

## SWE 645 : Assignment 2 README File

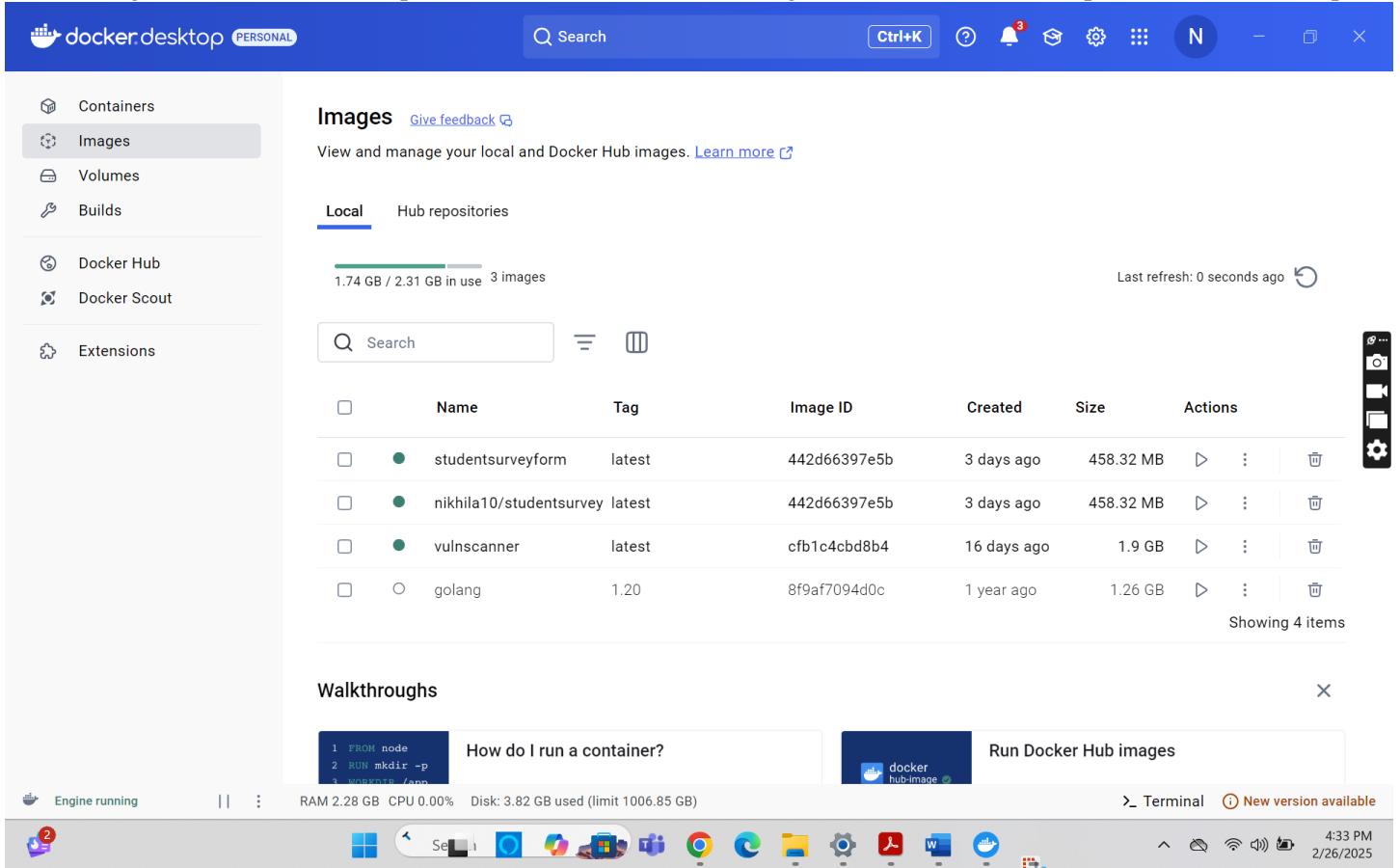
Welcome to the Readme File of SWE645 Assignment 2. We are required to follow a set of procedures in order to complete this assignment. This Readme file covers all the steps and procedures of the tasks in detail along with the screenshots. The Links/URLs used and achieved at various parts of the assignment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

The overview of the steps are mentioned below:

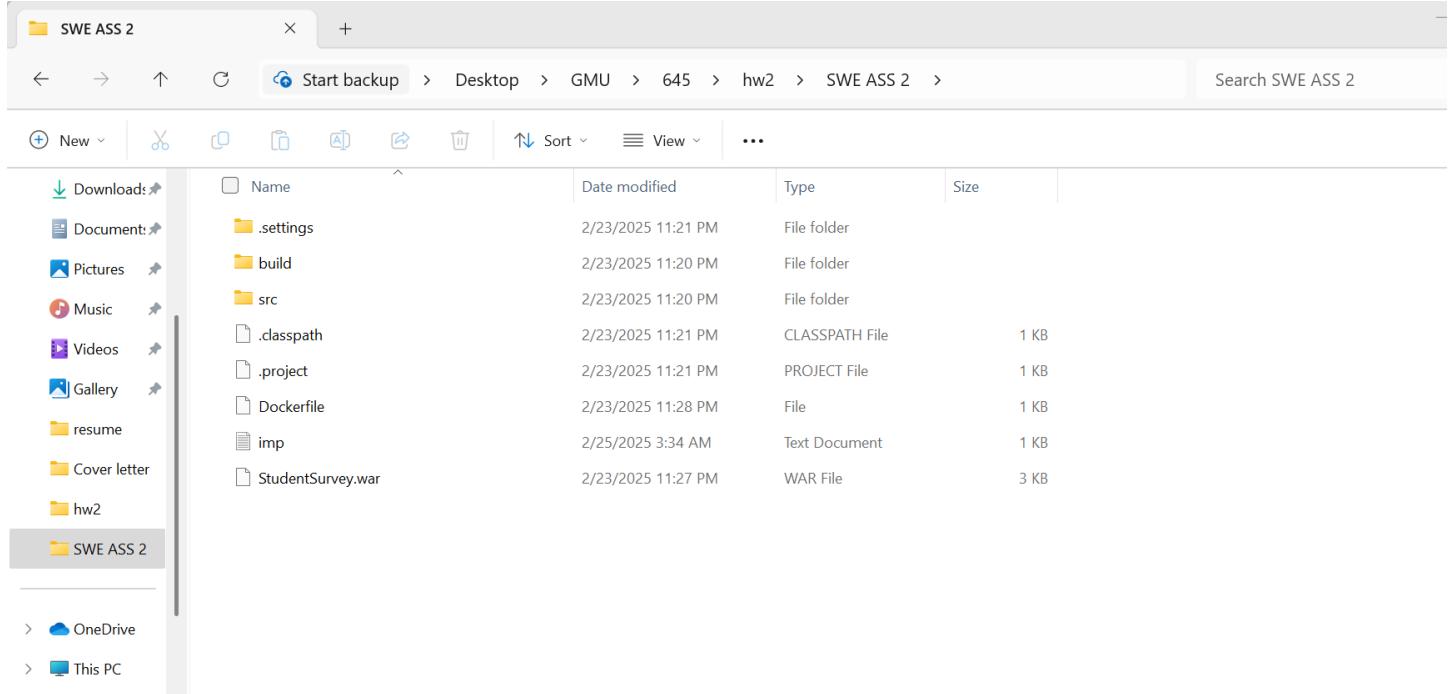
1. Using Dockerhub, we create an image of the application and push it into the dockerhub.
2. Setting up AWS EC2 instances for deploying the application on a Kubernetes cluster. We setup instances on AWS EC2 for deploying the application on a Kubernetes cluster.
3. A similar EC2 instance is created called 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 “StudentSurvey.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.

```

FROM tomcat:10.1-jre17
COPY StudentSurvey.war /usr/local/tomcat/webapps/
EXPOSE 8080

```

Ln 1, Col 1 | 85 characters | 100% | Windows (CRLF) | UTF-8

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

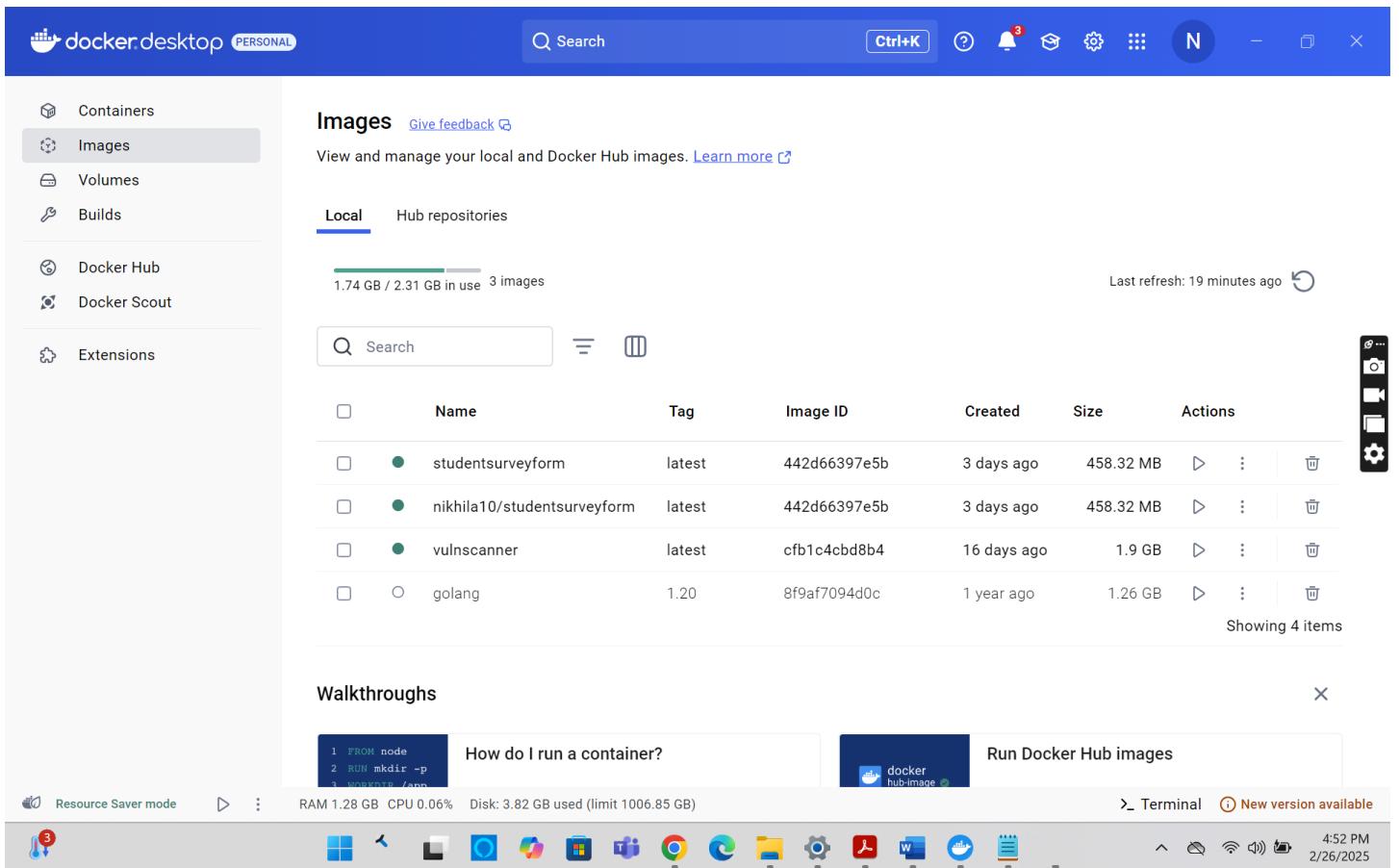
- Run the following commands to build a docker image : “ docker build -t studentsurveyform”
- Now an image of the application is created on the local system. To run this application on the localhost, the following command must be run on the command prompt : “ docker run -it -p 8183:8080 studentsurveyform”.
- 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/StudentSurvey/>

The screenshot shows the Docker Desktop interface. The left sidebar has options: Containers (selected), Images, Volumes, Builds, Docker Hub, Docker Scout, and Extensions. The main area is titled 'Containers' with a 'Search' bar and a 'Only show running containers' toggle. It displays two containers:

|                          | Name            | Container ID | Image             | Port(s)   | CPU (%) | Actions |
|--------------------------|-----------------|--------------|-------------------|-----------|---------|---------|
| <input type="checkbox"/> | goofy_edison    | 0e771f91b643 | studentsurveyform | 8183:8080 | N/A     |         |
| <input type="checkbox"/> | unruffled_kalam | 616225a10cbe | vulnscanner       | 8080:8080 | N/A     |         |

At the bottom, it says 'Showing 2 items'. Below this is a 'Walkthroughs' section with links to 'Multi-container applications' (8 mins) and 'Containerize your application' (\$ docker init) (3 mins). The taskbar at the bottom shows system status: Engine running, RAM 2.54 GB, CPU 0.00%, Disk: 3.82 GB used (limit 1006.85 GB), Terminal, New version available, and the date/time 4:38 PM 2/26/2025.

- 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 studentsurveyform nikhila10/studentsurveyform” followed by “docker push nikhila10/studentsurveyform”
  - 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 : “swe645”, 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.

Screenshot of the AWS CloudShell interface showing the process of associating an Elastic IP address (54.167.186.221) with an instance.

The top navigation bar includes tabs for Email, Cryptar, Launch, Workbooks, Session, localhc, Getting, assignr, swe645, Rancher, Student, and various system icons.

The main content area shows the "Associate Elastic IP address" page for EC2. It displays the following information:

- Elastic IP address:** 54.167.186.221
- Resource type:** Instance (selected)
- Private IP address:** Choose a private IP address (dropdown menu showing options: i-01bb0dd7e92fe6e52 (Kubernetes-worker) - running, i-080c990ddb558f46b (jenkins) - running, i-0de543a652bf6f2a0 (Rancher-Master) - running)
- Reassociation:** Choose an instance (dropdown menu showing options: i-01bb0dd7e92fe6e52 (Kubernetes-worker) - running, i-080c990ddb558f46b (jenkins) - running, i-0de543a652bf6f2a0 (Rancher-Master) - running)

The bottom navigation bar includes CloudShell, Feedback, Privacy, Terms, and Cookie preferences.

- 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”.

The screenshot shows the AWS Management Console interface for managing security groups. The URL in the address bar is `us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#ModifyInboundSecurityGroupRules:securityGroupId=sg-04b0...`. The page title is "Inbound rules".

| Security group rule ID | Type       | Protocol | Port range | Source                                     | Description - optional   |
|------------------------|------------|----------|------------|--|--|
|                        | Info       | Info     | Info       | Info                                       | Info   |
| sgr-083daaede2e4f77d2  | HTTP       | TCP      | 80         | C... <input type="button" value="Delete"/> | <input type="text"/> 0.0.0.0/0 <input type="button" value="Delete"/> |
| sgr-0731146904e866933  | HTTPS      | TCP      | 443        | C... <input type="button" value="Delete"/> | <input type="text"/> 0.0.0.0/0 <input type="button" value="Delete"/> |
| sgr-0382253bb257619a3  | Custom TCP | TCP      | 8080       | C... <input type="button" value="Delete"/> | <input type="text"/> 0.0.0.0/0 <input type="button" value="Delete"/> |
| sgr-01ee85995377951c3  | SSH        | TCP      | 22         | C... <input type="button" value="Delete"/> | <input type="text"/> 0.0.0.0/0 <input type="button" value="Delete"/> |

**Add rule**

At the bottom of the page, there are links for CloudShell, Feedback, and various system icons. The footer includes copyright information for 2025, privacy terms, cookie preferences, and a timestamp of 5:03 PM on 2/26/2025.

- 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”.

```

aws Services Search [Alt+S]
Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

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.

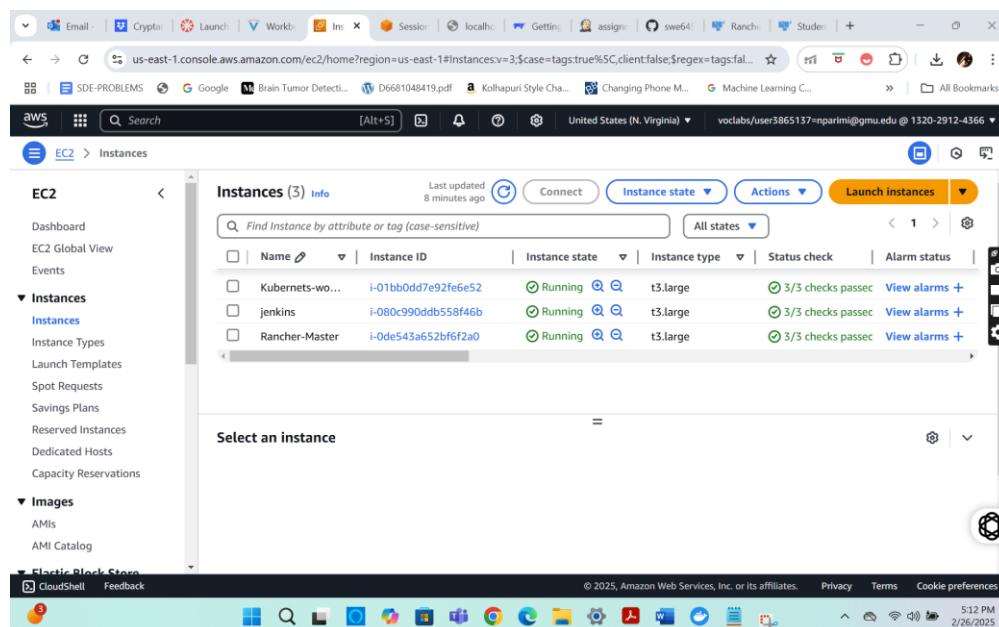
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-85-15:~$ sudo apt-get update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease [109 kB]
Get:4 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 Packages [14.1 MB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe Translation-en [5652 kB]

ubuntu@ip-172-31-95-147:~$ 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 134 not upgraded.
Need to get 69.7 MB of archives.
After this operation, 267 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y

```

- 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.



## Setting up the Rancher

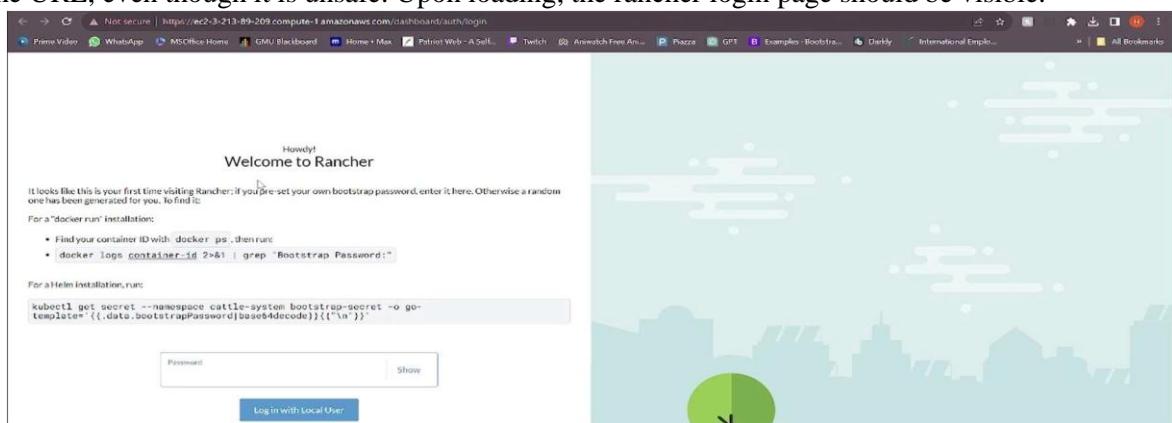
- Now that we have both the instances up and in “running” state, we will proceed to set up “rancher” on the master node i.e., “Rancher-Master” instance.
- Open the browser and go to the following website: <https://www.rancher.com/quick-start>. Here, scroll down to the “Start the Server” section under “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 ”.



- 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
e3f06fff7244: 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
b3a6ad6d7cf4: Download complete
ee78742af818: Download complete
f9b52d555bc3: Download complete
f29d8c2c0a8a: Download complete
7fdfe77566dc6: Download complete
9fb2942ef2e1: Download complete
8d71b8d4e7bb: 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.
- 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.



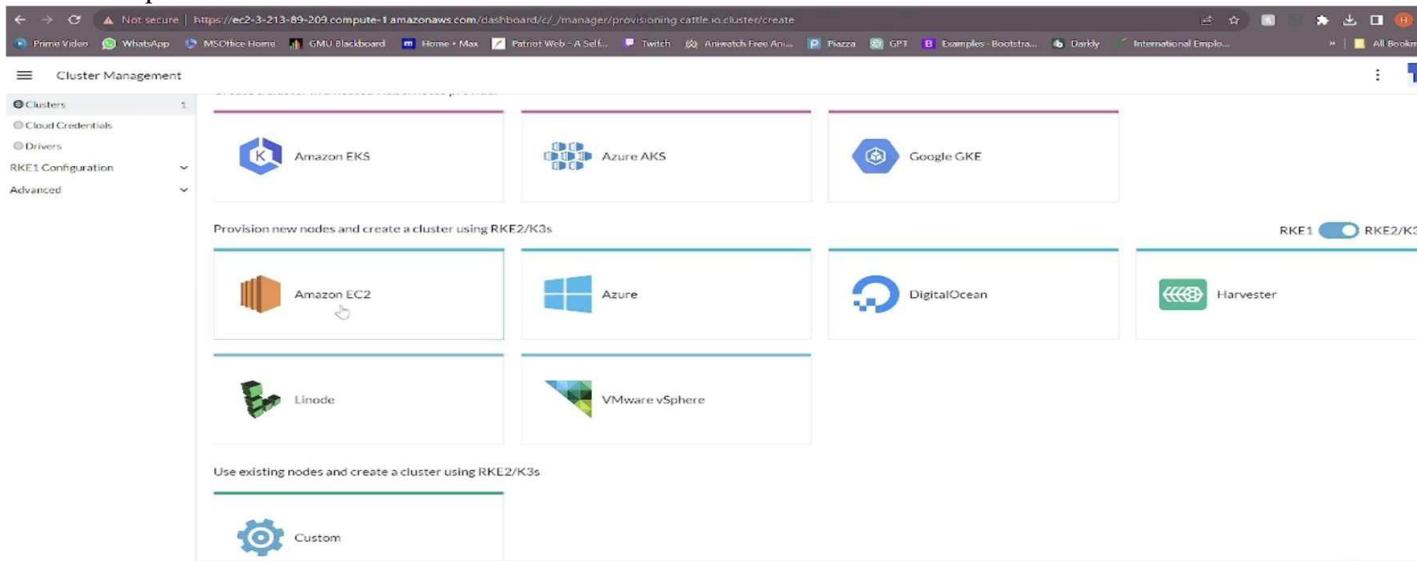
- Here, on this page it asks for a password. To get the password, copy the command displayed above on the same webpage and edit the container id. The container id is retrieved from the earlier stored output and run it on the instance console. The command should look like this: “sudo docker logs d4398782d860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

```
ubuntu@ip-172-31-95-147:~$ sudo docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS
49d9b5c86913        rancher/rancher   "entrypoint.sh"    19 seconds ago   Up 9 seconds      0.0.0.0:80->80/tcp, 0.0.0.0:443->443/tcp, 0.0.0.0:443->443/tcp
ubuntu@ip-172-31-95-147:~$ ^C
ubuntu@ip-172-31-95-147:~$ sudo docker logs 49d9b5c86913 2>&1 | grep "Bootstrap Password:"
2023/10/13 20:38:04 [INFO] Bootstrap Password: 7847tmn2q68k76g9cc9fvfw2wbrzh7jr780fp8nghz7qv48jc6lb7fI
```

- 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.

Cluster: Create Custom

Cluster Name: swe645assignment2

Cloud Provider: Default - RKE2 Embedded

Kubernetes Version: v1.26.9+rke2r1

Container Network: calico

CIS Profile: (None)

Pod Security Admission Configuration Template: Default - RKE2 Embedded

Create

- 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.

Cluster: swe645assignment2 Updating

Namespace: fleet-default Age: 1 secs

waiting for at least one control plane, etcd, and worker node to be registered

Provisioner: RKE2

Machines Provisioning Log Registration Snapshots Conditions Recent Events Related Resources

**Step 1**

**Node Role**  
Choose what roles the node will have in the cluster. The cluster needs to have at least one node with each role.

etcd  Control Plane  Worker

Show Advanced

**Step 2**

**Registration Command**  
Run this command on each of the existing Linux machines you want to register.

```
curl -fL https://ec2-3-213-89-209.compute-1.amazonaws.com/system-agent-install.sh | sudo sh -s --server https://ec2-3-213-89-209.compute-1.amazonaws.com --label 'cattle.io/os=linux' --token h4l49mfqqxsgfl2f8n91kd9prnd8csbb67t8kkpsqzdr6svx85tt --ca-checksum fcdb76f524abc17ade29a96db90c22011e99e33eb3839d7ef660130e4fdffbe --etcd --controlplane --worker
```

Insecure: Select this to skip TLS verification if your server has a self-signed certificate.

- 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 13 20:06:08 2023 from 18.206.107.29
ubuntu@ip-172-31-25-223:~$ curl --insecure -fL https://ec2-54-174-218-98.compute-1.amazonaws.com/system-agent-install.sh | sudo sh -s - --server https://ec2-54-174-218-98.compute-1.amazonaws.com
--label "cattle.io/os=linux" --token zort9wrps2l4whnlt6pnsgsb4cv5mvpv44cgh92gnmcvprcfk7gkgc --ca-checksum 71cbl5dbe4ellc63cc723b951a075af4c7ebdaf6a0ff68591dc0ac26ce2a83fd --etcd --controlplane --worker
  % Total    % Received   % Xferd  Average Speed   Time     Time      Current
                                 Dload  Upload   Total Spent  Left  Speed
100 30907    0 30907    0    0  1235k  0:00:00  0:00:00  0:00:00  1235k
[INFO] Label: cattle.io/os=linux
[INFO] Role requested: etcd
[INFO] Role requested: controlplane
[INFO] Role requested: worker
[INFO] Using default agent configuration directory /etc/rancher/agent
[INFO] Using default agent var directory /var/lib/rancher/agent
[INFO] Determined CA is necessary to connect to Rancher
[INFO] Successfully downloaded CA certificate
[INFO] Value from https://ec2-54-174-218-98.compute-1.amazonaws.com/cacerts is an x509 certificate
[INFO] Successfully tested Rancher connection
[INFO] Downloading rancher-system-agent binary from https://ec2-54-174-218-98.compute-1.amazonaws.com/assets/rancher-system-agent-amd64
[INFO] Successfully downloaded the rancher-system-agent binary.
[INFO] Downloading rancher-system-agent-uninstall.sh script from https://ec2-54-174-218-98.compute-1.amazonaws.com/assets/system-agent-uninstall.sh
[INFO] Successfully downloaded the rancher-system-agent-uninstall.sh script.
[INFO] Generating Cattle ID
```

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

Clusters

[Import Existing](#) [Create](#)

| Download KubeConfig | Take Snapshot     | Download YAML  | Delete      | Filter   |        |                         |
|---------------------|-------------------|----------------|-------------|----------|--------|-------------------------|
| State               | Name              | Version        | Provider    | Machines | Age    |                         |
| Active              | local             | v1.26.4+k3s1   | K3s         | 1        | 7 mins | <a href="#">Explore</a> |
| Active              | swe645assignment2 | v1.26.8+rke2r1 | Custom RKE2 | 1        | 5 mins | <a href="#">Explore</a> |

- 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 : “assignment2-deployment”, Replicas : “3” (no. of pods), Container image : “nikhila10/studentsurveyform: latest” (same as the name of the image on the docker hub).

Email | Crypta | Launch | Workbe | Instances | Session | localho | Getting | swe645assignment2 | assign... | swe645 | Studen | +

Not secure https://ec2-54-173-40-206.compute-1.amazonaws.com/dashboard/c/c-m-jds2lbj9/explorer/apps.deployment/creat...

SDE-PROBLEMS Google Brain Tumor Detect... D6681048419.pdf Kolhapuri Style Cha... Changing Phone M... Machine Learning C...

All Bookmarks

swe645assignment2 Only User Namespaces

Cluster Workloads CronJobs DaemonSets Deployments Jobs StatefulSets Pods Apps Service Discovery Storage Policy More Resources S62

Deployment: Create

Namespace \* default Name \* assignment2-deployment Description Any text you want that better describes this resource

Replicas \* 3

Deployment Pod container-0 + Add Container

General Health Check Resources Security Context Storage

Container Name container-0 Init Container Standard Container

Container Image Pull Policy

Create

Cancel Edit as YAML

v2.10.2

5:30 PM 2/26/2025

The screenshot shows the AWS Lambda console interface. On the left, there's a sidebar with various icons and a 'S62' badge. The main area is titled 'swe645assignment2'. A dropdown menu says 'Only User Namespaces'. The 'Deployments' tab is selected. It shows a table with one item: 'Deployment' (Container Name: 'container-0'). Below the table are tabs for 'General', 'Health Check', 'Resources', 'Security Context', and 'Storage'. Under 'General', there's a 'Container Name' field set to 'container-0'. Under 'Image', the 'Container Image' is 'nikhila10/studentsurveyform:latest' and the 'Pull Policy' is 'Always'. Under 'Networking', there's a 'Port' section with a 'Add port or service' button. At the bottom right are 'Cancel', 'Edit as YAML', and 'Create' buttons. The status bar at the bottom shows a battery icon with '3', a date/time of '2/26/2025 5:29 PM', and other system icons.

- 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.

The screenshot shows the Kubernetes Dashboard interface. On the left, a sidebar menu lists various resources: Cluster, Workloads (CronJobs, DaemonSets, Deployments), Jobs, StatefulSets, Pods, Apps, Service Discovery, Storage, Policy, and More Resources. The 'Deployments' item is currently selected and highlighted in blue. The main panel is titled 'swe645assignment2'. It displays a 'Networking' section with a sub-section for 'Service Type: Node Port'. A form is being filled out with the following details:

- Name: nodeport
- Private Container Port: 8080
- Protocol: TCP
- Listening Port: 8080

Below the networking section is a 'Command' section with fields for 'Command' (e.g., /bin/sh) and 'Arguments' (e.g., /usr/sbin/httpd -f httpd.conf). There are also fields for 'WorkingDir' (e.g., /myapp) and 'Stdin' (No). At the bottom right of the main panel are three buttons: 'Cancel', 'Edit as YAML', and '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 interface. On the left, a sidebar menu lists various resources: Cluster, Workloads (selected), CronJobs, DaemonSets, Deployments (selected), Jobs, StatefulSets, and Pods. The Deployments section shows 1 item. The main content area is titled "Deployments" and displays a table of deployment details. The table has columns for State, Name, Image, Ready, Up To Date, Available, Restarts, Age, and Health. One deployment is listed: "assignment2-deployment" (nikhila10/studentsurveyform) with 3/3 pods ready, 3 available, 0 restarts, and an age of 2 days. The status is "Active". The dashboard also includes a toolbar with "Redeploy", "Download YAML", and "Delete" buttons, and a "Create" button. The top navigation bar shows the URL as <https://ec2-54-173-40-206.compute-1.amazonaws.com/dashboard/c/c-m-jds2lbj9/explorer/apps.deployment>. The bottom taskbar shows the Windows Start button, search bar, and various pinned icons.

- 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 Cattle UI interface for managing a Kubernetes cluster named 'swe645assignment2'. The left sidebar navigation includes 'Cluster', 'Workloads', 'Apps', 'Service Discovery', 'HorizontalPodAutoscalers', 'Ingresses', 'Services' (selected), 'Storage', 'Policy', and 'More Resources'. The main content area is titled 'Services' and displays a table of services in the 'default' namespace. The table columns are 'State', 'Name', 'Target', 'Selector', 'Type', and 'Age'. The services listed are:

|                          | Name                            | Target            | Selector  | Type       | Age    |
|--------------------------|---------------------------------|-------------------|---|------------|--------|
| <input type="checkbox"/> | assignment2-deployment          | nodeport 8080/TCP | workload.user.cattle.io/workloadselector=apps.deployment-default-assignment2-deployment | Cluster IP | 2 days |
| <input type="checkbox"/> | assignment2-deployment-nodeport | 30559/TCP         | workload.user.cattle.io/workloadselector=apps.deployment-default-assignment2-deployment | Node Port  | 2 days |
| <input type="checkbox"/> | kubernetes                      | https 6443/TCP    |   | Cluster IP | 2 days |

- Now to the URL, append “/StudentSurvey” and open it. You should be able to see the application running.
- URL : <https://ec2-54-173-40-206.compute-1.amazonaws.com/k8s/clusters/c-m-ids2lbj9/api/v1/namespaces/default/services/http:assignment2-deployment:8080/proxy/StudentSurvey/>
- With this, we will have successfully deployed the application on a Kubernetes cluster.

Student Survey Form

First Name: \*

Last Name: \*

Email: \*

Phone: \*

Street Address: \*

City: \*

State: \*

ZIP Code: \*

- 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 Kubernetes dashboard interface. On the left, there's a sidebar with various navigation items. The 'Workloads' item is currently selected. The main content area is titled 'Workloads' and shows a single deployment named 'assignment2-deployment'. The deployment details are as follows:

| State  | Name                   | Type       | Image                      | Restarts | Age    |
|--------|------------------------|------------|----------------------------|----------|--------|
| Active | assignment2-deployment | Deployment | nikhil10/studentsurveyform | 0        | 2 days |

At the bottom of the dashboard, there's a Windows taskbar with several pinned icons and the date/time (5:34 PM, 2/26/2025).

### 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. On the left, there's a navigation sidebar with options like Dashboard, EC2 Global View, Events, Instances (selected), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Images (AMIs, AMI Catalog), and Elastic Block Store (CloudShell, Feedback). The main area displays a table of instances:

| Name            | Instance ID         | Instance state | Instance type | Status check      | Alarm status                  |
|-----------------|---------------------|----------------|---------------|-------------------|-------------------------------|
| Kubernets-wo... | i-01bb0dd7e92fe6e52 | Running        | t3.large      | 3/3 checks passed | <a href="#">View alarms +</a> |
| jenkins         | i-080c990ddb558f46b | Running        | t3.large      | 3/3 checks passed | <a href="#">View alarms +</a> |
| Rancher-Master  | i-0de543a652bf6f2a0 | Running        | t3.large      | 3/3 checks passed | <a href="#">View alarms +</a> |

Below the table, a message says "Select an instance". At the bottom of the page, there's a footer with links to CloudShell, Feedback, and various AWS services, along with copyright information and a timestamp (5:35 PM 2/26/2025).

- 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-17-jdk” and give permission when prompted as “Y”. This should install java jdk-17.

```
root@jammy-updates:~# sudo apt update
root@jammy-updates:~# sudo apt update
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
alsa-topology-conf alsa-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 libcups2 libdconf1 libdrm-amdgpu libdrm-intel
libdrm-nouveau libdrm-radeon libfontconfig libfontenc libgbif7 libglib libglib-2.0-0 libglib-2.0-0 libglib-2.0-0 libglib-2.0-0 libglib-mesa
libglx0 libgraphite2-3 libharfbuzz0 libice-dev libice6 libjpeg-turbo8 libjpeg8 liblcms2-2 liblvm15 libpciaccess0 libpcsc-lite
libpthread-stubs0-dev libsensors-config libsensors5 libsm-dev libsm6 libx11-dev libx11-xcb libxau-dev libxaw7 libxcb-drivers-0
libxcb-glx0 libxcb-present0 libxcb-shape0 libxcb-shm0 libxcb-sync1 libxcb-xfixes0 libxcb1-dev libxcompositel libxdmcp-dev libxf86-video
libxft2 libx16 libxinerama libxkbfile1 libxmu6 libxpm4 libxrandr2 librender1 libxshmfence1 libxt-dev libxt6 libxtst6 libxv1 libxf86dg1
libxf86vml 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 alsaview 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 alsa-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-0b201728fd454489f (Jenkins)

PublicIPs: 3.133.26.57 PrivateIPs: 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-172-31-86-97:~$ 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-172-31-86-97:~$ 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
```

- 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-172-31-86-97:~$ sudo apt install jenkins
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
```

- 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-172-31-86-97:~$ sudo systemctl start jenkins.service
ubuntu@ip-172-31-86-97:~$ sudo ufw allow 8080
Rules updated
Rules updated (v6)
ubuntu@ip-172-31-86-97:~$
sudo cat /var/lib/jenkins/secrets/initialAdminPassword
7ce996ea15de4093b9611537eb28a3ea
```

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

```
ubuntu@ip-172-31-86-97:~$ sudo snapd install kubectl --classic
sudo: snapd: command not found
ubuntu@ip-172-31-86-97:~$ sudo snap install kubectl --classic
kubectl 1.28.2 from Canonical✓ installed
```

- 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://54.167.186.221:8080>”.
- Enter the admin password obtained earlier to unlock Jenkins.
- Now proceed to install the suggested plugins.

## Customize Jenkins

Plugins extend Jenkins with additional features to support many different needs.

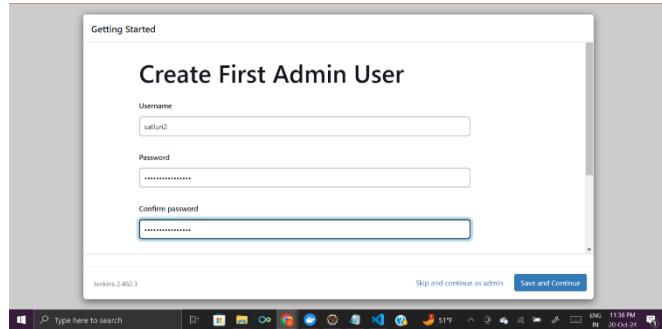
### Install suggested plugins

Install plugins the Jenkins community finds most useful. 

### Select plugins to install

Select and install plugins most suitable for your needs.

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



- 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”.

```

apiVersion: v1
kind: Config
clusters:
- name: "swe645assignment2"
  cluster:
    server: "https://ec2-3-213-89-209.compute-1.amazonaws.com/k8s/clusters/c-m-mhfg88dd"
    certificate-authority-data: "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUJ2VENDQ\\
      VdPZ0F3SUJBZ01CQURBS0JnZ3Foa2pPUFFRREFqQkdNUnd3R2dZRFZRUUtFeE5rZVc1aGJXbGoKY\\
      kdsemRHvnValveI0YjNkfk1TwXdkQV1EV1FRERCMnt1VzVoYldsamJHblHpkR1Z1WlhJdFKyRkFN\\
      Fk1TnpJeQpPVFE0T0RBZUZ3MH1NeKv3TVRNNeU1ETTRNRGhhRncwek16RXdNVEF5TURNNE1EaGFNR\\
      V14SERBYUJnT1ZCQW9UckUyUjVibUZ0YVd0c2FYTjBaVzVsY2kxdmNtY3hKakFrQmdOVkJBTU1IV\\
      1IIYm1GdGFXTnNhWE4wWlc1bGNpM0kWVVBeE5qazNNakk1TkRnNE1Ga3dFd11IS29aSXpqMENBU\\
      V1JS29aSXpqMERBUWNEUwdBRWMMWHBmNERGbt1GUQo4b0tYSkszMhjhjaNp1KytJcfRSK0JnVFpxM\\
      HBGU0xuQ1ZsQ1lyUldqdTdyYnZSeDNFUTlnVmYya0JhUDNYU9yCjBHdFp1YitjazzOQ01FQXdEZ\\
      11EV1IwUEFRSC9CQVFEQwdLa01BOedBMVVKRXdFQi93UUZNQJ1CQNY4d0hRWUQKV1IwT0JClUVGS\\
      kgza3lhMkQrb1ZxN1I3WXhWVVZKVU1ZNXIyTUFvR0NDcUdTTQ5QKFNQ0EwZ0FNRRVVDSUQ2dwowR\\
      E91K1A4NzJIQnkvb3ZNclRi0UtBbUNobTlaYmNBWXLrT2tHd1hxQw1FQnWxeCswQXZOZFFVdWhRV\\
      kdOaEY3Ci81dGppQzF4ZGNzc2cvUnprUHzakswPQotLS0tLUVORCBDRVJUSUZJQ0FURS0tLS0t"
users:
- name: "swe645assignment2"
  user:
    token: "kubeconfig-user-29ms5x2kr6:9rmcs5wz78bbntbhct7mjhg58jd7p87fhhz18z9rpvpcqb2bwmtmc6"
contexts:
- name: "swe645assignment2"
  context:
    user: "swe645assignment2"
    cluster: "swe645assignment2"
current-context: "swe645assignment2"

```

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

```

ubuntu@ip-172-31-86-97:~$ sudo su jenkins
jenkins@ip-172-31-86-97:/home/ubuntu$ cd ../../
jenkins@ip-172-31-86-97:$ cd /var/lib/jenkins
jenkins@ip-172-31-86-97:~$ cd /var/lib/jenkins/
jenkins@ip-172-31-86-97:~$ mkdir .kube
jenkins@ip-172-31-86-97:~$ cd .kube
jenkins@ip-172-31-86-97:~/ kube$ vi config
jenkins@ip-172-31-86-97:~/ kube$ kubectl config current-context
swe645assignment2

```

- 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.

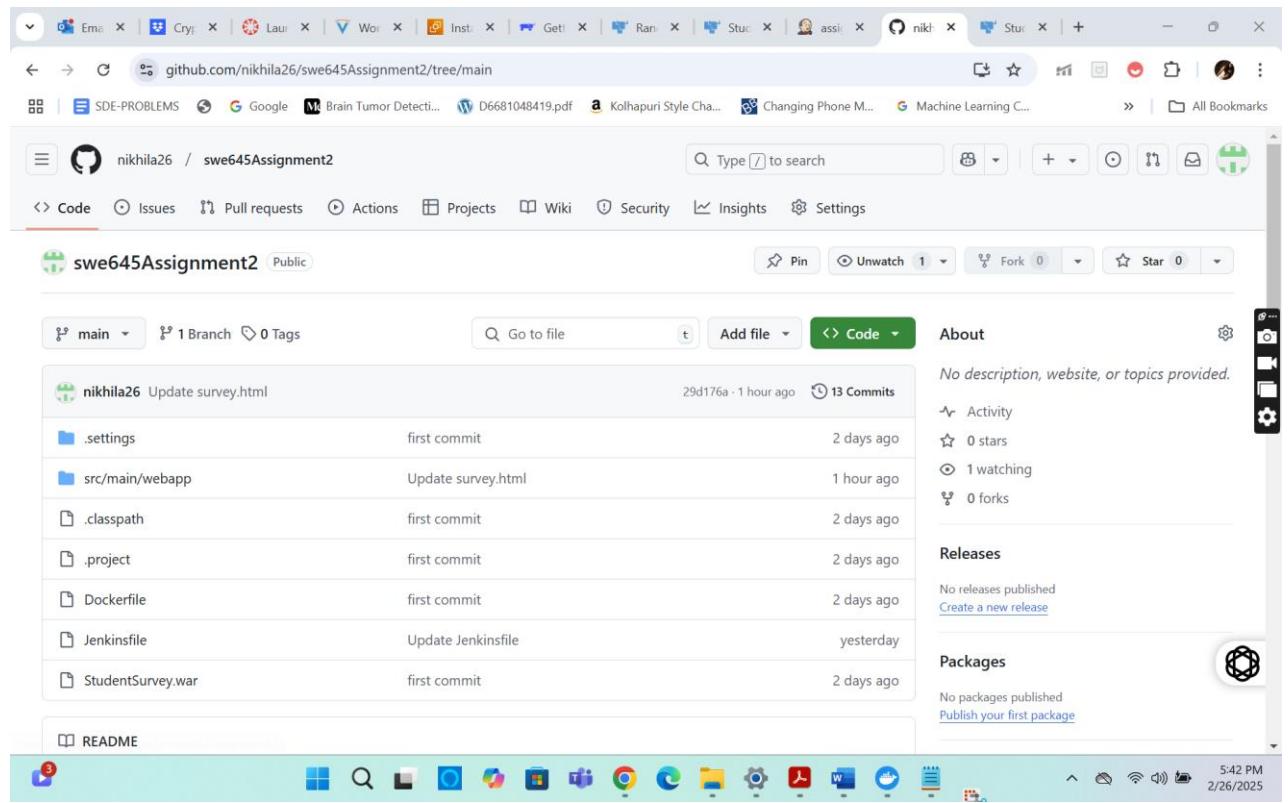
```
ubuntu@ip-172-31-86-97:~$ sudo apt install docker.io
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base pigz runc ubuntu-fan
Suggested packages:
  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 132 not upgraded.
Need to get 69.7 MB of archives.
After this operation, 267 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
```

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

```
ubuntu@ip-172-31-86-97:~$ sudo chmod 777 /var/run/docker.sock
ubuntu@ip-172-31-86-97:~$
```

#### **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/nikhila26/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.

Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >

Username with password

Scope ?  
Global (Jenkins, nodes, items, all child items, etc)

Username ?  
harenak

Treat username as secret ?

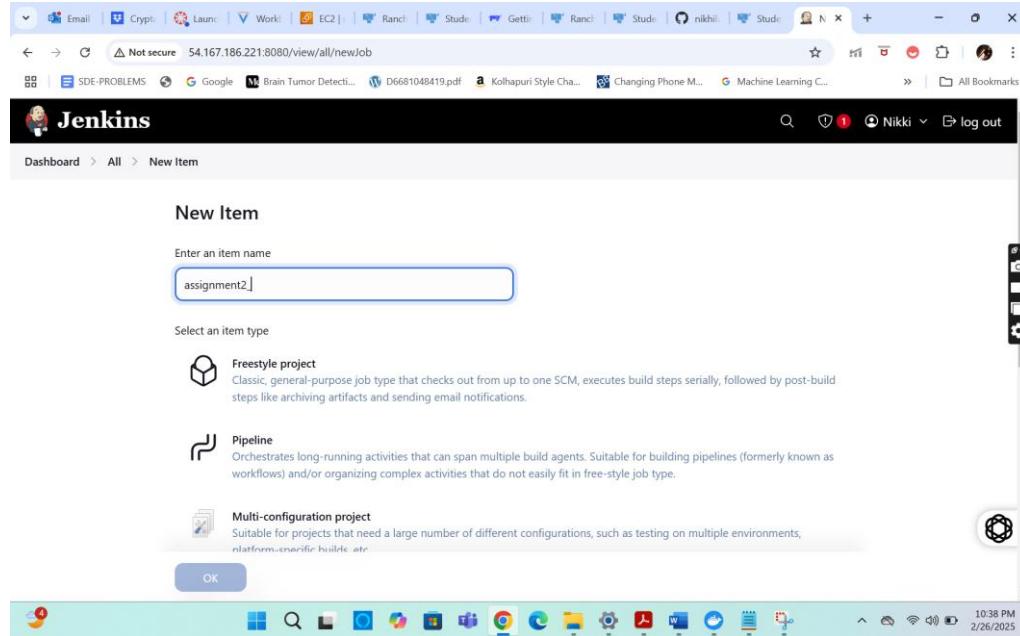
Password ?  
.....

ID ?  
dockerhub

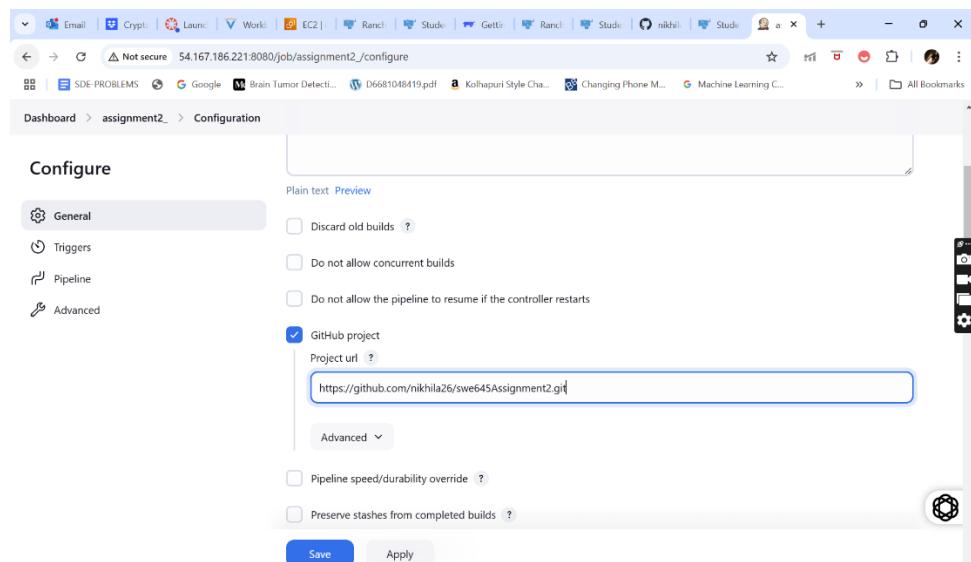
Description ?

**Create**

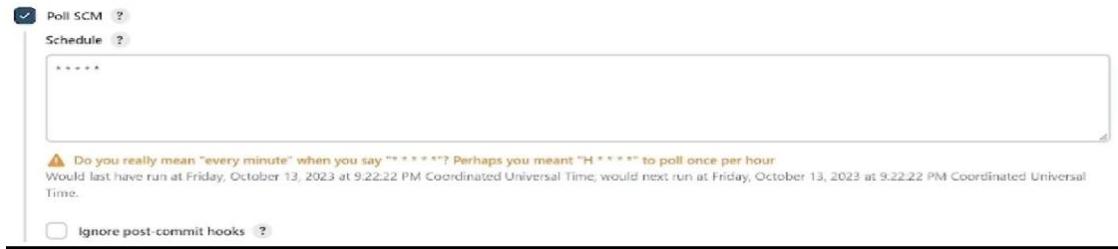
- Repeat the above steps to save GitHub credentials as well, give the id as “github”. For github, we will have to use PAT (Personal Access Tokens) as the password.
- Now go to the dashboard and click on “New Item”.
- Give the name as “assignment2” and click on “Pipeline” option.



- Now, on the next page, check the “Github Project” checkbox, and in the project URL, paste the URL from Github repository : <https://github.com/nikhila26/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/nikhila26/swe645Assignment2.git>”.
  - Under Credentials, select the github credentials.
  - Then, change the branch specifier : “\*/main”.

Configure

General

Triggers

Pipeline

Advanced

SCM ?

Git

Repositories ?

Repository URL ?

https://github.com/nikhila26/swe645Assignment2.git

Credentials ?

- none -
- nikhila10/\*\*\*\*\*
- nikhila26/\*\*\*\*\***

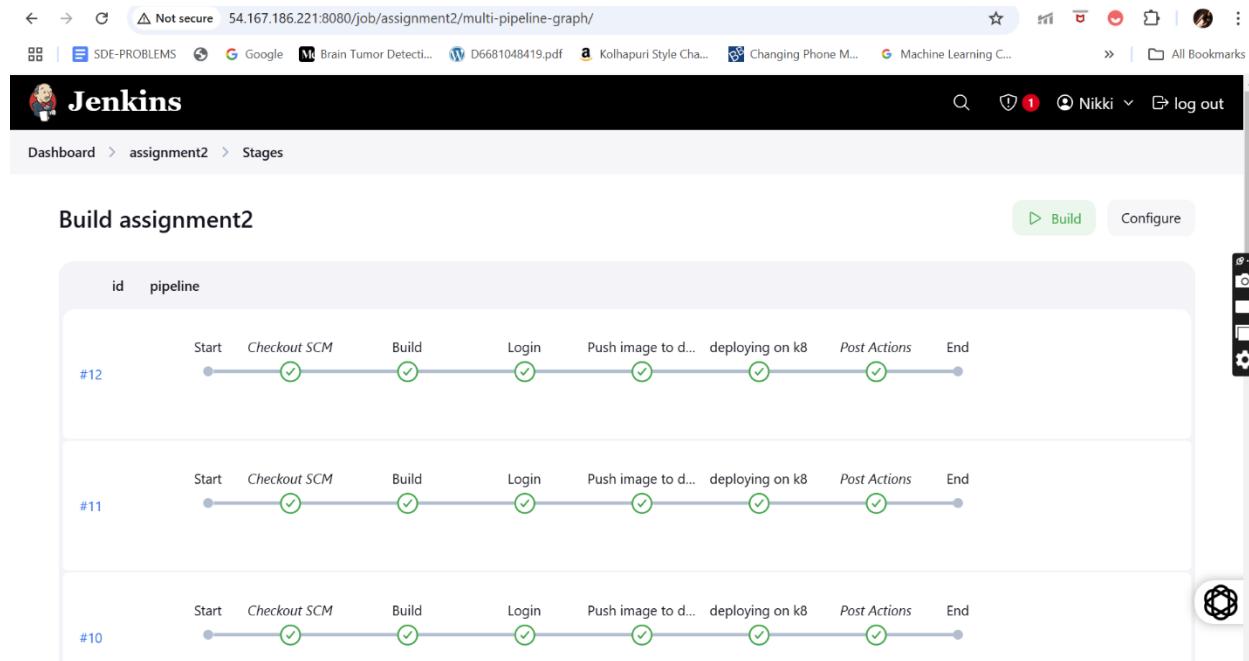
Save Apply

- 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 GitHub Actions pipeline editor. On the left, a sidebar lists repository settings, branches, Dockerfiles, Jenkinsfiles, and StudentSurvey.yaml. The main area displays a workflow named 'main' with the following code:

```
name: main
on: [push, pull_request]
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout code
        uses: actions/checkout@v2
      - name: Set up Java
        uses: actions/setup-java@v2
        with:
          java-version: 11
      - name: Build project
        run: mvn clean package
      - name: Publish artifact
        uses: actions/upload-artifact@v2
        with:
          name: student-survey
          path: target
  test:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout code
        uses: actions/checkout@v2
      - name: Set up Java
        uses: actions/setup-java@v2
        with:
          java-version: 11
      - name: Build project
        run: mvn clean package
      - name: Run tests
        run: mvn test
  deploy:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout code
        uses: actions/checkout@v2
      - name: Set up Java
        uses: actions/setup-java@v2
        with:
          java-version: 11
      - name: Build project
        run: mvn clean package
      - name: Deploy to AWS Lambda
        uses: actions/deploy-to-aws-lambda@v1
        with:
          function-name: student-survey
          code-path: target
  publish:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout code
        uses: actions/checkout@v2
      - name: Set up Java
        uses: actions/setup-java@v2
        with:
          java-version: 11
      - name: Build project
        run: mvn clean package
      - name: Publish artifact
        uses: actions/upload-artifact@v2
        with:
          name: student-survey
          path: target
```

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



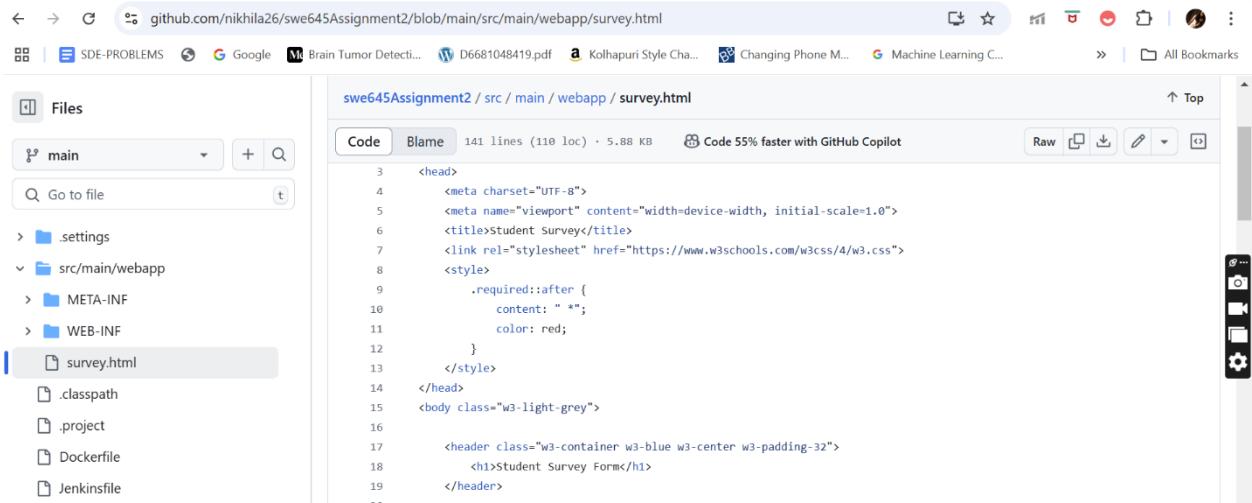
- Now after a successful build, open the application deployed on Kubernetes cluster using the same old URL. Don't forget to append the "/StudentSurvey/" to the URL again.
- And you should be able to see the application webpage.

The screenshot shows a web browser displaying a "Student Survey Form". The form contains the following fields with red asterisks indicating required input:

- First Name: \*
- Last Name: \*
- Email: \*
- Phone: \*
- Street Address: \*
- City: \*
- State: \*

The URL in the address bar is <https://ec2-54-173-40-206.compute-1.amazonaws.com/k8s/clusters/c-m-jds2lbj9/api/v1/namespaces/default/services/ht...>

- 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 heading panel on our website to blue color from pink. So, we edit the CSS file and commit the changes.



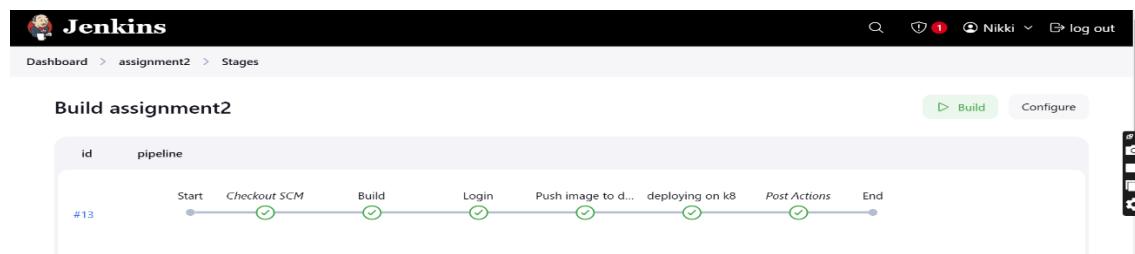
```

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Student Survey</title>
  <link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">
  <style>
    .required::after {
      content: " *";
      color: red;
    }
  </style>
</head>
<body class="w3-light-grey">

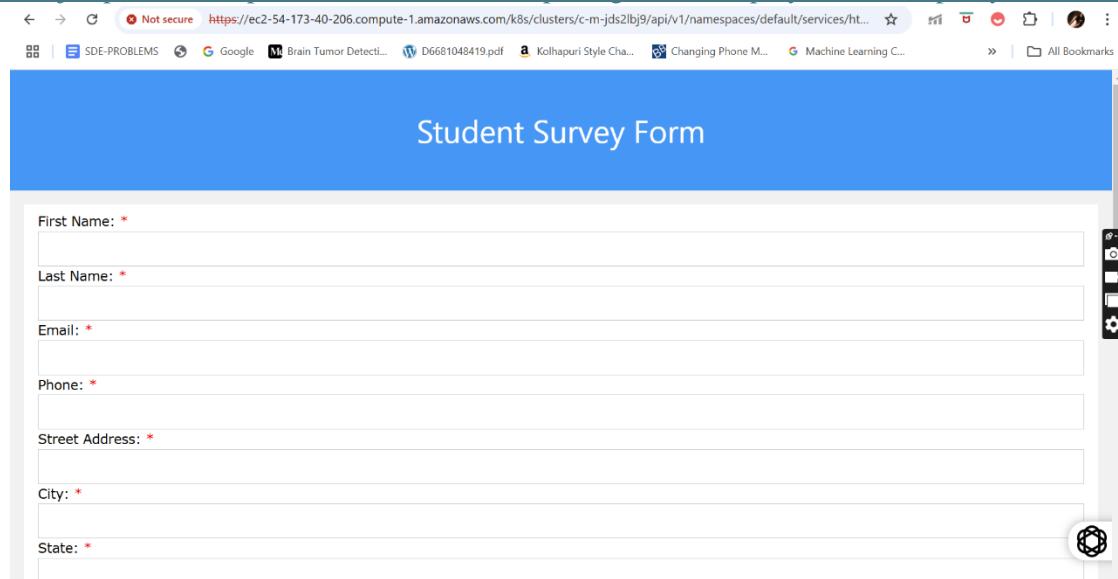
  <header class="w3-container w3-blue w3-center w3-padding-32">
    <h1>Student Survey Form</h1>
  </header>

```

- 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://ec2-54-173-40-206.compute-1.amazonaws.com/k8s/clusters/c-m-jds2lbj9/api/v1/namespaces/default/services/http:assignment2-deployment:8080/proxy/StudentSurvey/>



The screenshot shows a web browser displaying the Student Survey Form. The page title is "Student Survey Form". The form contains the following fields:

- First Name: \*
- Last Name: \*
- Email: \*
- Phone: \*
- Street Address: \*
- City: \*
- State: \*

The background color of the header and the input fields is gold, reflecting the changes made to the CSS file.

### **URL/Links of the application deployed**

1. GitHub URL : <https://github.com/nikhila26/swe645Assignment2.git>
2. Docker hub URL : <https://hub.docker.com/r/nikhila10/studentsurveyform>
3. Application deployed on cluster URL : <https://ec2-54-173-40-206.compute-1.amazonaws.com/k8s/clusters/cm-jds2lbj9/api/v1/namespaces/default/services/http:assignment2-deployment:8080/proxy/StudentSurvey/>
4. Docker URL : <http://localhost:8183/studentsurveyform/>
5. Rancher start URL : <https://www.rancher.com/quick-start>

### **References**

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

### **Group Members and their Contributions**

1. **Nikhila Parimi (G01456266)**: 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://g01456266-hw1-p1.s3-website-us-east-1.amazonaws.com>

2. **Manish Nalluri (G01453450)** : Worked on creating and deploying application on Kubernetes cluster, worked on creating the docker image and the Github repository. 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.