

SWE 645 : Assignment 2 README File

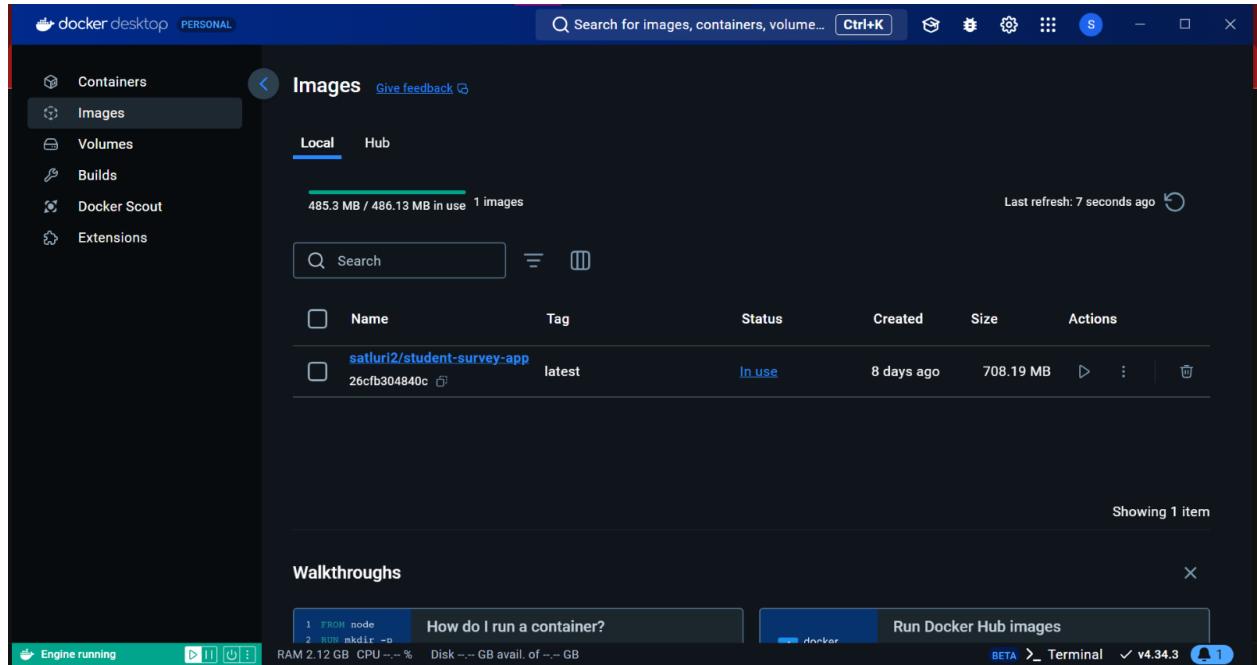
Welcome to the Readme File of SWE645 Assignment 2. We are required to follow a set of procedures in order to complete this assignment. This Readme file covers all the steps and procedures of the tasks in detail along with the screenshots. The Links/URLs used and achieved at various parts of the assignment are also enclosed in this file.

The overview of the steps are mentioned below:

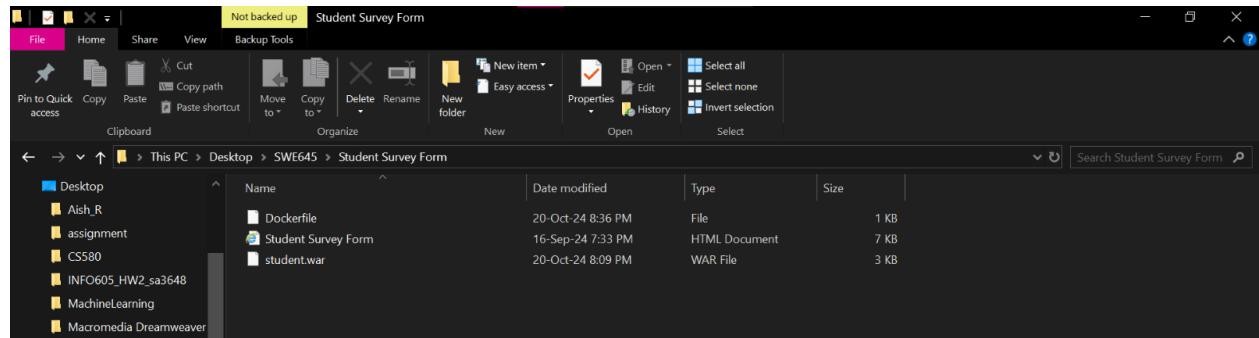
1. Using Dockerhub, we create an image of the application and push it into the dockerhub.
2. Setting up AWS EC2 instances for deploying the application on a Kubernetes cluster. We setup instances on AWS EC2 for deploying the application on a Kubernetes cluster.
3. A similar EC2 instance is created called Rancher Master in order to set up the rancher.
4. Creating a Kubernetes cluster and deploying the application.
5. We setup the Jenkins instance in order to install and run jenkins.
6. Setting up GitHub repository for deploying the webpages.
7. Creating a CI/CD pipeline and running it.

Using Dockerhub, we create an image of the application and push it into the dockerhub

- Go to <https://hub.docker.com/> and create an account. You can also download the docker desktop which is optional.
- Login to the docker desktop/ docker hub on the browser using the account created, or open the docker desktop.



- Now create a “Dockerfile” in the same directory where you have the “.war” file. In our assignment, you can check the screenshot below, where we have the “Dockerfile” and the “student.war” file in the same directory. We have obtained the .war file using Eclipse IDE.



- Now, open the “Dockerfile” in an editor and write the following commands in it, as shown in the screenshot below.

```

Dockerfile - Notepad
File Edit Format View Help
# Use the official Tomcat image from Docker Hub
FROM tomcat:9.0

# Set the working directory in the container
WORKDIR /usr/local/tomcat/webapps

# Copy your student.war file into the webapps directory of Tomcat
COPY student.war /usr/local/tomcat/webapps/

# Expose port 8080 (where Tomcat will listen)
EXPOSE 8080

# Start the Tomcat server when the container starts
CMD ["catalina.sh", "run"]

```

Now open command prompt from the directory where the war file and Dockerfile are present.

- Run the following commands to build a docker image : “ docker build -t `student-survey-app`”
- Now an image of the application is created on the local system. To run this application on the localhost, the following command must be run on the command prompt : “ docker run -it -p 8183:8080 `student-survey-app` ”.
- The application must be deployed on localhost:8183 and the server must be up and running.
- Now go to a browser and open the localhost:8183 and then append “/student” to the URL and the application webpage should be visible on the browser.
- URL : <http://localhost:8183/student/>

Docker Desktop Personal

Containers [Give feedback](#)

Container CPU usage: 0.81% / 400% (4 CPUs available)

Container memory usage: 102MB / 3.68GB

Show charts

	Name	Image	Status	Port(s)	CPU (%)	Last started	Actions
<input type="checkbox"/>	focused_w	satluri2/student-survey-app: <none>	Running	8183:8080 ↗ http://localhost:8183 %	1 minute ago	<input type="button" value=""/>	
<input type="checkbox"/>	unruffled_t	satluri2/student-survey-app: <none>	Exited (143)	8186:8080	0%	8 days ago	<input type="button" value=""/>

Showing 2 items

Engine running RAM 1.34 GB CPU ---% Disk ---% avail. of ---% GB

BETA Terminal v4.34.3 1

New Tab Student Survey

http://localhost:8183/student/

BETA Paused

Student Survey Form

Help us improve your experience!

First Name *

Last Name *

Street Address *

City *

State *

- Now, we have the application image on our local system, and we must push it into the docker hub. We will run the following commands on the command prompt from the same directory, where we have run the commands previously.
- First step is to login to dockerhub using the credentials of the account created earlier : “docker login”. Enter the password when prompted.
- Next, tag the image to be pushed and push it into the hub using the following commands : “docker tag `student-survey-app` `satluri2/student-survey-app`” followed by “docker push `satluri2/student-survey-app`”.

```
C:\Windows\System32\cmd.exe - docker push satluri2/student-survey-app
Microsoft Windows [Version 10.0.19045.5011]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Sravya Atluri\Desktop\SWE645\Student Survey Form>docker login
Authenticating with existing credentials...
Login Succeeded

C:\Users\Sravya Atluri\Desktop\SWE645\Student Survey Form>docker push satluri2/student-survey-app
Using default tag: latest
The push refers to repository [docker.io/satluri2/student-survey-app]
2d3b0ff4b2e6: Pushed
b33ac22b05ec: Pushing [=====>] 14.77MB/14.77MB
1319a08af1e7: Pushed
05c11ba7eb7a: Pushed
ff6b0b525b2f: Pushing [====>] 18.87MB/158.6MB
e0a6e3ea5965: Pushed
f2c36bff6355: Pushed
08e008c6c70a: Pushing [=====>] 13.63MB/19.75MB
ff65ddf9395b: Pushed
4f4fb700ef54: Pushed
```

```
Google X | ChatGP X | Submit X | Oracle | X | how to X | Student X | localho X | satluri2 X
C:\Windows\System32\cmd.exe - docker run -p 8183:8080 satluri2/student-survey-app
Authenticating with existing credentials...
Login Succeeded

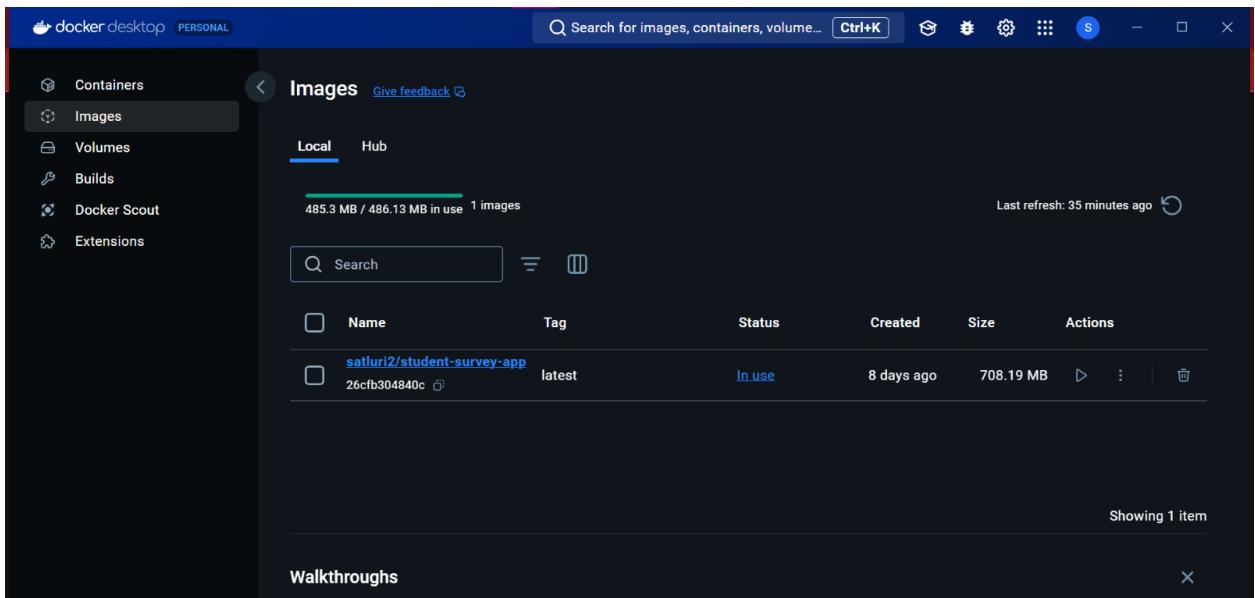
C:\Users\Sravya Atluri\Desktop\SWE645\Student Survey Form>docker push satluri2/student-survey-app
Using default tag: latest
The push refers to repository [docker.io/satluri2/student-survey-app]
2d3b0ff4b2e6: Pushed
b33ac22b05ec: Pushed
1319a08af1e7: Pushed
05c11ba7eb7a: Pushed
ff6b0b525b2f: Pushed
e0a6e3ea5965: Pushed
f2c36bff6355: Pushed
08e008c6c70a: Pushed
ff65ddf9395b: Pushed
4f4fb700ef54: Pushed
latest: digest: sha256:26cfb304840cb9c40c1c305e7e385f008021b8113694b1d5c57b7c59579b8555 size: 856

C:\Users\Sravya Atluri\Desktop\SWE645\Student Survey Form>docker pull satluri2/student-survey-app
Using default tag: latest
latest: Pulling from satluri2/student-survey-app
Digest: sha256:26cfb304840cb9c40c1c305e7e385f008021b8113694b1d5c57b7c59579b8555
Status: Image is up to date for satluri2/student-survey-app:latest
docker.io/satluri2/student-survey-app:latest

What's next:
View a summary of image vulnerabilities and recommendations → docker scout quickview satluri2/student-survey-app

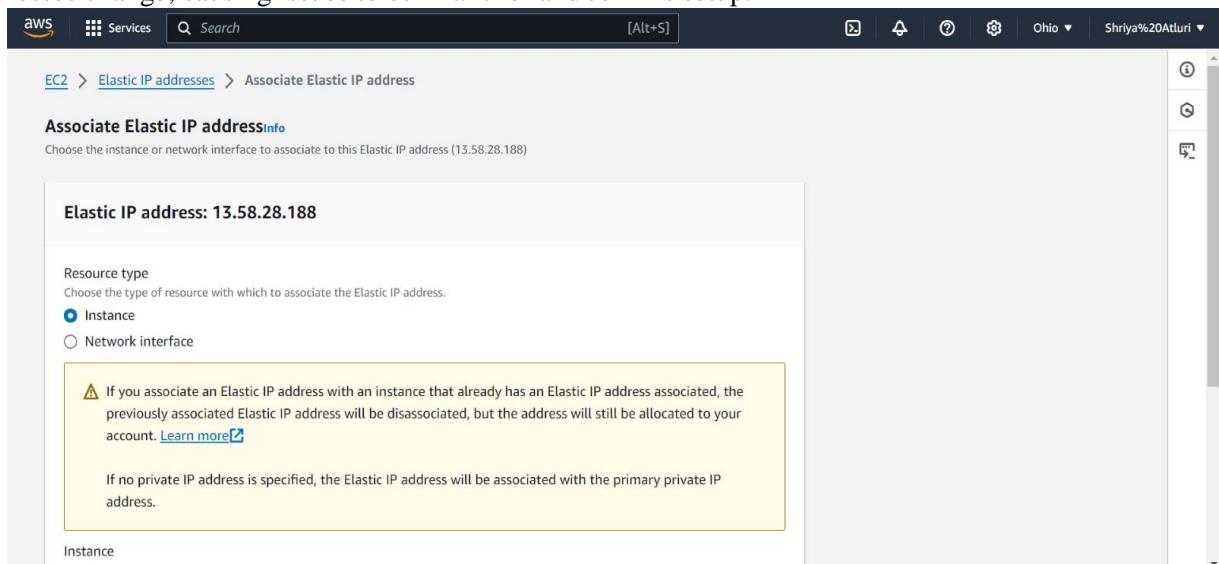
C:\Users\Sravya Atluri\Desktop\SWE645\Student Survey Form>docker run -p 8183:8080 satluri2/student-survey-app
NOTE: Picked up JDK_JAVA_OPTIONS: --add-opens=java.base/java.lang=ALL-UNNAMED --add-opens=java.base/java.io=ALL-UNNAMED
```

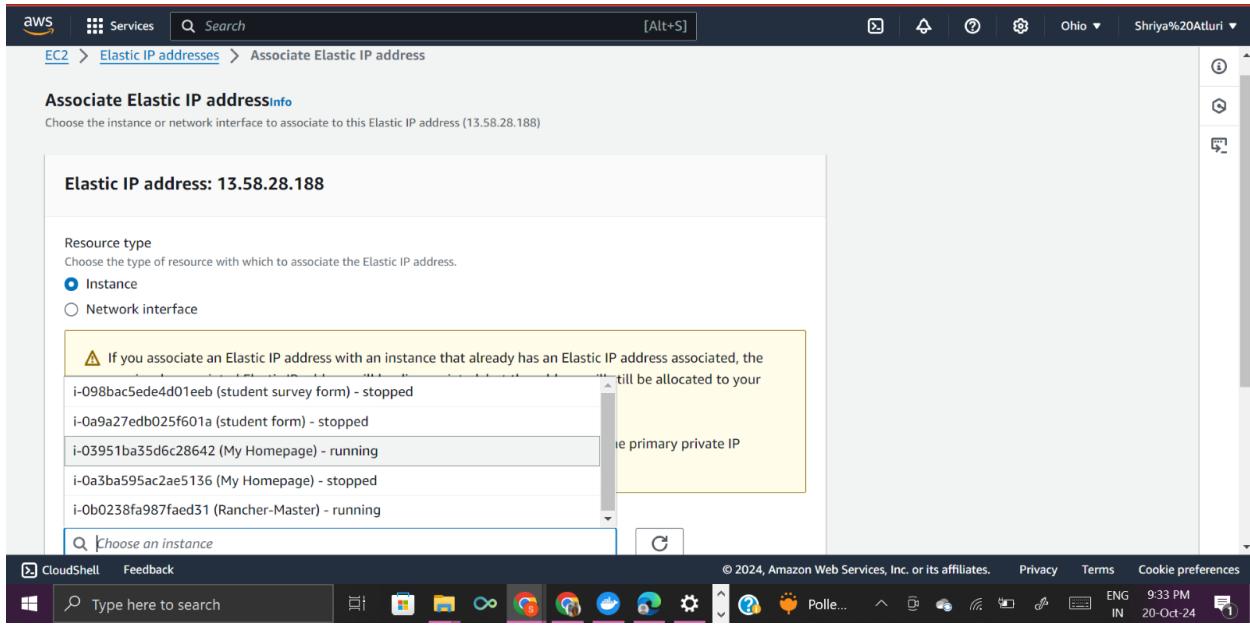
- Now the image should be pushed into the docker hub. You can go to the docker desktop and check it accordingly.



Setting up the AWS EC2 instances for deploying the application on a Kubernetes cluster

- For the rest of the assignment, we need two AWS EC2 instances.
- Login to the AWS Academy and then open AWS. Navigate to the EC2 service and open the dashboard.
- Here, click on “Launch Instance” to create the instance.
- Give the name of the instance as “Rancher-Master”, select AMI : “Ubuntu Server 22.04 LTS (HVM) SSD volume Type”, Instance type as : “t3.large”. Select a key-pair you already have, in this case, it is : “645”, Click on the checkboxes to allow traffic from HTTP and HTTPS from the internet, and finally increase storage from 8Gibs (default) to 30Gibs. And finally click on “Launch instance” button.
- Now, once the instance is created and running, associate it to an Elastic IP address. To do this, go to the “Elastic IPs” in the EC2 menu on the left pane of the screen. Create an Elastic IP using the default settings and associate it to this instance. We do this because whenever the AWS lab is restarted, it restarts the instances on EC2 and the IP addresses change, causing issues to our Rancher and Jenkins setup.





- Now, click on the instance and go to the Security tab. Here, in the “inbound rules”/”Outbound rules” section, click on the security group wizard.
- Scroll down and from either of the “inbound”/”outbound” tab click on edit rule option.
- Here, add a rule using the following configuration, Type : “Custom TCP”, Port range : “8080”, Source : “custom” and “0.0.0.0/0” and click on “Save rules”.

Inbound rules Info						
Security group rule ID	Type Info	Protocol	Port range Info	Source Info	Description - optional	Info
sgr-08db697053a133ff9	SSH ▼	TCP	22	Cus... ▼	<input type="text"/> 0.0.0.0/0 X	Delete
sgr-06aead81f8cac817d	Custom TCP ▼	TCP	8080	Cus... ▼	<input type="text"/> 0.0.0.0/0 X	Delete
sgr-09a267c9daa91c649	HTTPS ▼	TCP	443	Cus... ▼	<input type="text"/> 0.0.0.0/0 X	Delete
sgr-0ffb32172306e5426	HTTP ▼	TCP	80	Cus... ▼	<input type="text"/> 0.0.0.0/0 X	Delete

[Add rule](#)

- Now, click on the EC2 instance and click on “connect”. Go to “EC2 instance connect” tab, give the username as “ubuntu” and click on connect. A shell opens in a new tab.
- Here, we will give commands to install the docker on the instances.
- Commands to install docker : “ sudo apt-get update” followed by “sudo apt install docker.io”, Give permission when prompted: “Y”.

```

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-18-71:~$ sudo apt-get update
Hit:1 http://us-east-2.ec2.archive.ubuntu.com/ubuntu jammy InRelease [128 kB]
Get:2 http://us-east-2.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease [127 kB]
Get:3 http://us-east-2.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease [129 kB]
Get:4 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Get:5 http://us-east-2.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 Packages [14.1 MB]
Get:6 http://us-east-2.ec2.archive.ubuntu.com/ubuntu jammy/universe Translation-en [5652 kB]
Get:7 http://us-east-2.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 c-n-f Metadata [286 kB]
Get:8 http://us-east-2.ec2.archive.ubuntu.com/ubuntu jammy/multiverse amd64 Packages [217 kB]

i-0b777ba1e05be65a3 (Rancher-Master)
Public IPs: 18.223.52.41 Private IPs: 172.31.18.71

```



```

Reading package lists... Done
ubuntu@ip-172-31-18-71:~$ sudo apt install docker.io
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
bridge-utils containerd dns-root-data dnsmasq-base pigz runc ubuntu-fan
Suggested packages:
ifupdown aufs-tools cgroupfs-mount | cgroup-lite debootstrap docker-doc rinse zfs-fuse | zfsutils
The following NEW packages will be installed:
bridge-utils containerd dns-root-data dnsmasq-base docker.io pigz runc ubuntu-fan
0 upgraded, 8 newly installed, 0 to remove and 15 not upgraded.
Need to get 75.5 MB of archives.
After this operation, 284 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://us-east-2.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 pigz amd64 2.6-1 [63.6 kB]
Get:2 http://us-east-2.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 bridge-utils amd64 1.7-1ubuntu3 [34.4 kB]
Get:3 http://us-east-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 runc amd64 1.1.12-0ubuntu2-22.04.1 [8405 kB]
Get:4 http://us-east-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 containerd amd64 1.7.12-0ubuntu2-22.04.1 [37.8 MB]
Get:5 http://us-east-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 dns-root-data all 2023112702-ubuntu0.22.04.1 [5136 B]
Get:6 http://us-east-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 dnsmasq-base amd64 2.90-0ubuntu0.22.04.1 [374 kB]
Get:7 http://us-east-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 docker.io amd64 24.0.7-0ubuntu2-22.04.1 [28.8 MB]

i-0b777ba1e05be65a3 (Rancher-Master)
Public IPs: 18.223.52.41 Private IPs: 172.31.18.71

```



```

ubuntu@ip-172-31-18-71:~$ sudo docker run --privileged -d --restart=unless-stopped -p 80:80 -p 443:443 rancher/rancher
Unable to find image 'rancher/rancher:latest' locally
latest: Pulling from rancher/rancher
1797539a9e5e: Pull complete
21c4959e98e9: Pull complete
4f4fb7000ef54: Pull complete
4d58a02d3f92: Pull complete
659eb0d911b9: Pull complete
8c1d069e0658: Extracting [==>                               ] 18.94MB/364MB
e74d9cfce2e60: Download complete
4ad1f81f76c6: Download complete
b858e3a34133: Download complete
536dadf7f52c: Download complete
4b95ffcc70a8e: Download complete
ca1f59cc1b4d: Download complete
b3531871e7ee: Download complete
2a4d504935af: Download complete
8e376d229af8: Download complete
bcb0dfa8ac7c: Download complete
db8bb2b2cb1f2: Download complete
0ca9ae10bfb7: Download complete

i-0b777ba1e05be65a3 (Rancher-Master)
Public IPs: 18.223.52.41 Private IPs: 172.31.18.71

```

- After successfully running of the commands, docker should be successfully installed.
- Repeat all the steps mentioned above, to create the second instance with the name: “Kubernetes-Worker”. And proceed to install docker as completed for the first instance.

- This second instance will be our Worker node and the first instance will be the Master node.
- At the end make sure that both the instances are in “running” state.

The screenshot shows the AWS EC2 Instances page with three instances listed:

Name	Instance ID	Instance State	Type	Status Check	Alarm Status	Availability Zone
Rancher-Master	i-0b77ba1e05be65a3	Running	t3.large	3/3 checks passed	View alarms	us-east-2b
My Homepage	i-03951ba35d6c28642	Running	t2.micro	2/2 checks passed	View alarms	us-east-2c
Kubernetes-W...	i-0998c0b917f50e16c	Running	t3.large	3/3 checks passed	View alarms	us-east-2b

Setting up the Rancher

- Now that we have both the instances up and in “running” state, we will proceed to set up “rancher” on the master node i.e., “Rancher-Master” instance.
- Open the browser and go to the following website: <https://www.rancher.com/quick-start>. Here, scroll down to the “Start the Server” section under “Deploy Rancher”.
- Copy the command present there, which is : “ \$ sudo docker run --privileged -d --restart=unless-stopped -p 80:80 -p 443:443 rancher/rancher ”.

The screenshot shows the “Start the server” section of the Rancher quick-start guide. It includes a command to run the Docker container and instructions to access the Rancher UI.

```
$ sudo docker run --privileged -d --restart=unless-stopped -p 80:80 -p 443:443 rancher/rancher
```

To access the Rancher server UI, open a browser and go to the hostname or address where the container was installed. You will be guided through setting up your first cluster. To get started quickly, have a look at our additional resources and getting started guide.

[Get Started Guide](#)

- Now, connect to the “Rancher-Master” instance in the same way as described in the earlier section.

```
ubuntu@ip-172-31-29-126:~$ sudo docker run --privileged -d --restart=unless-stopped -p 80:80 -p 443:443 rancher/rancher
Unable to find image 'rancher/rancher:latest' locally
latest: Pulling from rancher/rancher
83a83d08b76a: Pulling fs layer
e3f06ffff7244: Pulling fs layer
4f4fb700ef54: Pulling fs layer
e50df727d370: Pull complete
c502336bf586: Pull complete
b99d4ccaf6b7: Extracting [=====] 329.2MB/334.1MB
bb60c5e7735b: Download complete
708002f0f717: Download complete
b3a6ad5d7cf4: Download complete
ee78742af818: Download complete
f9b52d555bc3: Download complete
f29d8c2c0a8a: Download complete
7fd775566dc6: Download complete
9fb2942ef2e1: Download complete
0d71b8d4e7bb: Download complete
935e3e307a66: Download complete
```

- Here, run the copied command. After it has run successfully, rancher will have been installed on this instance and can be accessed using the public IPv4 DNS address of the instance.

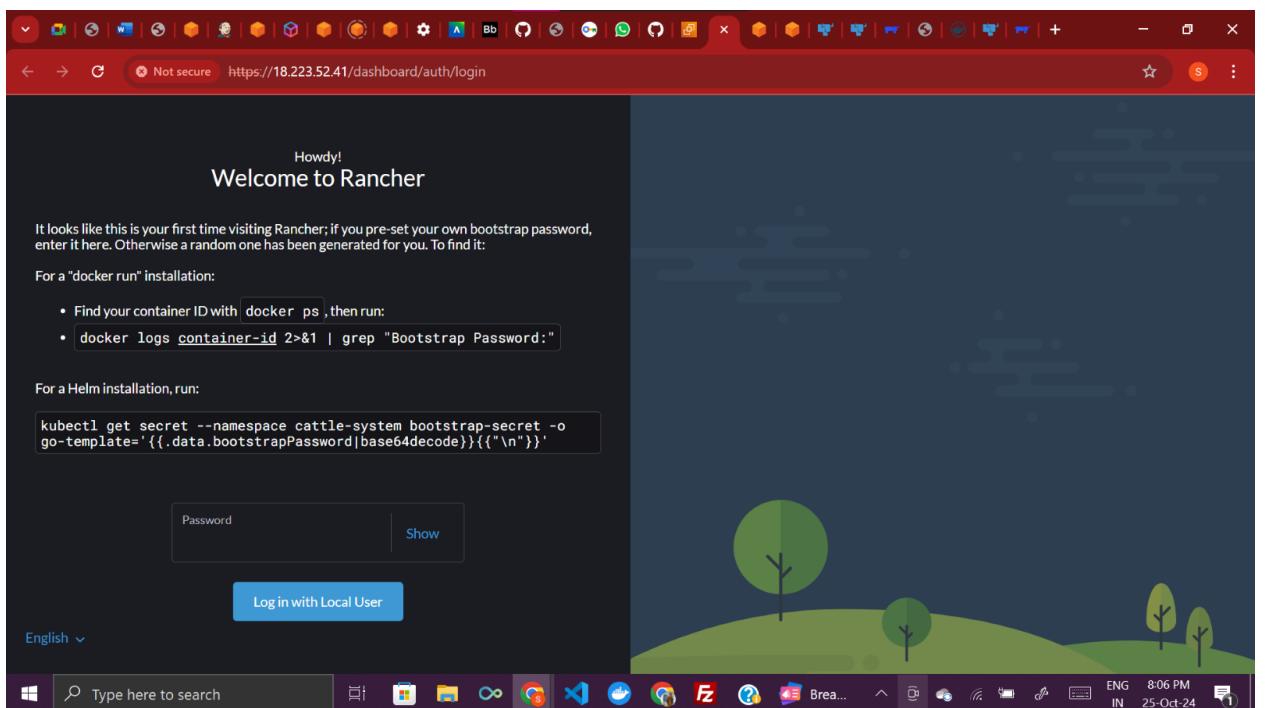
- Now, on the instance, run the following command to get the details of the container running on the instance: “ sudo docker ps ”. Store the result obtained somewhere (in notepad preferably), we will need this information later.

```
ubuntu@ip-172-31-18-71:~$ sudo docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS
 NAMES
d4398782d860        rancher/rancher   "entrypoint.sh"    29 seconds ago   Up 23 seconds   0.0.0.0:80->80/tcp, :::80->80/tcp, 0.0.0.0:443->443/tcp, :::443->443/tcp
3/tcp   amazing_feistel

ubuntu@ip-172-31-18-71:~$
```

i-0b777ba1e05be65a3 (Rancher-Master)
PublicIPs: 18.223.52.41 PrivateIPs: 172.31.18.71

Now, in the EC2 instance page, open the “Public IPv4 DNS” URL in the new tab. Give permission to proceed to the URL, even though it is unsafe. Upon loading, the rancher login page should be visible.



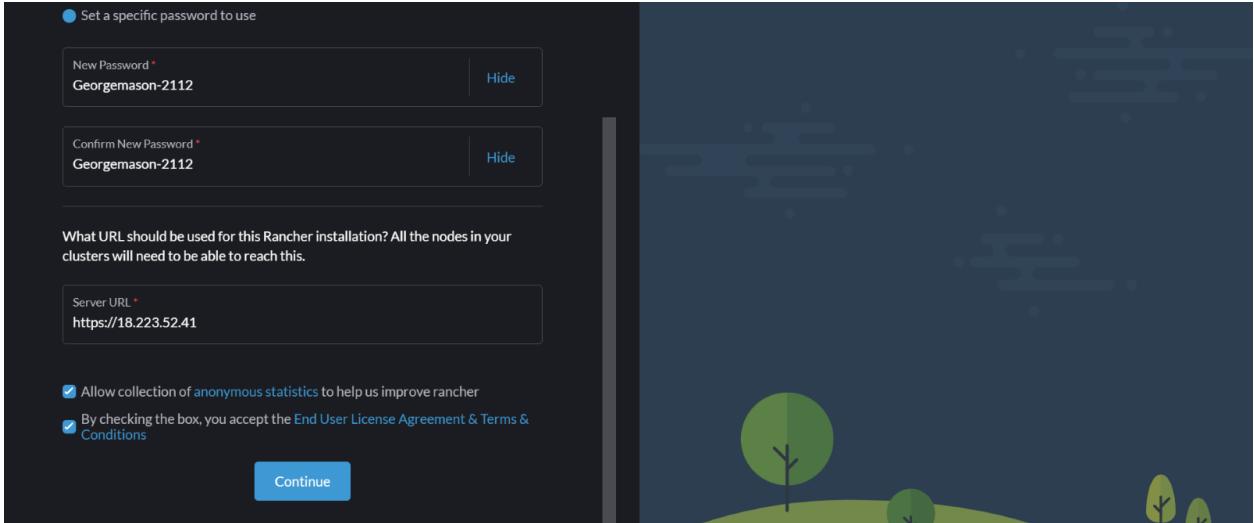
- Here, on this page it asks for a password. To get the password, copy the command displayed above on the same webpage and edit the container id. The container id is retrieved from the earlier stored output and run it on the instance console. The command should look like this: “sudo docker logs d4398782d860 2>&1 | grep “Bootstrap Password:”. The console will display the Password on the screen.

```
ubuntu@ip-172-31-18-71:~$ sudo docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS
 NAMES
d4398782d860        rancher/rancher   "entrypoint.sh"    29 seconds ago   Up 23 seconds   0.0.0.0:80->80/tcp, :::80->80/tcp, 0.0.0.0:443->443/tcp, :::443->443/tcp
3/tcp   amazing_feistel

ubuntu@ip-172-31-18-71:~$ sudo docker logs container-id 2>&1 | grep "Bootstrap Password:"
ubuntu@ip-172-31-18-71:~$ sudo docker logs container-id 2>&1 | grep "Bootstrap Password:"
ubuntu@ip-172-31-18-71:~$ sudo docker logs d4398782d860 2>&1 | grep "Bootstrap Password:"
2024/10/26 00:03:38 [INFO] Bootstrap Password: h4hrnchqgh5f297rdwkhc4ggvrrwkb44sm7rqnjlqdznc5d7tstg
ubuntu@ip-172-31-18-71:~$
```

i-0b777ba1e05be65a3 (Rancher-Master)
PublicIPs: 18.223.52.41 PrivateIPs: 172.31.18.71

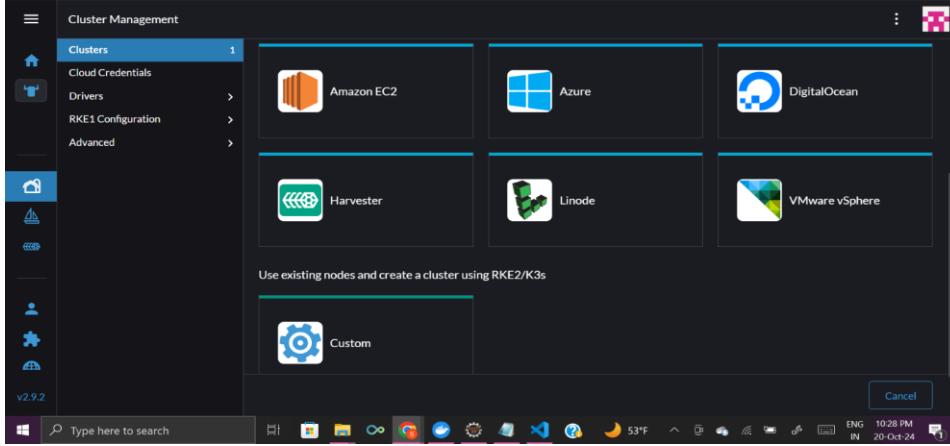
- Copy the password and paste it on the rancher login screen to login.
- On successful login, set up a new password using the prompts on the screen.



- Now the rancher dashboard will be displayed, where we can proceed with creating a cluster and deploying the application.

Creating a Kubernetes cluster and deploying the application.

- Now, as we have the rancher up and running, we will be creating a cluster and deploying our application on the cluster using the Rancher UI.
- On the dashboard, click on “Create Cluster” button. It should show various cluster options, select the “Custom” option and a create cluster form should be visible.



- Here, fill the form by giving the cluster name : “swe645assignment2” and leave everything else as default and click on the “Create” button.
- On the next page, go to “Registration” tab and under Step1, make sure that the 3 checkboxes of “etcd”, “control pane” and “worker” are checked.
- Now, copy the command present below in Step2.

- Open the AWS EC2 instance and connect to the “Kubernetes-Worker” node.
- Here, Run the copied command but, append the “—insecure” after the “curl” word in the command, and later run it.

```
Last login: Fri Oct 25 23:59:18 2024 from 3.16.146.5
ubuntu@ip-172-31-20-24:~$ curl --insecure -fL https://18.223.52.41/system-agent-install.sh | sudo sh -s - --server https://18.223.52.41 --label 'cattle.io/os=linux' --token vqgpbqk74szrbyvdjdnv79swfb7795rgcznrtsdhhp645zmr8 --ca-checksum dd0852303869969acf9301e750bf5c8575bcb718682245674adb25562
35f7b...etcd--controlplane--worker
curl: (52) Empty reply from server
Insecure: Select this to skip TLS verification if your server has a self-signed certificate.

Run this command in PowerShell on each of the existing Windows machines you want to register. Windows nodes can only be workers.

i-0998c0b917f50e16c (Kubernetes-Worker)
PublicIP: 18.188.60.195 PrivateIP: 172.31.20.24
```

- After the command is successfully executed, the cluster should be in the “Updating” state and after a few minutes it will change to “Active” state after which the cluster will be ready for deployment. This can be viewed under the “Clusters” option in “Cluster Management” option on the Rancher UI.

Clusters

State	Name	Version	Provider	Machines	Age
Active	local	v1.30.2+k3s2 Amd64	Local K3s	1	24 mins
Updating	swe645assignment2	v1.30.5+rke2r1 Amd64	Custom RKE2	0	4.2 mins

Configuring bootstrap node(s) custom-bd429a9b92f9: waiting for probes: calico

- Once the cluster is in the “Active” state, click on the “Explore” button which is next to the cluster.

Clusters

State	Name	Version	Provider	Machines	Age
Active	local	v1.30.2+k3s2 Amd64	Local K3s	1	26 mins
Active	swe645assignment2	v1.30.5+rke2r1 Amd64	Custom RKE2	1	6 mins

- On the left pane, under “Workloads”, select “Deployments”.
- Click on the “Create” button and a form should be visible on the screen.
- Fill in the form using the following details. Namespace : “default”, Name : “surveyassign2-deploy”, Replicas : “3” (no. of pods), Container image : “satluri2/student-survey-app: latest” (same as the name of the image on the docker hub).

- Under Networking, click on “Add port or service” button.
- Give the following details, Service type : “Node Port”, name : “nodeport”, private container port : “8080”, protocol : “TCP”.
- Leave everything else as default and click on create.

- Now the deployment is complete, and it should be in “Updating” state. Give it a few minutes and it should be in “Active” state.

The screenshot shows the Kubernetes dashboard with the URL <https://18.223.52.41/dashboard/c-m-wd7lj842/explorer/apps.deployment>. The left sidebar shows the navigation menu with 'Deployments' selected. The main panel displays the 'Deployments' list for the 'default' namespace. One deployment, 'surveyassign2-deploy', is shown in the 'Active' state, using the image 'satluri2/studen-t-survey-app', with 3/3 pods ready, 3 available, 0 restarts, and an age of 44 seconds.

- Once it is in the “Active” state, click on the deployment and navigate to the “Services” tab.
- Here, find the deployment created and click on the “nodeport” to open it in a new tab, under the target option.

The screenshot shows the Kubernetes dashboard with the URL <https://18.223.52.41/dashboard/c-m-wd7lj842/explorer/service>. The left sidebar shows the navigation menu with 'Service Discovery' selected. The main panel displays the 'Services' list for the 'default' namespace. Three services are listed: 'kubernetes' (Cluster IP, 6443/TCP), 'surveyassign2-deploy' (Cluster IP, nodeport 8080/TCP), and 'surveyassign2-deploy-nodeport' (Node Port, 32114/TCP). The 'surveyassign2-deploy-nodeport' service is highlighted.

- Now to the URL, append “/student” and open it. You should be able to see the application running.
- URL : <https://ec2-18-223-52-41.us-east-2.compute.amazonaws.com/k8s/clusters/c-m-wd7lj842/api/v1/namespaces/default/services/http:surveyassign2-deploy:8080/proxy/student>

- With this, we will have successfully deployed the application on a Kubernetes cluster.

The screenshot shows a web browser window with the title "Rancher - swe645assignment2" and the URL "https://ec2-18-223-52-41.us-east-2.compute.amazonaws.com/k8s/clusters/c-m-wd7lj842/api/v1/namespaces/default/services/http:surveyassign2-d...". The page displays a "Student Survey Form" with the heading "Help us improve your experience!". It contains five input fields: "First Name *", "Last Name *", "Street Address *", "City *", and "State *".

- For the next step we will need the “KubeConfig” file. To download it, go to our cluster page. On the top right, from the menu displayed, click on the “Download KubeConfig file” and save it in your local machine.

The screenshot shows the Rancher UI for the cluster "swe645assignment2". The left sidebar shows navigation options like Cluster, Workloads, CronJobs, DaemonSets, Deployments, Jobs, StatefulSets, Pods, Apps, Service Discovery, and Storage. The "Workloads" section is selected. A table titled "Workloads" lists one item: "surveyassign2-deploy" (Deployment) from the "saturn2/student-survey-app" namespace, which is active, has 0 restarts, and is 3.8 days old. There are buttons for Redeploy, Download YAML, and Delete.

We setup the Jenkins instance in order to install and run jenkins

- Now, we will focus on creating a CI/CD pipeline using Jenkins, to automatically build and reflect any changes in our source code.
- For this, we need to have Jenkins installed on our machine. So we will start off by creating an AWS EC2 instance in our AWS lab.
- We will create the instance in the same manner as we did to create our both the EC2 instances to deploy our application on the Kubernetes cluster.
- Give the name of the EC2 instance as “Jenkins”, keeping all the configurations same, give an elastic IP and edit the Security group.

The screenshot shows the AWS EC2 Instances page with four instances listed:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
My Homepage	i-03951ba35d6c28642	Running	t2.micro	2/2 checks passed	View alarms	us-east-2c
Kubernetes-W...	i-0998c0b917f50e16c	Running	t3.large	3/3 checks passed	View alarms	us-east-2b
Rancher-Master	i-0b777ba1e05be65a3	Running	t3.large	3/3 checks passed	View alarms	us-east-2b
Jenkins	i-0ca3c8d5d42853c92	Running	t3.large	3/3 checks passed	View alarms	us-east-2b

- Connect to the instance using “EC2 instance connect”. Make sure that the instance is in the “running” state.
- First, we need to install java before proceeding with Jenkins.
- Now, we run the following commands : “sudo apt-get update” followed by “sudo apt update”.
- Now run “ sudo apt install openjdk-11-jdk” and give permission when prompted as “Y”. This should install java jdk-11.

```
j@jenny-updates:~$ sudo apt upgrade
[...]
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
alsa-topology-conf alsu-ucm-conf at-spi2-core ca-certificates-java dconf-gsettings-backend dconf-service fontconfig-config fonts-dejavu-core
fonts-dejavu-extra gsettings-desktop-schemas java-common libasound2 libasound2-data libatk-bridge2.0-0 libatk-wrapper-java libatk-wrapper-java-jni
libatk1.0-0 libatk1.0-data libatspi2.0-0 libavahi-client3 libavahi-common libavahi-common3 libcurl3 libdconf1 libdrm-amdgpu libdrm-intel
libdrm-nouveau libdrm-radeon libfontconfig libfontenc libgif7 libgl1 libgl1-amber-dri libgl1-mesa-dri libglapi-mesa libglvnd0 libglx-mesa
libglx0 libgraphite2-3 libharfbuzz0b libice-dev libice6 libjpeg-turbo8 libjpeg8 liblcms2-2 liblomm15 libpciaccess0 libpcsc-lite1
libpthread-stubs0-dev libsensors-config libsm-dev libsm6 libx11-dev libx11-xcb1 libxau-dev libxaw7 libxcb-dr12-0 libxcb-dr13-0
libxcb-glx0 libxcb-present0 libxcb-randr0 libxcb-shm0 libxcb-sync1 libxcb-xfixes0 libxcb1-dev libxcomposite1 libxdamage-dev libxf86-fixes3
libxf72 libxi6 libxinerama1 libxkbfile1 libxmu6 libxpm4 libxrandr2 libxrender1 libxshmfence1 libxt-dev libxt6 libxtst6 libxv1 libxf86dg1
libxxf86vm1 openjdk-11-jdk-headless openjdk-11-jre openjdk-11-jre-headless session-migration x11-common x11-utils x11proto-dev xorg-sgml-doctools
xtrans-dev
Suggested packages:
default-jre libasound2-plugins alsu-utils cups-common libice-doc liblcms2-utils pcscd lm-sensors libsm-doc libx11-doc libxcb-doc libxt-doc
openjdk-11-demo openjdk-11-source visualvm libnss-mdns fonts-ipafont-gothic fonts-ipafont-mincho fonts-wqy-microhei fonts-wqy-zenhei fonts-indic
mesa-utils
The following NEW packages will be installed:
alsa-topology-conf alsu-ucm-conf at-spi2-core ca-certificates-java dconf-gsettings-backend dconf-service fontconfig-config fonts-dejavu-core
fonts-dejavu-extra gsettings-desktop-schemas java-common libasound2 libasound2-data libatk-bridge2.0-0 libatk-wrapper-java libatk-wrapper-java-jni
```

i-0b1728fd454489f (Jenkins)

Public IPs: 3.133.26.57 Private IPs: 172.31.29.57

- Now Jenkins can be installed by running the following commands: “sudo curl -fsSL <https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key> | sudo tee /usr/share/keyrings/jenkins-keyring.asc > /dev/null” to get the Jenkins package.
- Next run “ sudo echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] <https://pkg.jenkins.io/debian-stable> binary/ | sudo tee /etc/apt/sources.list.d/jenkins.list > /dev/null ” to unzip Jenkins.

```

ubuntu@ip-172-31-29-57:~$ https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | sudo tee /usr/share/keyrings/jenkins-keyring.asc > /dev/null
ubuntu@ip-172-31-29-57:~$ sudo curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | sudo tee /usr/share/keyrings/jenkins-keyring.asc > /dev/null
ubuntu@ip-172-31-29-57:~$ sudo apt update
Hit:1 http://us-east-2.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://us-east-2.ec2.archive.ubuntu.com/ubuntu jammy-security InRelease
Hit:3 http://us-east-2.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease
Ign:4 https://pkg.jenkins.io/debian-stable binary/ InRelease
Get:5 https://pkg.jenkins.io/debian-stable binary/ Release [2044 B]
Get:6 https://pkg.jenkins.io/debian-stable binary/ Release.gpg [833 B]
Hit:7 http://security.ubuntu.com/ubuntu jammy-security InRelease
Get:8 https://pkg.jenkins.io/debian-stable binary/ Packages [27.9 kB]

i-Ob201728fd454489f (Jenkins)
PublicIPs: 3.133.26.57 PrivateIPs: 172.31.29.57

```

- Now update apt and install Jenkins using the commands : “ sudo apt update” followed by “sudo apt install jenkins” . Give permission as “Y” when prompted and jenkins should be installed successfully.

```

ubuntu@ip-172-31-29-57:~$ sudo apt install jenkins
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  net-tools
The following NEW packages will be installed:
  jenkins net-tools
0 upgraded, 2 newly installed, 0 to remove and 15 not upgraded.
Need to get 91.5 MB of archives.
After this operation, 94.3 MB of additional disk space will be used.
Do you want to continue? [Y/n] y

```

- Now, start jenkins using the command : “sudo systemctl start jenkins.service”.
- Expose the port 8080 using the command : “sudo ufw allow 8080”.
- Now, run the following command to get admin password : “sudo cat /var/lib/jenkins/secrets/initialAdminPassword ” . It will display the password. Store It safely.

```

ubuntu@ip-172-31-29-57:~$ sudo systemctl start jenkins.service
ubuntu@ip-172-31-29-57:~$ sudo ufw allow 8080
Rules updated
Rules updated (v6)
ubuntu@ip-172-31-29-57:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
6471785a7bea4c63a9f74a9f729e2b37
ubuntu@ip-172-31-29-57:~$ 

i-Ob201728fd454489f (Jenkins)
PublicIPs: 3.133.26.57 PrivateIPs: 172.31.29.57

```

- Run the following commands to install snapd : “sudo apt install snapd”.
- Now using snap, install kubectl : “ sudo snap install kubectl --classic”.

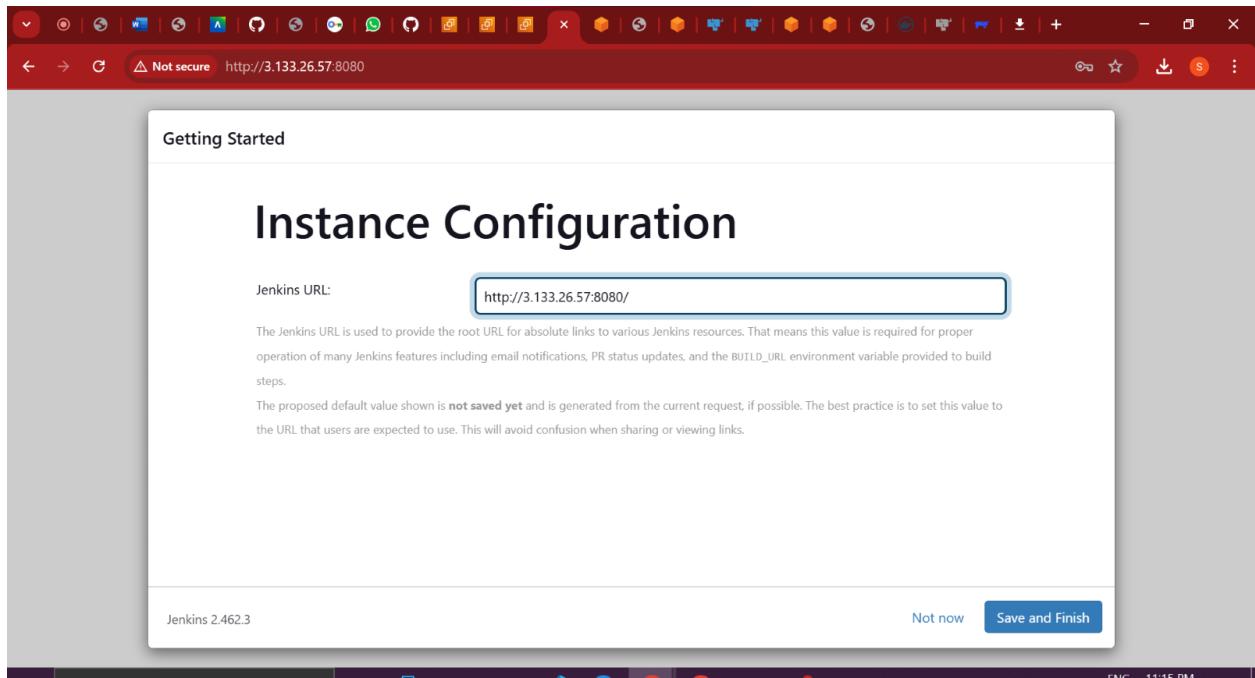
```

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-29-57:~$ sudo snap install kubectl --classic
kubectl 1.31.2 from Canonical/ installed
ubuntu@ip-172-31-29-57:~$ 

i-Ob201728fd454489f (Jenkins)
PublicIPs: 3.133.26.57 PrivateIPs: 172.31.29.57

```

- Go to the AWS EC2 instance “Jenkins” and open the “Public IPv4 address” in a new tab.
- To the URL, append the port 8080 and edit “https” to “http”. It should look like the following: “<http://3.133.26.57:8080>”.
- Enter the admin password obtained earlier to unlock jenkins.



- Now proceed to install the suggested plugins.

The screenshot shows the 'Customize Jenkins' section of the 'Getting Started' page. It features two main options: 'Install suggested plugins' (which installs community-recommended plugins) and 'Select plugins to install' (which allows users to choose specific plugins). A note below says 'Plugins extend Jenkins with additional features to support many different needs.'

- Next create an admin user. Click on "Save and create user" followed by "Save and Finish" and lastly "Start using jenkins".

The screenshot shows the 'Create First Admin User' dialog box. It requires three inputs: 'Username' (set to 'saturn2'), 'Password', and 'Confirm password'. There are also 'Skip and continue as admin' and 'Save and Continue' buttons at the bottom.

- You should be able to see the jenkins dashboard.

- Now, go back to the EC2 console. We need to create a config file.
- Run the following commands : “sudo su jenkins” to go to jenkins home.
- Next, go to root directory : “ cd ../../”.
- Go to the directory : “cd /var/lib/jenkins”
- Create a directory and enter this : “mkdir .kube” followed by “cd .kube”.
- Create the file and open it : “vi config”
- Now copy the contents from the “KubeConfig” file downloaded earlier and paste it here.
- Save the file using “:wq”.

The screenshot shows the Jenkins job configuration page for 'swe645assignment2'. Under the 'KubeConfig' section, the 'Content' tab is selected, displaying the YAML configuration for the kubeconfig file. The configuration includes contexts, users, and a current-context entry. At the bottom of the configuration, there is a note: 'PublicIPs: 3.133.26.57 PrivateIPs: 172.31.29.57'.

```

apiVersion: v1
clusters:
- cluster:
    name: "kubernetes"
    clusterInfo:
      certificateAuthority: "https://172.31.29.57:6443"
      server: "https://172.31.29.57:6443"
      token: "c2e03f3a-1a2d-4a2a-8a2d-0a2d0a2d0a2d"
      caCert: "-----BEGIN CERTIFICATE-----\nMIIEvTCCBQgGMA0GCSqGSIb3DQEBCwQDZJXWzjRzRpCnBqdxdLcUVY-EtOQ01tQxdZ\n-----END CERTIFICATE-----"
  context: "kubernetes"
  user: "kubernetes"
contexts:
- context:
    cluster: "kubernetes"
    user: "kubernetes"
current-context: "kubernetes"
kind: Config
users:
- name: "swe645assignment2"
  user:
    token: "kubeconfig-user-9xjh54zjxf:9bp96pdv2xnjnd2f94mfpv98wcpj9xn8v5qt9fw68x4lx4qdf5g9g"

```

- Now to verify, run the command : “kubectl config current-context”. It should display the cluster name as the output.

The screenshot shows a terminal window on an Ubuntu 18.04 LTS system. The user runs the command “kubectl config current-context”, which outputs “kubernetes”. Below the terminal window, the Jenkins job status is shown as “i-0b201728fd454489f (Jenkins) PublicIPs: 3.133.26.57 PrivateIPs: 172.31.29.57”.

```

ubuntu@ip-172-31-29-57:~$ sudo snap install kubectl --classic
kubectl 1.31.2 from Canonical✓ installed
ubuntu@ip-172-31-29-57:~$ sudo su jenkins
jenkins@ip-172-31-29-57:~/home/ubuntu$ cd ../../
jenkins@ip-172-31-29-57:~/var/lib/jenkins
jenkins@ip-172-31-29-57:~/var/lib/jenkins$ mkdir .kube
jenkins@ip-172-31-29-57:~/var/lib/jenkins/.kube$ vi config
jenkins@ip-172-31-29-57:~/var/lib/jenkins/.kube$ kubectl config current-context
swe645assignment2
jenkins@ip-172-31-29-57:~/var/lib/jenkins/.kube$ 
```

- Next, exit the jenkins mode using : “exit” command.
- Now proceed to install docker using the commands : “sudo apt-get update” and “sudo apt update” followed by “sudo apt install docker.io”. Give permission as “Y” when asked.

```
14 packages can be upgraded. Run 'apt list --upgradable' to see them.
ubuntu@ip-172-31-29-57:~$ sudo apt install docker.io
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
bridge-utils containerd dns-root-data dnsmasq-base pigz runc ubuntu-fan
Suggested packages:
ifupdown aufs-tools cgroupfs-mount | cgroup-lite debootstrap docker-doc rinse zfs-fuse | zfsutils
The following NEW packages will be installed:
bridge-utils containerd dns-root-data dnsmasq-base docker.io pigz runc ubuntu-fan
0 upgraded, 8 newly installed, 0 to remove and 14 not upgraded.
Need to get 75.5 MB of archives.
After this operation, 284 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://us-east-2.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 pigz amd64 2.6-1 [63.6 kB]
Get:2 http://us-east-2.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 bridge-utils amd64 1.7-lubuntu3 [34.4 kB]
Get:3 http://us-east-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 runc amd64 1.1.12-Ubuntu2-22.04.1 [8405 kB]
Get:4 http://us-east-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 containerd amd64 1.7.12-Ubuntu2-22.04.1 [37.8 MB]
Get:5 http://us-east-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 dns-root-data all 2023112702-ubuntu0.22.04.1 [5136 B]
Get:6 http://us-east-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 dnsmasq-base amd64 2.90-Ubuntu0.22.04.1 [374 kB]
Get:7 http://us-east-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 docker.io amd64 24.0.7-Ubuntu2-22.04.1 [28.8 MB]
```

i-0b201728fd454489f (Jenkins)

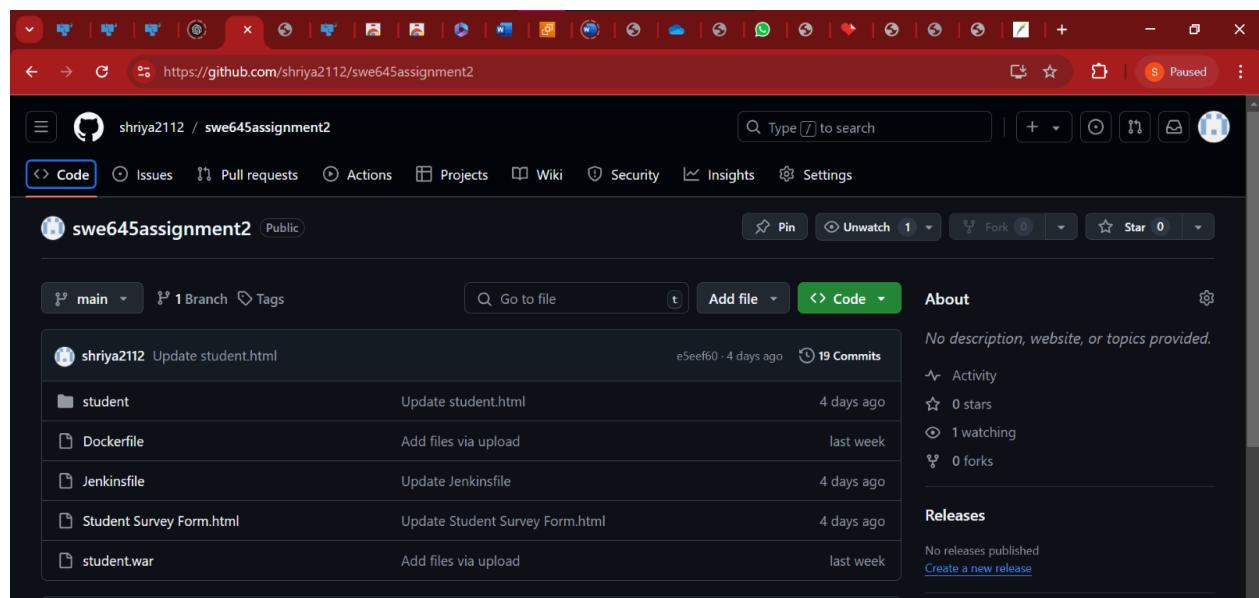
PublicIPs: 3.133.26.57 PrivateIPs: 172.31.29.57

- After installing, change the file permissions for socket use, using the command: “sudo chmod 777 /var/run/docker.sock”.

```
no ve guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-29-57:~$ sudo chmod 777 /var/run/docker.sock
ubuntu@ip-172-31-29-57:~$
```

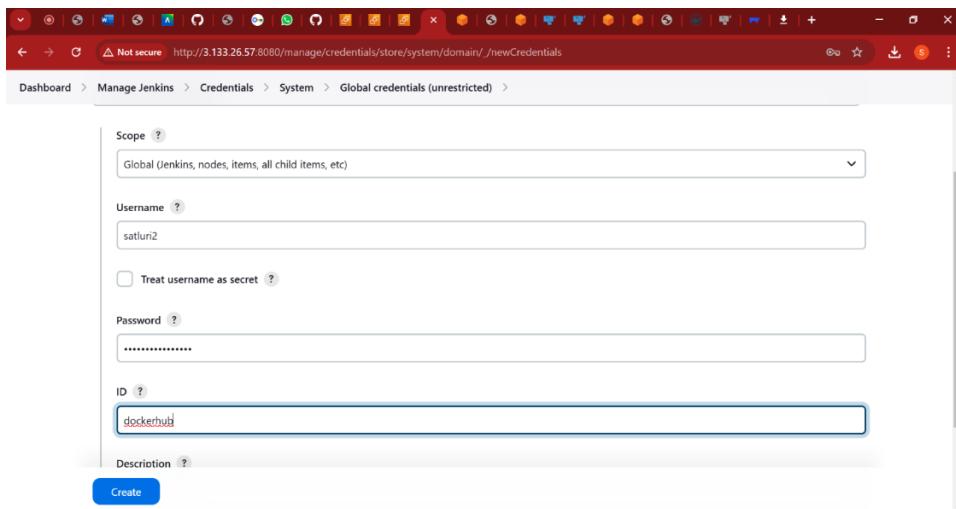
Setting up the GitHub repository for deploying the webpages

- In order to create our pipeline, we need to have the source code saved somewhere from where we can attach it to the pipeline, and it can look for any changes. For this purpose, we will be using GitHub.
- Go to <https://github.com/> and create an account and login to it.
- Now, create a new repository : “swe645assignment2”.
- To that repository : add your source files, Dockerfile and the war file, and then click on commit.
- We will be using this repository for our pipeline.
- Github repository URL : <https://github.com/shriya2112/swe645assignment2.git>

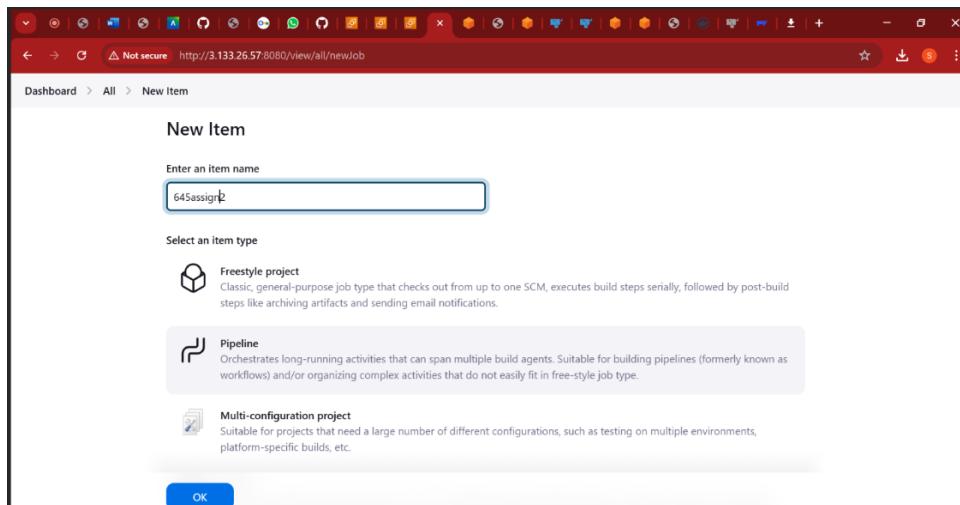


Creating a CI/CD pipeline and running it

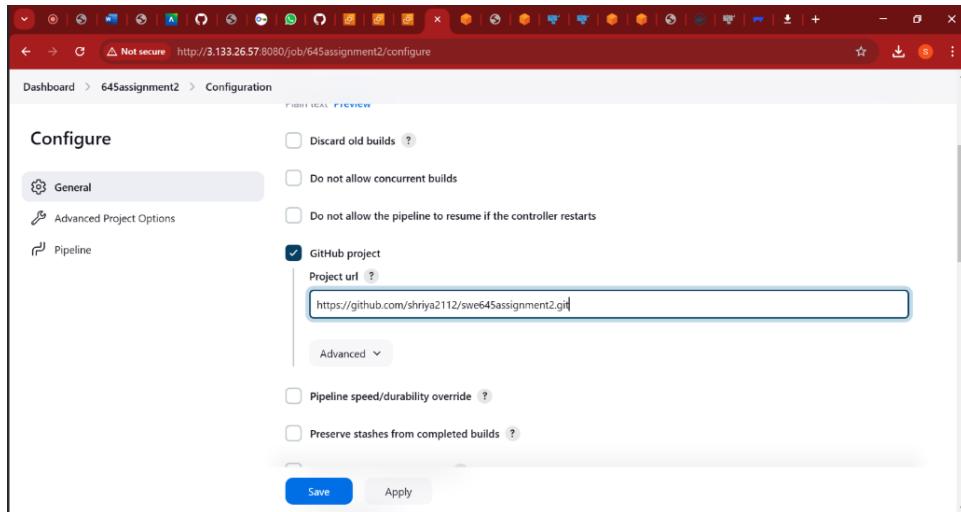
- Now, we have everything set up and ready to create our pipeline.
- Go to the jenkins dashboard, click on “Manage Jenkins”, scroll and select the “Credentials” option.
- Here, select the “global” option and then click on “Add credentials”.
- Select the Kind : “Username and password”, enter the dockerhub username and password and give id as “dockerhub” and click on create.



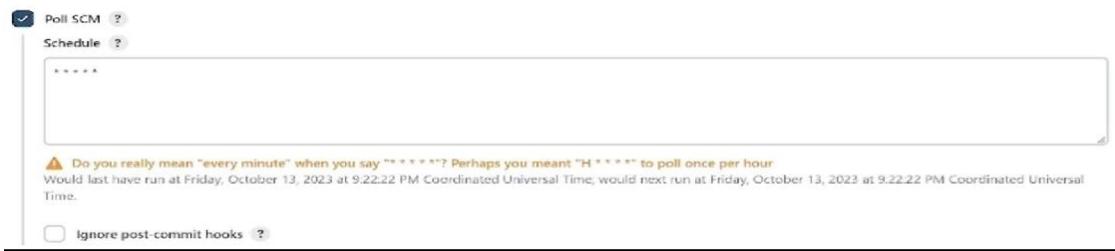
- Repeat the above steps to save GitHub credentials as well, give the id as “github”. For github, we will have to use PAT (Personal Access Tokens) as the password.
- Now go to the dashboard and click on “New Item”.
- Give the name as “645assign2” and click on “Pipeline” option.



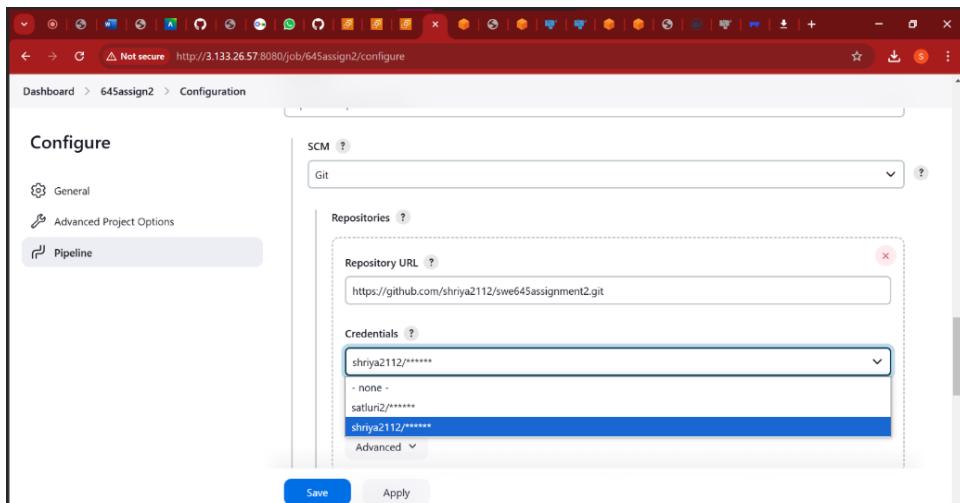
- Now, on the next page, check the “Github Project” checkbox, and in the project URL, paste the URL from Github repository : <https://github.com/shriya2112/swe645assignment2.git>.



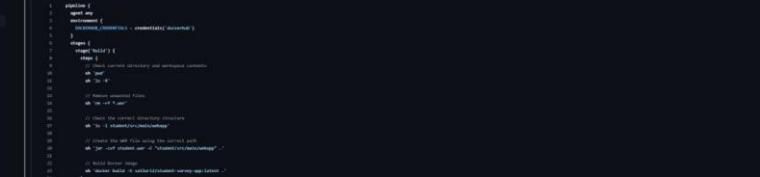
- Scroll down and check the “Poll SCM” checkbox.
- Give the Schedule as : “* * * * *”.



- Scroll down to the “Pipeline” section.
- Select the definition: “Pipeline from SCM” and SCM: “git”.
- Give the Github repository : “<https://github.com/shriya2112/swe645assignment2.git>”.
- Under Credentials, select the github credentials.
- Then, change the branch specifier : “*/main”.



- Go to the github repository and then create a new file : “Jenkinsfile” and then commit changes.
 - Click on Save.
 - Now a build will automatically start on this pipeline.
 - Now go to the Github repository => “Jenkinsfile” and click on the edit option.
 - Enter the commands given in the screenshot below and then, commit the changes. These are the commands that should be executed when any commit or changes occur to the files in the repository. They contain the commands on how to generate a new war file, create a docker image, login to the docker and push image including the deploy image on Kubernetes cluster.



The screenshot shows a Jenkins pipeline configuration. The left sidebar lists Jenkinsfile, main, and a GitHub repository named 'student'. The main panel displays the Jenkinsfile code, which includes a stage for updating the repository and a stage for running a survey application.

```
stage('Update repository') {
    git 'https://github.com/.../student.git'
}

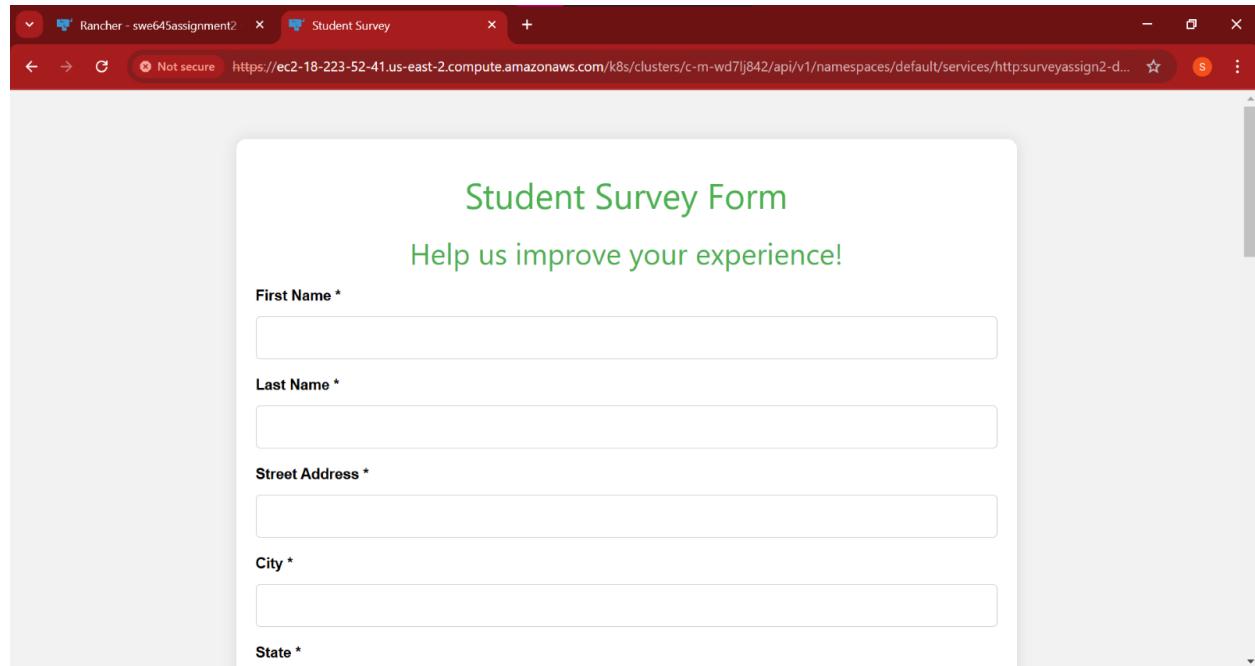
stage('Run Survey Application') {
    sh "cd $WORKSPACE/survey-app; ./gradlew build & ./gradlew run"
}
```

- Now when a build occurs, there might be some errors in the path, depending on the contents of the Jenkins file. Solve them by editing the Jenkinsfile accordingly.

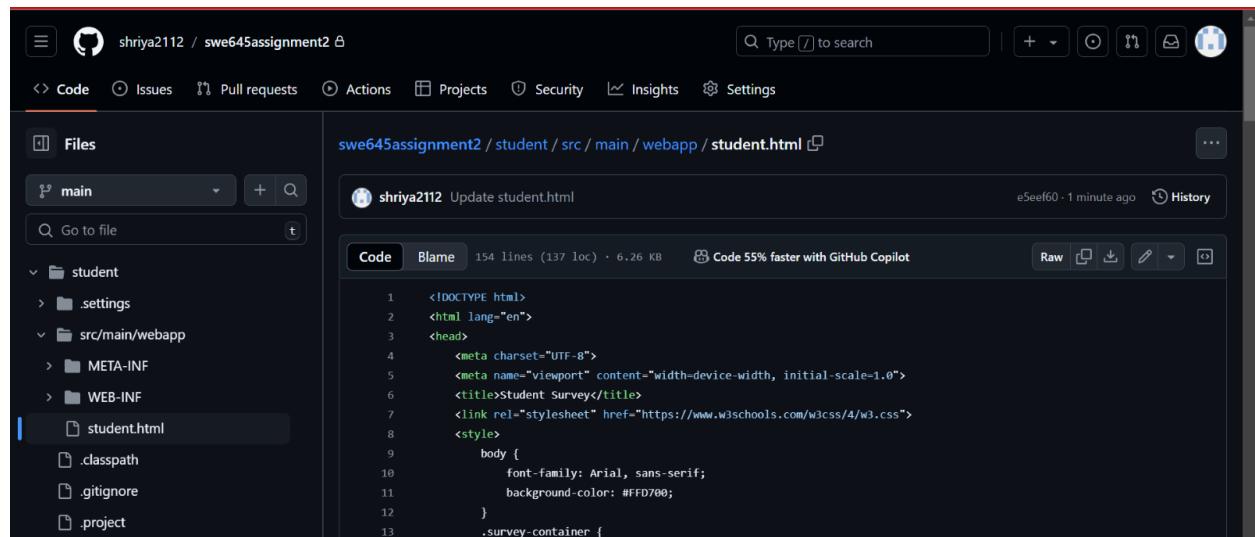
The screenshot shows the Jenkins multi-pipeline graph interface. At the top, there's a header bar with browser icons and a URL bar indicating 'Not secure' at <http://3.133.26.57:8080/job/645assign2/multi-pipeline-graph/>. The main title is 'Jenkins' with a user icon. A search bar says 'Search (CTRL+K)' with a help icon. To the right are notifications (bell), security (shield), and user (Shriya Aturi) icons, along with 'log out'. Below the header is a navigation bar with 'Dashboard > 645assign2 > Stages'. The main content area is titled 'Build 645assign2' with 'Build' and 'Configure' buttons. It displays two pipelines: Pipeline #2 and Pipeline #1. Pipeline #2 has stages: Start, Checkout SCM (green checkmark), Build (green checkmark), Login (green checkmark), Push image to ... (green checkmark), Deploying on K... (green checkmark), Post Actions (green checkmark), and End. Pipeline #1 has stages: Start and End.

- Now after a successful build, open the application deployed on Kubernetes cluster using the same old URL. Don't forget to append the "/student/" to the URL again.

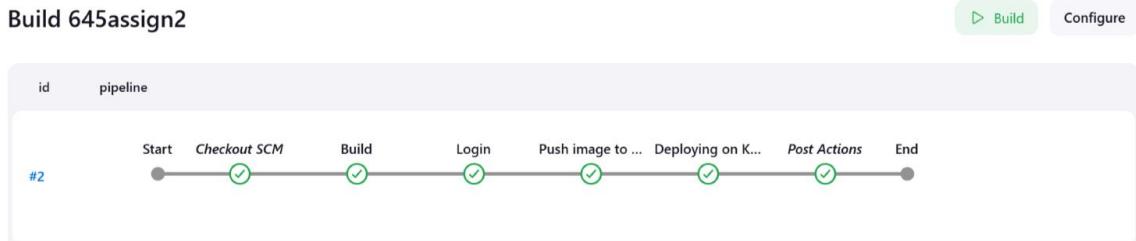
- And you should be able to see the application webpage.



- Now to test the pipeline, we will bring about some changes to the source file. Open your source file in github and perform some changes. In our case, we have decided to change the background color of the survey page on our website to gold color from white, following the GMU color schemes. So, we edit the CSS file and commit the changes. Gold color = #FFD700



- After the commit, the pipeline automatically creates a build. After the build is successful, if we reload the webpage, we will be able to see the changes being reflected and the background color changes to gold.



- URL : <https://ec2-18-223-52-41.us-east-2.compute.amazonaws.com/k8s/clusters/c-m-wd7lj842/api/v1/namespaces/default/services/http:surveyassign2-deploy:8080/proxy/student/>

Student Survey Form
Help us improve your experience!

First Name *

Last Name *

Street Address *

City *

State *

URL/Links of the application deployed

- GitHub URL : <https://github.com/shriya2112/swe645assignment2.git>
- Docker hub URL : <https://hub.docker.com/r/satluri2/student-survey-app>
- Application deployed on cluster URL : <https://ec2-18-223-52-41.us-east-2.compute.amazonaws.com/k8s/clusters/c-m-wd7lj842/api/v1/namespaces/default/services/http:surveyassign2-deploy:8080/proxy/student/>
- Docker URL : <http://localhost:8183/student/>
- Rancher start URL : <https://www.rancher.com/quick-start>

References

- <https://www.digitalocean.com/community/tutorials/how-to-install-jenkins-on-ubuntu-20-04>
- Lecture 4 and 5, 6 and 7 class notes.
- https://docs.docker.com/get-started/02_our_app/
- <https://docs.rke2.io/>
- How to Push Eclipse Project into GitHub: <https://youtu.be/gO20QGT6aW8?si=IWoe8x6ODSEJItjU>
- Build & Push Docker Image using Jenkins Pipeline :
<https://youtu.be/PKcGy9oPVXg?si=s6XUUQj7tQAdhE3H>

File Description

1. Front End files folder - Contains the HTML and CSS files of the assignment.
2. Student Survey Form – Contains the source code of the assignment.
3. Dockerfile – Docker file used to build the image.
4. Jenkinsfile – Jenkinsfile used to perform the builds after any commits/changes occur to the source code.
5. student.war – War file of the application.
6. student.zip – Zipped folder of all the source code of the application (from eclipse IDE).
7. swe645assignment2.yaml – Kubeconfig file of the cluster.
8. Swe 645 Assignment 2 – Execution video .mp4 – Video recording of the entire assignment development and execution.
9. SWE645_Assignment 2_README_SSP.docx – Word document , README file of the assignment.
10. SWE645_Assignment 2_README_SSP.pdf – PDF version of the README file.

Group Members and their Contributions

1. **Shriya Atluri (G01476041)**: Developed the source file, worked on creating and deploying application on Kubernetes cluster, worked on creating the docker image and the Github repository, Set up the CI/CD pipeline, worked on solving the issues and errors in setting up the jenkins and the CI/CD pipeline, contributed to the recording of the video for this assignment and the entire documentation part of the assignment.

AWS Homepage URL : <http://shriya-2112.s3-website.us-east-2.amazonaws.com>

2. **Sai Sampreeth Joshi (G01408085)** : Worked on creating and deploying application on Kubernetes cluster, worked on creating the docker image and the Github repository, Set up the CI/CD pipeline, worked on solving the issues and errors in setting up the jenkins and the CI/CD pipeline, contributed to the documentation part of the assignment.
3. **Pragathi Reddy Guduri (G01478980)** : Helped in setting up the jenkins environment and solving the issues and errors in setting up the jenkins and CI/CD pipeline, contributed to the recording of the video for this assignment.