

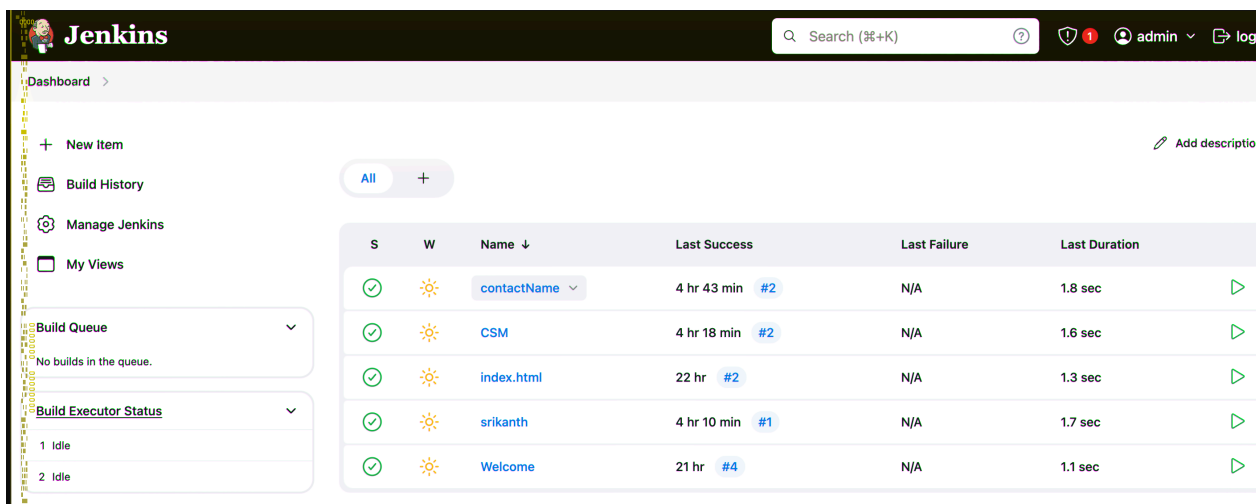
Program 5: Demonstrate continuous integration and development using Jenkins.

CI/CD pipeline with a GitHub repository:

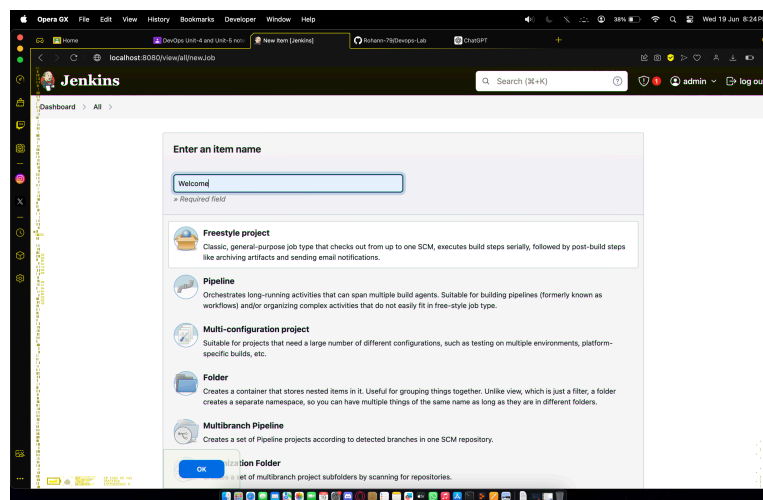
Description:

Continuous Integration (CI) and Continuous Development (CD) are practices in software development that aim to automate and streamline the process of building, testing, and deploying software.

- Go to Dashboard click on new item and give the item a name.



- Select the item as freestyle project and then click OK.



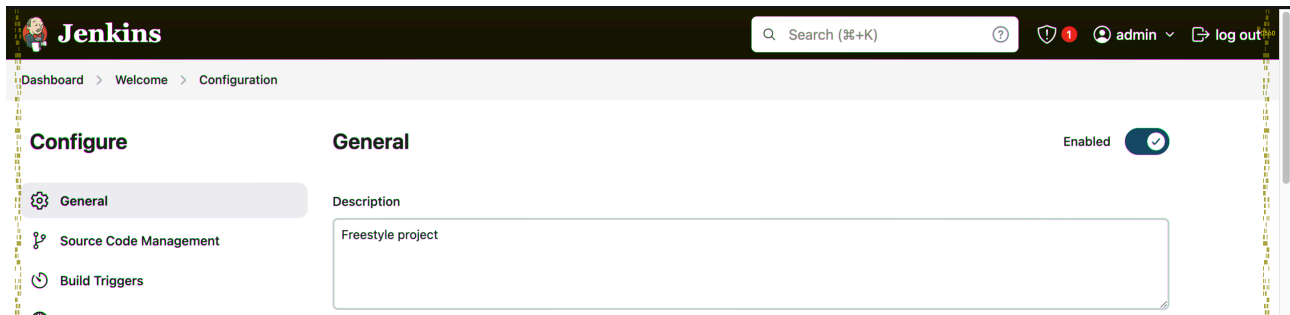
Laboratory Record of
DEVOPS

Sheet No. _____

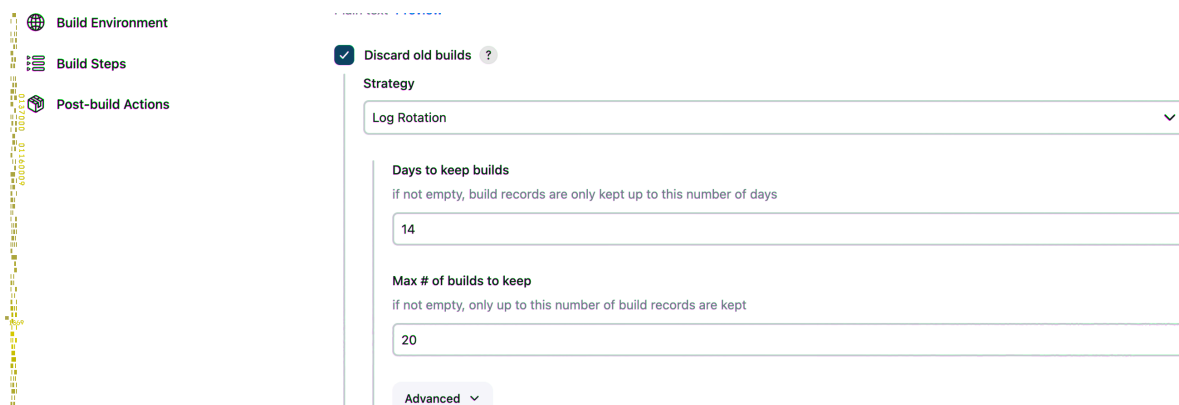
Experiment No. _____

Date _____

- Click on Configure-> General -> Description(Give Some description



- Select discard old builds(radio buttons)
Strategy: Select it as log rotation days to keep bills (14) and maximum number of Bills to keep 20



- Select Github project -> Goto Github repository and copy the link. Go to project URL and place the Github URL in the text box



Roll No. 21261A6631

MGIT

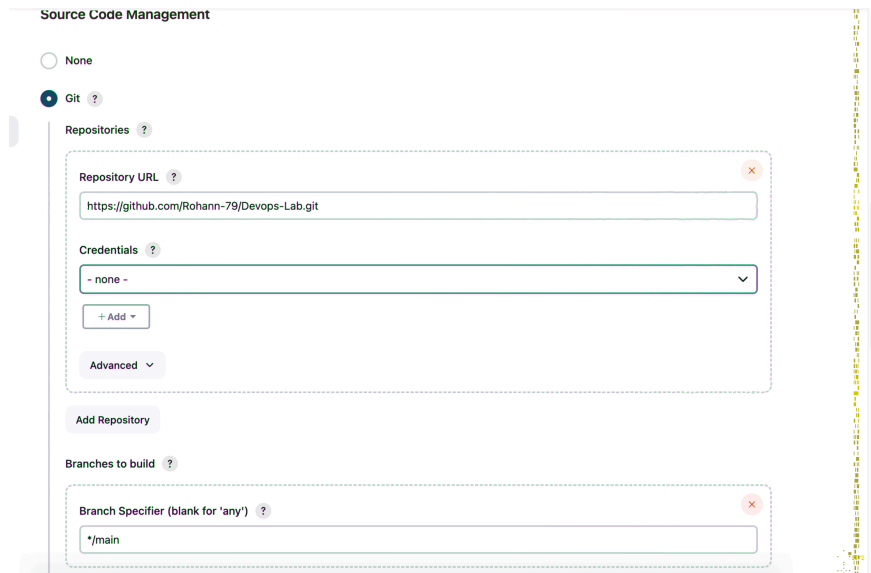
Laboratory Record of
DEVOPS

Sheet No. _____

Experiment No. _____

Date _____

- In source Code management select git and give the Github URL.



Source Code Management

☐ None

☒ Git ?

Repositories ?

Repository URL ?

Credentials ?

+ Add

Advanced

Add Repository

Branches to build ?

Branch Specifier (blank for 'any') ?

- Build Triggers select 3rd Checkbox(Build Periodically)-> Schedule, In that textbox we must give time duration

TZ=IST
H 21 6 * 0

Build Triggers

☐ Trigger builds remotely (e.g., from scripts) ?

☐ Build after other projects are built ?

☒ Build periodically ?

Schedule ?

TZ=IST
H 21 6 * 0

Would last have run at Sundav. 6 August. 2023. 9:55:44 pm India Standard Time: would next run at Sundav. 6 October. 2024. 9:55:44 pm

Roll No. 21261A6631

MGIT

Laboratory Record of

DEVOPS

Sheet No. _____

Experiment No. _____

Date _____

Click on Save now. Click on Build now. Click on Control Output.

Output



Console Output



```
Started by user admin
Running as SYSTEM
Building in workspace /Users/krohann/.jenkins/workspace/Welcome
The recommended git tool is: NONE
No credentials specified
> git rev-parse --resolve-git-dir /Users/krohann/.jenkins/workspace/Welcome/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/Rohann-79/Devops-Lab.git # timeout=10
Fetching upstream changes from https://github.com/Rohann-79/Devops-Lab.git
> git --version # timeout=10
> git --version # 'git version 2.39.3 (Apple Git-146)'
> git fetch --tags --force --progress -- https://github.com/Rohann-79/Devops-Lab.git +refs/heads/*:refs/remotes/origin/* #
timeout=10
> git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision 41bc08bcdcf290261b3b973f2197189c63b881b6e (refs/remotes/origin/main)
> git config core.sparsecheckout # timeout=10
> git checkout -f 41bc08bcdcf290261b3b973f2197189c63b881b6e # timeout=10
Commit message: "Add files via upload"
First time build. Skipping changelog.
Finished: SUCCESS
```

Roll No. 21261A6631

MGIT

Program 6: Explore Docker commands for content management.

Description: Docker is a popular platform that allows you to create and manage containers, which are lightweight, isolated environments for running applications. When it comes to content management, Docker provides several useful commands for efficiently managing containerised content.

- Run command: this is used to run a container from an image by specifying the Image ID or the Repository and/or Tag name.

```
$ docker run {image}
```

e.g \$ docker run nginx

if it already exists, the command runs an instance `nginx`

if it does not exist, it goes out to the docker hub (by default) and pulls the image down.

- ps command: this command lists all running containers and some basic information about them.

```
$ docker ps [option]
```

In option you can use various flags. To get information about the flags

```
$ docker ps --help
```



```
$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
133f5e0267a5	nginx	"/docker-entrypoint..."	10 seconds ago	Up 10 seconds

- ls command :like ps command, ls command can also be used for listing containers. –a flag can be used to list all containers.

```
$ docker container ls
```

- stop command: this command is used to stop a running container. After giving stop command can check with ps command, whether the container has stopped.

```
$ docker stop {container-id}
```

- rm command: this command removes a stopped or exited container.

\$ docker rm {CONTAINER NAME or ID}

- exec command: this command is used to go inside a running container. This is useful to debug running containers or do other work within a container

\$ docker exec -it {container} {command}

- logs command: in case a container is launched in detached mode and you want to see its logs, you can use logs command.

\$ docker logs {CONTAINER NAME or ID}

- cp command: to copy files between a container and localhost filesystem, this command is used.

\$ docker container cp {CONTAINER NAME or ID:SRC_PATH} {DEST_PATH}|-

- export command: this command exports the filesystem of a container as a TAR file.

\$ docker container export {CONTAINER NAME or ID}

- inspect command: this command gives detailed information about a container.

\$ docker inspect {CONTAINER NAME or ID}

- kill command: this command kills a running container with an option -signal or -s flag. Multiple containers can also be specified to be killed in one go.

\$ docker kill {CONTAINER NAME or ID} [--signal VAL]

Program 7: Develop a simple containerised application using Docker**Program:**

Choose any application on which you want to make a container using Docker. Here the file is 'package.json'.

- Create a file named dockerfile in the same folder as the file 'package.json' with the following contents.

```
# syntax=docker/dockerfile:1
FROM node:12-alpine
RUN apk add --no-cache python2 g++ make
WORKDIR /app
COPY . .
RUN yarn install --production
CMD ["node", "src/index.js"]
EXPOSE 3000
```

- Open a terminal and go to the 'app' directory with the 'Dockerfile'. Now build the container image using the 'docker build' command.

```
$ docker build -t getting-started
```

This command used the Dockerfile to build a new container image. You might have noticed that a lot of "layers" were downloaded. This is because we instructed the builder that we wanted to start from the node:12-alpine image. But, since we didn't have that on our machine, that image needed to be downloaded.

After the image was downloaded, we copied in our application and used yarn to install our application's dependencies. The CMD directive specifies the default command to run when starting a container from this image.

Finally, the -t flag tags our image. Think of this simply as a human-readable name for the final image. Since we named the image getting-started, we can refer to that image when we run a container

The '.' at the end of the docker build command tells Docker that it should look for the Dockerfile in the current directory.

Laboratory Record of

DEVOPS

Sheet No. _____

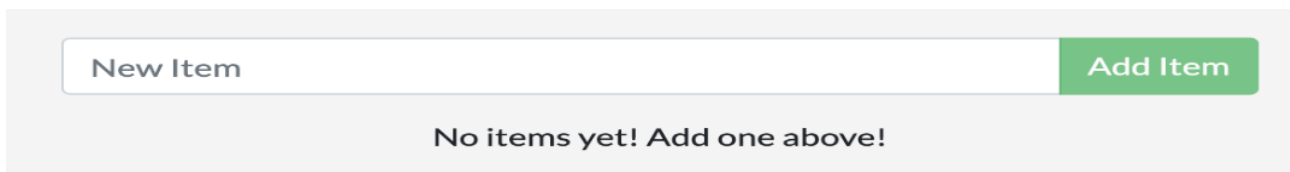
Experiment No. _____

Date _____

- Start your container using the docker run command and specify the name of the image we just created:
`$ docker run -dp 3000:3000 getting-started`

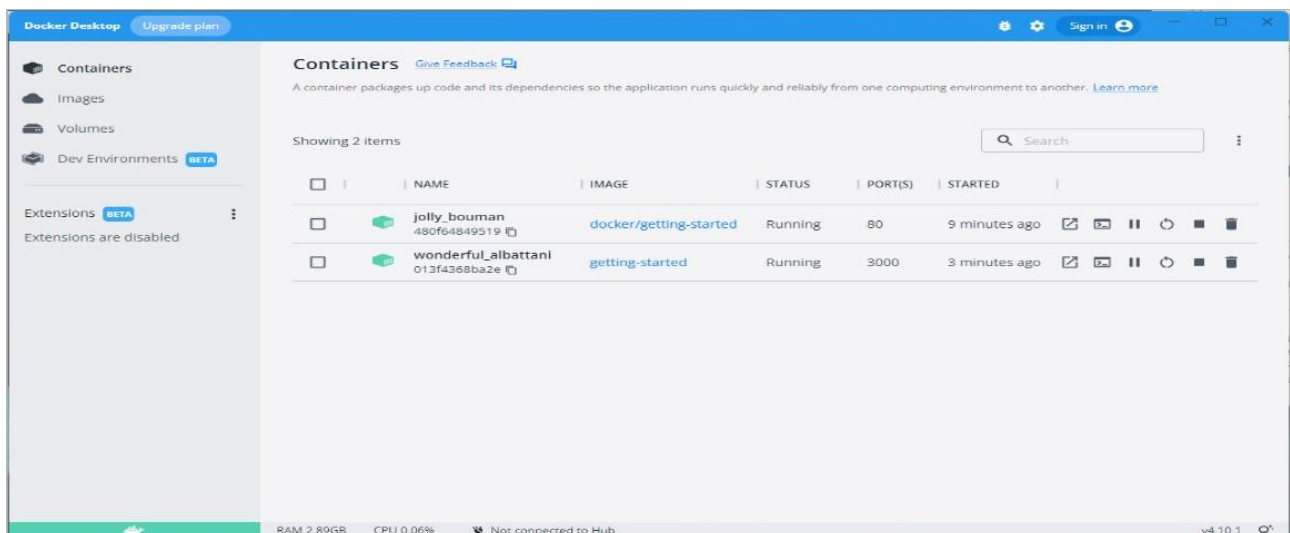
Remember the -d and -p flags? We're running the new container in "detached" mode (in the background) and creating a mapping between the host's port 3000 to the container's port 3000. Without the port mapping, we wouldn't be able to access the application.

- After a few seconds, open your web browser to <http://localhost:3000>. You should see our app.



Go ahead and add an item or two and see that it works as you expect. You can mark items as complete and remove items. Your frontend is successfully storing items in the backend.

Now the Docker dashboard will show two containers running.



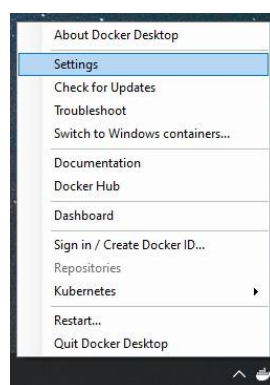
Roll No. 21261A6631

MGIT

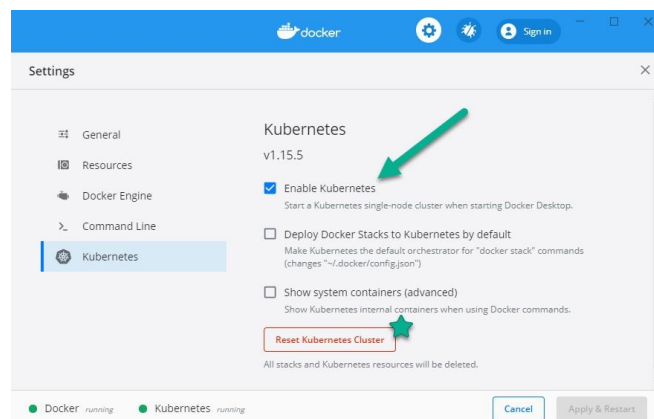
Program 8: Integrate kubernetes and Docker.

Install Docker desktop, enable kubernetes. Kubernetes itself runs in containers. When you deploy a Kubernetes cluster you first install Docker (or another container runtime like [containerd](#)) and then use tools like [kubeadm](#) which starts all the Kubernetes components in containers. Docker Desktop does all that for you.

Make sure you have Docker Desktop running - in the taskbar in Windows and the menu bar on the Mac you'll see Docker's whale logo. Click the whale and select *Settings*:



Click on Kubernetes and check the Enable Kubernetes checkbox:



Verify your Kubernetes cluster: like Docker uses 'docker' and 'docker-compose' commands to manage containers, kubernetes uses tool 'kubectl' to manage apps. Docker desktop installs kubectl too.

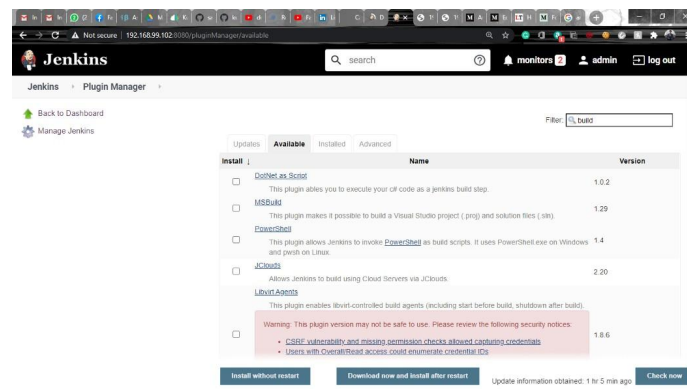
Check the state of Docker desktop cluster:

```
kubectl get nodes
```

Program 9: Automate the process of running containerised application developed in exercise 7 using Kubernetes.

Steps to be done :

- Create a job chain of Job1, Job2, Job3, Job4 using build pipeline plugin in Jenkins
 - Job1: pull the github repo automatically when some developers push the repo to GitHub.
 - Job2: automatically Jenkins should start respective language interpreter installed image container to deploy code on top of Kubernetes; expose your pod so the testing can be done; make the data to remain persistent.
 - Test your app if it is working.
-
- Load your Jenkins server and restart the services
\$ systemctl restart Jenkins
 - Creating jobs pipeline. Install Build plugin in Jenkins by going to Manage Jenkins → Manage plugins → Available → Search for build plugin, install and restart.



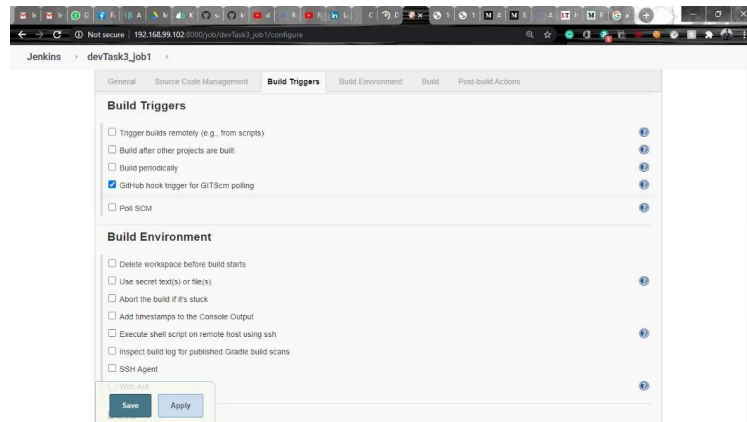
- Create job 1 to pull the GitHub repo automatically. Select SCM as Git and provide the URL of GitHub repo which will help this job to pull the code from GitHub repo.
- In Build Trigger, select [GitHub hook trigger](#) which makes the job pull the code only when there are some changes made in GitHub repo.

Laboratory Record of
DEVOPS

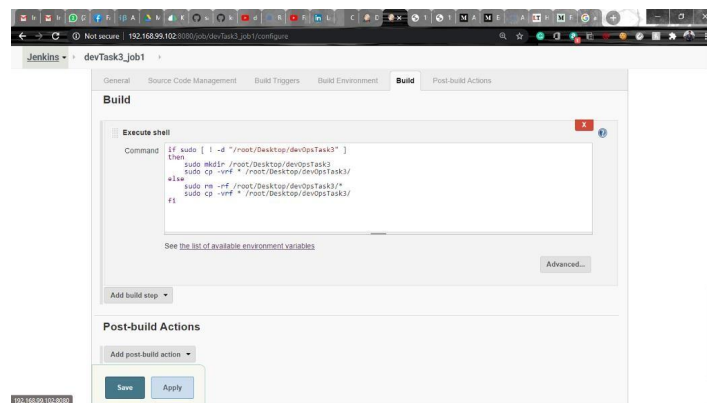
Sheet No. _____

Experiment No. _____

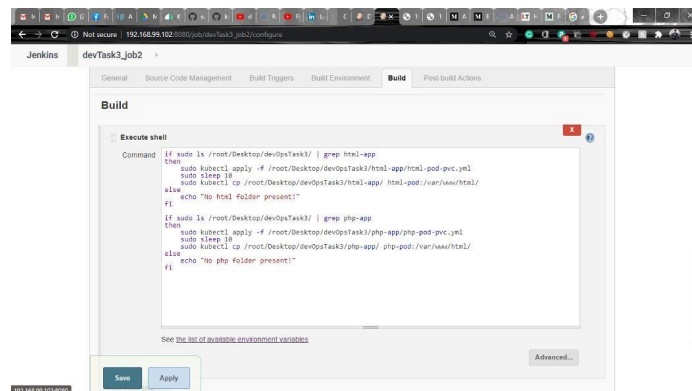
Date _____



- In Build, select Execute shell to perform following commands which will create a directory in the container and copy all the files from the GitHub repo to this directory.



- In Build, select Execute shell for performing this script which will automatically start the respective language interpreter installed image container to deploy code on top of Kubernetes with persistent storage and also exposing the pod for testing.



Roll No. 21261A6631

MGIT

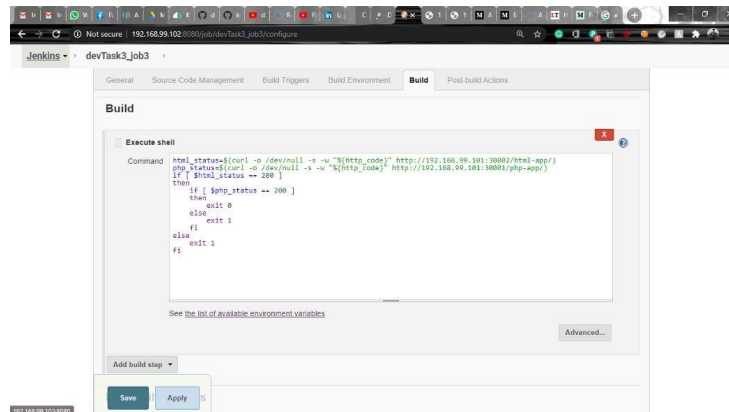
Laboratory Record of
DEVOPS

Sheet No. _____

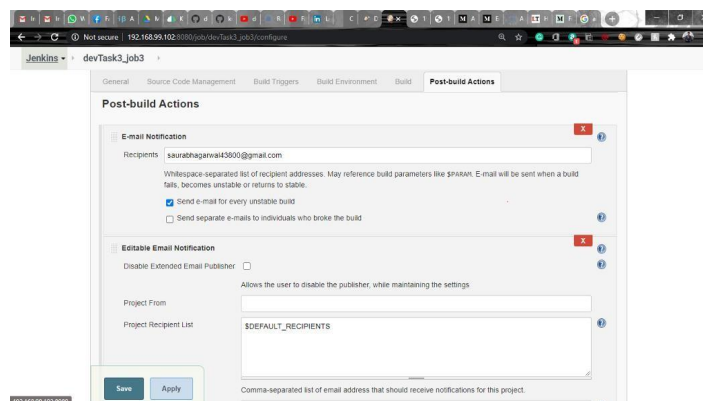
Experiment No. _____

Date _____

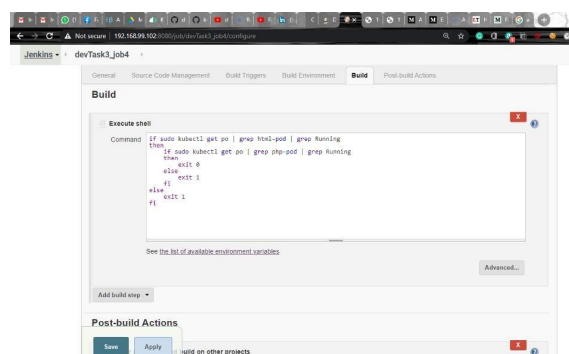
- Create job3 to test the app, if the app is not working send email to the developer with error message. In Build, select Execute shell for performing this script which will test the websites that they are running properly or not.



- In Post-build Actions, select E-mail notifications and write e-mail address, to notify the failure.



- In Build, select Execute shell to perform the following script so that it will continuously monitor if the pods are running or not and if fails it will make the job state as a failure.



Roll No. 21261A6631

MGIT

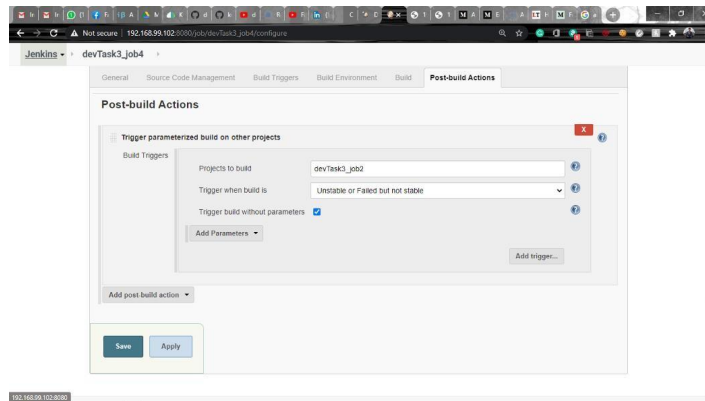
Laboratory Record of
DEVOPS

Sheet No. _____

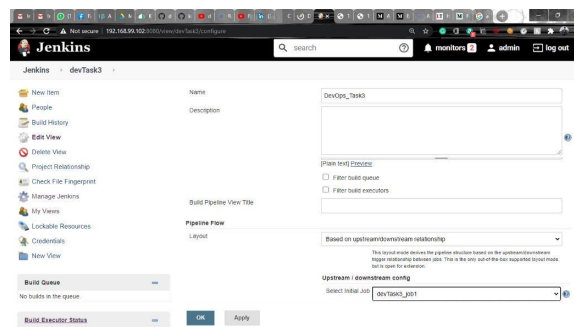
Experiment No. _____

Date _____

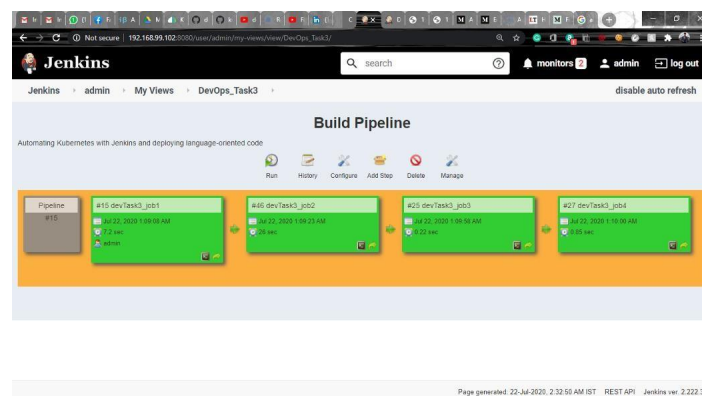
- In Post-build Actions, select Trigger parameterised build on other projects so that if the current job fails or unstable then it will automatically run the job2 to redeploy the application.



- Creating the build pipeline. Create a new view with build pipeline and give some name. Select the initial job as Job1(e.g. devTask3_job1)



- Run the build pipeline

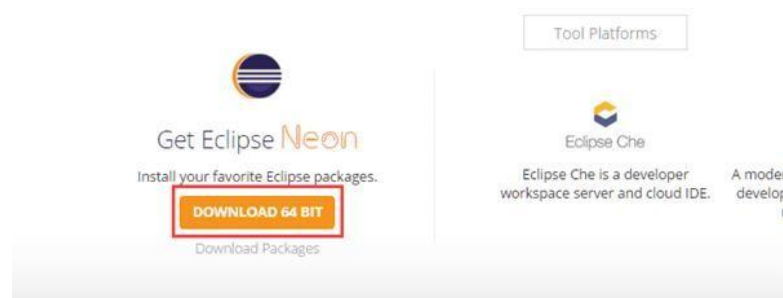


Roll No. 21261A6631

MGIT

Program 10: Install and explore Selenium for automated testing

Prerequisites: Java development kit. Install Eclipse IDE for java developers. <https://www.eclipse.org/downloads/>



Download the Selenium Java client driver. <https://www.selenium.dev/downloads/>

Selenium Client & WebDriver Language Bindings

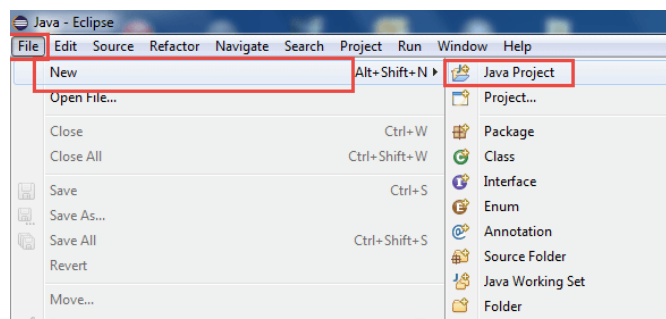
In order to create scripts that interact with the Selenium Server (Remote WebDriver) or create local Selenium WebDriver scripts, you need to make use of language-specific client drivers.

While language bindings for [other languages exist](#), these are the core ones that are supported by the main project hosted on GitHub.

LANGUAGE	VERSION	RELEASE DATE	
Ruby	3.142.6	October 04, 2019	Download ↴
JavaScript	4.0.0-alpha.5	September 08, 2019	Download ↴
Java	3.141.59	November 14, 2018	Download ↴
Python	3.141.0	November 01, 2018	Download ↴
C#	3.14.0	August 02, 2018	Download ↴

Configure Eclipse IDE with webDriver.

Launch eclipse.exe, create a new project through File>New>Java Project. Name the project as “newproject”.



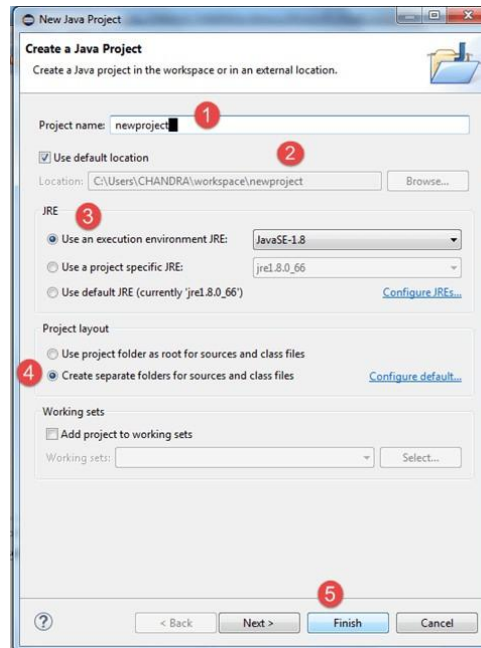
Laboratory Record of
DEVOPS

Sheet No. _____

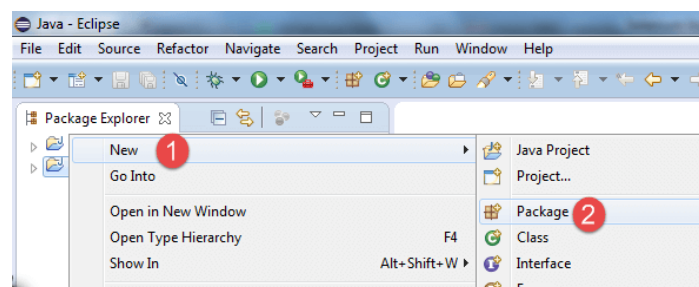
Experiment No. _____

Date _____

Enter project name, location to save project, select an execution JRE, select layout project option, click on Finish button.

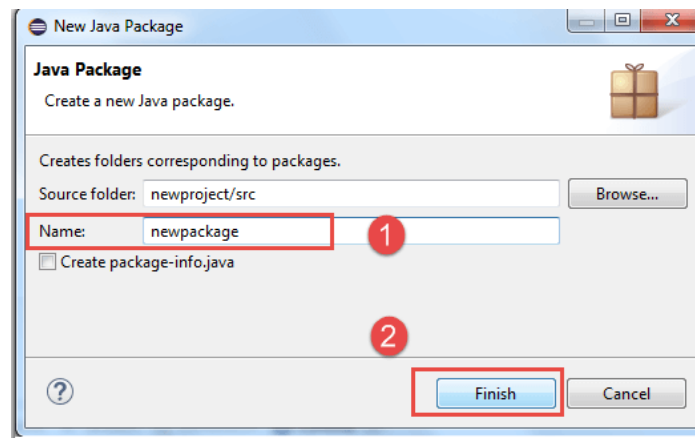


Make new package as “newpackage”.

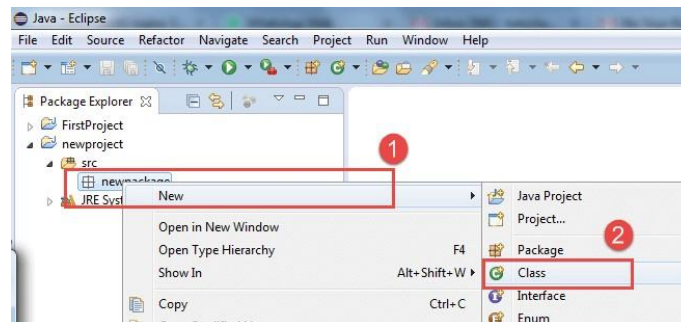


Roll No. 21261A6631

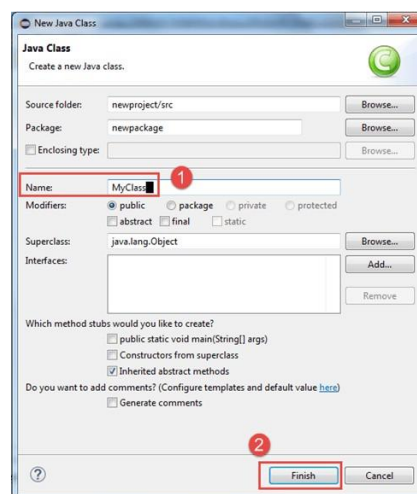
MGIT



Create a new Java class under newpackage by right-clicking on it and then selecting- New > Class, and then name it as “MyClass”. Your Eclipse IDE should look like the image below.



Give name of the class, click on finish button.



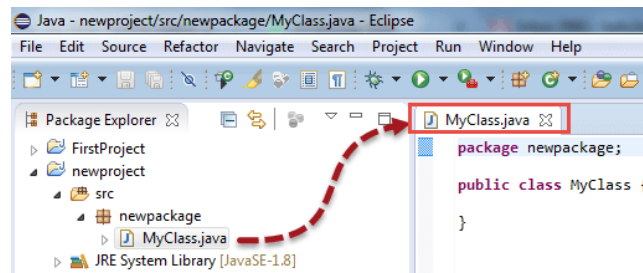
Laboratory Record of
DEVOPS

Sheet No. _____

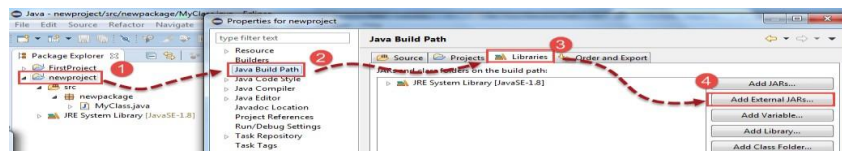
Experiment No. _____

Date _____

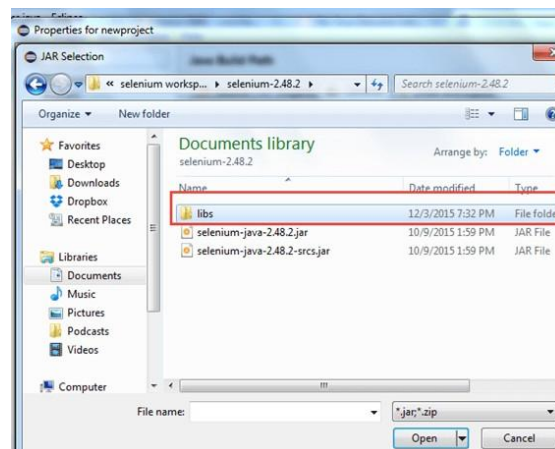
After creating the class:



Now selenium webdriver's into Java Build Path.



When you click on “Add External JARs..” It will open a pop-up window. Select the JAR files you want to add.



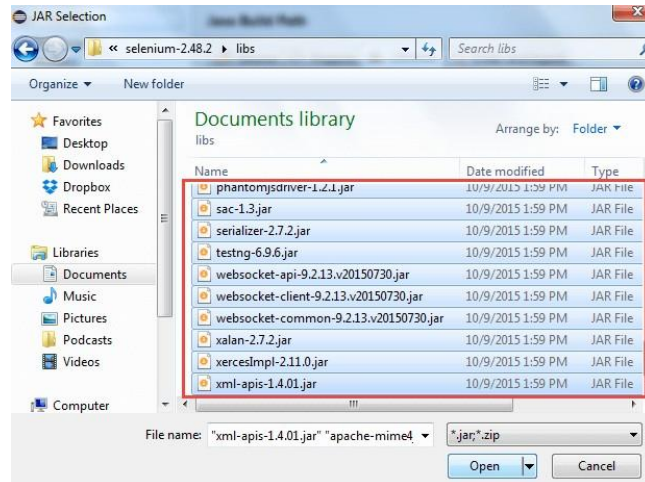
Laboratory Record of
DEVOPS

Sheet No. _____

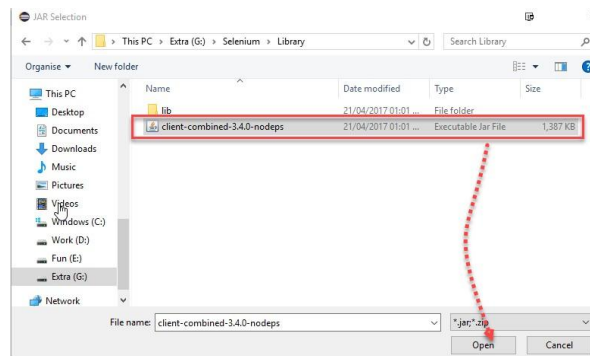
Experiment No. _____

Date _____

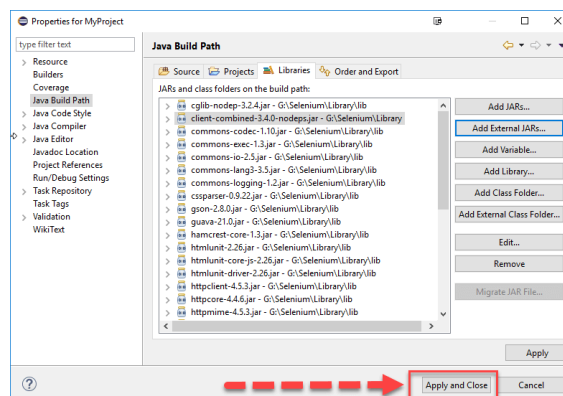
Select all files inside the lib folder.



Select files outside lib folder.



Once done, click “Apply and Close” button.



Roll No. 21261A6631

MGIT

Laboratory Record of

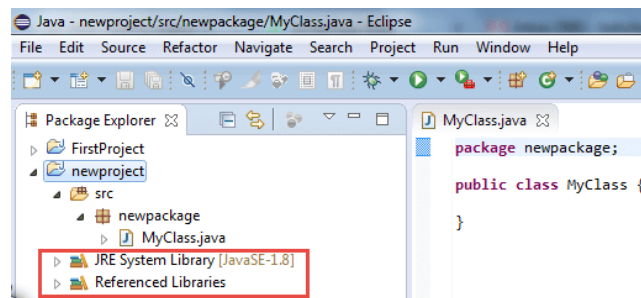
DEVOPS

Sheet No. _____

Experiment No. _____

Date _____

Add all the JAR files inside and outside the “libs” folder. Your Properties dialog should now look similar to the image below.



Finally, click OK and we are done importing Selenium libraries into our project.

Laboratory Record of

DEVOPS

Sheet No. _____

Experiment No. _____

Date _____

Roll No. 21261A6631

MGIT

Laboratory Record of

DEVOPS

Sheet No. _____

Experiment No. _____

Date _____

Roll No. 21261A6631

MGIT