HW Assignment-2 – Installation Guide

Team Members:
Mohnish Raval – G01373613

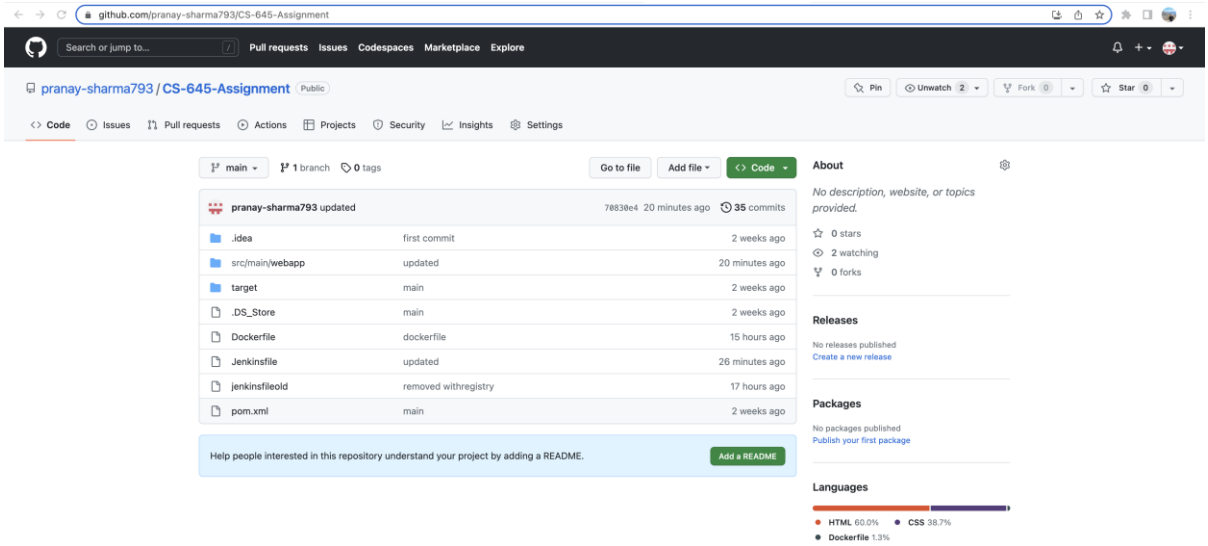
Pranay Sharma – G01393761

Dipak Shetty – G01380853

Anand Seshadri – G01351350

In this guide we will demonstrate the Installation process which was followed for the Homework Assignment 2.

1. Create a new GitHub repository and push all the existing code of Homework assignment-1 onto it along with Dockerfile and Jenkinsfile in root directory of GitHub repository.

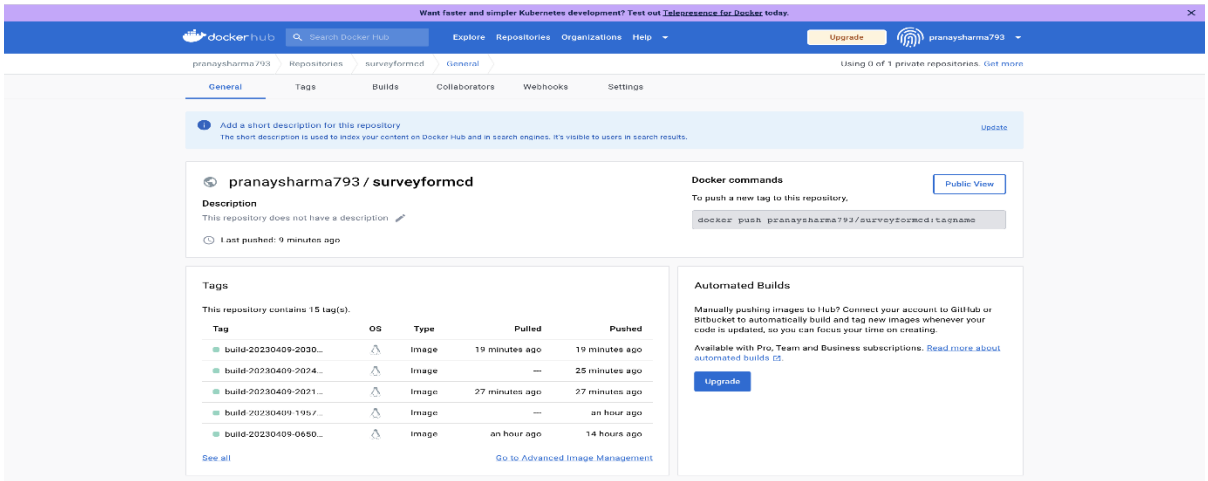


2. Create an account on Docker Hub and push your image there.
   Docker run & build –

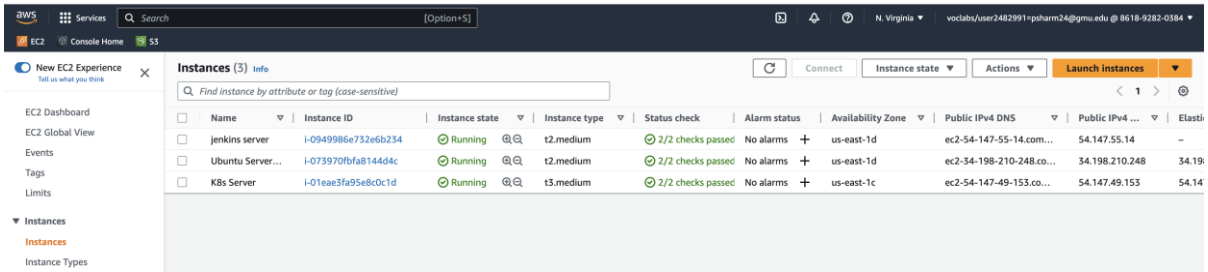
   docker build -t surveyformcd .
   docker run -p 8080:8080 surveyformcd

   Docker tag & push -

   docker tag surveyformcd:latest docker.io/surveyformcd:latest
   docker login docker.io
   docker push pranaysharma793/surveyformcd:latest



3. Log on to your free-tier AWS account via Learner Lab. From here we would be creating our CI-CD pipeline using Rancher and Jenkins.
   - To begin with create, 3 EC2 instances, one would be our Rancher server, the second would be for the Kubernetes cluster and the third will be the Jenkins instance.



4. We begin with the first instance, which is the Rancher server. Connect to this server and install docker in this instance using the following commands:
   - Update: *sudo apt-get update*
   - Install Docker: *sudo apt install docker.io*
   - Check docker version once installed: *docker –v*

   Once docker is installed, we download rancher using the command:

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

   Refer the screenshots below.

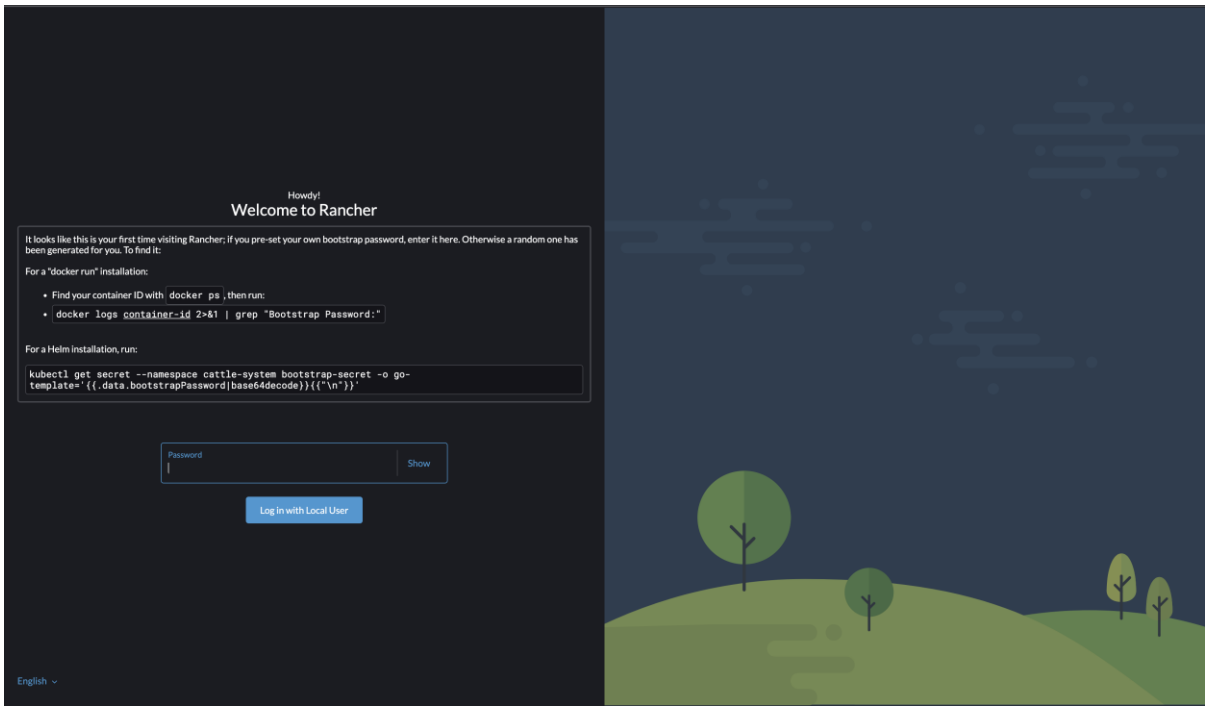
```
Fetched 26.6 MB in 4s (6722 kB/s)
Reading package lists... Done
ubuntu@ip-172-31-91-40:~$ sudo docker -v
sudo: docker: command not found
ubuntu@ip-172-31-91-40:~$ sudo apt install docker.io
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following 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 12 not upgraded.
Need to get 72.4 MB of archives.
After this operation, 287 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 pigz amd64 2.6-1 [63.6 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 bridge-utils amd64 1.7-1ubuntu3 [34.4 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 runc amd64 1.1.4-0ubuntu1-22.04.1 [4241 kB]
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 containerd amd64 1.6.12-0ubuntu1-22.04.1 [34.4 MB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 dns-root-data all 2021011101 [5256 B]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 dnsmasq-base amd64 2.86-1.1ubuntu0.2 [354 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 docker.io amd64 20.10.21-0ubuntu1-22.04.2 [33.2 MB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 ubuntu-fan all 0.12.16 [35.2 kB]
Fetched 72.4 MB in 2s (43.0 MB/s)
Preconfiguring packages ...
Selecting previously unselected package pigz.
(Reading database ... 63657 files and directories currently installed.)
Preparing to unpack .../0-pigz_2.6-1_amd64.deb ...
Unpacking pigz (2.6-1) ...
Selecting previously unselected package bridge-utils.
Preparing to unpack .../1-bridge-utils_1.7-1ubuntu3_amd64.deb ...
Unpacking bridge-utils (1.7-1ubuntu3) ...
Selecting previously unselected package runc.
Preparing to unpack .../2-runc_1.1.4-0ubuntu1-22.04.1_amd64.deb ...
Unpacking runc (1.1.4-0ubuntu1-22.04.1) ...
Selecting previously unselected package containerd.
Preparing to unpack .../3-containerd_1.6.12-0ubuntu1-22.04.1_amd64.deb ...
Unpacking containerd (1.6.12-0ubuntu1-22.04.1) ...
Selecting previously unselected package dns-root-data.
Preparing to unpack .../4-dns-root-data_2021011101_all.deb ...
Unpacking dns-root-data (2021011101) ...
Selecting previously unselected package dnsmasq-base.
```

```
aws  Services  Search  [Option+S]  N. Virginia  voclabs/user2482991=psharm24@gmu.edu @ 8618-9282-0384
EC2  Console Home  S3
Adding group `docker' (GID 122) ...
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-172-31-91-40:~$ sudo docker run -d --restart=unless-stopped -p 80:80 -p 443:443 rancher/rancher
Unable to find image 'rancher/rancher:latest' locally
latest: Pulling from rancher/rancher
fb44d0195361: Pull complete
28e8c69365bc: Pull complete
69133ddd606f: Pull complete
552cf618daee: Pull complete
f4ccca29e2ec: Pull complete
46c3a18b4dc4: Pull complete
45702284aa96: Pull complete
8df327249a50: Pull complete
54c378396e21: Pull complete
d6ba306d9dd5: Pull complete
b44793c1e7d3: Pull complete
ffde21a6ab12: Pull complete
aaeeb000d225: Pull complete
aa421db55e87: Pull complete
907c7a663008: Pull complete
8745a92f349a: Pull complete
a4cc830ed766: Pull complete
ce5f83f3158f: Pull complete
b11162943c37: Pull complete
194836056959: Pull complete
Digest: sha256:188ac186125ca1d4bef1e741568ec381eed1af4e9a876c7b15eabcc98325f2b0
Status: Downloaded newer image for rancher/rancher:latest
943b49195684c57d9b7d83303ba685eab6cfdbee634779db8f55d67077677ee5
ubuntu@ip-172-31-91-40:~$
```

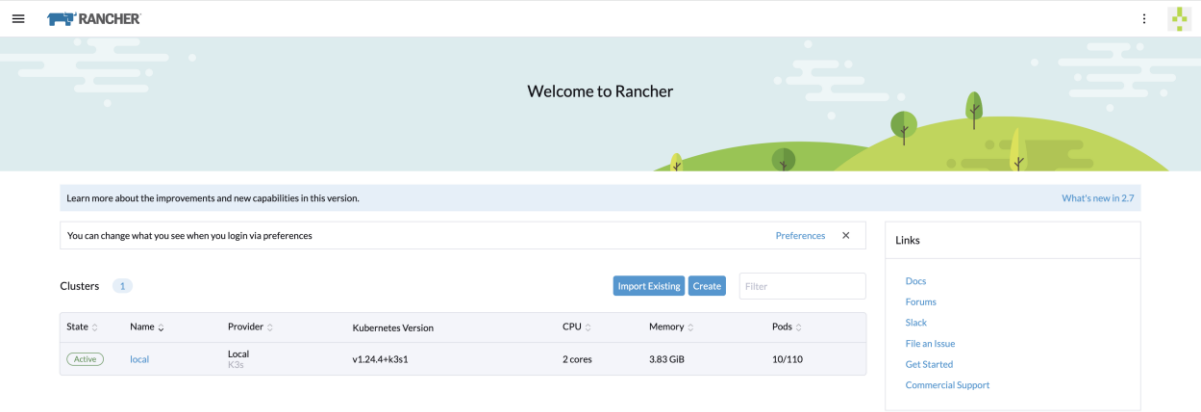
5. Next, check if Rancher is running successfully using the `*docker ps*` command as shown below.

```
ubuntu@ip-172-31-91-40:~$ sudo docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
ubuntu@ip-172-31-91-40:~$ sudo docker run --privileged=true -d --restart=unless-stopped -p 80:80 -p 443:443 rancher/rancher
39d74ac0ad4748f901592371dac9f1c9ee5f484e333f90ea301ae335c1f87081
ubuntu@ip-172-31-91-40:~$ sudo docker ps
CONTAINER ID   IMAGE             COMMAND           CREATED         STATUS         PORTS                                                                                   NAMES
39d74ac0ad47   rancher/rancher   "entrypoint.sh"   3 seconds ago   Up 2 seconds   0.0.0.0:80->80/tcp, :::80->80/tcp, 0.0.0.0:443->443/tcp, :::443->443/tcp   beautiful_ellis
ubuntu@ip-172-31-91-40:~$ sudo docker ps
```

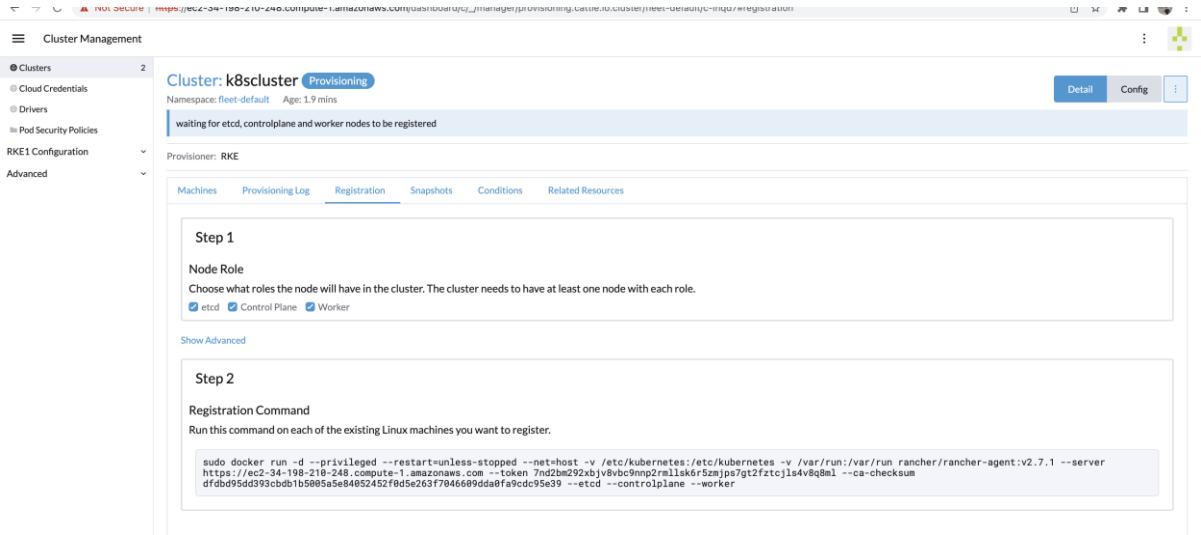
6. After checking if rancher is running, wait for a bit and then open our browser and copy/paste the public DNS of our first instance, which is the Rancher server. After some warnings, we get this screen as shown below.



7. Once you are logged in, click on the ‘Create’ button and create a Kubernetes cluster.



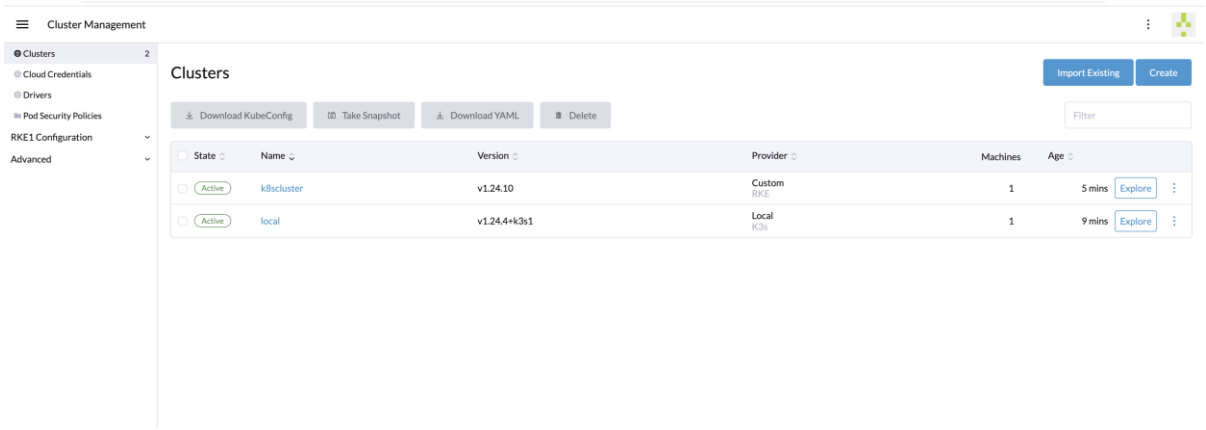
8. Here we are creating a Custom cluster by the name of k8cluster as shown below (choose the Custom option).



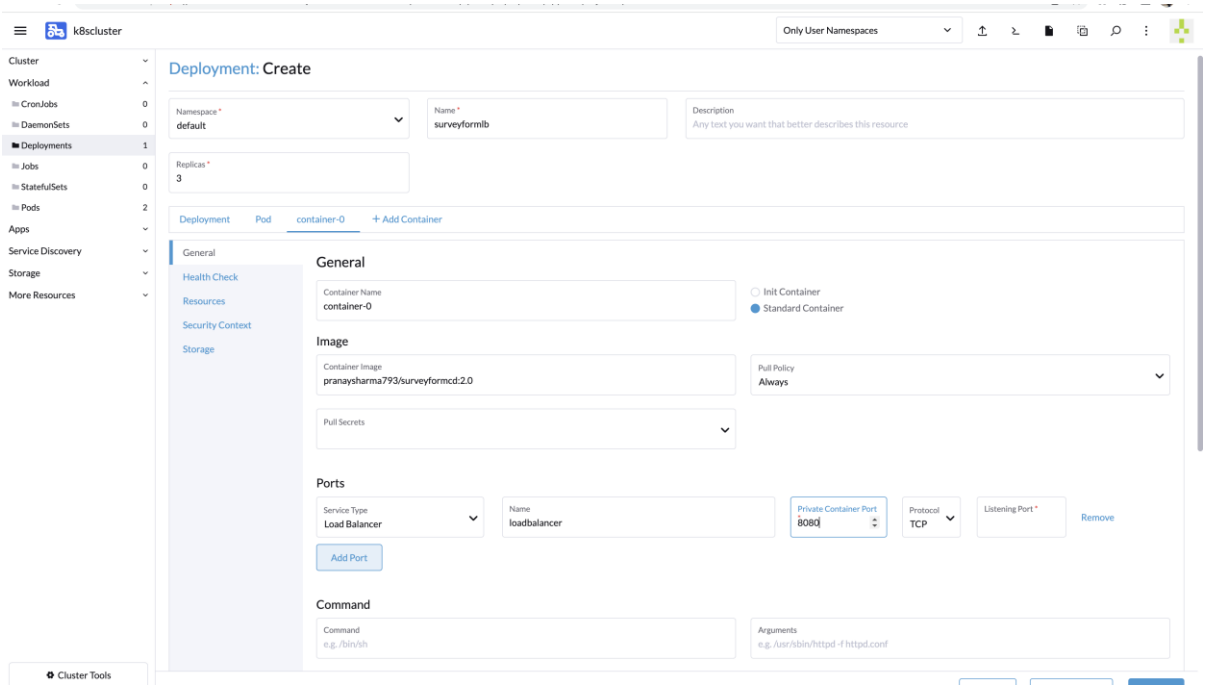
9. Next you need to register the etcd, the control plane and the worker node on the second EC2 instance which was created for the kubernetes cluster as mentioned earlier. To do this, run the command obtained above (while creating the custom cluster) by logging into the second EC2 instance as shown below.

```
Last login: Sun Apr  9 06:44:07 2023 from 18.206.107.28
ubuntu@ip-172-31-7-245:~$ sudo docker run -d --privileged --restart=unless-stopped --net=host -v /etc/kubernetes:/etc/kubernetes -v /var/run:/var/run  rancher/rancher-agent:v2.7.1 --server https://e
c2-34-198-210-248.compute-1.amazonaws.com --token 7nd2bm292xbjv8vbc9nnp2rmllsk6r5zmjps7gt2fztcjls4v8q8ml --ca-checksum dfdbd95dd393cbdb1b5005a5e84052452f0d5e263f7046609dda0fa9cdc95e39 --worker
2e2dd022aaa97ea901ad7ff5e116c983ca1052b7df99b20f872b2ff36f25fd4f
ubuntu@ip-172-31-7-245:~$
```

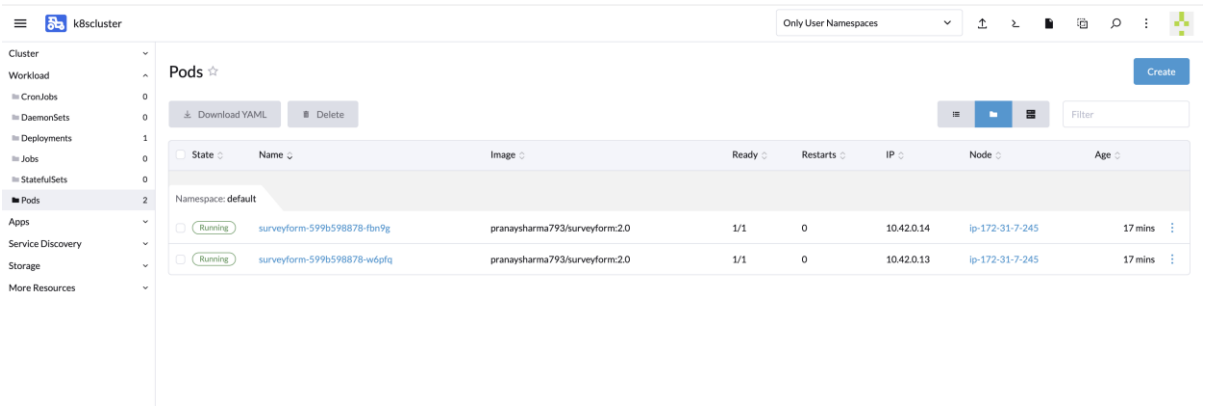
10. Once it is created, you can see the created k8cluster as `Active`.



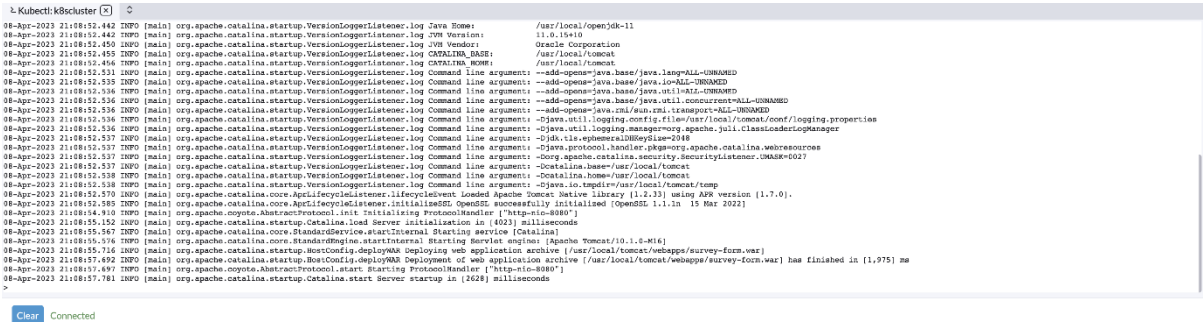
11. Once the cluster is created, create two deployments inside the cluster. One would be the nodeport deployment and the other would be the loadbalancer deployment.
    The deployment shown below is for the loadbalancer, wherein:
    - Give the name for the deployment as `surveyformlb`.
    - Give number of Replicas as 3.
    - Give your docker image name in the Container Image field. This would be the image which we created earlier and pushed to DockerHub.
    - Next, add a port with Service type as Load Balancer on port 8080, with the protocol set as TCP.



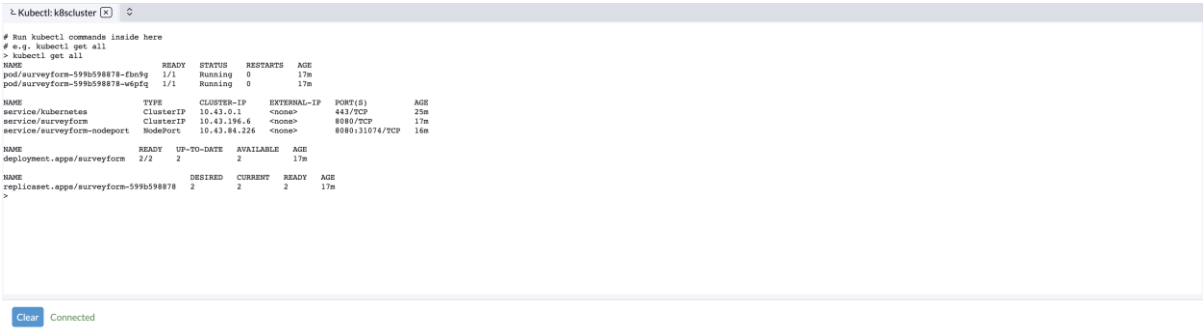
12. Once it is created, the two pods mentioned in the deployment can be seen running as shown below.



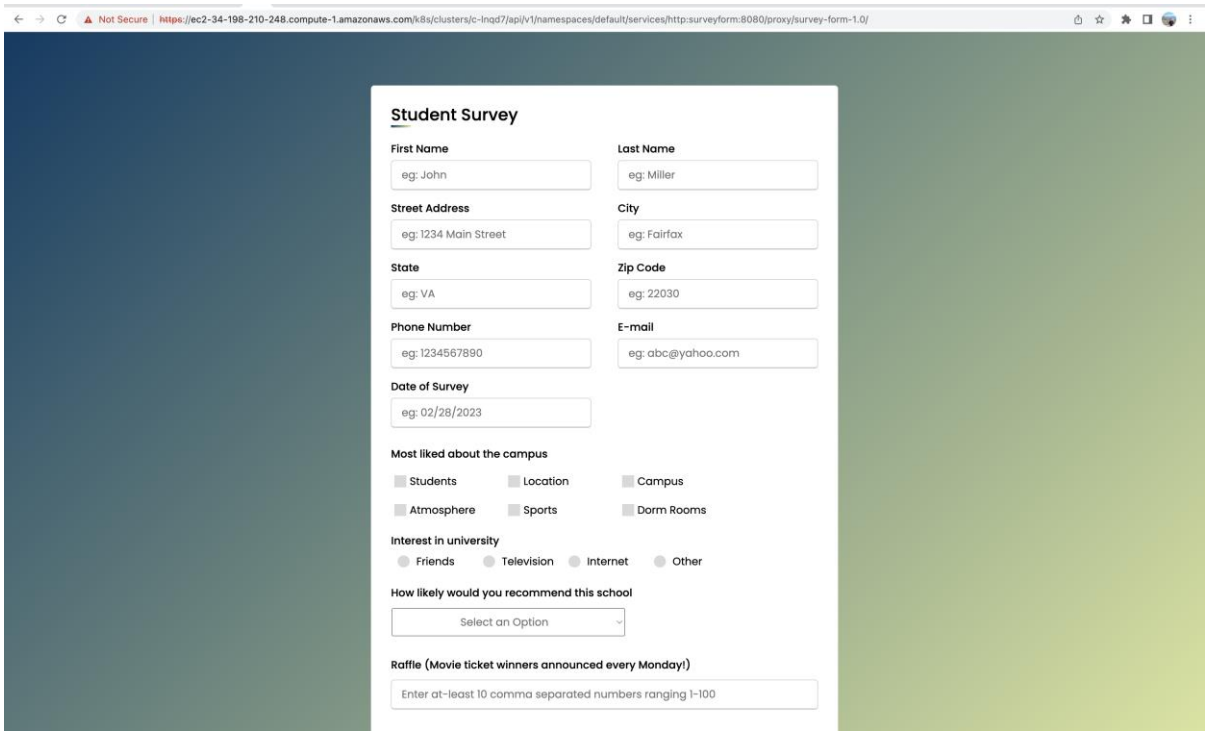
13. Also, when we check the deployment logs, we can see that our application war (survey-form.war) was loaded successfully as shown below.



14. Using the `kubectl get all` command, we can see that the pods are in the running state and ready.



15. We can now access our form using the TCP url of this pod.



16. Now, log in to the third EC2 instance, which was created for Jenkins. After connecting to this instance, first download JDK as shown below.

```
16 updates can be applied immediately.
12 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


Last login: Sat Apr  8 20:50:24 2023 from 129.174.255.57
ubuntu@ip-172-31-91-40:~$ 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 libgif7 libgl1 libgl1-amber-dri libgl1-mesa-dri
  libglapi-mesa libglvnd0 libglx-mesa0 libglx0 libgraphite2-3 libharfbuzz0b libice-dev libice6 libjpeg-turbo8 libjpeg8 liblcms2-2 libllvm15
  libpciaccess0 libpcsclite1 libpthread-stubs0-dev libsensors-config libsensors5 libsm-dev libsm6 libx11-dev libx11-xcb1 libxau-dev libxaw7
  libxcb-dri2-0 libxcb-dri3-0 libxcb-glx0 libxcb-present0 libxcb-shape0 libxcb-shm0 libxcb-sync1 libxcb-xfixes0 libxcb1-dev libxcomposite1
  libxdmcp-dev libxfixes3 libxft2 libxi6 libxineramaI libxkbfile1 libxmu6 libxpm4 libxrandr2 libxrender1 libxshmfence1 libxt-dev libxt6 libxtst6
  libxv1 libxxf86dga1 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 pcscd lm-sensors libsm-doc libx11-doc libxcb-doc libxt-doc
  openjdk-11-demo openjdk-11-source visualvm libnss-mdns fonts-ipafont-gothic fonts-ipafont-mincho fonts-wqy-microhei | fonts-wqy-zenhei
  fonts-indic mesa-utils
The following NEW packages will be installed:
  alsa-topology-conf alsa-ucm-conf at-spi2-core ca-certificates-java dconf-gsettings-backend dconf-service fontconfig-config fonts-dejavu-core
  fonts-dejavu-extra gsettings-desktop-schemas java-common libasound2 libasound2-data libatk-bridge2.0-0 libatk-wrapper-java
  libatk-wrapper-java-jni 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 libgif7 libgl1 libgl1-amber-dri libgl1-mesa-dri
  libglapi-mesa libglvnd0 libglx-mesa0 libglx0 libgraphite2-3 libharfbuzz0b libice-dev libice6 libjpeg-turbo8 libjpeg8 liblcms2-2 libllvm15
```

17. Once JDK is successfully installed, use the following 3 commands to install Jenkins as shown in the screenshot below. Which are:
    - *sudo apt-get update*
    - *sudo apt-get install fontconfig openjdk-11-jre*
    - *sudo apt-get install jenkins*

```
curl -fsSL https://pkg.jenkins.io/debian/jenkins.io-2023.key | sudo tee     /usr/share/keyrings/jenkins-keyring.asc > /dev/null
echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc]     https://pkg.jenkins.io/debian binary/ | sudo tee     /etc/apt/sources.list.d/jenkins.list > /dev/null
sudo apt-get update
sudo apt-get install fontconfig openjdk-11-jre
sudo apt-get install jenkins
```

18. Once Jenkins is installed we can check if it is running by using the `systemctl status jenkins` command, which will show the active (running) status as shown below.

```
No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-91-40:~$ systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
     Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor preset: enabled)
     Active: active (running) since Sat 2023-04-08 21:34:18 UTC; 58s ago
   Main PID: 12807 (java)
      Tasks: 49 (limit: 4686)
     Memory: 1.1G
        CPU: 49.132s
     CGroup: /system.slice/jenkins.service
             └─12807 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080

Apr 08 21:33:57 ip-172-31-91-40 jenkins[12807]: 5fddaaa582a741e5af1a206e6d5431c7
Apr 08 21:33:57 ip-172-31-91-40 jenkins[12807]: This may also be found at: /var/lib/jenkins/secrets/initialAdminPassword
Apr 08 21:33:57 ip-172-31-91-40 jenkins[12807]: *************************************************************
Apr 08 21:33:57 ip-172-31-91-40 jenkins[12807]: *************************************************************
Apr 08 21:33:57 ip-172-31-91-40 jenkins[12807]: *************************************************************
Apr 08 21:34:18 ip-172-31-91-40 jenkins[12807]: 2023-04-08 21:34:18.193+0000 [id=28]	INFO	jenkins.InitReactorRunner$1#onAttained: Completed initialization
Apr 08 21:34:18 ip-172-31-91-40 jenkins[12807]: 2023-04-08 21:34:18.216+0000 [id=22]	INFO	hudson.lifecycle.Lifecycle#onReady: Jenkins is fully up and running
Apr 08 21:34:18 ip-172-31-91-40 systemd[1]: Started Jenkins Continuous Integration Server.
Apr 08 21:34:18 ip-172-31-91-40 jenkins[12807]: 2023-04-08 21:34:18.333+0000 [id=46]	INFO	h.m.DownloadService$DownloadableLoad: Obtained the updated data file for hudson.tasks.Maven.MavenInstaller
Apr 08 21:34:18 ip-172-31-91-40 jenkins[12807]: 2023-04-08 21:34:18.335+0000 [id=46]	INFO	hudson.util.Retrier#start: Performed the action check updates server successfully at the attempt #1
ubuntu@ip-172-31-91-40:~$
```

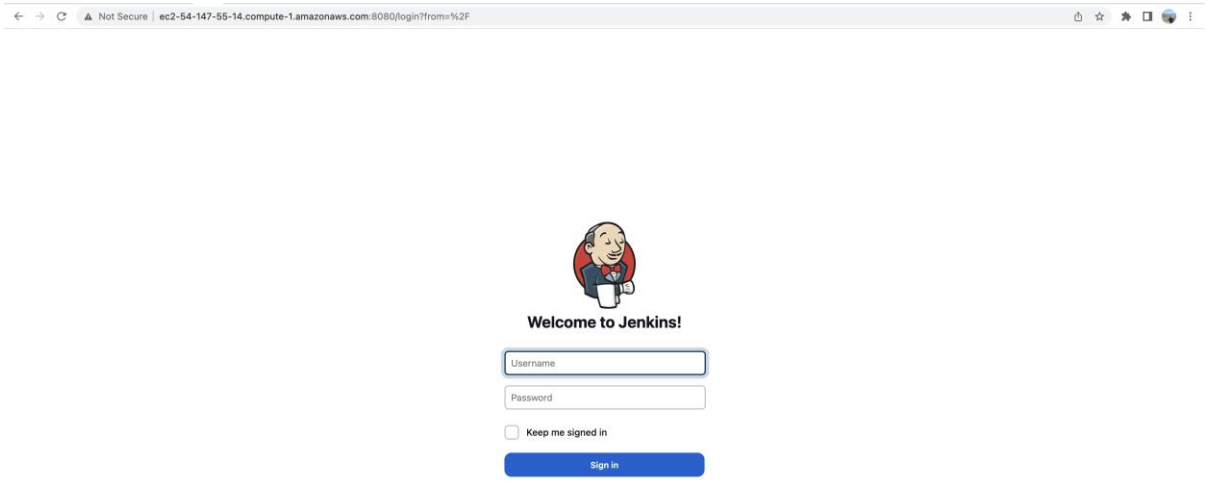
19. Now, before we log in to Jenkins, we already have the kubeconfig file downloaded. We will use this to create the config file inside the `*.kube*` directory. Paste the content of the config here.

```
jenkins@ip-172-31-92-171:~/.kube$ cat config
apiVersion: v1
kind: Config
clusters:
- name: "k8scluster"
  cluster:
    server: "https://ec2-34-198-210-248.compute-1.amazonaws.com/k8s/clusters/c-lnqd7"
    certificate-authority-data: "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCklJSUJ2RENDQ\
      VdPZ0F3SUJBZ01CQURBS0JnZ3Foa2pPUFFRREFqQkdNUnd3R2dZRFZRUUtFeE5rZVc1aGJXbGoKY\
      kdsemRHVnVaWEl0YjNKbk1TWXdKQVlEVlFRRERCMWt1VzVoYldsamJHbHpkR1Z1WlhJdFkyRkFNV\
      Fk0TURrNApOekUxTnpBZUE3MHlNekEwTURneU1EVXlNemRhRncwek16QTBNRFV5TURVeU16ZGFNR\
      Vl4SERBYUJnT1ZCQW9UCkUyUjViUE0YVdOc2FYTjBaVzVsY2kxdmNtY3hKakFrQmdOVkJBTU1IV\
      lI1YmlGdGFXTnNhWE4wWlclbGNpMWoKWVVBeE5qZ3dPVGczTVRVM01Ga3dFdllIS29aSXpqMENBU\
      VlJS29aSXpqMEBBUWNEUWdBRVdHMmxHVDh0bGhDKwpCZU14QmxjVVp4UE9RUmc4cTc4dOV3eDluS\
      jdXU2lFNWNXbGp2RmE5TjY2UTZiVEQraEd2Y2I5blA4S0ZNNUNLCmcwNzE0ZDlDb2FOQ01FQXdEZ\
      llEVlIwUEFRSC9CQVFEQWdLa01BOEdBMVVkRXdFQi93UUZNQU1CQWY4d0hRWUQKVlIwT0JCWUVGR\
      XBPMzhpTGt4TTdxMGsvVEY1SitOVE9mVnZkTUFvR0NDcUdTTTQ5QkFNQ0EwY0FNRVFDSUY5SApzV\
      S9meXJmNlVMa0d3UjRFa3ArUjBubFNuc3ZxdHV0eWltQXYvbElLQWlCTkNGVUJIaXVjKys2NXZKN\
      jZVWEkxClQ3L2MyZ1ZGZ01QVjArRU1raEhINkE9PQotLS0tLUVORCBDRVJUSUZJQ0FURS0tLS0t"
- name: "k8scluster-ip-172-31-7-245"
  cluster:
    server: "https://172.31.7.245:6443"
    certificate-authority-data: "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCklJSUM0VENDQ\
      WNtZ0F3SUJBZ01CQURBTkJna3Foa2lHOXcwQkFRc0ZBREFTTVJBd0RnWURWUVFERXdkcmRXSmwKT\
      FdOaE1CNFhEVE16TURRd09ESXdOVGd5TkZvWERUTXpNRFF3TlRJd05UZ3lORm93RWpFUU1BNEdBM\
      VVFQXhNSAphM1ZpWlMxallUQ0NBU013RFFZSktvWklodmNOQVFFQkJRQURnZ0VQQURDQ0FRb0Nnz\
      0VCQUs4cGZDZVdaUkZvCnJxb0hDZkZUU044V3dxSENXT01JY2hBL05JYW5sSml0bDN3Zmk0Vjq0a\
      XJUMDl4SzBlMGlGa3VGVmzhOVZvbEQKY2NFNloyUVlTSnFQOUh5Z2tWdX1zUW44VFVCblJTU0lvU\
      24vUm0d3kzNHRCeWxWM3hwSnY1NONlaVlDdlVxQgpqTWRzU9QWERua2thY1NEM01TaEFiMHdrO\
      HJxeHq0YnlmVzVzRW9rQTRSZUJMMWhVYnlCNU1rNkNTcURkdi9ECkZVbm4wNmRuOTlkck9Hgy94c\
      CtMNnJJbXVycXpmY1JqSitVMExBaUoxU3VpQTNDeExtbVIlclRDbWlkNjg1Yk8KWFJ2aHh3elF2b\
      mNqOTI0OVRDbEliWk4lVXBqbVhsSFo5cXZmV1RidUVIU2VMUStHWllCQnUrM0didXFhYUxZMQp6e\
      lRCbldNS0V1RUNBd0VBQWFOQ01FQXdEZllEVlIwUEFRSC9CQVFEQWdLa01BOEdBMVVkRXdFQi93U\
      UZNQU1CCkFmOHdIUVlEVlIwT0JCWUVGTXd4ekNnWnFmbU5GUUVnUDJxQ25Dejc4aVF5TUEwR0NTc\
      UdTSWIzRFFFQkN3VUEKQTRJQkFRQTZlT3BsTHVZMUk4Q3VhS05PYk9EVFlsNllpaKQ4bDZDNTdLR\
      EF5NzN1U3NwQlRCK3BiWEhqaJJVKwpnYXJXZXluVk8zK2xobUkzdGZ0clJqcUtENThMdNhRWVlsa\
      UhxcHlkZDZsdzZWektPK2cwU2U2ZStZZ0J0MmNDCmJrL2FHbldoVEkyRGJ4RnpUODBnOUFvVHlzW\
      DM4dVFYaHpWTDhrU0RmQlhsSTIrKzBycjJZb2hQNEYyU0dqR28KaCtOdUdHSWNHbHlKUld6bE5XR\
      3JURmc0QTFzb3lmYUpkLzI4YkdJenREUkkxbXMlTzhZZUV2YTA4aHFqRXhyWAo3eTVzUDJVc3NNN\
      Dg3VkFUbzdMNDlmRmZxdzQwWjVtVU1Rd1MzVUZod3E5cUlHcUFYaVdsL0w4N1JjVS9zLzVTCjRKd\
      2tCSVRzY0wlV1VoZFRTT2Q3RTBoWS9qVlcKLS0tLS1FTkQgQ0VSVElGSUNBVEUtLS0tLQo="

users:
- name: "k8scluster"
  user:
```

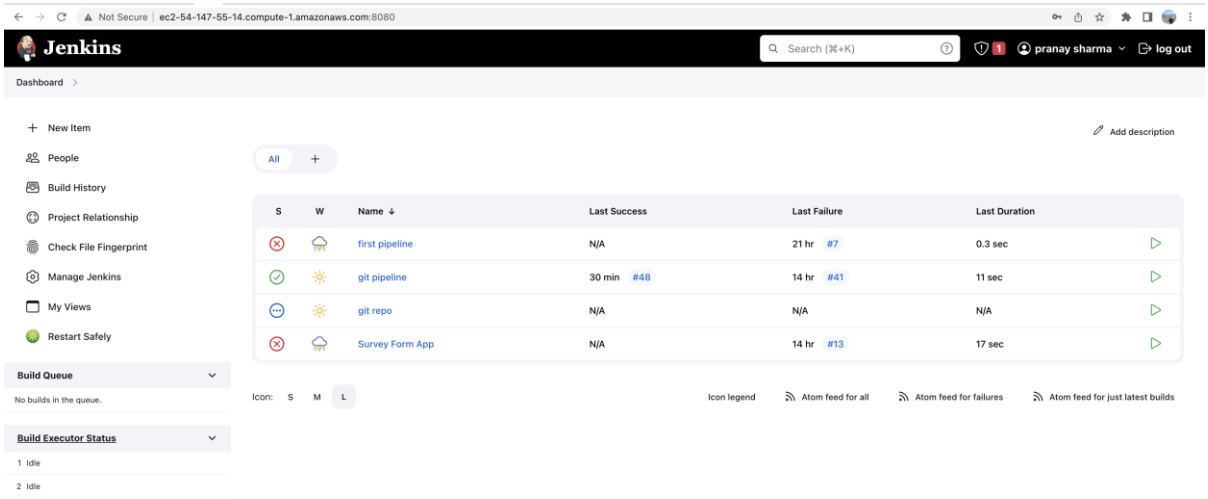
20. Next, run the following command `*kubectl config current-context*` which return the context name, which will match with your created cluster name which we gave on Rancher.

```
jenkins@ip-172-31-92-171:/home/ubuntu$ cd ~
jenkins@ip-172-31-92-171:~$ kubectl config current-context
k8scluster
jenkins@ip-172-31-92-171:~$
```

21. Next, create an account and log in to Jenkins using the url from the third EC2 instance used for Jenkins.



22. Now, create a `New Item`. This will be the pipeline for our project.



To do this, check your Dockerfile in your project directory, weather it is using the proper tomcat JDK, and if it is taking the proper war file and placing it in the *`/tomcat/webapps/`* foder. Check the COPY command shown below. Also, make sure it is exposed on port 8080 and if the command to run – the CMD line, is given properly as shown below.

```
#FROM tomcat:10.1-jdk11-openjdk
#
#COPY /target/survey-form.war /usr/local/tomcat/webapps/
#
#EXPOSE 8080
#
#CMD ["catalina.sh", "run"]

# Use official Tomcat runtime as a parent image
FROM tomcat:9.0.31-jdk8-openjdk

# Copy the war file into the container
COPY target/*.war /usr/local/tomcat/webapps/

# Expose port 8080 for the container
EXPOSE 8080

# Start Tomcat server
CMD ["catalina.sh", "run"]
```

23. Also, along with the Dockerfile, Jenkins requires a Jenkins file, which has to be placed in your GitHub repository. A JenkinsFile is essentially a text file that contains the definition of a Jenkins Pipeline and is checked into source control. This file would have all the configs and steps required to build the pipeline.

The DockerHub password will be passed as an environmental variable to the jenkinsFile and so will the timestamp.

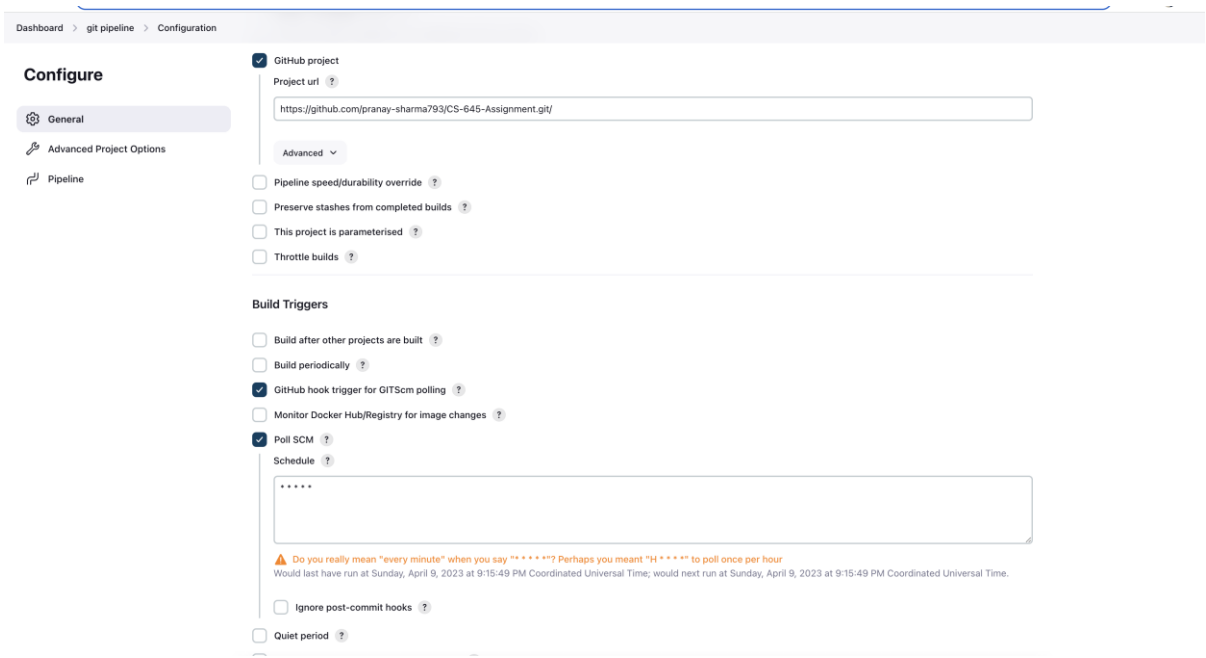
As you can see below, it has several stages, like `build`, `Push to Docker Hub` and then `Deploying the image on Rancher single node` which is the surveyfromlb node which we created earlier.

```
@NonCPS
def generateTag() {
    return "build-" + new Date().format("yyyyMMdd-HHmmss")
}

pipeline {
    environment {
        registry = "pranaysharma793/surveyformcd"
        registryCredential = 'dockercred'
    }
    agent any

    stages{

        stage('Build') {
            steps {
                script {
                    sh 'mvn clean package'
                    sh 'echo ${BUILD_TIMESTAMP}'
//                      echo "Prakhar@79" | docker login --username pranaysharma793 --password-stdin
                    tag = generateTag()
                    docker.withRegistry('',registryCredential){
                      def customImage = docker.build("pranaysharma793/surveyformcd:"+tag)
                    }
                }
            }
        }

        stage('Push to Docker Hub') {
            steps {
                script {
                    sh 'echo ${BUILD_TIMESTAMP}'
//                      echo "Prakhar@79" | docker login --username pranaysharma793 --password-stdin
                    docker.withRegistry('',registryCredential) {
                        def image = docker.build('pranaysharma793/surveyformcd:'+tag, '.')
                        docker.withRegistry('',registryCredential) {
                            image.push()
                        }
                    }
                }
            }
        }

      stage('Deploying Rancher to single node') {
          steps {
              script{
                  sh 'kubectl set image deployment/surveyform container-0=pranaysharma793/surveyformcd:'+tag
```

```
                    docker.withRegistry('',registryCredential){
                      def customImage = docker.build("pranaysharma793/surveyformcd:"+tag)
                    }
                }
            }
        }

        stage('Push to Docker Hub') {
            steps {
                script {
                    sh 'echo ${BUILD_TIMESTAMP}'
//                      echo "Prakhar@79" | docker login --username pranaysharma793 --password-stdin
                    docker.withRegistry('',registryCredential) {
                        def image = docker.build('pranaysharma793/surveyformcd:'+tag, '.')
                        docker.withRegistry('',registryCredential) {
                            image.push()
                        }
                    }
                }
            }
        }

      stage('Deploying Rancher to single node') {
          steps {
              script{
                sh 'kubectl set image deployment/surveyform container-0=pranaysharma793/surveyformcd:'+tag
              }
          }
      }

    stage('Deploying Rancher to Load Balancer') {
        steps {
            script{
              sh 'kubectl set image deployment/surveyformlb container-0=pranaysharma793/surveyformcd:'+tag
            }
        }
    }


    }
}
```

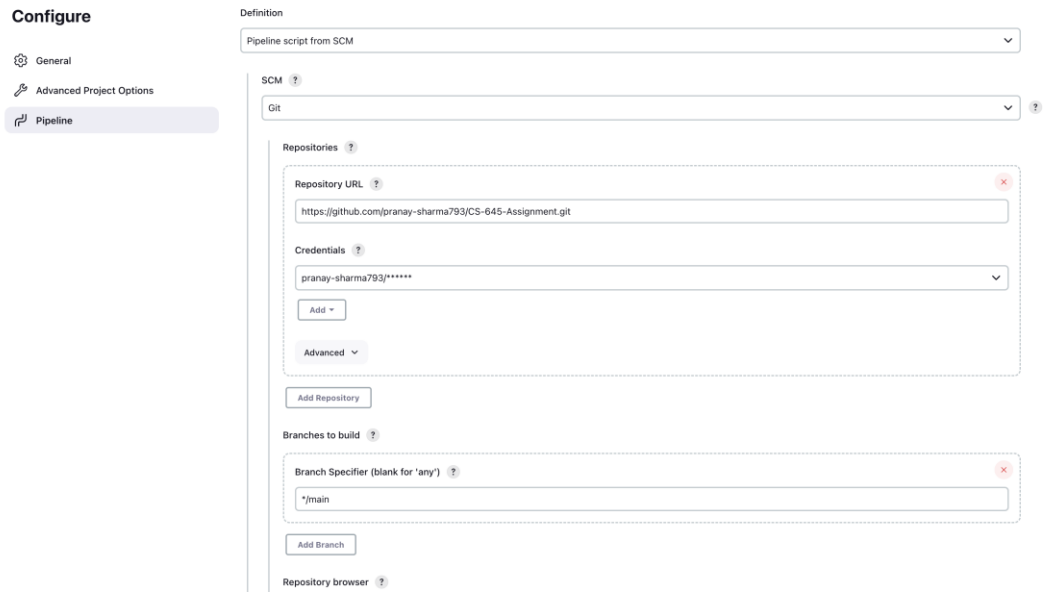
24. Next, to configure the pipeline,

    - Give your GitHub url in the Project url box.
    - Check the Poll SCM. Since we are polling it every one minute, add `* * * * *` in the schedule below. (This is make sure any updated code will be pulled frequently)



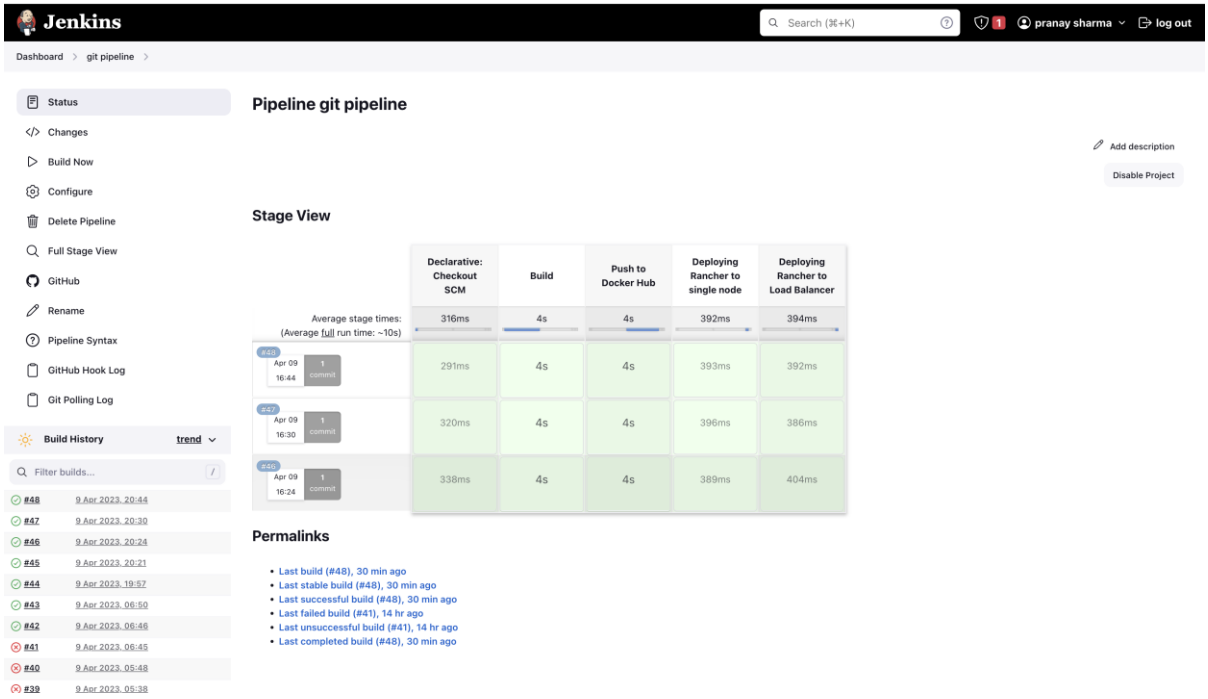
Continuing the configuration of the pipeline:

- For the Jenkins file, give a `Definition`, wherein we mention to take the file from our Git repository.
- Provide your GitHub url and your credentials, for it to access your Jenkinsfile.

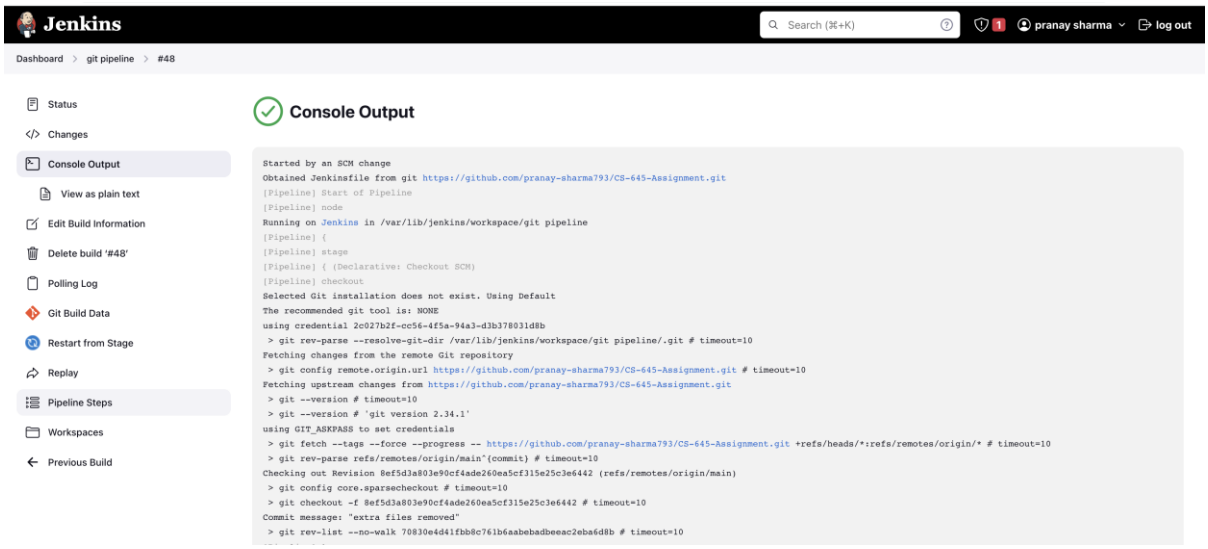


25. Next, come the Jenkins UI which shows the various stages of the Pipeline.

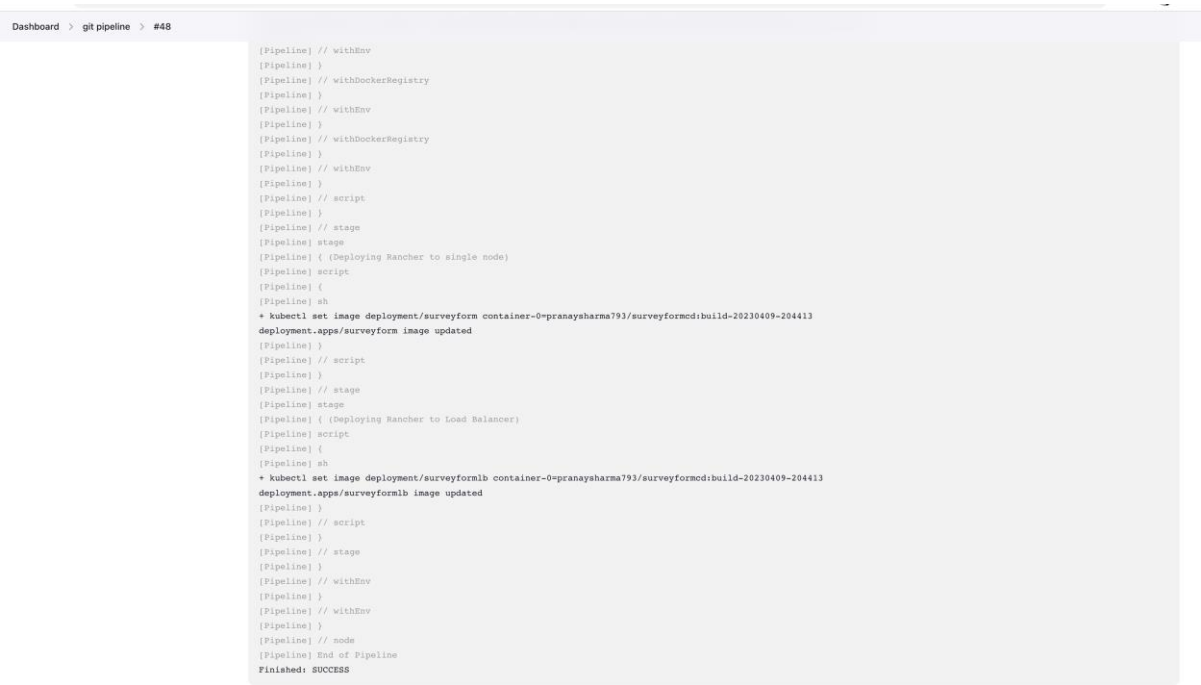
- Here to test the working of the pipeline, go to your code and make a change in your project and push it to your GitHub repository.
- Once the commit is done, we will see a new build being polled on Jenkins. This is because Jenkins is polling every minute to see if a change has been made. We see that a new build is created.



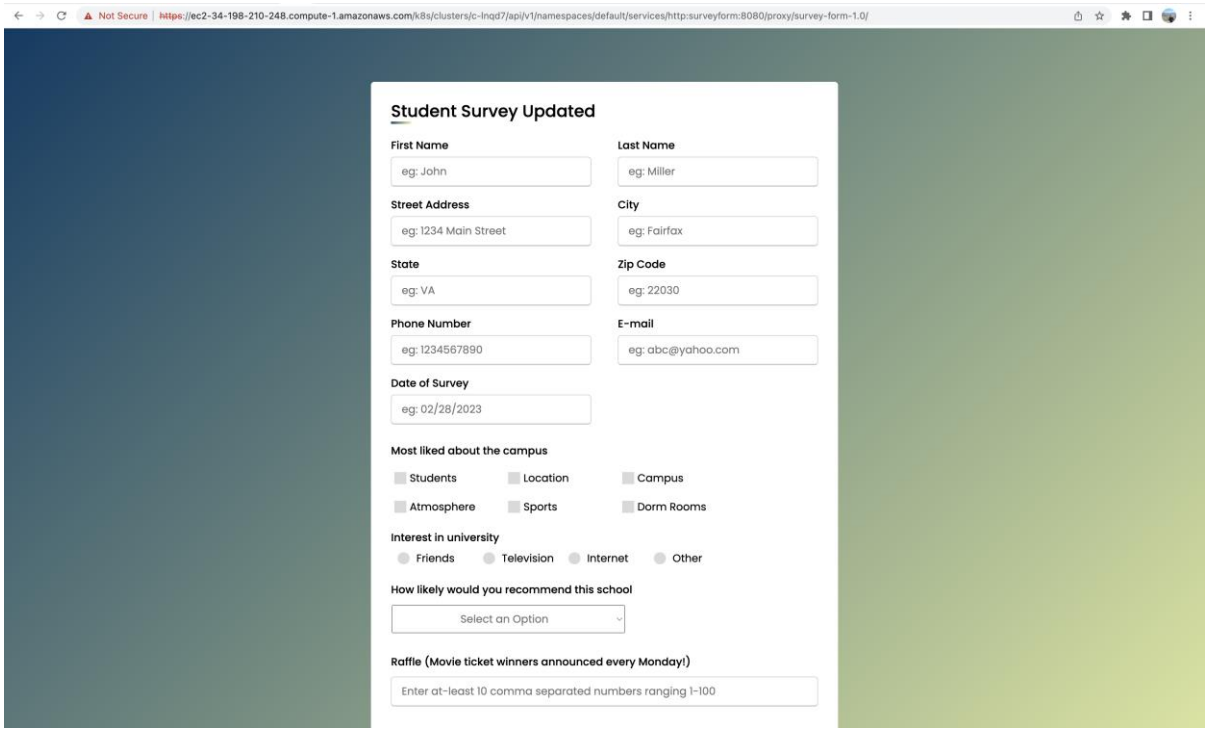
- Here we can see the logs once the build is done, where we see that our remote GitHub repository was accessed.



- We can also see in the logs below that it is deployed successfully on Rancher, wherein the image has gotten updated in both deployments namely `surveyform` and `surveyformlb`.
- Consequently check Rancher where you can see that an image has been updated, and check DockerHub to verify that a new build was pushed recently.



26. Now, check your survey-form url. You will see that the changes you have made have been reflected in the latest build. This demonstrate the complete working of our pipeline.



References:

Setting up git in Eclipse:

https://www.geo.uzh.ch/microsite/reproducible_research/post/rr-eclipse-git/

Building docker image from the war file:
https://aspetraining.com/resources/blog/deploying-your-first-web-app-to-tomcat-on-docker

To push docker image into the docker hub:
https://ropenscilabs.github.io/r-docker-tutorial/04-Dockerhub.html

Setting up Jenkins: https://pkg.jenkins.io/debian/

Pipeline in Jenkins: https://www.guru99.com/jenkins-pipeline-tutorial.html