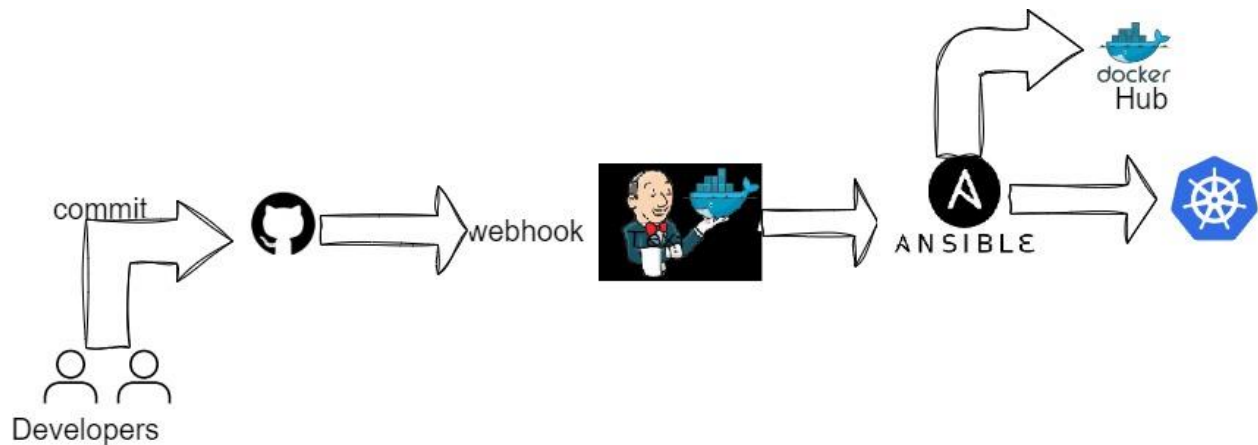


Deployment on Kubernetes cluster using Jenkins CI/CD

In This article I will show you how to deploy Kubernetes cluster using Jenkins CI/CD pipeline. In this demo project we are taking help of various DevOps tools like GitHub, Jenkins, Ansible, Docker Hub and Kubernetes Minikube cluster.



Let's go through the contents of DevOps tools uses:

Developer: Developer will write source code and docker file and push into the code to GitHub repository.

GitHub: GitHub is a code hosting platform for version control and collaboration. And it will store our source code into a repository.

Webhook: Webhook informs Jenkins whenever new code is available and tells Jenkins to build the new code.

Jenkins: Jenkins will pull out the code from GitHub repository, then execute the CI/CD pipeline.

Ansible: Ansible server will execute the Kubernetes deployment and service yaml files to deploy the application.

Kubernetes: Pods and application will run on Kubernetes cluster.

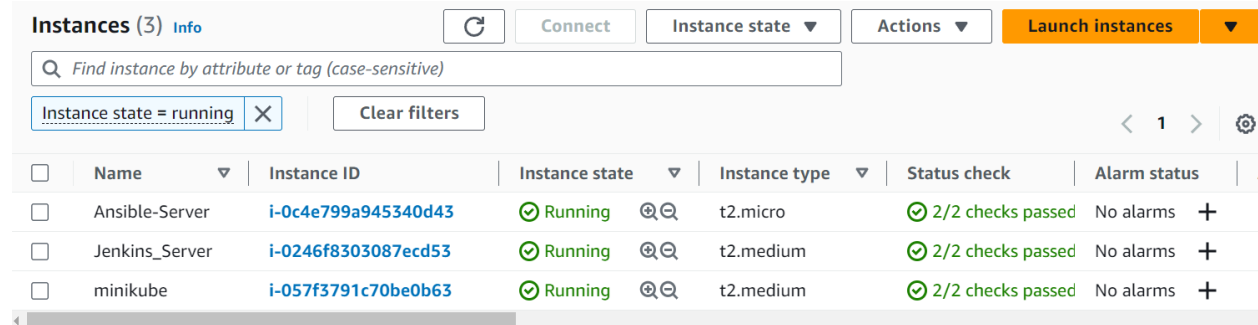
Prerequisite:

GitHub, Jenkins, Docker, Jenkins, Docker Hub Account, Ansible, Kubernetes (deployment & services yaml files).

3 EC2 Ubuntu Instances required:

1. Jenkins (default-jre+Jenkins), t2.Medium+20GB
2. Ansible (python+ansible+docker), t2.micro + 8GB
3. Webapp, Kubernetes Cluster (docker+minikube), t2.Medium+20GB

Signed In AWS Console and created three Ubuntu Instances.



The screenshot shows the AWS Management Console 'Instances' page. At the top, there are buttons for 'Connect', 'Instance state', 'Actions', and 'Launch instances'. A search bar contains the text 'Find instance by attribute or tag (case-sensitive)'. Below the search bar, there is a filter 'Instance state = running' and a 'Clear filters' button. The main table lists three instances, all in a 'Running' state with '2/2 checks passed' and 'No alarms'.

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status
<input type="checkbox"/>	Ansible-Server	i-0c4e799a945340d43	Running	t2.micro	2/2 checks passed	No alarms
<input type="checkbox"/>	Jenkins_Server	i-0246f8303087ecd53	Running	t2.medium	2/2 checks passed	No alarms
<input type="checkbox"/>	minikube	i-057f3791c70be0b63	Running	t2.medium	2/2 checks passed	No alarms

How to install Jenkins on Ubuntu: t2.Medium+20GB

```
# sudo apt update
```

```
# sudo apt install openjdk-11-jdk
```

```
# java --version
```

Jenkins can easily be installed on Ubuntu by importing and adding the GPG keys to the system.

Now you got to add GPG keys:

```
# wget -p -O - https://pkg.jenkins.io/debian/jenkins.io.key | sudo apt-key add -
```

After adding GPG keys, add the Jenkins package address to the sources list by typing the command given below:

```
# sudo apt update
```

Let's move forward and do the real work of installing Jenkins.

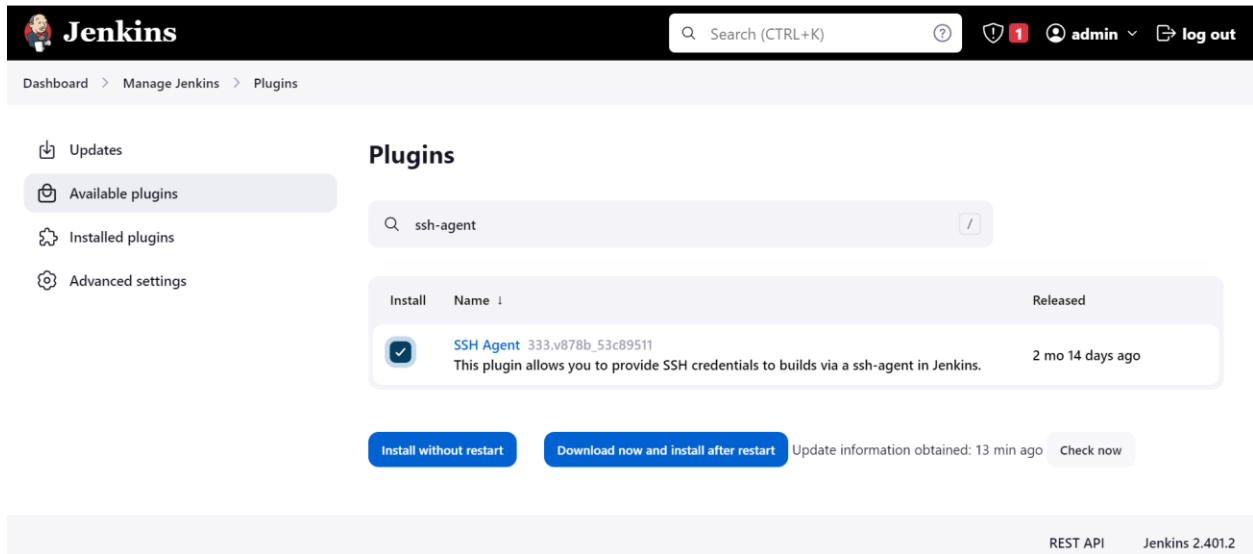
```
# sudo apt install Jenkins
```

```
# sudo systemctl status Jenkins
```

```
# sudo systemctl start Jenkins
```

After started Jenkins install ssh-agent plugin using below steps:

Jenkins → Dashboard → Manage Jenkins → Plugins → click on available plugins → search(ssh-agent) → click on Install without restart.



The screenshot shows the Jenkins web interface. At the top, there's a header with the Jenkins logo, a search bar, and user information (admin). Below the header, a breadcrumb trail shows 'Dashboard > Manage Jenkins > Plugins'. On the left sidebar, there are links for 'Updates', 'Available plugins', 'Installed plugins', and 'Advanced settings'. The main content area is titled 'Plugins' and features a search bar with 'ssh-agent' entered. Below the search bar, a table lists the installed plugins. The table has columns for 'Install', 'Name', and 'Released'. One plugin is listed: 'SSH Agent' with version '333.v678b_53c89511'. Below the table, there are two buttons: 'Install without restart' and 'Download now and install after restart'. To the right of these buttons, it says 'Update information obtained: 13 min ago' and a 'Check now' button. At the bottom right of the page, it says 'REST API' and 'Jenkins 2.401.2'.

Install	Name	Released
<input checked="" type="checkbox"/>	SSH Agent 333.v678b_53c89511 This plugin allows you to provide SSH credentials to builds via a ssh-agent in Jenkins.	2 mo 14 days ago

[Install without restart](#) [Download now and install after restart](#) Update information obtained: 13 min ago [Check now](#)

Install Ansible on Ubuntu server: t2.micro + 8GB

Connect to ansible ubuntu instance and execute below commands as normal user.

```
# sudo apt-add-repository ppa:ansible/ansible -y
```

```
# sudo apt update -y
```

```
# sudo apt install ansible -y
```

```
# ansible --version
```

Docker Installation steps.

```
apt install docker.io -y
usermod -aG docker ubuntu
systemctl restart docker
systemctl enable docker.service
```

Install minikube cluster with docker: t2.Medium+20GB

Run the following commands to update all system packages to the latest release:

```
sudo apt update
```

```
sudo apt upgrade
```

If a reboot is required after the upgrade then perform the process

```
[ -f /var/run/reboot-required ] && sudo reboot -f
```

Docker Installation steps.

```
apt install docker.io -y
usermod -aG docker ubuntu
systemctl restart docker
systemctl enable docker.service
```

Download Minikube on Ubuntu

You need to download the minikube binary. I will put the binary under /usr/local/bin directory since it is inside \$PATH.

```
# wget https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
# chmod +x minikube-linux-amd64
# sudo mv minikube-linux-amd64 /usr/local/bin/minikube
```

Check the minikube version using below command

```
# minikube version
```

Install kubectl on Ubuntu

We need kubectl which is a command line tool used to deploy and manage applications on Kubernetes.

```
curl -LO https://storage.googleapis.com/kubernetes-release/release/`curl -s
https://storage.googleapis.com/kubernetes-release/release/stable.txt`/bin/linux/amd64/kubectl
```

Make the kubectl binary executable

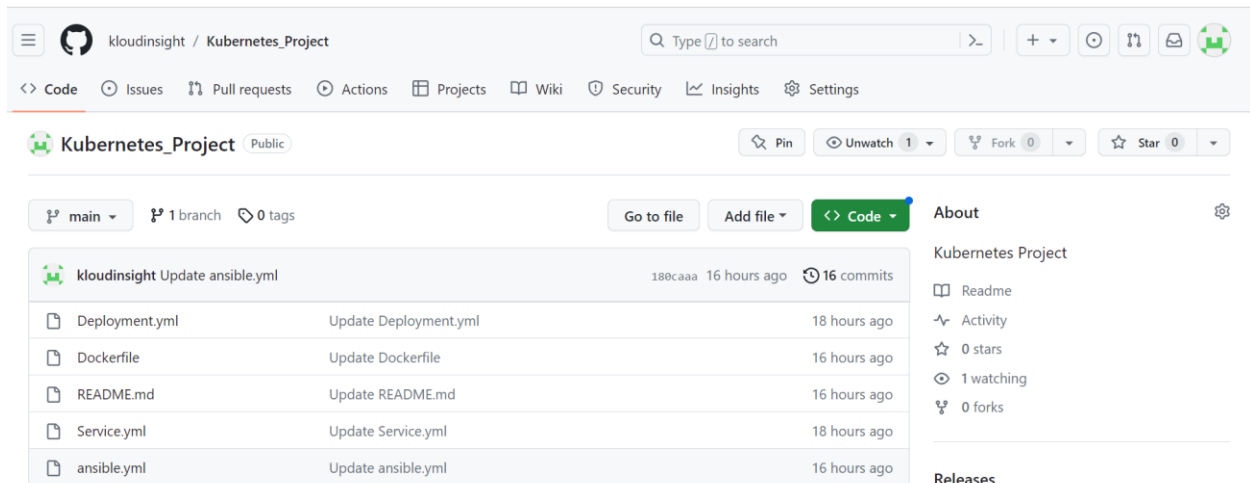
```
# chmod +x ./kubectl
# Move the binary in to your PATH:
# sudo mv ./kubectl /usr/local/bin/kubectl
# sudo usermod -aG docker $USER && newgrp docker
```

Starting minikube on Ubuntu

Now that components are installed, you can start minikube. VM image will be downloaded and configured for Kubernetes single node cluster.

minikube start

Login to GitHub remote repository create below listed Dockerfile, deployment.yml, service.yml and ansible.yml files.



The screenshot shows the GitHub interface for a repository named 'Kubernetes_Project' by user 'kloudinsight'. The repository is public and has 1 branch (main) and 0 tags. The file list shows several files: 'Deployment.yml', 'Dockerfile', 'README.md', 'Service.yml', and 'ansible.yml', all updated 16 hours ago. The 'About' section on the right provides details about the repository, including 0 stars, 1 watching, and 0 forks. The 'Releases' section is also visible.

Dockerfile

```
FROM centos:7
MAINTAINER VenkatKumar
RUN yum install -y httpd \
    zip\
    unzip
ADD https://www.free-css.com/assets/files/free-css-
templates/download/page254/photogenic.zip /var/www/html/
WORKDIR /var/www/html/
RUN unzip photogenic.zip
RUN cp -rvf photogenic/* .
RUN rm -rf photogenic photogenic.zip
CMD ["/usr/sbin/httpd", "-D", "FOREGROUND"]
EXPOSE 80
```

Deployment.yml

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: kloudinsight
spec:
  replicas: 2
```

```
selector:      # tells the controller which pods to watch/belong to
  matchLabels:
    app: kloudinsight
template:
  metadata:
    labels:
      app: kloudinsight
  spec:
    containers:
      - name: kloudinsight
        image: kloudinsight/pipeline-demo
        imagePullPolicy: Always
        ports:
          - containerPort: 80
```

Service.yml

```
kind: Service
apiVersion: v1
metadata:
  name: kloudinsight
  labels:
    app: kloudinsight
spec:
  ports:
    - port: 8080
      targetPort: 80
      nodePort: 31200
  selector:
    app: kloudinsight
  type: LoadBalancer
```

ansible.yml

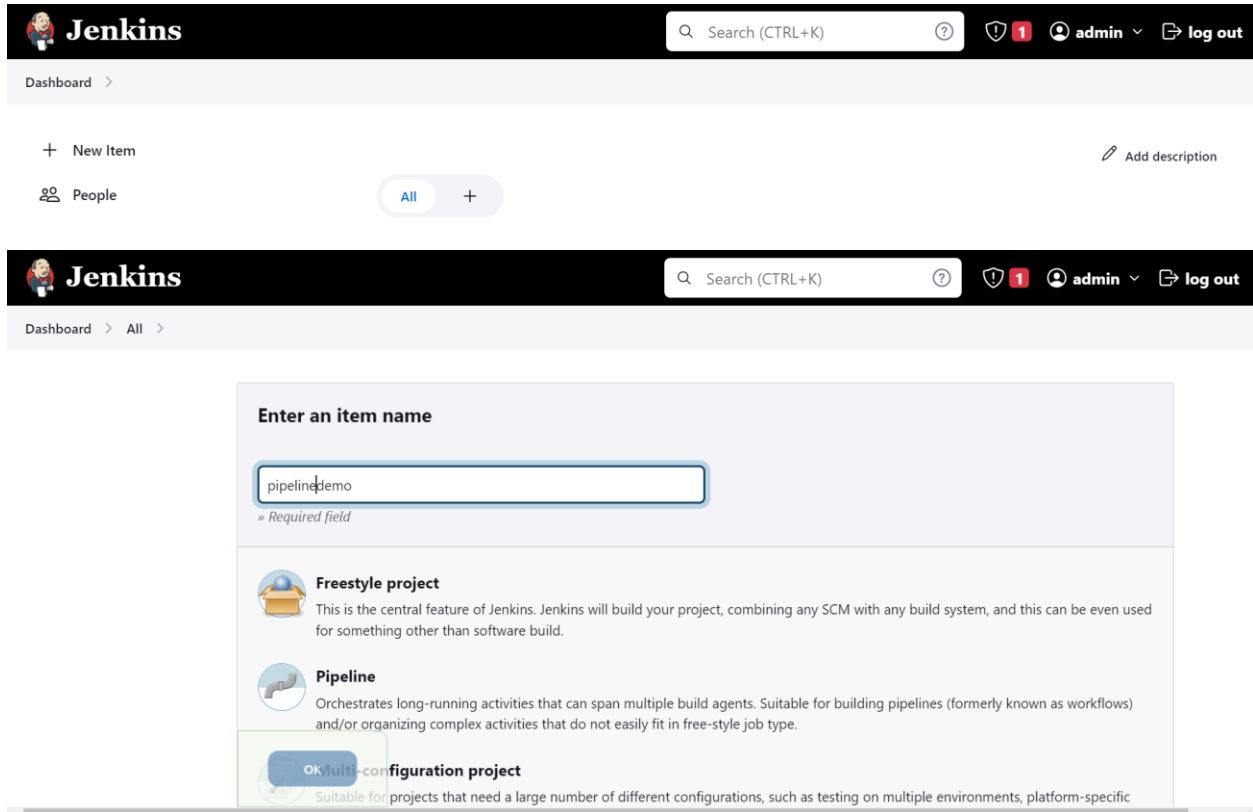
```
- hosts: all

tasks:
  - name: create new deployment
    command: kubectl apply -f /home/ubuntu/deployment.yml

  - name: create new service
    command: kubectl apply -f /home/ubuntu/service.yml
```

Pipeline Script Steps:

Login into Jenkins server and click on New Item, Enter pipeline name and select Pipeline and click on OK.



The image shows the Jenkins web interface. At the top is a black header with the Jenkins logo, a search bar, a notification bell with a red '1', the user 'admin', and a 'log out' button. Below the header is a light gray breadcrumb bar showing 'Dashboard >'. The main content area has a '+ New Item' button and an 'Add description' link. A 'People' section shows 'All' and a '+' button. A modal dialog is open in the center, titled 'Enter an item name'. It contains a text input field with 'pipeline' and a 'Required field' error message. Below the input are three options: 'Freestyle project', 'Pipeline', and 'Multi-configuration project'. The 'Pipeline' option is highlighted with a blue border and a blue 'OK' button. The 'Multi-configuration project' option is partially visible and highlighted with a green border.

Enter an item name

pipelinedemo

» Required field

Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific

Navigate to pipeline configuration select pipeline script and copy the pipeline groovy script then click on apply and save.

Dashboard > pipelinedemo > Configuration

Configure

- General
- Advanced Project Options
- Pipeline**

Pipeline script

Script ?

try sample Pipeline...

☒ Use Groovy Sandbox ?

Pipeline Syntax

Save Apply

1. Replace your ansible server username and IP address(ubuntu@172.31.18.35)
2. Replace your kubernetes server username and IP address (ubuntu@172.31.81.94).

```
node {
    stage('Git Checkout'){
        git branch: 'main', url: 'https://github.com/kloudinsight/Kubernetes_Project.git'
    }
    stage('sending dockerfile to ansible server over ssh'){
        sshagent(['ansibledemo']) {
            sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.18.35'
            sh 'scp /var/lib/jenkins/workspace/pipeline-demo/* ubuntu@172.31.18.35:/home/ubuntu/'
        }
    }
    stage('Docker build image'){
        sshagent(['ansibledemo']){
            sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.18.35 cd /home/ubuntu/'
            sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.18.35 docker image build -t $JOB_NAME:v1.$BUILD_ID .'
        }
    }
    stage('Docker image tagging'){
        sshagent(['ansibledemo']){
```



```

    sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.18.35 cd /home/ubuntu/'

    sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.18.35 docker image tag $JOB_NAME:v1.$BUILD_ID
kloudinsight/$JOB_NAME:v1.$BUILD_ID'

    sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.18.35 docker image tag $JOB_NAME:v1.$BUILD_ID
kloudinsight/$JOB_NAME:latest'

}

}

stage ('Push the docker image to Docker Hub'){
    sshagent(['ansibledemo']){
        withCredentials([string(credentialsId: 'dockehubpasswd', variable: 'dockehubpasswd')]) {
            sh "ssh -o StrictHostKeyChecking=no ubuntu@172.31.18.35 docker login -u kloudinsight -p ${dockehubpasswd}"
            sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.18.35 docker image push kloudinsight/$JOB_NAME:v1.$BUILD_ID'
            sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.18.35 docker image push kloudinsight/$JOB_NAME:latest'
        }
    }
}

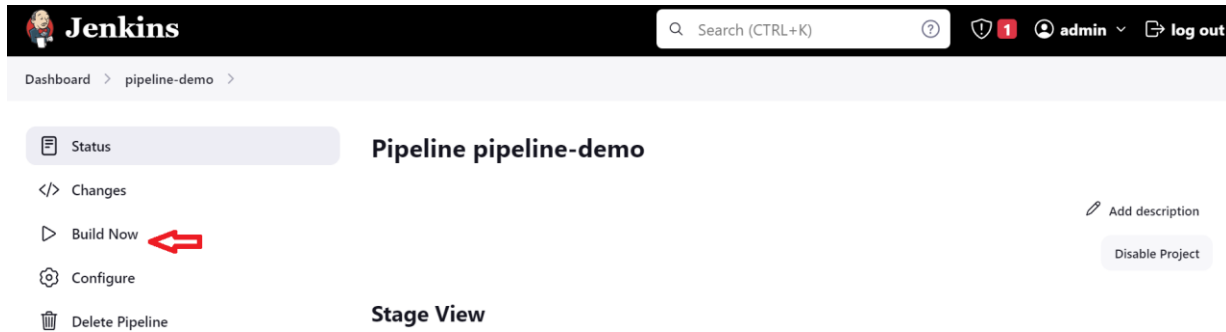
stage ('Copy files from ansible to kubernetes server'){
    sshagent(['ansibledemo']){
        sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.81.94 cd /home/ubuntu/'
        sh 'scp /var/lib/jenkins/workspace/pipeline-demo/* ubuntu@172.31.81.94:/home/ubuntu/'

    }
}

stage ('Copy files from ansible to kubernetes server'){
    sshagent(['ansibledemo']){
        sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.18.35 cd /home/ubuntu/'
        sh 'ssh -o StrictHostKeyChecking=no ubuntu@172.31.18.35 ansible-playbook ansible.yml'
    }
}
}

```

Login into Jenkins server navigate to pipeline click on build now.



The screenshot shows the Jenkins web interface for a pipeline named 'pipeline-demo'. The top navigation bar includes the Jenkins logo, a search bar, and user information. The left sidebar contains a 'Status' tab and a 'Build Now' button, which is highlighted with a red arrow. The main content area displays the pipeline's stages and a 'Stage View' button.

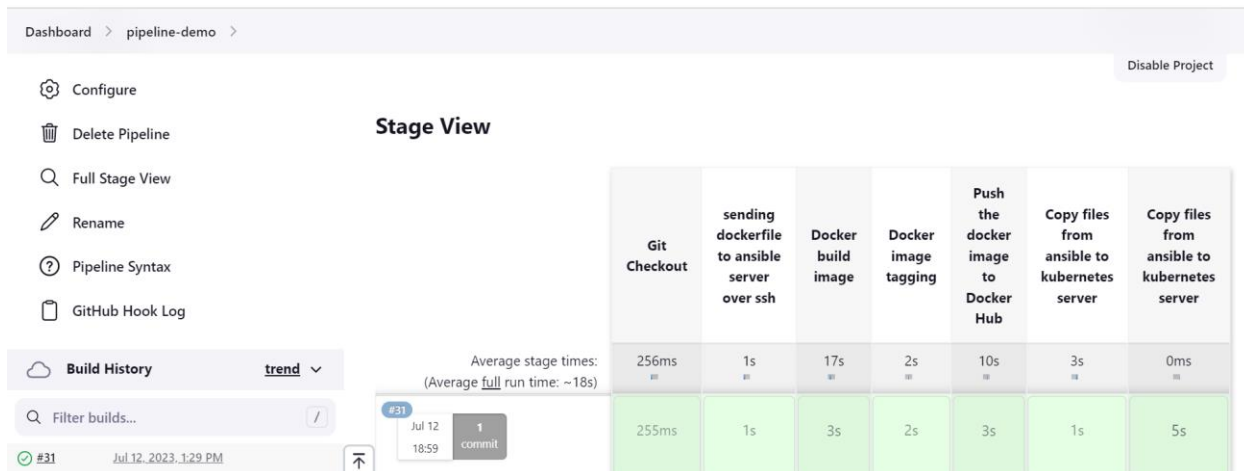
Dashboard > pipeline-demo >

Pipeline pipeline-demo

Build Now

Stage View

Pipeline Demo project build success.



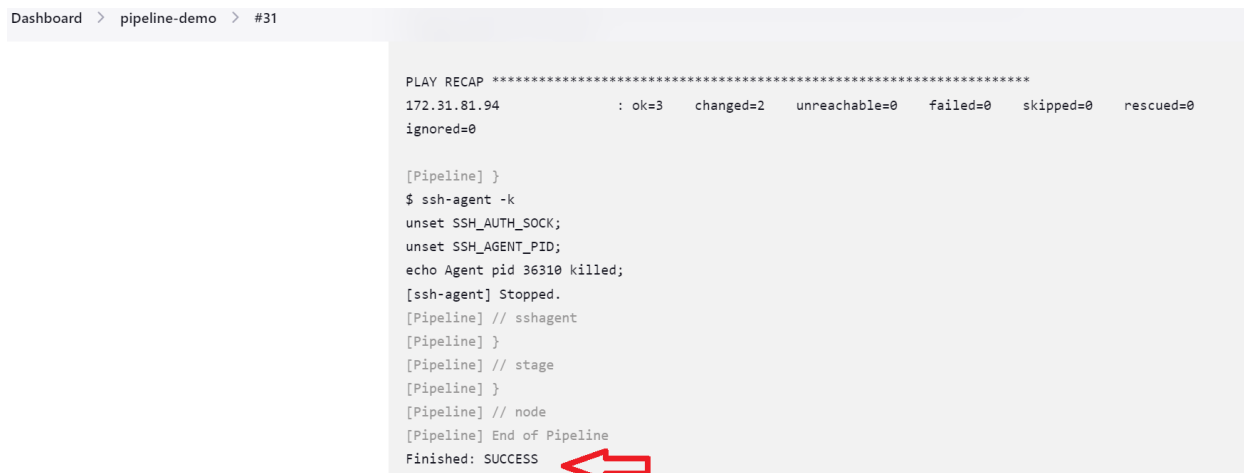
The screenshot shows the Jenkins web interface for the 'pipeline-demo' project. The left sidebar contains a 'Build History' tab and a 'Build Now' button. The main content area displays the pipeline's stages and a 'Stage View' button. The 'Build History' table shows a successful build for commit #31.

Dashboard > pipeline-demo >

Stage View

Git Checkout	sending dockerfile to ansible server over ssh	Docker build image	Docker image tagging	Push the docker image to Docker Hub	Copy files from ansible to kubernetes server	Copy files from ansible to kubernetes server
256ms	1s	17s	2s	10s	3s	0ms
255ms	1s	3s	2s	3s	1s	5s

Check the pipeline demo console output / logs.



The screenshot shows the Jenkins web interface for the 'pipeline-demo' project, specifically the console output for build #31. The output shows the pipeline's execution steps, including 'ssh-agent' and 'stage' commands, and ends with 'Finished: SUCCESS', which is highlighted with a red arrow.

Dashboard > pipeline-demo > #31

```
PLAY RECAP *****
172.31.81.94      : ok=3   changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

[Pipeline] }
$ ssh-agent -k
unset SSH_AUTH_SOCK;
unset SSH_AGENT_PID;
echo Agent pid 36310 killed;
[ssh-agent] Stopped.
[Pipeline] // sshagent
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

Connect to kubernetes cluster machine and check the minikube cluster is ready or not
execute the below command

kubectl get nodes

```
ubuntu@ip-172-31-81-94:~$ kubectl get nodes
NAME          STATUS    ROLES          AGE    VERSION
minikube      Ready     control-plane   62m    v1.26.3
ubuntu@ip-172-31-81-94:~$
```

Verify the pods running or not execute the below command.

kubectl get pods

```
ubuntu@ip-172-31-81-94:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
kloudinsight-8556f6b4dd-g9h1b      1/1     Running   0           8m45s
kloudinsight-8556f6b4dd-tf6gj      1/1     Running   0           8m45s
ubuntu@ip-172-31-81-94:~$
```

Verify the application home page.

<http://54.144.213.63:31200>



-----END-----

Thank you for reading 😊

Venkat Kumar

