building CI/CD pipeline project using Jenkins

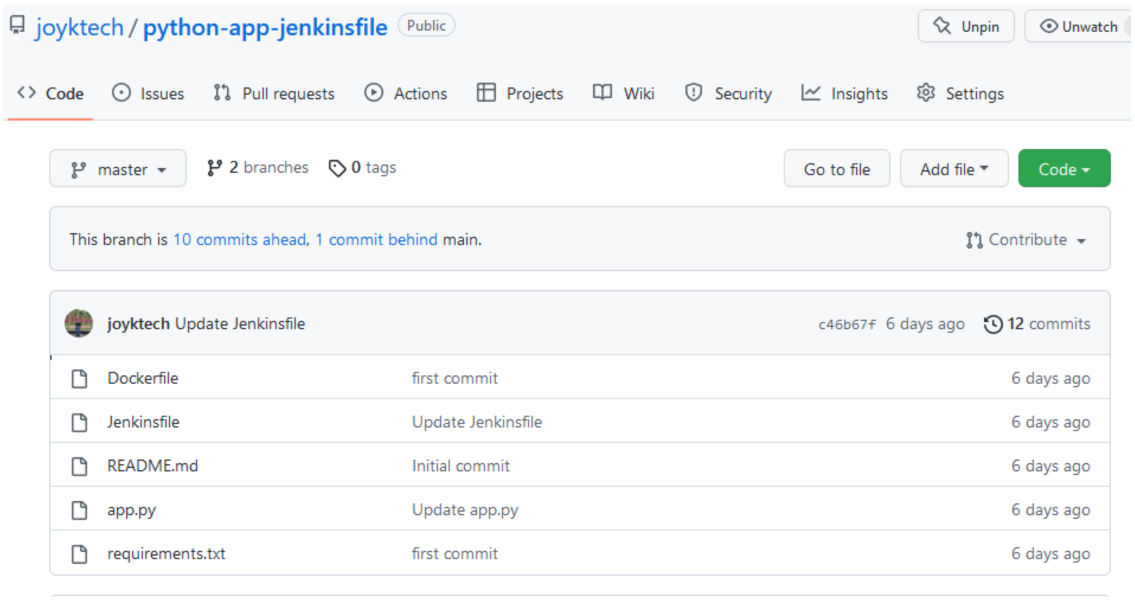
Requirements:

1.

- Application Server (installation package: Docker)
- Jenkins Server (Installation Package: Jenkins, Docker)

Note: Configure ssh keygen for password less login from Jenkins server to application server
This viedw will help you to create pass less: https://www.youtube.com/watch?v=9M56CrVbOgk

2. Docker HUB & Git Account
Note: Please create repository and add require file. You can download all file from my repo:
https://github.com/joyktech/python-app-jenkinsfile.git
branch should be **master**
look like this—

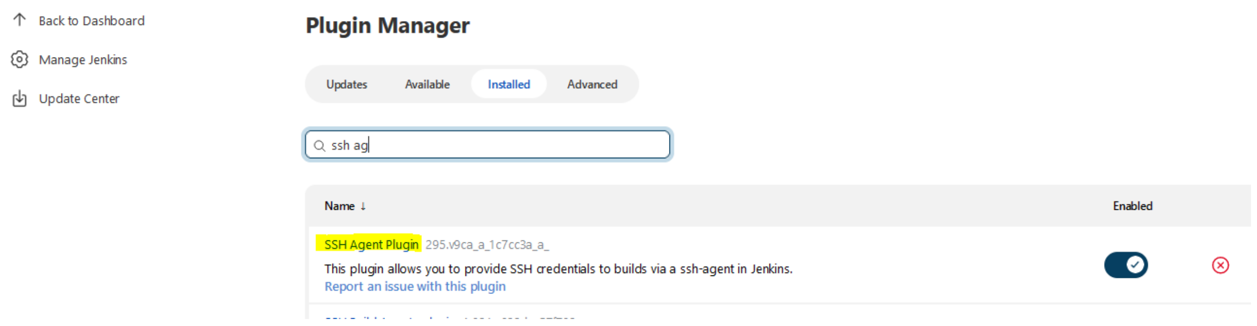


This branch is 10 commits ahead, 1 commit behind main.

joyktech Update Jenkinsfile c46b67f 6 days ago 12 commits

Dockerfile	first commit	6 days ago
Jenkinsfile	Update Jenkinsfile	6 days ago
README.md	Initial commit	6 days ago
app.py	Update app.py	6 days ago
requirements.txt	first commit	6 days ago

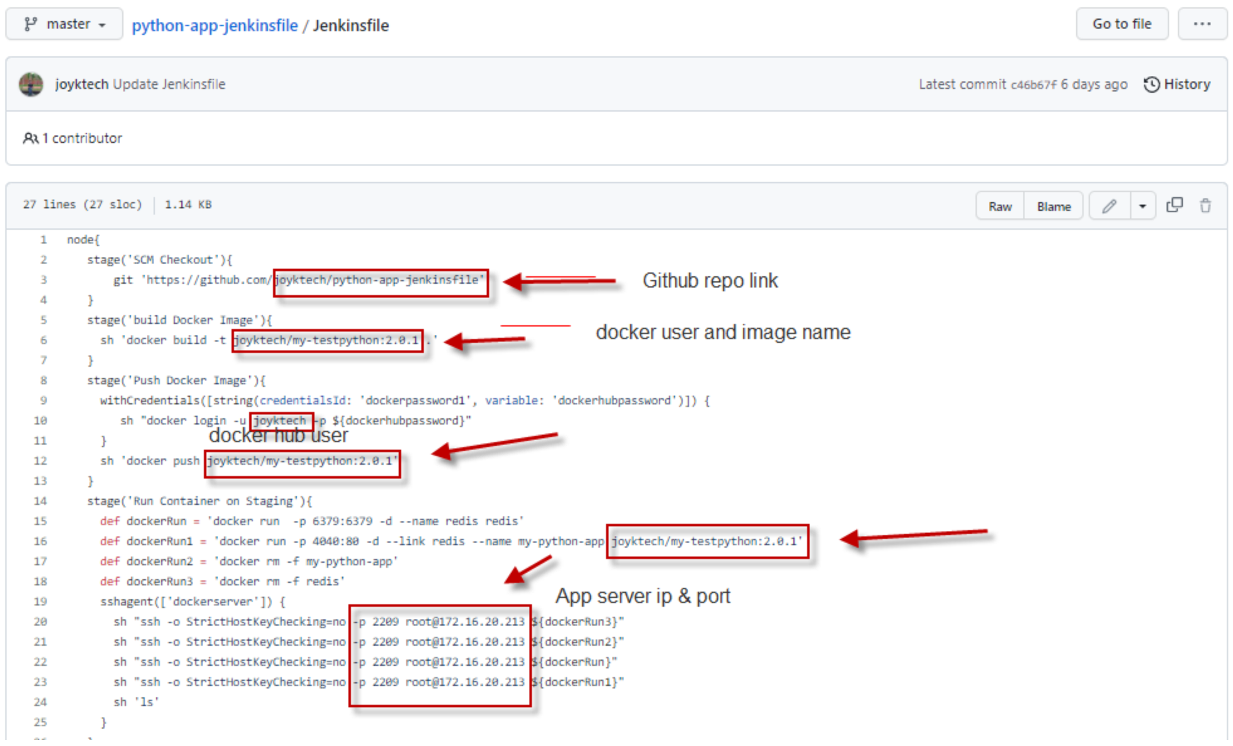
3. login jenkins account from web and install **ssh agent** plugin from manage plugin.



SSH Agent Plugin 295.v9ca_a_1c7cc3a_a_
This plugin allows you to provide SSH credentials to builds via a ssh-agent in Jenkins.
Report an issue with this plugin

Your environment is ready now. Let’s start deployment!!!!!

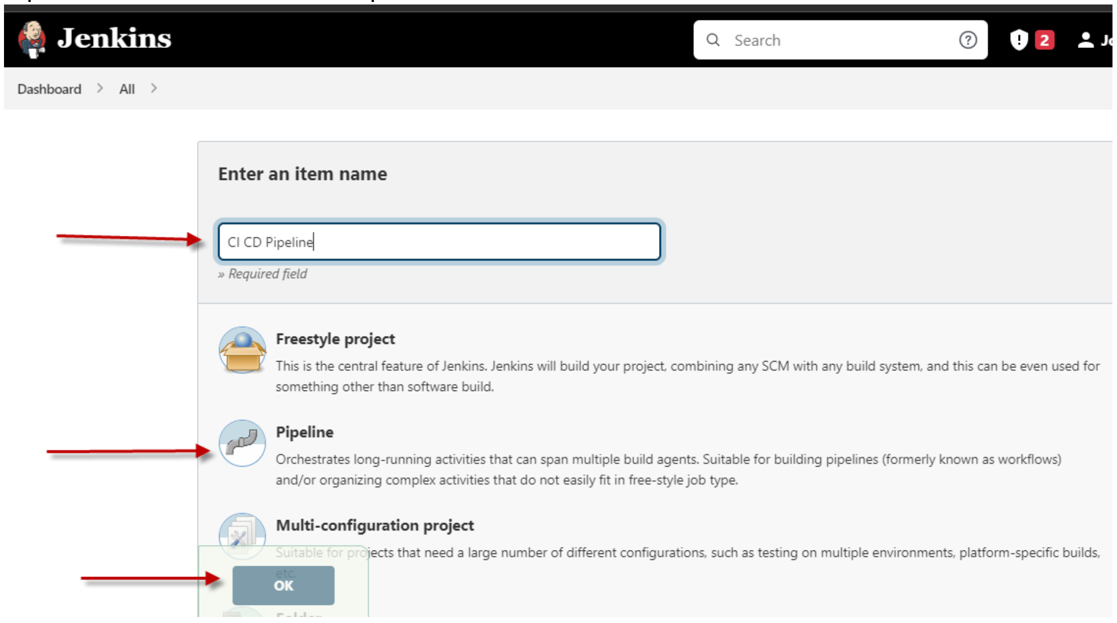
Step-1: Open Jenkins file from your git and modify it look like this…


master
python-app-jenkinsfile / Jenkinsfile
Go to file
joyktech Update Jenkinsfile
Latest commit c46b67f 6 days ago
History
1 contributor
27 lines (27 sloc) | 1.14 KB
Raw
Blame
Github repo link
docker user and image name
docker hub user
App server ip & port

```
node{
  stage('SCM Checkout'){
     git 'https://github.com/joyktech/python-app-jenkinsfile'
  }
  stage('build Docker Image'){
    sh 'docker build -t joyktech/my-testpython:2.0.1 .'
  }
  stage('Push Docker Image'){
    withCredentials([string(credentialsId: 'dockerpassword1', variable: 'dockerhubpassword')]) {
       sh "docker login -u joyktech -p ${dockerhubpassword}"
    }
    sh 'docker push joyktech/my-testpython:2.0.1'
  }
  stage('Run Container on Staging'){
    def dockerRun = 'docker run  -p 6379:6379 -d --name redis redis'
    def dockerRun1 = 'docker run -p 4040:80 -d --link redis --name my-python-app joyktech/my-testpython:2.0.1'
    def dockerRun2 = 'docker rm -f my-python-app'
    def dockerRun3 = 'docker rm -f redis'
    sshagent(['dockerserver']) {
      sh "ssh -o StrictHostKeyChecking=no -p 2209 root@172.16.20.213 ${dockerRun3}"
      sh "ssh -o StrictHostKeyChecking=no -p 2209 root@172.16.20.213 ${dockerRun2}"
      sh "ssh -o StrictHostKeyChecking=no -p 2209 root@172.16.20.213 ${dockerRun}"
      sh "ssh -o StrictHostKeyChecking=no -p 2209 root@172.16.20.213 ${dockerRun1}"
      sh 'ls'
    }
```

Modify as per above snap & save Jenkinsfile.

Step-2: Now login Jenkins from browser and create new pipeline:
Go to new item—Item name – (write item name)—choose pipeline – click ok
Pipeline created. Check below pic for visual..

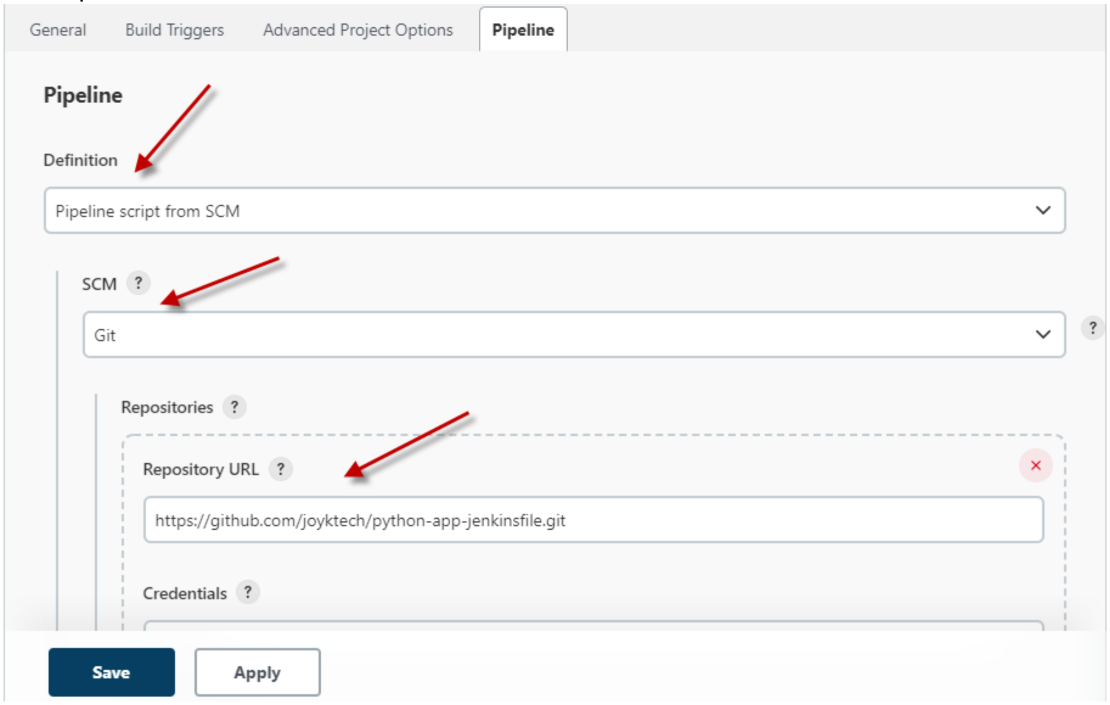


This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

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.

Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Step-3: Under pipeline go to definition and select pipeline scrip from SCM
From SCM section select git
From Repositories option provides your repositories URL
Brance name should be master
Below pic for visualize!!



Pipeline

Definition: Pipeline script from SCM

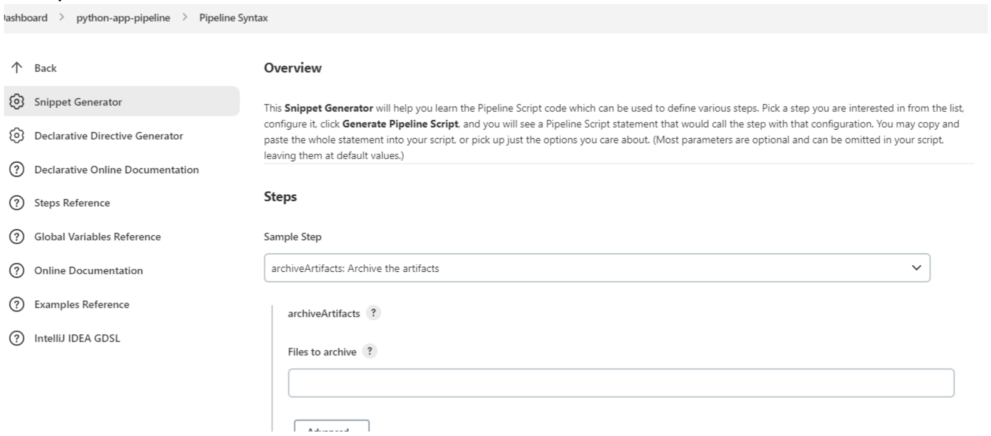
SCM: Git

Repositories

Repository URL: https://github.com/joyktech/python-app-jenkinsfile.git

Credentials

Step-4: In the last section from pipeline you will be find **Pipeline Syntax** option Please click it and it will be open from new TAB. Look like this----


ashboard > python-app-pipeline > Pipeline Syntax
Back
Snippet Generator
Declarative Directive Generator
Declarative Online Documentation
Steps Reference
Global Variables Reference
Online Documentation
Examples Reference
IntelliJ IDEA GDSL

Overview

This **Snippet Generator** will help you learn the Pipeline Script code which can be used to define various steps. Pick a step you are interested in from the list, configure it, click **Generate Pipeline Script**, and you will see a Pipeline Script statement that would call the step with that configuration. You may copy and paste the whole statement into your script, or pick up just the options you care about. (Most parameters are optional and can be omitted in your script, leaving them at default values.)

Steps

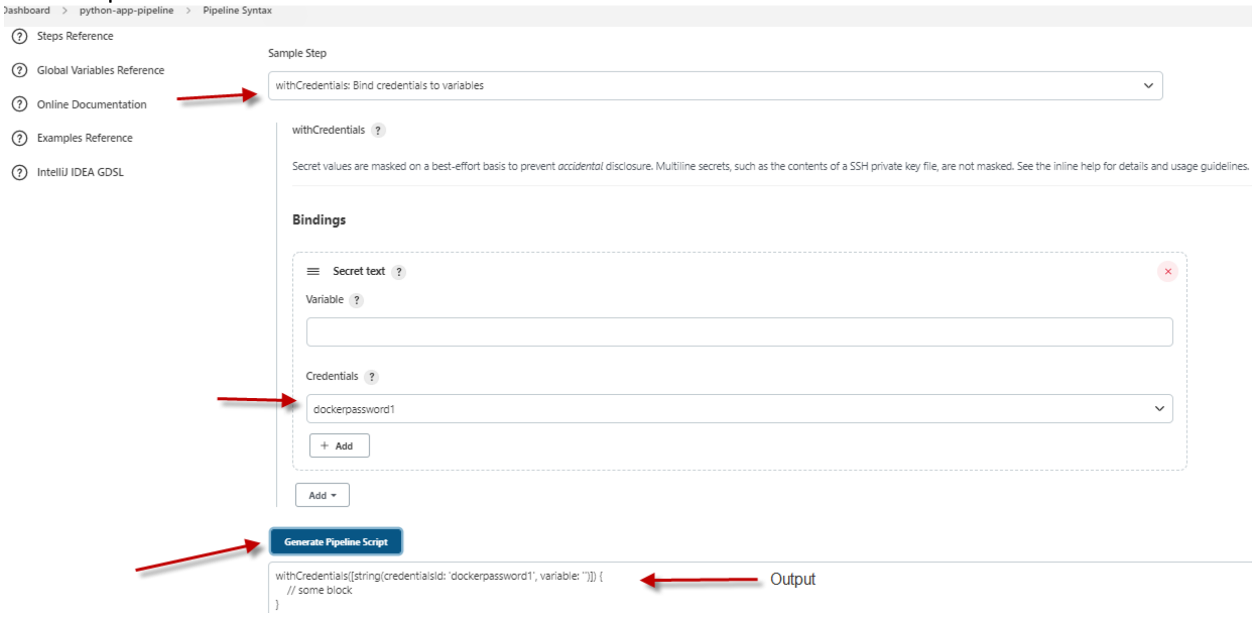
Sample Step: archiveArtifacts: Archive the artifacts

archiveArtifacts

Files to archive

Note: From this pipeline syntax we will get many types of syntax for Jenkins File.
We need to 2(two pipeline syntax for Jenkins file which one for credentials & another one for ssh agent)

Let’s create syntax from sample step:
First we need to choose from sample step **withCredentials: Bind credentials to variables**
From **Bindings** click Add & choose Secret text
Vairable: `dockerhubpassword`
From Credentials—Add—Jenkins—Kind—Secret text—Secret= (in this secret section provide your docker hub account password)—ID= `dockerpassword1 (this is credential ID which you can find from Jenkinsfile)`–Description= dockerhubpassword
Now Click **add** button then click **Generate Pipeline Script**
Your output should be like this---


Dashboard > python-app-pipeline > Pipeline Syntax

Sample Step: withCredentials: Bind credentials to variables

withCredentials

Secret values are masked on a best-effort basis to prevent *accidental* disclosure. Multiline secrets, such as the contents of a SSH private key file, are not masked. See the inline help for details and usage guidelines.

Bindings

Secret text

Variable:

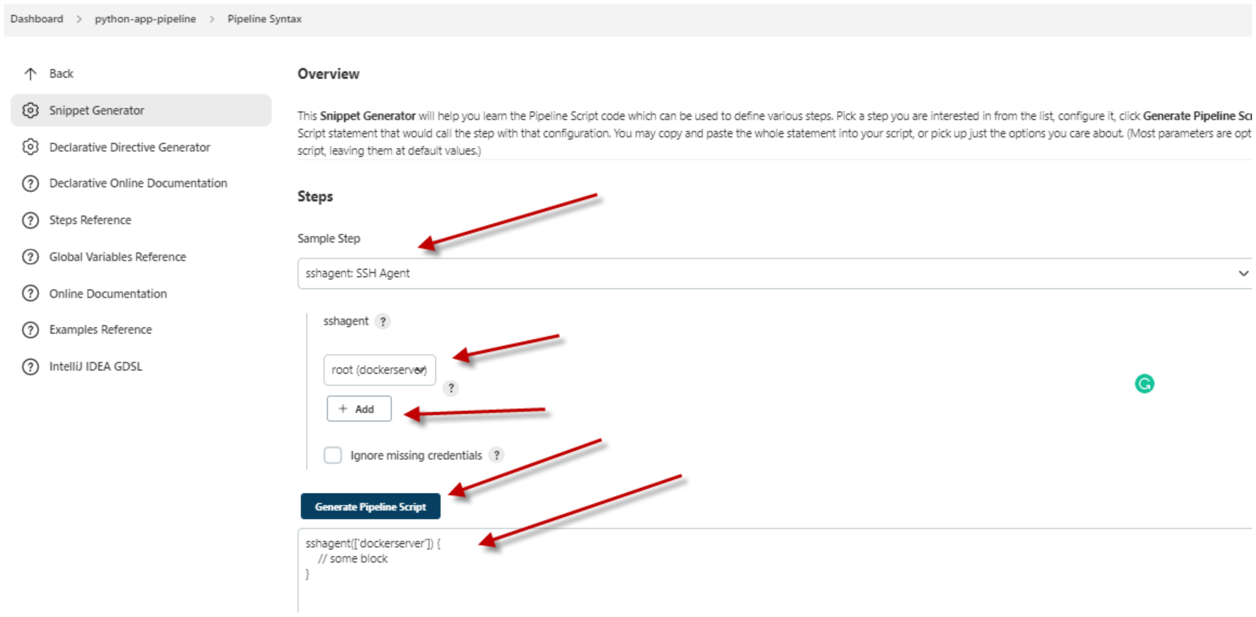
Credentials: dockerpassword1

```
withCredentials([string(credentialsId: 'dockerpassword1', variable: '')]) {
   // some block
}
```

Step-5: Now we Generate pipeline script for **SSHAgent**
From pipeline syntax-
From Sample Step to select **sshagent: SSH Agent**
Click Add—Jenkins
From kind select **SSH username with private key**
ID= `dockerserver`
Username= root (provide jenkins server username)
From Private Key select Enter directly
Click add and provide private key from your Jenkins server.
How to find private key from Jenkins server:
Login Jenkins server
#cd .ssh
#ls
#cat id_rsa
Here you find private key select all and paste it.

After provide private key click add.
Then click Generate pipeline Script

SSH Agent syntax done! See below pic you got output look like this…


Dashboard > python-app-pipeline > Pipeline Syntax
Back
Snippet Generator
Declarative Directive Generator
Declarative Online Documentation
Steps Reference
Global Variables Reference
Online Documentation
Examples Reference
IntelliJ IDEA GDSL
Overview
Steps
Add
Generate Pipeline Script

This **Snippet Generator** will help you learn the Pipeline Script code which can be used to define various steps. Pick a step you are interested in from the list, configure it, click **Generate Pipeline Sc** Script statement that would call the step with that configuration. You may copy and paste the whole statement into your script, or pick up just the options you care about. (Most parameters are opt script, leaving them at default values.)

Sample Step: sshagent: SSH Agent

sshagent: root (dockerserver)

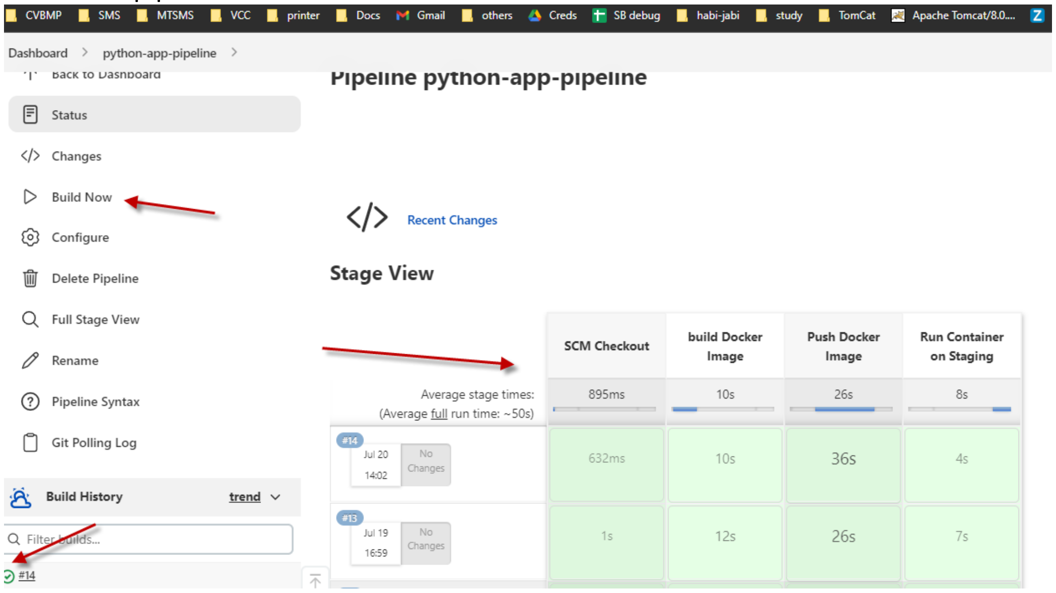
Ignore missing credentials

```
sshagent(['dockerserver']) {
    // some block
}
```

Step-6: We have almost near to build our pipeline. Before bulding pipeline we need to give permission to our Jenkins server for running Docker Deamon. Let’s do it.
Login Jenkins server:
#usermod –aG docker Jenkins
#reboot

After rebooting the server login your Jenkins from web browser and bulid your pipeline from build now section. If job build is success that’s means your configuration ok, if any failed check log and resolve it.

Successful pipeline build look like this…

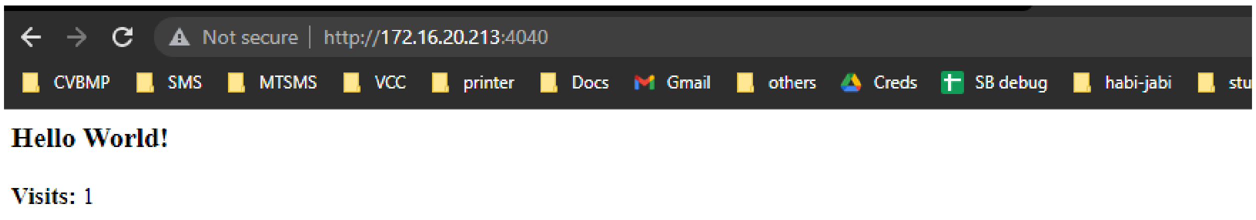


Pipeline python-app-pipeline

Stage View

	SCM Checkout	build Docker Image	Push Docker Image	Run Container on Staging
Average stage times: (Average full run time: ~50s)	895ms	10s	26s	8s
#14 Jul 20 14:02 No Changes	632ms	10s	36s	4s
#13 Jul 19 16:59 No Changes	1s	12s	26s	7s

For check open your browser and write your application server IP with 4040 port!



Hello World!

Visits: 1

Every visit will be count!!!!!

Project Done!

Note: This documents created by Joy Kumar and it's totally free for everyone. You can share with everyone without any hesitance. Have a nice day!

Caution/Warning: This documents user for commercial purpose is strictly probhited.