README 645 HW2

Group Members:

Srini Jammula – G01426257
Tejeswar Sadanandan – G01451353
Aditya Gottipati – G01459535
Sruthi Sivasamy – G01453460

Website Endpoint Url- http:// 54.159.234.5:30007/demo1/

Github Repository Url- https://github.com/srinijammula/swe645hw2

Rancher Public IPv4 DNS- https:// ec2-3-218-63-246.compute-1.amazonaws.com/dashboard/
Credentials username- admin, password- University@123

Jenkins Public IPv4 DNS- http://ec2-54-159-234-5.compute-1.amazonaws.com:8080/

Credentials username- srinijammula, password- University@123

1. Create a war file

Install eclipse and create a project. Paste your form(index.html) in webapp. Create web.xml under web-inf and paste this code in it.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="https://jakarta.ee/xml/ns/jakartaee"
xsi:schemaLocation="https://jakarta.ee/xml/ns/jakartaee
https://jakarta.ee/xml/ns/jakartaee/web-app_6_0.xsd" id="WebApp_ID"
version="6.0">
 <display-name>SurveyForm</display-name>
 <welcome-file-list>
      <welcome-file>forms.html</welcome-file>
      <welcome-file>index.html</welcome-file>
      <welcome-file>index.jsp</welcome-file>
      <welcome-file>index.htm</welcome-file>
      <welcome-file>default.html</welcome-file>
      <welcome-file>default.jsp</welcome-file>
      <welcome-file>default.htm</welcome-file>
 </welcome-file-list>
</web-app>
```

Replace forms.html with your survey form name. (mine is index.html).
In file->export->war give name(demo1.war) and destination of where you want to place your war file like your project folder and create.

2. Create Dockerfile

Open your project folder in Visual Studio Code and create a file with name 'Dockerfile'. It will automatically give a docker symbol beside. Now paste this in your Dockerfile.

```
Dockerfile
Dockerfile > FROM
1  FROM tomcat:8.0-alpine
2
3  COPY ./demo1.war /usr/local/tomcat/webapps/
4
5  EXPOSE 8080
6
7  CMD ["catalina.sh", "run"]
```

This file is used to build your image from the base image tomcat and your war file.

3. Check your image in localhost

This step is to verify if you can build an image correctly on given war and Dockerfile. To see if your image is working in local host, install docker desktop and open it for the Docker Engine to run. Check if docker is installed with command 'docker --version'.
Now open cmd at your project folder and run below commands.
To build an image,

docker build -t image-name .

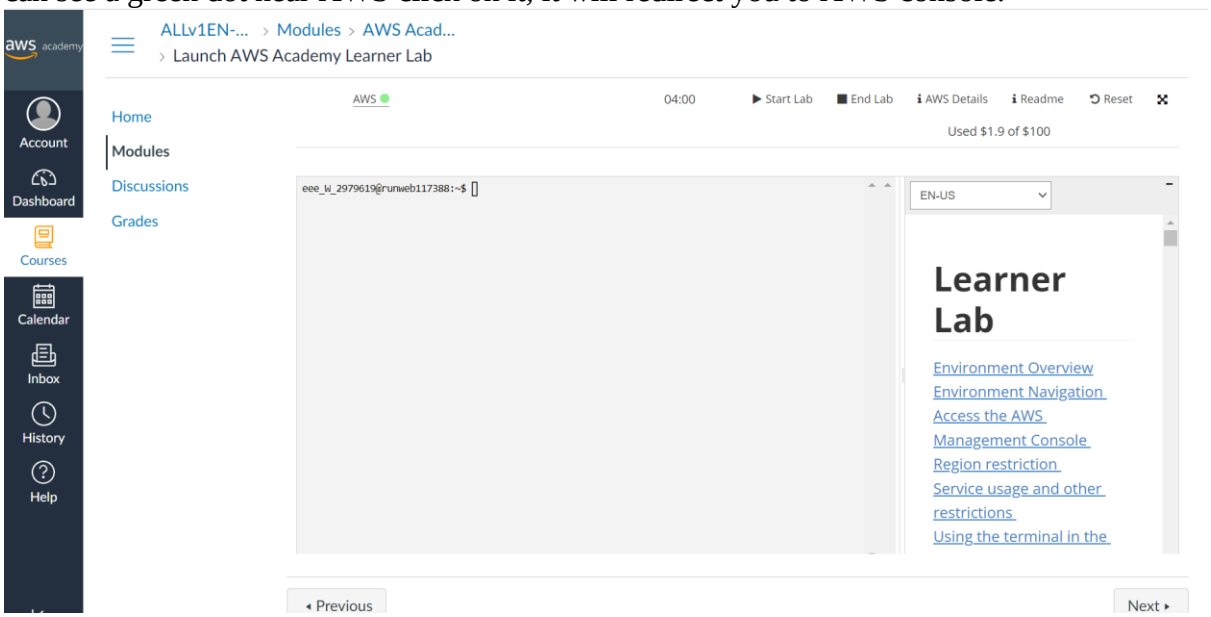
To create a container,

docker run -d -it -p 8080:8080 image-name

Check if you can see your form in http://localhost:8080/war-file-name

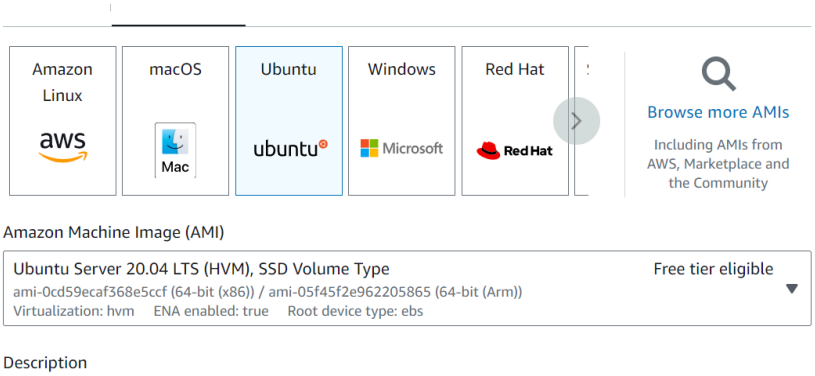
4. Create EC2 instances

Launch your AWS Lab. Got to modules and launch AWS Academy learner lab, start lab. After you can see a green dot near AWS click on it, it will redirect you to AWS console.

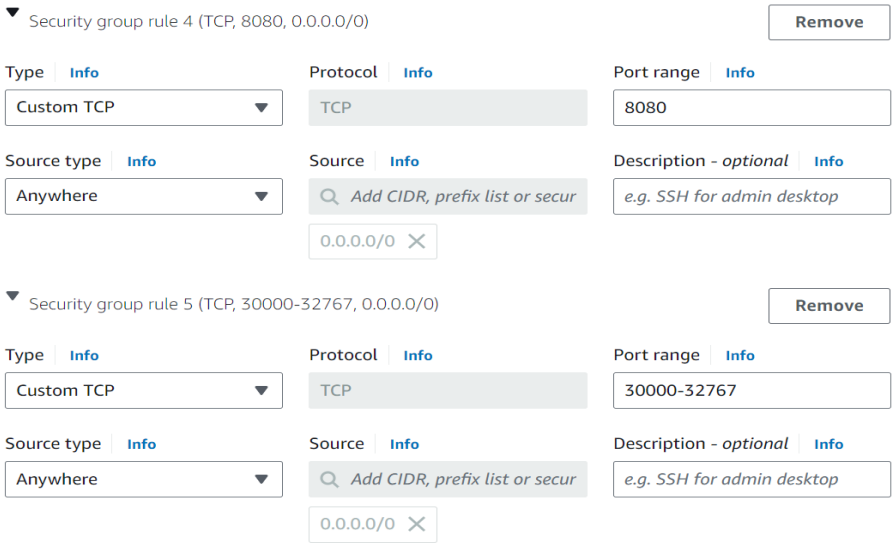


Search for EC2->Launch instance. Create three different instances with same configurations mentioned below. (demo1, clusterdemo1, jenkinsdemo1)

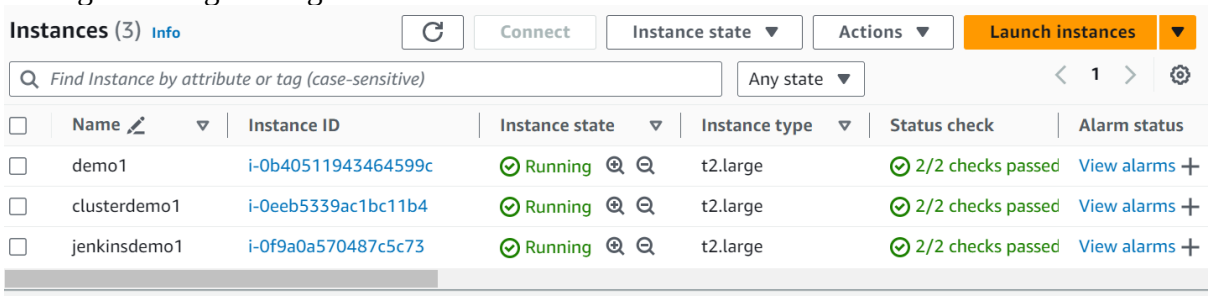
For AMI choose Ubuntu 20.04 LTS,



Instance type- t2.large
Create a new key-pair or use an existing one (I am using demo1.pem).
In network settings check on all allow and click on edit to add port 8080 and 30000-32767 (for nodeport) and put source type anywhere.



Configure storage to 30gb and Launch instance.



Search for elastic ips in ec2 and allocate each for each instance. Just select instance name and default settings. Give names to IP.

Name	Allocated IPv4 addr...	Type	Allocation ID
elastic1	3.218.63.246	Public IP	eipalloc-05afcc0e14d0b51f6
elastic2	52.45.191.29	Public IP	eipalloc-07d853181f3fcbf00
elastic3	54.159.234.5	Public IP	eipalloc-08fd7a3061811bb8a

Now go to demo1 instance and click on connect, run below commands.
Install docker,

sudo apt-get update
sudo apt install docker.io

Get your war and Dockerfile from your system to ec2 using gitbash.

scp -i /path/to/your-key.pem /path/to/your-file.war ec2-user@[your-ec2-instance-ip]:/path/on/ec2/where/to/put/file

Change your pem file permissions if it doesn't allow access,

chmod 400 ~/Downloads/your-key.pem

```
chmod 400 ~/Downloads/demo1.pem

scp -i ~/Downloads/demo1.pem ~/Downloads/645hw2/demo1.war ubuntu@ec2-54-236-56-11.compute-1.amazonaws.com:home/ubuntu/
```

Now check in your demo1 instance if the files are transferred using ls command.

Build an image and create a container, tag and push it to docker hub.

```
# Build the Docker image
sudo docker build -t your-image-name .

# Tag the Docker image for Docker Hub
sudo docker tag your-image-name your-dockerhub-username/your-repo-name:your-tag

# Log in to Docker Hub
sudo docker login

# Push the image to Docker Hub
sudo docker push your-dockerhub-username/your-repo-name:your-tag

# Run a container from your image, mapping port 8080 from the container to port 8080
sudo docker run -d -it -p 8080:8080 your-image-name
```

Check your image in public ipv4 dns address:8080/app

5. Creating cluster and deploying pods

Now execute below commands in demo1 to start rancher and login using public ipv4 dns address of demo1.

To start rancher,

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

To get the container-id of rancher image→*sudo docker ps*

Get the password to enter for first time using this and change it later,

docker logs your-container-Id 2>&1 | grep "Bootstrap Password:"

Paste it in rancher website opened using public ipv4 dns address of demo1.

Go to navigation bar-> Cluster management-> Custom-> Create-> give a name-> create.

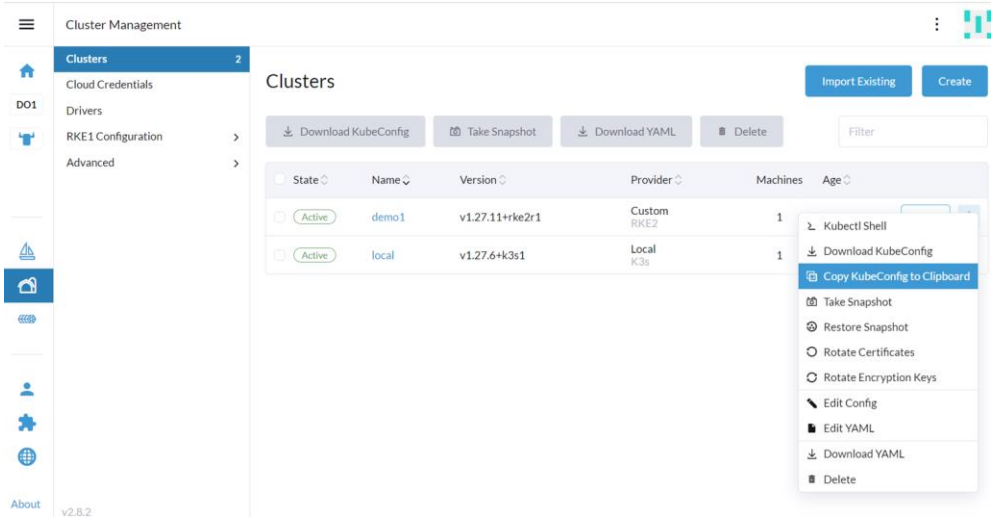
Copy the curl command under registration after checking insecure.

Now to clusterdemo1 and connect. Paste the command and run.

Install kubectl,

snap install kubectl –classic

Copy the kubeconfig to your clipboard from you cluster



Create a .kube directory and paste it in config.

mkdir .kube

nano config

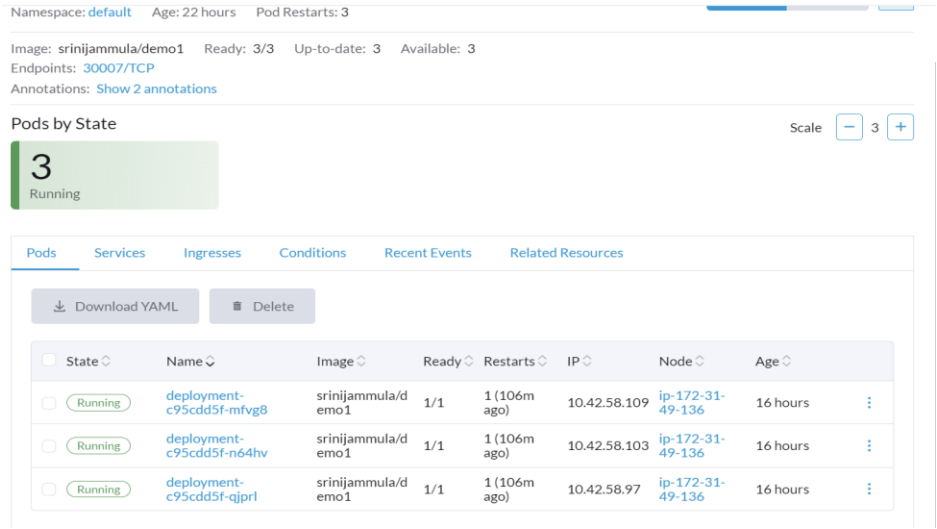
Ctrl+O to save, enter and Ctrl+X to exit.

Come out of your .kube directory and create deployment.yaml and service.yaml using nano command, customize them accordingly.

```
! deployment.yaml M ×
! deployment.yaml > {} spec > {} template > {} spec > [ ] containers > {}
      io.k8s.api.apps.v1.Deployment (v1@deployment.json)
1   apiVersion: apps/v1
2   kind: Deployment
3   metadata:
4     name: deployment
5   spec:
6     replicas: 3
7     selector:
8       matchLabels:
9         app: demo1
10    template:
11      metadata:
12        labels:
13          app: demo1
14      spec:
15        containers:
16        - name: demo1
17          image: srinijammula/demo1
18          ports:
19          - containerPort: 8080
```

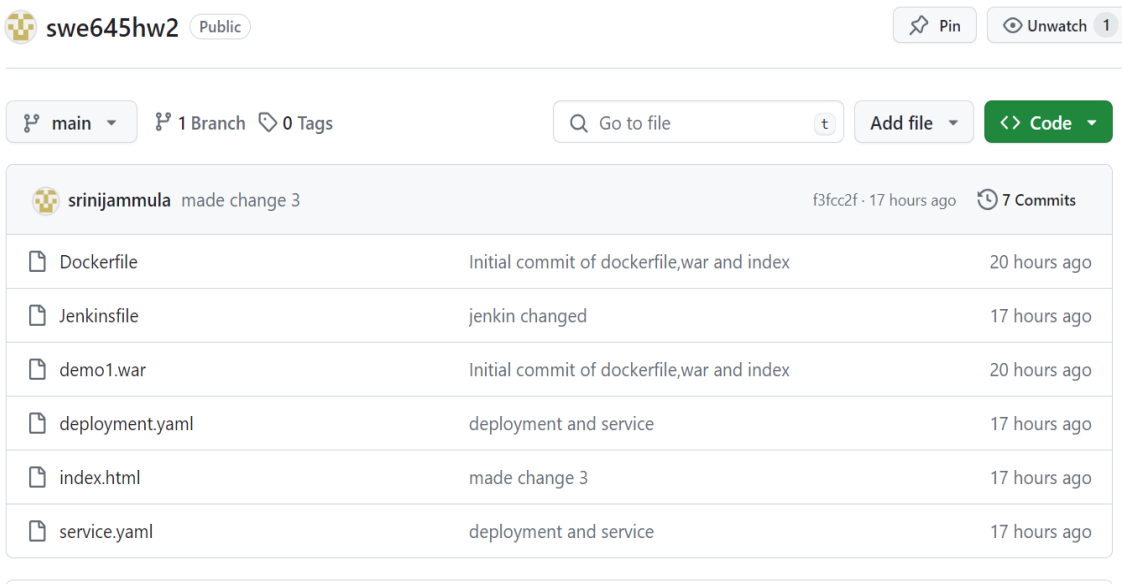
```yaml
! deployment.yaml M ×   ! service.yaml ×
! service.yaml > {} spec > [ ] ports > {} 0
        io.k8s.api.core.v1.Service (v1@service.json)
apiVersion: v1
kind: Service
metadata:
  name: service
spec:
  type: NodePort
  selector:
    app: demo1
  ports:
    - protocol: TCP
      port: 8080
      targetPort: 8080
      nodePort: 30007
```

Apply both using *kubectl apply -f deployment.yaml* and *kubectl apply -f service.yaml*. Check your pods are running using *kubectl get pods*, you can also see them in rancher->worknodes->deployment.
Now check if you can see the form from,
http://publicipv4_addressof_clusterinstance:nodeport/containername



6. Github repository

Create a github account if you don't have one. Click on create new repository and give a name like swe645hw2 and click on create.
Connect from your local system to git repository and push your index.html, war file, Dockerfile, Jenkins file (create one), deployment.yaml and service.yaml to your repository.



7. Setup Jenkins

Open jenkinsdemo1 instance and execute these commands.

To update and install jdk,

sudo apt-get update

sudo apt install openjdk-11-jdk

Add Jenkins repository,

curl -fsSL https://pkg.jenkins.io/debian-stable/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-stable binary/" | sudo tee /etc/apt/sources.list.d/jenkins.list > /dev/null

Update and install Jenkins,

sudo apt update

sudo apt install jenkins

Start Jenkins and check its status,

sudo systemctl start jenkins

sudo systemctl status Jenkins

Install docker to run containers,

sudo apt-get update

sudo apt-get install docker.io

sudo systemctl status docker

Add Jenkins User,

sudo usermod -a -G docker Jenkins

Install kubectl,

sudo snap install kubectl –classic

Configure Kubernetes Access,

cd /var/lib/jenkins/.kube

sudo chmod -R u+w /var/lib/jenkins/.kube\

Create Config file in .kube and paste kubeconfig contents from cluster you created in rancher,

sudo nano config

Verify Jenkins Status,

sudo systemctl status jenkins

Access Jenkins Web Interface,

http://54.159.234.5:8080

Initial Admin password to unlock Jenkins and create account

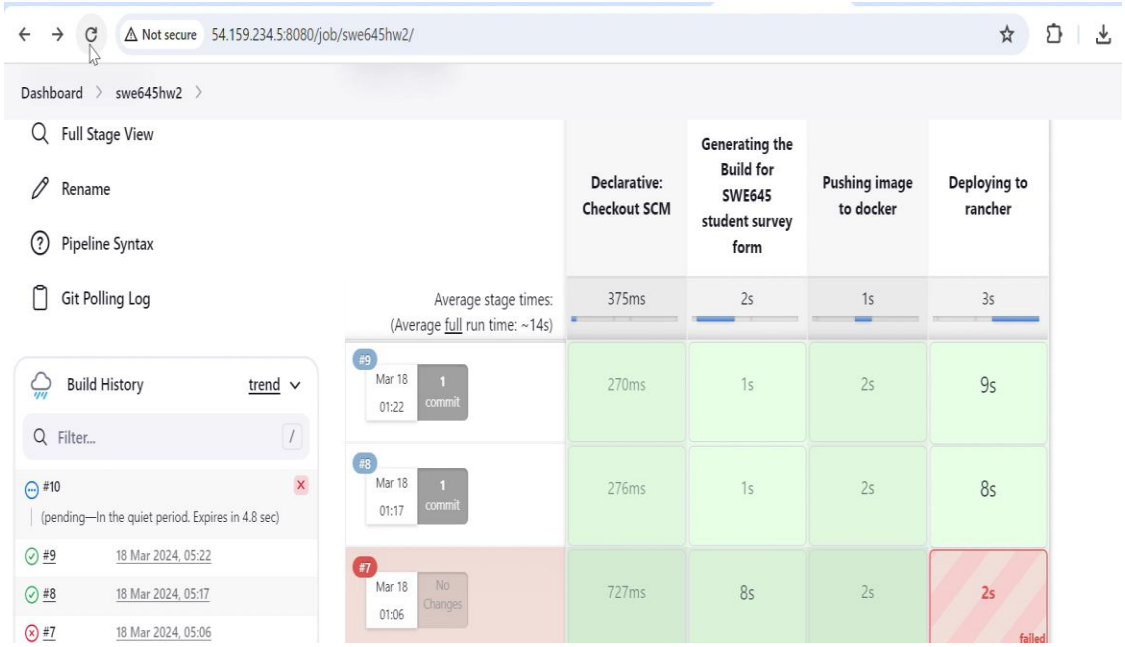
sudo cat /var/lib/jenkins/secrets/initialAdminPassword

Create a Jenkins Pipeline following these steps,

- Select "+ New Item", choose “Pipeline”, and give it a name.
- Under "Build Trigger", select "Poll SCM" and enter *** * * * *** (runs every minute).

- Under "Pipeline", choose "Pipeline script from SCM", select Git as SCM, provide your repository URL and branch, and set the Jenkinsfile path.
- Save the pipeline configuration.

Check the build of Jenkins if everything is deployed correctly. If you get any errors refer the console at the bottom (click on failed build) to know the exact error.



CONTRIBUTIONS

- Step 1,2 and 3 were contributed by Sruthi Sivasamy
- Step 4 was contributed by Aditya Gottipati
- Step 5 and 6 were contributed by Srini Jammula
- Step 7 was contributed by Tejeswar Sadanandan

REFERENCES

- https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html
- https://www.baeldung.com/linux/jenkins-install-run
- https://kubernetes.io/docs/tasks/tools/install-kubectl-linux
- https://www.jenkins.io/doc/book/pipeline/jenkinsfile/
- https://www.jenkins.io/blog/2023/03/27/repository-signing-keys-changing/