

645 Homework -2

By: N. Dheeraj Krishna, Akhilesh Dhavileswarapu
G01448319, G01472450

This manual outlines the possible ways of deploying a containerized web application using a pipeline of CI/CD with Docker, Kubernetes, Rancher, and Jenkins. Well, the implementation involves several interrelated modules that, in the end, would prove to be an excellent deployment architecture.

Overview

1. Create and publish a Docker image
2. Configure an AWS EC2 instance to deploy Kubernetes.
3. Rancher management for clusters.
4. Deploy an application to Kubernetes.
5. Create a GitHub repository for version control.
6. Configure Jenkins so that it will integrate with the CI/CD pipeline.
7. Build and set up CI/CD Pipeline

1) Create and publish a Docker image

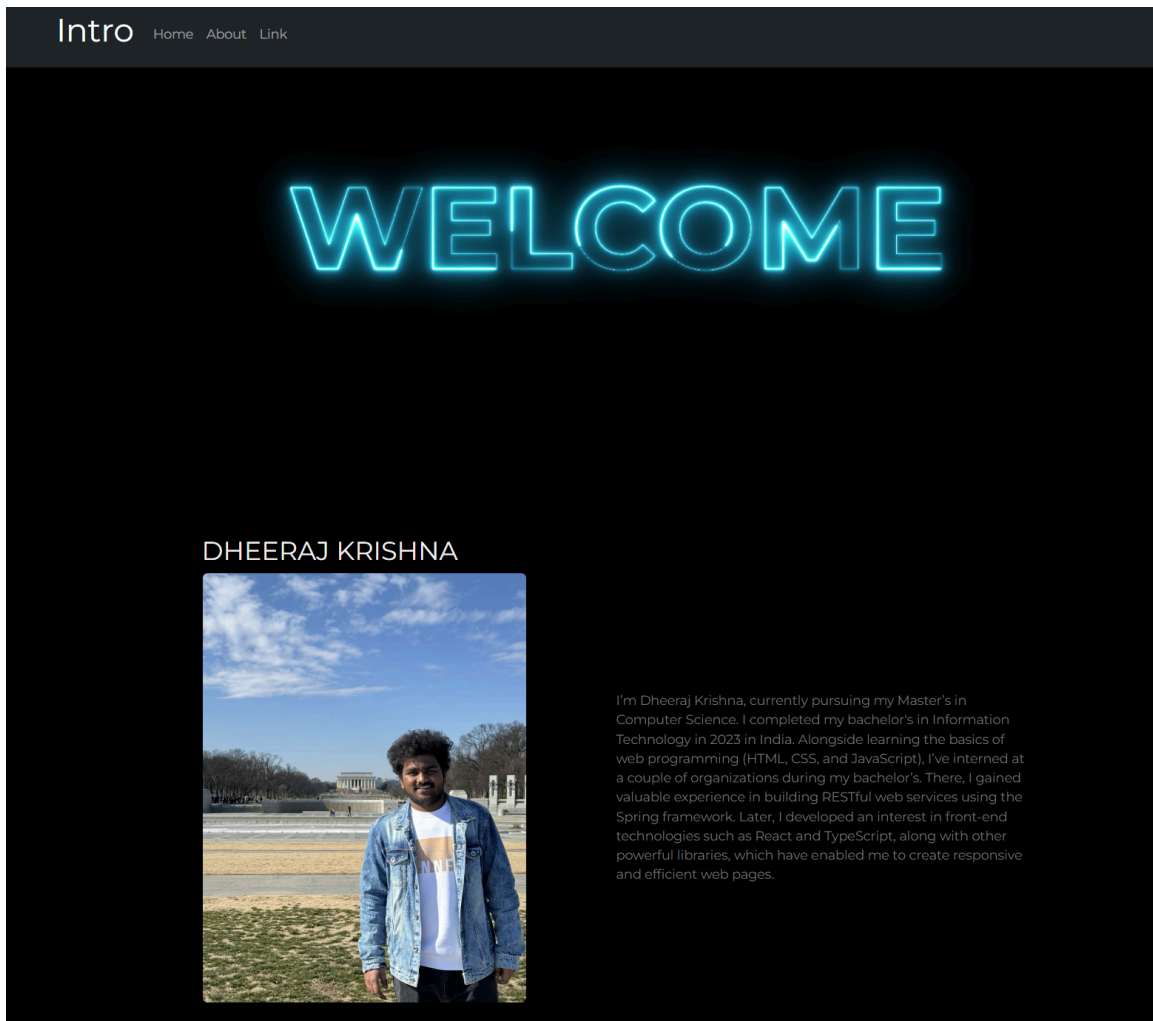
To containerize the application:

- Open account in Docker Hub and install Docker Desktop
- Place your WAR file and a new Dockerfile in the same directory
- In the Dockerfile, set up the Tomcat base image, copy the WAR file to the webapps directory, and set up port exposure like mentioned in the image below.

```
1 FROM tomcat:latest
2
3
4 RUN rm -rf /usr/local/tomcat/webapps/ROOT
5
6 COPY StudentSurvey.war /usr/local/tomcat/webapps/ROOT.war
7
8 EXPOSE 8080
9
10 CMD ["catalina.sh", "run"]
```

- Build the Docker image using the command `docker build -t <image name>:<tag>`
- Test locally using the command `docker run -d -p 8080:8080 <image name>` here we are mapping container port 8080 to local port 8080.
- Verify it against your personal local host URL, which is `http://localhost:8080/...`

This should load our html pages in the browser.



Heyy You!! Welcome to My Homepage



About Me

Student at GMU. Class of 2025

This is Akhilesh, I am a passionate software engineering student interested web development, and distributed systems. This website showcases my work for SWE645. Below you should be seeing two buttons. They will direct you to the pages which were asked for in the Assignment Rubric. Peace.

[Student Survey Form »](#)

[Go to Error Page »](#)



Department of Computer Science

Survey!

Let us know what are your thoughts about your first campus visit!

Firstname *

Lastname *

Street Address *

Zip Code *

City *

State *

Telephone Number *

Email *

Date of Survey *

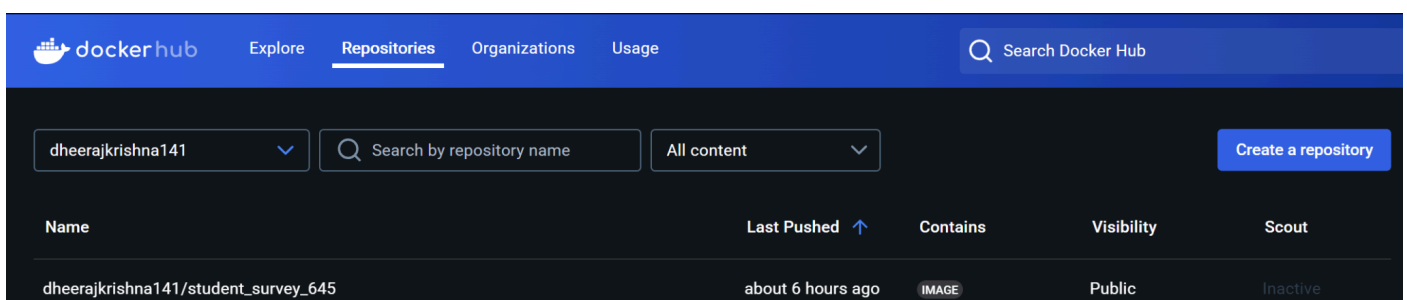


- Finally, tag and push the image to Docker Hub:

The application's image is now locally installed, and it is to be pushed from the Docker desktop to the Docker Hub. The following commands must then be executed from the same directory as above on the prompt.

First, log in to Docker Hub using the credentials of the previously created account: `docker login -u <docker id>` . Supply the password when prompted.

Next, type the commands `docker tag <source_image> <target_image>` and `docker push <target_image>` to tag the image for pushing and pushing into the hub. The image is now available on Docker Hub, as shown in the picture below.



2) Configure AWS EC2 instance for deploying the application on a Kubernetes cluster using Rancher.

For the remaining portions of the assignment, an AWS EC2 instance is of necessity.

- Log in first and then get into the AWS Academy, go to launch AWS, and select EC2 services then initiate the dashboard.
- Here click on Launch Instance to create the instance.
- "swe_645" is the instance name, "t3.large" is the instance type, and "Ubuntu Server SSD volume Type" is the AMI. Choose a key pair you already have like "swe_645". Enable HTTP and HTTPS traffic from the internet by clicking these boxes. Here storage increases from the default of 8GB to 32GB. Lastly, select the Launch instance.
- As soon as the instance is instantiated and becomes operational, get an elastic IP address for it.
- On the left pane of the screen in the EC2 menu, click on "Elastic Ips" and create an elastic IP with the default configuration, linking it to this instance. We do this since every time the AWS lab gets rebooted, the EC2 instances also go down and do not operate until the reboot is complete. The IP addresses change, hence problems are encountered with the Jenkins and Rancher configuration.



The screenshot shows the AWS Elastic IP addresses console. At the top, there's a title "Elastic IP addresses (2)" and a search bar. Below the search bar is a table with columns: Name, Allocated IPv4 address, Type, Allocation ID, Reverse DNS record, and Associated instance. There are two rows of data:

Name	Allocated IPv4 address	Type	Allocation ID	Reverse DNS record	Associated instance
swe_645_jenkins	35.170.214.169	Public IP	eipalloc-0111ec1e3f386b3ee	–	i-0004ec21af8fd6
swe_645EI	52.55.96.9	Public IP	eipalloc-031f0fe0c04cb7da1	–	i-0bb7643ea3a2ff

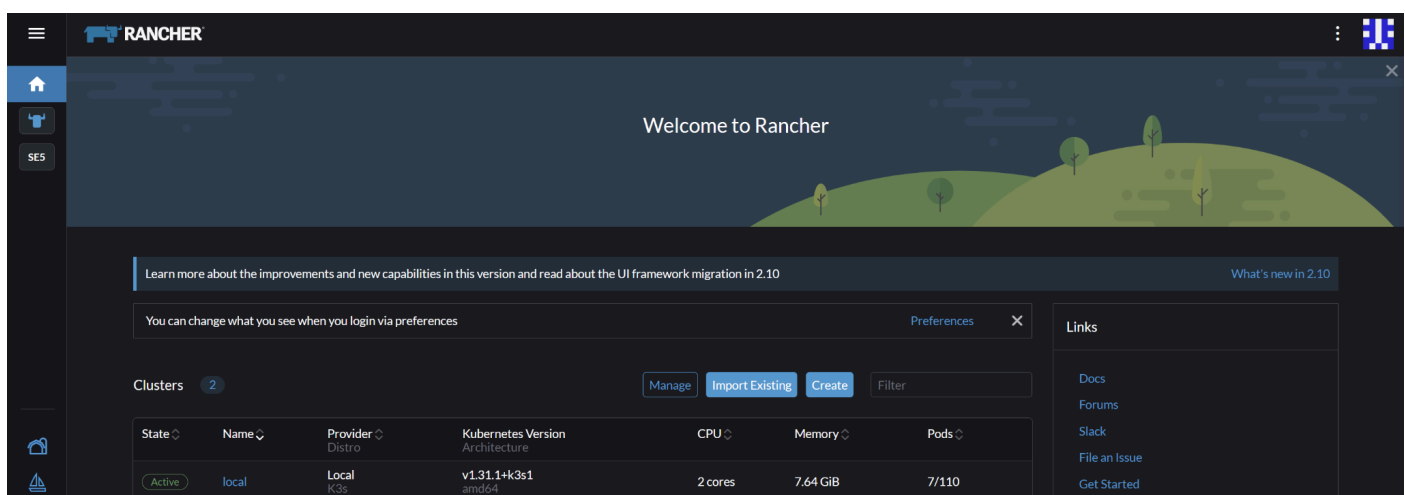
#IMAGE

Now, select the security tab and click on the instance. Enter the security group wizard in the "inbound rules"/"outbound rules" section.

- Click on the "edit rule" option within the "inbound" or "outbound" tabs.
 - Then you add a rule with the settings: Type: "Custom TCP," Port range: "8080," Source: "custom" and "0.0.0.0/0." Click "Save rules."
 - After that, click the selected EC2 instance and press the connect option. Choose the EC2 instance connect option and enter the username "root." Then click on connect. A new tab with a shell appears.
 - Then we tell you how to install Docker on the instances.
 - Commands to install docker: "sudo apt update" followed by "sudo apt install docker.io -y".
- On Successful completion of tasks, Docker should be successfully installed.

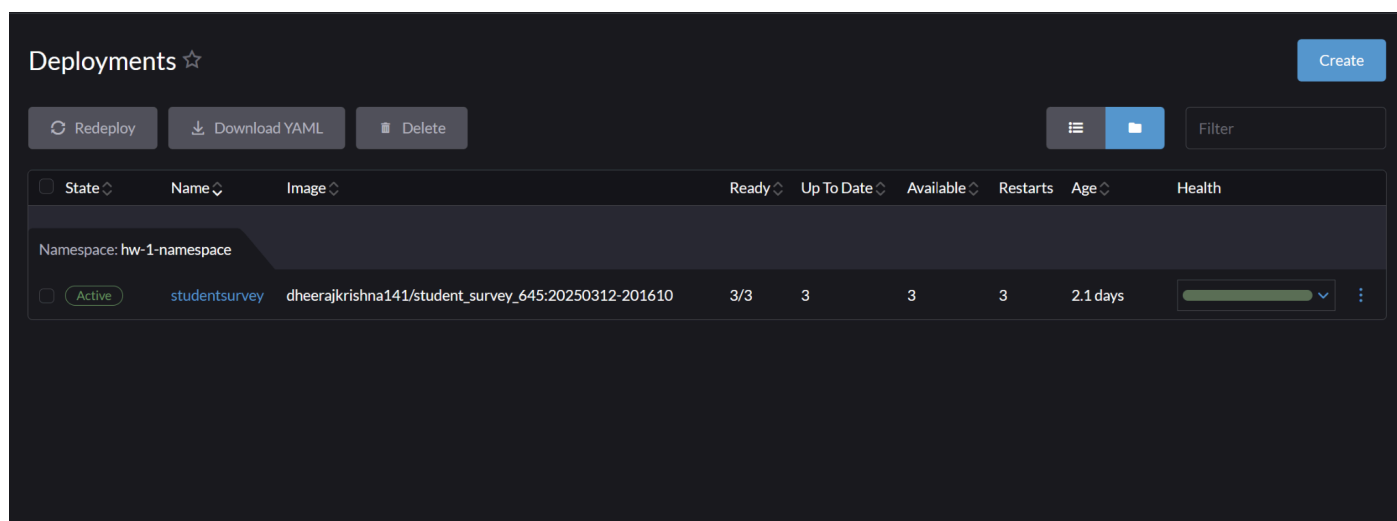
3) Setting Up Rancher

- Open a web browser and go to the page <https://www.rancher.com/quick-start>. There are two sections in the "Deploy Rancher" section you scroll: the first is "Start the Server."
- Copy that command, which is : `"$ sudo docker run --privileged=true -d --restart=unless-stopped -p 80:80 -p 443:443 rancher/rancher"`.
- Now connect to the instance in the same way as was described in the earlier section.
- Execute the command copied over here. Once it completes successful execution, Rancher will have been installed on this instance and can be accessed using the public IPv4 DNS address of the instance.
- To get information regarding the container that is running on this instance, use this command in the instance: `"sudo docker ps."` It will be required later, so save the output to a safe location (preferably notebook).
- Now go and open the "Public IPv4 DNS" link in another tab. Open the link now despite it being risky. The rancher login page should appear as it loads.
- There, it asks for a password. Get it by copying the command displayed above on the same webpage and changing the container id to the one from the earlier stored output and running it on the instance console. That command should end up resulting in the Password appearing on the screen.
- Paste it into the rancher login screen to log in.
- After you log in, follow the prompts on the screen to create a new password.
- Now the rancher dashboard should appear for us to continue with the creation of the cluster and application deployment.



4. Using the Rancher UI to deploy the application and set up a Kubernetes cluster.

- With Rancher in place, it will allow us, through the Rancher UI, to create a cluster for deploying our application.
- On the dashboard, click the "Create Cluster" button. After selecting the "Custom" option from among the many cluster options, a form should appear for cluster creation.
- Select "Registration" on the next page and verify that on "step1," the three checkboxes labelled etcd, control plane, and worker are checked.
- Now, copy the command from "step2" below.
- Here, run the copied command but append "`—insecure·`" after the word 'curl' in the command and execute.
- After a successful run of the command, the cluster should be in the 'Updating' state and will transition into the 'Active' state a few minutes later, at which point it is ready for deployment. You can see this option in the Rancher UI under 'Clusters' within 'cluster management'.
- After the cluster is in the 'Active' state, click the 'Explore' button representing the cluster. On the left pane under 'workloads', select 'Deployments'.
- Upon clicking the 'create' button, a form should appear on the screen.
- Fill in the form with these details. Namespace: default Name: swe645deploy Replicas: 3 (no. of Pods) Container image: **dheerajkrishna141/student_survey_645: latest**
- Under Networking, select the "Add Port or Service" button.
- Enter the following: Protocol: TCP, Private Container Port: 8080, Service Type: node Port, Name: nodeport.
- Leave everything else as it is and click create.
- The deployment has completed, and it should be "Updating". After some minutes, it should be in "Active".
- Once in Active, click the deployment, going to the Services tab.



Deployment: studentsurvey Active
Detail Config YAML

Namespace: hw-1-namespace Age: 2.1 days Pod Restarts: 3

Image: dheerajkrishna141/student_survey_645:20250312-201610 Ready: 3/3 Up-to-date: 3 Available: 3
Annotations: [Show 1 annotation](#)

Pods by State

3

Running


Scale - 3 +

Pods Services Ingresses Conditions Recent Events Related Resources

Download YAML Delete

<input type="checkbox"/>	State	Name	Image	Ready	Restarts	IP	Node	Age	
<input type="checkbox"/>	Running	studentsurvey-5cc7fcd7b-lv8db	dheerajkrishna141/student_survey_645:20250312-201610	1/1	1 (175m ago)	10.42.126.113	ip-172-31-76-254	7 hours	
<input type="checkbox"/>	Running	studentsurvey-5cc7fcd7b-nqb5h	dheerajkrishna141/student_survey_645:20250312-201610	1/1	1 (175m ago)	10.42.126.112	ip-172-31-76-254	7 hours	
<input type="checkbox"/>	Running	studentsurvey-5cc7fcd7b-sncct	dheerajkrishna141/student_survey_645:20250312-201610	1/1	1 (175m ago)	10.42.126.118	ip-172-31-76-254	7 hours	

- Click the button “nodeport” to open the deployment in a new tab under the target option. Then add "/survey.html" to the URL and click "Open". The application ought to be visible to you as it runs.



Department of **Computer Science**

Survey!

Let us know what are your thoughts about your first campus visit!

Firstname *


LastName *

Street Address *

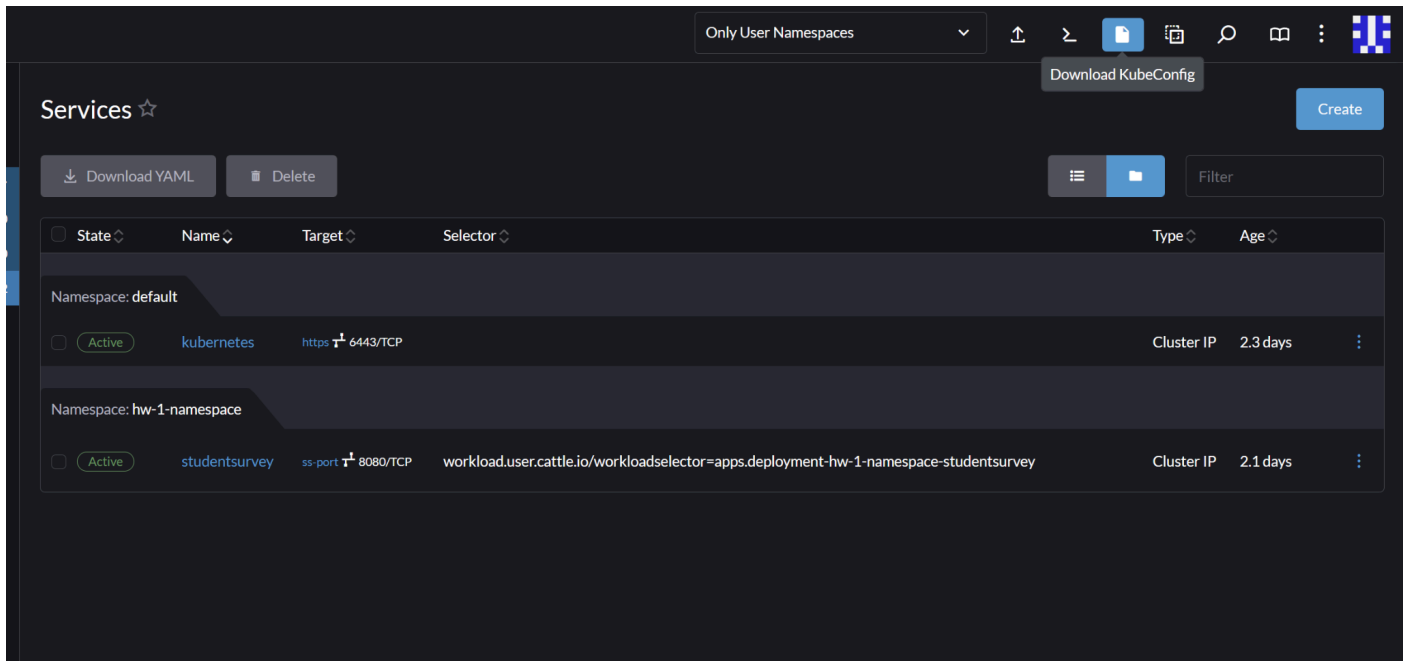
Zip Code *

City *

State *



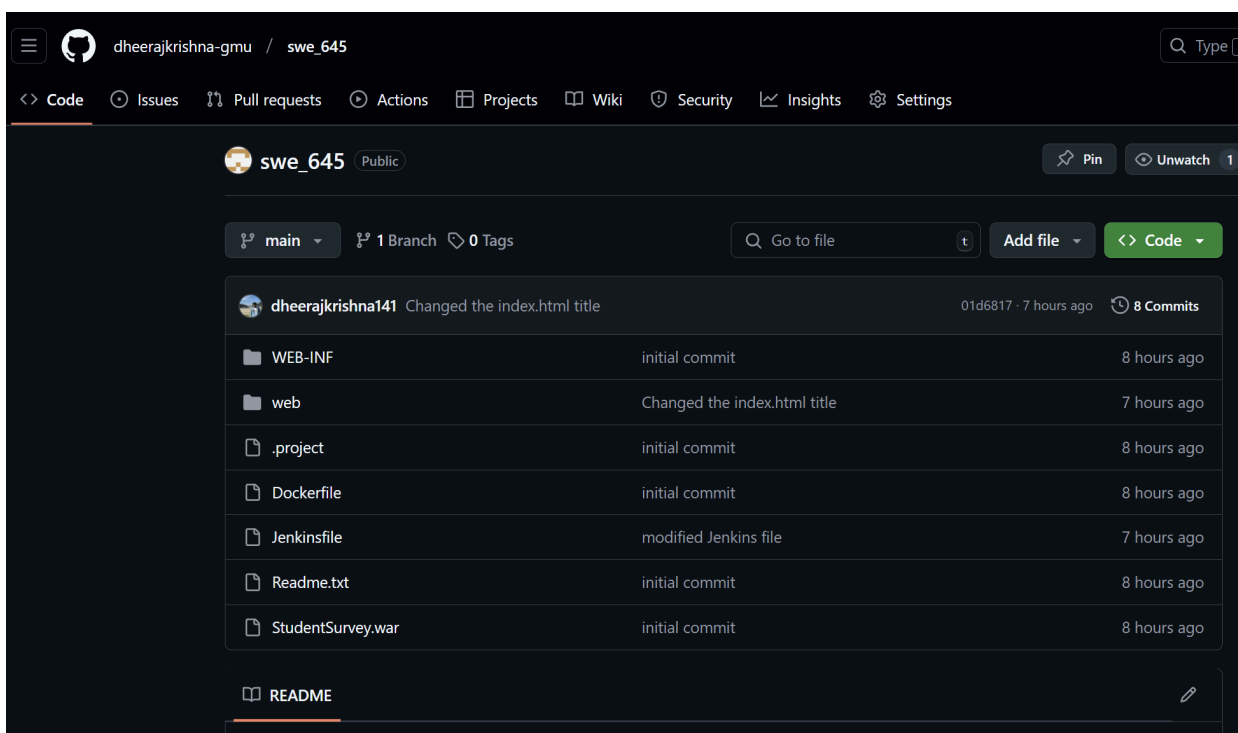
The next thing required is the "KubeConfig" file, which can be obtained by visiting our cluster page. Click the "Download KubeConfig file" option in the upper right and save it to the local drive.



5. Setting up the project's GitHub repository.

A source code should be saved somewhere to attach to the pipeline and can be used for change detection prior to generating it. We are going to use GitHub for this.

- Create an account and log in at <https://github.com/>.
- Next, create a new repository: dheerajkrishna-gmu/swe_645
- Add your source files, docker file and war file to that repository.
- Finally, commit them.
- This will be the repository that our pipeline consumes.
- Github repository URL.



6. Setting up an AWS EC2 instance for Jenkins installation and operation.

- The straight path begins at this place - and with Jenkins we begin to give the application a self-sufficient pipeline.
- We will first install Jenkins in our computer, and from there we shall come to set up an AWS EC2 instance in our AWS laboratory.
- We shall configure the instance for deploying our app onto the Kubernetes cluster like we did for the other EC2 instance.
- Before using Jenkins, it requires that Java be installed first.
- For that, use the command "sudo apt update".
- Now install it with the command "*sudo apt install openjdk-17-jdk -y*".
- Now install Jenkins through the following commands, that is,
"*sudo wget -O /usr/share/keyrings/jenkins-keyring.asc https://pkg.jenkins.io/debianstable/jenkins.io2023.key; echo "deb [signedby=/usr/share/keyrings/jenkins-keyring.asc] https://pkg.jenkins.io/debian-stable binary/ | sudo tee /etc/apt/sources.list.d/jenkins.list > /dev/null*" and "*sudo apt-get update*" followed by next step,
- *sudo apt-get install Jenkins -y* and then Jenkins should be installed successfully.
- Now run jenkins using command : "*sudo systemctl start jenkins.service*".
- Verify the status of the jenkins service by using "*sudo systemctl status jenkins*", it should show the status like shown in the image below.

```
jenkins@ip-172-31-76-93:~/.kube$ sudo systemctl status jenkins
Warning: The unit file, source configuration file or drop-ins of jenkins.service changed on disk. Run 'systemctl daemon-reload' to reload units.
• jenkins.service - Jenkins Continuous Integration Server
   Loaded: loaded (/usr/lib/systemd/system/jenkins.service; enabled; preset: enabled)
   Active: active (running) since Thu 2025-03-13 00:50:13 UTC; 3h 13min ago
     Main PID: 547 (java)
       Tasks: 55 (limit: 9365)
      Memory: 484.3M (peak: 494.3M)
         CPU: 1min 25.008s
        CGroup: /system.slice/jenkins.service
                └─547 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080

Mar 13 00:50:10 ip-172-31-76-93 jenkins[547]: 2025-03-13 00:50:10.103+0000 [id=39] INFO jenkins.InitReactorRunner$1#onAttained: Started all plugins
Mar 13 00:50:10 ip-172-31-76-93 jenkins[547]: 2025-03-13 00:50:10.299+0000 [id=39] INFO jenkins.InitReactorRunner$1#onAttained: Augmented all extensions
Mar 13 00:50:10 ip-172-31-76-93 jenkins[547]: 2025-03-13 00:50:10.745+0000 [id=36] INFO h.p.b.g.GlobalTimeoutConfigurationLoad: global timeout not set
Mar 13 00:50:12 ip-172-31-76-93 jenkins[547]: 2025-03-13 00:50:12.654+0000 [id=38] INFO jenkins.InitReactorRunner$1#onAttained: System config loaded
Mar 13 00:50:12 ip-172-31-76-93 jenkins[547]: 2025-03-13 00:50:12.661+0000 [id=36] INFO jenkins.InitReactorRunner$1#onAttained: System config adapted
Mar 13 00:50:13 ip-172-31-76-93 jenkins[547]: 2025-03-13 00:50:13.136+0000 [id=37] INFO jenkins.InitReactorRunner$1#onAttained: Loaded all jobs
Mar 13 00:50:13 ip-172-31-76-93 jenkins[547]: 2025-03-13 00:50:13.139+0000 [id=36] INFO jenkins.InitReactorRunner$1#onAttained: Configuration for all jobs updated
Mar 13 00:50:13 ip-172-31-76-93 jenkins[547]: 2025-03-13 00:50:13.246+0000 [id=39] INFO jenkins.InitReactorRunner$1#onAttained: Completed initialization
Mar 13 00:50:13 ip-172-31-76-93 jenkins[547]: 2025-03-13 00:50:13.300+0000 [id=30] INFO hudson.lifecycle.Lifecycle#onReady: Jenkins is fully up and running
Mar 13 00:50:13 ip-172-31-76-93 systemd[1]: Started jenkins.service - Jenkins Continuous Integration Server.
jenkins@ip-172-31-76-93:~/.kube$
```

Expose port 8080 using command : "*sudo ufw allow 8080*".

- Run the following command to get the admin password: "*sudo cat /var/lib/jenkins/secrets/initialAdminPassword*". It will display the password. Store it safely now.
- Run the following commands to install snapd: "*sudo apt install snapd*".
- Now with "*sudo snap install kubectl --classic*", install kubectl using snap.
- Now, proceed to the "Jenkins" AWS EC2 instance and browse the "Public IPv4 address" in a fresh tab.
- To the url append the port 8080 while changing from "https" to "http". The URL in our case is *http://35.170.214.169:8080/*

- Use the admin password obtained earlier to access Jenkins.
- Install the recommended plugins next.
- Next, create an admin user. "Save and create user," "Save and finish," and "Start using Jenkins" had better be clicked in succession.
- The Jenkins dashboard should now be visible.
- Now go back to the EC2 console. We now build a configuration file.
- Execute the following commands: "*sudo su jenkins*" to go to jenkins home.
- Then *cd ../../* to go to the root directory.
- *cd /var/lib/jenkins* to head to that directory.
- *mkdir .kube*; *cd .kube* to create and enter a folder.
- *vi config* to create and open that file.
- Now copy from the KubeConfig file downloaded earlier and paste it here.
- Then save and quit with *:wq*.

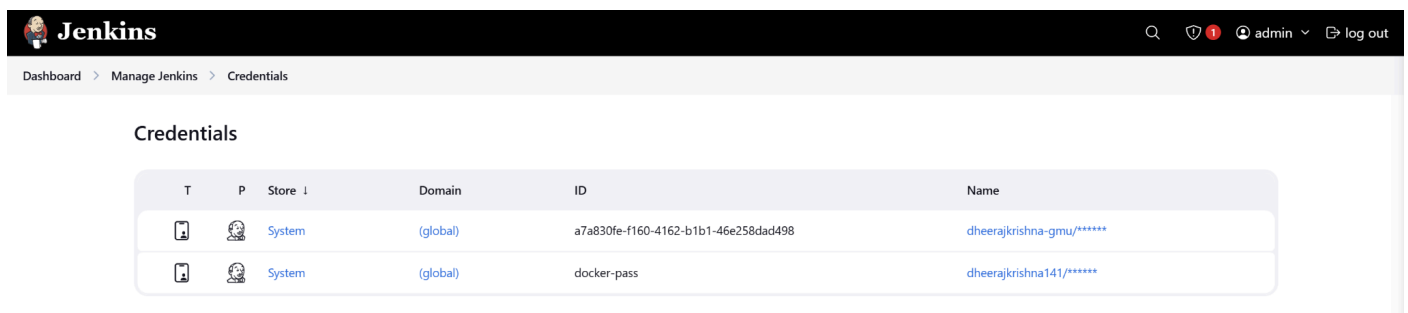
Use the command `"kubectl config current-context"` to verify, and it should return the cluster name as output.

- In the next bread, exit from Jenkins mode by using the command "exit."

- Now install docker using the following commands: "*sudo apt update*" and "*sudo apt install docker.io -y*".
- Install Docker. Using "*sudo apt-get update*" and "*sudo apt update*" followed by "*sudo apt install docker.io.*" Answer with a "Y."

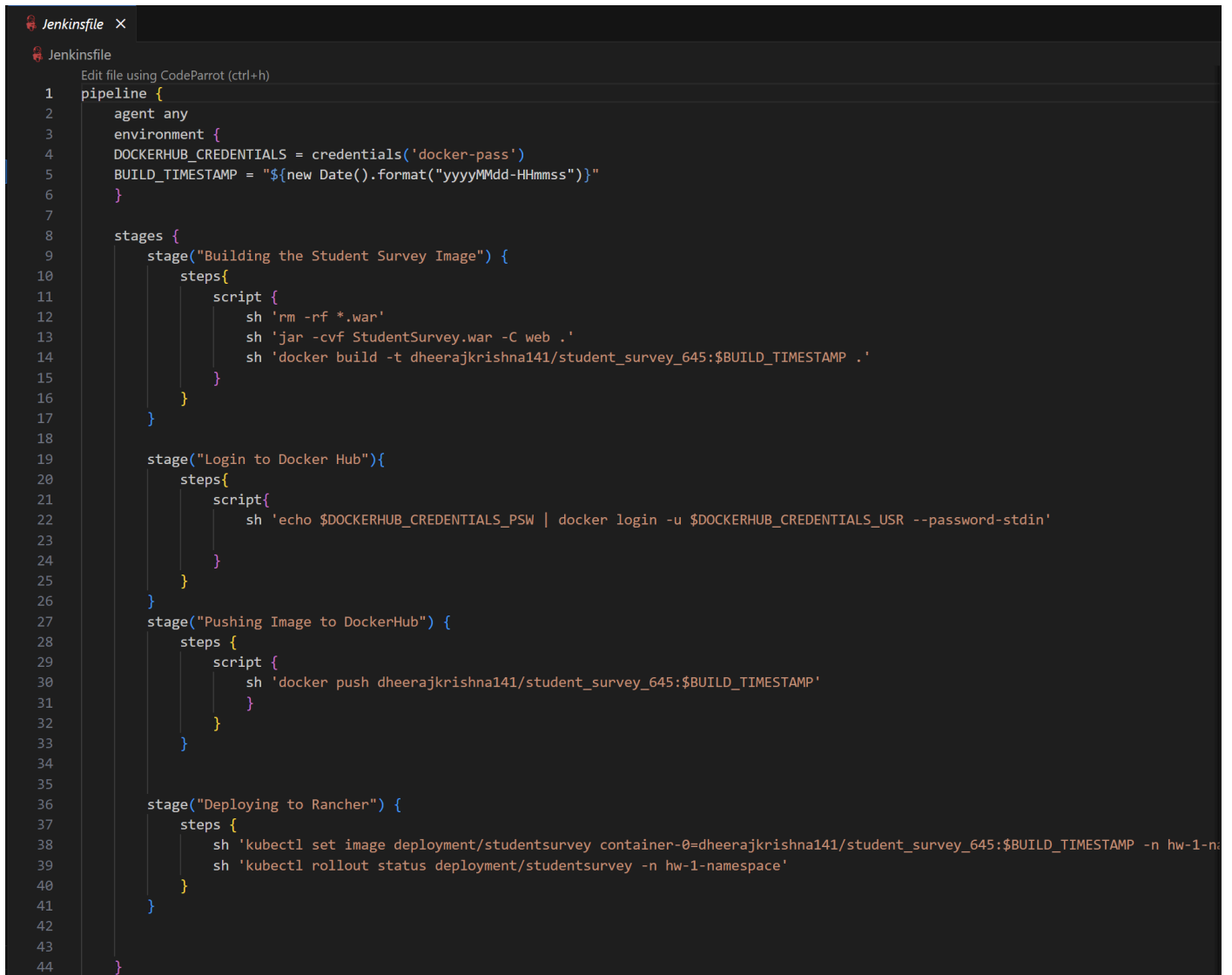
7. Constructing a CI/CD pipeline and running it fully configured on Jenkins.

- All is ready for the actual pipeline creation.
- Next, on the Jenkins Dashboard, click on Credentials, scroll down, and click Manage Jenkins.
- Now, in the Global section, click on Add Credentials.
- You will find at this point the form asking for Kind: "Username and password," in which you should enter your docker hub username and password with ID "dockerhub."
- Do the same for saving GitHub credentials, giving ID as "github."



- Now, go to the dashboard and select "New Item."
- Name it pipeline and select "Pipeline."
- In the next window, check the "GitHub Project" checkbox, and in the project URL, paste the URL from GitHub repository:
- Proceed downward to the "Poll SCM" checkbox and select it.
- The schedule should be: "* * * * *."
- Go down to the "Pipeline" area.
- Under the definition, choose: "Pipeline from SCM" and SCM: "git".
- In GitHub repository enter: "https://github.com/dheerajkrishna-gmu/StudentSurvey".
- Under Credentials select the GitHub credentials.
- Now change the branch specifier to: "*/main".
- Go to GitHub repository and create a file named: "Jenkinsfile". Commit the changes.
- Click Save.
- This should now kick off a build of that pipeline.
- Now, go to the GitHub repository ->"Jenkinsfile" and click edit.

- Enter in commands shown in the screenshot below and commit the changes. These commands should be run on every commit or change in the files of the repository to build a new war file, log into Docker and push the image, and finally deploy that image on the Kubernetes cluster.



```
1 pipeline {
2   agent any
3   environment {
4     DOCKERHUB_CREDENTIALS = credentials('docker-pass')
5     BUILD_TIMESTAMP = "${new Date().format("yyyyMMdd-HH:mm:ss")}"
6   }
7
8   stages {
9     stage("Building the Student Survey Image") {
10      steps{
11        script {
12          sh 'rm -rf *.war'
13          sh 'jar -cvf StudentSurvey.war -C web .'
14          sh 'docker build -t dheerajkrishna141/student_survey_645:$BUILD_TIMESTAMP .'
15        }
16      }
17    }
18
19    stage("Login to Docker Hub"){
20      steps{
21        script{
22          sh 'echo $DOCKERHUB_CREDENTIALS_PSW | docker login -u $DOCKERHUB_CREDENTIALS_USR --password-stdin'
23        }
24      }
25    }
26
27    stage("Pushing Image to DockerHub") {
28      steps {
29        script {
30          sh 'docker push dheerajkrishna141/student_survey_645:$BUILD_TIMESTAMP'
31        }
32      }
33    }
34
35
36    stage("Deploying to Rancher") {
37      steps {
38        sh 'kubectl set image deployment/studentsurvey container-0=dheerajkrishna141/student_survey_645:$BUILD_TIMESTAMP -n hw-1-n'
39        sh 'kubectl rollout status deployment/studentsurvey -n hw-1-namespace'
40      }
41    }
42
43
44  }
```