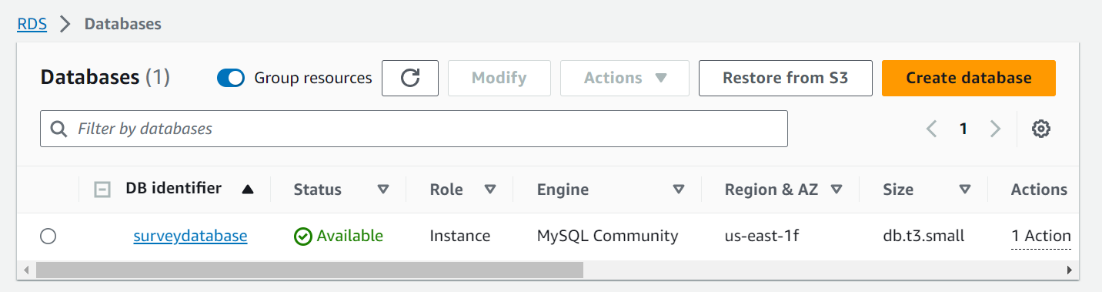
# SWE 645 : Assignment 3 README File

To complete this assignment there are a few tasks that need to be completed. This document covers all the steps and procedures of the tasks in detail along with the screenshots. Also enclosed are the Links/URLs of the various aspects of the assignments.

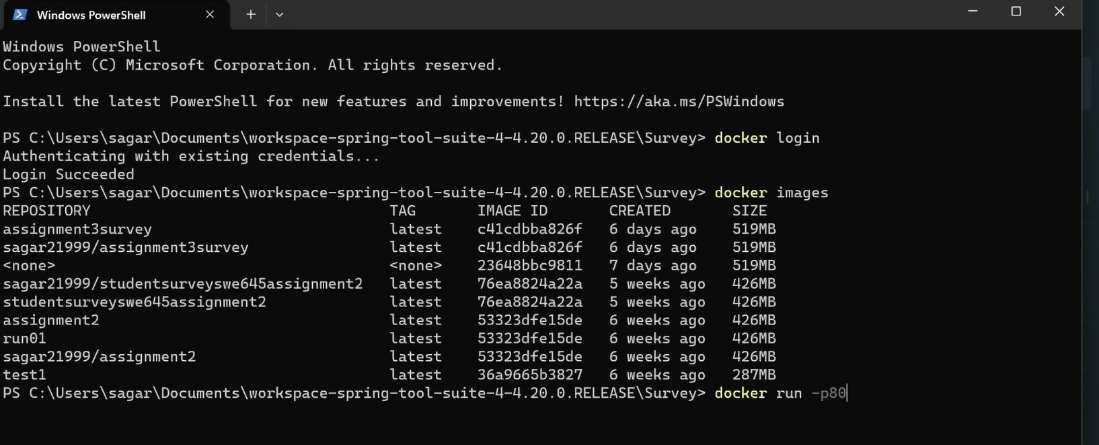
**Installation and Setup Instructions**

1. Creating MySQL database using amazon RDS
2. Creating an image of the application and pushing it into the docker hub.
3. Setting up AWS (Amazon Web Services) EC2 instances for deploying the application on a Kubernetes cluster.
4. Setting up rancher.
5. Creating a Kubernetes cluster and deploying the application.
6. Setting up AWS EC2 instance to install and run Jenkins.
7. Setting up GitHub repository.
8. Creating a CI/CD pipeline and running it.
9. Executing the application deployed on the cluster on postman.
10. **Creating MYSQL database using amazon RDS**
    * Access the AWS Management Console and navigate to the RDS section.
    * Locate and click the "Launch a DB instance" button. Upon the engine selection page, opt for MySQL and proceed by clicking "Select."
    * Now, choose the Dev/Test option compatible with the AWS Free Usage Tier, and advance to the subsequent steps.
    * Enter the database specifics, including the username, password, and select a database identifier name on the provided page.
    * Once all configurations are in place, finalize by clicking the "Launch Instance" button to deploy the database instance.

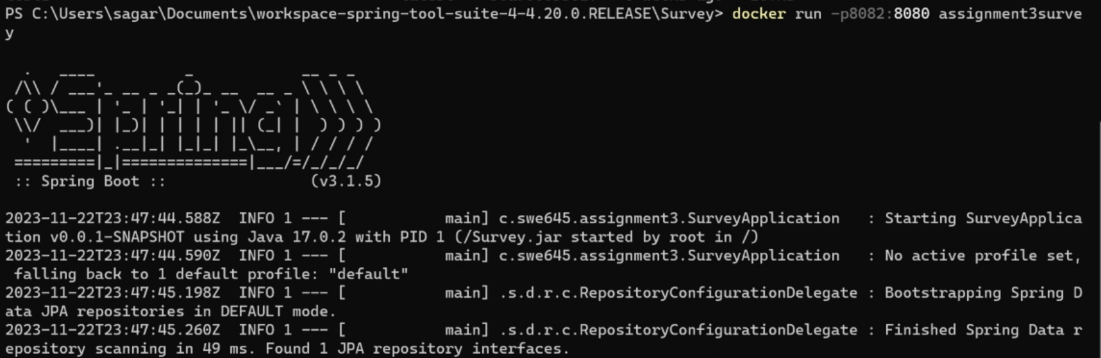


1. **Creating an image of the application and pushing it into the docker hub.**

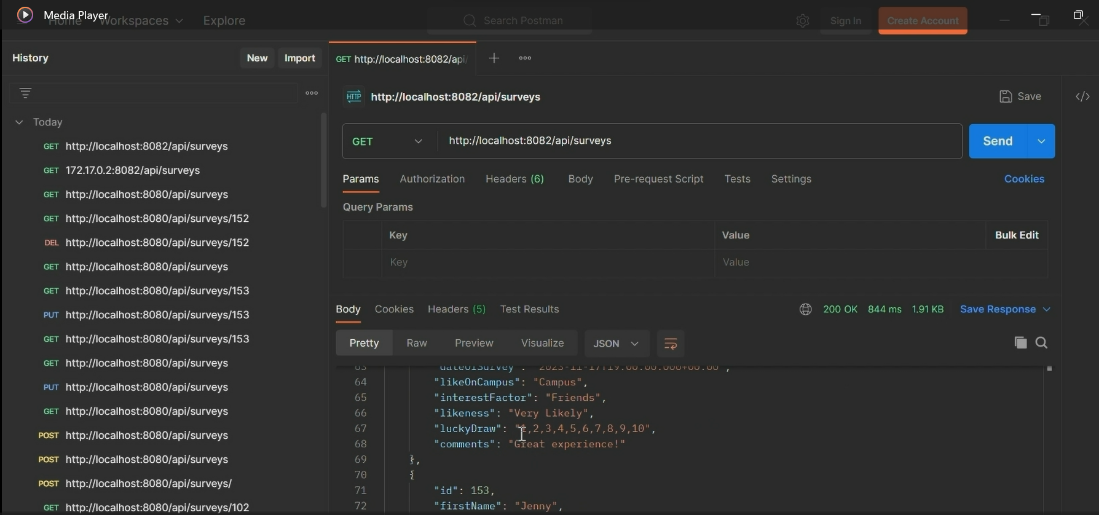
* Go to <https://hub.docker.com/> and create an account. Also download docker desktop (optional).
* Next Login to docker desktop/ docker hub on browser using the account created.
* Now create a “Dockerfile” in the same directory where you have jar file.
* Open the “Dockerfile” in an editor and have the following commands as shown in the image below written into it.
* Now open command prompt from the directory where the jar file and Dockerfile are present.
* Run the following commands to build a docker image : “ docker build -t assignment3survey .”



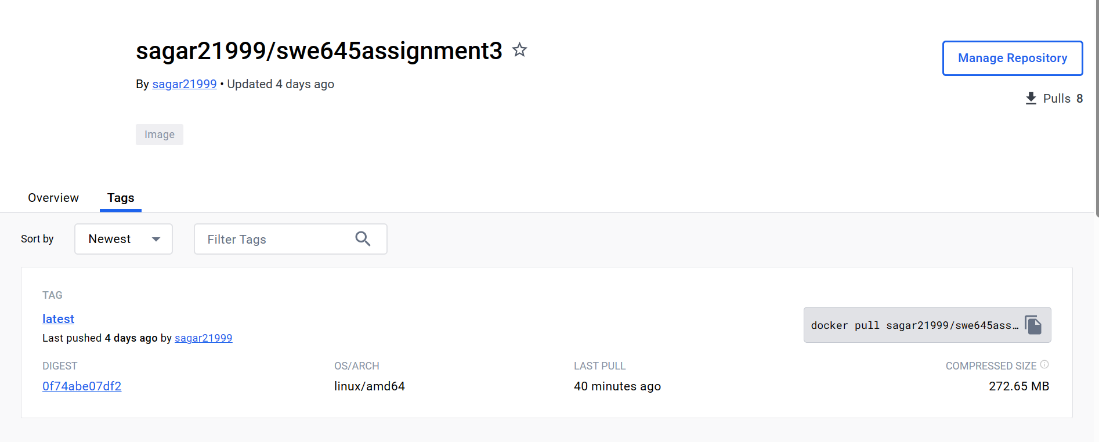
* Now an image of the application is created on the local system. To run this application on the localhost, the following command must be run on the command prompt : “ docker run -it -p 8082:8080 assignment3survey”.



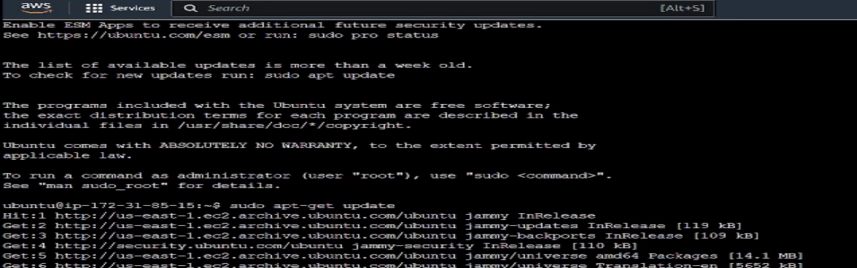
* The application must be deployed on localhost:8082 and the server must be up and running.
* Now go to a browser and open the localhost:8082 and then append to the URL “/api/surveys” and the application webpage should be visible on the browser.
* URL : [http://localhost:8082/api/surveys](http://localhost:8082/StudentSurvey/)

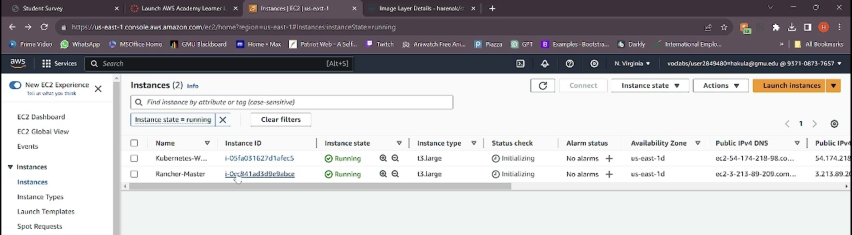


* Now we have the application image on our local system, and we must push it into the docker hub. For that we run the following commands on the command prompt from the same directory where we have run the earlier commands.
* First is to login to dockerhub using the credentials of the account created earlier : “docker login -u sagar21999”. Enter the password when prompted.
* Next tag the image to be pushed and push it into the hub using the following commands : “ docker tag assignment3survey sagar21999/swe645assignment3” followed by “ docker push sagar21999/swe645assignment3”.
* Now the image should be pushed into the docker hub.



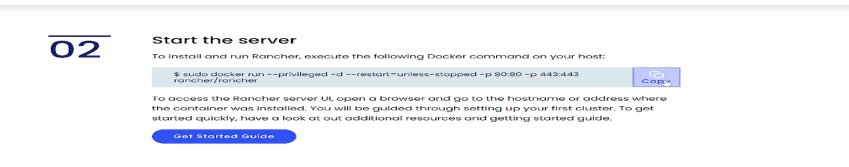
1. **Setting up AWS EC2 instances for deploying the application on a Kubernetes cluster.**

* For the rest of the assignment, we need to have 2 AWS EC2 instances.
* Login to AWS Academy and then open AWS, navigate to EC2 service and open dashboard.
* Here, click on “Launch Instance” to create the instance.
* Give the name of the instance as “Rancher-Master”, AMI : “Ubuntu Server 22.04 LTS (HVM) SSD volume Type”, Instance type : “t3.large”, Select a key-pair you already have, in this case : “SWE645Assignmen2”, Click on the checkboxes to allow traffic from HTTP and HTTPS from internet, and finally increase storage from 8Gibs (default) to 30Gibs. And finally click on “Launch instance”.
* Now, once the instance is created and is running, associate it to an elastic ip address. To do this go to “Elastic Ips” in the EC2 menu on the left pane of the screen, create an elastic ip using the default settings and associate it to this instance. We do this because whenever the AWS lab is restarted it restarts the instances on EC2 and the ip addresses change causing issues to our rancher and Jenkins setup.
* Now, click on the instance and go to the security tab. Here, in the “inbound rules”/”Outbound rules” section, click on the security group wizard.
* Scroll down and from either of the “inbound”/”outbound” tab click on edit rule option.
* Here add a rule using the following configuration, Type : “Custom TCP”, Port range : “8080”, Source : “custom” and “0.0.0.0/0” and click on “Save rules”.
* Now, click on the EC2 instance and click on “connect”. Go to “EC2 instance connect” tab, give username as “ubuntu” and click on connect. A shell opens in a new tab.
* Here, we will give commands to install docker on the instances.
* Commands to install docker : “ sudo apt-get update” followed by “sudo apt install docker.io”, Give permission when prompted: “Y”.
* 
* 
* After successfully running of the commands, docker should be successfully installed.
* Repeat all the steps mentioned above to create the second instance with the name : “Kubernetes-Worker”. And proceed to install docker as done for the first instance.
* This second instance will be our Worker node. And the first instance will be the Master node.
* At the end make sure that both the instances are in “running” state.

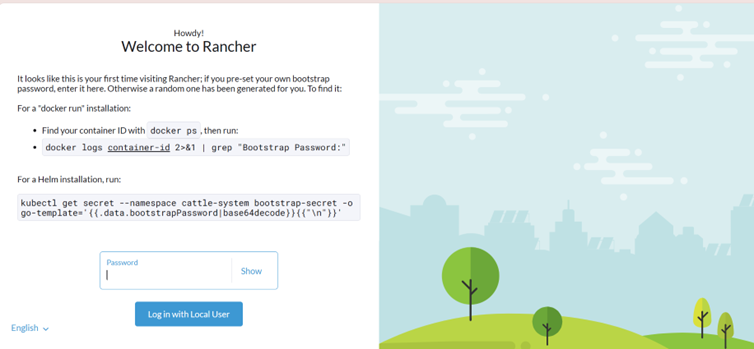
`

1. **Setting up Rancher.**

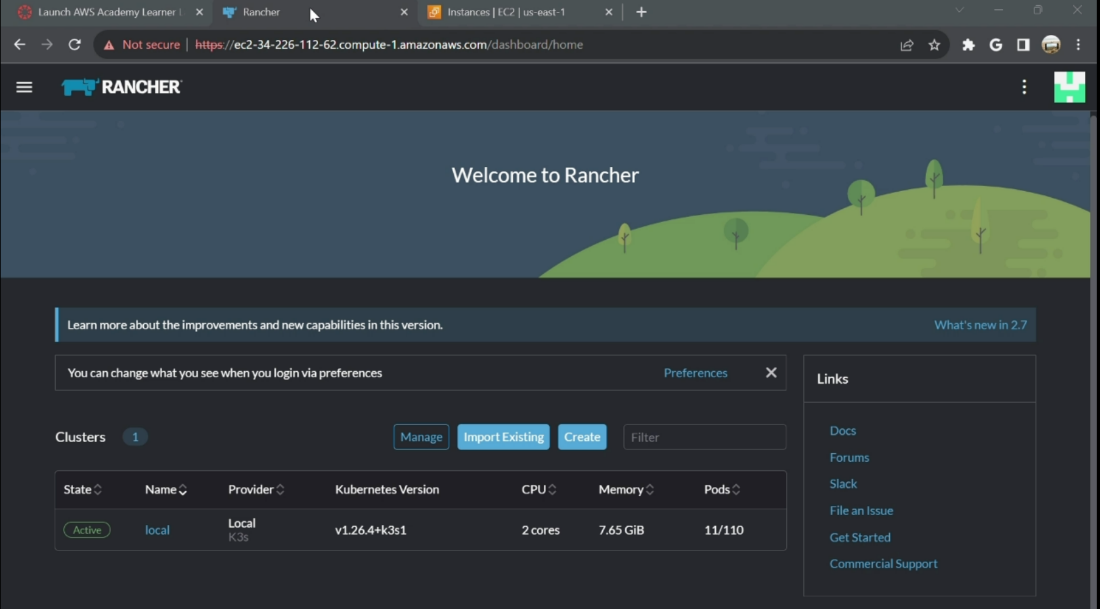
* Now that we have both the instances up and in “running” state, we will proceed to set up “rancher” on the master node i.e., “Rancher-Master” instance.
* Open browser and go to the following website: <https://www.rancher.com/quick-start> . Here scroll down to the “Start the Server” section under “Deploy Rancher”.
* Copy the command present there, which is : “ $ sudo docker run --privileged -d --restart=unless-stopped -p 80:80 -p 443:443 rancher/rancher “.



* Now, connect to the “Rancher-Master” instance in the same way as described in the earlier section.
* Here, run the copied command. After it has run successfully, rancher will have been installed on this instance and can be accessed using the public IPv4 DNS address of the instance.
* Now, on the instance, run the following command to get the details of the container running on the instance : “ sudo docker ps “. Store the result obtained somewhere (in notepad preferably), we will need this information later.
* Now, in the EC2 instance page open the “Public IPv4 DNS” URL in new tab. Give permission to proceed to the URL, even though it is unsafe. Upon loading the rancher login page should be visible.

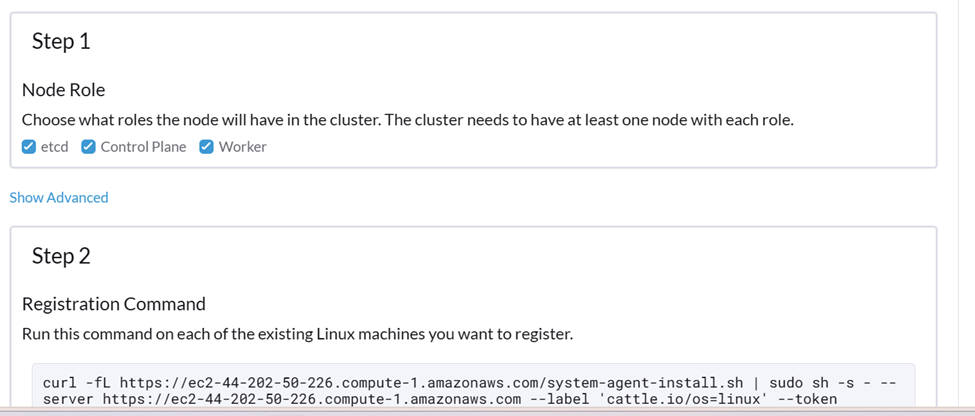


* Here, on this page it asks for a password. To get the password copy the command displayed above on the same webpage and edit the container id to have the container id from the earlier stored output and run it on the instance console. The command should look like this : “sudo docker logs 49d9b5c86913 2>&1 | grep “Bootstrap Password:”. The console will display the Password on the screen.
* Copy the password and paste it on the rancher login screen to login.
* On successful login, set up a new password using the prompts on the screen.
* Now the rancher dashboard will be displayed, where we can proceed with creating a cluster and deploying the application.

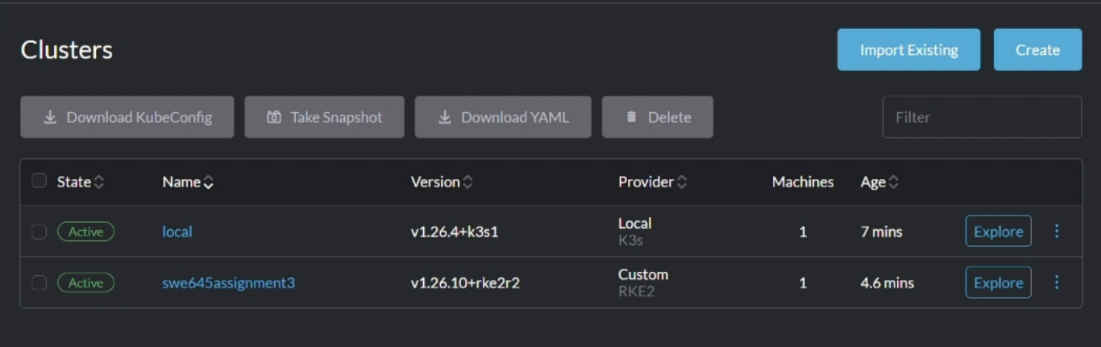


1. **Creating a Kubernetes cluster and deploying the application.**

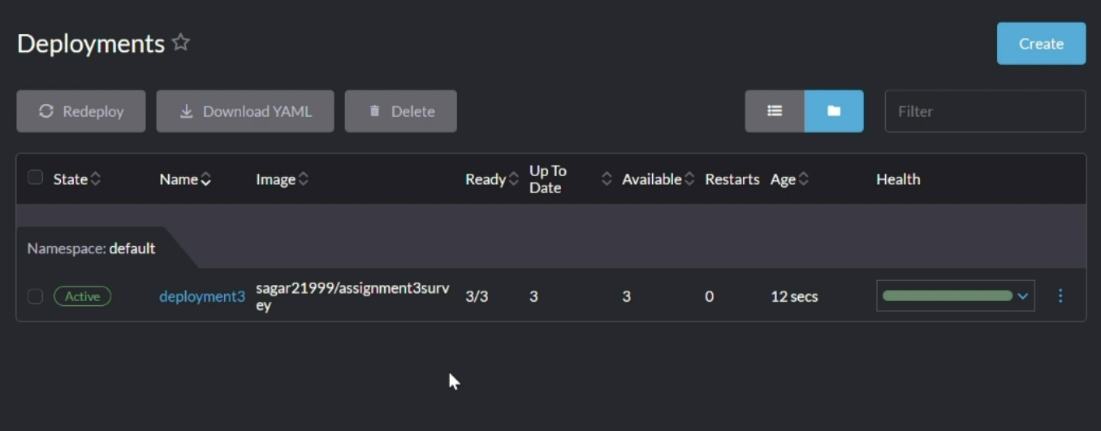
* Now, as we have rancher up and running, we will be creating a cluster and deploying our application on the cluster using the Rancher UI.
* On the dashboard, click on “Create Cluster” button. It should show various cluster options, select the “Custom” option and a create cluster form should be visible.
* Here fill the form by giving the cluster name : “swe645assignment3” and leave everything else as default and click the “Create” button.
* On the next page, go to “Registration” tab and under “step1” make sure that the 3 checkboxes of “etcd”, “control pane” and “worker” are checked.



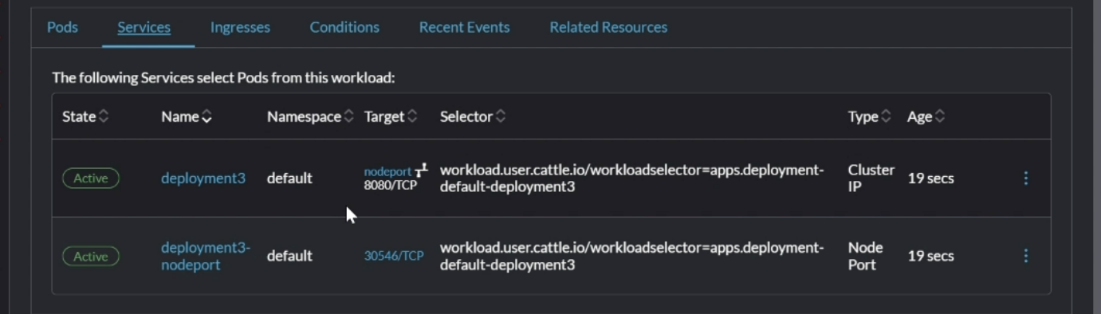
* Now copy the command present below in “step2”.
* Open AWS EC2 instance and connect to the “Kubernetes-Worker” node.
* Here run the copied command but append the “—insecure" after the “curl” word in the command then run it.
* After the command is successfully executed, the cluster should be in “Updating” state and after a few minutes it will change to “Active” state and the cluster will be ready for deployment. This can be viewed under the “Clusters” option in “cluster management” on the Rancher UI.



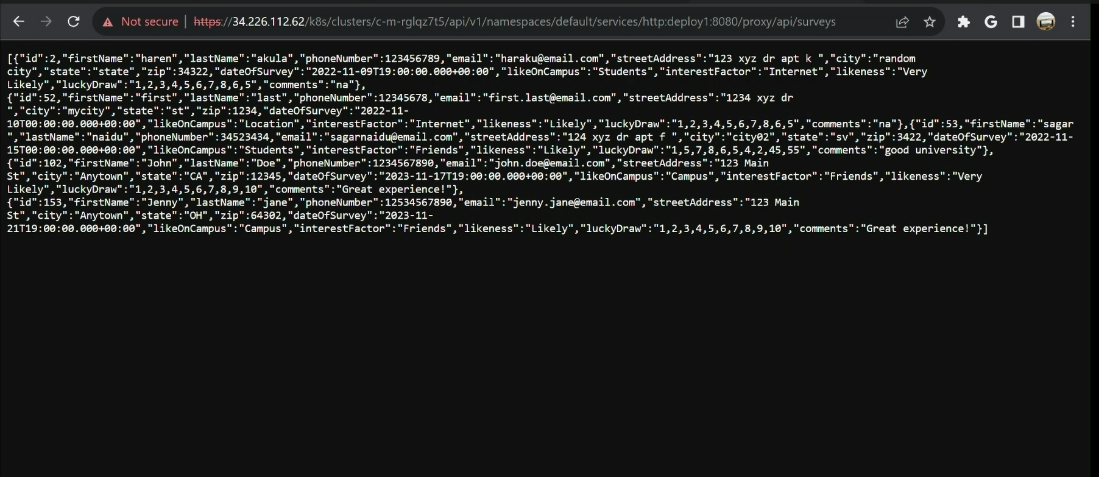
* Once the cluster is in “Active” state, click on the “Explore” button next to the cluster.
* On the left pane, under “workloads” select “Deployments”.
* Click on the “create” button and a form should be visible on the screen.
* Fill in the form using the following details. Namespace : “default”, Name : “assignment3deployment”, Replicas : “3” (no. of pods), Container image : “sagar21999/swe645assignment3:latest” (same as the name of the image on the docker hub).
* Under Networking, click on “Add port or service” button.
* Give the following details, Service type : “node Port”, name : “nodeport”, private container port : “8080”, protocol : “TCP”.
* Leave everything else as default and click on create.
* Now the deployment is complete, and it should be in “Updating” state. Give it a few minutes and it should be in “Active” state.



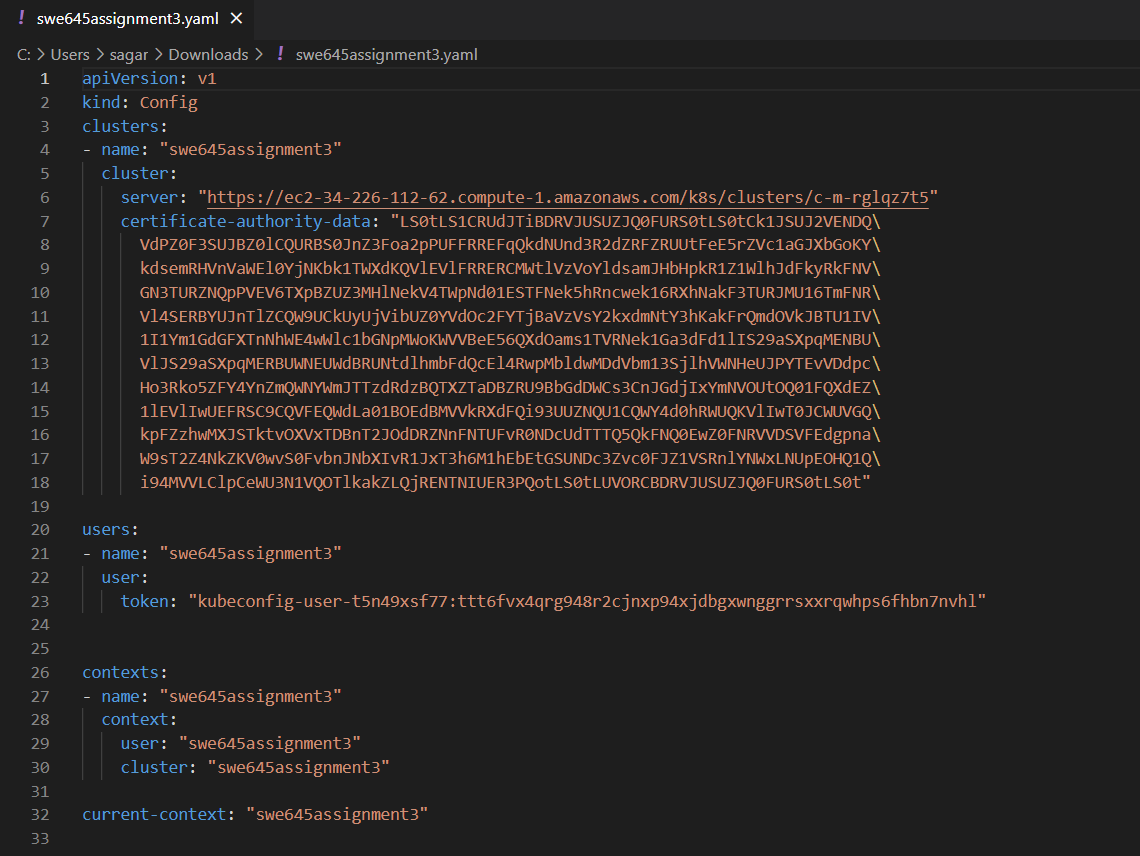
* Once it is in the “Active" state, click on the deployment, navigate to the “Services” tab.
* Here, find the deployment created and click on the “nodeport” to open it in a new tab under the target option.



* Now to the URL, append “/api/surveys” and open it. You should be able to see the application running.
* <https://34.226.112.62/k8s/clusters/c-m-rglqz7t5/api/v1/namespaces/default/services/http:deploy1:8080/proxy/api/surveys>
* With this, we will have successfully deployed the application on a Kubernetes cluster

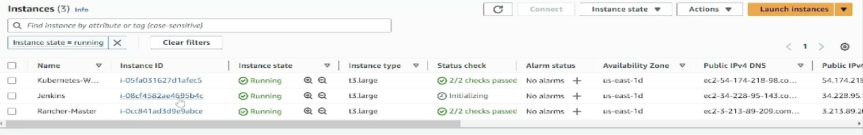


* For the next step we will need the “KubeConfig” file. To download it go to our cluster page. On the top right, from the menu click on “Download KubeConfig file” and save it in your local machine.

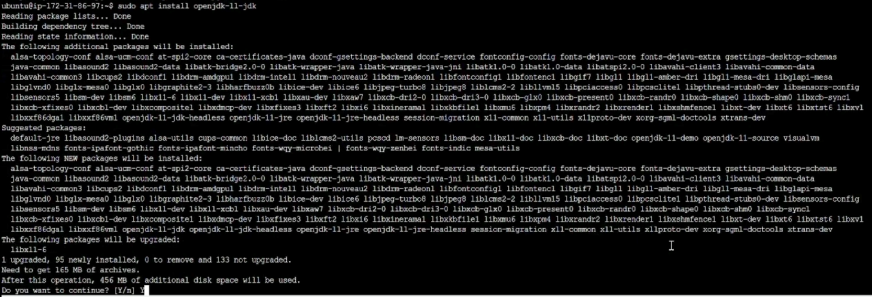


1. **Setting up AWS EC2 instance to install and run Jenkins.**

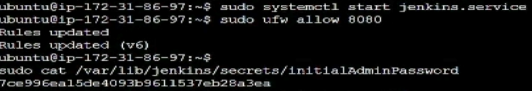
* Now, we will focus on creating a CI/CD pipeline using Jenkins, to automatically build and reflect any changes in our source code.
* For this, we need to Jenkins installed on our machine, so we will start off by creating an AWS EC2 instance in our AWS lab.
* We will create the instance in the same manner as we did to create our 2 EC2 instances to deploy our application on the Kubernetes cluster.
* Give the name of the EC2 instance as “Jenkins”, use the same configuration, give an elastic ip and edit the security group.



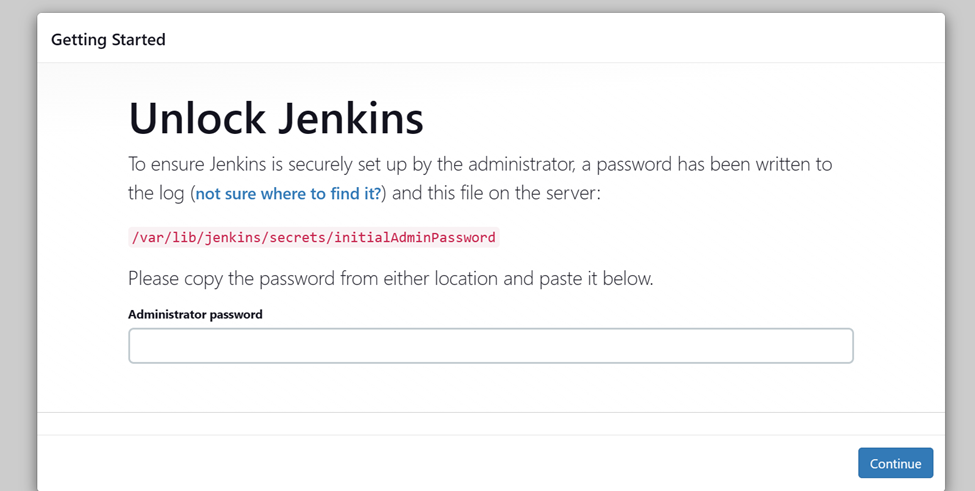
* Connect to the instance using “EC2 instance connect”. Make sure that the instance is in “running” state.
* First, we need to install java before going for Jenkins.
* For that we run the following commands : “sudo apt-get update” followed by “sudo apt update”.
* Now run “ sudo apt install openjdk-11-jdk” and give permission when prompted as “Y”. This should install java jdk-11.



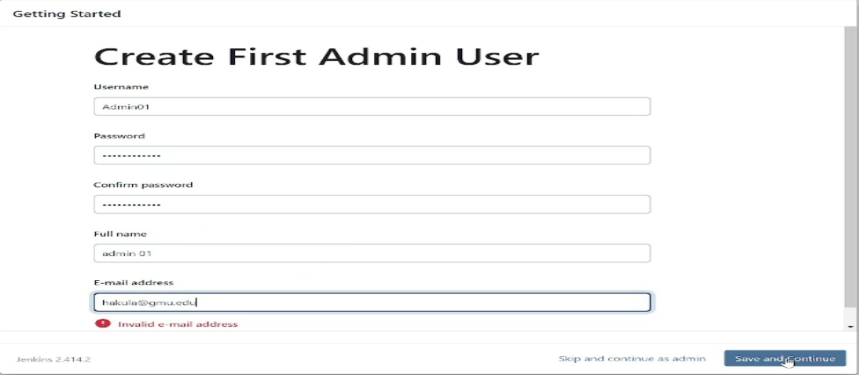
* Now Jenkins can be installed by running the following commands: “sudo curl -fsSL <https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key> | sudo tee /usr/share/keyrings/jenkins-keyring.asc > /dev/null” to get Jenkins package.
* Next run “ sudo echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] <https://pkg.jenkins.io/debian-stable> binary/ | sudo tee /etc/apt/sources.list.d/jenkins.list > /dev/null “ to unzip Jenkins.
* Now update apt and install Jenkins using the commands : “ sudo apt update” followed by “sudo apt install jenkins” . Give permission as “Y” and jenkins should be successfully installed.
* Now start jenkins using the command : “sudo systemctl start jenkins.service”.
* Expose the port 8080 using the command : “sudo ufw allow 8080”.
* Now run the following command to get admin password : “sudo cat /var/lib/jenkins/secrets/initialAdminPassword ” . It will display the password. Store It safely.



* Run the following commands to install snapd : “sudo apt install snapd”.
* Now using snap install kubectl : “ sudo snap install kubectl –classic”.
* Now go to AWS EC2 instance “Jenkins” and open the “Public IPv4 address” in a new tab.
* To the url append the port 8080 and change from “https” to “http”. It should look like the following: “ [http://34.228.95.143:8080](http://34.228.95.143:8080/)”.
* Enter the admin password obtained earlier to unlock jenkins.



* Now proceed to install the suggested plugins.
* Next create an admin user. Click on “Save and create user” followed by “save and finish” and lastly “Start using jenkins”.



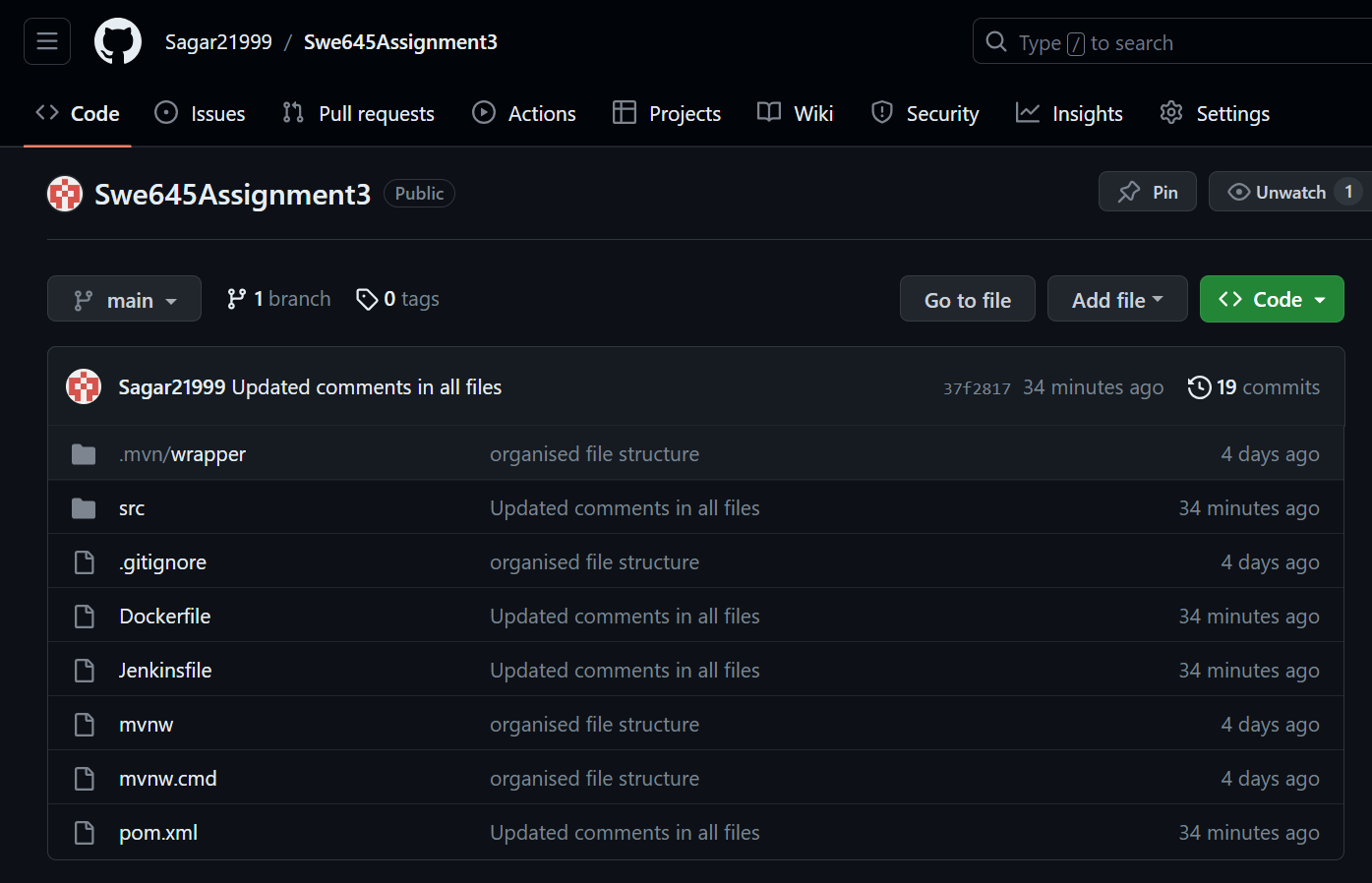
* You should be able to see the jenkins dashboard.
* Now go back to the EC2 console. Now we need to create a config file.
* Run the following commands : “sudo su jenkins” to go to jenkins home.
* Next go to root directory : “ cd ../../”.
* Go to the directory : “cd /var/lib/jenkins”
* Create a directory and enter it : “mkdir .kube” followed by “cd .kube”.
* Create the file and open it : “vi config”
* Now copy the contents from the “KubeConfig” file downloaded earlier and paste it here.
* Save the file using “:wq”.



* Now to verify run the command : “kubectl config current-context”. It should display the cluster name as output.
* Next exit the jenkins mode using : “exit” command.
* Now install docker using the commands : “sudo apt-get update” and “sudo apt update” followed by “sudo apt install docker.io”. Give permission as “Y”.
* After installing, change file permissions for socket using command: “sudo chmod 777 /var/run/docker.sock”.

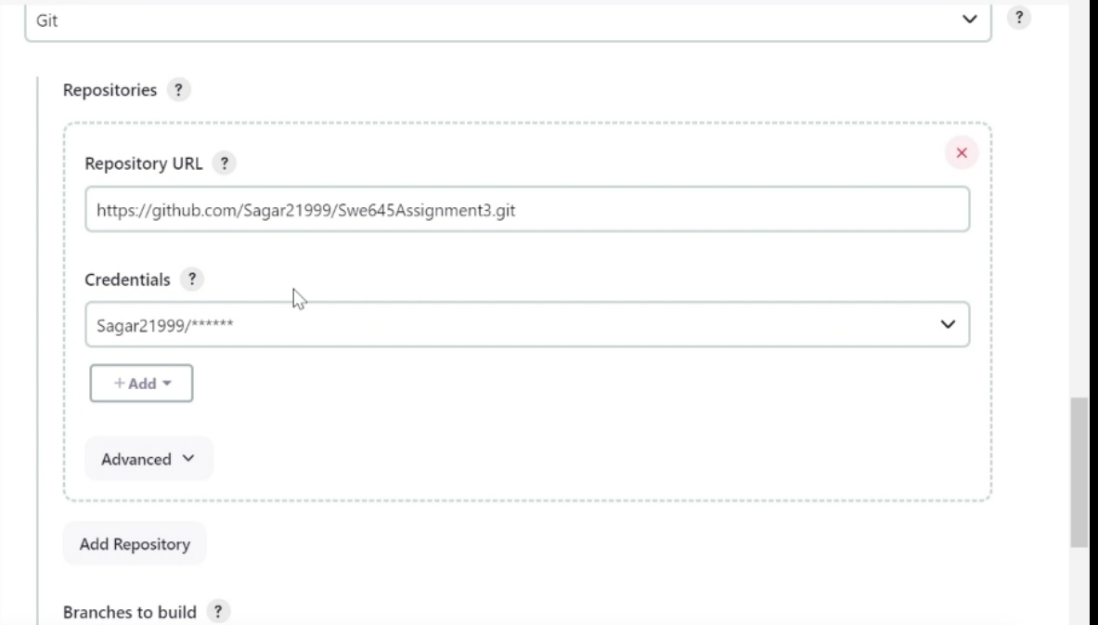
1. **Setting up GitHub repository.**

* In order to create our pipeline, we need to have the source code saved somewhere, from where we can attach it to pipeline, and it can look for any changes. For this purpose, we will be using GitHub.
* Go to <https://github.com/> and create an account and login to it.
* Now create a new repository : “ Swe645Assignment3”.
* To that repository add your source files, Dockerfile and the jar file.
* And then commit them.
* We will be using this repository for our pipeline.
* Github repository URL : <https://github.com/Sagar21999/Swe645Assignment3.git>

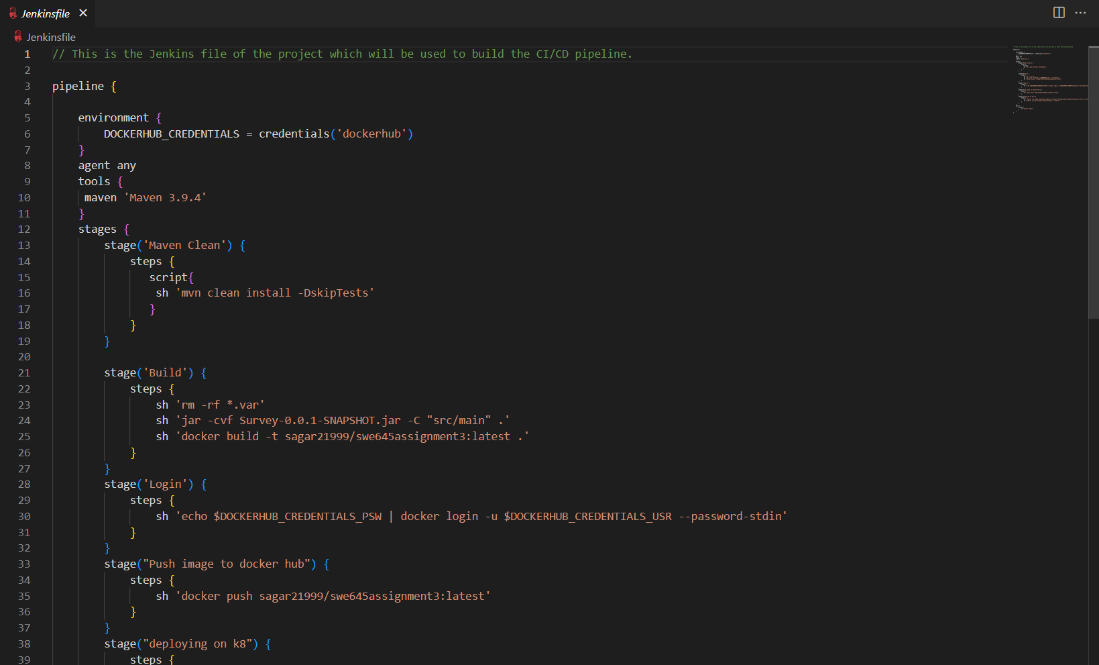


1. **Creating a CI/CD pipeline and running it.**

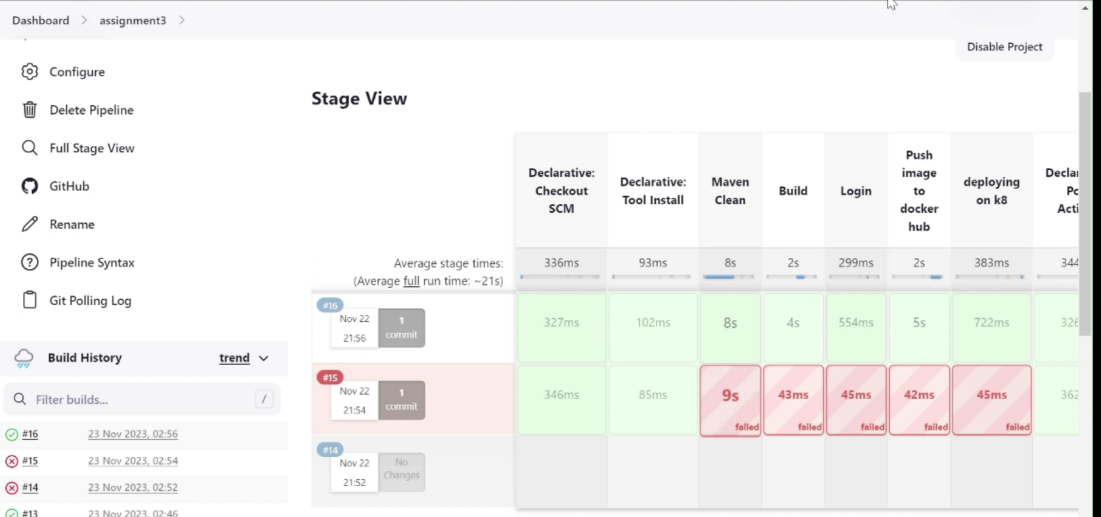
* Now we have everything set up and ready to create our pipeline.
* Go to the jenkins dashboard, click on “mange jenkins”, scroll and select the “Credentials” option.
* Here select the “global” option and then click on “Add credentials”.
* Select Kind : “Username and password”, enter the dockerhub username and password, give id as “dockerhub” and click on create.
* Repeat the above step to save GitHub credentials as well, give id as “github”.
* Now go to dashboard and click on “New Item”.
* Give name as “assignment3” and click on “Pipeline”.
* Now on the next page, check the “Github Project” checkbox, and in the project URL paste the URL from Github repository : “https://github.com/Sagar21999/Swe645Assignment3”.
* Scroll down and check the “Poll SCM” checkbox.
* Give the Schedule as : “\* \* \* \* \*”.
* Scroll down to the “Pipeline” section.
* Select the definition : “Pipeline from SCM” and SCM : “git”.
* Give the Github repository : “<https://github.com/Sagar21999/Swe645Assignment3.git> .
* Under Credentials select the github credentials.
* Then change the branch specifier : “\*/main”.

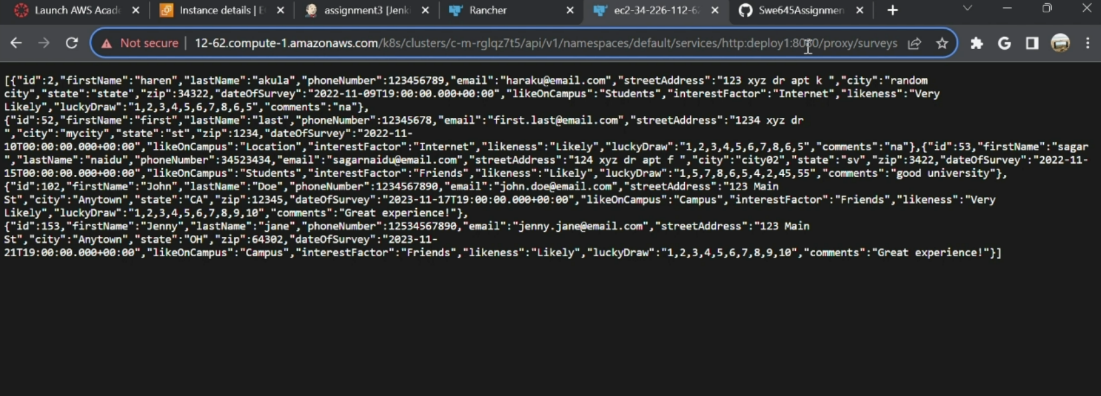


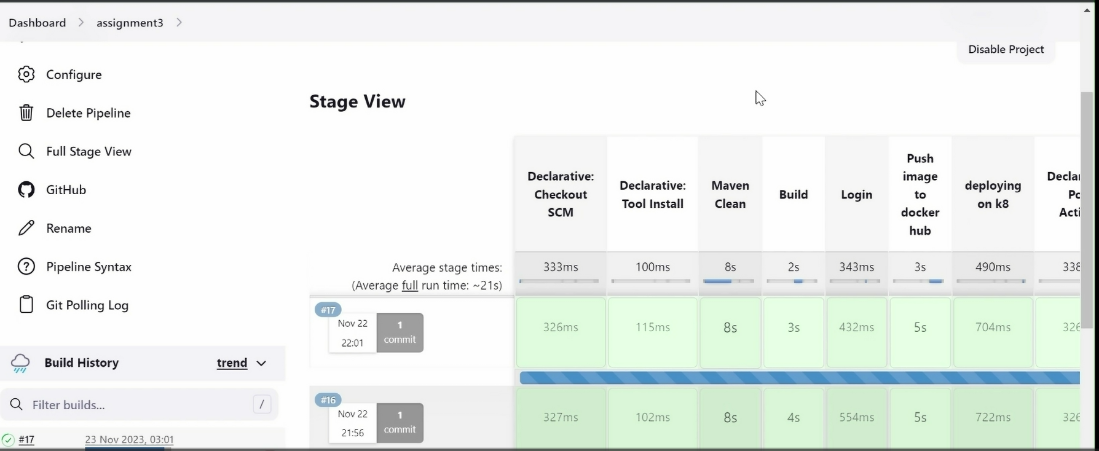
* Go to GitHub repository and then create a new file : “Jenkinsfile” and then commit changes.
* Click on Save.
* Now a build will automatically start on this pipeline.
* Now go to the Github repository -> “Jenkinsfile” and click on edit.
* Enter the commands given in the screenshot below and commit the changes. These are the commands that should be executed when any commit or changes occur to the files in the repository, They contain commands on how to generate a new war file, create a docker image, login to docker and push image, deploy image on Kubernetes cluster.
* Now when a build occurs, there might be some errors in the path and all, solve them by editing the Jenkinsfile.

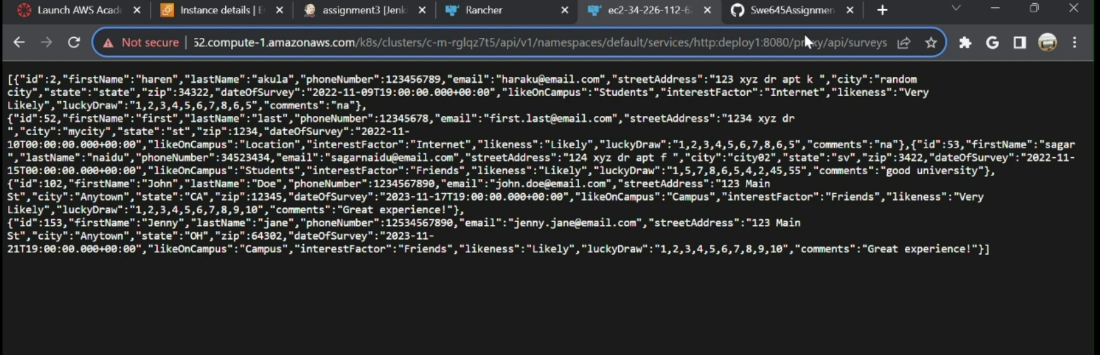


* Now after a successful build, open the application deployed on Kubernetes cluster using the same old URL. And you should be able to see the application webpage.

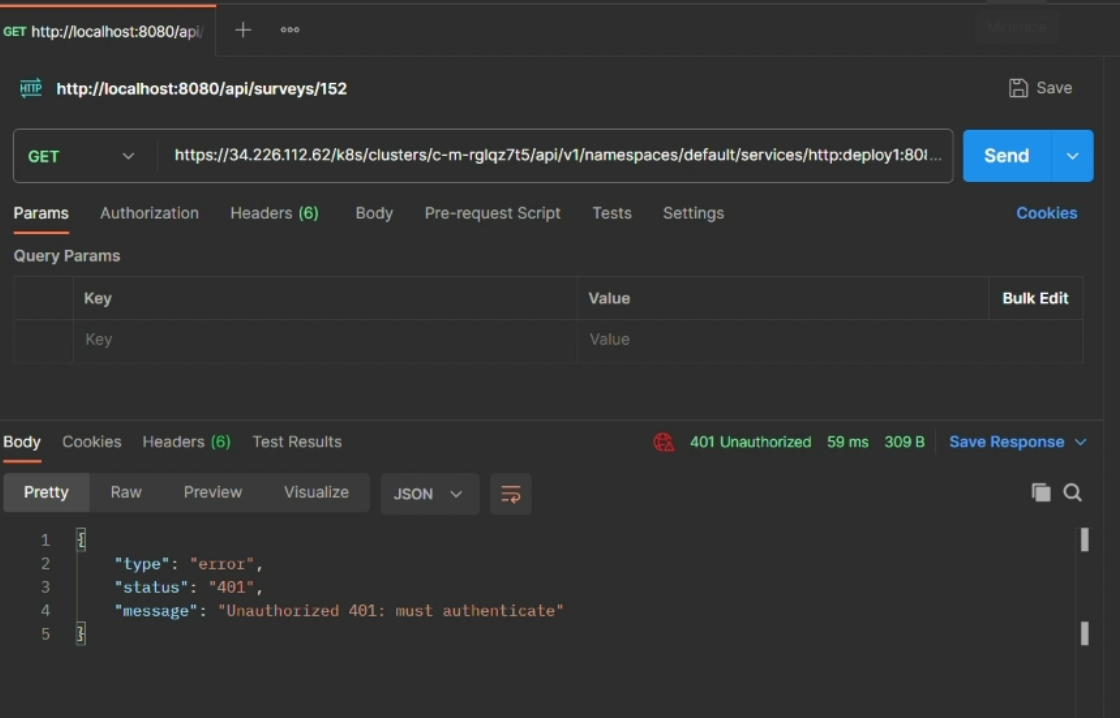




* Now to test the pipeline, open your source file in github and perform some changes. In our case we have decided to change the url from “/surveys” to “/api/surveys” commit the changes.
* After the commit, the pipeline automatically creates a build. After the build is successful, if we reload the webpage, we will be able to see the changes.Inserting image...
* URL : <https://34.226.112.62/k8s/clusters/c-m-rglqz7t5/api/v1/namespaces/default/services/http:deploy1:8080/proxy/api/surveys>



1. **Executing the application deployed on the cluster on postman.**
   * It should be noted that any application deployed on the cluster will run on that browser only and will not run on postman as it cannot authenticate the traffic. We do not have an ingress controller deployed at this point (part of bonus) to handle external traffic. Therefore, we will not be able to run this application on postman.

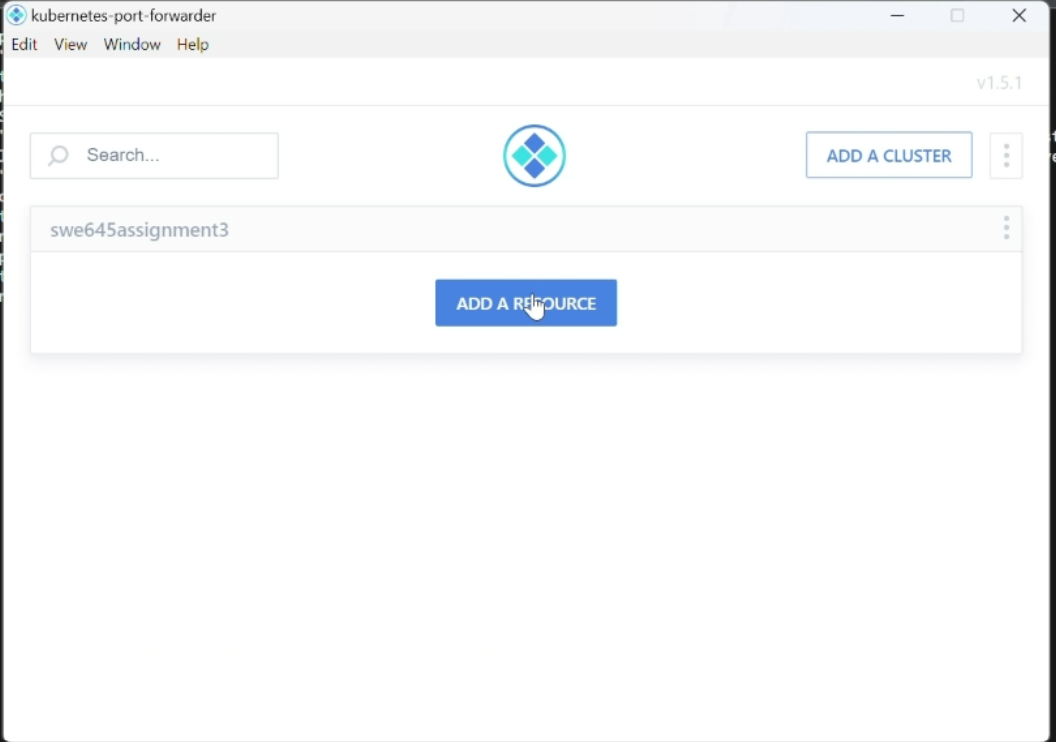


* But we can use a workaround for this and have it run on postman. For this we will be using the concept of **port forwarding.** Using port forwarding we will have the service running on the cluster rerouted to our local host and have it running on our local host from where we will be able to run and access the api from external source such as postman.

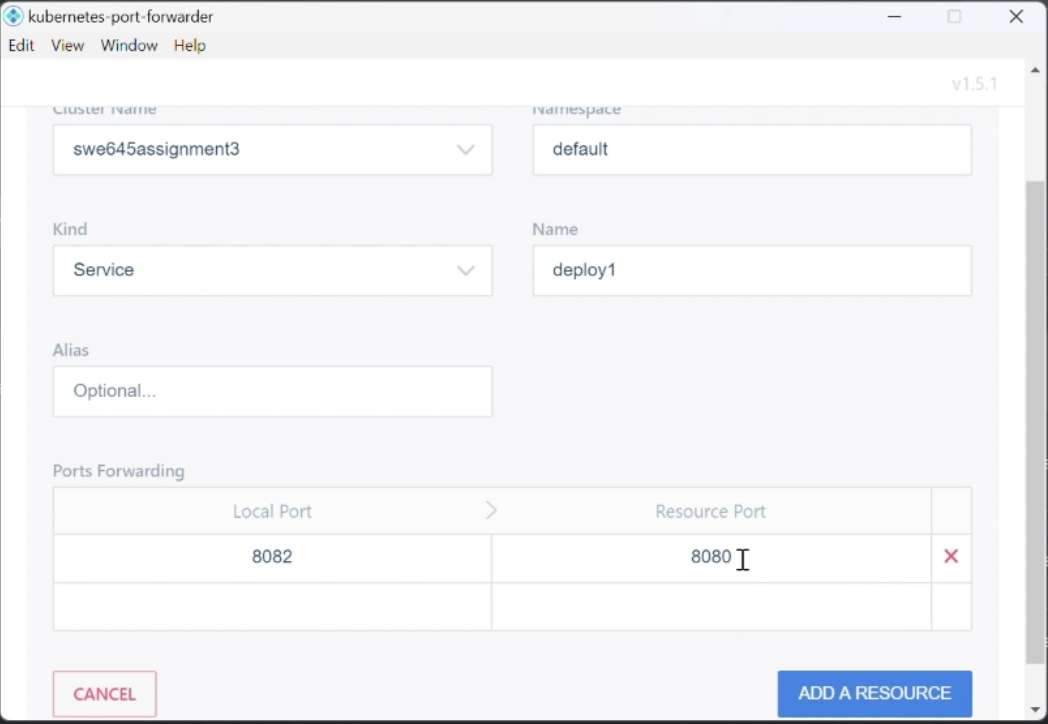
To use port forwarding, we can use kubectl commands or the GUI based tool (Kube forwarder). For simplicity purposes we will be using the GUI tool. To download the tool, use the following link : <https://kube-forwarder.pixelpoint.io/>

Follow the below steps once you download the tool:

* Open the Kube Forwarder tool.
* Click on “Add a Cluster” button.
* Select the “Add Kubernetes Config manually” option.
* Select “Add from Open file(s)” and then upload the Kubernetes config file of the cluster downloaded from Kubernetes cluster.
* Now click on “Add a resource”.

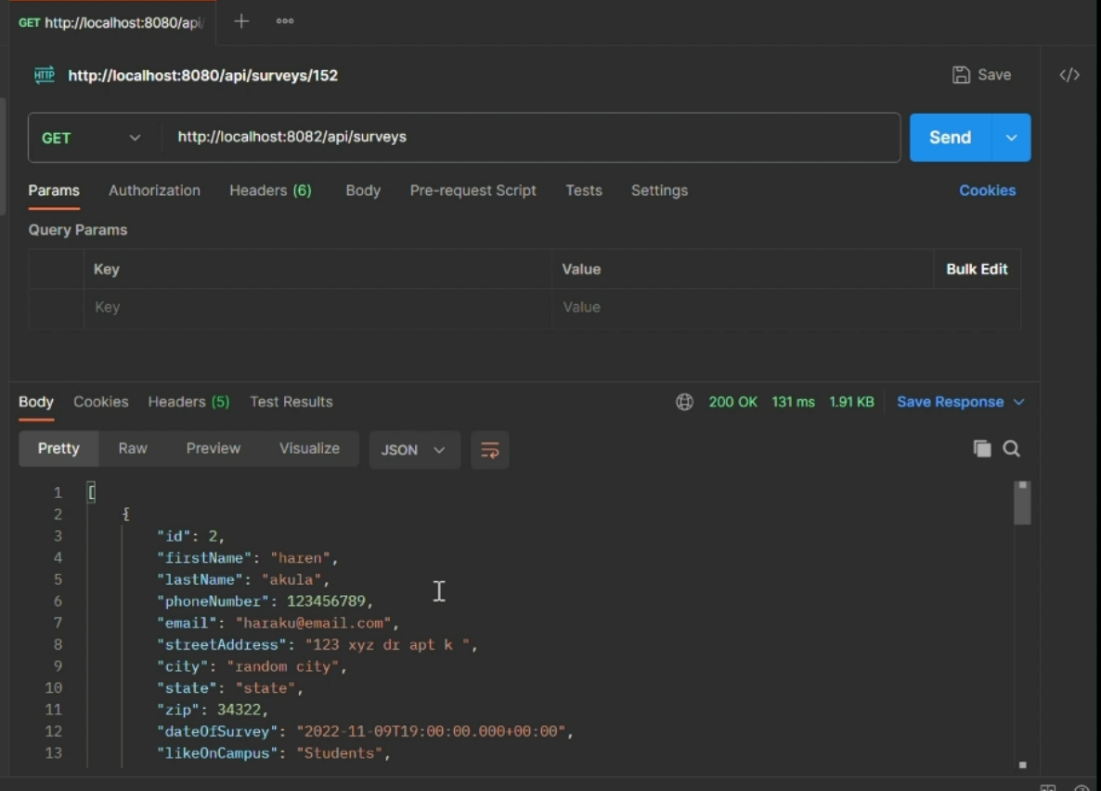


* Now on the form, fill the following details for the service running on the cluster which as given below in the image.
* So now the service will be running on our local host on port 8082.
* Click on “Add a Resource”.



* Now after the resource is created. You'll see the below screen. Start the resource and now you can run it on local host by clicking on the port given there.



* Append “/api/surveys” at the end of the URL to run it and get the result.
* 

**Implementation Details**

* We have designed the Spring Boot-based API using the RESTful principles and annotations like @RestController, @RequestMapping, @PathVariable, and others.
* Our API can be accessed through GET and POST requesting the postman at specified API endpoints (URIs).
* Our survey entity is structured as a JPA entity within the model. It involved a repository responsible for CRUD operations, a service layer handling business logic, and a controller defining API endpoints.
* For Displaying the survey forms on localhost use API end point**-** [**http://localhost:8082/api/surveys/{id}**](http://localhost:8080/api/surveys/%7bid%7d)
  + Here, to access a specific student use id
  + To access all the student's data, remove id
* For creating the survey use POST /api/surveys, use [**http://localhost:8082/api/surveys**](http://localhost:8080/api/surveys)
  + Add data in body with data options as raw, Json
* For deleting use DELETE **http://localhost:8082/api/surveys/{id}**
  + Replace {id} with the specific student if you want to delete
* To modify an existing student data use[**http://localhost:8082/api/surveys/{id}**](http://localhost:8080/api/surveys/%7bid%7d)
  + Add the data to be modified in the body with data options as raw, JSON
* For API endpoints On Kubernetes Cluster just change the "http://localhost:8082/" with <https://34.226.112.62/k8s/clusters/c-m-rglqz7t5/api/v1/namespaces/default/services/http:deploy1:8080/proxy/api/surveys>

**URL/Links of the application deployed:**

1. GitHub URL : <https://github.com/Sagar21999/Swe645Assignment3.git>
2. Docker hub URL : [Image Layer Details - sagar21999/swe645assignment3:latest | Docker Hub](https://hub.docker.com/r/sagar21999/swe645assignment3)
3. Application deployed on cluster URL : <https://34.226.112.62/k8s/clusters/c-m-rglqz7t5/api/v1/namespaces/default/services/http:deploy1:8080/proxy/api/surveys>

**References:**

1. <https://www.digitalocean.com/community/tutorials/how-to-install-jenkins-on-ubuntu-20-04>
2. <https://docs.docker.com/get-started/02_our_app/>
3. <https://docs.rke2.io/>
4. <https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_GettingStarted.html>
5. [Spring | Guides](https://spring.io/guides#gettingStarted)
6. <https://komodor.com/learn/kubectl-port-forwarding-how-it-works-use-cases-examples/#:~:text=It%20works%20by%20creating%20a,service%20within%20the%20Kubernetes%20cluster.>

**File description:**

1. Dockerfile – Docker file used to build image.
2. Jenkinsfile – Jenkins file used to perform build after any commits/changes to source code.
3. Swe645assignment3.yaml - KubeConfig file of the cluster.
4. SurveyApplication.java – main class file
5. SurveyController.java– file to handle the http requests
6. ResourceNotFoundException.java - Java file for Exception handling
7. SurveyRepository.java - Java file for accessing and managing relational database.
8. SurveyService.java - this java file contains the blueprint for handling survey related operations of our application
9. SurveyServiceImpl.java - java file for implementing survey-related services, handling database operations through the repository class.
10. Survey.java - entity class file containing the data structure of survey form
11. pom.xml - configuration file of our maven project
12. application.properties - file containing database information and connection details.
13. Swe645assignment3.zip - zipped file containing all the project source files.
14. Swe 645 Assignment 3 – Execution video .mp4 – Video recording of the entire assignment development and execution.
15. SWE 645 Assignment 3 README .docx – Word document, README file of the assignment.
16. SWE 645 Assignment 3 README.pdf – PDF version of the README file.

**Link to video of execution :** [**https://gmuedu-my.sharepoint.com/:v:/g/personal/spotana\_gmu\_edu/EUnv7pI6KQxKhCtclDx3954BgoEt0ziUljYbUUMpobsaVw?e=ZCiyFZ**](https://gmuedu-my.sharepoint.com/:v:/g/personal/spotana_gmu_edu/EUnv7pI6KQxKhCtclDx3954BgoEt0ziUljYbUUMpobsaVw?e=ZCiyFZ)