SWE 645 - Project 2
Installation and Setup instructions

Team Members:

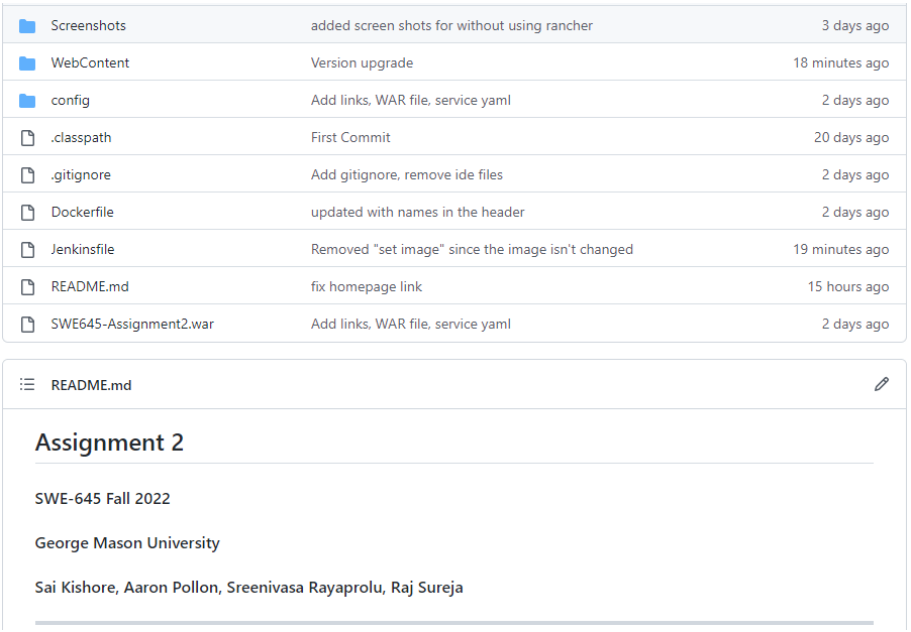
- Aaron Pollon (apollon@gmu.edu)
- Raj Sureja (rsureja@gmu.edu)
- Sai Kishore Salaka (ssalaka@gmu.edu)
- Sreenivasa Rayaprolu (srayapr@gmu.edu)

Prerequisites:

1. Docker Desktop installed on local machine to build the initial docker image
2. Accounts with GitHub, Google, Docker Hub
3. AWS running on Learner Lab

Project Steps:

1. Create a GitHub repository for the source code. The WebContent folder would contain the webpages, styles, scripts, configuration, etc. required for the application. The Dockerfile and Jenkinsfile are added to the root directory.



2. Setup the initial docker image on Docker Hub:
   2.1. Using an IDE or the command "`jar -cvf SWE645-Assignment2.war -C WebContent/ .`" generate a WAR file in the root directory
   2.2. Create a Dockerfile that installs Tomcat in the container, and copies the generated WAR file into the `tomcat/webapps` directory

```
7 lines (5 sloc)  263 Bytes
1  # Dockerfile for Student Survey web application.
2  # Uses Tomcat base image and places WAR into webapps directory.
3  # Authors: Sai Kishore, Aaron Pollon, Sreenivasa Rayaprolu, Raj Sureja
4
5  FROM tomcat:9.0-jdk15
6
7  COPY SWE645-Assignment2.war /usr/local/tomcat/webapps/
```

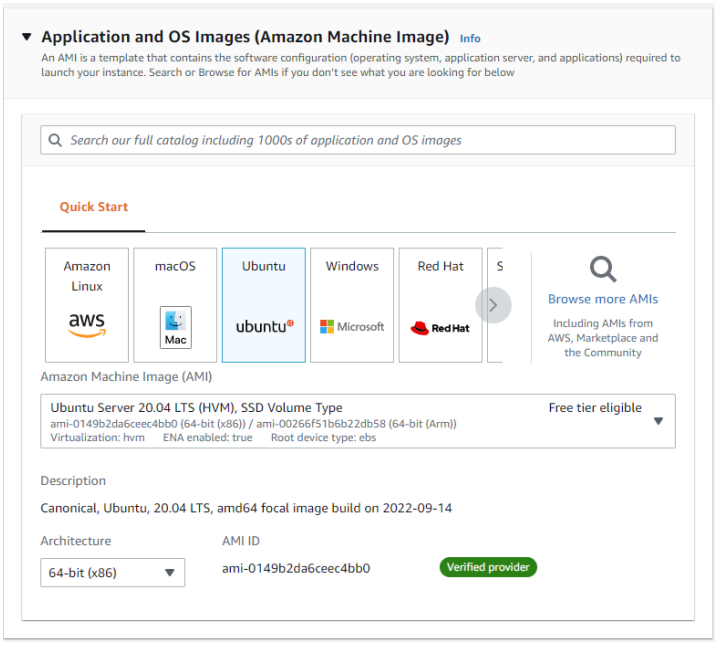
2.3. Build the docker image locally using the command "`docker build -t swe645-assignment2 . --progress=plain`"

2.4. You may optionally test the image by running it and verifying that the application works using the command "`docker run -itd -p 8080:8080 --name swe645-assignment2 swe645-assignment2`"

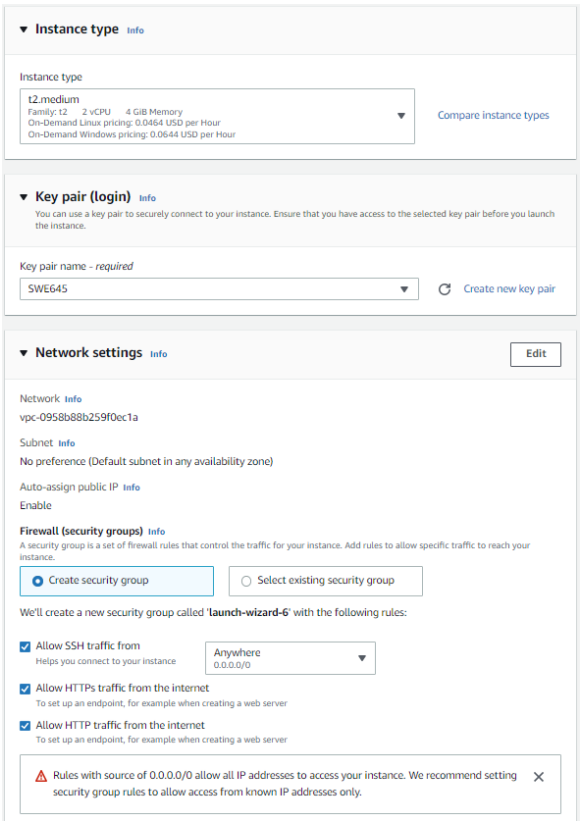
2.5. Then, tag the image and push it to Docker Hub (need to be logged in to Docker Hub through the terminal). In our case, the command was "`docker tag swe645-assignment2 coder0112358/swe645-assignment2`"

3. Install Rancher on an EC2 instance:

3.1. Navigate to EC2 on Learner Lab and create a new EC2 instance based on the "Ubuntu Server 20.04" AMI

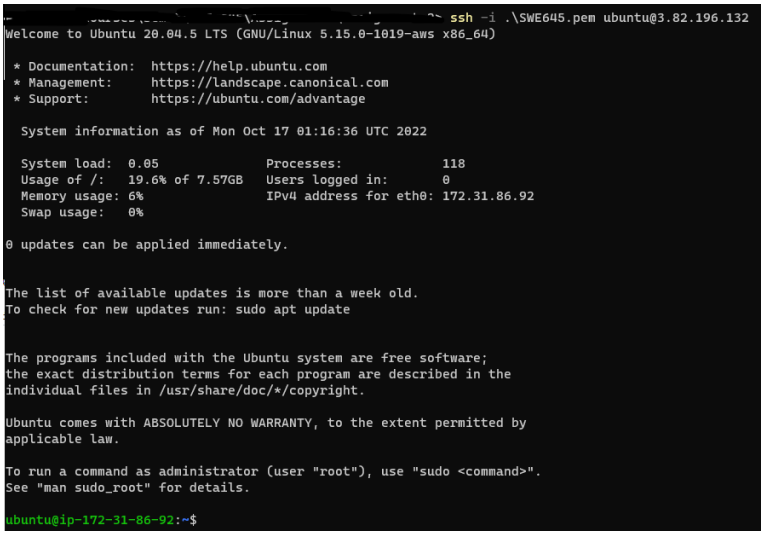


3.2. Set instance type as "t2.medium", create or select a Key pair for authentication, and allow traffic to SSH, HTTP and HTTPS, and launch the instance



3.3. Create an ElasticIP for Rancher EC2 instance, to ensure a static IP is assigned to the instance (ease of access when the Learner Lab shuts down automatically). This IP should be used everywhere instead of the EC2's IP

3.4. Once the instance is running, SSH into the instance with the pem/ppk key using the command "`ssh -i key_path ubuntu@elastic_ip`". Note that the key used for authentication should be accessible only by the administrator, and when connecting to the EC2 instances, the terminal needs to be opened as an admin



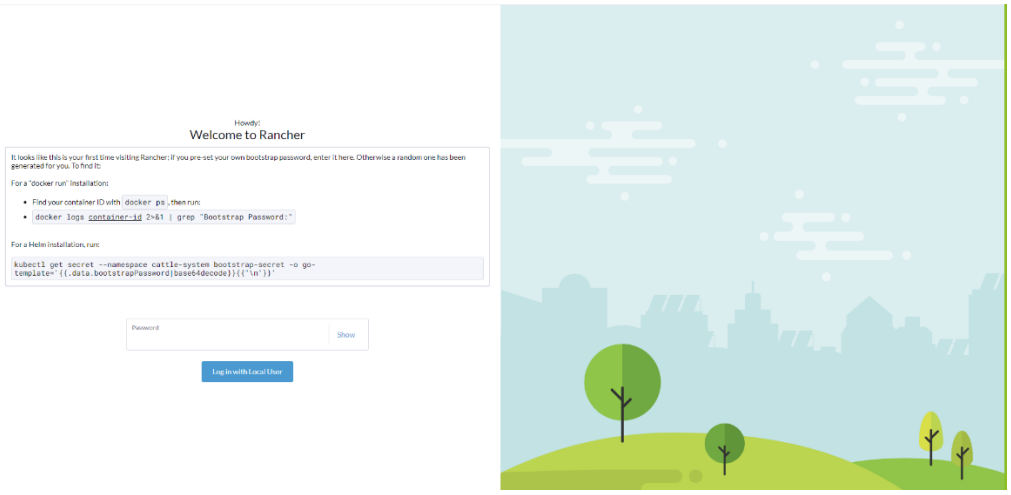
3.5. Update the package information using the command "`sudo apt-get update`" and install docker using "`sudo apt install docker.io`"

3.6. After installation, verify that docker is successfully installed using "`sudo docker -v`"

3.7. Note that any docker commands should be used with sudo, and don't add the ubuntu user to the docker group (to use docker without sudo) as this caused an issue where Rancher didn't start

3.8. Rancher will be installed and ran via a docker image using the command "`sudo docker run --privileged=true -d --restart=unless-stopped -p 80:80 -p 443:443 rancher/rancher`"

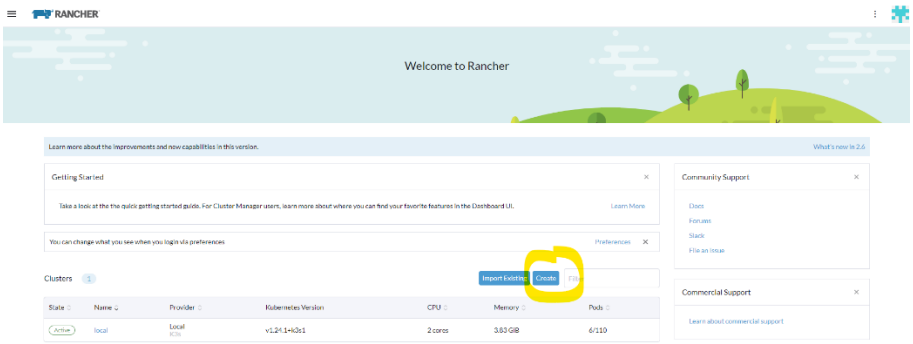
3.9. This would take some time to download and run Rancher. After a while, access the public IP through a browser and the following webpage should be seen



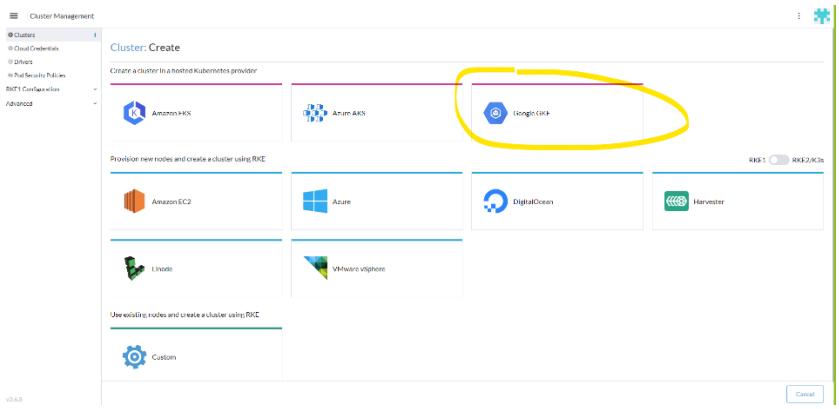
3.10. Follow the instructions in the welcome page to setup Rancher credentials (used docker with sudo)

4. Create a cluster in GKE using Rancher:

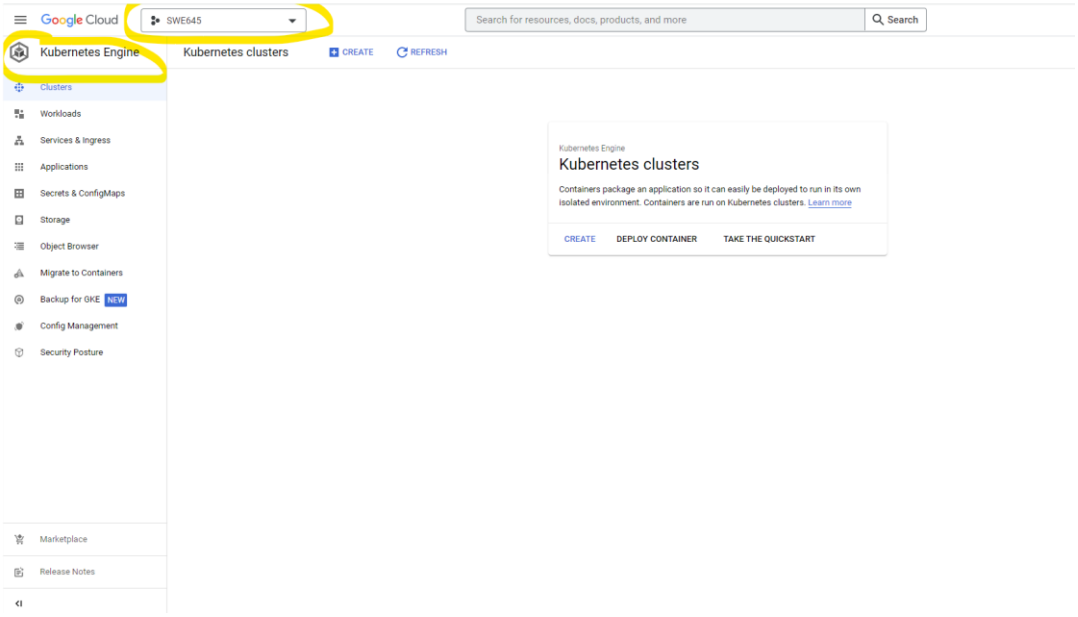
4.1. Login to Rancher, and there should be the "local" cluster already available. Click the "Create" option to create a new cluster



4.2. Choose Google GKE and in the next page give a name to the cluster

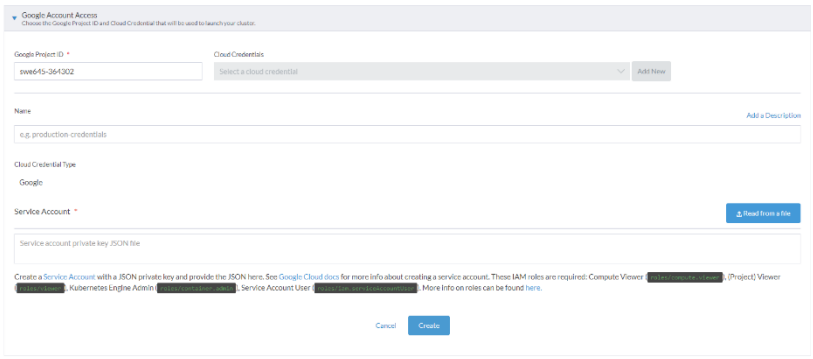


4.3. Login to [Google Cloud Platform](), create a new project, search for Kubernetes Engine and enable it for that project. You should end up with the page below with no clusters

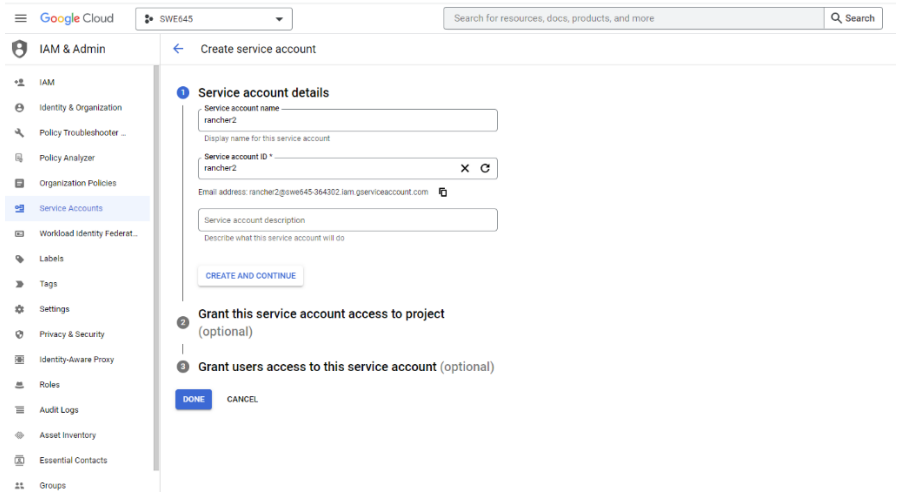


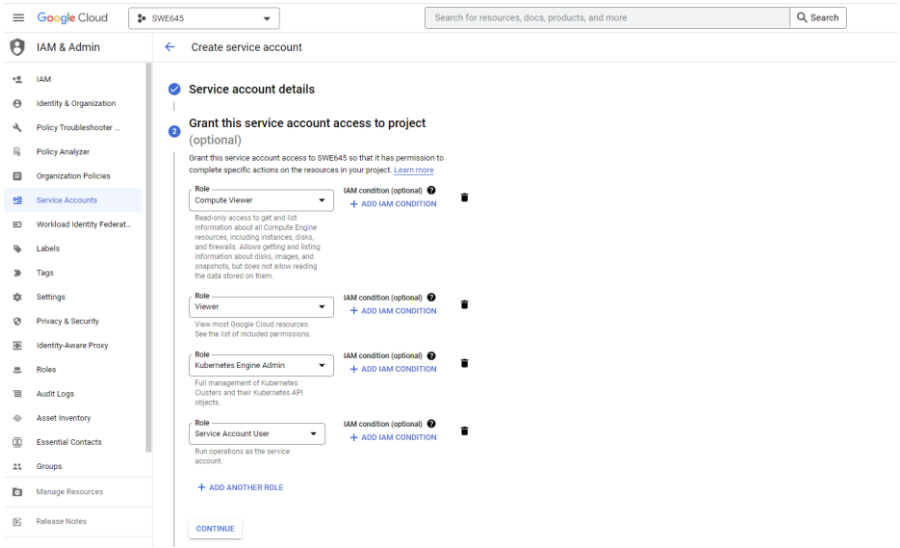
4.4. Retrieve the Google project ID from GCP either from the list of all projects in the top nav or from the project's home page

4.5. Provide the project ID in Rancher, and use the hyperlink provided to create a Service Account

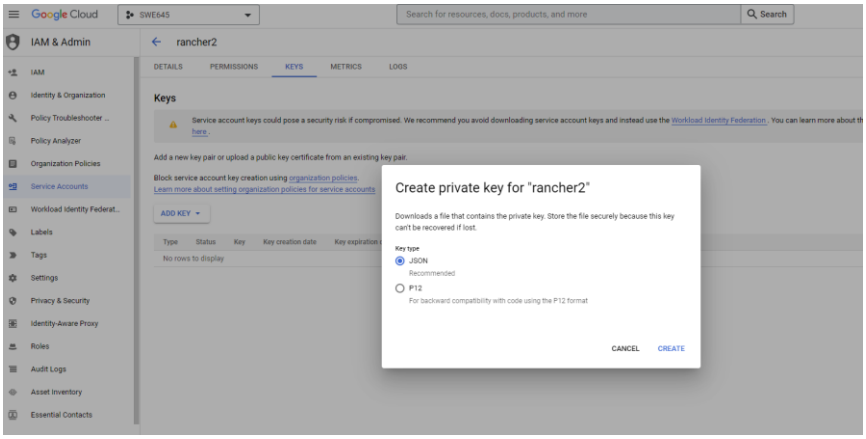


4.6. Provide a name for the service account, add the 4 required roles as specified by Rancher, and click Done



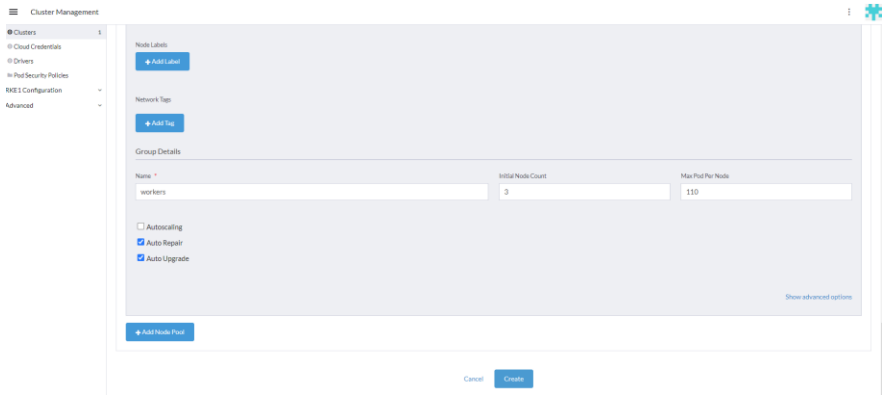


4.7. Once the service account is created, use the Actions menu to “Manage Keys” for that account, choose “Add Key”, and create a JSON key

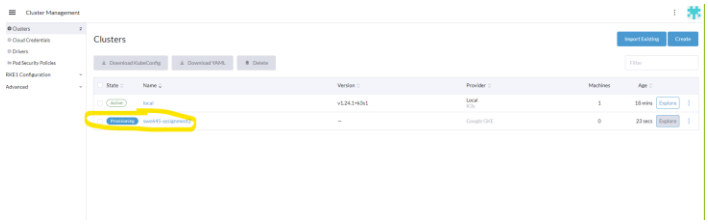


4.8. Navigate back to Rancher, select “Read from a file” (seen in 4.5), and upload the downloaded key

4.9. Under “Node Pools”, give a name to the work node group (e.g. “workers”) and set the “Initial Node Count” to 3. Note that the name *must* be in lowercase

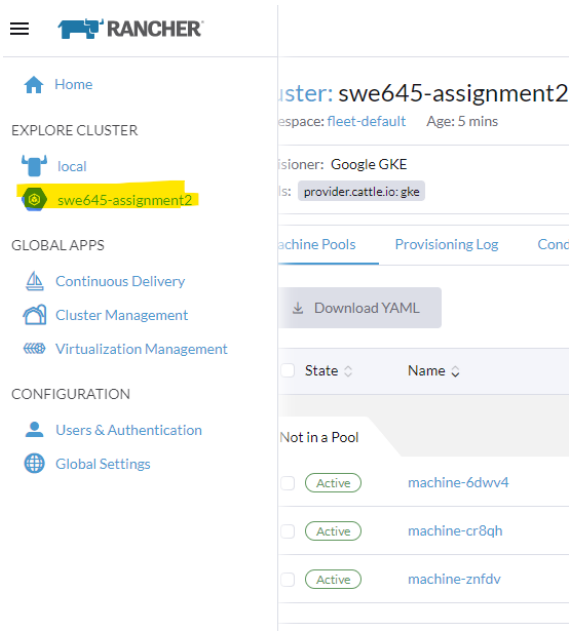


4.10. Leave the remaining settings to default and create the cluster. This usually takes 5-10 minutes to provision. In case of errors open the cluster in Rancher to view the error or check the logs in GCP

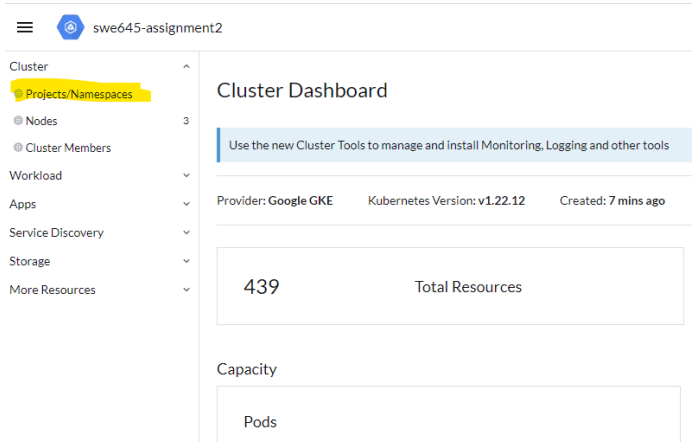


5. Create a Deployment in the Cluster:

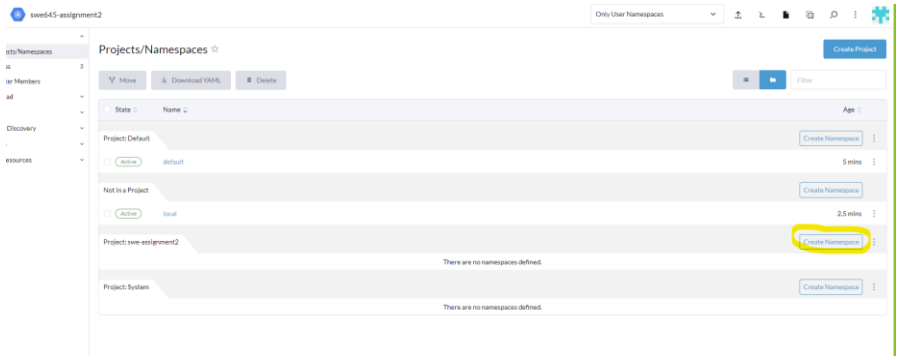
5.1. Select the created cluster from the “hamburger” menu



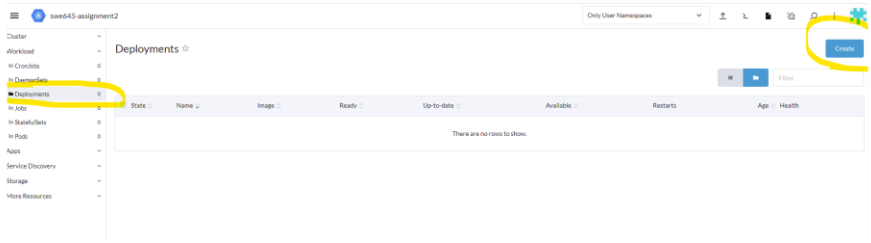
5.2. Select ”Projects/Namespace” from left-side nav



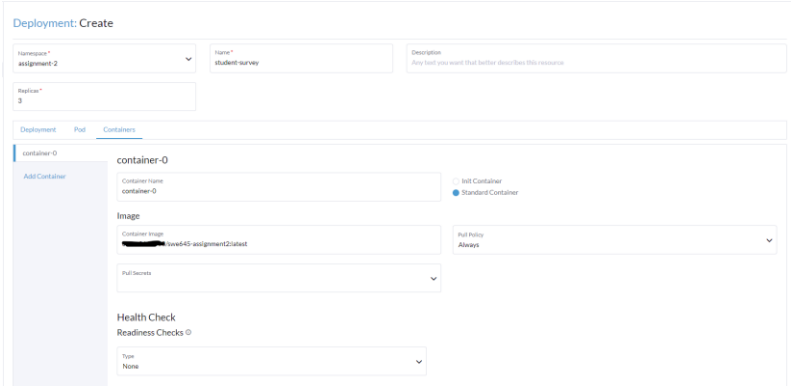
5.3. Create a new project and then create a namespace under this project



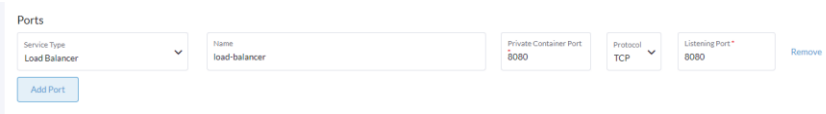
5.4. Select “Deployments” in the left-side nav and create a new one



5.5. Choose the previously created namespace, provide a name, and set the replicas to 3. Leave the “Deployment” and “Pod” configuration to the default, and in the “Containers” set the image to the one in Docker Hub

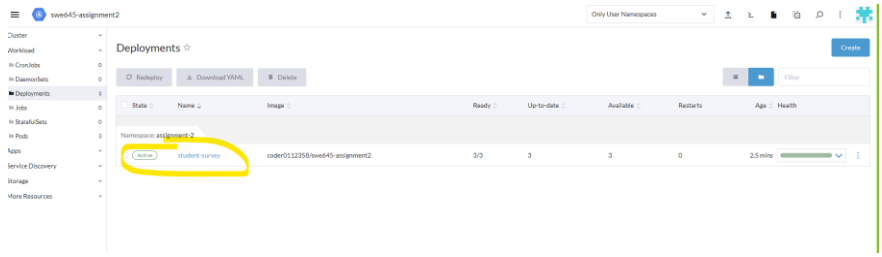


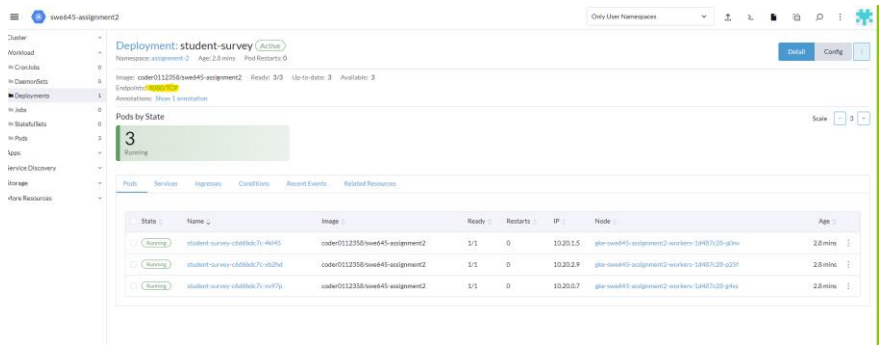
5.6. Scroll down to “Ports” and add a new one. Select “Load Balancer” as the service type, and expose the 8080 port



5.7. Set everything else to default and create the deployment. It usually takes 2-5 mins to create

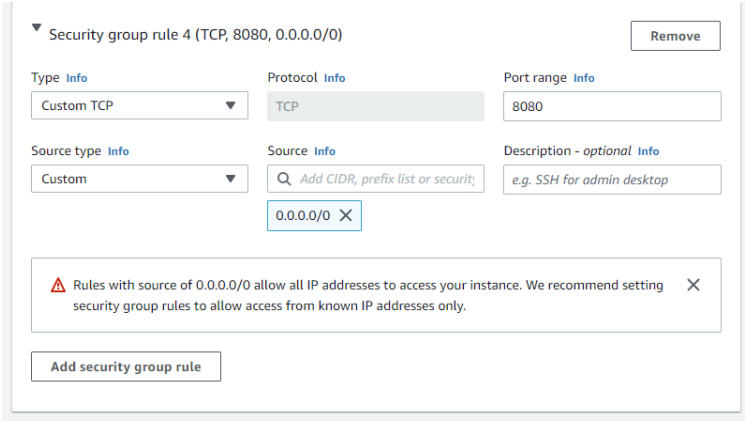
5.8. Confirm that the deployment works by selecting the deployment created, and accessing the “8080/TCP” Endpoints hyperlink



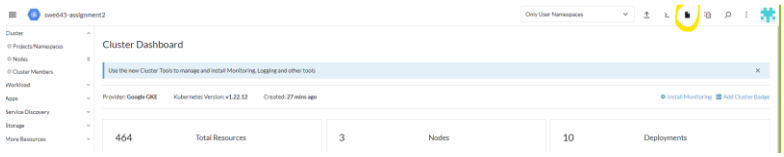


5.9. Append the name of the WAR file to access the application

6. Install Jenkins on an EC2 instance:
   6.1. Follow the steps 3.1 to 3.6 to create another EC2 instance and assign an ElasticIP to this as well
   6.2. Note that traffic to the 8080 port should be enabled in the Security Group for this one



   6.3. After installing docker, add the jenkins user to the docker group using "`sudo usermod -aG docker jenkins`". In simpler terms, this enables using docker commands without sudo
   6.4. Install Java 11 on the EC2 instance using "`sudo apt install openjdk-11-jdk`"
   6.5. Follow the steps specified here, to install Jenkins on the machine
   6.6. Verify that Jenkins is running by using "`systemctl status jenkins`"
   6.7. Install kubectl to enable Continuous Deployment through Jenkins, using the commands "`sudo apt install snapd`" and "`sudo snap install kubectl --classic`"
   6.8. Go to Rancher and download the cluster configuration using the icon in the top nav

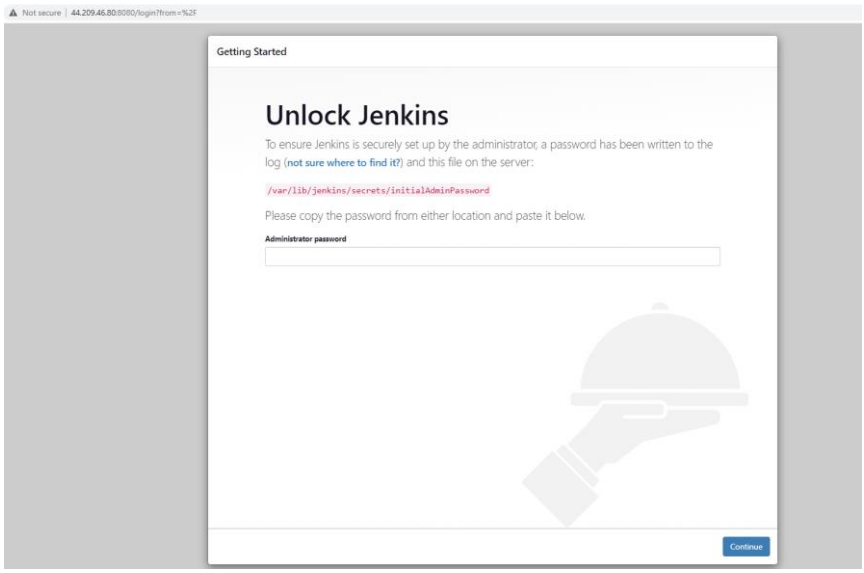


6.9. In the EC2 shell, switch to jenkins user using "`sudo su jenkins`"

6.10. Create the file "`$HOME/.kube/config`" using vi editor and paste the contents of the downloaded cluster configuration YAML

6.11. Use the command "`kubectl config current-context`" to verify that kubectl and the configuration work. The output should be the name of the cluster

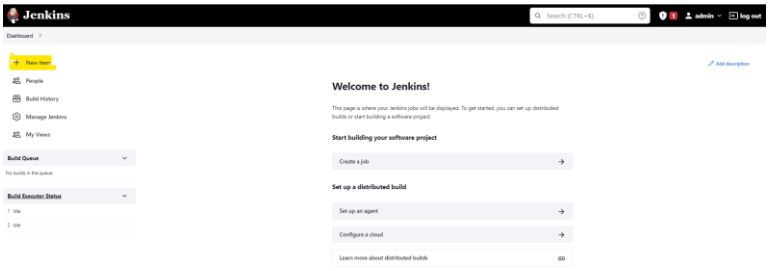
6.12. Use the configured ElasticIP with the 8080 port to access Jenkins from the browser. Follow the steps in the page to create the credentials

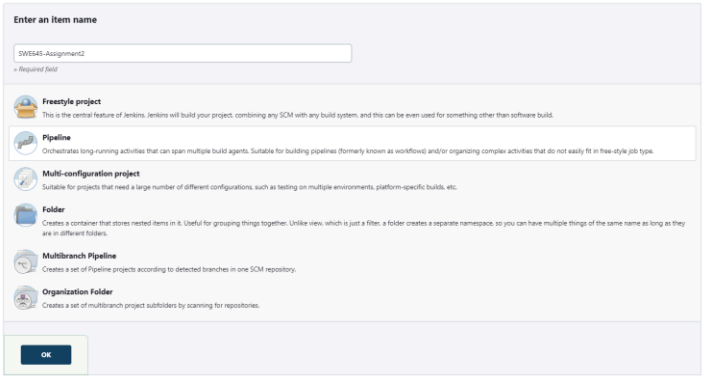


6.13. Irrespective of whether the suggested plugins were installed or not, make sure that the following plugins are installed (if not install them and restart Jenkins). Access the pipelines through "Manage Jenkins" -> "Manage Plugins"

- Pipeline
- Git
- GitHub
- Docker
- Docker Pipeline

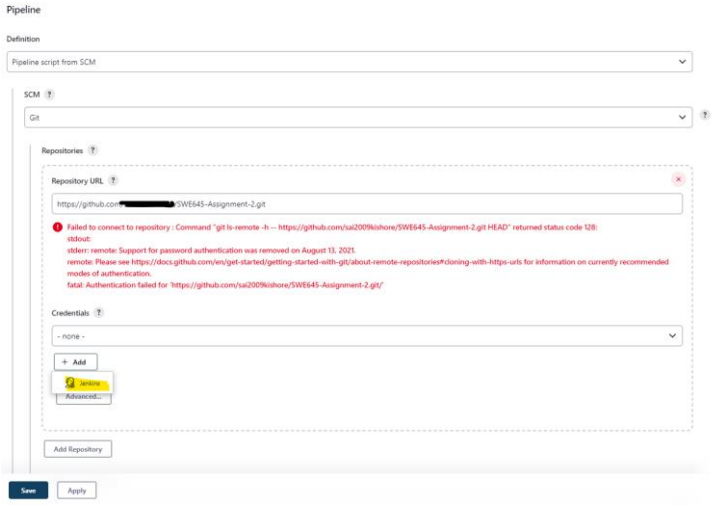
6.14. Then use the "New Item" option to create a pipeline. Provide a name and select the type as Pipeline

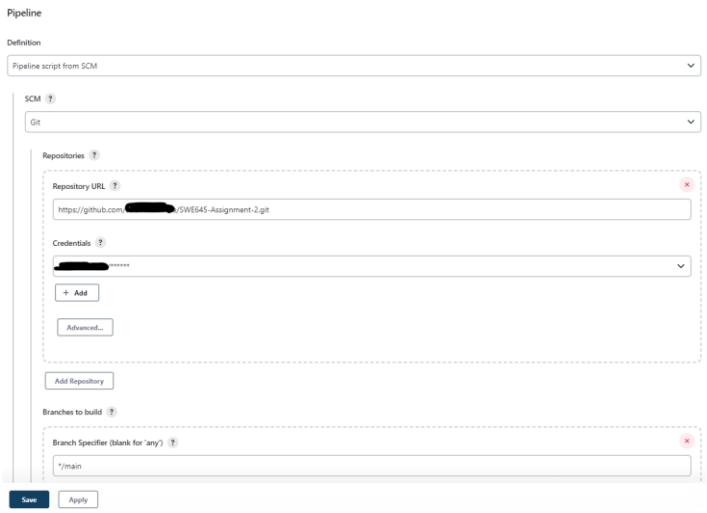




6.15. In the “Build Triggers”, set “Poll SCM” to “* * * * *” such that a change in the repository is checked every minute

6.16. In the “Pipeline”, set the Definition to “Pipeline script from SCM”, select the SCM as Git, and provide the repository URL. For GitHub, a personal access token should be used instead of the password, and the token must have read and write access to the repository. The error should be gone when the credentials work





6.17. Tip: It might not validate immediately after adding the credentials and shows the error even when the credentials are correct, so change something in the URL and undo to trigger a validation

6.18. Leave the Script Path if your Jenkinsfile is in the root directory, and uncheck Lightweight checkout

6.19. With everything else set to default, create the pipeline

7. The Jenkinsfile

```
// Jenkins file for CI/CD Pipeline. Build, tags and pushes the Docker image.
// Then deploys to the Kubernetes cluster
// Authors: Sai Kishore, Aaron Pollon, Sreenivasa Rayaprolu, Raj Sureja

pipeline {
        agent any
        environment {
                DOCKER_CREDENTIAL = credentials('docker-creds')
        }
        stages {
                stage("Building the WAR file and docker image") {
                        steps {
                                script {
                                        checkout scm
                                        sh 'rm -rf *.war'
                                        sh 'jar -cvf SWE645-Assignment2.war -C WebContent/ .'
                                        sh 'docker login -u ${DOCKER_CREDENTIAL_USR} -p ${DOCKER_CREDENTIAL_PSW}'
                                        def customImage = docker.build('coder0112358/swe645-assignment2:latest')
                                }
                        }
                }
                stage("Pushing the image to DockerHub") {
                        steps {
                                script {
                                        sh 'docker push coder0112358/swe645-assignment2:latest'
                                }
                        }
                }
                stage("Restarting the deployment to pull the latest image") {
                        steps {
                                sh 'kubectl rollout restart deploy student-survey -n assignment-2'
                        }
                }
        }
}
```

7.1. First, use this guide to add docker credentials to Jenkins. The type should be username/password

7.2. The script first retrieves the configured docker credentials

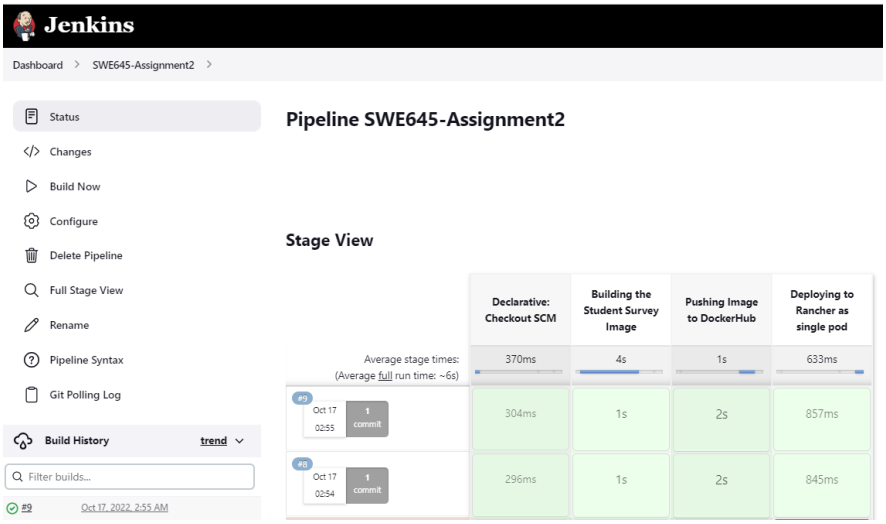
7.3. In the first stage, the repository is cloned, any previous WAR files are deleted, a new WAR file is generated, and the docker image is built

7.4. In the second stage, the built docker image is pushed to docker hub

7.5. In the final stage, the deployment is restarted. Since the image isn't changed (same tag) the deployment will check and pull the image from Docker Hub, which has the latest image with the updates

7.6. You may monitor the deployment in Rancher and observe that the pods are restarted one at a time (high availability)

8. To test the pipeline, make a noticeable change to the webpage (like updating a version) and push it. If everything is configured correctly, a pipeline should be triggered within a minute and the changes will be deployed. Check the version change in the webpage.



References:

- Previous work by Emile Issaelkhoury
- Using docker without sudo - https://docs.docker.com/engine/install/linux-postinstall/
- Install Jenkins in Linux - https://www.jenkins.io/doc/book/installing/linux/#long-term-support-release
- Kubectl cheat sheet - https://kubernetes.io/docs/reference/kubectl/cheatsheet/
- Add credentials to Jenkins - https://www.jenkins.io/doc/book/using/using-credentials/