

SWE 645 : HOME WORK 2

Kiran Muddana – G01413581

Venu Dammalapati – G01460891

Yashasree Marram – G01380842

This guide will walk through deploying an application using a CI/CD pipeline. Here's a breakdown of the steps:

1. Containerize your application: We'll build a Docker image and store it in Docker Hub.
2. Prepare the deployment environment: We'll set up Amazon EC2 instances to run a Kubernetes cluster using Rancher.
3. Deploy the application: We'll create a Kubernetes cluster on the EC2 instances and deploy your containerized application there.
4. Set up continuous integration and delivery (CI/CD): We'll configure a Jenkins server on another EC2 instance and create a pipeline to automate building, testing, and deploying your application.
5. Version control: We'll assume you have a Git repository set up to store your application code.

1. Containerize your application:

1. Setting Up Docker:

- Create a Docker Hub account at <https://hub.docker.com/>.
- Download Docker Desktop for a graphical interface (optional).
- Login to Docker Desktop or Docker Hub using your account credentials.

2. Building the Docker Image:

Create a file named Dockerfile in the same directory as your ".war" file (e.g. SurveyForm.war).

Edit Dockerfile to include the specific instructions for your application.

```
FROM tomcat:10.1-jdk21
COPY SurveyForm.war /usr/local/tomcat/webapps/
EXPOSE 8080
```


4. Pushing the Image to Docker Hub:

Login to Docker Hub using your credentials: `docker login -u <username>` (skm05 in my case) and enter the password

Tag the image for pushing: `docker tag studentsurveyform`

`<username>/studentsurveyform` (replace `<username>` with your actual username).

Push the image to Docker Hub: `docker push <username>/studentsurveyform`.

```
C:\Windows\System32\cmd.e x + v
Microsoft Windows [Version 10.0.22631.3296]
(c) Microsoft Corporation. All rights reserved.

C:\Users\skmud\OneDrive\Desktop\MS\645\Assignment2>docker login -u skm05
Password:

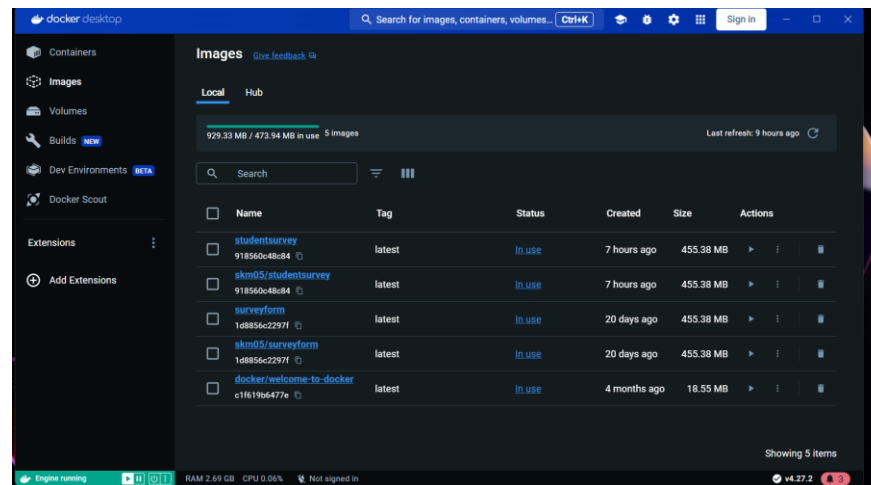
Login Succeeded

C:\Users\skmud\OneDrive\Desktop\MS\645\Assignment2>docker tag studentsurvey skm05/studentsurvey

C:\Users\skmud\OneDrive\Desktop\MS\645\Assignment2>docker push skmo5/studentsurvey
Using default tag: latest
The push refers to repository [docker.io/skmo5/studentsurvey]
An image does not exist locally with the tag: skmo5/studentsurvey

C:\Users\skmud\OneDrive\Desktop\MS\645\Assignment2>docker push skm05/studentsurvey
Using default tag: latest
The push refers to repository [docker.io/skmo5/studentsurvey]
64ae749c4a30: Pushed
f57cd456a2a2: Mounted from library/tomcat
88ca17fbd2cd: Mounted from library/tomcat
6f37c5d016c8: Mounted from library/tomcat
c1793f638462: Mounted from library/tomcat
4df676480fd4: Mounted from library/tomcat
d9d7e0852802: Mounted from library/tomcat
cbb50874016a: Mounted from library/tomcat
5498e8c22f69: Mounted from library/tomcat
latest: digest: sha256:7e026a0c616ffd1c8f63ab064260ef719160e95ddad9dc62e5f6c45f7d913079 size: 2203

C:\Users\skmud\OneDrive\Desktop\MS\645\Assignment2>|
```



This process creates a Docker image for your application, allowing you to run it consistently on any machine with Docker installed. You can then push the image to Docker Hub for sharing or deployment in a containerized environment.

localhost:8184/SurveyForm/

Student Survey Form

First Name:

Last Name:

Street Address:

City:

State:

ZIP:

Telephone Number:

E-mail:

Date of Survey:

What did you like most about the course?

2. Prepare the deployment environment :

1. Launch and Configure the Instance:

Log in to your AWS account and navigate to the Amazon Elastic Compute Cloud (EC2) service.

Click on "Launch Instance" to create a new instance.

Name: Enter "homework2demo" for the instance name.

Launch an instance

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags

Name: homework2demo

Application and OS Images (Amazon Machine Image)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below.

Search our full catalog including 1000s of application and OS images

Summary

Number of instances: 1

Software Image (AMI): Amazon Linux 2023 AMI 2023.3.2...read more

Virtual server type (Instance type): t2.micro

Firewall (security group): New security group

Storage (volumes): 1 volume(s) - 8 GiB

Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 750 hours of public IPv4 address usage per month, 30 GiB

Launch Instance

Machine Image (AMI): Select "Ubuntu Server 22.04 LTS (HVM) SSD volume Type".

Instance Type: Choose "t3.large" for the instance type.

Key Pair: Select your existing key pair named "homework2". This allows you to connect to the instance securely.

Security Group: Check the boxes to allow both HTTP and HTTPS traffic from anywhere on the internet. This is necessary for certain functionalities.

Storage: Increase the storage size from the default 8 GB to 30 GB for better performance.

Click "Launch Instances" to create the instance.

The screenshot shows the 'Configure storage' tab in the AWS Management Console. It displays a configuration for a single volume (1x) of 30 GiB using the gp2 storage type. A note indicates that free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage. Below this, there is a section for 'Advanced details' which includes a warning about the selected AMI containing more instance store volumes than the instance allows, and a section for 'File systems' showing 0 x File systems. On the right side, there is a summary of the instance configuration: ami-080e1f13689e07408, Virtual server type (instance type) t3.large, Firewall (security group) New security group, and Storage (volumes) 1 volume(s) - 30 GiB. A 'Free tier' notice is also visible, stating that in the first year, it includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 750 hours of public IPv4 address usage per month, and 30 GiB. At the bottom, there are 'Cancel' and 'Launch instance' buttons, along with a 'Review commands' link.

2. Assign an Elastic IP Address:

Once the instance is running, navigate to "Elastic IPs" in the EC2 menu on the left side of the screen.

Create a new Elastic IP address using the default settings.

The screenshot shows the 'Create Elastic IP address' dialog in the AWS Management Console. It is set to the 'us-east-1' region. Under 'Public IPv4 address pool', 'Amazon's pool of IPv4 addresses' is selected. Below this, there are options for 'Public IPv4 address that you bring to your AWS account with BYOIP' (disabled) and 'Customer-owned pool of IPv4 addresses created from your on-premises network for use with an Outpost' (disabled). Under 'Global static IP addresses', there is a 'Create accelerator' button. At the bottom, there is a 'Tags - optional' section with an 'Add new tag' button. The dialog has 'Cancel' and 'Associate' buttons at the bottom right.

Associate this newly created Elastic IP address with the "homework2demo" instance you just created.

Elastic IP address: 54.243.111.113

Resource type
Choose the type of resource with which to associate the Elastic IP address.

☒ Instance
☐ Network interface

Warning: If you associate an Elastic IP address with an instance that already has an Elastic IP address associated, the previously associated Elastic IP address will be disassociated, but the address will still be allocated to your account. [Learn more](#)

If no private IP address is specified, the Elastic IP address will be associated with the primary private IP address.

Instance
i-06cdc8d18338a296d

Private IP address
The private IP address with which to associate the Elastic IP address.
Choose a private IP address

Reassociation
Specify whether the Elastic IP address can be reassociated with a different resource if it already associated with a resource.
☐ Allow this Elastic IP address to be reassociated

3. Configure Security Group for Docker Access:

Go to the "homework2demo" instance and navigate to the "Security Groups" tab.

Click on "Security group wizard" within the "Inbound rules" section.

Scroll down and click "Edit rule" in the "Inbound" tab.

Create a new rule with these settings:

Type: "Custom TCP"

Port Range: "8080"

Source: "Custom" (allows access from anywhere)

Source Address: "0.0.0.0/0" (allows access from any IP address)

Click "Save rules" to apply the changes.

Inbound rules

Security group rule ID	Type	Protocol	Port range	Source	Description - optional
sg-02ee9646ae44c58	SSH	TCP	22	Custom	0.0.0.0/0
sg-0c2f3d0764473eb	HTTP	TCP	80	Custom	0.0.0.0/0
sg-08865ac4358d1d6	HTTPS	TCP	443	Custom	0.0.0.0/0
-	Custom TCP	TCP	8080	Anyw...	0.0.0.0/0

[Add rule](#)

Rules with source of 0.0.0.0/0 or ::/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

[Cancel](#) [Preview changes](#) [Save rules](#)

4. Connect and Install Docker:

Click on the "homework2demo" instance and then "Connect."

Go to the "EC2 Instance Connect" tab, enter username "ubuntu," and click "Connect." This opens a terminal window.

Update package lists: "sudo apt-get update".

Install Docker: "sudo apt install docker.io"

When prompted, type "Y" to grant permission for the installation.

Verify successful installation.

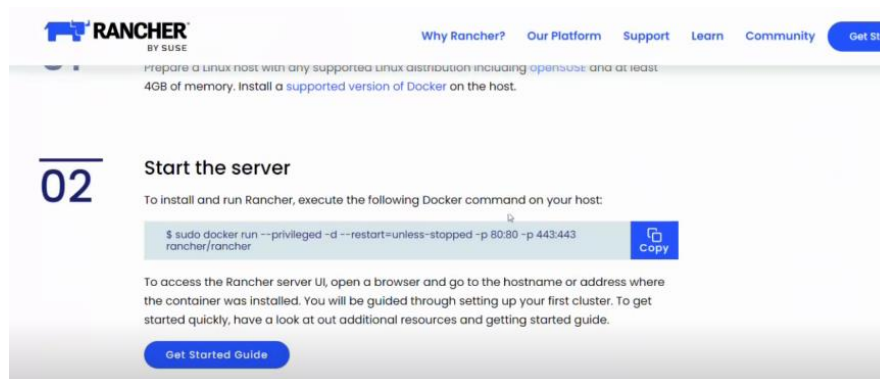
```
Reading 30.2 MB in 3s (30.2 MB/s)
Reading package lists... Done
root@ip-172-31-33-131:~# sudo apt install docker.io
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  bridge-utils containerd dns-root-data 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 19 not upgraded.
Need to get 69.8 MB of archives.
After this operation, 263 MB of additional disk space will be used.
```

5. Get the Rancher Server Command:

Visit the Rancher quick start guide: <https://www.rancher.com/quick-start>

Scroll down to "Start the Server" under "Deploy Rancher."

Copy the provided command, which will look something like: `$ sudo docker run --privileged -d -restart=unless-stopped -p 80:80 -p 443:443 rancher/rancher`

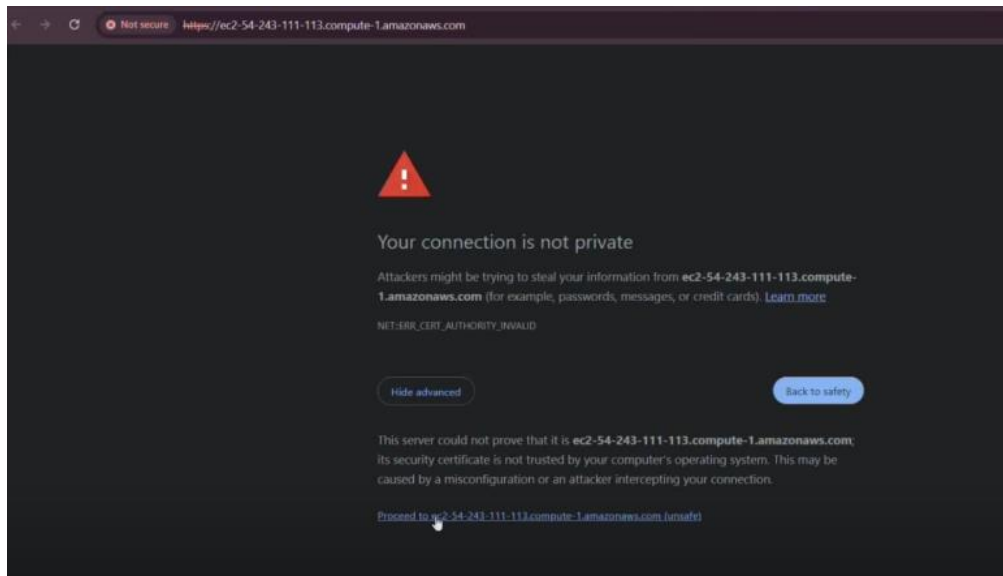


6. Install Rancher on the Node:

Connect to your "homework2demo" instance using EC2 Instance Connect (same method as before).

Paste the copied Rancher server command and run it.

Once successful, Rancher is installed and accessible through your master node's public IP address.



7. Get the Container ID and Password:

Run `sudo docker ps` on the master node to list running containers.

Note the container ID for the Rancher container.

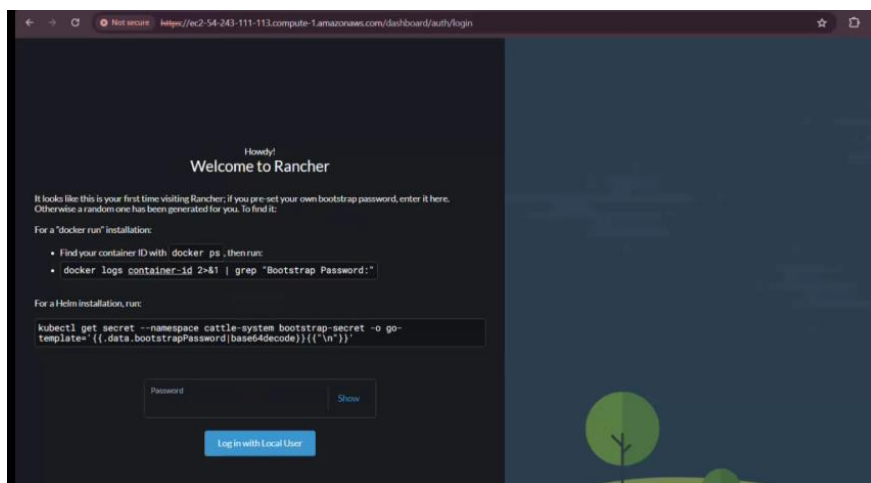
In your browser, open the master node's public IP address (even if the security warning appears). The Rancher login page should load.

On the login page, you'll need a password.

Go back to the instance console and run a modified version of the docker logs command, replacing the container ID with the one you noted:

```
sudo docker logs <container_id> 2>&1 | grep "Bootstrap Password:"
```

This will display the Rancher password in the console.




```

root@ip-172-31-33-131:~# docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS
AMES
1d816892c58f   rancher/rancher  "entrypoint.sh"         About a minute ago    Up About a mi
everent_sanderson
root@ip-172-31-33-131:~# docker logs container-id 2>41 | grep "Bootstrap Password:"
root@ip-172-31-33-131:~#

```

8. Login to Rancher:

Copy the password from the console and paste it into the Rancher login screen.

Login and set up a new password following the on-screen prompts.

9. Rancher Dashboard:

You'll now see the Rancher dashboard, where you can create a Kubernetes cluster and deploy your application.

3. Deploy the application

Create a Custom Cluster:

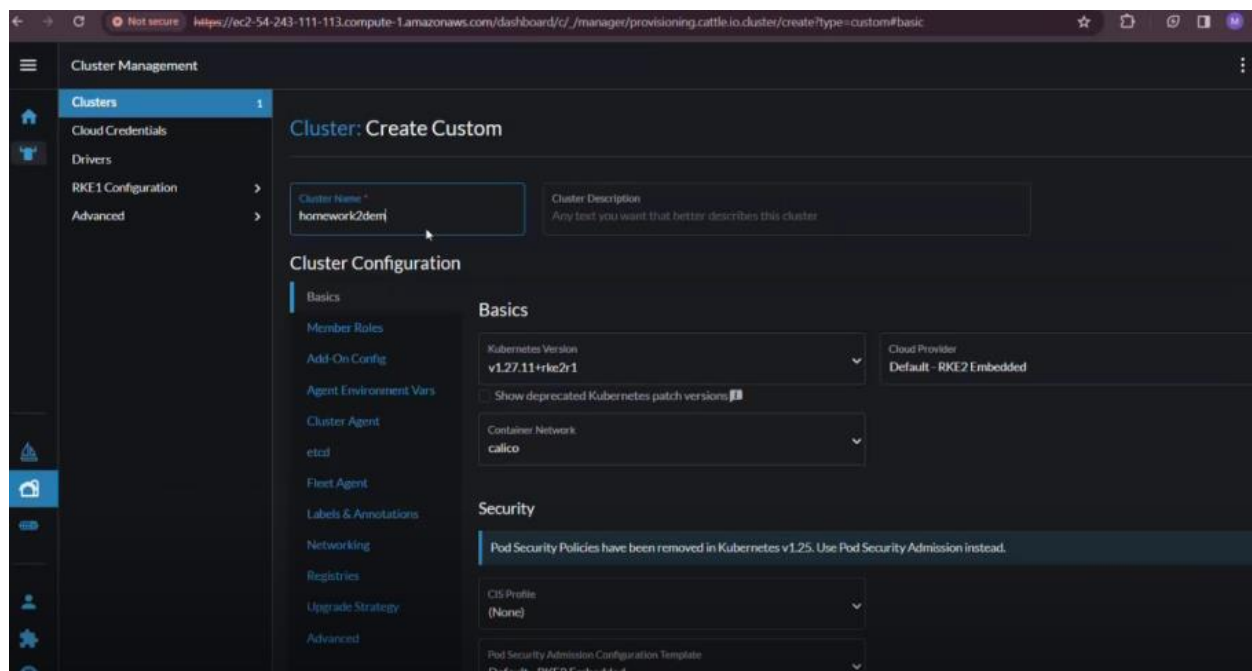
In the Rancher dashboard, click "Create Cluster."

Choose "Custom" from the options and fill out the form:

Cluster Name: "homework2demo"

Leave all other settings as default.

Click "Create" to initiate cluster creation.



2. Register the Node :

Navigate to the "Registration" tab on the cluster creation page.

Under "Step 1," ensure all checkboxes for "etcd," "Control Plane," and "Worker" are selected.

Copy the registration command displayed under "Step 2."

Connect to your " homework2demo " node via EC2 Instance Connect

Run the copied command on the worker node, appending --insecure after curl.

3. Verify Cluster Status:

The cluster status should change from "Updating" to "Active" after a few minutes.

You can monitor the progress in the "Clusters" section under "Cluster Management" on the Rancher UI.

4. Deploy the Application:

Once the cluster is active, click "Explore" next to the cluster name.

In the left pane, navigate to "Workloads" and select "Deployments."

Click "Create" to start defining your deployment.

Fill in the deployment details:

Namespace: "default"

Name: "assignment2-deployment"

Replicas: "3" (number of pods running the application)

Container Image: "skm05/studentsurveyform:latest" (your Docker Hub image name)

5. Configure Service for NodePort Access:

Under the "Networking" section, click "Add port or service."

Configure the service with:

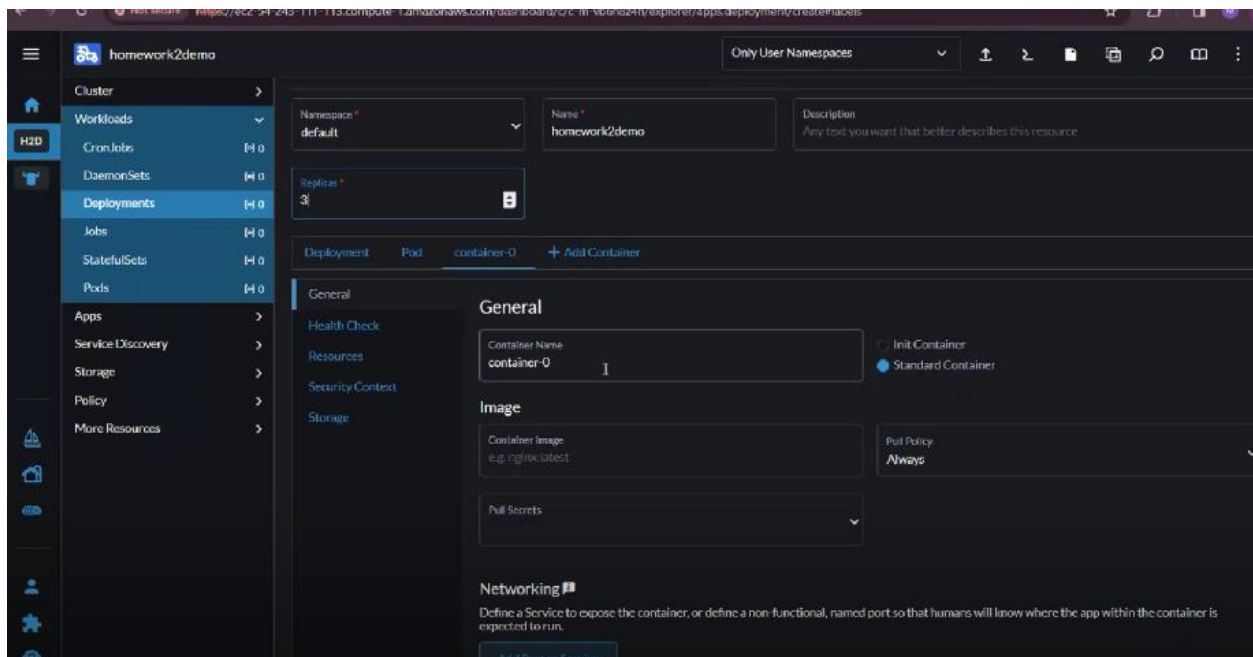
Service Type: "NodePort" (allows access from outside the cluster)

Name: "nodeport"

Private Container Port: "8080" (port your application listens on)

Protocol: "TCP" (communication protocol)

Leave other settings as default and click "Create."



6. Access the Application:

After the deployment finishes (may take a few minutes), navigate to the "Services" tab in the deployment details.

Find the "nodeport" service and click on it.

In the target URL, append `"/SurveyForm"` (your application endpoint) and open it in a new browser tab.

You should now see your application running at the provided URL (similar to this format):

7. Download Kubeconfig File:

Go back to the cluster page in Rancher.

From the top right menu, click "Download KubeConfig file." Save this file locally for future interaction with your Kubernetes cluster.

4. Set up continuous integration and delivery (CI/CD):

Begin by creating a new AWS EC2 instance in the AWS lab environment.

Follow the same process as before for creating EC2 instances to deploy the application on the Kubernetes cluster.

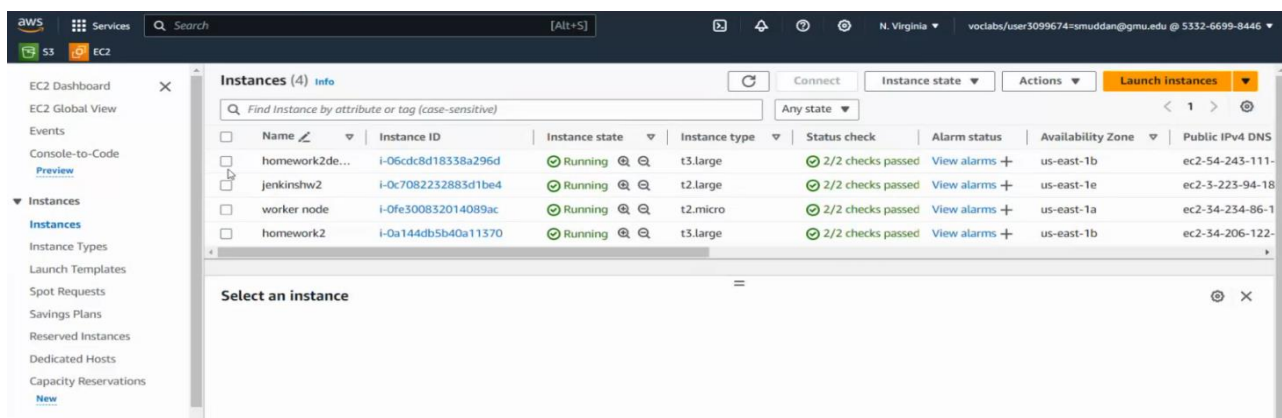
Name the EC2 instance "Jenkins".

Utilize the same configuration parameters as used previously, including choosing the appropriate AMI, instance type, and key pair.

Assign an elastic IP to the Jenkins instance to maintain a static IP address, ensuring consistency even during AWS lab restarts.

Adjust the security group settings for the Jenkins instance as needed, ensuring necessary inbound and outbound rules are configured.

Once the instance is created and running, proceed to set up Jenkins on this instance to establish a CI/CD pipeline for automated build and deployment processes.



To connect to the EC2 instance using "EC2 instance connect," ensure that the instance is in a "running" state. Then follow these steps:

Update the package index on the instance by running: `sudo apt-get update`

Update the package index again using: `sudo apt update`

```
aws Services Search [Alt+S] N. Virginia voclabs/user3099674:smuddan@gmu.edu @ 5332-6699-8446
S3 EC2
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
root@ip-172-31-33-131:~# 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]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 c-n-f Metadata [286 kB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse amd64 Packages [217 kB]
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse Translation-en [112 kB]
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse amd64 c-n-f Metadata [8372 B]
Get:11 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [1490 kB]
Get:12 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main Translation-en [288 kB]
Get:13 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 Packages [1605 kB]
Get:14 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/restricted Translation-en [268 kB]
Get:15 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [1057 kB]
Get:16 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe Translation-en [239 kB]
Get:17 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 c-n-f Metadata [22.1 kB]
Get:18 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 Packages [42.1 kB]
Get:19 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/multiverse Translation-en [10.1 kB]
Get:20 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 c-n-f Metadata [472 B]
Get:21 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [1246 kB]
Get:22 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/main amd64 Packages [67.1 kB]
Get:23 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/main Translation-en [11.0 kB]
Get:24 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/main amd64 c-n-f Metadata [388 B]
Get:25 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/restricted amd64 c-n-f Metadata [116 B]
Get:26 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/universe amd64 Packages [28.4 kB]
Get:27 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/universe Translation-en [16.2 kB]
Get:28 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/universe amd64 c-n-f Metadata [644 B]
Get:29 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/multiverse amd64 c-n-f Metadata [116 B]
```

i-06cdc8d18338a296d (homework2demo)
PublicIPs: 54.243.111.113 PrivateIPs: 172.31.33.131

Install OpenJDK 11 by running the following command: `sudo apt install openjdk-11-jdk`

When prompted for permission, type "Y" and press Enter to proceed with the installation.

Following these steps will install Java JDK 11 on the EC2 instance, preparing it for Jenkins installation.

```
aws Services Search [Alt+S] N. Virginia voclabs/user3099674:smuddan@gmu.edu @ 5332-6699-8446
S3 EC2
Get:38 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/multiverse amd64 c-n-f Metadata [116 B]
Fetched 30.2 MB in 5s (6409 KB/s)
Reading package lists... Done
ubuntu@ip-172-31-59-109:~$ sudo apt install openjdk-11-jdk
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-data libavahi-common3 libcups2 libdconf1 libdrm-amdgpu1 libdrm-intel1 libdrm-nouveau2 libdrm-radeon1 libfontconfig1
  libfontenc1 libgl1 libgl1-mesa-dri libgl1-mesa-glx libglapi-mesa libglvnd0 libglx-mesa0 libglx0 libgraphite2-3 libharfbuzz0b libice-dev libice6
  libjpeg-turbo8 libjpeg8 liblcms2-2 libllvm55 liblpcaccess0 libpcsc-lite libpthread-stubs0-dev libensors-config libensors5 libsm-dev libsm6 libx11-dev libx11-xcb1
  libxau-dev libxaw7 libxcb-dri2-0 libxcb-dri3-0 libxcb-glx0 libxcb-present0 libxcb-randr0 libxcb-shape0 libxcb-shm0 libxcb-sync1 libxcb-xf86vm0 libxcb1-dev
  libxcomposite1 libxdmcp-dev libxfixes3 libxft2 libxi6 libxinerama1 libxkbfile1 libxmu6 libxpm4 libxrandr2 libxrender1 libxshmfence1 libxt-dev libxt6 libxtst6 libxv1
  libxxf86dgal 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 alsa-utils cups-common libice-doc liblcms2-utils psmisc 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 libatk1.0-0 libatk1.0-data
  libatspi2.0-0 libavahi-client3 libavahi-common-data libavahi-common3 libcups2 libdconf1 libdrm-amdgpu1 libdrm-intel1 libdrm-nouveau2 libdrm-radeon1 libfontconfig1
  libfontenc1 libgl1 libgl1-mesa-dri libgl1-mesa-glx libglapi-mesa libglvnd0 libglx-mesa0 libglx0 libgraphite2-3 libharfbuzz0b libice-dev libice6
  libjpeg-turbo8 libjpeg8 liblcms2-2 libllvm55 liblpcaccess0 libpcsc-lite libpthread-stubs0-dev libensors-config libensors5 libsm-dev libsm6 libx11-dev libx11-xcb1
  libxau-dev libxaw7 libxcb-dri2-0 libxcb-dri3-0 libxcb-glx0 libxcb-present0 libxcb-randr0 libxcb-shape0 libxcb-shm0 libxcb-sync1 libxcb-xf86vm0 libxcb1-dev
  libxcomposite1 libxdmcp-dev libxfixes3 libxft2 libxi6 libxinerama1 libxkbfile1 libxmu6 libxpm4 libxrandr2 libxrender1 libxshmfence1 libxt-dev libxt6 libxtst6 libxv1
  libxxf86dgal 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
0 upgraded, 95 newly installed, 0 to remove and 19 not upgraded. I
```

i-Oe2c617723ccdc75b (jenkinsdemo)
PublicIPs: 34.224.54.50 PrivateIPs: 172.31.59.109

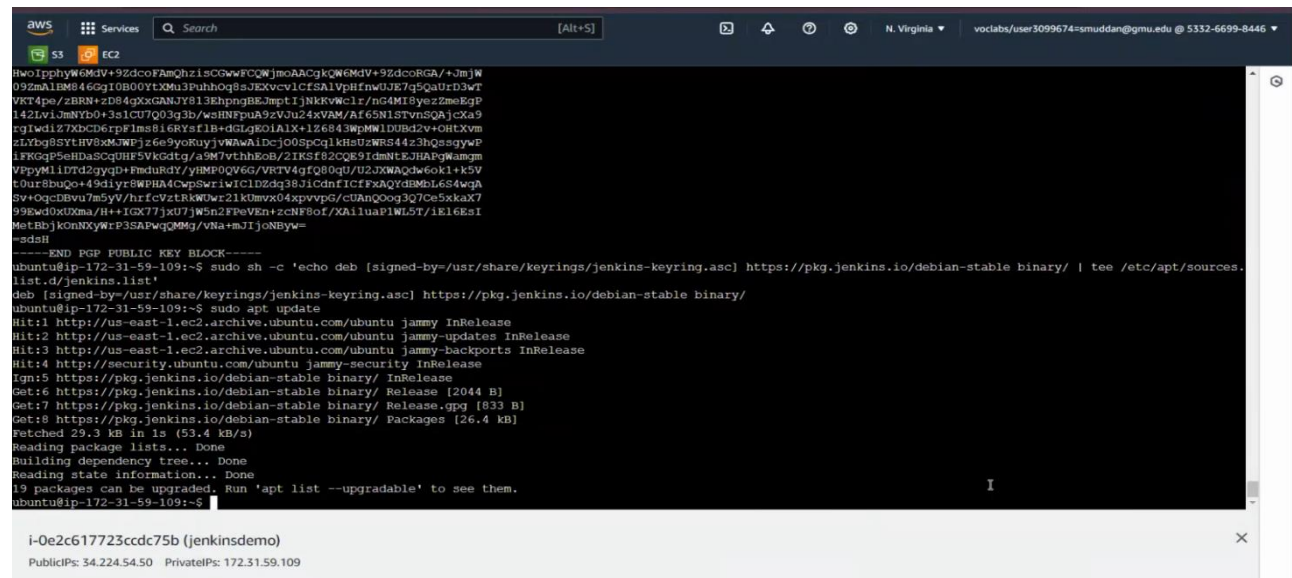
To install Jenkins on the EC2 instance, follow these steps:

Obtain the Jenkins package key by running the following command:

```
sudo curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | sudo tee
/usr/share/keyrings/jenkins-keyring.asc > /dev/null
```

Add the Jenkins package repository by running:

```
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
```



```
aws
Services
Search
[Alt+S]
N. Virginia
voclabs/user3099674:smuddan@gmu.edu @ 5332-6699-8446
SS EC2
HwoIpphyW6Mdv+9Zdc0FAMQhZisCGwWFCQWjmoAACgkQW6Mdv+9Zdc0RGA/+JmJW
09ZmAlBMS46c0I0B00YEXMu3PunhoqasJEKvcv1Cf5AlVpHfneUJE7q5qaurD3wT
Vxt4pe/zlRM+z084pKxGMU7r13BapqjE3mpt1jMRVWw1z/n04MI3yzzm0EgP
142Lw1JmMYb0+3s1CD7Q03q3b/waHNPua3zV3u24kVAM/Af65N1STvNSQj)ckA9
rgIwd1Z7XbcD6rfP1as816Ysf1B+dGLqE01A1X+1Z6843wPMW1DUBd2v+OHTXvm
zLYbG8SYtHV8xM.WFJz6e9y0RuyjvWAWA1DcJo0SpCq1kHsUzWRS4423hQasgywP
iFKGqP5eHDSqUHF5V6Gdtg/a9M7vthhEob/2IKSf82CQ8S1dmNtEJHAPgWamgm
VPPyM1LD7d2gyqD+fmduRdy/yHMP0QV6G/VRTV4gfQ80qU/U2JXWAQd6ok1+k5V
t0ur8buqo+49diyr8WPHA4CwpSwriwICLD2dq38JicdficFFXaqYdEMbL6S4wga
Sv+QgcBvu7u5yV/hfrcvztrKwDwz21k0wmx04xprvps/cUanQoog3q7ce5xkaX7
99Ewd0X0ma/H++UG7fj0qjJW5n2FpeVEn+zcHf8of/XqiluaP1W5T/iB16Esi
MetBbjk0NNxywP3SAPwqMMg/vNa+mJj0NBw+
-----END PGP PUBLIC KEY BLOCK-----
ubuntu@ip-172-31-59-109:~$ sudo sh -c 'echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] https://pkg.jenkins.io/debian-stable binary/
list.d/jenkins.list'
deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] https://pkg.jenkins.io/debian-stable binary/
ubuntu@ip-172-31-59-109:~$ sudo apt update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu jammy-security InRelease
Ign:5 https://pkg.jenkins.io/debian-stable binary/ InRelease
Get:6 https://pkg.jenkins.io/debian-stable binary/ Release [2044 B]
Get:7 https://pkg.jenkins.io/debian-stable binary/ Release.gpg [833 B]
Get:8 https://pkg.jenkins.io/debian-stable binary/ Packages [26.4 kB]
Fetched 29.3 kB in 1s (53.4 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
19 packages can be upgraded. Run 'apt list --upgradable' to see them.
ubuntu@ip-172-31-59-109:~$
```

These commands will download the Jenkins package key and add the Jenkins package repository to the list of package sources on the EC2 instance. This prepares the instance for installing Jenkins.

To start Jenkins and expose port 8080 on the EC2 instance, follow these steps:

Start Jenkins service by running the command: `sudo systemctl start jenkins.service`

Allow incoming traffic on port 8080 using the Uncomplicated Firewall (UFW) by running:

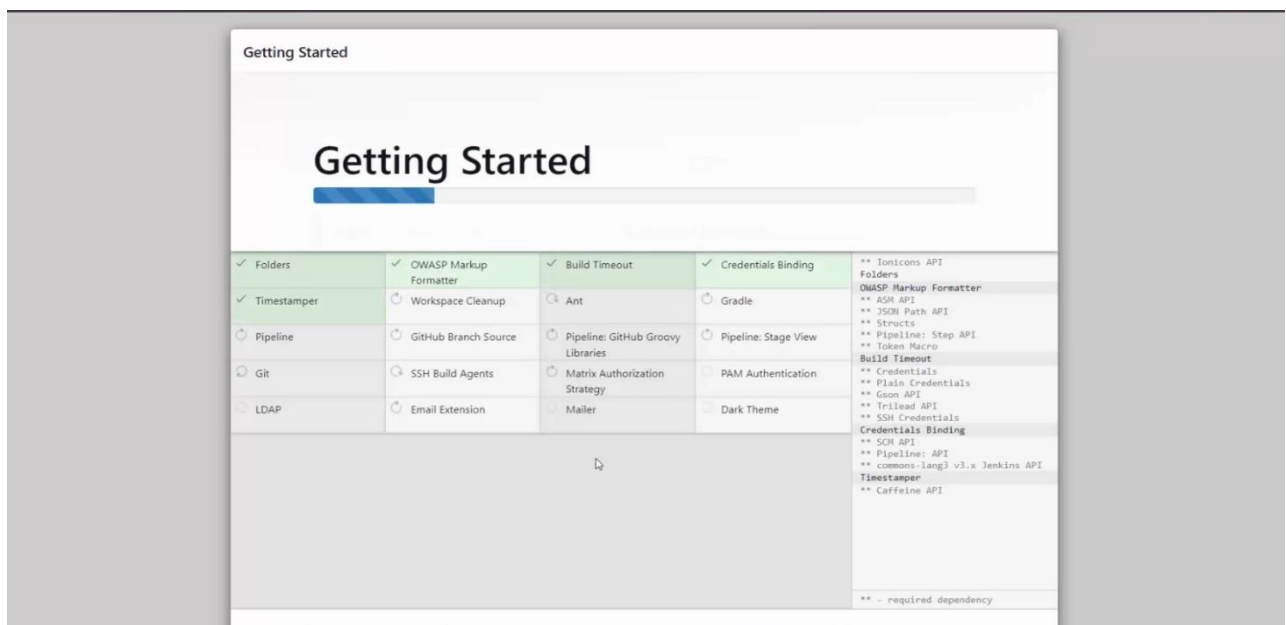
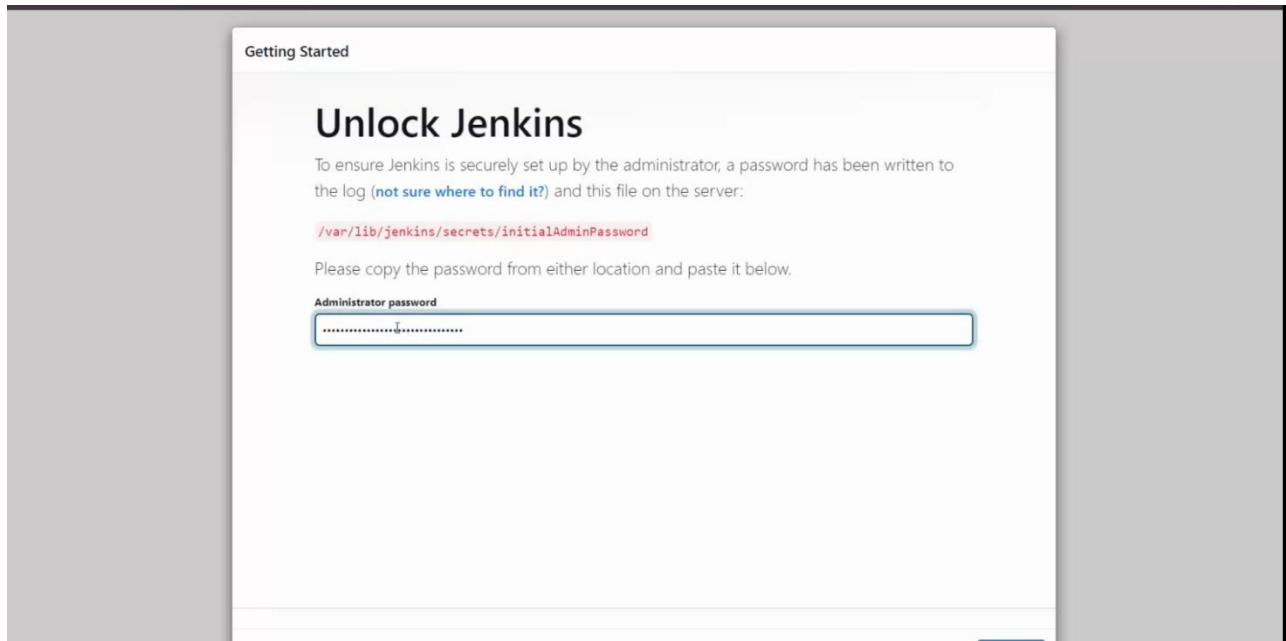
```
sudo ufw allow 8080
```

Retrieve the initial admin password generated by Jenkins using the following command:

```
sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

This command will display the generated password. Make sure to store it securely as it is required during the Jenkins setup process.

These steps will start Jenkins, open port 8080 for incoming traffic, and provide you with the initial admin password needed to set up Jenkins.



To install snapd and then kubectl, follow these steps:

Install snapd by running the command: `sudo apt install snapd`

Once snapd is installed, use snap to install kubectl by running: `sudo snap install kubectl --classic`

To access Jenkins on the AWS EC2 instance, follow these steps:

Open a web browser and enter the public IPv4 address of the Jenkins instance in the address bar. For example: `http://34.224.54.50`

Append the port 8080 to the URL, so it looks like this: `http://34.224.54.50:8080`

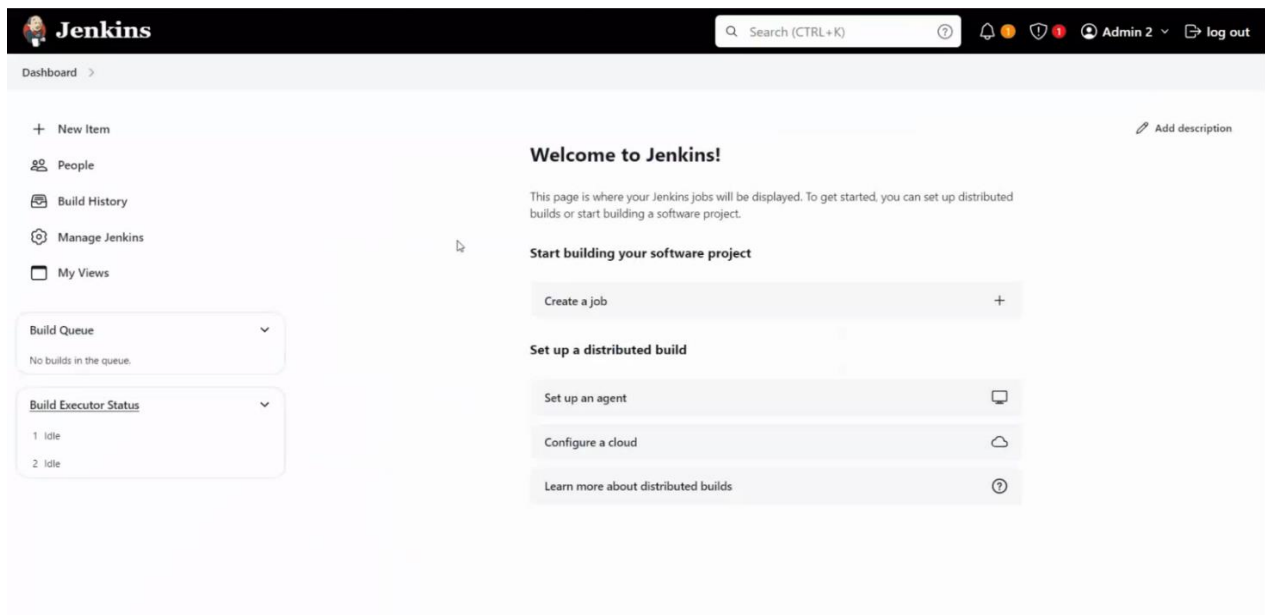
Press Enter to navigate to the Jenkins login page.

If prompted, enter the initial admin password obtained earlier when prompted to unlock Jenkins.

Following these steps will allow you to access Jenkins via its public IPv4 address on port 8080 and unlock it using the admin password.

Now install the suggested plugins.

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



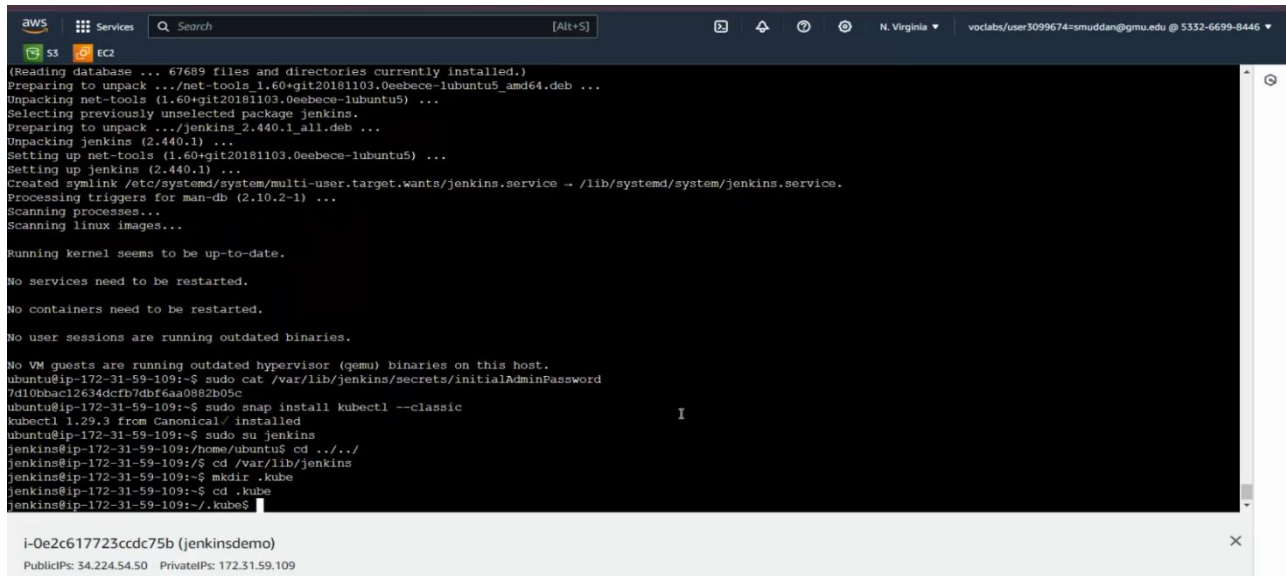
To create the Kubernetes config file for Jenkins, follow these steps:

Go back to the EC2 console.

Switch to the Jenkins user by running: `sudo su Jenkins`

Navigate to the root directory: `cd ../../`

Go to the Jenkins home directory: `cd /var/lib/Jenkins`



```
(Reading database ... 67689 files and directories currently installed.)
Preparing to unpack .../net-tools_1.60+git20181103.0eebece-lubuntu5_amd64.deb ...
Unpacking net-tools (1.60+git20181103.0eebece-lubuntu5) ...
Selecting previously unselected package jenkins.
Preparing to unpack .../jenkins_2.440.1_all.deb ...
Unpacking jenkins (2.440.1) ...
Setting up net-tools (1.60+git20181103.0eebece-lubuntu5) ...
Setting up jenkins (2.440.1) ...
Created symlink /etc/systemd/system/multi-user.target.wants/jenkins.service -> /lib/systemd/system/jenkins.service.
Processing triggers for man-db (2.10.2-1) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-59-109:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
7d10bbac12634dcfb7dbf6aa0882b05c
ubuntu@ip-172-31-59-109:~$ sudo snap install kubectl --classic
kubectl 1.29.3 from Canonical/ installed
ubuntu@ip-172-31-59-109:~$ sudo su jenkins
jenkins@ip-172-31-59-109:/home/ubuntu$ cd ../../
jenkins@ip-172-31-59-109:/var/lib/jenkins$
jenkins@ip-172-31-59-109:~$ mkdir .kube
jenkins@ip-172-31-59-109:~$ cd .kube
jenkins@ip-172-31-59-109:~/.kube$
```

i-Oe2c617723ccdc75b (jenkinsdemo)
PublicIPs: 34.224.54.50 PrivateIPs: 172.31.59.109

Create a directory named ".kube" and enter it:

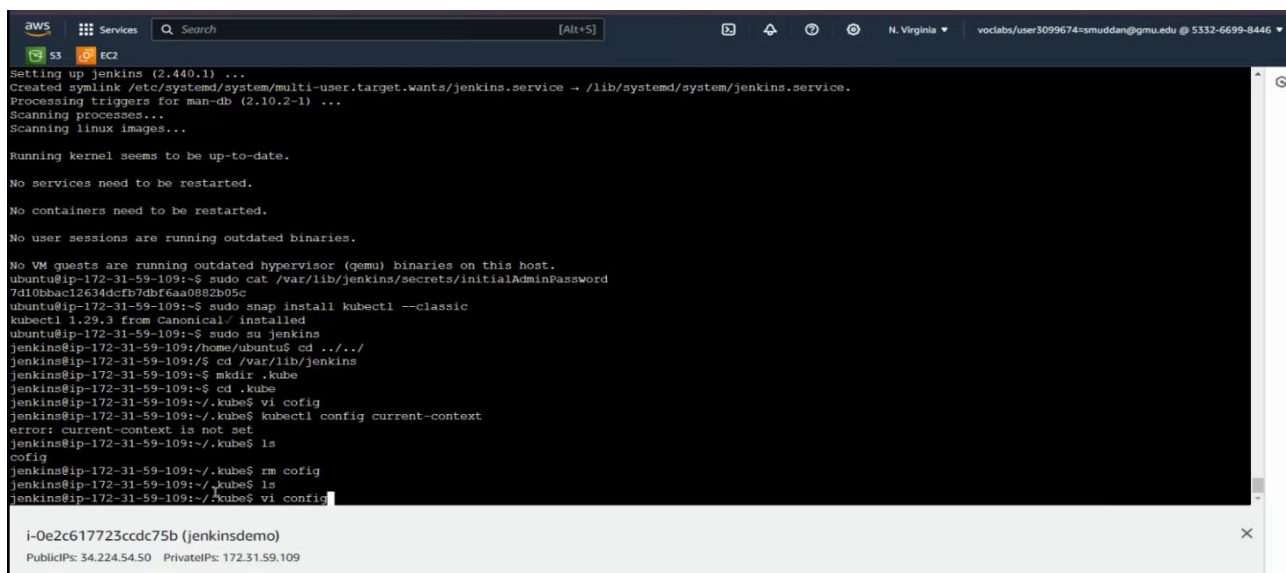
```
mkdir .kube
```

```
cd .kube
```

Create and open the config file using the vi editor: vi config

Copy the contents from the "KubeConfig" file downloaded earlier and paste it into the vi editor.

Save the file and exit vi by pressing Esc, then typing ":wq" and pressing Enter.



```
Setting up jenkins (2.440.1) ...
Created symlink /etc/systemd/system/multi-user.target.wants/jenkins.service -> /lib/systemd/system/jenkins.service.
Processing triggers for man-db (2.10.2-1) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-59-109:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
7d10bbac12634dcfb7dbf6aa0882b05c
ubuntu@ip-172-31-59-109:~$ sudo snap install kubectl --classic
kubectl 1.29.3 from Canonical/ installed
ubuntu@ip-172-31-59-109:~$ sudo su jenkins
jenkins@ip-172-31-59-109:/home/ubuntu$ cd ../../
jenkins@ip-172-31-59-109:/var/lib/jenkins$
jenkins@ip-172-31-59-109:~$ mkdir .kube
jenkins@ip-172-31-59-109:~$ cd .kube
jenkins@ip-172-31-59-109:~/.kube$ vi config
jenkins@ip-172-31-59-109:~/.kube$ kubectl config current-context
error: current-context is not set
jenkins@ip-172-31-59-109:~/.kube$ ls
config
jenkins@ip-172-31-59-109:~/.kube$ rm config
jenkins@ip-172-31-59-109:~/.kube$ ls
jenkins@ip-172-31-59-109:~/.kube$ vi config
```

i-Oe2c617723ccdc75b (jenkinsdemo)
PublicIPs: 34.224.54.50 PrivateIPs: 172.31.59.109

These steps will create the Kubernetes config file named "config" in the ".kube" directory within the Jenkins user's home directory, allowing Jenkins to interact with the Kubernetes cluster.

To verify the Kubernetes configuration and install Docker, follow these steps:

Run the following command to check the current Kubernetes context:

```
kubectl config current-context
```

This command should display the cluster name as expected.

Exit the Jenkins mode by running: `exit`

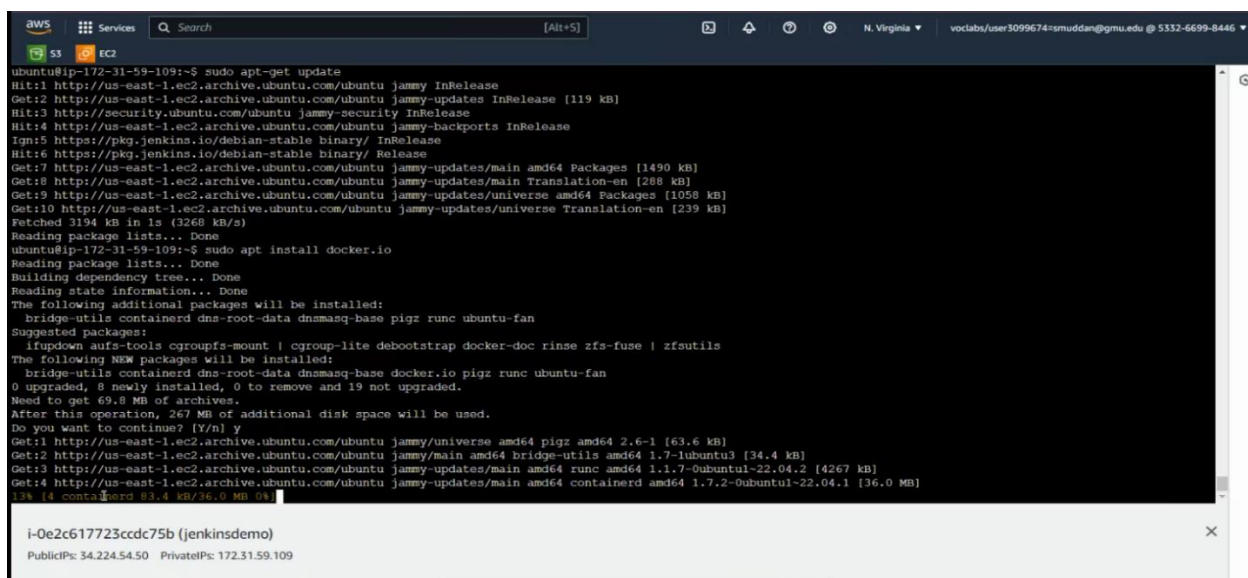
Update the package index and upgrade the packages by running:

```
sudo apt-get update
```

```
sudo apt update
```

Install Docker by running: `sudo apt install docker.io`

When prompted, type "Y" and press Enter to confirm the installation.



```
aws
Services
Search
[Alt+S]
N. Virginia
voclabs/user3099674-smuddan@gmu.edu @ 5332-6699-8446
S3 EC2
ubuntu@ip-172-31-59-109:~$ 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]
Hit:3 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease
Ign:5 https://pkg.jenkins.io/debian-stable binary/ InRelease
Hit:6 https://pkg.jenkins.io/debian-stable binary/ Release
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [1490 kB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main Translation-en [288 kB]
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [1058 kB]
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe Translation-en [239 kB]
Fetched 3194 kB in 1s (3268 kB/s)
Reading package lists... Done
ubuntu@ip-172-31-59-109:~$ 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 cgroups-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 19 not upgraded.
Need to get 69.8 MB of archives.
After this operation, 267 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:11 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 pigz amd64 2.6-1 [63.6 kB]
Get:12 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 bridge-utils amd64 1.7-lubuntu3 [34.4 kB]
Get:13 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 runc amd64 1.1.7-0ubuntu1-22.04.2 [4267 kB]
Get:14 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 containerd amd64 1.7.2-0ubuntu1-22.04.1 [36.0 MB]
13% [4 containerd 93.4 kB/36.0 MB 0s]
i-Oe2c617723ccdc75b (jenkinsdemo)
PublicIP: 34.224.54.50 PrivateIP: 172.31.59.109
```

After installing Docker, change the file permissions for the Docker socket by running: `sudo chmod 777 /var/run/docker.sock`

```
aws Services Search [Alt+S] N. Virginia voclabs/user3099674-smuddan@sgmu.edu @ 5332-6699-8446
S3 EC2
Preparing to unpack .../7-ubuntu-fan-0.12.16_all.deb ...
Unpacking ubuntu-fan (0.12.16) ...
Setting up dnsmasq-base (2.90-0ubuntu0.22.04.1) ...
Setting up runc (1.1.7-0ubuntu1-22.04.2) ...
Setting up dns-root-data (2023112702-ubuntu0.22.04.1) ...
Setting up bridge-utils (1.7-1ubuntu3) ...
Setting up pigz (2.6-1) ...
Setting up containerd (1.7.2-0ubuntu1-22.04.1) ...
Created symlink /etc/systemd/system/multi-user.target.wants/containerd.service → /lib/systemd/system/containerd.service.
Setting up ubuntu-fan (0.12.16) ...
Created symlink /etc/systemd/system/multi-user.target.wants/ubuntu-fan.service → /lib/systemd/system/ubuntu-fan.service.
Setting up docker.io (24.0.5-0ubuntu1-22.04.1) ...
Adding group 'docker' (GID 123) ...
Done.
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /lib/systemd/system/docker.service.
Created symlink /etc/systemd/system/sockets.target.wants/docker.socket → /lib/systemd/system/docker.socket.
Processing triggers for dbus (1.12.20-2ubuntu4.1) ...
Processing triggers for man-db (2.10.2-1) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-59-109:~$ sudo chmod 777 /var/run/docker.sock
ubuntu@ip-172-31-59-109:~$
```

i-0e2c617723ccdc75b (jenkinsdemo)
PublicIPs: 34.224.54.50 PrivateIPs: 172.31.59.109

These steps will verify the Kubernetes configuration, exit the Jenkins mode, and then install Docker while ensuring proper permissions for the Docker socket.

Setting up Github Repo:

To set up the GitHub repository for the pipeline, follow these steps:

Go to <https://github.com/> and create an account or log in if you already have one.

Once logged in, click on the "+" icon in the top-right corner and select "New repository".

Name the repository "Swe645Homework2".

Choose whether the repository will be public or private (this depends on your preference and needs).

Click on the "Create repository" button.

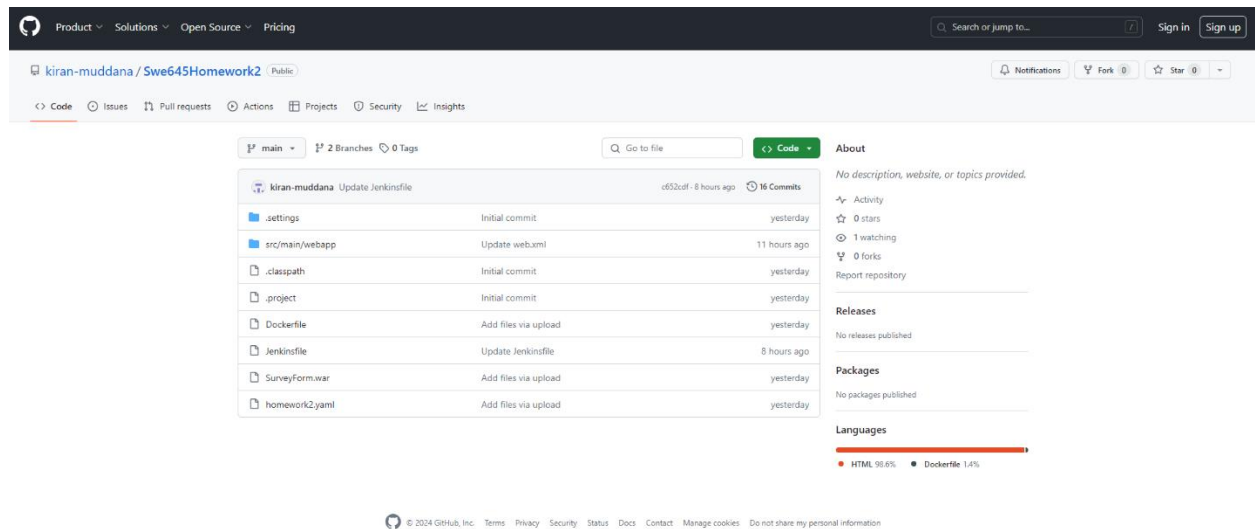
Once the repository is created, navigate to it and upload your source files, Dockerfile, and the WAR file by clicking on the "Add file" button and selecting "Upload files".

Drag and drop the files or use the file selector to add them to the repository.

Once the files are added, scroll down and click on the "Commit changes" button to commit them to the repository.

Now, you can use this GitHub repository URL for your pipeline:

URL: <https://github.com/kiran-muddana/Swe645Homework2>



To create credentials for DockerHub in Jenkins, follow these steps:

Go to the Jenkins dashboard and click on "Manage Jenkins".

Scroll down and select the "Manage Credentials" option.

Click on "Global credentials (unrestricted)".

Click on "Add credentials".

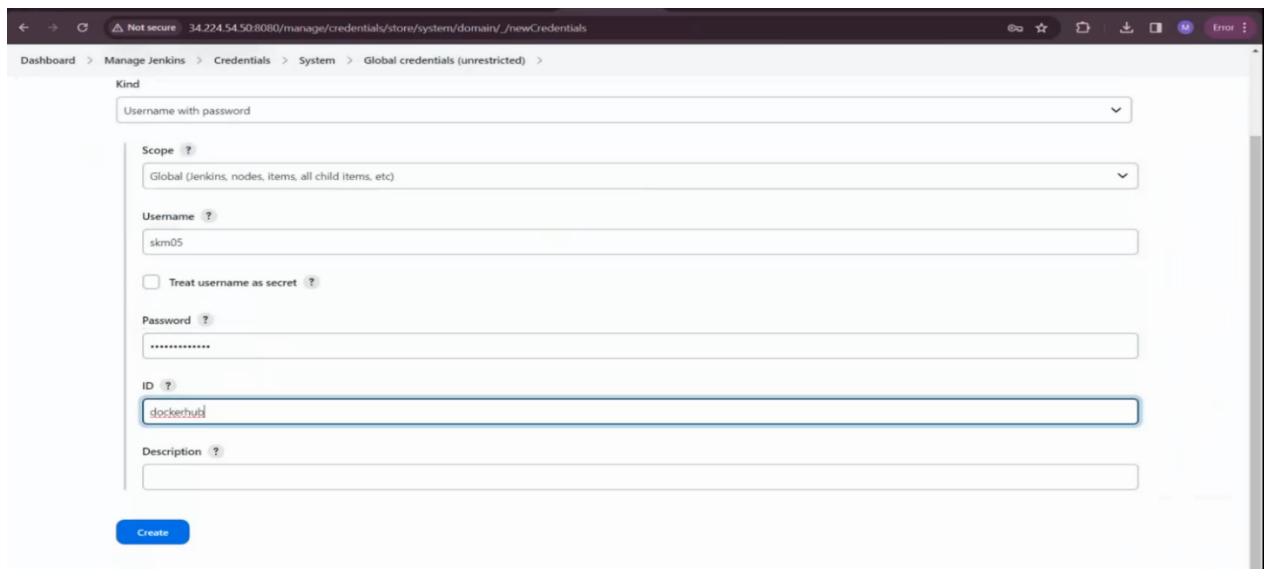
In the Kind dropdown menu, select "Username with password".

Enter your DockerHub username and password in the respective fields.

Set the ID field as "dockerhub".

Click on the "OK" or "Create" button to save the credentials.

These steps will add DockerHub credentials to Jenkins, allowing it to access DockerHub repositories during pipeline execution.



The screenshot shows the Jenkins 'New Credentials' form. The breadcrumb trail is 'Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted)'. The form fields are: 'Kind' (dropdown set to 'Username with password'), 'Scope' (dropdown set to 'Global (Jenkins, nodes, items, all child items, etc)'), 'Username' (text field with 'skm05'), 'Treat username as secret' (checkbox, unchecked), 'Password' (password field with masked characters), 'ID' (text field with 'dockerhub'), and 'Description' (empty text field). A blue 'Create' button is at the bottom left.

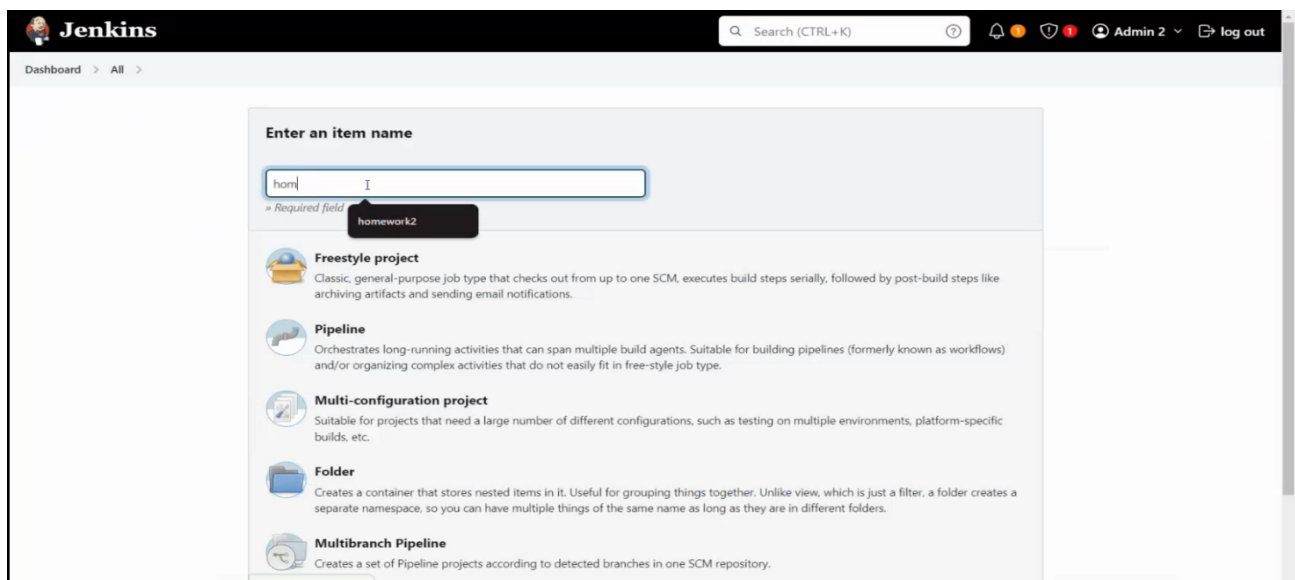
Repeat the above step to save GitHub credentials as well, give id as “github”.

Now go to dashboard and click on “New Item”.

Give name as “homework2demo” and click on “Pipeline”.

Now on the next page, check the “Github Project” checkbox, and in the project URL paste the URL from

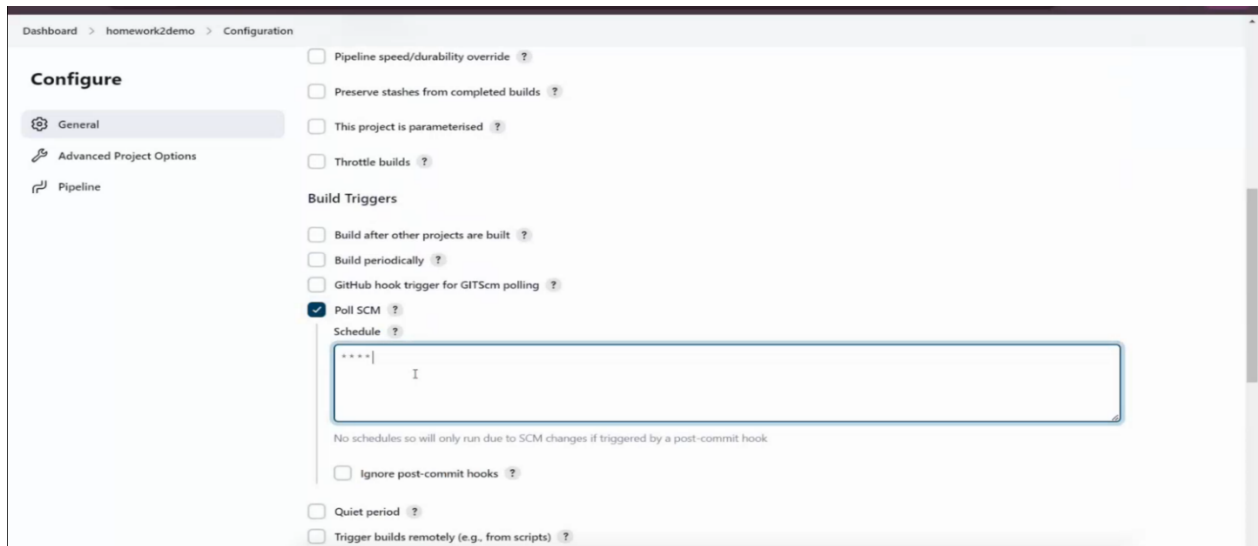
Github repository: <https://github.com/kiran-muddana/Swe645Homework2>



The screenshot shows the Jenkins 'New Item' page. The breadcrumb trail is 'Dashboard > All'. The 'Enter an item name' section has a text field with 'hom' and a dropdown menu showing 'homework2'. Below this, there are several job type options: 'Freestyle project' (Classic, general-purpose job type), 'Pipeline' (Orchestrates long-running activities), 'Multi-configuration project' (Suitable for projects that need a large number of different configurations), 'Folder' (Creates a container that stores nested items), and 'Multibranch Pipeline' (Creates a set of Pipeline projects according to detected branches).

Scroll down and check the “Poll SCM” checkbox.

Give the Schedule as : “* * * * *”.



To configure the Jenkins pipeline to fetch the Jenkinsfile from the GitHub repository, follow these steps:

Scroll down to the "Pipeline" section in Jenkins.

Select the "Pipeline script from SCM" option.

In the SCM dropdown menu, select "Git".

Enter the URL of your GitHub repository in the "Repository URL" field:

<https://github.com/kiran-muddana/Swe645Homework2.git>

Choose the appropriate credentials from the "Credentials" dropdown menu.

In the "Branches to build" or "Branch specifier" field, specify the branch where your Jenkinsfile is located. If it's the main branch, enter: */main

Save the configuration.

Go to github repository and then create a new file : “Jenkinsfile” and then commit changes.

Click on Save.

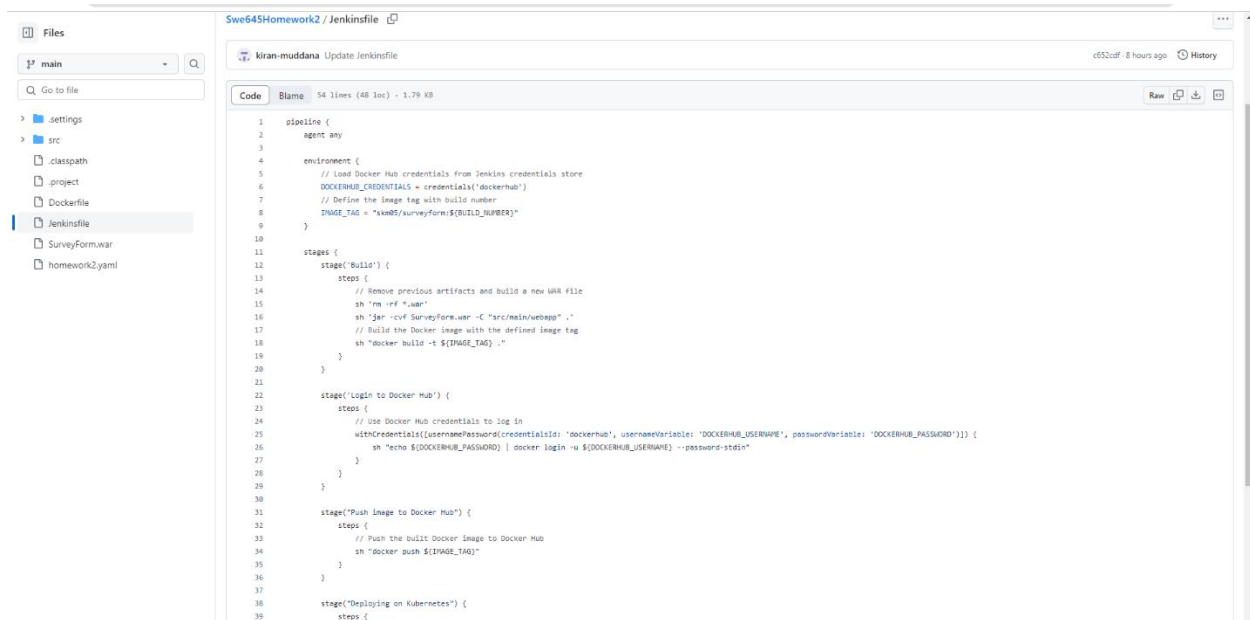
Now a build will automatically start on this pipeline.

Now go to the Github repository -> “Jenkinsfile” and click on edit.

Enter the commands given in the screenshot below and 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 commands on how to generate a new war file, create a docker image, login to docker and push image, deploy image on Kubernetes cluster.



```
1 pipeline {
2   agent any
3
4   environment {
5     // Load Docker Hub credentials from Jenkins credentials store
6     DOCKERHUB_CREDENTIALS = credentials('dockerhub')
7     // Define the image tag with build number
8     IMAGE_TAG = "sws64/surveyform:${BUILD_NUMBER}"
9   }
10
11   stages {
12     stage('Build') {
13       steps {
14         // Remove previous artifacts and build a new WAR file
15         sh 'rm -rf *.war'
16         sh 'jar -cvf SurveyForm.war -C "src/main/webapp" .'"
17         // Build the Docker image with the defined image tag
18         sh "docker build -t ${IMAGE_TAG} ."
19       }
20     }
21
22     stage('Login to Docker Hub') {
23       steps {
24         // Use Docker Hub credentials to log in
25         withCredentials([usernamePassword(credentialsId: 'dockerhub', usernameVariable: 'DOCKERHUB_USERNAME', passwordVariable: 'DOCKERHUB_PASSWORD')]) {
26           sh "echo ${DOCKERHUB_PASSWORD} | docker login -u ${DOCKERHUB_USERNAME} --password-stdin"
27         }
28       }
29     }
30
31     stage('Push Image to Docker Hub') {
32       steps {
33         // Push the built Docker image to Docker Hub
34         sh "docker push ${IMAGE_TAG}"
35       }
36     }
37
38     stage('Deploying on Kubernetes') {
39       steps {
```

To address any errors that may occur during the build process by adjusting the Jenkinsfile, follow these steps:

Navigate to your GitHub repository.

Locate the Jenkinsfile and click on "Edit".

Review the Jenkinsfile for any paths, commands, or configurations that may be causing errors.

Make necessary corrections or adjustments to resolve the errors. Here are some common adjustments you might need to make:

Check the paths to ensure they are correctly pointing to the necessary files or directories.

Verify that commands are written correctly and include any required arguments or options.

Double-check configurations to ensure they match the setup of your environment and project.

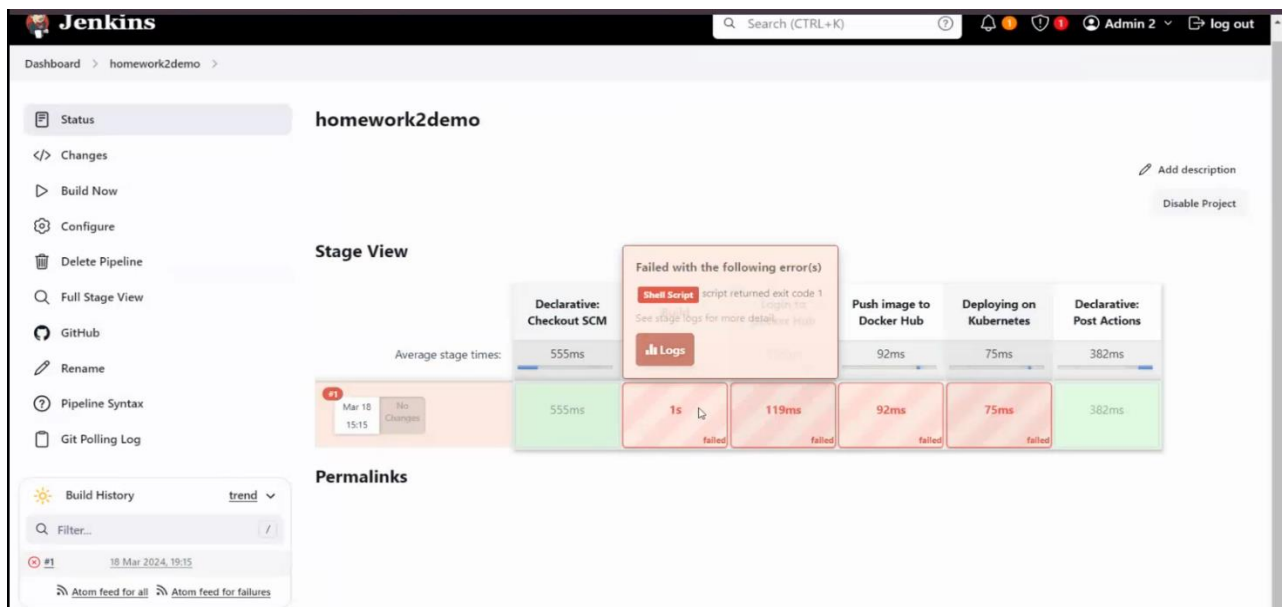
Save the changes and commit them to the repository.

Trigger a new build in Jenkins to apply the changes made to the Jenkinsfile.

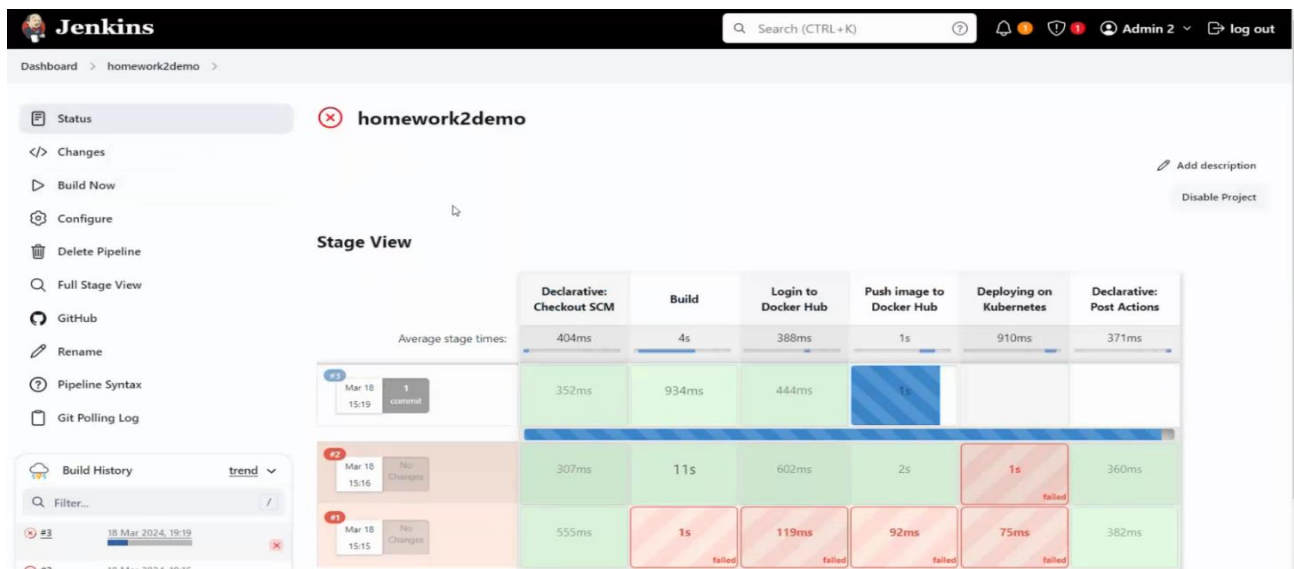
Monitor the build process for any errors or issues and adjust the Jenkinsfile further if necessary.

Repeat this process as needed until the build process completes successfully without errors.

By iteratively adjusting the Jenkinsfile based on error feedback and testing, you can gradually refine the build process and ensure its smooth execution.



After the build is successful, if we reload the webpage, we will be able to see the changes being reflected.



Student Survey Form

First Name:

Last Name:

Street Address:

City:

State:

ZIP:

Telephone Number:

E-mail:

Date of Survey:

Links:

Github URL: <https://github.com/kiran-muddana/Swe645Homework2>

Docker URL: <https://hub.docker.com/r/skm05/studentsurvey>

Application URL: <http://ec2-54-243-111-113.compute-1.amazonaws.com:30463/SurveyForm/>

References:

1. <https://www.digitalocean.com/community/tutorials/how-to-install-jenkins-on-ubuntu-20-04>
2. Lecture 6 class notes.
3. Lecture 7 class notes.
4. https://docs.docker.com/get-started/02_our_app/
5. <https://docs.rke2.io/>

Contributions:

Kiran: Developed the source file, worked on creating docker image and the Github repository, worked on solving issues and errors in setting up jenkins and CI/CD pipeline. (making of video)

Venu: Setting up CI/CID pipeline and documentation. (edits/corrections made for the video)

Yashasree: Solving errors and issues, helped in setting up Jenkins and CI/CD pipeline