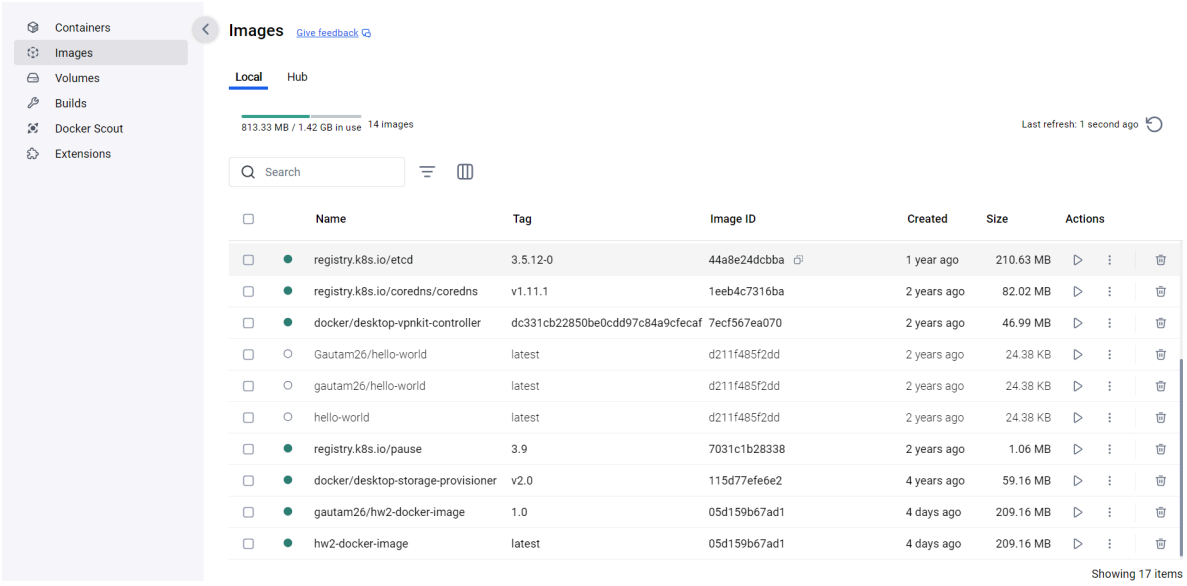
SWE 645 ASSIGNMENT 2

Group Name: Momtrimo
Name: Satya Subrahmanya Gautama Shastry Bulusu Venkata(G01477340)
Name: Maheedhar Sai Omtri Mohan(G01478890)

Objective 1:
Containerize the web application from Homework 1- Part 2 (survey page) using Docker Container technology

Completed by: Maheedhar Sai Omtri Mohan
Steps to Create the docker image:

1. First, register at hub.docker.com, log in, and download Docker Desktop. You can verify its installation by using the command: docker -version
2. On the project structure, you already have the index.html file; now, create the Dockerfile file. The index.html is the survey page from Homework 1. We use that to make a docker image using the command: **docker build -t hw2-docker-image .**
3. You can verify if the images are created using the command: **docker images**
4. Tag and push the image to the Docker registry:
   docker tag hw2-docker-image gautam26/hw2-docker-image:1.0
   docker push gautam26/hw2-docker-image:1.0
5. Below is the screenshot of the docker image registry. It confirms our docker image is in the registry

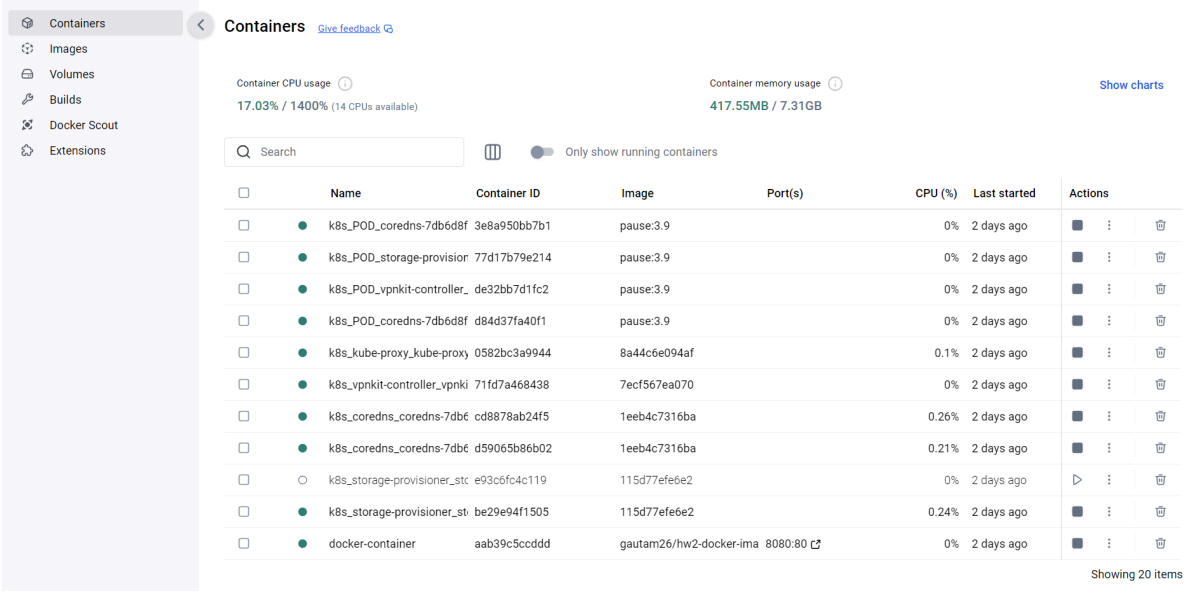


6. Below is the command written in the Dockerfile

```
FROM --platform=linux/amd64 ubuntu:latest
RUN apt-get update
RUN apt-get install -y nginx
```

```
RUN rm /var/www/html/index.nginx-debian.html
COPY index.html /var/www/html/
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

7. Create and upload the docker image into a Docker Container using the following command: **docker run -d -p –name docker-container gautam26/hw2-docker-image:1.0**
8. Verify whether the container is created in Docker Desktop



Objective 2:
Deploy the containerized application on kubernetes to achieve scalability and resiliency. I used Rancher to create Kubernetes Cluster.

Completed by: Satya Subrahmanya Gautama Shastry Bulusu Venkata
Steps to perform this task:

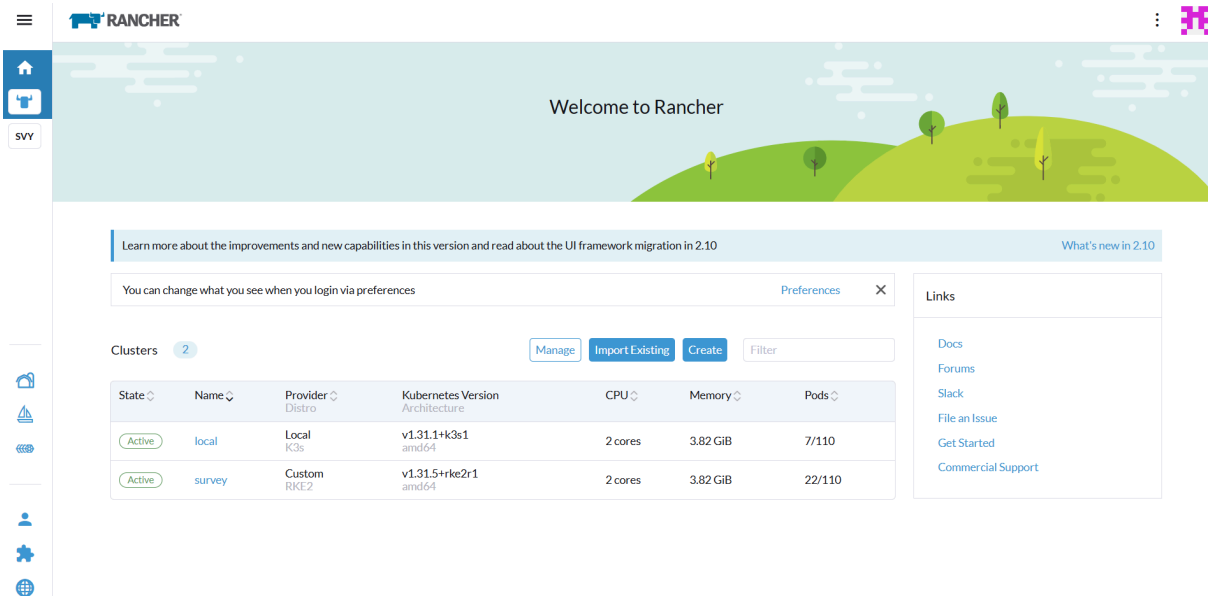
1. Before building a Kubernetes cluster, We need to login to AWS Learner's Lab and open the AWS Management Console. We create 3 ec2 instances named instance1, instance2, instance 3.
2. Create the instances using the following configuration:
    a. Use Ubuntu Server 22.04 AMI LTS(HVM), SSD Volume Type.
    b. Security groups allowing inbound traffic on ports 8080, 80, 443 and 22 from anywhere(0.0.0.0)
    c. Assign 16 GB storage to each instance.
    d. Create a key (**survey_key.pem),** and use the same key for all the three instances.
3. Allocate Elastic IPs for each instance.
4. Open git bash and connect to instances 1 and 2 using the command: **ssh -i survey_key.pem ubuntu@<IP address of instance>**

Then we follow the command below for updating the packages in the instances and install docker in both instance 1 and instance 2.

   a. **sudo su -** (for switching to root user)
   b. **sudo apt-get update**
   c. **sudo apt install docker.io**

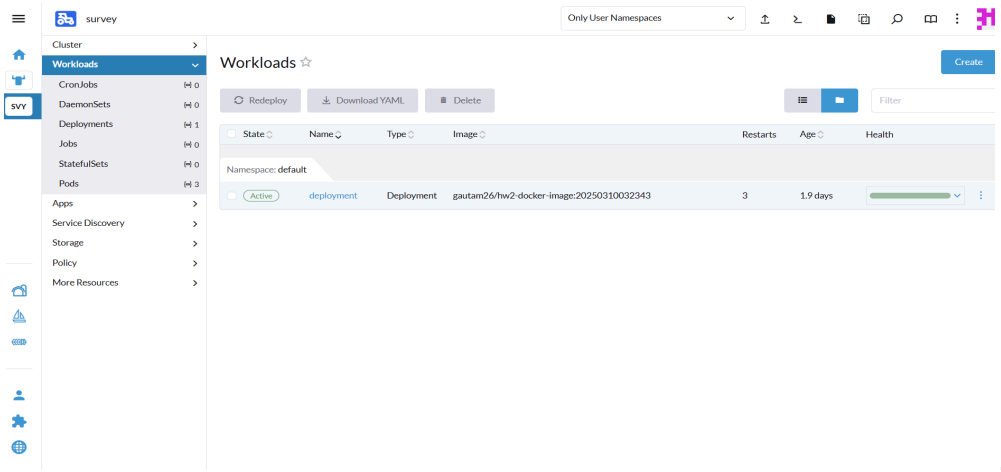
5. Next we start the Rancher server on instance. We just follow the below commands and that should start the server.
   a. **sudo docker run –privileged -d –restart=unless-stopped -p 80:80 -p 443:443 rancher/rancher**
   b. Now run **sudo docker ps** to view the current docker instance information and see the container Id. It will be useful for us.
   c. The container Id must be used in the below command: **sudo docker logs container-ID 2>&1 | grep “Bootstrap Password:”**. We get the bootstrap password which will be useful for first time login in Rancher UI. Racher UI will be opened by the link: **http://<Public IPv4 address of instance 1>**
   d. Use the bootstrap password to login, and then set a specific password for your future purposes.
6. Let’s create a Kubernetes Cluster by clicking the create buttonand select custom under “Use existing nodes and create and cluster using RKE”. Give a name to the cluster and then click create. In next page click the three checkboxes for etcd, Control Plane, and Worker and apply the checkbox Insecure: select this to skip TLS Verification if your server has self-signed certificate. This would give us a command to paste in instance 2 connect console: **curl --insecure -fL https://ec2-44-197-122-170.compute-1.amazonaws.com/system-agent-install.sh | sudo sh -s - --server https://ec2-44-197-122-170.compute-1.amazonaws.com --label 'cattle.io/os=linux' --token 5zcl9s4c2zcwv6t7fldj4dj75wl6zwmzzsbd54jbzgdcgnw5nnz7wl --ca-checksum 2012e7b3fe46f2774a434ec483c6701a26a1dfa9791bff1a291fb3b4fcf23aba --etcd --controlplane --worker**

   Wait till the cluster state changes from “updating” to “active”. Once the state is active the cluster is ready to deploy



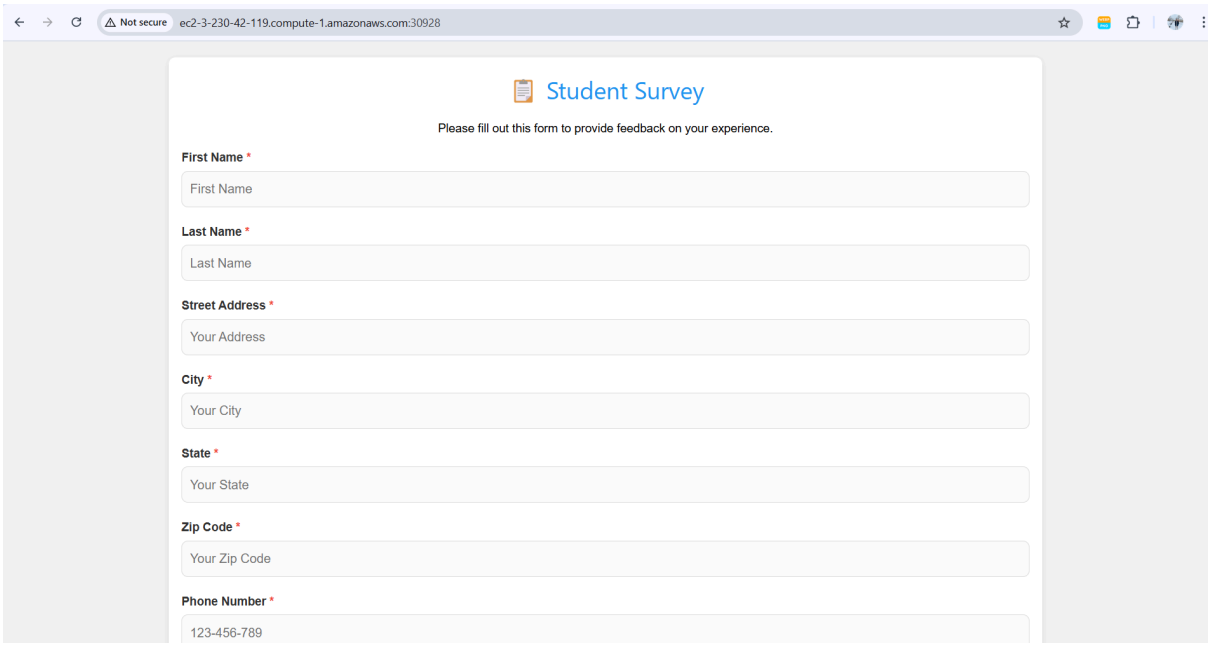
Download the KubeConfig file for future use. It's named as **survey.yaml**

7. Create the deployment object using Docker container image:
    a. From the left side menu you can see "SVY", Its the cluster name. Click it and then open the "Workloads" section and select "Deployments".
    b. Initially it is empty as there are no deployments. Click "Create" button and fill the following fields: Give a name for the deployment object. I gave "**deployment**". Increase the count in replicas field from "1" to "3", as we want to run three pods all the time. Fill the container field(give some name, I gave same name as that of docker container). Fill the container image field(same as docker image). Click Add Port or Service. Give NodePort in Service Type and Port number 80(same as the port given in Dockerfile). Give some name like **nodeport** for this service. Click Create button.
    c. You have successfully created a deployment object, wait until it's state changes from "updating" to "active"



8. If we go to Service Discovery, We can see the NodePort number. Mine is “30928”.

Access the Application through:
http://<IPv4 of Instance 2>:<NodePort>/
Mine is **http://ec2-3-230-42-119.compute-1.amazonaws.com:30928/**



Objective 3:
Use GitHub as source code repository and set up a CI/CD Pipeline using Jenkins for automated build and deployment of the application on Kubernetes.

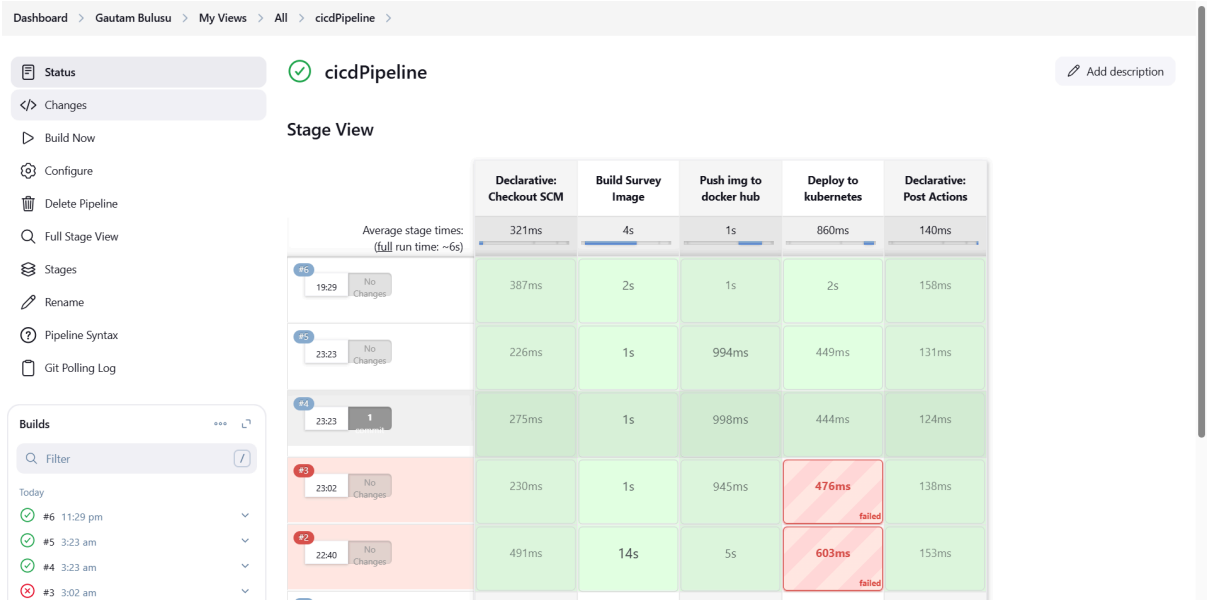
Completed by: Satya Subrahmanya Gautama Shastry Bulusu Venkata & Maheedhar Sai Omtri Mohan

Steps Performed in this task:

1. Connect to the instance 3 through: **ssh -i survey_key.pem ubuntu@<IP of instance 3>.** And change to root user: **sudo su -**
2. Install all the necessary packages like Docker, Jenkins and JDK:
   Just follow this commands: **sudo apt-get update** (Update system packages)**, sudo apt-get install -y docker.io** (Install Docker)**, sudo apt install openjdk-17-jre-headless** (Install JDK)**, wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add -** (Add Jenkins repository key)**, sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'**(Add Jenkins repository).
3. You installed all the necessary packages. Now try to start the jenkins service.

sudo systemctl start jenkins. We tried to start the service after installing JDK version 11 initially but failed. So we installed JDK version 17. You can check the current version by **java –version**

4. Verify whether the jenkins server is running **sudo systemctl status jenkins**
5. Access the Jenkins UI through: **http://<IPv4 instance 3>:8080**
6. Retrive the initial jenkins password in **sudo cat /var/lib/jenkins/secret/intialAdminPassword** You can use the password to login into the Jenkins UI. We changed it to different password so that we can login again.
7. Install the required plugins like Git, Docker Pipeline, and Kubernetes CLI. Go to **Manage Jenkins -> Plugins**
8. Add GitHub and Docker credentials in **Manage Jenkins -> System -> global -> Add Credentials.** Use Username and Password and add them as credentials.
9. Add Jenkins user to the Docker group to allow Jenkins to run Docker commands: **sudo usermod -aG docker jenkins.** Restart jenkins to apply any changes: **sudo systemctl restart jenkins**
10. Create a CI/CD pipeline in Jenkins. Go to **New Item.** Provide a name for the pipeline “cicdPipeline” and Select “Pipeline”. In “Build Triggers”, select “Poll SCM” and set schedule to “* * * * *” for every minute. Under “Pipeline”, select “Pipeline script from SCM” and chose “Git” as the SCM. Provide the Git repository URL and select the saved credentials. Specify the “/main” branch and the script file as “Jenkinsfile”. Click **Save.**
11. Create a “Jenkinsfile” on your project directory and write the code for your pipeline.
12. Test your pipeline by committing to the GitHub repo to trigger the build.



As you can see in Build #4, it automatically build and deployed the application to Kubernetes after committing changes to github repo.

AWS URL of Homepage and Deployed Application on Kubernetes:
http://ec2-3-230-42-119.compute-1.amazonaws.com:30928/
GitHub repo:
https://github.com/GautamaShastry/SWE645_assign2