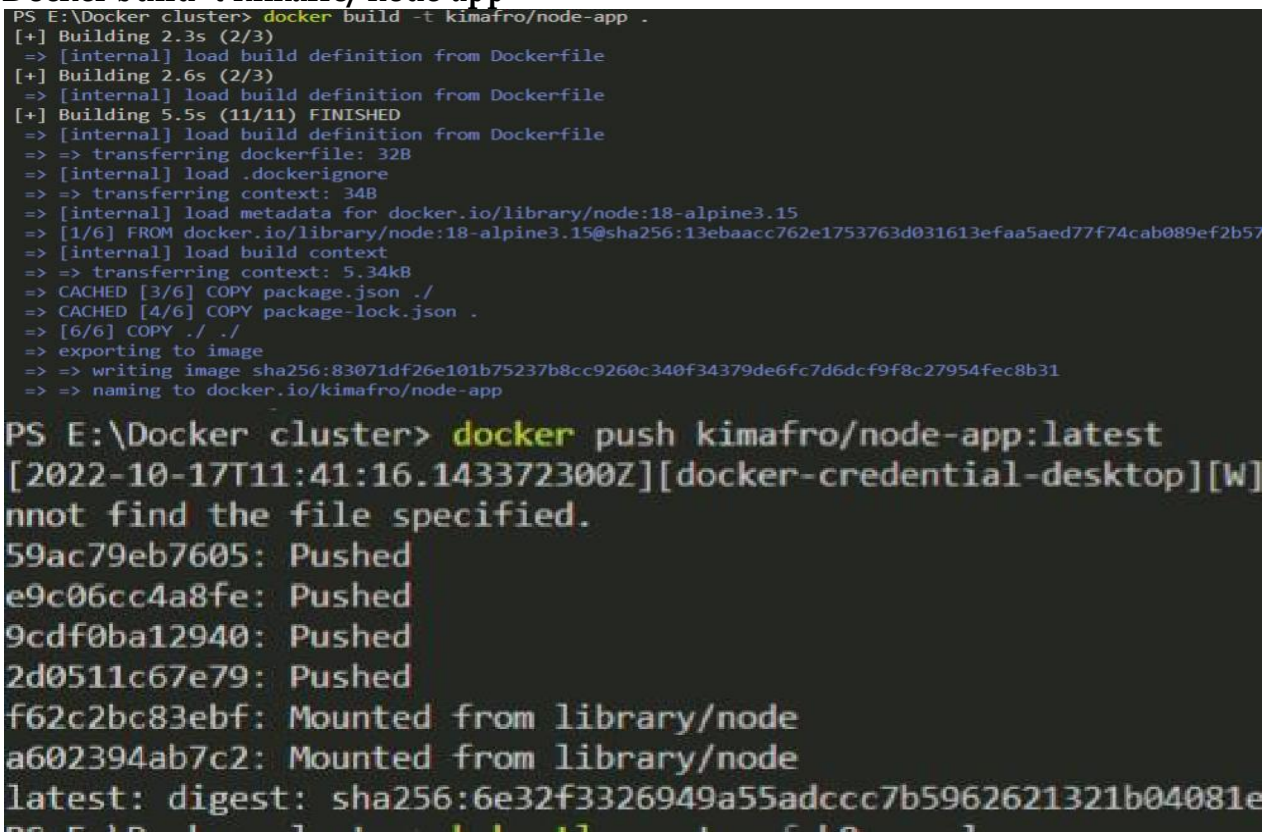
docker login

Provide your correct Docker Hub username and password. Proceed and build an image using your username as follows:

docker build –t <Enter your Docker Hub username>/node-app

Here, ensure you have the correct username added to the above command. This will form the image name of your application. Forexample:

Docker build -t kimafro/ node-app



EXPERIMENT NO.: 10.

Install and Explore Selenium for automated testing

AIM: To install the necessary software and explore Selenium WebDriver for automated web testing. This includes setting up the Java Development Kit (JDK), Selenium WebDriver, browser drivers, and an Integrated Development Environment (IDE), and then writing and executing a basic test script.

DESCRIPTION:

- Selenium is a widely-used open-source framework for automating web browsers. It provides a suite of tools for automating web applications for testing purposes but is also used for web scraping and automating repetitive tasks.
- Selenium supports a variety of programming languages through the use of language-specific bindings or drivers. The languages supported by Selenium include java, C#, python, Ruby, Javascript, perl.
- Selenium test scripts can be written in any of the supported programming languages and executed in modern web browsers.
- Selenium supports a wide range of web browsers, including:
  - **Google Chrome**
  - **Mozilla Firefox**
  - **Internet Explorer**
  - **Microsoft Edge**
  - **Safari**

A simple example demonstrating how to use Selenium WebDriver with Java to open Google's homepage.

Step1: Install JDK and set up environmental variables.

Step2: Install selenium web driver

Selenium WebDriver is a set of APIs that allow you to control web browsers.

Go to https://www.selenium.dev/downloads/ and download java driver under selenium clients and webdriver language bindings. Extract the zip file and place in C:\Drivers\

Step3: Install browser driver.

Each supported browser has a corresponding driver that Selenium WebDriver uses to communicate with the browser. For example:

- ChromeDriver: For Google Chrome
- GeckoDriver: For Mozilla Firefox
- IEDriverServer: For Internet Explorer
- SafariDriver: For Safari.

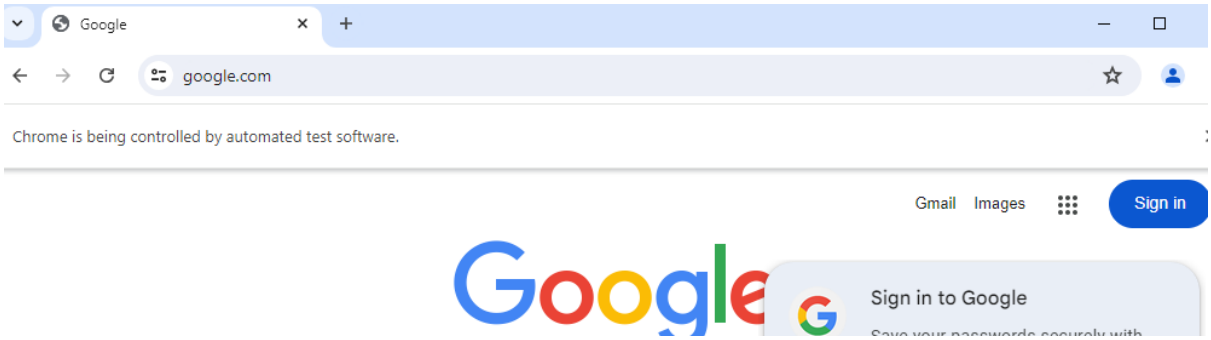
For chrome driver, check the google chrome version and go to https://googlechromelabs.github.io/chrome-for-testing/ . Download the stable release, Extract the zip file and place in C:\Drivers\selenium-java-<version>\

Step4: Create a simple test application test.java

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class Test {
    public static void main(String[] args) {
        // Set the path to the ChromeDriver executable
        System.setProperty("webdriver.chrome.driver", "C:\\Drivers\\selenium-java-4.23.0\\chromedriver-win64\\chromedriver.exe");

        // Initialize the ChromeDriver
        WebDriver driver = new ChromeDriver();

        // Open a webpage
        driver.get("https://www.google.com");

        // Print the title of the page
        System.out.println("Page title is: " + driver.getTitle());

        // Close the browser
        driver.quit();
    }
}
```

Step5: Compile and run the program with the following commands

```
C:\Users\kuppa\testing>javac -cp "C:\Drivers\selenium-java-4.23.0\*" Test.java

C:\Users\kuppa\testing>java -cp "C:\Drivers\selenium-java-4.23.0\*;." Test
Page title is: Google
```



EXPERIMENT NO.: 11.
Write a simple program in JavaScript and perform testing using Selenium

AIM: To write a selenium test case for a simple JavaScript program that increments a counter on a webpage. The javaScript program adds a counter functionality to a button. The goal is to test this functionality using Selenium.

Description:

Step1: Write JavaScript Code: Develop a web application with JavaScript that performs certain tasks (e.g., incrementing a counter).

Step2: Write Selenium Test in Java: Use Selenium to write a test in Java that interacts with the web page, triggering JavaScript actions and verifying outcomes.

Step3: Download the following jar files and place them in c:\Drivers

- **JUnit:** Defines and runs test cases in Java, checking that the Selenium WebDriver interactions produce the expected outcomes. Install Junit jar and place it in c:\Drivers
- **Hamcrest:** Provides expressive assertion methods for verifying conditions in your tests. Install Hamcrest jar and place it in c:\Drivers

counter.html

```html
<!DOCTYPE html>
<html>
<head>
    <title>Simple JavaScript Program</title>
</head>
<body>
    <p id="output">0</p>
    <button id="increment-button">Increment</button>
    <script>
        const output = document.getElementById("output");
        const incrementButton = document.getElementById("increment-button");
        let count = 0;
        incrementButton.addEventListener("click", function() {
            count += 1;
            output.innerHTML = count;
        });
    </script>
</body>
</html>
```

Main.java

```java
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;

public class Main {
    private WebDriver driver;

    @Before
    public void setUp() {
        System.setProperty("webdriver.chrome.driver", "C:\\Drivers\\selenium-java-4.23.0\\chromedriver-win64\\chromedriver.exe");
        driver = new ChromeDriver();
    }

    @Test
    public void testIncrementButton() throws InterruptedException {
        driver.get("file:///C:/Users/sample/counter.html");
        driver.findElement(By.id("increment-button")).click();
        String result = driver.findElement(By.id("output")).getText();
        assert result.equals("1") : "Expected output to be 1, but got " + result;
        Thread.sleep(5000);
    }

    @After
    public void tearDown() {
        driver.quit();
    }
}
```

Step4: compile and run the program.

```
C:\Users\sample>javac -cp "C:\Drivers\selenium-java-4.23.0\*" Main.java

C:\Users\sample>java -cp "C:\Drivers\selenium-java-4.23.0\*;." org.junit.runner.JUnitCore Main
JUnit version 4.13.2
.
Time: 10.976

OK (1 test)
```


Simple JavaScript Program
File
C:/Users/sample/counter.html
Chrome is being controlled by automated test software.
1
Increment

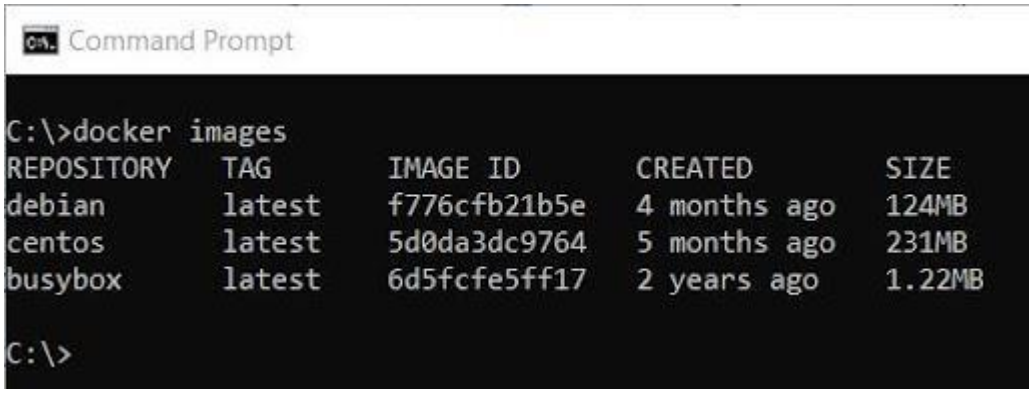
Lab12:

Develop test cases for the above containerized application using selenium.

Step1: Pull the docker image

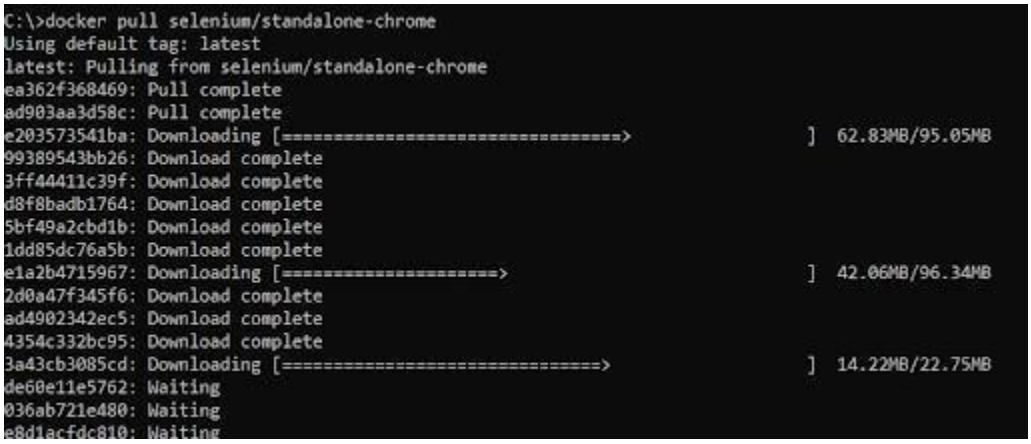
To get a list of all the already existing images on the system, run the following command in the command prompt:

```
docker images
```



If you do not have the selenium stand alone -chrome docker image, run the following command to download a copy of the image on to the system.

```
docker pull selenium/standalone-chrome
```



Upon rerunning the command docker images, selenium/standalone-chrome image appears in the list.

Step2: Running the Selenium Web driver Docker container

Upon pulling the selenium/standalone-chrome image on to the system, start the container

by running the following command:

```
docker run-d-p 4444:4444 -v/dev/shm:/dev/shmselenium/standalone-chrome
```

```
C:\Users\bshrikanth>docker run -d -p 4444:4444  selenium/standalone-chrome
837f375519d620fb2f53f70ded60eaabf87756e5819757a736df1aca62fe1bf4
```

Starting Docker Container

The command, when run, will return the Container ID.

Open the browser and navigate to http://localhost:4444/. It reflects Selenium Grid UI, as shown below.

Step3: Creating a sample test file

Selenium supports tests written in different languages of which Java and Python are most popularly used. In this example, using Python to create Selenium Test.

```
from selenium import webdriver
```

To open file in chrome browser, Chrome driver is necessary. Therefore, initializing Chrome Options to declare any connection and driver options necessary while instantiating a browser instance.

```
options =

webdriver.ChromeOptions()options.add_argument('

--ignore-ssl-errors=yes')
```

Creating an instance of the Remote webdriver and passing the selenium end point and chrome options defined in the previous step.

```
driver =

webdriver.Remote(command_executor='http://localhost

:4444/wd/hub',options=options

)
```

Navigate to the Browser Stack web site and click on the Get Started for Free button. Inducing waittime between the two actions allows viewing the execution of tests.

```
driver.get("https://www.browserstack.com/")

driver.find_element_by_link_text("Getstartedfree").click()
```

seleniumDockerTest.py

```
from selenium import webdriver

import time



print("Test Execution Started")

options= webdriver.ChromeOptions()

options.add_argument('--ignore-ssl-errors=yes')

options.add_argument('--ignore-certificate-errors')

driver =

webdriver.Remote(command_executor='http://localhost

:4444/wd/hub',options=options

)

#maximize the window size

driver.maximize_window()t

ime.sleep(10)

#navigate to browserstack.com

driver.get("https://www.browserstack.com/")

time.sleep(10)

#clickontheGetstartedforfreebutton
```

```
driver.find_element_by_link_text("Getstartedfree").click()time.sleep(10)

#close the browser

driver.close()driv

er.quit()

print("TestExecutionSuccessfullyCompleted!")
```

Step4 : Executing the test case

Python test case file can be run using either an IDE or command prompt. To run it from the command prompt, open a command prompt and run the following command:

```
python<filename>
```