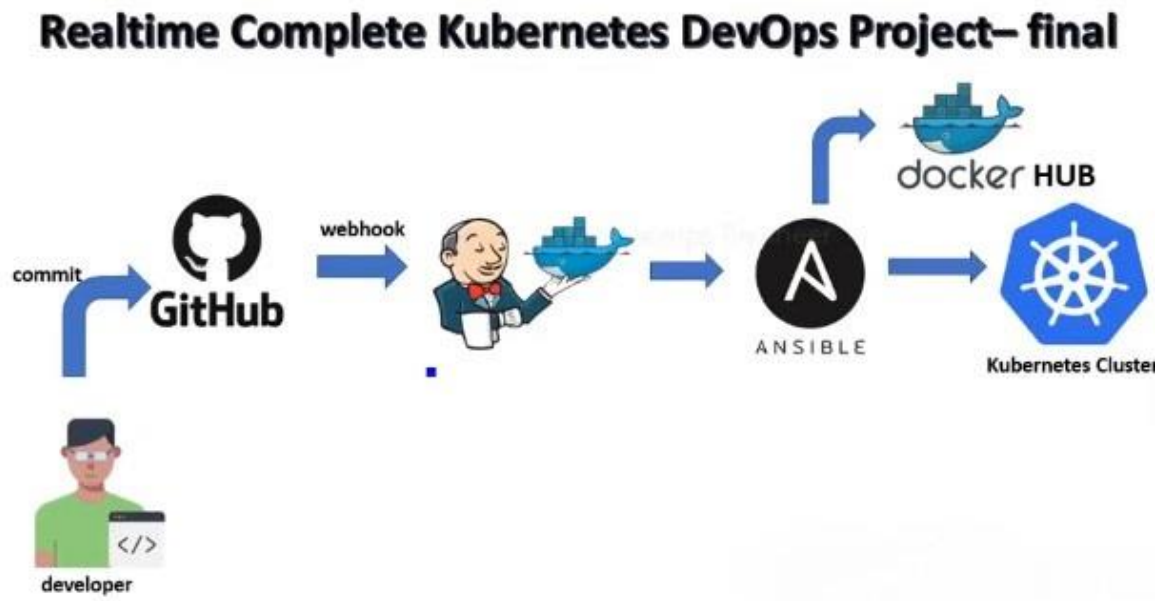


We Will be using

GitHub
Jenkins
Docker
Ansible
Minikube for Kubernetes

“This project incorporates various components to streamline the building and deployment of a Docker image to a Kubernetes cluster. It starts with the developer creating a Dockerfile and pushing it to GitHub. Jenkins triggers an automated pipeline upon receiving commits.

Jenkins connects to an Ansible server via SSH to build the image based on the Dockerfile. After successful building, the

image is tagged and pushed to a Docker registry like Docker Hub for storage and accessibility.

The Ansible server then establishes an SSH connection with the Kubernetes server and executes a playbook. This playbook utilizes kubectl commands to fetch the latest image from the Docker registry. The Kubernetes cluster creates a container for the application, which can be accessed using the assigned IP address and port specified in the Kubernetes configuration.

By automating with Jenkins and Ansible, the workflow reduces manual intervention, minimizes errors, and boosts productivity. It enables organizations to adopt continuous integration and delivery, facilitating rapid and reliable application deployment while ensuring scalability and flexibility."

Pre-Requisites:

Git, Linux,Jenkins,Docker,DockerHub Account,Ansible,
K8s(Deployment and service)

3 EC2 Instances
Jenkins (Default-jre+jenkins) (all traffic allow)(t2 micro)
Ansible (python+ansible+docker) (all traffic allow)(t2 micro)

WebApp(kubernetes cluster) → (docker+minikube)(t2 medium)(all traffic allow)

Prerequisites:

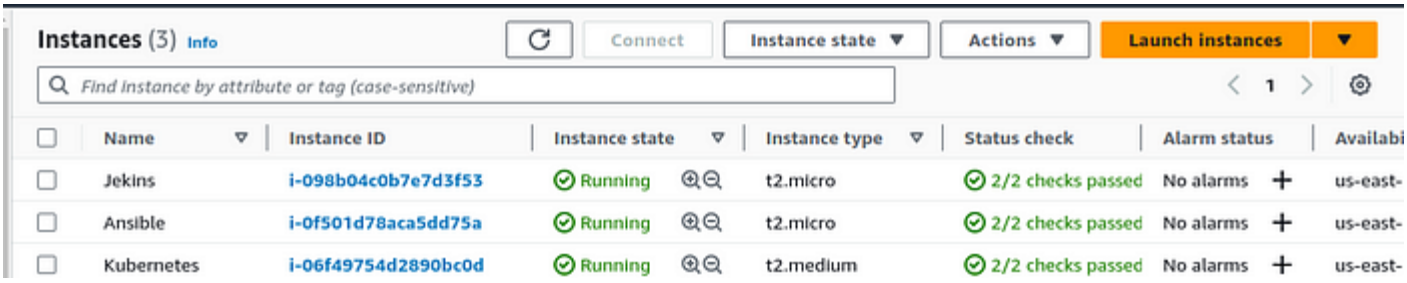
To begin this project, ensure that you have the following components and configurations in place:

1. Git, Linux, Jenkins, Docker, DockerHub Account, Ansible, and Kubernetes (Deployment and Service) should be installed and set up.
2. **Three EC2 instances are required:**

a. Jenkins: Use a t2.micro instance with the default JRE and Jenkins installed. Allow all traffic.

b. Ansible: Utilize a t2.micro instance with Python, Ansible, and Docker installed. Allow all traffic.

c. WebApp (Kubernetes Cluster): Use a t2.medium instance with Docker and Minikube installed. Allow all traffic.



Here are the simplified steps for setting up Jenkins:

SSH into the Jenkins server and follow the instructions in the documentation: Jenkins Installation Guide.

After installing Jenkins, access it using the public IP with port 8080 in your web browser.

Retrieve the Jenkins administrator password using the following command:

```
cat /var/lib/jenkins/secrets/initialAdminPassword
```

1. Install the suggested plugins and set up your admin user.
2. In the Jenkins dashboard, go to “Manage Jenkins” and install the SSH Agent plugin.
3. Restart Jenkins to apply the changes.

SSH into the Ansible server

Create a bash script file named `ansible.sh` using the command `nano ansible.sh`.

Paste the following data into the `ansible.sh` file:

```
sudo apt-add-repository ppa:ansible/ansible -y
sudo apt update -y
sudo apt install ansible -y
```

Save the file and exit the editor.

Execute the script by running the command `sh ansible.sh`.

SSH into the Kubernetes instance

Install Docker and Minikube by following the instructions provided in this GitHub repository: [AWS Kubernetes].

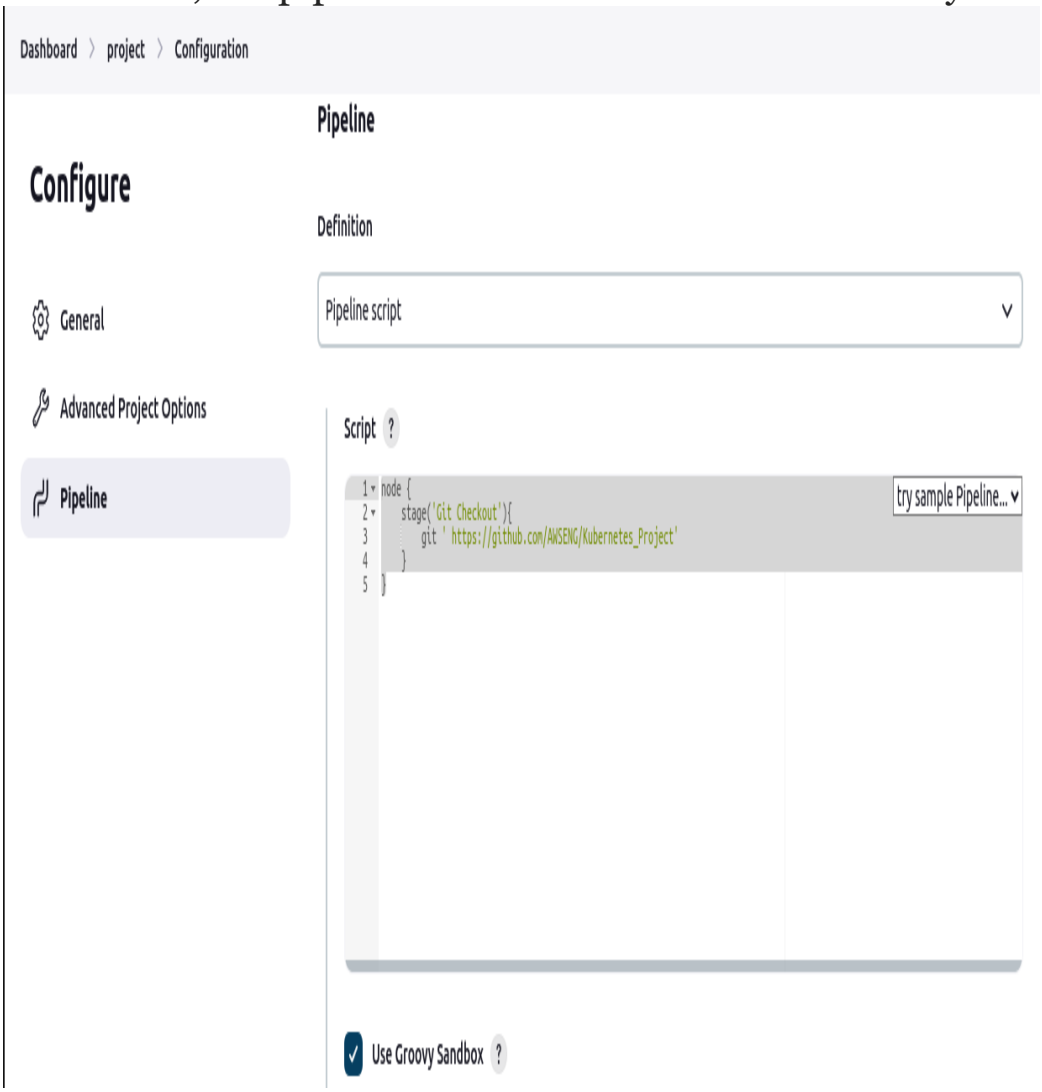
Once Docker and Minikube are successfully installed and configured, your instances are all set and ready for use.

Trigger Jenkins Job using Webhook

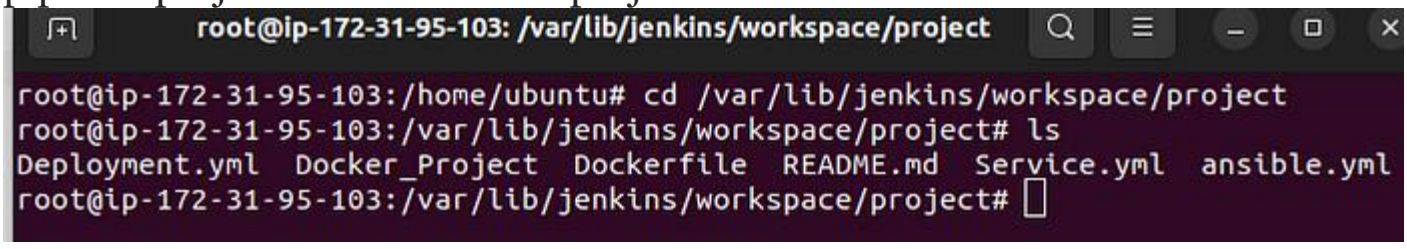
1. Set up a GitHub repository containing all the necessary content for the project. You can use the following repository: [AWS Kubernetes Project].
2. Use the provided Dockerfile from the repository. Here's the link to the Dockerfile: [Dockerfile].
3. Access your Jenkins dashboard and create a new item. Choose the pipeline option and write a Groovy script.

```
node {
    stage('Git Checkout'){
        git ' https://github.com/AWSENG/Kubernetes_Project'
    }
}
```

Apply and save the pipeline configuration. Trigger the Jenkins job by clicking the ‘Build Now’ option. You will see the execution status, and if it succeeds, the pipeline has been executed successfully.



Get back to your Jenkins server and run this command with your pipeline project name to see the project.



To automate the process using a webhook, follow these steps:

1. Go to the configuration section of your pipeline project in Jenkins.
2. Check the box for “GitHub hook trigger for GITScm polling”.
3. Apply the changes and save

To integrate Jenkins with your GitHub repository:

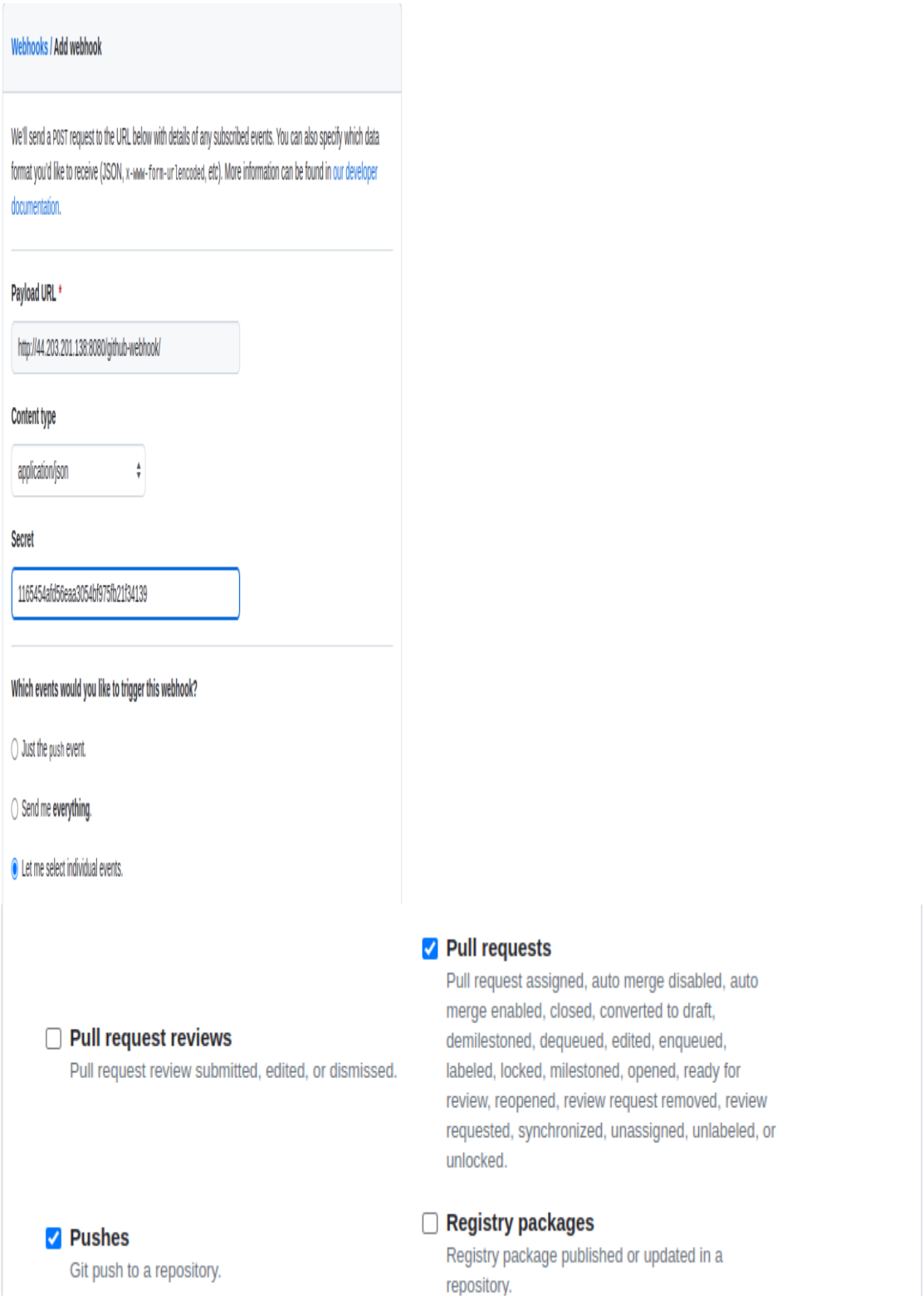
1. Go to the repository settings in GitHub.
2. Add a new webhook.
3. Specify the Jenkins server’s public IP followed by port 8080, and append “/github-webhook/” to the URL.
4. Save the webhook configuration.

Build Triggers

- [ ] Build after other projects are built ?
- [ ] Build periodically ?
- [x] GitHub hook trigger for GITScm polling ?
- [ ] Poll SCM ?
- [ ] Quiet period ?
- [ ] Trigger builds remotely (e.g., from scripts) ?

Advanced Project Options

Save Apply



To add a secret in Jenkins:

1. In the Jenkins configuration section, go to “Manage Jenkins” -> “Manage Users”.

2. Create a token from the “API Token” section.

3. In the pipeline project configuration, add the token as a secret.

4. Set up the webhook in the GitHub repository. Once configured, manual intervention won’t be necessary anymore. The pipeline will be triggered automatically through the webhook.

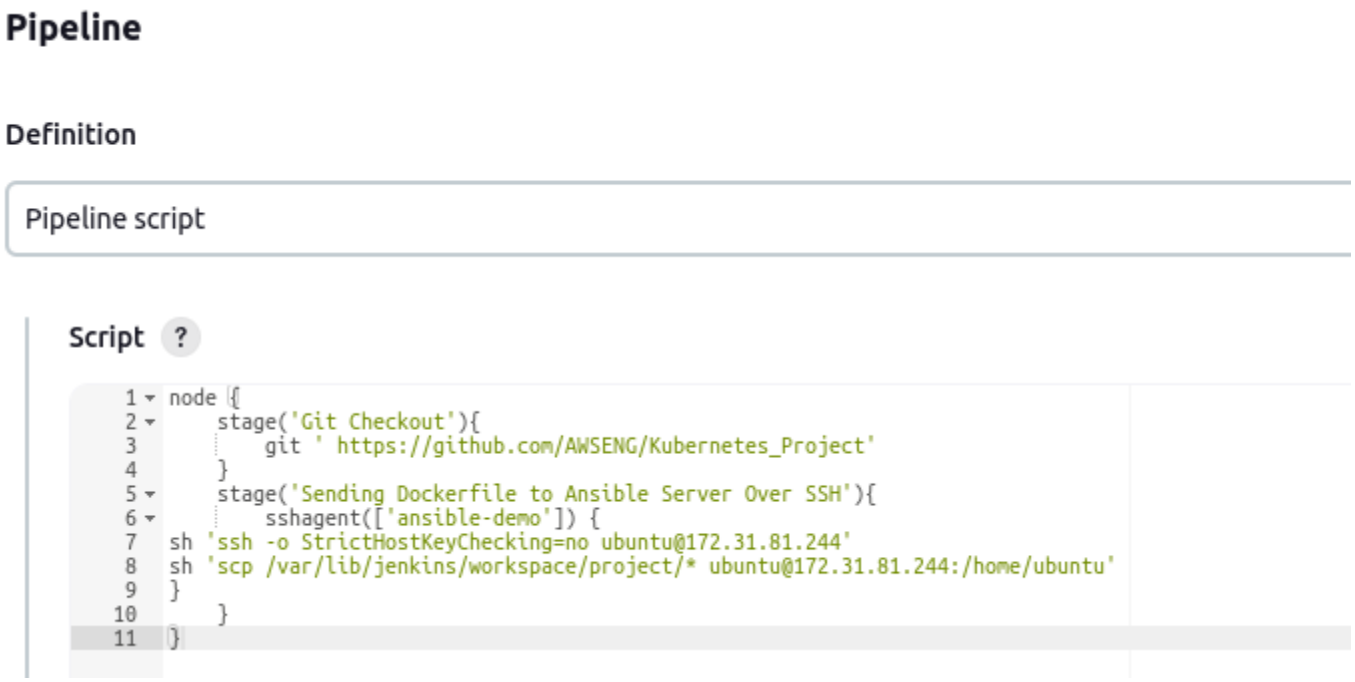
To work on Ansible and send the Dockerfile to the Ansible server:

1. Install SSH agent in the Jenkins dashboard. This allows secure communication between Jenkins and the Ansible server.

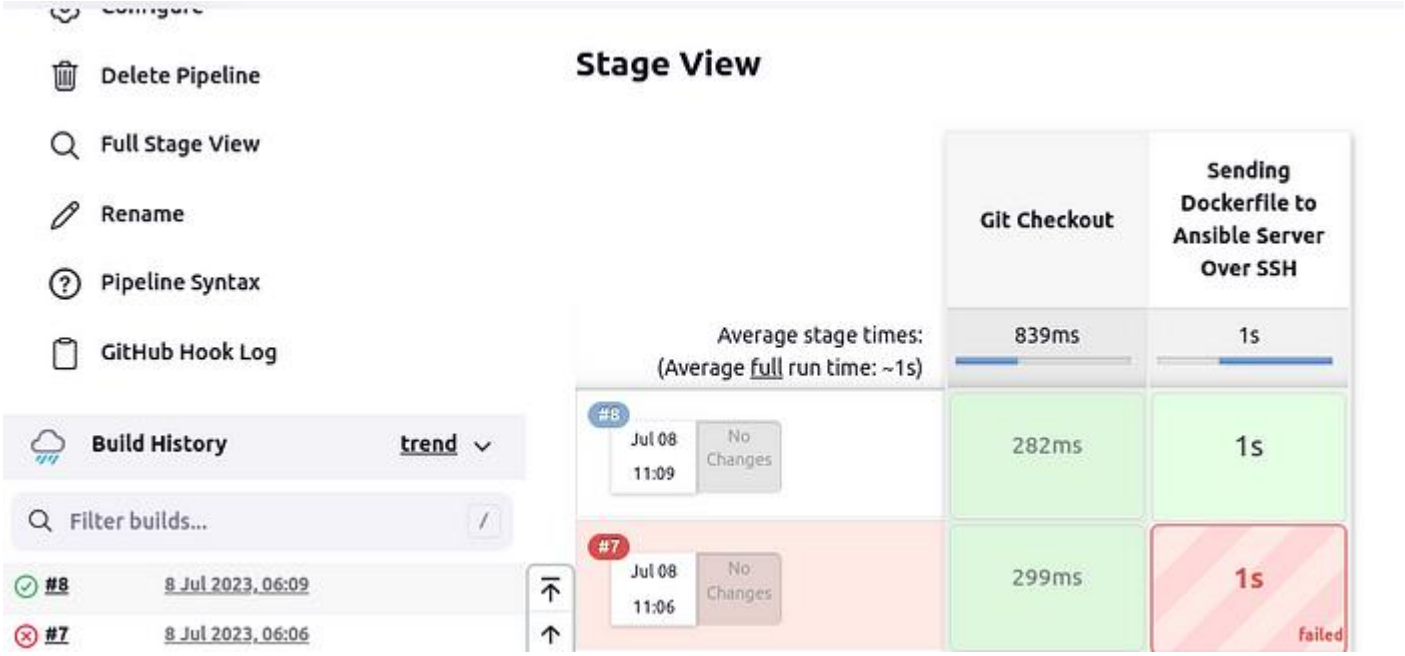
2. In the pipeline configuration section, add the following code to the Groovy script. Make sure to replace the Ansible Server Private IP and the project name directory with your specific values:

```
stage('Sending Dockerfile to Ansible Server Over SSH') {
    sshagent(['ansible-demo']) {
        sh 'ssh -o StrictHostKeyChecking=no ubuntu@54.197.136.241'
        sh 'scp /var/lib/jenkins/workspace/project-name/*
ubuntu@54.197.136.241:/home/ubuntu'
    }
}
```

To access the full Groovy script, you can visit the following link:(https://github.com/AWSENG/groovy_script_k8s_project). This script contains the complete code and configuration for your reference in the Kubernetes project.



Build Now to see if it succeed. as you can see it is:



Now you can go to ansible server and see if there is docker image created or not by running this command

```
docker images
```

To add a stage for Docker image tagging in the Groovy script:

Do addition of this script as well. After adding this stage, save the pipeline configuration. [Full Groovy Script](#).

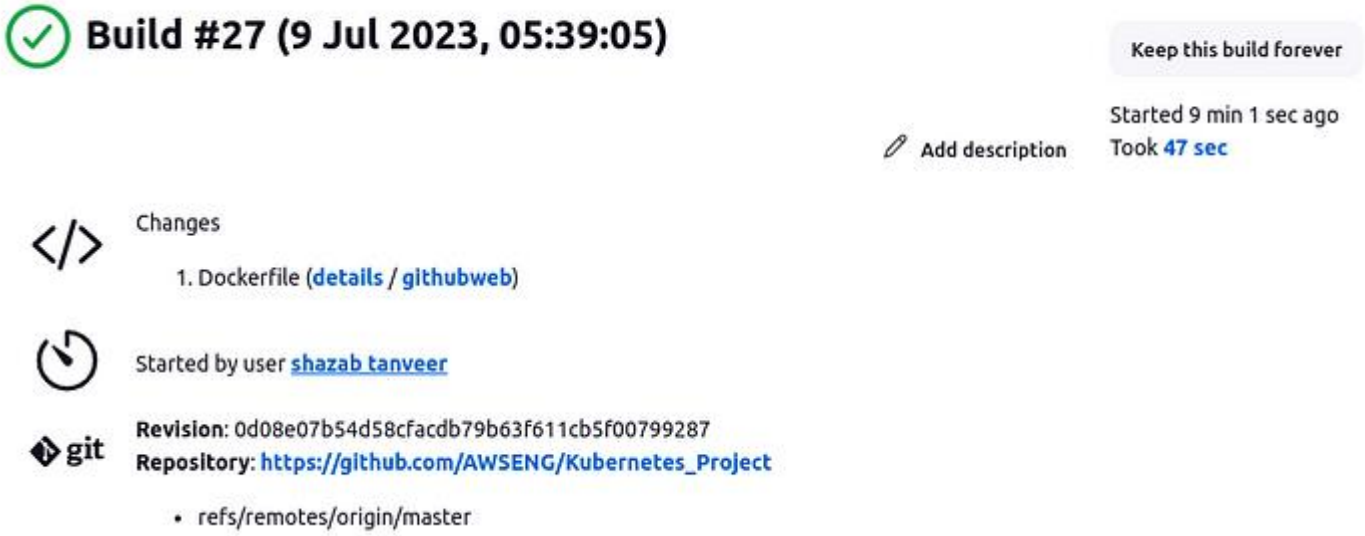
```
stage('Docker image tagging') {
    sshagent(['ansible-demo']) {
        sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.81.244 cd
/home/ubuntu'
        sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.81.244 sudo docker
image tag $JOB_NAME:v1.$BUILD_ID shazabtanveer/$JOB_NAME:v1.$BUILD_ID'
        sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.81.244 sudo docker
image tag $JOB_NAME:v1.$BUILD_ID shazabtanveer/$JOB_NAME:latest'
    }
}
```

Pipeline script

Script ?

```
node {
    stage('Git Checkout'){
        git ' https://github.com/AWSENG/Kubernetes_Project'
    }
    stage('Sending Dockerfile to Ansible Server Over SSH'){
        sshagent(['ansible-demo']) {
sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.81.244'
sh 'scp /var/lib/jenkins/workspace/project/* ubuntu@172.31.81.244:/home/ubuntu'
}
    }

stage(('Docker Build Image'){
 sshagent(['ansible-demo']){

    sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.81.244 cd /home/ubuntu'
        sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.81.244 docker image build -t $JOB_NAME:v1.$BUILD_ID .'
}

}

```

Once saved, trigger a build of the pipeline to check for any issues and ensure the execution is successful.



To push the Docker image to Docker Hub from Ansible using Jenkins, add the following stage to the Groovy script:

```
stage('Push Docker Images to DockerHub') {
    sshagent(['ansible-demo']) {
```

```
        withCredentials([string(credentialsId: 'dockerhub_passwd', variable:
'dockerhub_passwd')]) {
            sh "docker login -u shazabtanveer -p ${dockerhub_passwd}"
            sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.81.244 sudo
docker image push shazabtanveer/$JOB_NAME:v1.$BUILD_ID'
            sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.81.244 sudo
docker image push shazabtanveer/$JOB_NAME:latest'
        }
    }
}
```

Before saving the pipeline configuration, make sure to create your Docker Hub password variable using the Jenkins pipeline syntax. Replace my credentials with your own.

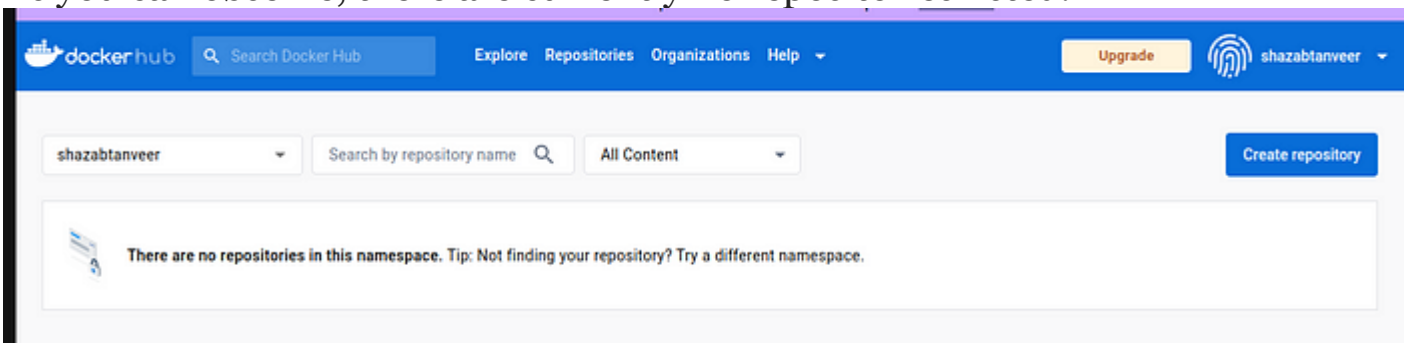
After saving, you can access the full Groovy script at this link: [Full Groovy Script](#).

To login to Docker Hub on the Ansible server:

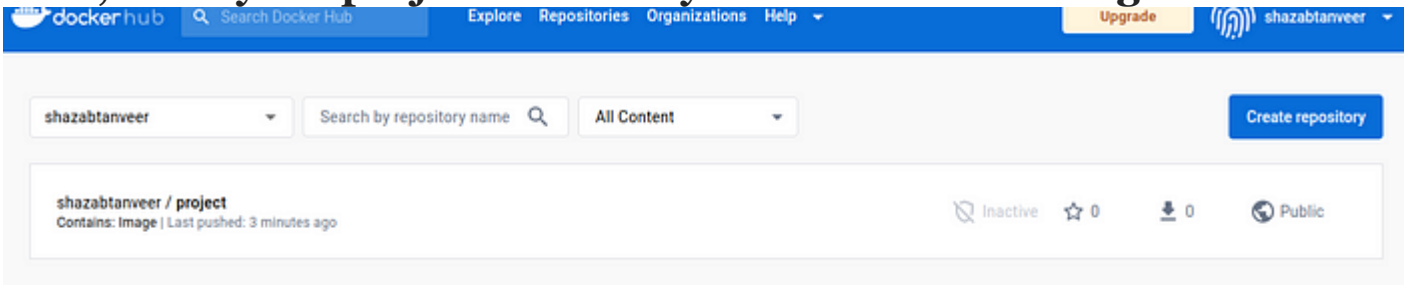
SSH into the Ansible server and Execute the following command:

```
docker login
```

As you can observe, there are currently no repositories listed.



Now, build your project and verify that the Docker images



Now its time to deploy on kubernetes cluster from docker hub.

Links to deployment.yml file, service.yml file and ansible.yml
https://github.com/AWSENG/Kubernetes_Project/blob/master/Deployment.yml
https://github.com/AWSENG/Kubernetes_Project/blob/master/Service.yml
https://github.com/AWSENG/Kubernetes_Project/blob/master/ansible.yml

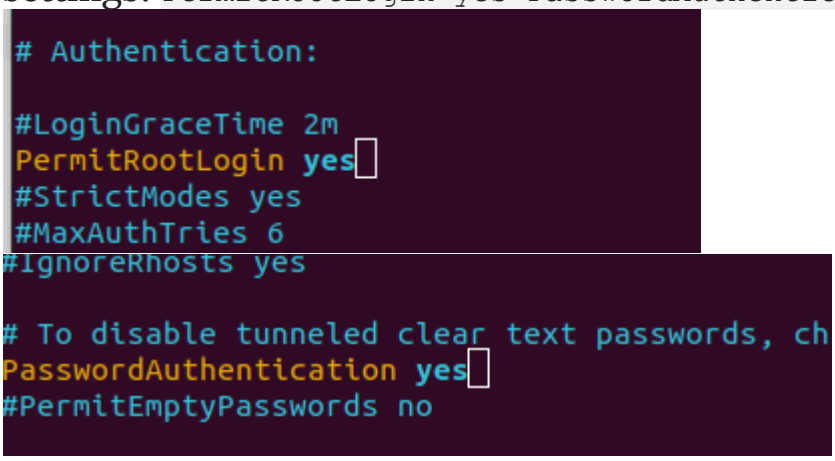
First of all we have to make SSH connection between ansible and kubernetes server and Jenkins and kubernetes server because we have to run ansible playbook.

To make the necessary changes in the SSH configuration on the Ansible server, follow these steps:

1. SSH into the Ansible server.
2. Open the SSH configuration file by running the command:

```
vi /etc/ssh/sshd_config
```

Uncomment the lines containing the following configuration settings: `PermitRootLogin yes PasswordAuthentication yes`



Save the changes and exit the file and Restart the SSH service to apply the modifications:

```
service sshd restart
```

Install ansible in ansible server from ansible.sh file

Go to that location and make these changes in that file anywhere.

[node]
IP of k8S server

```
vi /etc/ansible/hosts
```

```
## green.example.com
## blue.example.com
## 192.168.100.1
## 192.168.100.10
[node]
44.204.171.216
```

Now to check if it succeed run this command if ping success

```
ansible -m ping node
```

```
root@ansible:/home/ubuntu# ansible -m ping node
44.204.171.216 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
```

To establish an SSH connection between the Jenkins server and the Kubernetes server, and copy files from Ansible to the K8s server, follow these steps:

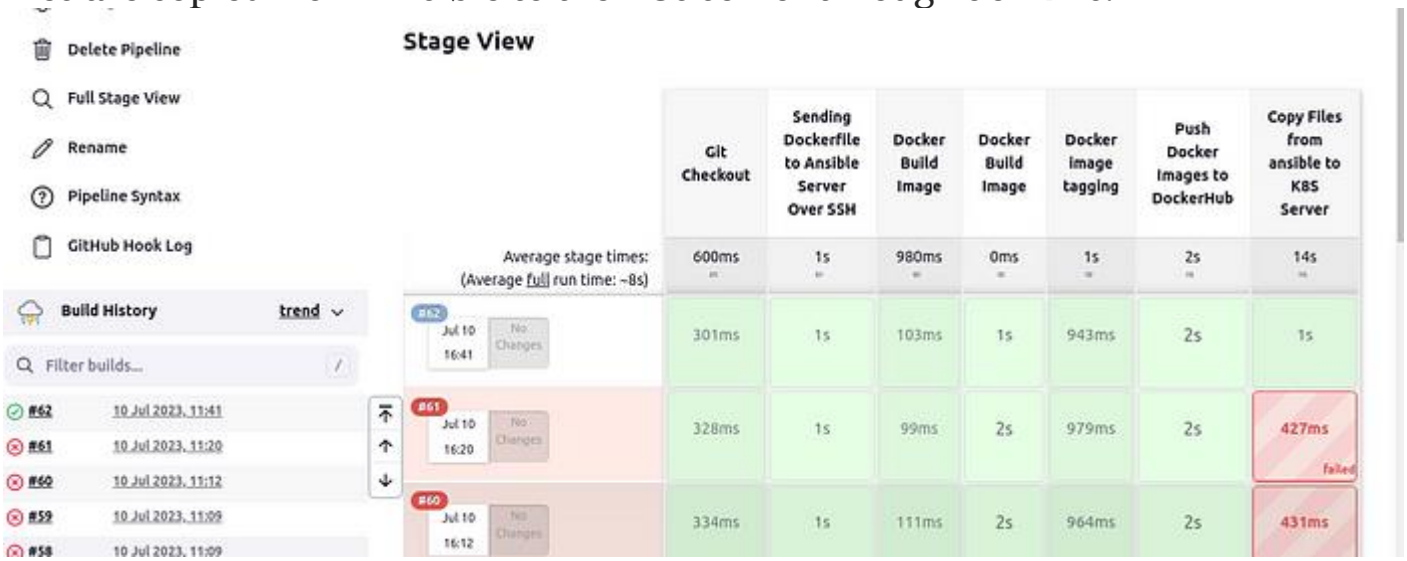
1. Go to the Jenkins server dashboard.
2. Create an SSH connection between Jenkins and the Kubernetes server directly from the Jenkins server.
3. Add the following stage to the Groovy script:

```
stage('Copy Files from Ansible to K8S Server') {
    sshagent(['kubernetes']) {
        sh 'ssh -o StrictHostKeyChecking=no ubuntu@k8s_ip'
        sh 'scp /var/lib/jenkins/workspace/project/*
ubuntu@k8s_ip:/home/ubuntu'
    }
}
```

Make sure to replace `k8s_ip` with the actual IP address of your Kubernetes server. You can access the full Groovy script at this link: [Full Groovy Script](#).

To generate the pipeline syntax for the K8s server, choose the SSH agent and SSH username with a private key, and fill in the other necessary details.

Trigger a build to see if the pipeline works successfully, ensuring the files are copied from Ansible to the K8s server through Jenkins.



To deploy Kubernetes using Ansible and perform additional checks:

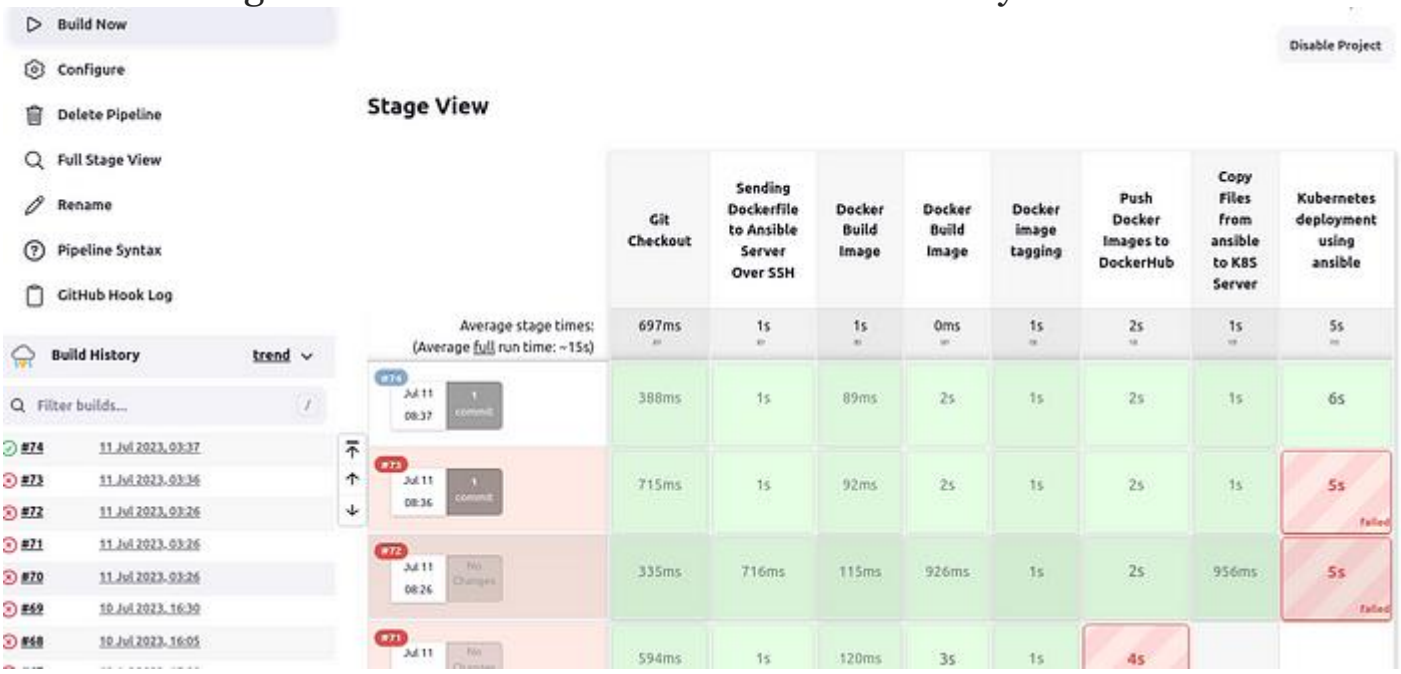
Add the following stage to the Groovy script: code

```
stage('Kubernetes deployment using Ansible') {
    sshagent(['ansible-demo']) {
        sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.81.244 cd
/home/ubuntu'
        sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.81.244 ansible -m
ping node'
        sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.81.244 ansible-
```

```
playbook ansible.yml'
    }
}
```

Make sure to start Minikube on the Kubernetes server.

Save the changes and check if the build works successfully.

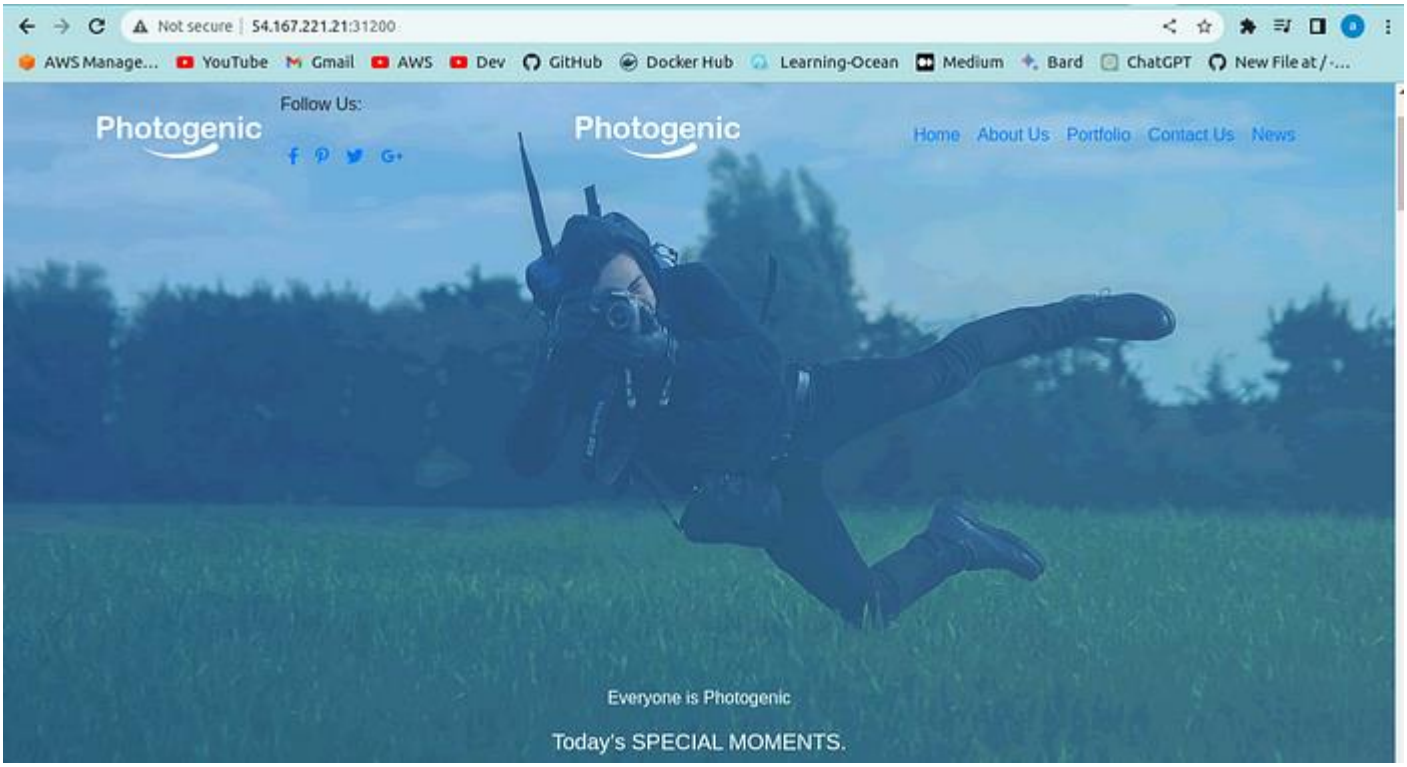


Additionally, you can run the following commands in the terminal to check the deployment:

```
kubectl get pods
kubectl get svc
```

To access the deployed page, paste the IP address of the Kubernetes server in the browser with port 31200 specified. You should be able to access the page if everything is set up correctly.

By following these steps, you can deploy Kubernetes using Ansible, perform necessary checks, and access the deployed page through the browser.



" Thank you for reading! I hope you found this article helpful in your learning journey. If you have any further questions or need additional assistance, feel free to ask. Remember, sharing knowledge is a wonderful way to help others, so if you found this article beneficial, please consider sharing it with others who might find it helpful too. Happy learning! "