SWE 645 Homework / Assignment 2

Author of document: Qasim Shahid
Date Due: 3/10/2025

GitHub Repository: https://github.com/qasimshahid/qshahid-swe645-asst2/
Note: This repo is currently public but will be made private after this assignment is graded.

Website link (this will be available via Elastic IP, but I'm not sure if you will be able to access it since AWS Learner Lab only allows me to have it up for 4 hours):
http://18.210.159.87:30001

In this homework assignment, we are tasked to build a CI/CD pipeline using a very simple HTML page. We containerize a simple survey HTML page using Docker and then deploy it to a Kubernetes cluster, running at least 3 replicas. We use Rancher as a Kubernetes management platform to manage our cluster.

The Jenkins pipeline is triggered upon a push to the main branch of the GitHub repository. The pipeline does the following:

1. Get the most recent source code by cloning the repository

2. Build the docker image defined in the Dockerfile using docker build

3. Push the image we just built to Docker Hub (this requires an account, by default, the source code assumes mine is being used with username qshahid)

4. Deploys the latest container image for our app the Kubernetes cluster, creating a Deployment with 3 replicas (basically deploying the container to 3 pods). The app is exposed externally via a NodePort service, with port 30001 mapped to the container's port 80, allowing external access at
http://<public-cluster-ip>:30001.

Please make sure you have a Docker Hub account, an AWS Learner Lab account, and a GitHub account. Please also clone/fork my GitHub repository to your own account so you can configure the webhook that allows us to automatically build and deploy whenever we push to the main branch. Please also make this GitHub repository public as that is what I used.

AWS EC2 Setup

This assignment makes extensive use of AWS. Please use the Learner Lab provided by the professor as this gives you a $50 budget to work with.

- 2 EC2 machines:
    - 1 t2.large machine, which will host Rancher
    - 1 t2.micro machine, which will host Jenkins
    - Please use Ubuntu on all of these machines and configure them to have 30 GB of storage.
    - From this point on, I will refer to them as the Rancher instance and the Jenkins instance.
- Elastic IPs
    - Please attach an elastic IP to both of these machines as it makes it easier to work with.

    -
- Please use the AWS Learner Lab and add this setting in "Advanced" when creating your instances:

    -
- Please proceed without a key, as we will be using Instance Connect.

    -

- Please create the following security group and apply it to both the Rancher instance and the Jenkins instance. Inbound rules image is shown first, and then outbound rules.

Details

Security group name	Security group ID	Description	VPC ID
launch-wizard-1	sg-0c8d28e561c50c1ae	launch-wizard-1 created 2025-03-08T21:34:11.068Z	vpc-0acbab5dcda107738

Owner	Inbound rules count	Outbound rules count	
728762828036	5 Permission entries	1 Permission entry	

Inbound rules | Outbound rules | Sharing - *new* | VPC associations - *new* | Tags

Inbound rules (5)

lame	Security group rule ID	IP version	Type	Protocol	Port range	Source	Description
	sgr-0549b3640fb9c0d21	IPv4	SSH	TCP	22	0.0.0.0/0	–
	sgr-097f573f8b6e9dcbb	IPv4	HTTP	TCP	80	0.0.0.0/0	–
	sgr-05f8c367cdc4a26be	IPv4	HTTPS	TCP	443	0.0.0.0/0	–
	sgr-0895faf11d28d11e1	IPv4	Custom TCP	TCP	30000 - 40000	0.0.0.0/0	–
	sgr-0668445d96ce81ff0	IPv4	Custom TCP	TCP	8080	0.0.0.0/0	–

Details

Security group name	Security group ID	Description	VPC ID
launch-wizard-1	sg-0c8d28e561c50c1ae	launch-wizard-1 created 2025-03-08T21:34:11.068Z	vpc-0acbab5dcda107738

Owner	Inbound rules count	Outbound rules count	
728762828036	5 Permission entries	1 Permission entry	

Inbound rules | **Outbound rules** | Sharing - *new* | VPC associations - *new* | Tags

Outbound rules (1)

	Name	Security group rule ID	IP version	Type	Protocol	Port range	Destination	Description
☐	–	sgr-05ec72ce919b34ed2	IPv4	All traffic	All	All	0.0.0.0/0	–

- The important thing here is to make sure that we can use ports 8080 (this is where we will access Jenkins) and ports 30000-40000 (specifically, we will be deploying to port 30001 with our NodePort service, so this needs to be accessible). Port 80 also needs to be accessible since this is how we will access Rancher. Overall, just copy my settings and that should be sufficient.
- After you do all this, you should be ready to proceed to the next step of setting up your EC2 machines. In particular, we need to install some dependencies.

For both machines, you need to execute the following commands. These will install some dependencies, such as Docker, kubectl, and generally just update the machines.

```
sudo su
sudo apt-get update -y
sudo apt-get upgrade -y

curl -fsSL https://get.docker.com -o get-docker.sh
sudo sh get-docker.sh

sudo snap install kubectl --classic
```

Setup Rancher Instance and Cluster

In order to get Rancher on your t2.large instance, do the following.
Login to the instance using instance connect:

Run all the commands mentioned at the end of the previous section.

```
sudo su
sudo apt-get update -y
sudo apt-get upgrade -y

curl -fsSL https://get.docker.com -o get-docker.sh
sudo sh get-docker.sh

sudo snap install kubectl –classic
```

Then, to use Rancher, run the following command:

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

After that, go to the following link to access Rancher:
http://{ec2-public-ip}:80
For example,
http://52.55.20.9:80

It will tell you the site is not secure, but just proceed past the warnings.
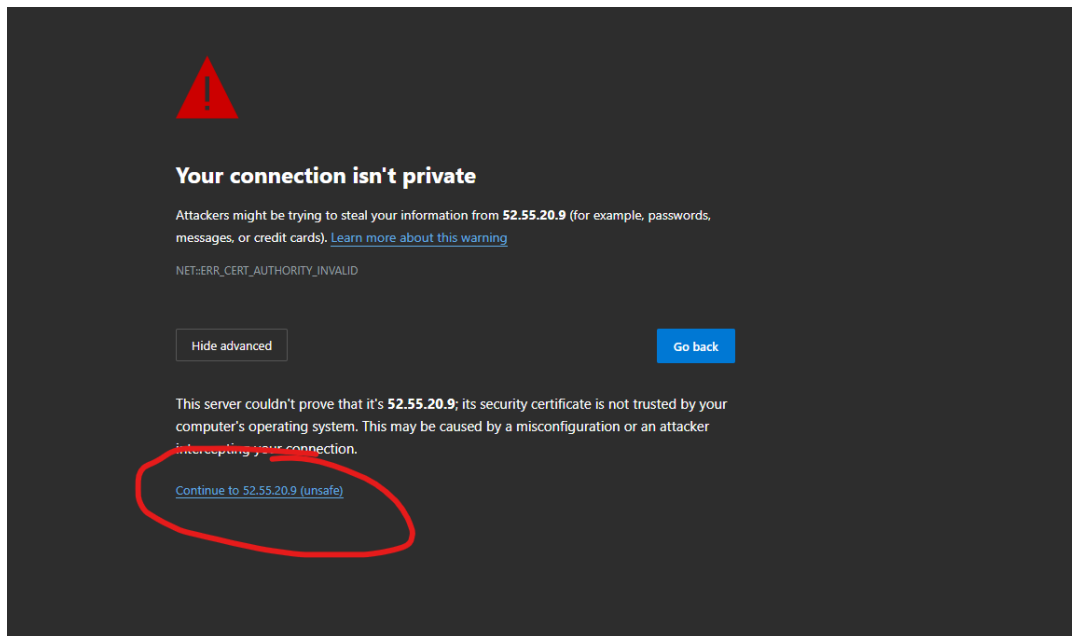
After that, you will be met by this UI and login page from Rancher.
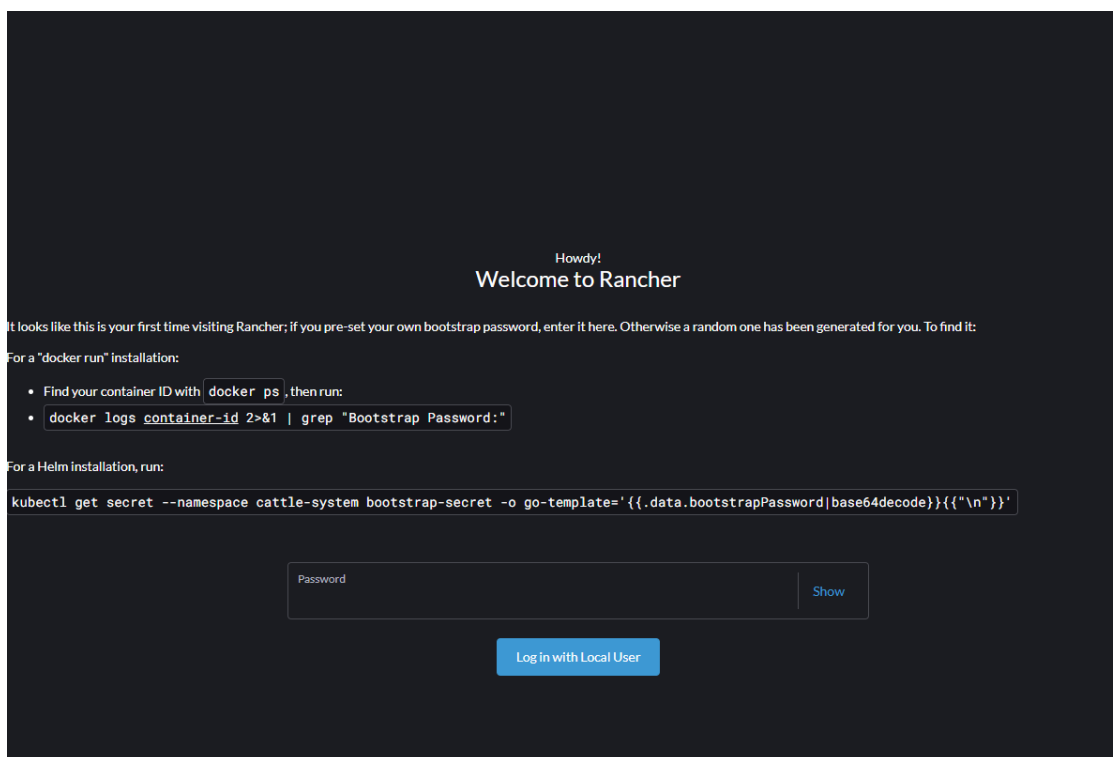
Follow the instructions and login to Rancher with the bootstrap password. Create your own password so that you remember the login. The username is always "admin". For me, I always use "temppass1234"
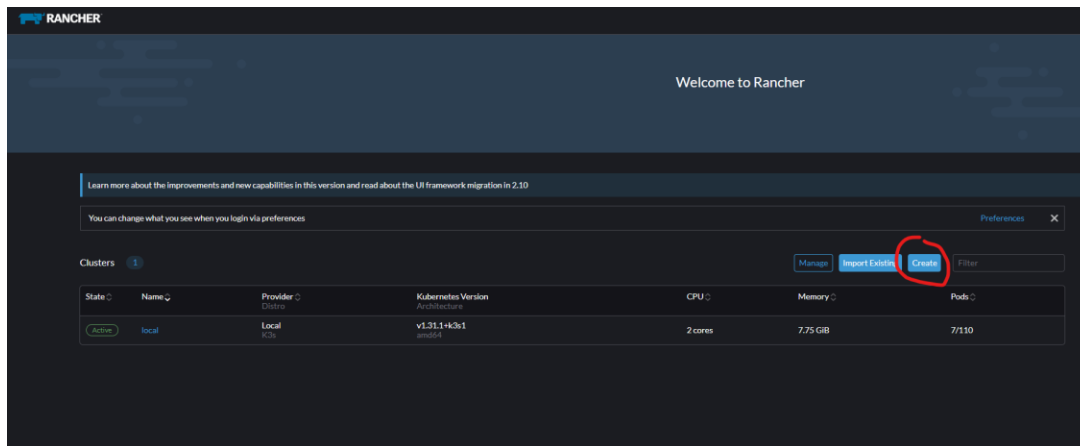
Let's create our new cluster. Use RKE2/3 to create a custom cluster.
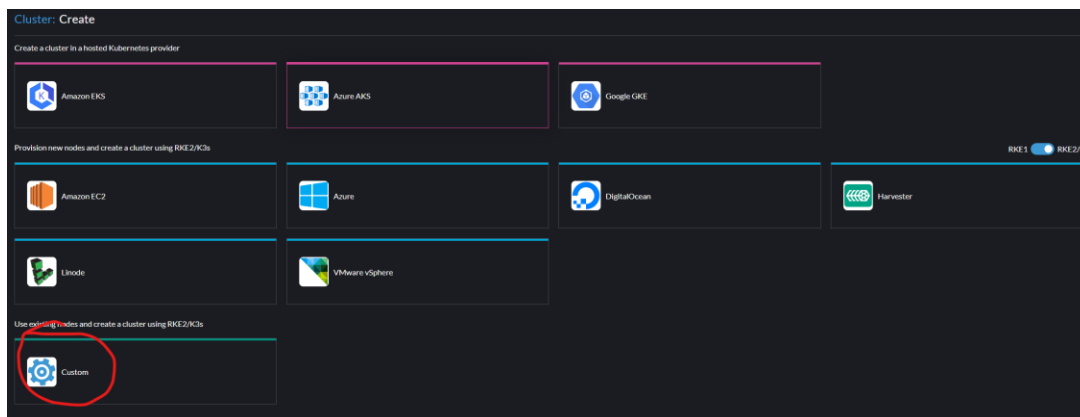
You don't need to do much. Just name it "cluster" and press create.
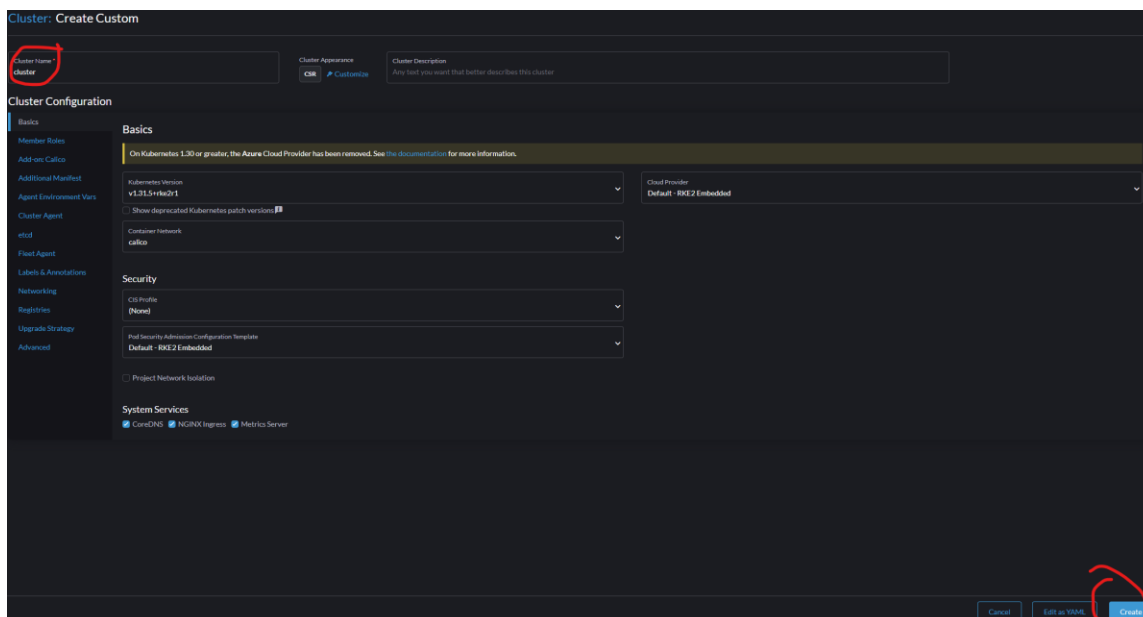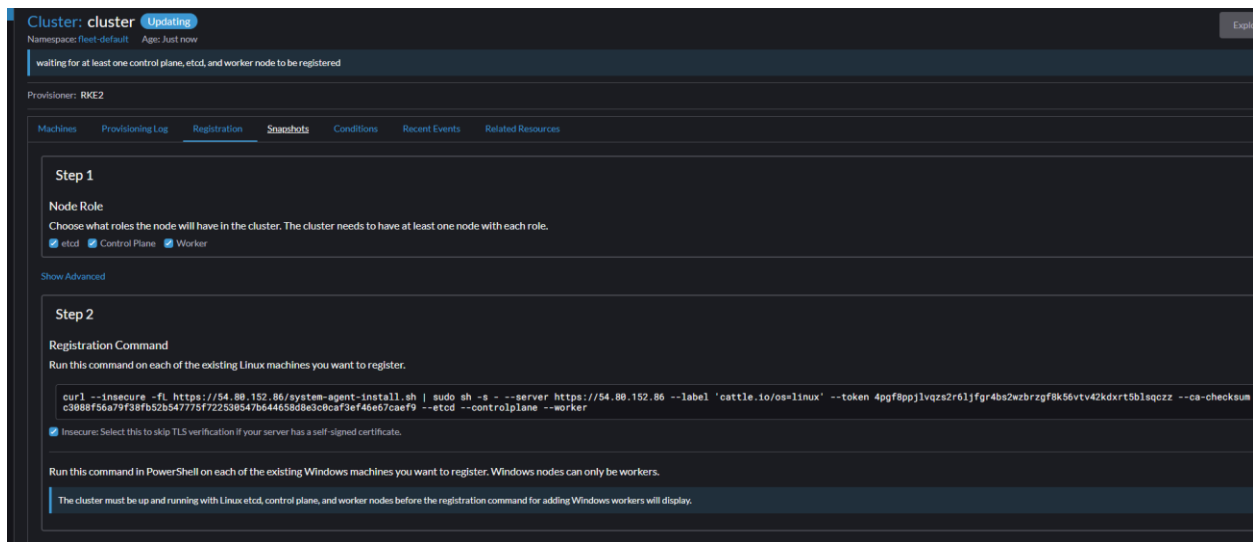
Then, you need to register a control plane, etcd, and worker node. You can do these on separate machines, but let's just do it here on the Rancher instance since this is a large machine and can handle it. Please select the "insecure" option of the registration command or this won't work. Copy it and run it on your Rancher instance.

Wait about 5 minutes at this point for everything to be set up properly. During this time, Rancher will show that the cluster is updating. Once 5 minutes has elapsed, you should now see that your new Kubernetes cluster is active.

We have our K8s cluster created, named "cluster", and it has 1 worker node, 1 control plane, and 1 etcd node, all on the same machine as where our cluster is.

The last step that you should do is download the kubeconfig file to your computer, as we will need it here shortly for the Jenkins instance:

This can be done from the "Cluster Management" tab on the left.
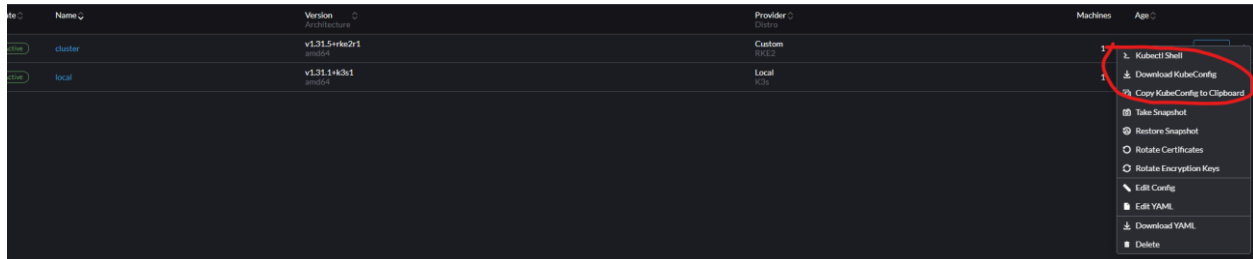
This allows us to connect to our cluster from different machines and instances (like our Jenkins instance that we will make after this) using the kubeconfig file.
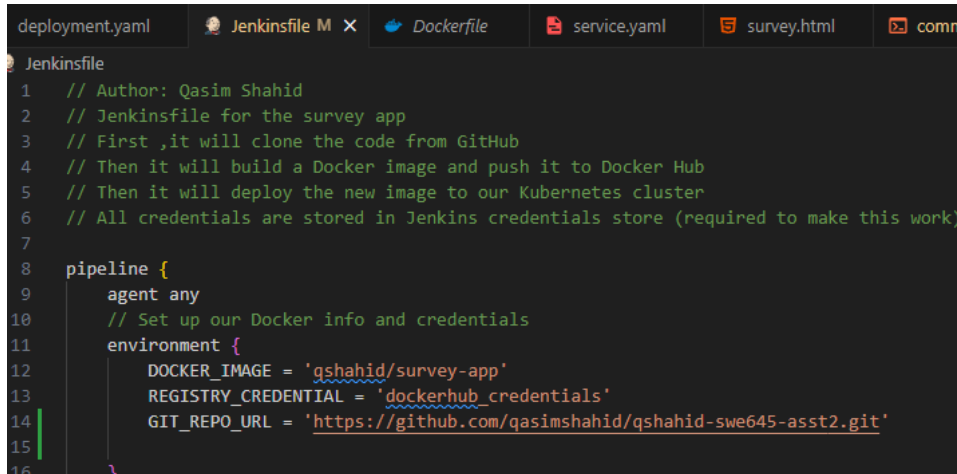
Congratulations, at this point, you are done with everything regarding the Rancher instance. Rancher has been setup, and you can access it using the elastic IP we assigned to the instance at http://{ec2-public-ip}:80

GitHub Repository Setup

GitHub Repository: https://github.com/qasimshahid/qshahid-swe645-asst2/
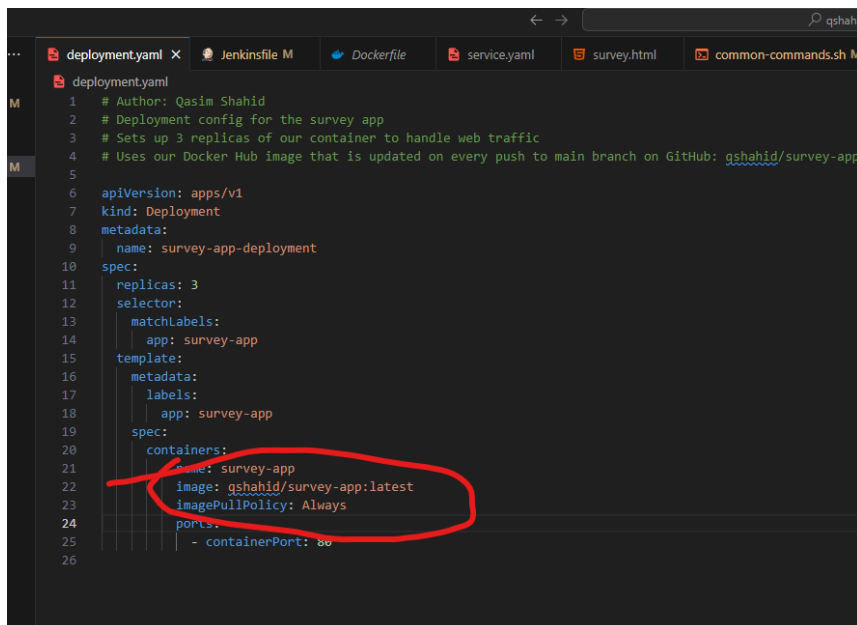
Please fork this repository or copy all the files from it and add them to a new repository that you created. This has pretty much everything you need, but you do need to make some changes.

First of all, please fix these settings to be your own. They should utilize the link of your GitHub repository and whatever your username is for Docker Hub and what your image should be called.

Don't worry about dockerhub_credentials value, we will do that together later on.

Change this value in the deployment.yaml file to be whatever your Docker Hub username is plus the name you give to the image.
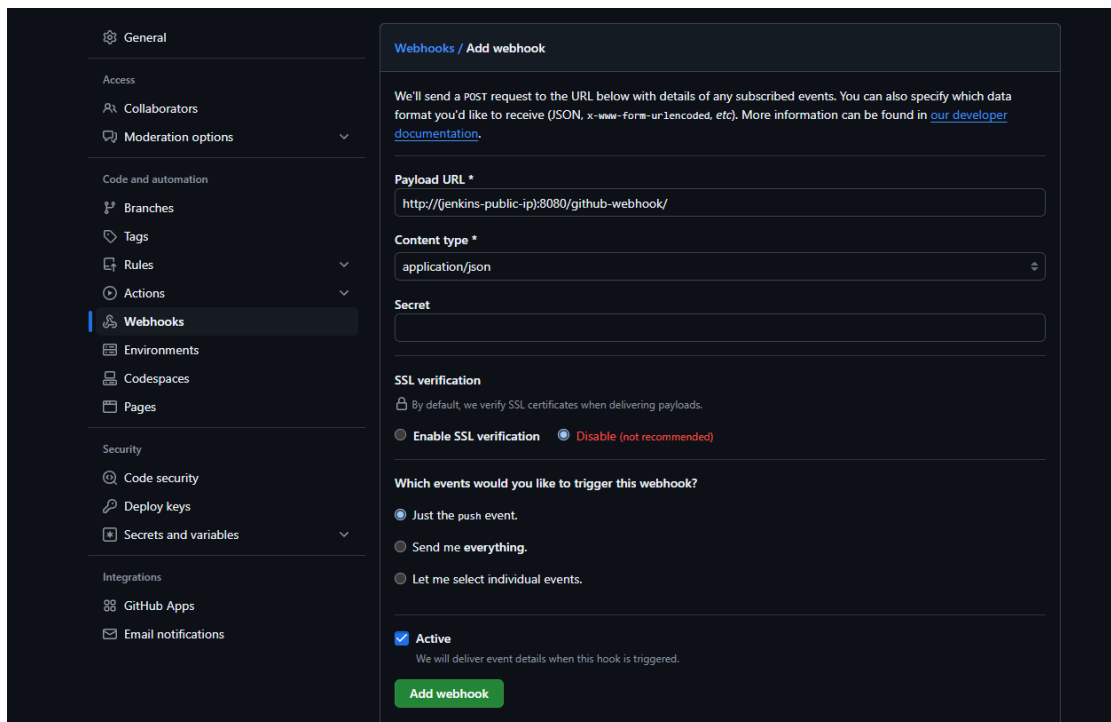
After you do all that, make sure the GitHub repository is public by going into the repository settings:
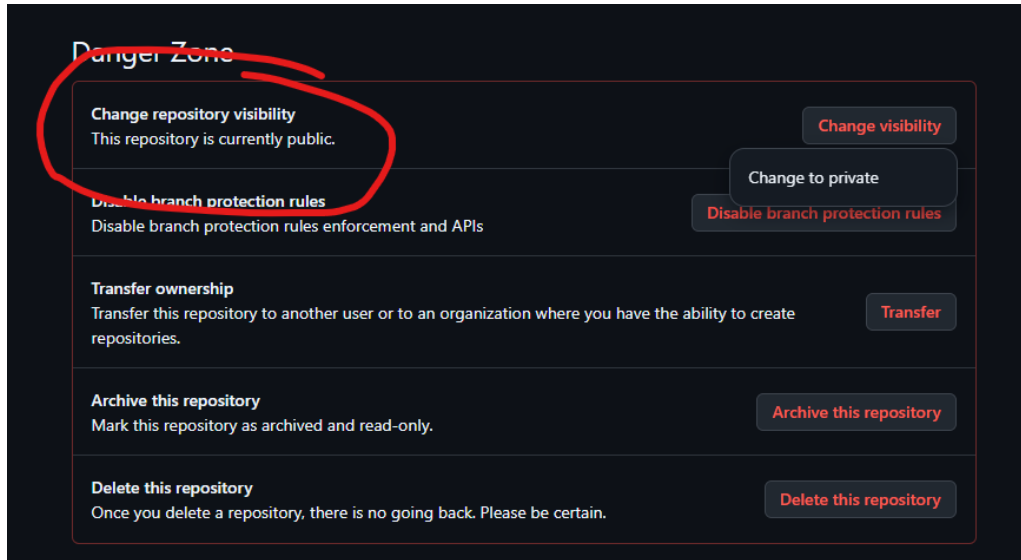
Please also at this point go into the webhook settings and create a new webhook that points to the following link:
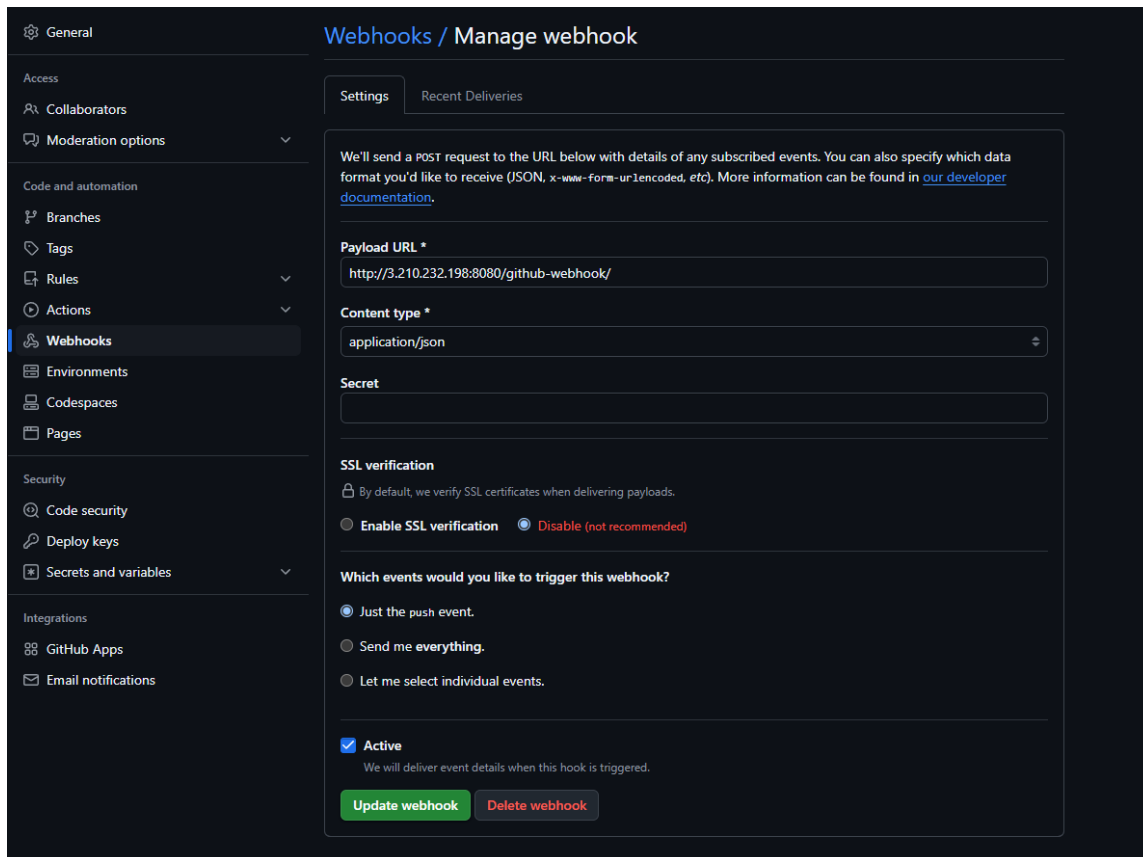
http://{jenkins-public-ip}:8080/github-webhook/

Replace {jenkins-public-ip} with the public elastic IP of your Jenkins instance and use the following settings:

Add this webhook.

Example:

This webhook will allow us to automatically build and deploy our application whenever we push to main branch on our GitHub repository. GitHub will recognize our push event and send a POST request to our Jenkins server, which will then kick off the pipeline.

Congratulations, at this point, you are done with setting up your GitHub repo. Push all the code changes and modifications you made and save all of these settings.

Jenkins Instance Setup

So, this is the other EC2 machine we created at the start, and this is where we will host our Jenkins server. This will allow us to automate our builds and make a pipeline that gets triggered whenever we push to the main branch of our GitHub repository.

This machine uses t2.micro as that is all we need. Remember to use 30 gb of storage and use a security group that has an inbound rule to accept all traffic for port 8080. This is all covered earlier, and you should have already completed this.

Please connect to the instance using Instance Connect:

After that, please execute these commands if you haven't already. These commands are required for both instances:

```
sudo su
sudo apt-get update -y
sudo apt-get upgrade -y

curl -fsSL https://get.docker.com -o get-docker.sh
sudo sh get-docker.sh

sudo snap install kubectl --classic
```

After that, please execute the following commands or follow the instructions from the official Jenkins site on how to download for Ubuntu (https://www.jenkins.io/doc/book/installing/linux/):

    sudo apt install fontconfig openjdk-17-jre -y

    sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \
      https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
    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
    sudo apt-get update
    sudo apt-get install jenkins -y

(This command is super important, make sure to execute this one or else builds might fail.)

    sudo usermod -a -G docker jenkins

    sudo systemctl enable jenkins
    sudo systemctl start jenkins
    sudo systemctl status jenkins

These commands will install Java and Jenkins. They will also make Docker accessible to Jenkins without needing to do sudo. The last command will also print a password, please store this as we will need it here in a second:

Now, the server is up. To go to it, use the elastic public IP we created at the start and go to this link:
http://{jenkins-public-ip}:8080
For example:
http://54.87.102.0:8080/
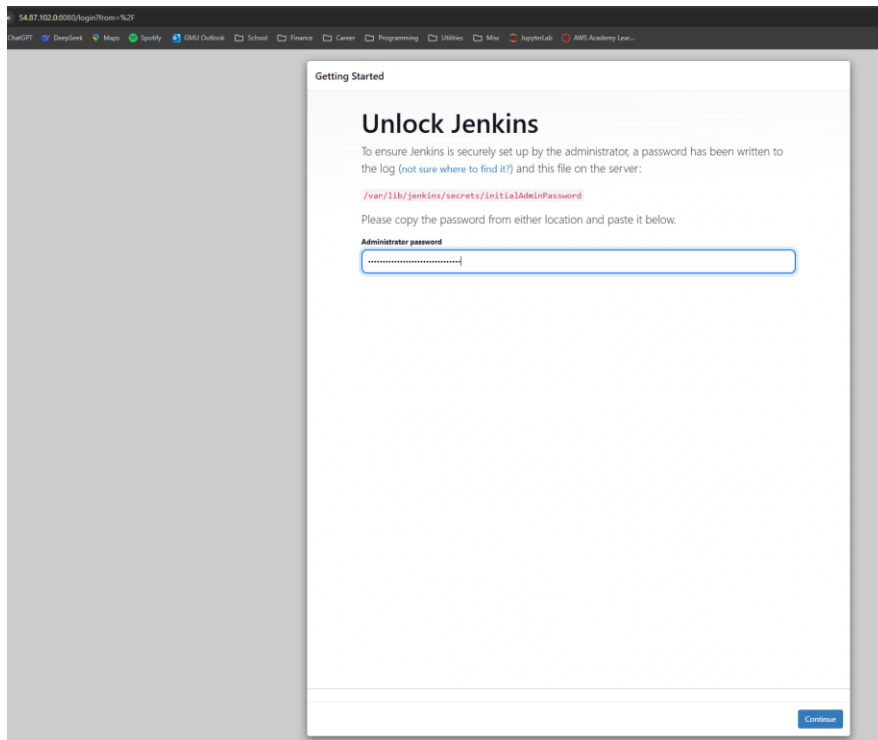
Go the link and Jenkins will ask you for the password from earlier. Paste it in.

Jenkins will now ask you what plugins you want. Select "Install suggested plugins." We will add some more later on.
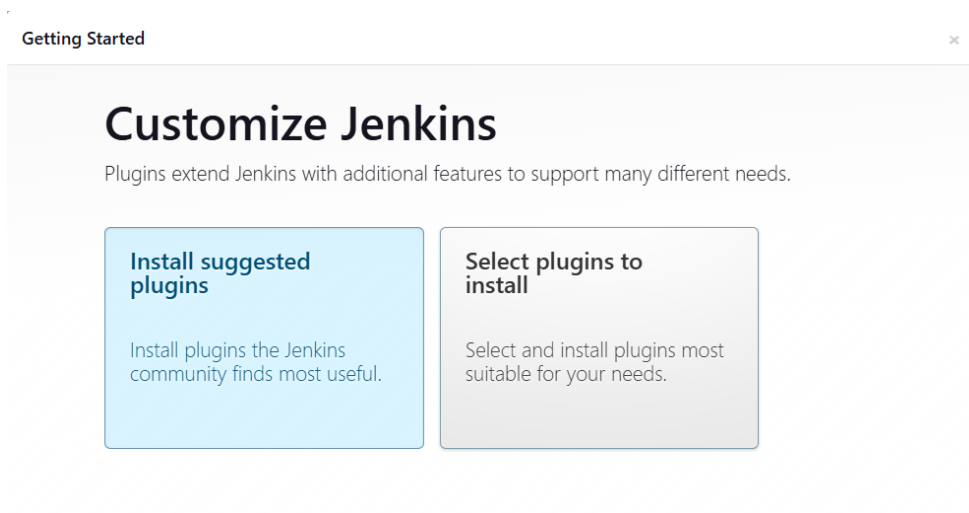
Then, create a username and password. I use "admin" and "temppass1234" with my GMU email. After this, keep clicking continue until the setup is complete.

We are now done with the initial setup. You should be on this screen:

Now, go to "Manage Jenkins" so that we can install some plugins.
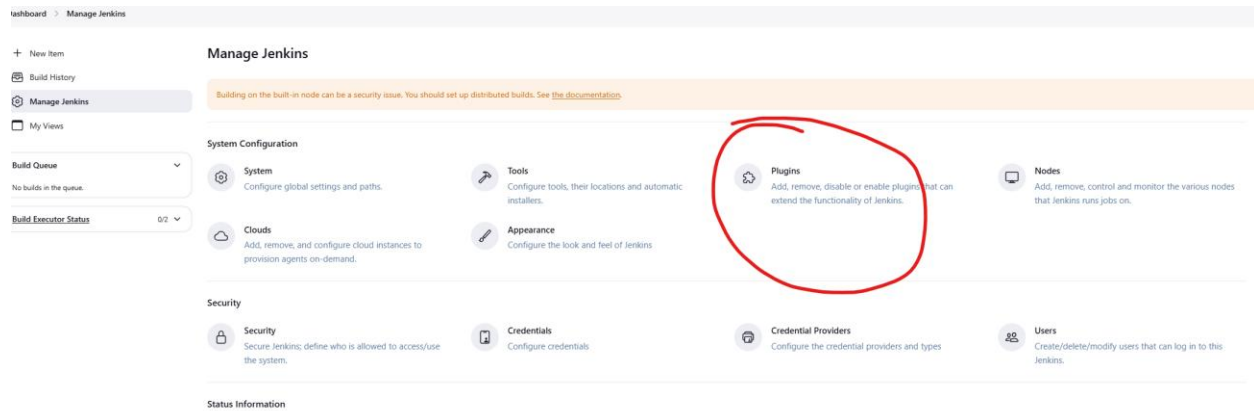
We need to install three plugins:
Docker plugin
Kubernetes CLI plugin
Docker Pipeline plugin
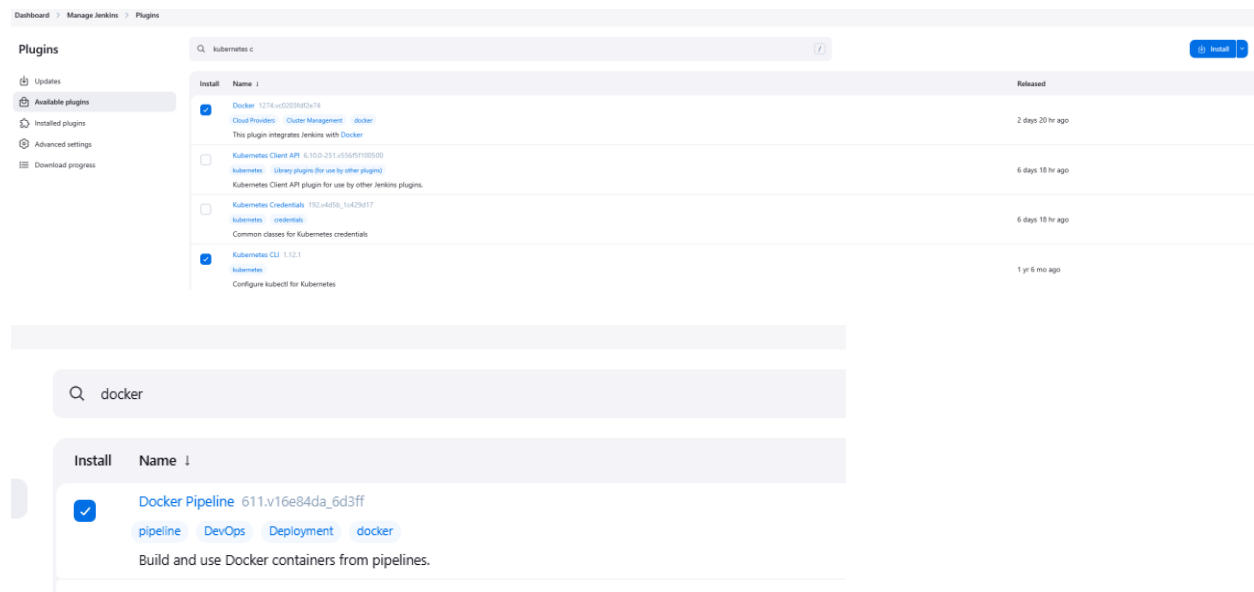
Install these and wait for them to finish downloading. We need these to build our image and deploy to our cluster.

After these are complete, go back to "Manage Jenkins" but this time, go to Credentials:
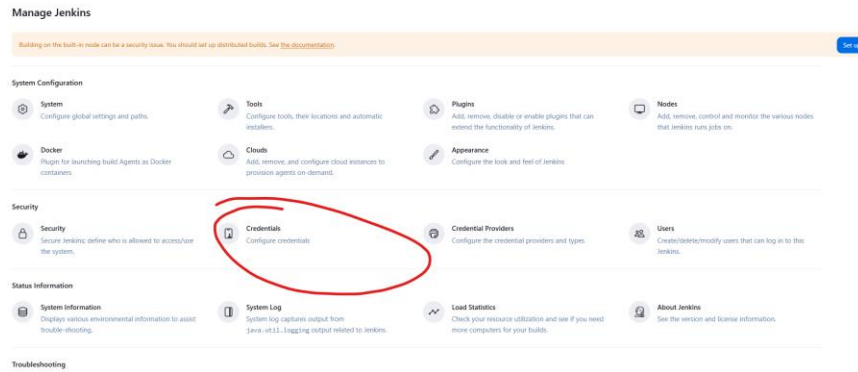
We will now add our Docker Hub credentials and the kubeconfig file that you should have saved from the Rancher Instance setup.

First, the kubeconfig file:

Add a new credential to System -> Global credentials.

Make it a "Secret file"

Copy these settings:
ID - kubeconfig_credentials
File – Select the .yaml file we downloaded earlier when setting up the Rancher Instance. This was the KubeConfig file of our cluster that we downloaded from Rancher.

This will allow us to deploy to our cluster even though we are not on the same instance. Jenkins can use Kubernetes CLI with this file to connect to our cluster on the other EC2 instance.

Second, lets add our Docker Hub credentials:

First, you need to go to Docker Hub and login to your account. Go to "Account settings."
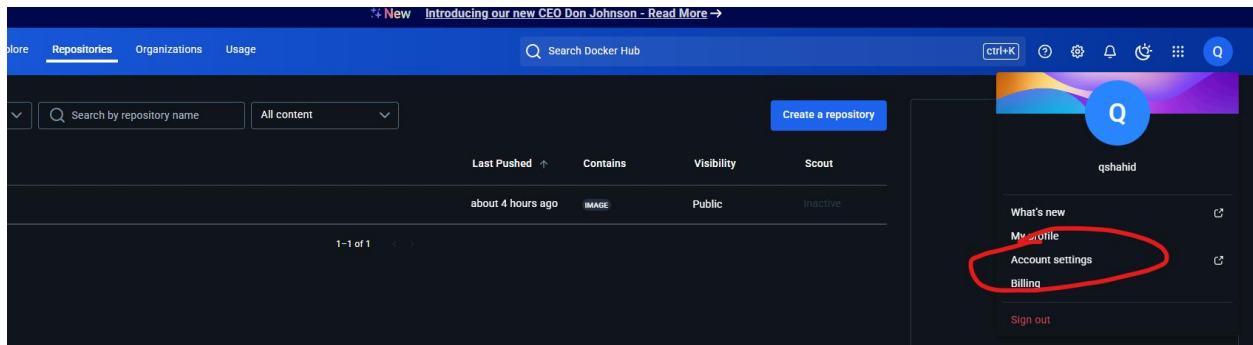
Go to "Personal access tokens" and create a new token.
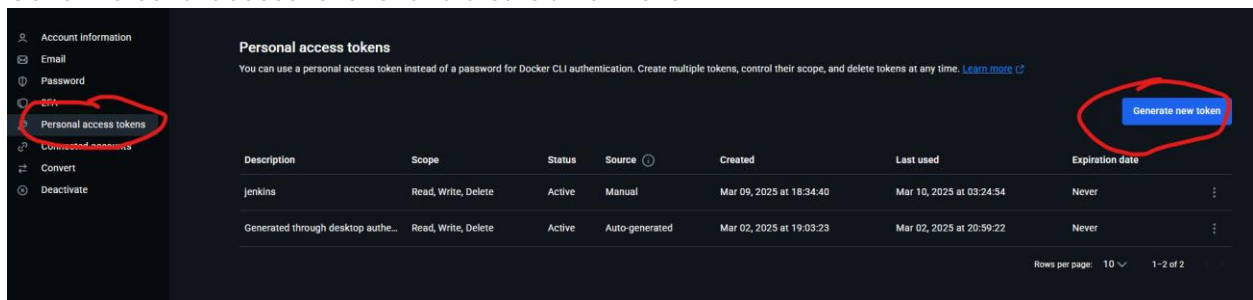
Copy these settings:
Access permissions – Read, Write, Delete
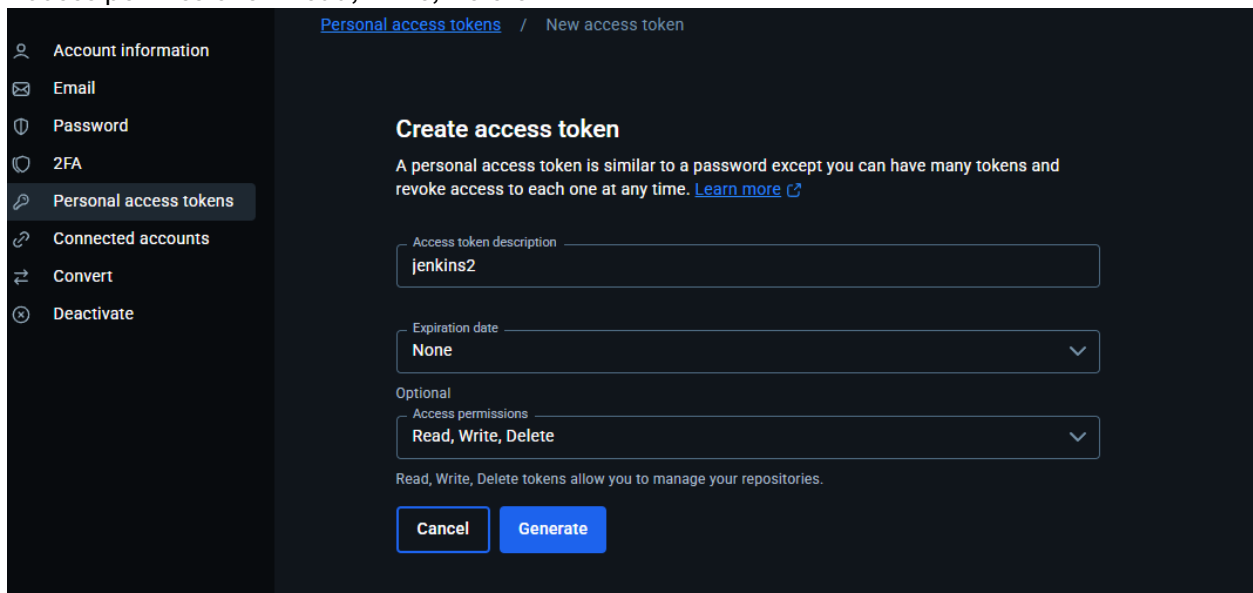
Click generate. After that, it will give you a password. Copy this and save it somewhere as we will need it here soon. This will only show once, so be careful and make sure to save it somewhere.
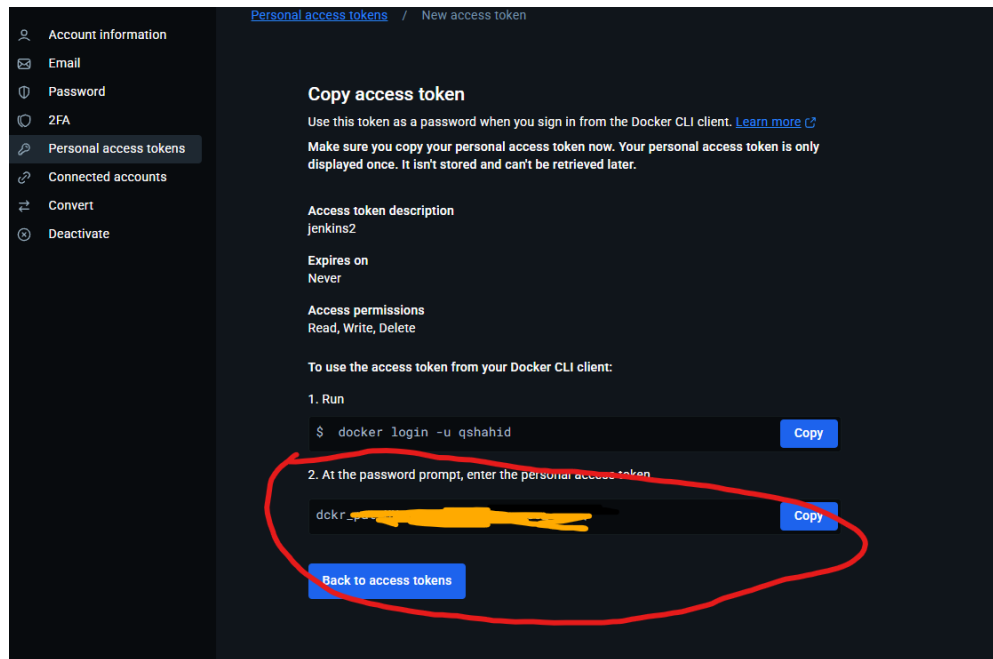
Now, go back to Jenkins and create another credential:

Make it as a "Username with password". Then, add your Docker Hub username and take the password you just copied from the Personal Access Token and paste it into the password field. Then, give it this id:
dockerhub_credentials

Save this credential. Here is an example:

Finally, we are ready to create our pipeline. Go back to the Jenkins dashboard and create a new item:

Create a new pipeline.

This will give you a bunch of options. We want to copy these settings:

Namely, do the following:

Select "GitHub hook trigger for GITScm polling" as a trigger

Pipeline definition should be from "Pipeline script from SCM". This means it will use our Jenkinsfile in our GitHub repository. By default, the name of the file is Jenkinsfile, so we don't need to change it.

Add the link to your GitHub repository.

Change the branch to "main". This tells Jenkins to build the main branch. This is important because it is no longer named "master," so make sure you enter "main" here.

After all of that, save the pipeline.

Congratulations, we are ready to do our first build. Everything should be setup properly now.

Running the Application

Assuming you followed all the rest of the steps, you should be ready to build and deploy the app to your cluster.

You can do these two ways.

Manually via Jenkins:
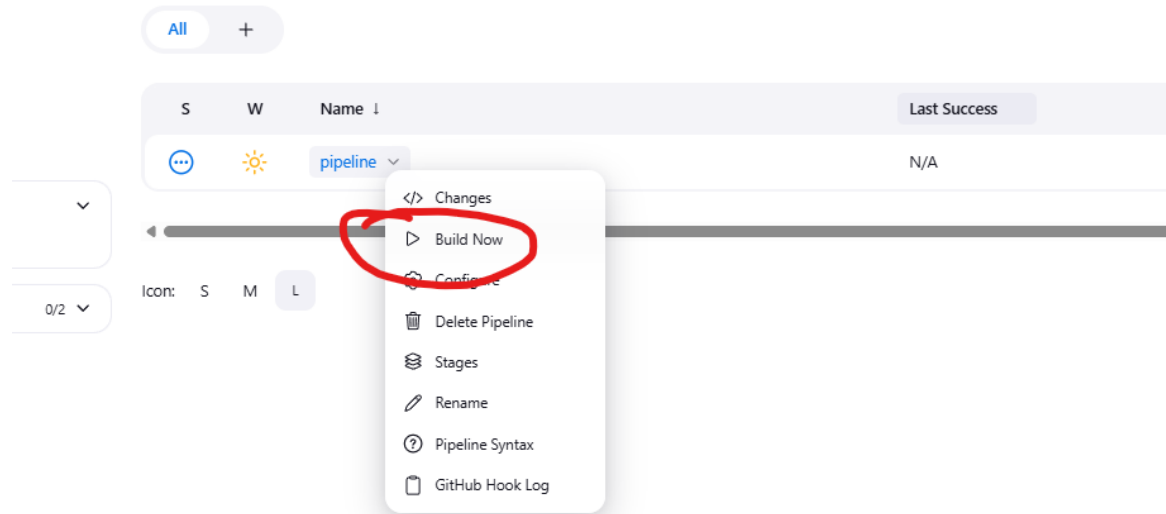
Alternatively, if you were able to correctly setup your webhook from your GitHub repository, then you can simply push updates to your main branch and the pipeline will be triggered automatically.

You can make changes to the survey.html, just as adding a timestamp, to do this.

IF YOU HAVE ANY ISSUES WITH THE DOCKER BUILD STEP:

Execute this command and then restart the Jenkins server.

sudo usermod -a -G docker Jenkins
sudo reboot

Once you either push a change to GitHub or manually trigger the build through Jenkins, you should see this in the build output console:

```
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // withDockerRegistry
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Deploy to Kubernetes)
[Pipeline] script
[Pipeline] {
[Pipeline] withKubeConfig
[Pipeline] {
[Pipeline] sh
+ kubectl apply -f deployment.yaml
deployment.apps/survey-app-deployment created
[Pipeline] sh
+ kubectl apply -f service.yaml
service/survey-app-service created
[Pipeline] sh
+ kubectl rollout restart deployment/survey-app-deployment
deployment.apps/survey-app-deployment restarted
[Pipeline] }
[kubernetes-cli] kubectl configuration cleaned up
[Pipeline] // withKubeConfig
[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Declarative: Post Actions)
[Pipeline] cleanWs
[WS-CLEANUP] Deleting project workspace...
[WS-CLEANUP] Deferred wipeout is used...
[WS-CLEANUP] done
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```


Here you can see our build succeeding (after a couple mistakes I made when recreating this, if you follow the guide, you should not have these.)
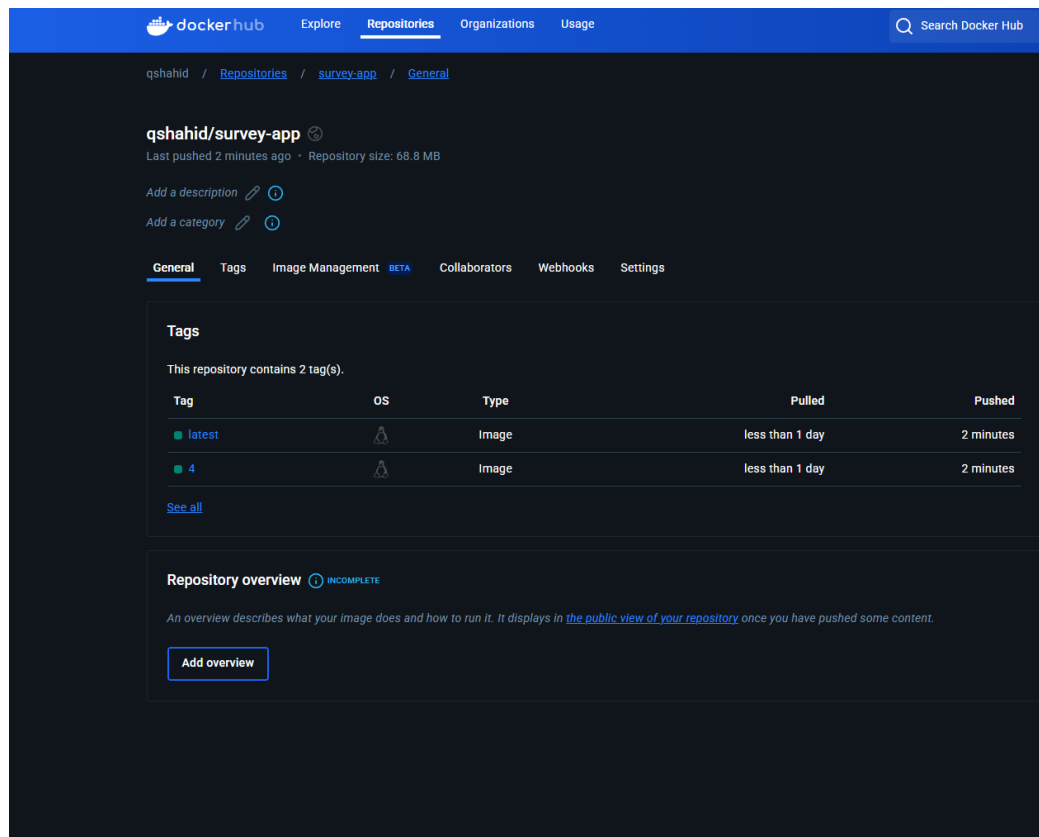
New image successfully pushed to Docker Hub under my account.
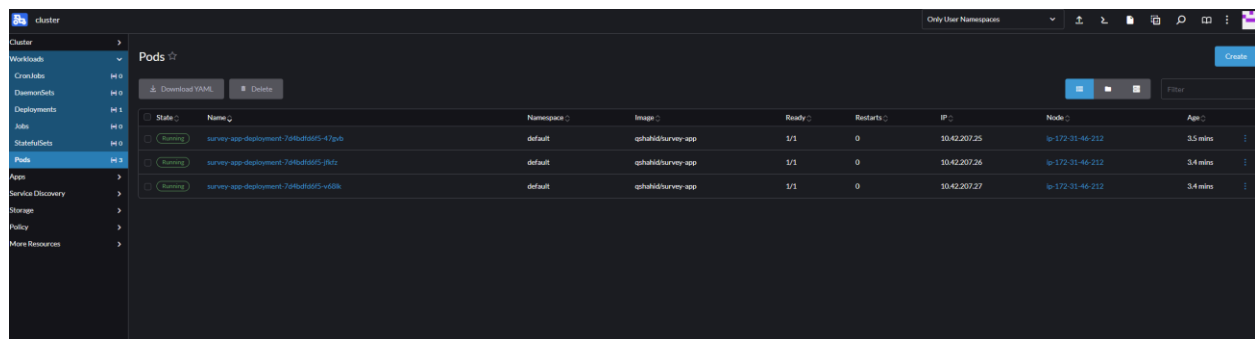
In Rancher, you can see the 3 pods / replicas we created as a part of our deployment.

You can access these at the Rancher Instance's public elastic IP. The NodePort service I used hosts the website on the 30001 port. So, you can go to:
http://<public-cluster-ip>:30001
For example:
http://54.80.152.86:30001

Student Survey Form 3/10/2025

First Name:

Last Name:

Street Address:

City: State: ZIP:

Telephone: Email:

Date of Survey:

mm/dd/yyyy

What did you like most about the campus?

- Students
- Location
- Campus
- Atmosphere
- Dorm Rooms
- Sports

How did you become interested in the university?

Here you can see my site.

I'll now push a small change to GitHub, let's see what happens.

Summary (required)

scription

Commit to main

mitted just now
ate survey.html

Undo

Status

Changes

Build Now

Configure

Delete Pipeline

Stages

Rename

Pipeline Syntax

GitHub Hook Log

Last GitHub Push

```
Started on Mar 10, 2025, 12:44:57 PM
Started by event from 140.82.115.60 ← http://54.87.102.0:8080/github-webhook/ on Mon Mar 10 12:44:56 UTC 2025
Using strategy: Default
[poll] Last Built Revision: Revision e594667ee5e9ae8b69e2a33f7ae5c43504357487 (refs/remotes/origin/main)
The recommended git tool is: NONE
No credentials specified
 > git --version # timeout=10
 > git --version # 'git version 2.43.0'
 > git ls-remote -h -- https://github.com/qasimshahid/qshahid-swe645-asst2/ # timeout=10
Found 1 remote heads on https://github.com/qasimshahid/qshahid-swe645-asst2/
[poll] Latest remote head revision on refs/heads/main is: 6fd8699394088e3876d09c12db71e2991fae8542
Using strategy: Default
The recommended git tool is: NONE
No credentials specified
 > git --version # timeout=10
 > git --version # 'git version 2.43.0'
 > git ls-remote -h -- https://github.com/qasimshahid/qshahid-swe645-asst2.git # timeout=10
Found 1 remote heads on https://github.com/qasimshahid/qshahid-swe645-asst2.git
[poll] Latest remote head revision on refs/heads/main is: 6fd8699394088e3876d09c12db71e2991fae8542
Done. Took 0.65 sec
Changes found
```
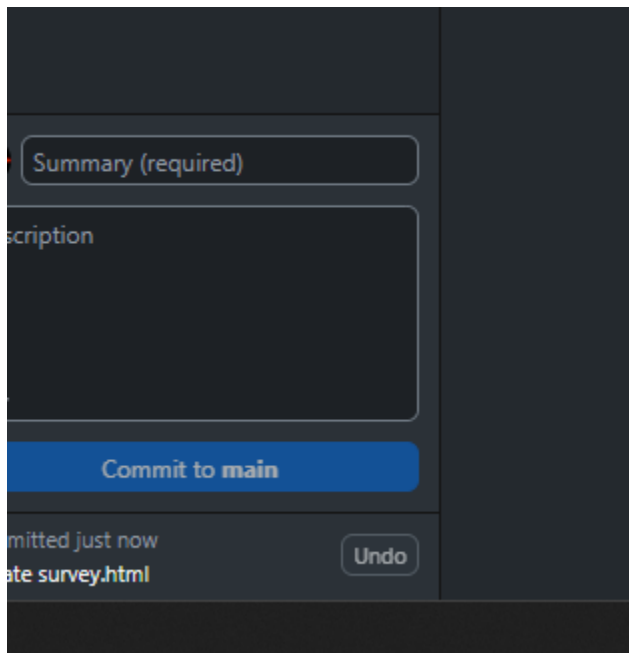
Builds

Filter

Today

#5 12:45 PM

#4 12:38 PM

#3 12:35 PM

#2 12:32 PM

#1 12:27 PM

Not secure 54.80.152.86:30001

Gmail YouTube ChatGPT DeepSeek Maps Spotify GMU Outlook School Finance Career Programming Utilities Misc JupyterLab AWS Academy Lear...

Student Survey Form 3/10/2025 9 AM

First Name:

Last Name:

Street Address:

City:

State:

ZIP:

Telephone:

Email:

Ignore the odd timing, I did this at 9 AM on 3/10/2025. You can see, pushing to main branch on my GitHub repository uses the webhook and triggers the build on the Jenkins server automatically.

References:
https://docs.docker.com/security/for-developers/access-tokens/
https://kubernetes.io/
https://plugins.jenkins.io/pipeline-github/
https://www.cprime.com/resources/blog/how-to-integrate-jenkins-github/
https://www.rancher.com/
https://ranchermanager.docs.rancher.com/
https://plugins.jenkins.io/kubernetes-cli/

Lessons Learned:
This was a great assignment and it taught me how to deploy my container images to a Kubernetes cluster, as well as how to make an automated CI/CD pipeline that automatically builds and pushes my container images to Docker Hub and then deploys to my cluster. I really learned a lot, such as learning how to setup Rancher and Jenkins. Jenkins scripting was also quite interesting and seeing how credentials are stored and so on was eye-opening for me. This experience will help me in my career hopefully in the future. It was awesome to see my website being updated in real time after I pushed my changes to GitHub.