**Swe645-hw2**

**Chao Chen G01476331**

1. **Introduction:**

For this assignment, I completed the establishment of a CI/CD pipeline by studying DevOps, git, docker, Kubernetes and Jenkins, and through hands-on practice. When the pipeline was set up, I had just completed the second task of swe642. Therefore, the webpage I used was an updated version of the webpage for assignment 1. I optimized the style and presentation and added functions in cookies and ajax.

<http://54.235.178.80:30080/cs-survey.html>

There are 5 html files, 3 jpg files, 1 dockerfile, 1 jenkinsfile, 1 JSON file, 1 CSS file, 1 yaml file, 1 Readme file, and 2 video files in the zip file. One of the videos is about the process of creating a CI/CD pipeline, and the other one demonstrates how Jenkins automatically updates the pipeline after updating the files in GitHub, and then you will see different results when accessing the website.

1. **Project Steps Overview**

**Step1: Prepare the Project Files:**

Repository URL: <https://github.com/cc2024upup/swe645-hw2>

1. Created a GitHub repository swe645-hw2
2. 图形用户界面, 应用程序

   AI 生成的内容可能不正确。

Click New

1. 图形用户界面, 文本, 应用程序

   AI 生成的内容可能不正确。

Enter the name of the repository(swe645-hw2), select add a README file and click create repository.

1. Subsequently, I organized and uploaded the static web page files (including HTML, CSS, images, etc.) that I had completed in swe642 hw2 to this GitHub repository.

**Step2：Create a new image and push it to Docker Hub**

1. After completing the upload of the web page files, I created a Dockerfile file in the project root directory to containerize the static website.

I use VS Code to create the Dockerfile file. The file doesn't need an extension. The full name of the file must be "dockerfile".

文本

AI 生成的内容可能不正确。

1. I used the terminal in Docker Desktop to execute the following command to create a new image:

**docker build -t chaochen46/survey-app:latest .**

1. After you create the image, you can check and confirm it through the command:

**docker images**

图形用户界面, 文本, 应用程序

AI 生成的内容可能不正确。

The picture shows that the image I wanted has been successfully built: chaochen46/survey-app

1. Run in PowerShell:

**docker run -d -p 8080:80 chaochen46/survey-app**

-d: Run container in the background

-p 8080:80: Map the local port 8080 to the internal port 80 (default for Nginx) within the container

1. Check whether the container is running successfully:

**docker ps**

文本

AI 生成的内容可能不正确。

1. Access the local website in the browser

[**http://localhost:8080**](http://localhost:8080)

**if was successfully, you could see:**

图形用户界面, 文本, 应用程序

AI 生成的内容可能不正确。图形用户界面, 文本, 应用程序, 电子邮件

AI 生成的内容可能不正确。

1. After the image build was successful, I logged into DockerHub and uploaded the image to DockerHub：

**docker login**

**docker push** **chaochen46/survey-app:latest**

if was successfully, you can check your docker hub and see:

图形用户界面, 应用程序

AI 生成的内容可能不正确。

8) At this point, we have created the image and uploaded it to DockerHub, but we haven't yet used EC2.

**Step 3: Deploying the Containerized Application on Kubernetes via Rancher**

1. Use the account provided by the professor to create an instance, give it a name, select Ubuntu, t2.large, 30GB, and add security groups as needed. This is my final security group. I will explain their functions one by one later:

图形用户界面, 应用程序

AI 生成的内容可能不正确。

1. After creating the instance, we connect to the EC2 instance and use the following command to update and download Docker to the EC2：

**$ sudo apt-get update**

**$ apt upgrade -y**

**$ sudo apt install docker.io**

1. After installing Docker, start the Rancher management interface through the Docker container:

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

--privileged \ # Grant the container more kernel control permissions (for underlying system operations)

-d \ # Run the container in the background

--restart=unless-stopped \ # Automatically restart the container if it crashes or the server restarts

-p 80:80 \ # Map the host's 80 port to the container's 80 port (HTTP)

-p 443:443 \ # Map the host's 443 port to the container's 443 port (HTTPS)

rancher/rancher:stable # Use the stable version image of Rancher

1. After the startup is successful, you can enter the public IP of your EC2 instance in the browser:

My public IP: 54.235.178.80

My URL: http:// 54.235.178.80:80 #-p 80:80

https:// 54.235.178.80:443 #-p 443:443

图形用户界面, 文本, 应用程序, 电子邮件

AI 生成的内容可能不正确。

1. The rancher added: Since I wrote the readme after completing the entire process, my page can be directly accessed. If you are entering for the first time, you will first enter the Rancher UI. First, on the EC2 instance terminal, use the "docker ps" command to obtain your container ID. Then, please copy the password command on the Rancher UI page and paste it onto the EC2 instance terminal. Note that you need to have "sudo" before the command. Replace the container ID area in the command with the obtained container ID.
2. Then you can obtain the password for the Rancher page. After logging in to Rancher with this new password, it will prompt you to set your preferred password. Please note that your username remains "admin", which has not been changed.
3. At this point, you need to go back to the EC2 instance, create a new Elastic IP, and bind the public IP to the Elastic IP. This way, when the EC2 instance is started or stopped and restarted, the public IP will not change.

文本, 电子邮件

AI 生成的内容可能不正确。

1. Now, let's start creating the cluster on Rancher. Select the 'Cluster Manager' from the left column, then choose 'Create', and select 'Custom'

图形用户界面, 应用程序, 网站

AI 生成的内容可能不正确。

1. Add a cluster name for the first page, and then hit next.

图形用户界面, 应用程序, 电子邮件

AI 生成的内容可能不正确。

1. Click the three checkboxes, a radio option and copy that command it gives you

图形用户界面, 文本, 应用程序, 电子邮件

AI 生成的内容可能不正确。

1. Copying that command, go to your terminal connected to instance2 and paste it within.
2. When the operation is completed, you will be able to see Active in front of your k8s:

图形用户界面, 网站

AI 生成的内容可能不正确。

1. I chose to use the kubectl tool to create new Kubernetes objects in the newly created cluster. So, first of all, I need to download and install kubectl on Ubuntu.

%snap install kubectl

1. You will also need to save the cluster configuration file in ~/.kube/config file on your workstation. You need to select your cluster on the Rancher cluster manager page, choose the config option in the upper right corner, then select the button with three dots on the right side, and choose the "Copy KubeConfig to Clipboard" option.

图形用户界面, 应用程序

AI 生成的内容可能不正确。

1. In the EC2 instance terminal, we need to execute the command: `mkdir .kube` to create a folder named `.kube`. Then, `cd .kube` to enter the folder. Next, use the `vi config` command to create and open a config file and input the content "Copy KubeConfig to Clipboard" that was just copied. After that, `:wq` to save and exit the config file to ensure that the `kubectl` command can be executed correctly.

文本

AI 生成的内容可能不正确。

1. After the settings are completed, go one level up in the CD, and use the command "kubectl get node" to check if the kubectl command is effective and if your Kubernetes cluster is functioning properly. If possible, you can see:文本

   AI 生成的内容可能不正确。
2. You can use Rancher UI to install your containerized image, but you should prefer using kubectl – Click the explore button on the cluster we created. – Then click workload, click deployments and click create.
3. Name, input the image you have stored on DockerHub, add Ports, select NodePort, and set the port number to 8080. You should see this:

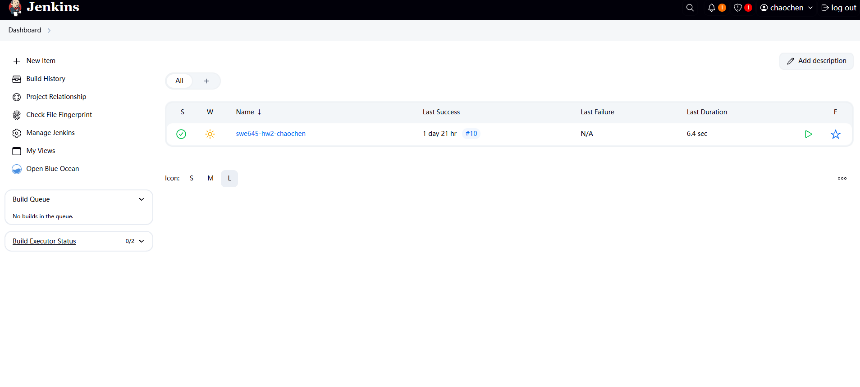
图形用户界面, 文本, 应用程序, 电子邮件

AI 生成的内容可能不正确。

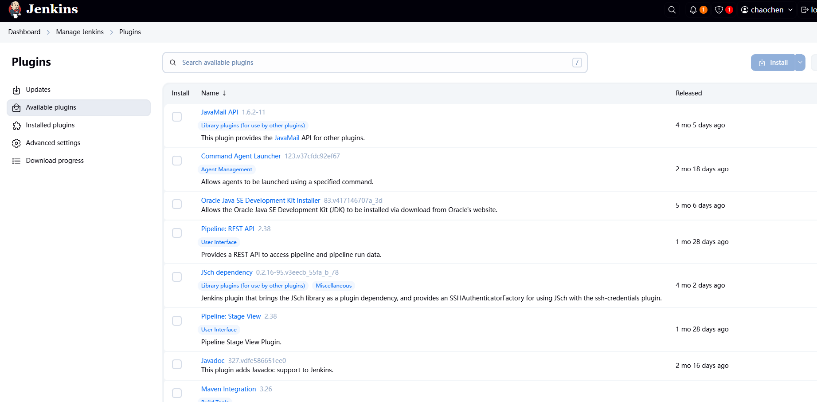
1. Or you can also manually create a YAML file yourself, create the deployment and the corresponding pods, and then Jenkins will automatically complete the resource deployment and load it into the Kubernetes cluster managed by Rancher.

Step4: Install the Jenkins plugin and use the Jenkins tool to complete the deployment of the CI/CD pipeline.

1. First of all, we need to download Java on Ubuntu because Jenkins requires Java.
2. We went to this website. <https://www.jenkins.io/doc/book/installing/linux/#debianubuntu>
3. copy the command for Long Term Support release from this URL and run on the command line
4. After downloading, go to your public IP address: 8080. This is the default Jenkins website address.
5. Get the password on the terminal using cat command: • $ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
6. Select to download the recommended plugin, set your username, password, full name and email address.
7. The webpage will automatically generate the Jenkins web address, which is usually your public IP address followed by the port number 8080.



1. Go to the Jenkins homepage, then enter "Manage Jenkins" and update all the plugins. This will help reduce version errors.
2. Go to the Jenkins homepage, first enter "Manage Jenkins", update all plugins, which helps to reduce version errors and ensure the installation of Docker pipeline, Git, Kubernetes CLI and GitHub Integration.

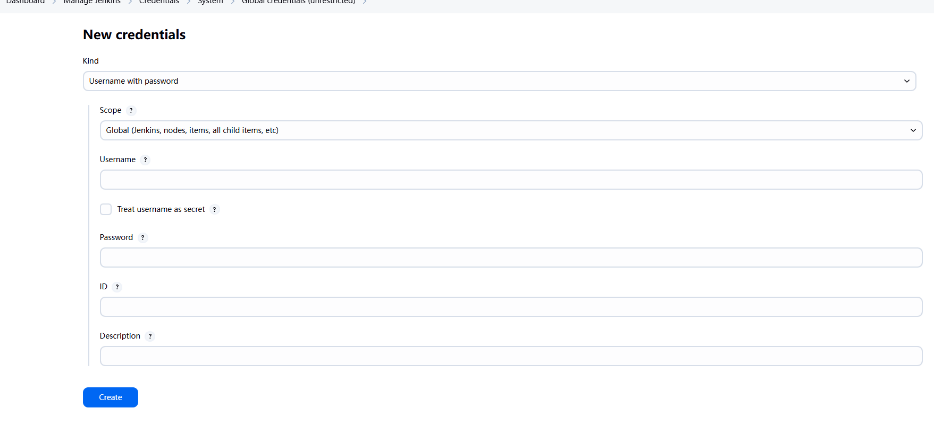


1. To enable Jenkins to automatically build and push Docker images, we need to ensure a stable connection between Jenkins and the Docker engine.
2. First, we confirm that Docker has been successfully installed in the EC2 instance where Jenkins is installed, and add the Jenkins user to the Docker user group to ensure that Jenkins can directly execute Docker commands.

**sudo usermod -aG docker jenkins**

**sudo systemctl restart jenkins**

1. Subsequently, we installed and enabled the Docker Pipeline plugin in the Jenkins management interface. This plugin supports using Docker DSL (such as docker.build() and docker.withRegistry()) in the Pipeline to build and upload images.
2. To ensure the security of the image push authentication, we added the username and password of DockerHub in the "Credential Management" section of Jenkins, and assigned a unique credential ID, such as dockerhub-credentials. In the Jenkinsfile, we safely reference this credential using the withCredentials block:



**withCredentials([usernamePassword(credentialsId: 'cc2024upup', usernameVariable: 'DOCKER\_USER', passwordVariable: 'DOCKER\_PASS')]) {**

**sh 'echo $DOCKER\_PASS | docker login -u $DOCKER\_USER --password-stdin'**

**sh 'docker push chaochen46/survey-app:latest'**

1. Since our code on GitHub is pulled from the main branch, in the stage 'Clone code', we need to write：

**stage('Build Docker Image') {**

**steps {**

**sh 'docker build -t chaochen46/survey-app:latest .'**

**}**

**}**

1. The Jenkinsfile also defines the functions for building the image and deploying it to Kubernetes:

**stage('Build Docker Image') {**

**steps {**

**sh 'docker build -t chaochen46/survey-app:latest .'**

**}**

**}**

**stage('Deploy to Kubernetes') {**

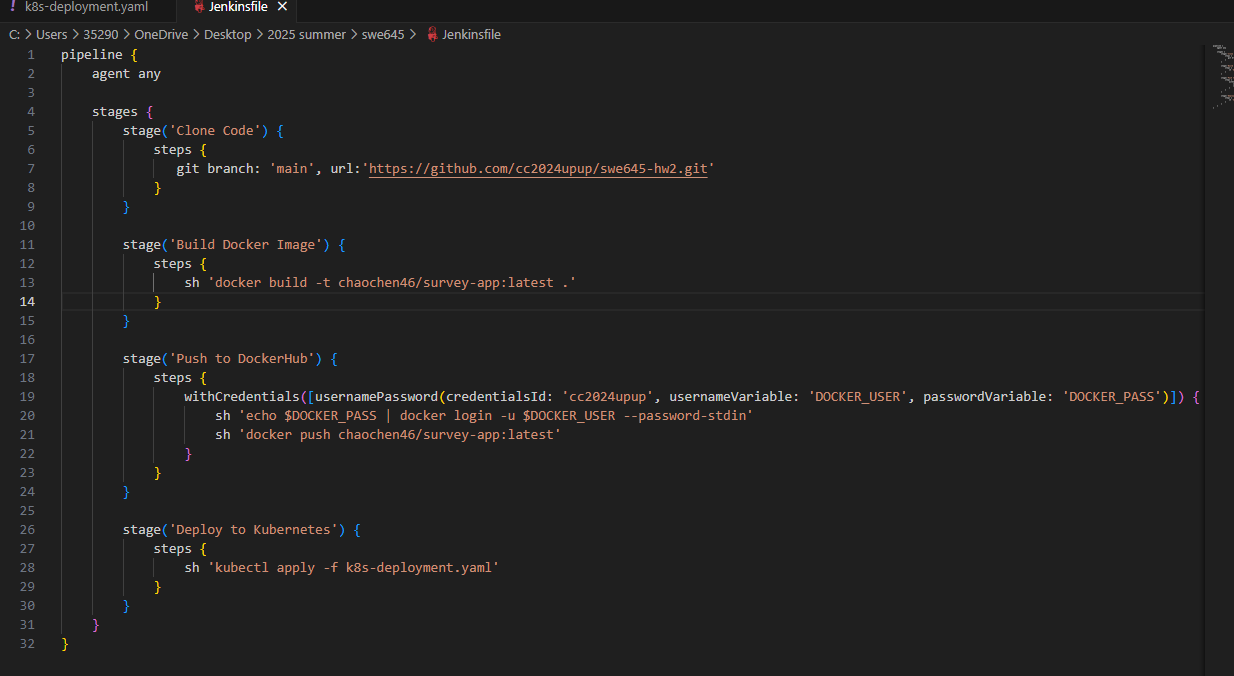
**steps {**

**sh 'kubectl apply -f k8s-deployment.yaml'**

**}**

**}**

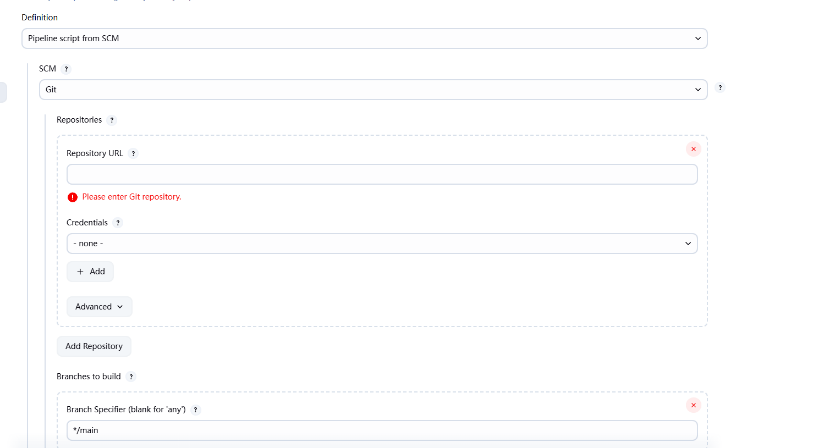
1. The final content of the Jenkinsfile is as follows:



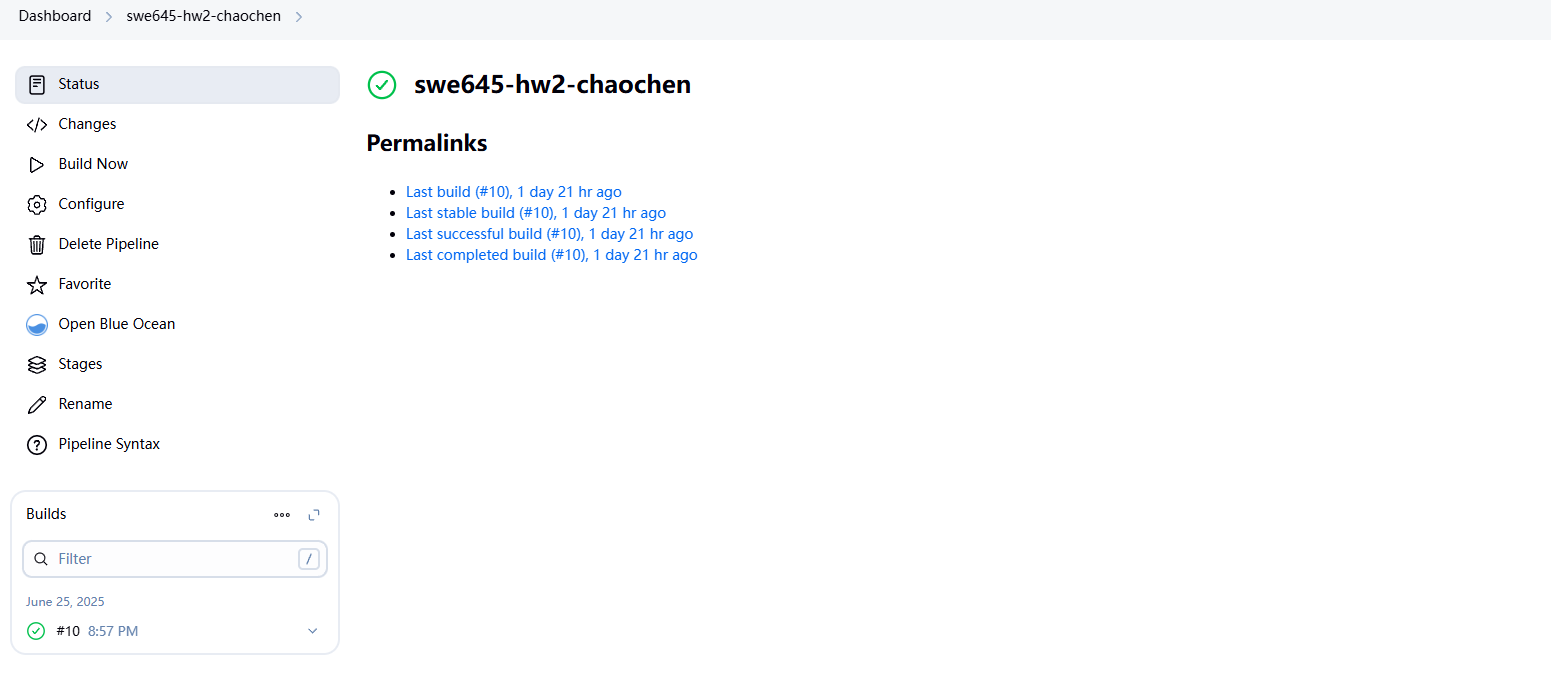
1. After creating the Jenkinsfile, we upload it to Github. Then, on the Jenkins page, we create a new project, enter the project name, select Pipeline, click Create. Next, on the new page, I select GitHub project, and add my git repo in the url section. Select Poll SCM and type ‘\* \* \* \* \*’ This will poll our repo every minute and if commits have been made, a new build will be scheduled. Then, under the Pipeline tab, we find Pipeline script from SCM. Select Git as the SCM type, fill in my repository address, and enter the branch as the /\*main mentioned in the Jenkinsfile. Keep the Jenkinsfile path as default: Jenkinsfile.

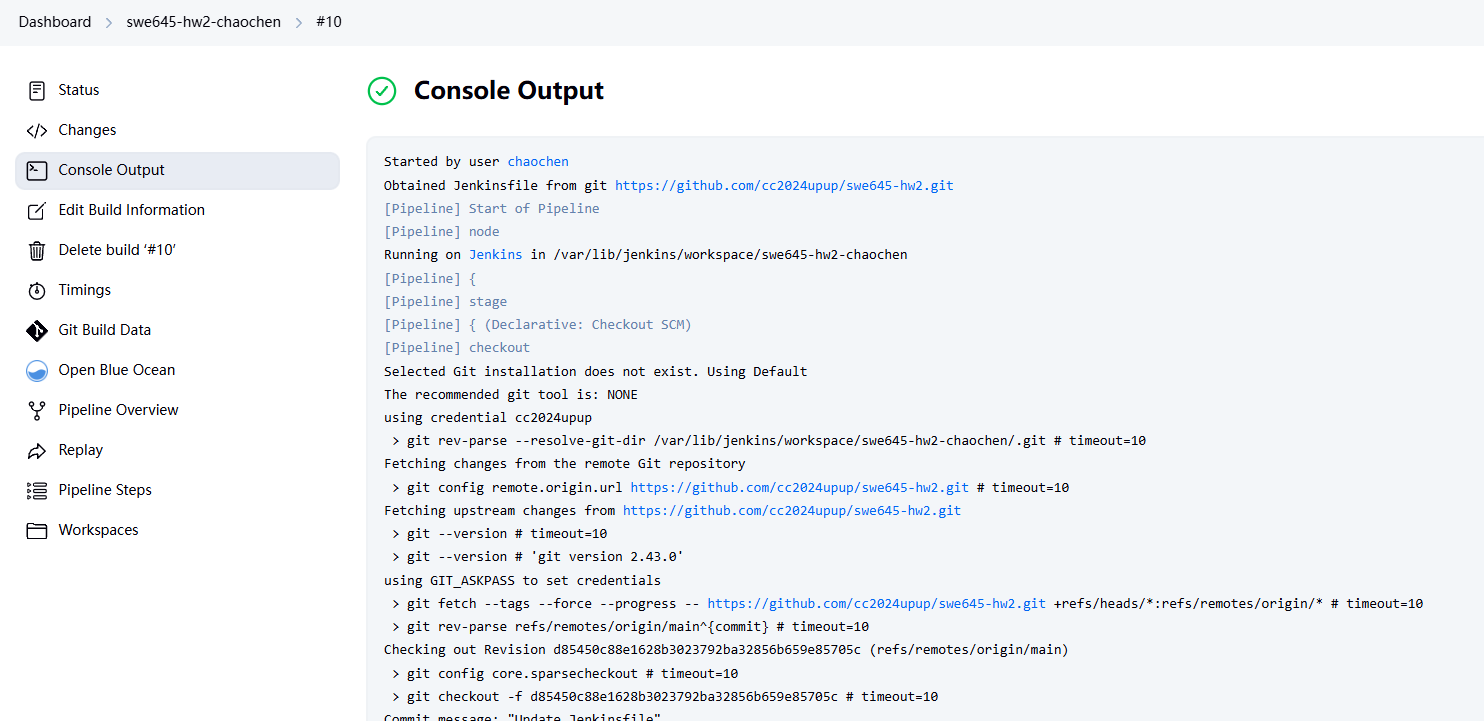
图形用户界面, 文本, 应用程序, 电子邮件, Teams

AI 生成的内容可能不正确。



1. After creating the project, we ensured that all files, including Jenkinsfile, dockerfile, html, etc., were uploaded to GitHub. Then we clicked 'Build Now'. We may not succeed at once because there will be various errors, but they are not major issues, such as the incorrect entry of the credential ID in the Jenkinsfile, or files not being uploaded, etc. We can check the console output of the build project to understand the error messages.





1. You can check the pipeline overview to observe the results of each step.

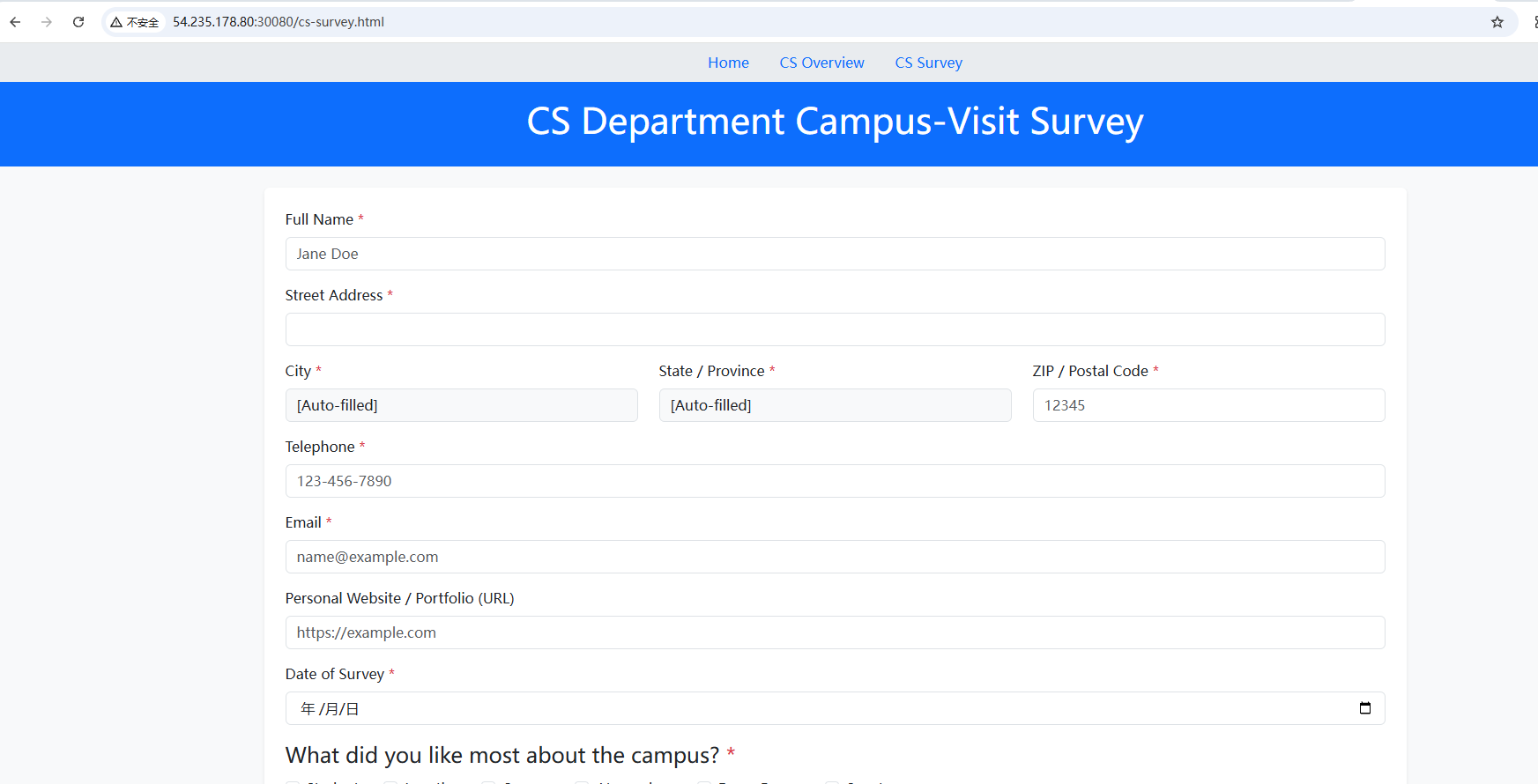
图形用户界面, 文本, 应用程序, 电子邮件

AI 生成的内容可能不正确。

1. Also access your application through the browser:

http://Public IP:nodeport

<http://54.235.178.80:30080/cs-survey.html>



PS： The input zip address discovered at the end of the demonstration video failed to automatically display the state and city correctly. The reason is that Kuberbnetes is still running old versions of Docker images. We need to modify the contents of deployment.yaml and jenkinsfile so that Jenkins always uses the latest Docker images instead of the cached ones.

deployment.yaml:

**spec:**

**containers:**

**- name: survey-app**

**image: chaochen46/survey-app:latest**

**imagePullPolicy: Always # add this line**

In the Jenkinsfile, I added the --no-cache command, and I also included the 'Clean Workspace' and 'Restart Deployment' stages each time.:

**Sh “docker build –no-cache -t chaochen46/survey-app:latest .”**

**stage('Clean Workspace') {**

**steps {**

**cleanWs()**

**}**

**}**

**stage('Restart Deployment') {**

**steps {**

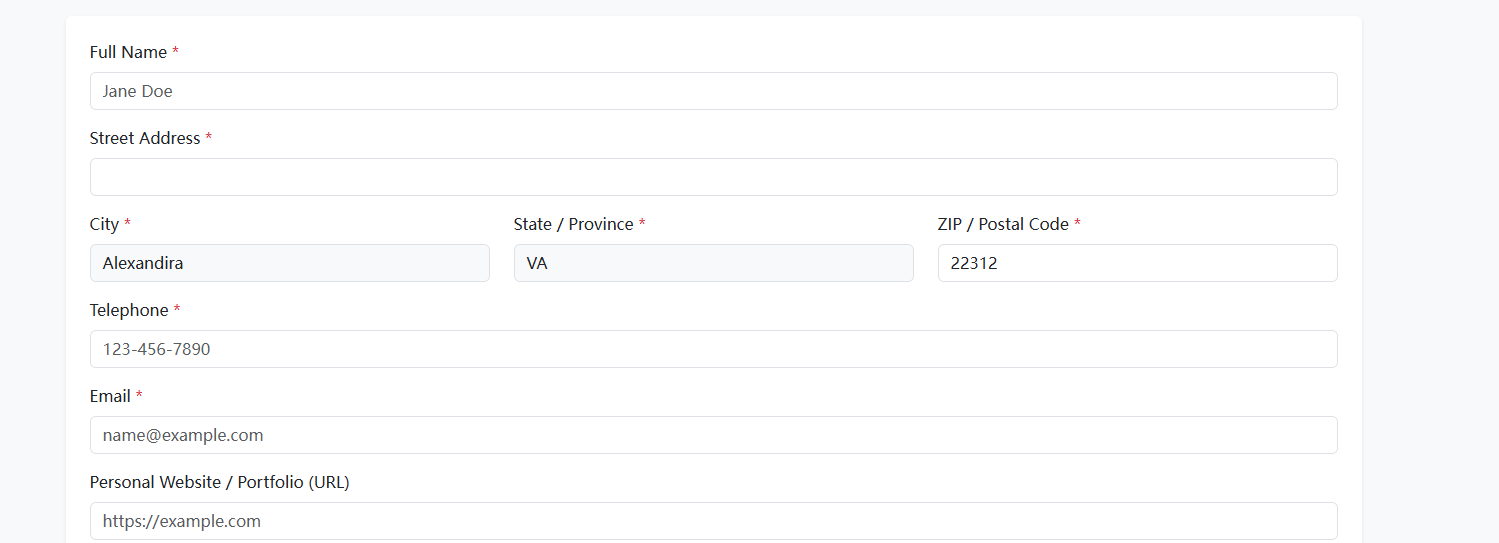
**sh 'kubectl rollout restart deployment survey-app'**

**}**

**}**

**Second video demonstration content:**

1. In the original browser, zip codes could be automatically matched with states and cities based on the input:



2.When I commented out all the Ajax code in the cs-survey.html file on GitHub and Jenkins automatically ran the pipeline, the browser could no longer automatically recognize the states and cities：图形用户界面, 文本, 应用程序, 电子邮件

AI 生成的内容可能不正确。

**Lessons Learned:**

1. When creating the dockerfile and jenkinsfile, the names must be simply "dockerfile" or "jenkinsfile" without any suffix.

2. When you build the image, the Dockerfile must be in the same directory as the required files.

3. When building the image, all the necessary files must be included. Omitting any files will result in incomplete functionality of your image.

4. When using the command to build the image, remember to add the '.' at the end, indicating that it is in the current directory.

5. After building the image, use the "ls" command to check if all the files are included in the image.

6. When building a Kubernetes cluster using Rancher, remember to check the radio button below the copy code section.

7. When creating the .kube folder, don't forget to include the prefix '.

8. After pasting the config attributes, press "Shift" + ":" to enter the input mode. Then, type "wq" to save and exit.

9. When setting up the credentials for Jenkins, please note that the username should be the username of DockerHub, the password should be the password of DockerHub, and the ID can be set by yourself.

10. When writing the Jenkinsfile, be sure to specify whether the GitHub branch is "main" or "major". During the stage of pushing to DockerHub, make sure that the ID is exactly the same as the one set in the credentials.

11. Don't confuse the Docker Desktop terminal, the EC2 instance terminal, Windows Command Prompt, and PowerShell.

12. Be aware that when you modify the file on GitHub and Jenkins automatically runs the pipeline, the web page content remains unchanged. This might be because Kubernetes is still running the old version of the Docker image.

I am very honored to have participated in the swe645 course. This is my first time building a CI/CD pipeline. I am very happy to have successfully completed it. Since the readme was written by me after completing all the content, there might not be any image explanations for some steps in the middle, such as the step of obtaining the UI password when downloading Rancher. However, there are very detailed explanations in the slides provided by the professor. Thank you very much for your reading.

Chao Chen