SWE 645 Extra Credit

- Dheeraj Krishna Nagula (G01448319),
Akhilesh Dhavileswarapu (G01472450)

To finish Extra credit, several steps need to be completed. This README file provides a detailed overview of the tasks and procedures, along with accompanying screenshots. It also includes links/URLs to various elements of the assignment for reference.

1. Building an image of the application and uploading it to Docker Hub using Docker Desktop.
2. Configuring AWS EC2 instances to deploy the application on a Kubernetes cluster using Rancher.
3. Setting up Rancher.
4. Using the Rancher UI to deploy the application and set up a Kubernetes cluster.
5. Setting up an AWS EC2 instance for Jenkins installation and operation.
6. Setting up the project's GitHub repository.
7. Constructing a CI/CD pipeline and running it fully configured on Jenkins.

1. Creating and Uploading an Application Image to Docker Hub Using Docker Desktop

- Start first by visiting https://hub.docker.com/ to set up an account. You can download and install Docker Desktop for a more seamless experience.
- Log in to your Docker account in Docker Desktop on your laptop and also in Docker Hub.
- In the root directory, a Dockerfile should be created.
- Open the Dockerfile in a code editor and write the commands needed as seen in the reference image.

```
FROM python:3.10-slim

WORKDIR /app

COPY . /app

RUN pip install --no-cache-dir -r requirements.txt

EXPOSE 8080

ENTRYPOINT ["uvicorn", "studentSurvey:app", "--host", "0.0.0.0", "--port", "8080"]
```

- From this directory, open up a command prompt.
- To build the Docker image, run the following command:
    docker build -t dheerajkrishna141/extra_credit:latest .
- This command builds the Docker image locally. To run the application on your local machine, execute:
    docker run -d -p 8080:8080 dheerajkrishna141/extra_credit:latest

- The next step in the process should be to start the server that will host the application on localhost:8080.
- Then, open your web browser and go to http://localhost:8080/api/survey/all to load the web page for the application.
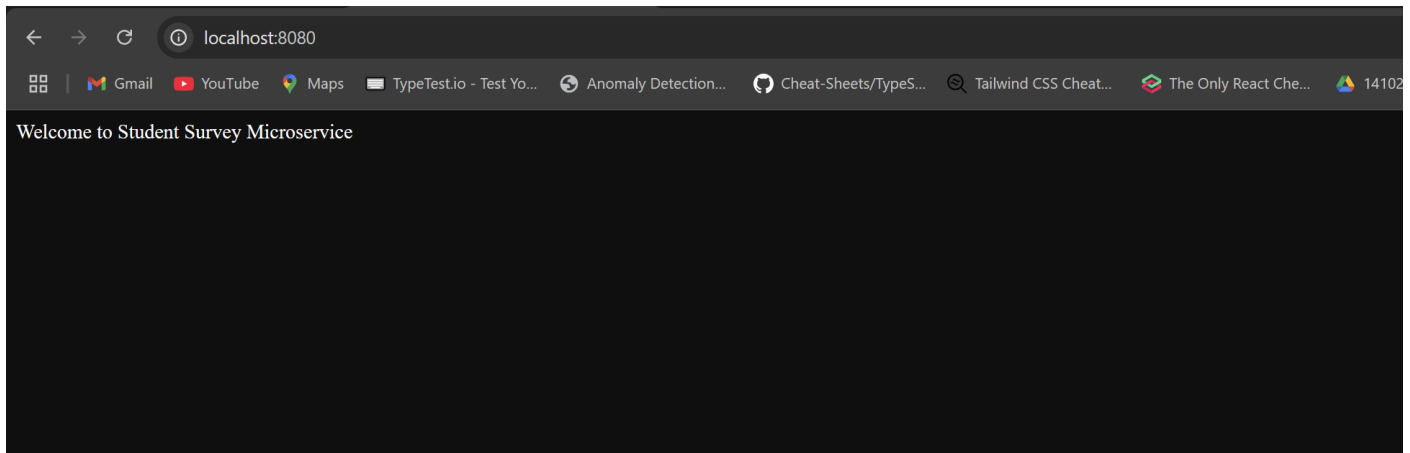
- After you have made sure that the application runs in your local environment, you can proceed to upload the image to Docker Hub.
- Use the following commands to log in and upload your image from the same directory:
  docker login -u dheerajkrishna141
- Enter Password. Tag and push the image.
  docker push dheerajkrishna141/extra_credit:latest

- Once completed, the image will be successfully uploaded and visible in your Docker Hub account.

2. Setting Up an AWS EC2 Instance for Deploying an Application on a Kubernetes Cluster via Rancher

- At this stage, you will be creating and configuring an AWS EC2 instance.
- Log into your AWS account (for instance, AWS Academy), select the EC2 service, and reach the EC2 dashboard.
- Click on "Launch Instance" to create a new instance.
- Using the following settings:
  - ❖ Instance Name: my-k8s-instance
  - ❖ Instance Type: t3.large
  - ❖ AMI: Ubuntu Server (SSD Volume Type)
  - ❖ Key Pair: one of your existing key pairs (for example, my-keypair)
  - ❖ Check HTTP and HTTPS for traffic allowed from the internet.
  - ❖ Increase storage, which is 8GB by default, to 30GB.
  - ❖ Click on "Launch instance" when finished.

- This is where you associate an Elastic IP with the instance running so you can keep that IP unchanged even after a reboot:
- From the EC2 menu, go to Elastic IPs to allocate a new IP (the default settings work fine), and associate it with your instance.
- This helps avoid the possible mess caused by changing IPs that disrupt services like Jenkins or Rancher after a restart.

Elastic IP addresses (2)   Actions ▼   **Allocate Elastic IP address**

	Name	Allocated IPv4 addr... ▽	Type	Allocation ID	Reverse DNS record	Associated instan
☐	swe_645_jenkins	35.170.214.169	Public IP	eipalloc-0111ec1e3f386b3ee	–	i-0004ec21af8fd6
☐	swe_645EI	52.55.96.9	Public IP	eipalloc-031f0fe0c04cb7da1	–	i-0bb7643ea3a2ff

- Security configuration:
- Once the requisite state is obtained, click the instance, then go to the Security tab; then click for the Security Groups settings.
- In the Inbound Rules section, edit the rules so that a new one can be added.

  - ❖ Type: Custom TCP
  - ❖ Port Range: 8080
  - ❖ Source: Custom – 0.0.0.0/0
  - ❖ Save the changes.

EC2 > Security Groups > sg-0e4703b4b5018c1b1 - launch-wizard-1

Details

Security group name	Security group ID	Description	VPC ID
launch-wizard-1	sg-0e4703b4b5018c1b1	launch-wizard-1 created 2024-09-30T21:38:02.030Z	vpc-0a2c35ef285b9cf0c

Owner	Inbound rules count	Outbound rules count	
316183840998	5 Permission entries	1 Permission entry	

Inbound rules | Outbound rules | Sharing – new | VPC associations – new | Tags

Inbound rules (5)   Manage tags   **Edit inbound rules**

	Name	Security group rule ID ▽	IP version ▽	Type ▽	Protocol ▽	Port range
☐	–	sgr-09d6e13b789e98fc4	IPv4	Custom TCP	TCP	8080
☐	–	sgr-0c5b9c86fd68f8e6a	IPv4	MYSQL/Aurora	TCP	3306
☐	–	sgr-055945150f6126665	IPv4	HTTP	TCP	80
☐	–	sgr-00888d382d57e5472	IPv4	SSH	TCP	22
☐	–	sgr-0a4d5ca5815dac0da	IPv4	HTTPS	TCP	443

(Left sidebar: Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, ▼ Images, AMIs, AMI Catalog, ▼ Elastic Block Store, Volumes, Snapshots, Lifecycle Manager, ▼ Network & Security, Security Groups, Elastic IPs, Placement Groups, Key Pairs, Network Interfaces, ▼ Load Balancing)

- Now, connect to the instance.
- Click Connect, select EC2 Instance Connect, type root as the username, and click Connect. A terminal window will show up in a new tab.
- Install Docker on the EC2 instance by executing the following commands in the terminal:
  "sudo apt update", followed by *"sudo apt install docker.io -y"*

3. Installing Rancher and Accessing Rancher

- Open any web browser and link to https://www.rancher.com/quick-start.
- Scroll down to the section "Start the Server" under "Deploy Rancher".
- Copy the Docker command given there. It looks like:

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

- Reconnect to the EC2 instance (say, my-k8s-instance) using the same steps as explained before.
- In the terminal, paste and run the copied command to install Rancher on the instance.
- When the installation is done, Rancher will be up and running on the Public IPv4 DNS address of the instance.
- Run the command to check if Rancher is running in the container:

  sudo docker ps

- Make a note of the container ID printed in the output, as you will need it later.
- On the EC2 Instances page, click the Public IPv4 DNS URL to open it in a web browser. If you see a warning message stating that the connection is not secure, choose to Proceed Anyway.
- The Rancher login screen should be displayed.
- To get the initial password for logging in, run this command in your terminal, making sure to substitute <container_id> with the actual container ID from the last step:
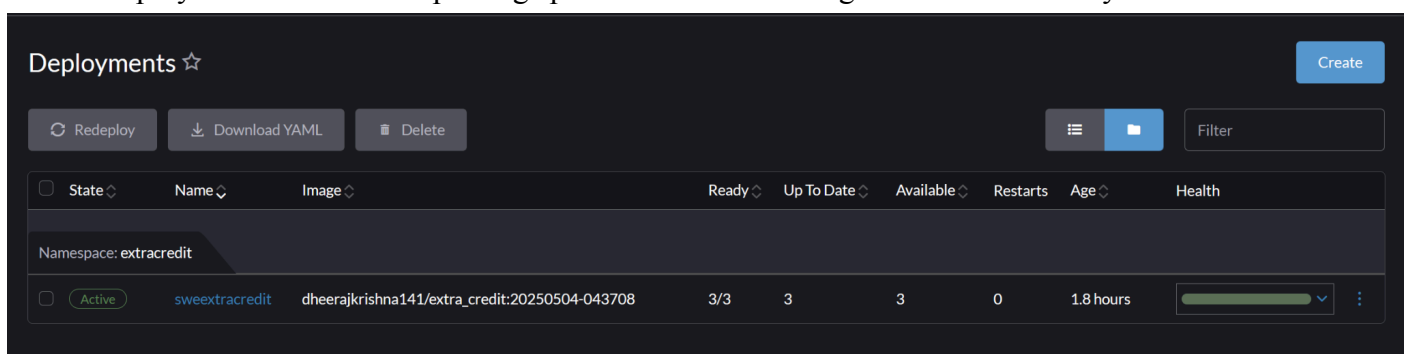
  sudo docker logs <container_id> 2>&1 | grep "Bootstrap Password:"

- This will print the password in your terminal. Copy it, and on the login page, enter it to log in to Rancher.
- Set a new password for yourself upon logging in, as prompted.
- The Rancher Dashboard should appear after the setup is completed, and you can continue creating Kubernetes clusters to deploy your application.
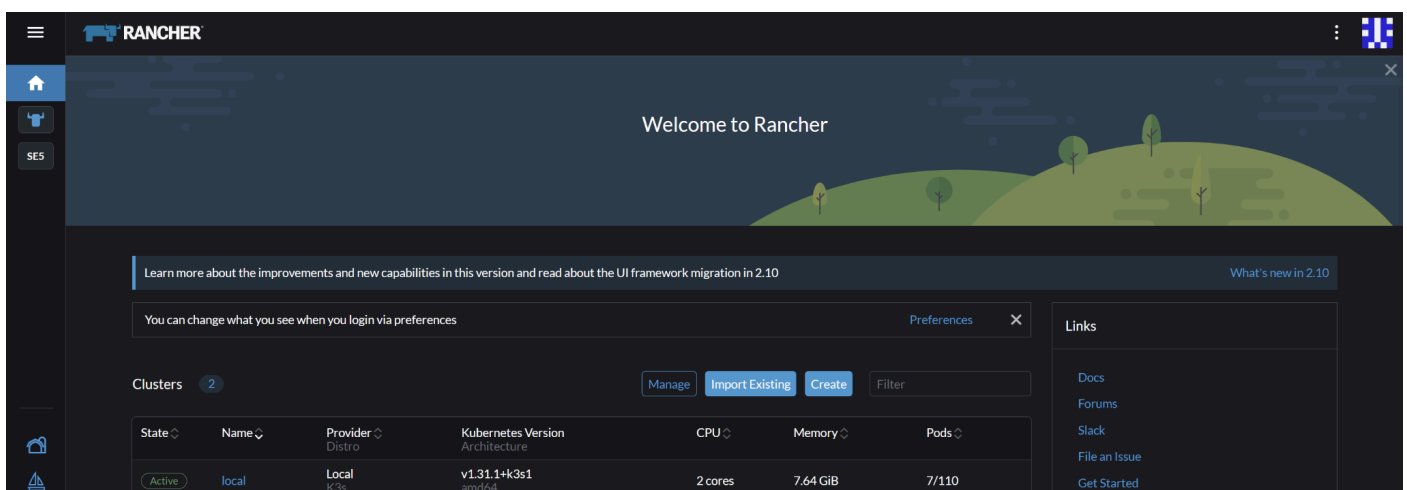
4. Deploy Application and Create Kubernetes Cluster Deploying Within Rancher UI

- Now, having Rancher running, let us go ahead and create a Kubernetes cluster while deploying the application using the Rancher interface.
- On the Rancher dashboard, click the Create Cluster button. Select the Custom option from the list of cluster types to open the cluster setup form.
  #rancher homepage pic

- In the form, set the name for the cluster as "swe645hw3", leave all other fields unchanged, and click Create.
- In the next screen, go to the Registration tab. Under Step 1, check all the three checkboxes: etcd, control plane, and worker.
- Then, copy the command that is seen in Step 2.
- Before running the copied command, place --insecure right after the word curl in the command string. Now execute it on your instance terminal.
- If this command executes successfully, the cluster will be starting in the "Updating" state and after a few minutes will change to "Active." Monitor this under "Clusters" within the Cluster Management section of your Rancher UI.
- Once the cluster reaches the "Active" status, click on "Explore" next to the cluster.
- Now, on the left-hand menu, go to Workloads > Deployments.
- Then click on the Create Button in order to bring up the deployment configuration form. Fill it out as seen below:
  - Namespace: default
  - Name: microservice
  - Replicas: 3 (for creating three pods)
  - Container Image: dheerajkrishna141/extra_credit

- Under Networking, click Add port or service. Set Protocol: TCP Container Port: 8080 Service Type: NodePort Name: nodeport Leave the remaining settings as-is but click Create.
- Deployment enters the "Updating" phase and should change to "Active" shortly thereafter.

- Once active, click on the deployment entry and go to the "Services" tab.

- Click the "nodeport" link under the Target column to open the application in a new browser tab.
- Test the app by appending /api/survey/all to the URL and opening the link. You should see the application running.


```
- student_surveys: [
    - {
        id: 1,
        street_address: "2943 Borer Field",
        state: "TG",
        phone_number: "761-726-6265",
        date_of_survey: "2025-01-31",
        interest_source: "OTHER",
        city: "Stoltenbergtown",
        first_name: "Monserrat",
        last_name: "fury",
        zip: "22030",
        email: "Adrienne_Bechtelar64@yahoo.com",
        - liked_most: [
            "STUDENTS"
        ],
        recommend_likelihood: "LIKELY"
    },
```

5. Setting Up an AWS EC2 Instance for Jenkins Installation and Configuration

- In this tutorial, we shall make use of Jenkins and build a CI/CD pipeline intended for the purpose of automatically building and updating our application regardless of the source code changes.
- First of all, we shall provision an EC2 instance in our AWS lab environment on which Jenkins will be installed.
- Go ahead and create the instance in nearly the same way you did for the two EC2 instances for the application, but this time assign it the name "swe645-hw3-jenkins." Do not forget to associate it with an Elastic IP and update the security group appropriately.
- Once the instance is created and up, connect via EC2 Instance Connect.
- Ensure that the instance is in "Running" state beforehand.
- Jenkins requires the presence of Java, so we will issue the installation command:

      sudo apt update
      sudo apt install openjdk-21-jdk -y

- Install Jenkins using the following command:

```
sudo wget -O /usr/share/keyrings/jenkins-keyring.asc https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
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
sudo apt-get update
sudo apt-get install jenkins -y
```

- Start Jenkins by running:
  - *sudo systemctl start jenkins.service*
- Allow external access to Jenkins by opening port **8080**:
  - *sudo ufw allow 8080*
- To retrieve the initial admin password for Jenkins setup, run:
  - *sudo cat /var/lib/jenkins/secrets/initialAdminPassword*
- Next, install **snapd** and use it to install **kubectl**:
  - *sudo apt install snapd*
  - *sudo snap install kubectl --classic*
- Open the Jenkins UI by going to your EC2 instance's Public IPv4 address in a browser. Append :8080 to the IP and change https to http. For example:
  - http://3.220.79.129:8080
- Previously obtained admin password should be entered to unlock Jenkins.
- Install the recommended plugins and then create the admin user. Continue by clicking:
  - "Save and create user"
  - "Save and finish"
  - "Start using Jenkins"

You are now on the Jenkins Dashboard.

- Return to the EC2 console and initiate the Kubernetes configuration for Jenkins.
- Type in sudo su jenkins to switch to the Jenkins user.
- Use the command cd ../../ to navigate to the root directory.
- To get to Jenkins home directory, cd in /var/lib/jenkins.
- Make a hidden .kube directory by doing mkdir .kube then enter it by cd .kube.
- Create a new config file with vi config.
- Open the previously downloaded KubeConfig file within your system, copy its contents, and paste them into the config file.
- Save and exit the file by typing :wq.

```
apiVersion: v1
kind: Config
clusters:
- name: "swe645"
  cluster:
    server: "https://52.23.200.181/k8s/clusters/c-m-cksvbx9m"
    certificate-authority-data: "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUJ2akND\
      VdPZ0F3SUJBZ0lCQURBS0JnZ3Foa2pPUFFREFqkdNUnd3R2dZRFZRUUtFeE5rZVc1aGJXbGoKY\
      kdsemRHVnVaWEl0YjNBBk1TWXdKQVlEVlFRERCMWtlVzVoYldssamJHbHpkR1Z1WlhJdFkyRkFNV\
      GMwTmpNeQpOVGt3TURBZUZ3MHlOVEExTURRd01qTXhOREJhRncwdek5UQTFNREl3TWpNeE5EQmFNR\
      Vl4SERBYUJnTlZCQW9UCkUyUjVjVibUZ0YVdkOc2FYTjBaVzVsY2kxdmNtY3hKakFrQmdOVkJBTU1IV\
      1I1Ym1GdGxFXTnNhWE4wWlc1bGNpMWoKWVVVBeE56UTJNekkxT1RBd01Ga3dFd1lIS29aSXpqMENBU\
      VlJS29aSXpqMERBUWNEUWdBRVV2UlE2RmhQNWd4UwpnWGlSUDRQaysrQmlsaWtvMFM5YkFHZVN5U\
      FV6eXFnZDFxNVlSWVlFZjdvaitJcWp4WUdwcXQxSHNXYkNRWW44CjV5Z1ppPZlc4bmFOQ01FQXdEZ\
      1lEVlIwUEFFRSC9CQVFEQWdLa01BOEdBMVVkRXdFQi93UUZNQU1CQWY4d0hRWUQKVlIwT0JCWUVGS\
      E4vK3I1YUQ3Sm1HcUcvNnp4aWtGWWdLYjU5TUFvR0NDcUdTTTQ5QkFNQ0Ewa0FNRVlDSVFDMwpEa\
      HRiZExXM3BoazZLR0ZKVC9MRTFDDT2l3c09hcG5XkJXMy9wMUFlSUFJaEFMcXZSd1U2UDV3Nm1Gb\
      3JGQQ3hFCjRaR0Q0bTVSN2JWa0wrVG50WnJttd2hhdAotLS0tLUVORCBDRVJUSUZJQ0FURS0tLS0t"

users:
- name: "swe645"
  user:
    token: "kubeconfig-user-cmx45c542q:hzlpn4q8jz2vxt24px75pz59m2qg225sqpzxdwsr76w9j9zgq8p8g6"

contexts:
- name: "swe645"
  context:
    user: "swe645"
    cluster: "swe645"

current-context: "swe645"
```

- The setup of your Kubernetes configuration can be verified by executing the command kubectl config current-context; it should print the name of your cluster.
- Exit from User mode of Jenkins by typing exit in the command prompt.
- Last but not least, update the latest packages lists from the repository with sudo apt update.
- Install it using sudo apt install docker.io -y. Press Y if prompted for confirmation.
- (Optional) If need be, you may also run sudo apt-get update and again run sudo apt install docker.io to ensure that Docker is properly installed.

6) Setting up project's Github repo

For our CI/CD Pipeline to monitor and respond to any changes made to the codebase, a remote repository is required for source code storage.

- GitHub will be our choice for the version control and integration platform.
- First, create an account or sign in at https://github.com.
- After you are logged in, create a new repository with the name.
- Put all the necessary files into this repository: source files, Dockerfile, and WAR file.
- Once you have added all files to the repository, commit them so that the changes will be saved to the repository.
- The repository will be treated as a source for the Jenkins environment for CI/CD pipeline execution.
- GitHub repository URL:

  https://github.com/dheerajkrishna-gmu/extra_credit

7. Creating and Running a Fully Functional CI/CD Pipeline in Jenkins

- With all components in place, we can now start the Jenkins pipeline setup.
- Start with the Jenkins dashboard, and then select "Credentials"; go down to "Manage Jenkins" and click it.
- Under the "global" scope, click "Add credentials."
- Choose Kind: "Username with password", enter your DockerHub credentials, and set the ID to "dockerhub", then hit "Create."
- Repeat this process for your GitHub credentials, and set the ID as "github."

- Back on your Jenkins dashboard, click on the option "New Item."
- Type in the name of the item as "microservice_pipeline", select "Pipeline", and click OK.
- On the configuration page, check the "GitHub Project" box and paste your GitHub repo URL: _____
- Enable "Poll SCM", and set the schedule to * * * * *, which means that every minute Jenkins will be checking for changes.
- Scroll down to the "Pipeline" section.
- For Definition, select "Pipeline from SCM", and select SCM: "git".
- Enter the repository URL: https://github.com/dheerajkrishna-gmu/extra_credit
- Select the previously saved GitHub credentials here and edit the Branch Specifier to */main.
- Go to your GitHub repo and create a new file called Jenkinsfile and commit it.
- This should trigger an automatic build in Jenkins for this new pipeline.

```
1    // DHEERAJ KRISHNA NAGULA
2    // G01448319
3    // Akhilesh Dhavileswarapu
4    // G01472450
5    pipeline {
6         agent any
7         environment {
8         DOCKERHUB_CREDENTIALS = credentials('docker-pass')
9         BUILD_TIMESTAMP = "${new Date().format("yyyyMMdd-HHmmss")}"
10        }
11
12        stages {
13             stage("Building the Student Survey Image") {
14                  steps{
15                       script {
16                            sh 'sudo docker build -t dheerajkrishna141/extra_credit:$BUILD_TIMESTAMP .'
17                       }
18                  }
19             }
20
21             stage("Login to Docker Hub"){
22                  steps{
23                       script{
24                            sh 'sudo echo $DOCKERHUB_CREDENTIALS_PSW | docker login -u $DOCKERHUB_CREDENTIALS_USR --password-stdin'
25
26                       }
27                  }
28             }
29             stage("Pushing Image to DockerHub") {
30                  steps {
31                       script {
32                            sh 'sudo docker push dheerajkrishna141/extra_credit:$BUILD_TIMESTAMP'
33                       }
34                  }
35             }
36
37
38             stage("Deploying to Rancher") {
39                  steps {
40                       sh 'kubectl set image deployment/sweextracredit container-0=dheerajkrishna141/extra_credit:$BUILD_TIMESTAMP -n extracredit'
41                       sh 'kubectl rollout status deployment/sweextracredit -n extracredit'
42                  }
43             }
44
45
46        }
47
48    }
```

- Edit the Jenkinsfile in the GitHub repository to define the pipeline stages for building the JAR file, DockerHub login, Docker image push, and Kubernetes cluster deployment.
- Commit these changes, and Jenkins will take action on these instructions with every commit.
- You will face problems with path or configuration in a few of the early builds; fix them in the Jenkinsfile.

Build Time Trend

S	Build ↑	Time Since	Duration	
✓	#10	3 hr 16 min	29 sec	▶_
✓	#9	3 hr 21 min	28 sec	▶_
✕	#8	3 hr 28 min	20 sec	▶_
✕	#7	3 hr 33 min	14 sec	▶_
✕	#6	3 hr 43 min	14 sec	▶_
✕	#5	3 hr 50 min	14 sec	▶_
✕	#4	3 hr 57 min	28 sec	▶_
✕	#3	4 hr 2 min	5 sec	▶_
✕	#2	4 hr 4 min	8.1 sec	▶_
✕	#1	4 hr 6 min	1.3 sec	▶_
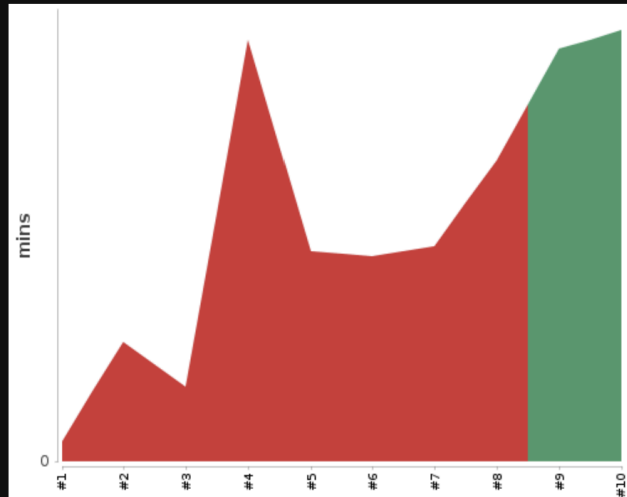
Icon: S   M   L

- When the build is successful, use the same URL and port on which the Kubernetes cluster was configured to access the deployed application from the browser.

https://52.55.96.9/k8s/clusters/c-m-qxnwtnz4/api/v1/namespaces/hw-1-namespace/services/http:microservice-loadbalancer:8080/proxy/api/survey/all

```
[
- {
    id: 1,
    firstName: "Robyn",
    lastName: "fury",
    streetAddress: "7527 Breitenberg Heights",
    city: "West Hayden",
    state: "TG",
    zip: null,
    phoneNumber: "682-388-9143",
    email: "Clementine31@gmail.com",
    dateOfSurvey: "2025-01-31",
  - likedMost: [
        "STUDENTS"
    ],
    interestSource: "OTHER",
    recommendLikelihood: "LIKELY"
  }
]
```

- Check the CI/CD with a change in a source file in GitHub and commit it-the pipeline should be triggered and deploy the updated application.

URL/Links of the application deployed:

1. GitHub URL : https://github.com/dheerajkrishna-gmu/extra_credit

2. Dockerhub URL:
https://hub.docker.com/repository/docker/dheerajkrishna141/extra_credit/general

3. Application deployed on cluster URL :
https://52.23.200.181/k8s/clusters/c-m-cksvbx9m/api/v1/namespaces/extracredit/services/http:sweextracredit:8080/proxy/api/survey/all

4. Nodeport URL with the instance public ipv4 DNS:
http://ec2-52-23-200-181.compute-1.amazonaws.com:30761/