simplilearn

Caltech | Center for Technology & Management Education

Build a Docker Jenkins Pipeline to Implement CI/CD Workflow

Submitted by
Ramya shanmugam

Contents

1. Introduction to the Project

Demonstrate the continuous integration and delivery by building a Docker Jenkins Pipeline.

1. Availability of the application and its versions in the GitHub
    - Track their versions every time a code is committed to the repository
2. Create a Docker Jenkins Pipeline that will create a Docker image from the Dockerfile and host it on Docker Hub
3. It should also pull the Docker image and run it as a Docker container
4. Build the Docker Jenkins Pipeline to demonstrate the continuous integration and continuous delivery workflow

Project Goal

deliver the product frequently to the production with high-end quality.

Tools Required

- Docker: To build the application from a Dockerfile and push it to Docker Hub
- Docker Hub: To store the Docker image
- GitHub: To store the application code and track its revisions
- Git: To connect and push files from the local system to GitHub
- Linux (Ubuntu): As a base operating system to start and execute the project
- Jenkins: To automate the deployment process during continuous integration

2. Installation of pre-requisites/tools

In this section we can see how the required tools are installed to execute the project

Note: This project is delivered in the “DevOps in AWS” Lab provided by SimpliLearn, so most of the tools may be already installed including the Ubuntu

2.1 Installing Git

Step 1: Verifying the Git installation

Use the following command to check the version of Git:

git --version

```
ramyashanmugam1@ip-172-31-17-74:~$ git --version
git version 2.31.1
ramyashanmugam1@ip-172-31-17-74:~$
```

Step 2: Installing the latest version of Git

Execute the following commands on the terminal to install Git:

sudo apt-get update

sudo apt-get install git

```
asmitaraysimpli@ip-172-31-79-103:~$ sudo apt-get update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu xenial InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu xenial-updates InRelease
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu xenial-backports InRelease [107 kB]
Hit:4 http://security.ubuntu.com/ubuntu xenial-security InRelease
Hit:5 https://deb.nodesource.com/node_14.x xenial InRelease
Hit:6 http://repo.zabbix.com/zabbix/3.0/ubuntu trusty InRelease
Hit:7 http://ppa.launchpad.net/remmina-ppa-team/remmina-next-daily/ubuntu xenial InRelease
Fetched 107 kB in 0s (213 kB/s)
Reading package lists... Done
W: http://repo.zabbix.com/zabbix/3.0/ubuntu/dists/trusty/InRelease: Signature by key FBABD5FB20255ECAB22EE194D13D58E479EA5ED4 uses wea
k digest algorithm (SHA1)
asmitaraysimpli@ip-172-31-79-103:~$ sudo apt-get install git
Reading package lists... Done
Building dependency tree
Reading state information... Done
git is already the newest version (1:2.7.4-0ubuntu1.10).
The following packages were automatically installed and are no longer required:
  linux-aws-headers-4.4.0-1118 linux-headers-4.4.0-1118-aws linux-image-4.4.0-1118-aws linux-modules-4.4.0-1118-aws
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 46 not upgraded.
asmitaraysimpli@ip-172-31-79-103:~$
```

2.2 Creating GitHub Account

Make sure you have a Github Account available. If not, please create one using the given link.

https://github.com/

2.3 **Setting up Jenkins**

Step 1: Downloading the Java Runtime Environment

1. Open the terminal.
2. Run ***sudo apt-get update*** to update the package lists.
3. Run ***sudo apt-get install openjdk-8-jdk*** to install the Java Runtime Environment.
4. Run ***java -version*** to verify the installation. It will print the JDK version as shown below:

```
susmitaadhyapak@susmitaadhyapak:~$ java -version
openjdk version "1.8.0_232"
OpenJDK Runtime Environment (build 1.8.0_232-8u232-b09-0ubuntu1~16.04.1-b09)
OpenJDK 64-Bit Server VM (build 25.232-b09, mixed mode)
```

Step 2: Downloading and installing the Jenkins app

2.1 Open the terminal.

2.2 Run ***wget -q -O - https://pkg.jenkins.io/debian/jenkins.io.key | sudo apt-key add -*** to install Jenkins.

```
susmitaadhyapak@susmitaadhyapak:~$ wget -q -O - https://pkg.jenkins.io/debian/je
nkins.io.key | sudo apt-key add -
```

2.3 Run ***sudo sh -c 'echo deb http://pkg.jenkins-ci.org/debian binary/ > /etc/apt/sources.list.d/jenkins.list'*** **command.**

```
susmitaadhyapak@susmitaadhyapak:~$  sudo sh -c 'echo deb http://pkg.jenkins.io/d
ebian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'
```

2.4 Run *sudo apt-get update*

2.5 Run *sudo apt-get install jenkins* to install Jenkins.

```
susmitaadhyapak@susmitaadhyapak:~$ sudo apt-get install jenkins
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  fonts-lato javascript-common jruby libbytelist-java libhawtjni-runtime-java
  libheadius-options-java libinvokebinder-java libjansi-java
  libjansi-native-java libjcodings-java libjffi-java libjffi-jni
  libjnr-constants-java libjnr-enxio-java libjnr-ffi-java libjnr-netdb-java
  libjnr-posix-java libjnr-unixsocket-java libjnr-x86asm-java
  libjoda-time-java libjruby-joni-java libjs-jquery libjzlib-java liblua5.2-0
  libreadline7 libssl1.0.2 libtcl8.6 libunsafe-mock-java libyaml-snake-java
  libyecht-java nailgun rake vim-gui-common
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  daemon
The following NEW packages will be installed:
  daemon jenkins
0 upgraded, 2 newly installed, 0 to remove and 306 not upgraded.
Need to get 70.6 MB of archives.
After this operation, 71.2 MB of additional disk space will be used.
```

2.6 Run ***sudo service jenkins status*** to check the status of the installation. Once you verify the status as active, you can press ***Ctrl+z*** to exit from the process.

```
susmitaadhyapak@susmitaadhyapak:~$ sudo service jenkins status
● jenkins.service - LSB: Start Jenkins at boot time
   Loaded: loaded (/etc/init.d/jenkins; bad; vendor preset: enabled)
   Active: active (exited) since Tue 2021-03-23 08:02:03 UTC; 1min 51s ago
     Docs: man:systemd-sysv-generator(8)

Mar 23 08:02:01 susmitaadhyapak systemd[1]: Starting LSB: Start Jenkins at boot
Mar 23 08:02:01 susmitaadhyapak jenkins[5672]: Correct java version found
Mar 23 08:02:01 susmitaadhyapak jenkins[5672]:  * Starting Jenkins Automation Se
Mar 23 08:02:01 susmitaadhyapak su[5737]: Successful su for jenkins by root
Mar 23 08:02:01 susmitaadhyapak su[5737]: + ??? root:jenkins
Mar 23 08:02:01 susmitaadhyapak su[5737]: pam_unix(su:session): session opened f
Mar 23 08:02:03 susmitaadhyapak jenkins[5672]:    ...done.
Mar 23 08:02:03 susmitaadhyapak systemd[1]: Started LSB: Start Jenkins at boot t
lines 1-13/13 (END)
```

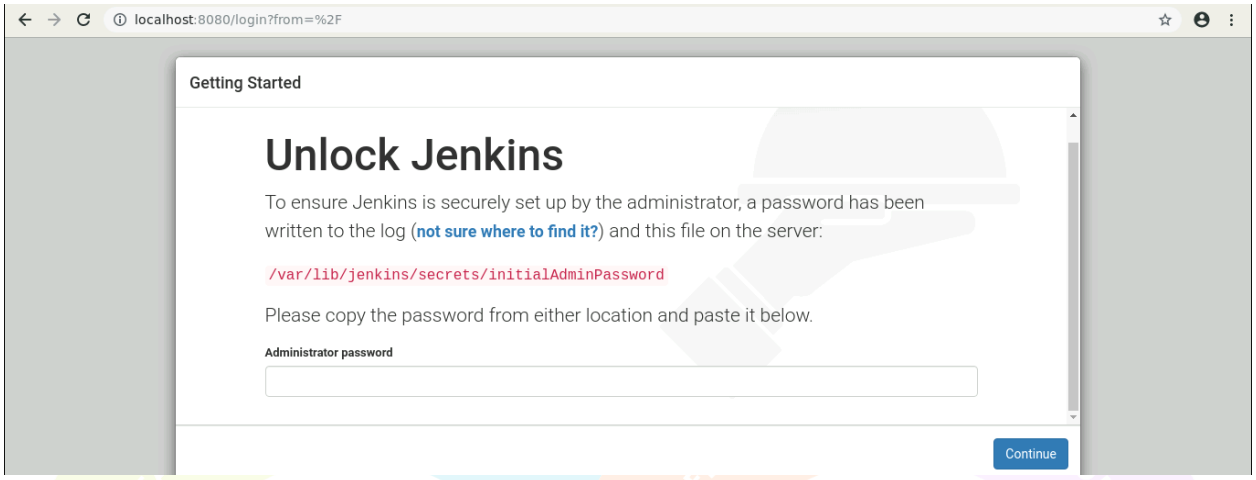
2.7 Run the following commands to start Jenkins.

sudo systemctl start jenkins

sudo systemctl status jenkins

```
susmitaadhyapak@susmitaadhyapak:~$ sudo systemctl start jenkins
susmitaadhyapak@susmitaadhyapak:~$ sudo systemctl status jenkins
● jenkins.service - LSB: Start Jenkins at boot time
   Loaded: loaded (/etc/init.d/jenkins; bad; vendor preset: enabled)
   Active: active (exited) since Tue 2021-03-23 08:02:03 UTC; 4min 2s ago
     Docs: man:systemd-sysv-generator(8)

Mar 23 08:02:01 susmitaadhyapak systemd[1]: Starting LSB: Start Jenkins at boot
Mar 23 08:02:01 susmitaadhyapak jenkins[5672]: Correct java version found
Mar 23 08:02:01 susmitaadhyapak jenkins[5672]:  * Starting Jenkins Automation Se
Mar 23 08:02:01 susmitaadhyapak su[5737]: Successful su for jenkins by root
Mar 23 08:02:01 susmitaadhyapak su[5737]: + ??? root:jenkins
Mar 23 08:02:01 susmitaadhyapak su[5737]: pam_unix(su:session): session opened f
Mar 23 08:02:03 susmitaadhyapak jenkins[5672]:    ...done.
Mar 23 08:02:03 susmitaadhyapak systemd[1]: Started LSB: Start Jenkins at boot t
Mar 23 08:05:53 susmitaadhyapak systemd[1]: Started LSB: Start Jenkins at boot t
lines 1-14/14 (END)
```

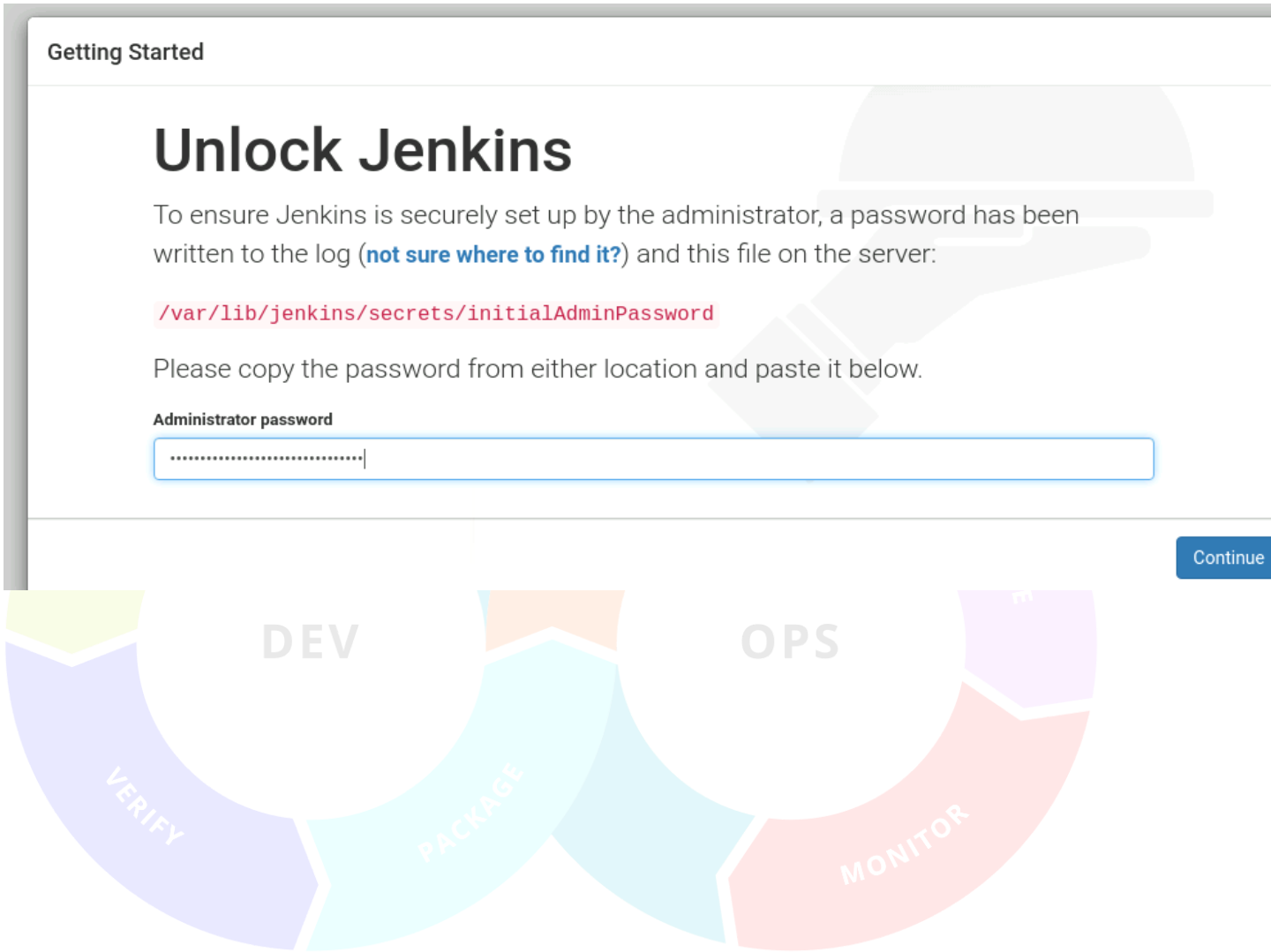
2.8 Open localhost:8080 in the browser, and you will need to enter the initial password.



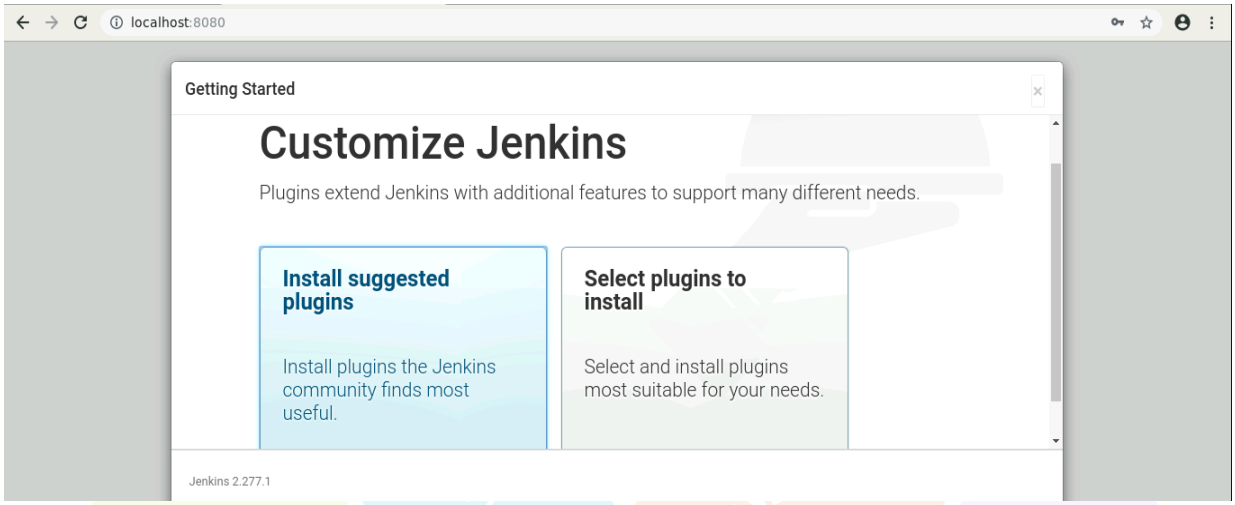
2.9 In your terminal run the following command:
sudo cat /var/lib/jenkins/secrets/initialAdminPassword

```
susmitaadhyapak@susmitaadhyapak:~$ sudo cat /var/lib/jenkins/secrets/initialAdmi
nPassword
876821d4689a453c87c48116a59a001b
susmitaadhyapak@susmitaadhyapak:~$
```

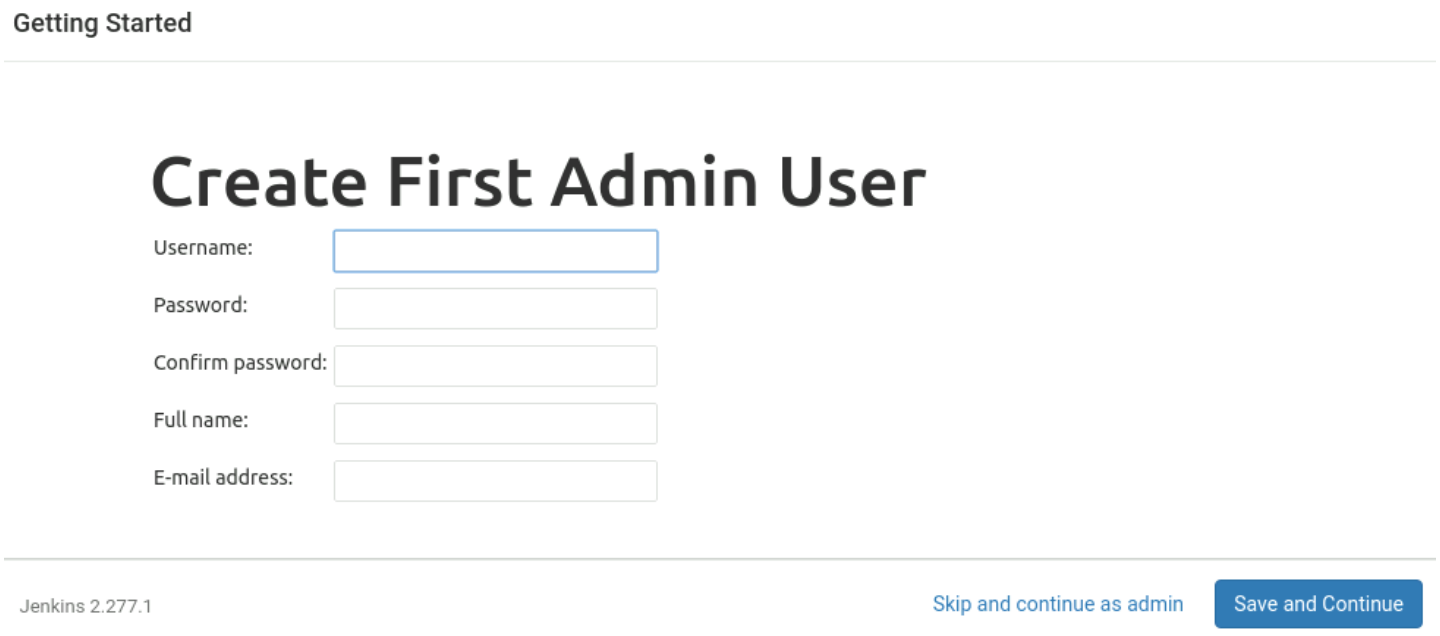
2.10 Copy this password and paste it on your Jenkins page in the browser.



2.11 Now, click on Install the suggested plugins.



2.12 You can either create an admin user or skip and continue as admin. Select Skip and continue as admin.



2.13 In the Instance configuration page, click on the Start using Jenkins button.

Getting Started

Jenkins is ready!

You have skipped the **setup of an admin user**.

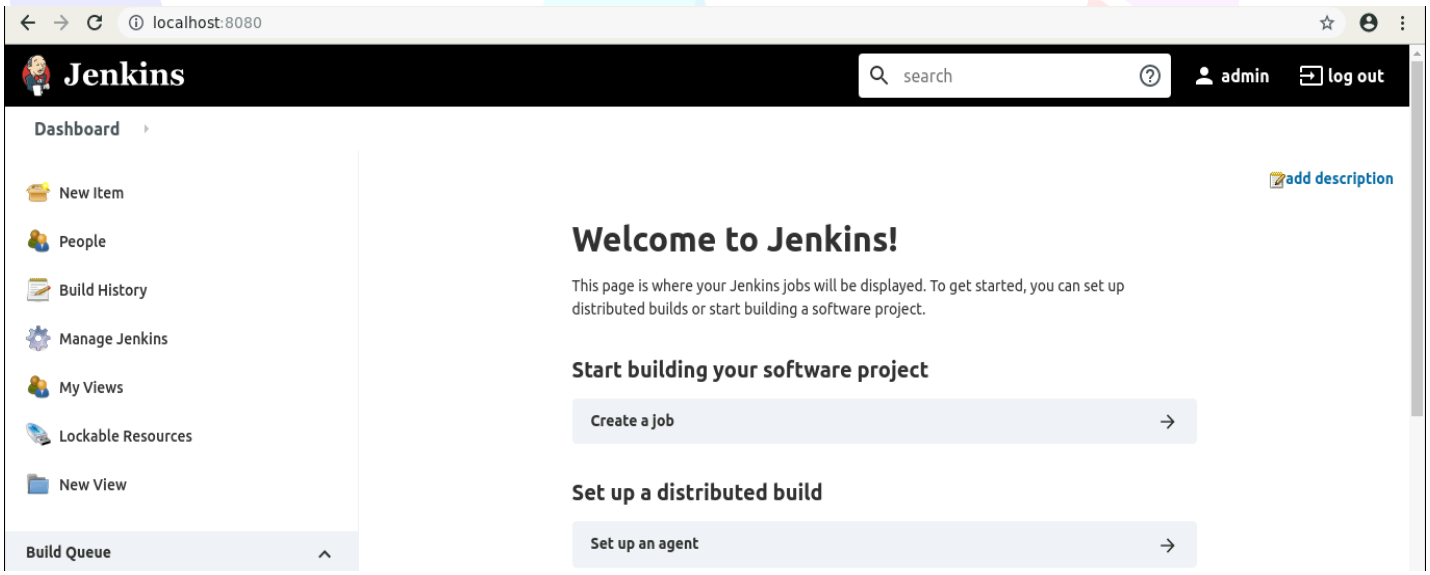
To log in, use the username: "admin" and the administrator password you used to access the setup wizard.

Your Jenkins setup is complete.

Start using Jenkins

Jenkins 2.277.1

2.14 Now, you can work with Jenkins as shown in the screenshot below.



2.4 Docker Community Edition Installation

Step 1: Install the Docker CE from Docker Repository

1.1 Use the following command to update the apt package:

sudo apt-get update

```
labsuser@ip-172-31-29-216:~$ sudo apt-get update
Hit:1 http://us-west-2.ec2.archive.ubuntu.com/ubuntu bionic InRelease
Get:2 http://us-west-2.ec2.archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:3 http://us-west-2.ec2.archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
Hit:4 https://download.docker.com/linux/ubuntu bionic InRelease

Hit:5 http://dl.google.com/linux/chrome/deb stable InRelease

Hit:6 http://packages.microsoft.com/repos/vscode stable InRelease
```

1.2 Use the following command to install packages to allow the apt to use a repository over HTTPS

```
sudo apt-get install \

apt-transport-https \

ca-certificates \

 curl \

gnupg \

lsb-release
```

```
[redacted]~$ sudo apt-get install \
> ca-certificates \
> curl \
> gnupg \
> lsb-release
Reading package lists... Done
Building dependency tree
Reading state information... Done
apt-transport-https is already the newest version (1.2.32ubuntu0.2).
ca-certificates is already the newest version (20210119~16.04.1).
curl is already the newest version (7.47.0-1ubuntu2.19).
gnupg is already the newest version (1.4.20-1ubuntu3.3).
lsb-release is already the newest version (9.20160110ubuntu0.2).
0 upgraded, 0 newly installed, 0 to remove and 52 not upgraded.
[redacted]~$
```

1.3 Use the following curl command to add Docker's official GPG key:

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg

```
rishabhpathaksi@ip-172-31-66-101:~$  curl -fsSL https://download.docker.com/linu
x/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.
gpg
```

1.4 Use the following command to set up a stable repository:

echo \ "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu \ $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

```
rishabhpathaksi@ip-172-31-66-101:~$ echo \
>   "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] h
ttps://download.docker.com/linux/ubuntu \
>
> $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /d
ev/null
rishabhpathaksi@ip-172-31-66-101:~$
```

1.5 Use the following commands to install the latest version of Docker CE and check the version:

sudo apt-get install docker-ce

docker --version

```
labsuser@ip-172-31-29-216:~$ sudo apt-get install docker-ce
Reading package lists... Done
Building dependency tree
Reading state information... Done
docker-ce is already the newest version (5:20.10.2~3-0~ubuntu-bionic).
The following packages were automatically installed and are no longer required:
  liblttng-ust-ctl4 liblttng-ust0 liburcu6 linux-aws-5.3-headers-5.3.0-1019
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 261 not upgraded.
2 not fully installed or removed.
After this operation, 0 B of additional disk space will be used.
Do you want to continue? [Y/n] Y
Setting up docker-ce (5:20.10.2~3-0~ubuntu-bionic) ...
Setting up docker-ce-rootless-extras (5:20.10.2~3-0~ubuntu-bionic) ...
W: APT had planned for dpkg to do more than it reported back (3 vs 7).
   Affected packages: docker-ce:amd64
labsuser@ip-172-31-29-216:~$ docker --version
Docker version 19.03.11, build 42e35e61f3
labsuser@ip-172-31-29-216:~$
```

Step 2: Verify the correctly installed Docker engine

2.1 Use the following command to verify the Docker engine:

sudo docker run hello-world

```
labsuser@ip-172-31-29-216:~$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
0e03bdcc26d7: Pull complete
Digest: sha256:1a523af650137b8accdaed439c17d684df61ee4d74feac151b5b337bd29e7eec
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/

labsuser@ip-172-31-29-216:~$
```

Step 3: Create a Docker Hub account to store or get images from remote repository

Go to https://hub.docker.com/ and create your own account

3 Execution of the Project

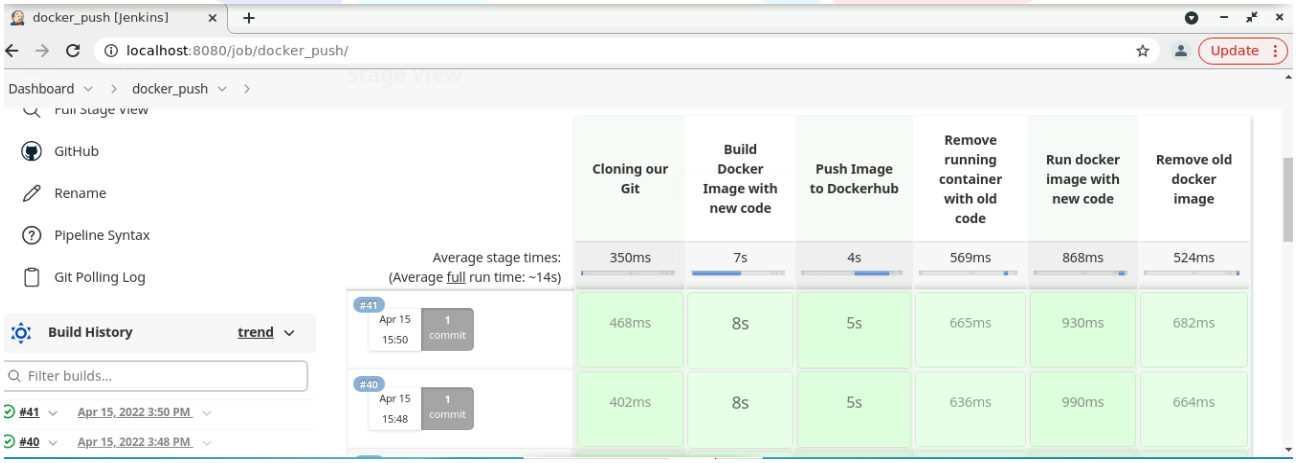
All the necessary code is created in the below GitHub repository, this repo can be cloned to execute the Jenkins pipeline

https://github.com/ramyaashanmugam/ramyaashanmugam.git

The Repo contains a sample Flask application and when the Jenkins pipeline runs, flask application is run and it is deployed in `python:alpine3.7` docker image(docker hub).

Any commit made in the repo automatically triggers the Jenkins pipeline, which does all the actions of below:

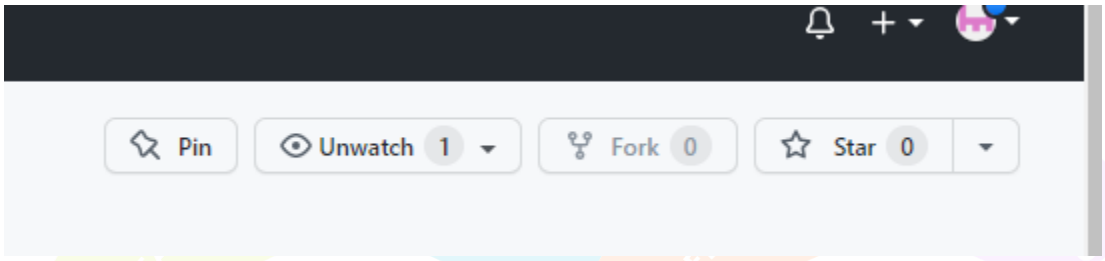
1. Create build with jenkins pipeline by using the new code committed
2. Create a new `python:alpine3.7` docker image
3. Push the new image to docker hub
4. Remove already running container in the server with old code
5. Deploy the new code with the updated version of image from Docker Hub
6. Remove the old image which has stored locally



Step 1:

Fork the Github project

By clicking the Fork button as shown below in github, you can fork the project into your github account



Step 2:

with $git clone https://github.com/ramyaashanmugam/ramyaashanmugam.git command you can clone the project into your machine and the changes to be made in Jenkinsfile available in the repository

Step 3:

Change registry name with the docker account ID in Jenkinsfile

1 contributor

99 lines (61 sloc) | 1.98 KB

```
pipeline {

    environment {

        registry = "ramyaashanmugam/flask_app"
        name="flask_app"

        registryCredential = 'docker_hub'

        dockerImage = ''

    }

```

Step 4:

Change the repo URL in the Jenkinsfile , this should be your forked Repo URL

```
stages {

    stage('Cloning our Git') {

    steps {

        git branch: 'main', credentialsId: 'github_access', url: 'https://github.com/ramyaashanmugam/ramyaashanmugam.git'


     }
    }
```

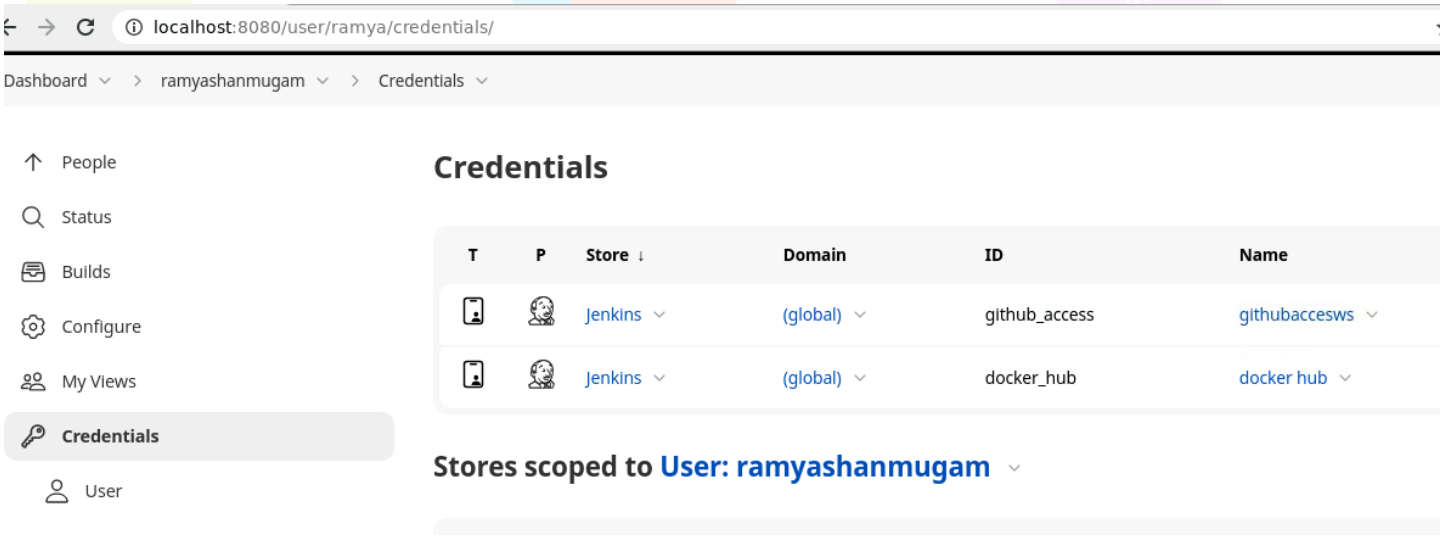
Step 5:

Docker Hub Credentials to be maintained in Jenkins server as mentioned in the *Jenkinsfile*

```
stage('Push Image to Dockerhub') {

    steps {

        script {

            docker.withRegistry( '', registryCredential ) {

                dockerImage.push()

            }

        }

    }
}
```

```
environment {

    registry = "ramyaashanmugam/flask_app"
    name="flask_app"

    registryCredential = 'docker_hub'

    dockerImage = ''

}
```

In the Jenkins server , go to **Manage Jenkins -> Manage Credentials** to maintain the Docker hub userID and Password with the same Credential ID “docker_hub”as maintained in Jenkinsfile of the repository



Step 6:

The user jenkins needs to be added to the group docker with the below linux command:

sudo usermod -a -G docker Jenkins

Note: If this is not done, then the Jenkins server cannot create docker containers and we will get permission error

You can check if the above command was successful by doing grep docker /etc/group and you can see something like below:

Docker:x:998:[user]

Step 7:

Review the Dockerfile available in the repo, which needs no change from your end.

Port :5001

main ramyaashanmugam / Dockerfile

ramyaashanmugam Update Dockerfile

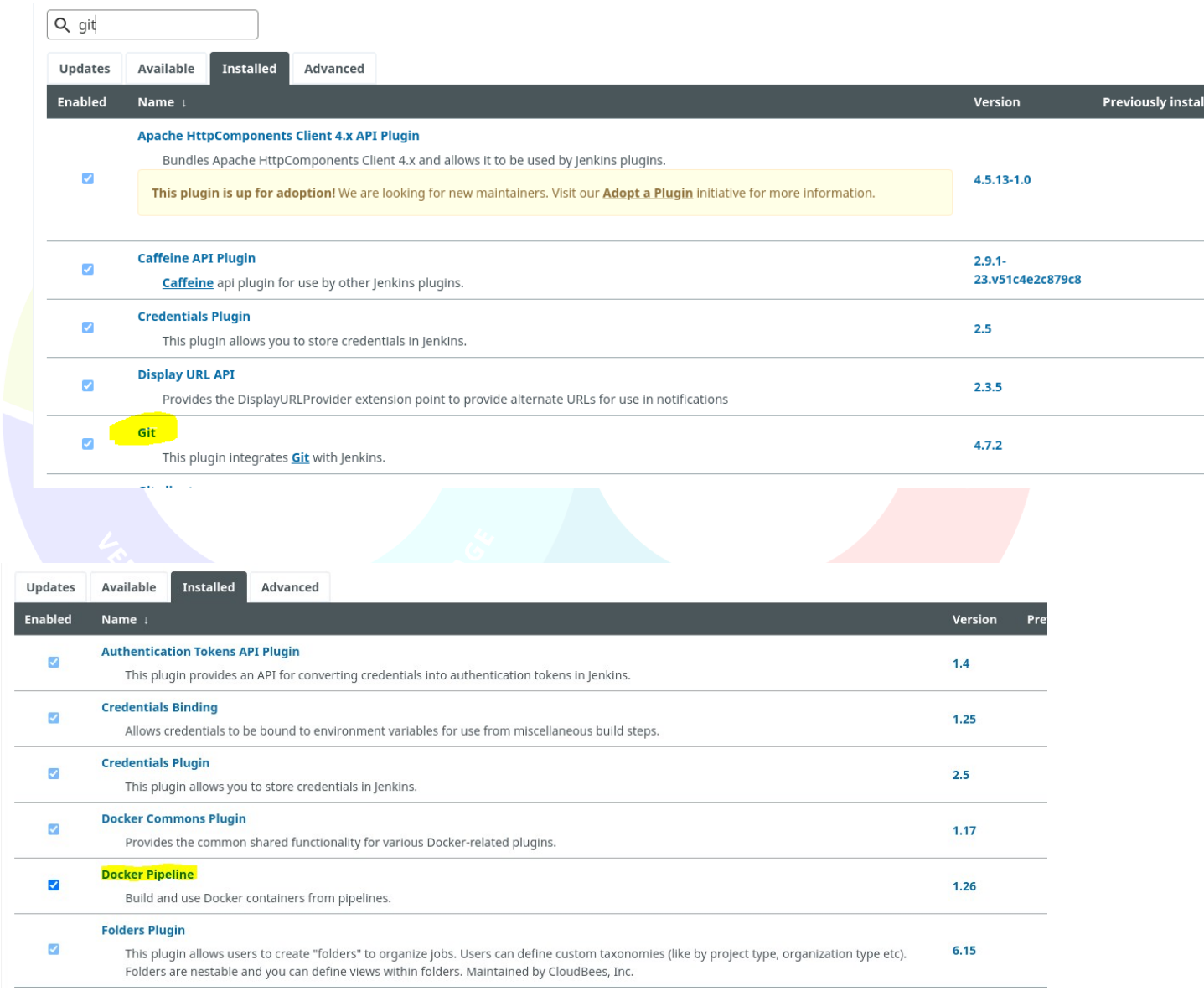
1 contributor

13 lines (7 sloc) | 156 Bytes

```
1  FROM python:alpine3.7
2
3  COPY . /app
4
5  WORKDIR /app
6
7  RUN pip install -r requirements.txt
8
9  EXPOSE 5001
10
11 ENTRYPOINT [ "python" ]
12
13 CMD [ "flask_pjt/weather.py" ]
```

Step 8:

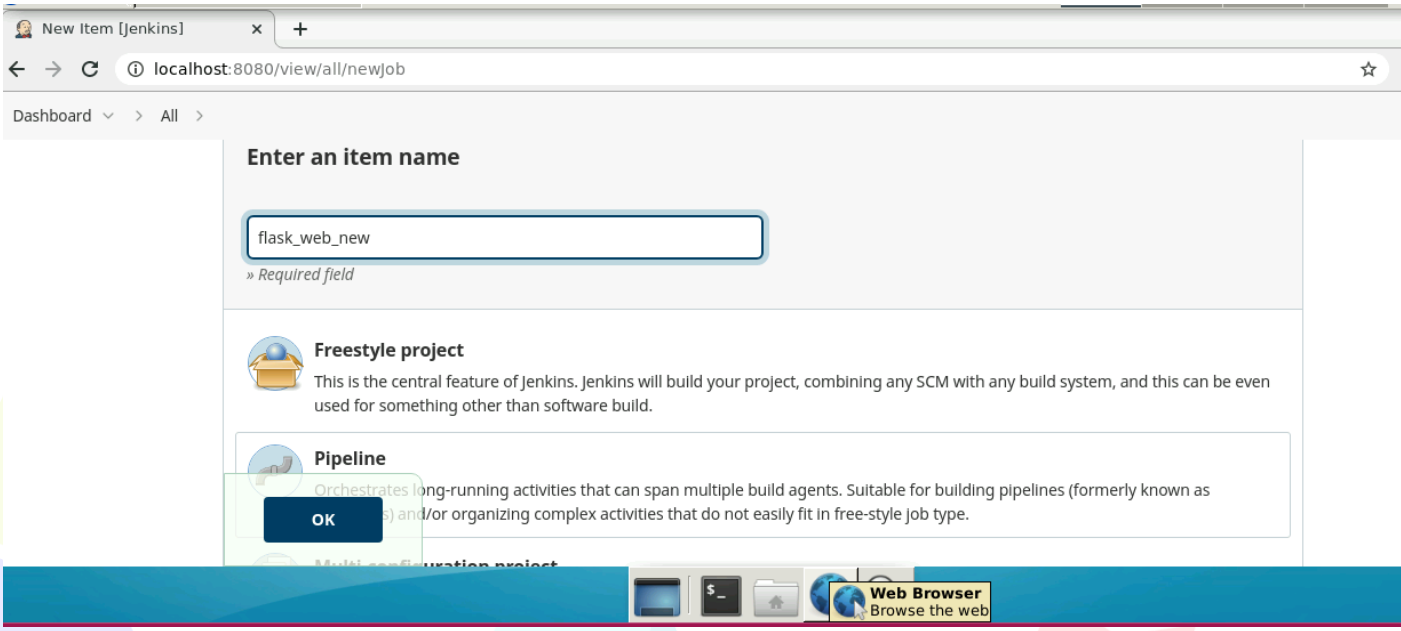
Install required Jenkins plugins if not installed already Use manage plugins option to install the plugins



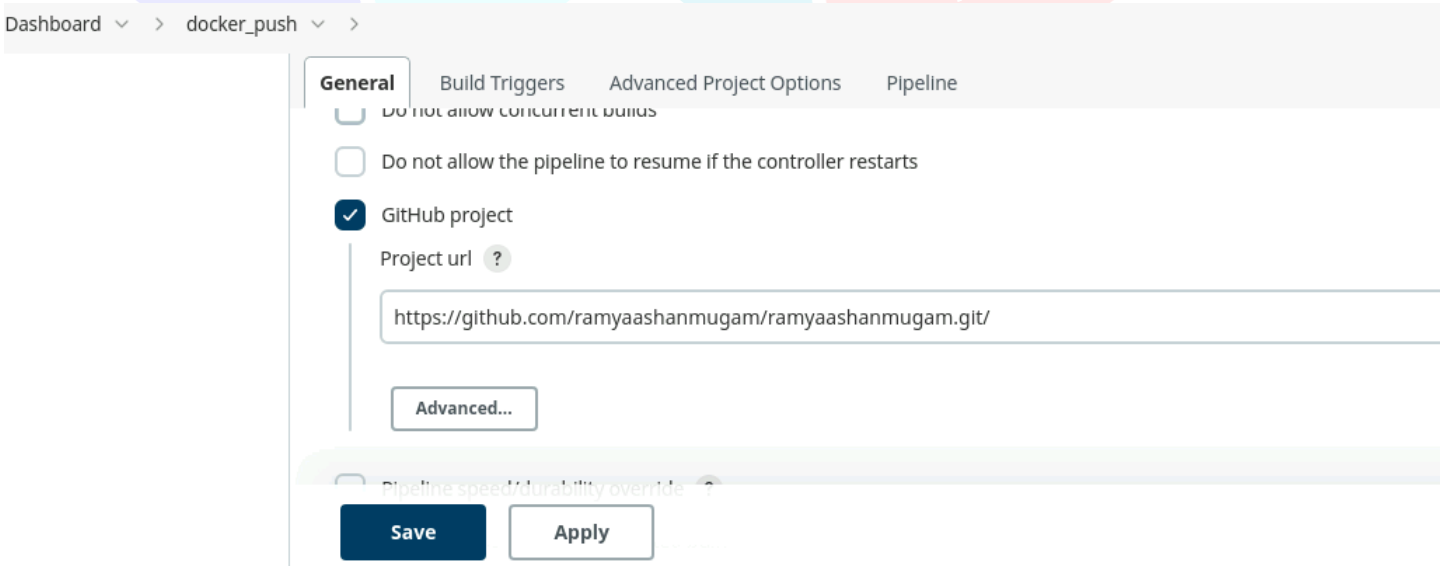
Step 9:

Create a Jenkins pipeline form the Git repository

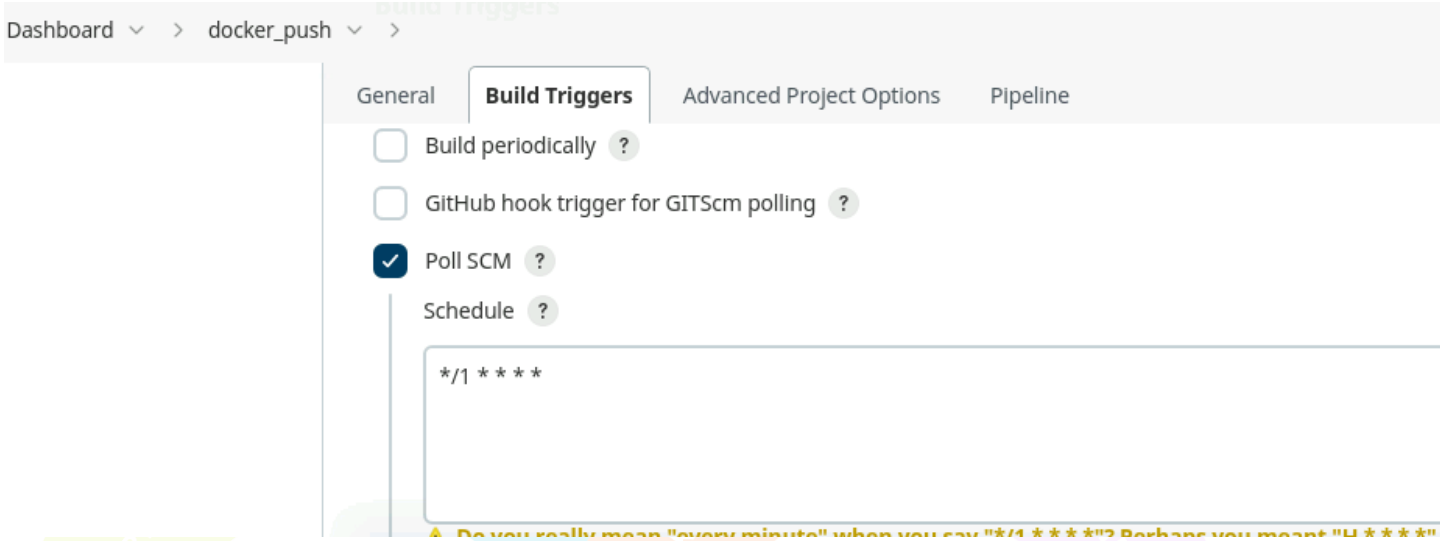
New Pipeline Project has to be created in Jenkins server as below



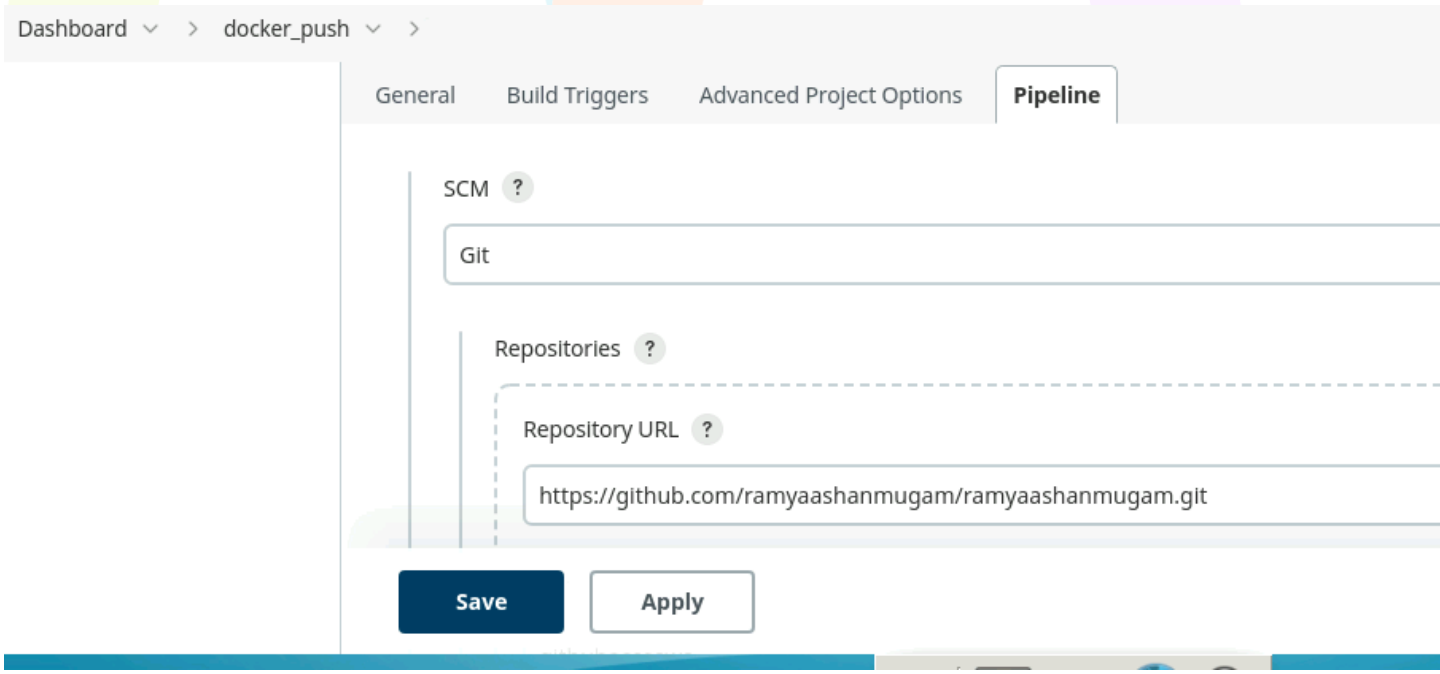
In General Tab, GitHub Project URL to be configured

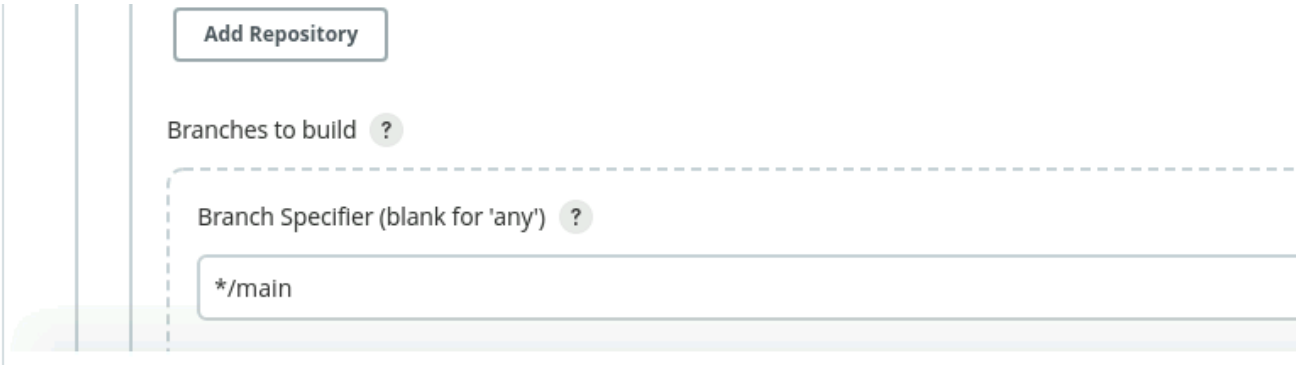


In the build triggers section, Poll SCM to be configured



In the pipeline section, our repo needs to be configured





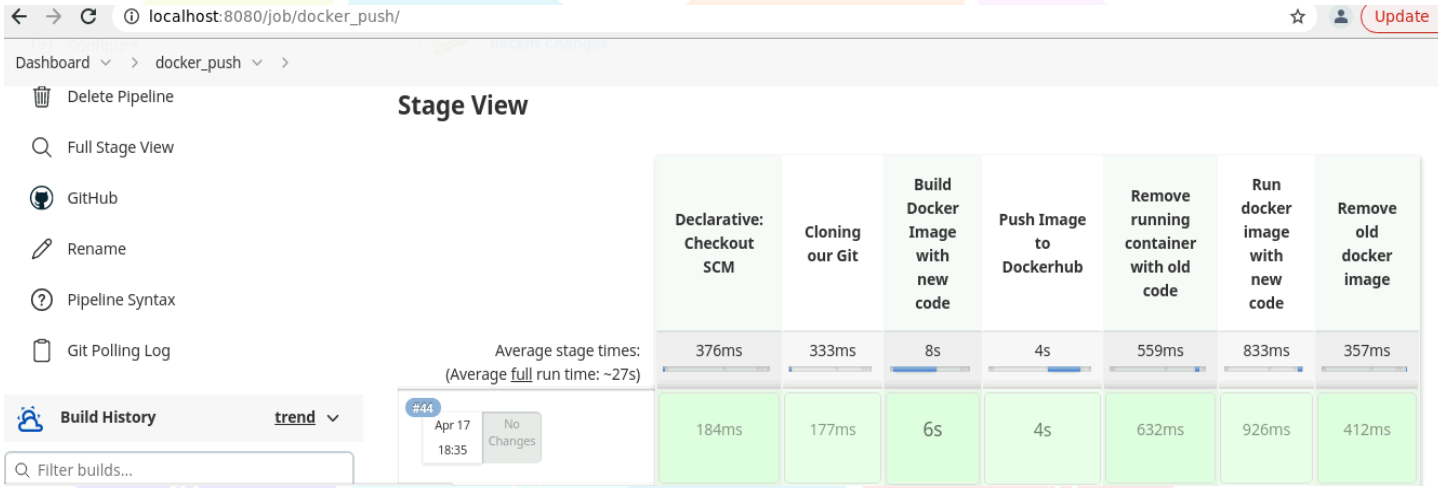
Step 10:

In out Repository the index.html this HTML file can be modified and the changes can be committed to the repository which will trigger the Jenkins Poll SCM job and the CI/CD process will be triggered to deploy the new application changes

4. Project Results

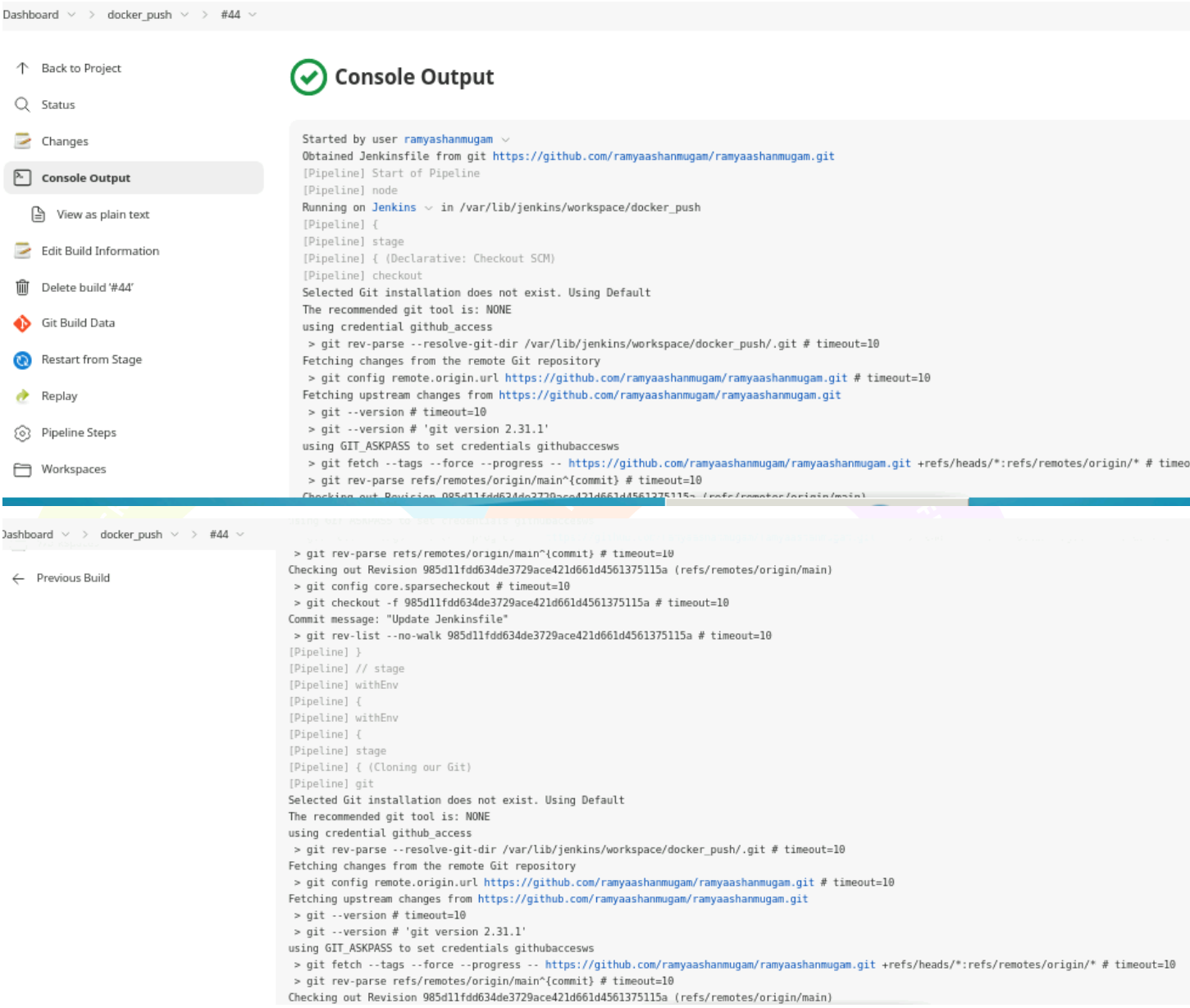
Result 1:

Jenkins job will be triggered automatically corresponding to the repository commit version, this can be seen in the Job build history



Result 2:

Every build and its console output can be seen to check the status of every Stage in the Jenkins pipeline


Dashboard
docker_push
#44
Back to Project
Status
Changes
Console Output
View as plain text
Edit Build Information
Delete build '#44'
Git Build Data
Restart from Stage
Replay
Pipeline Steps
Workspaces
Console Output
Started by user ramyashanmugam
Obtained Jenkinsfile from git https://github.com/ramyaashanmugam/ramyaashanmugam.git
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/docker_push
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Declarative: Checkout SCM)
[Pipeline] checkout
Selected Git installation does not exist. Using Default
The recommended git tool is: NONE
using credential github_access
> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/docker_push/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/ramyaashanmugam/ramyaashanmugam.git # timeout=10
Fetching upstream changes from https://github.com/ramyaashanmugam/ramyaashanmugam.git
> git --version # timeout=10
> git --version # 'git version 2.31.1'
using GIT_ASKPASS to set credentials githubaccesws
> git fetch --tags --force --progress -- https://github.com/ramyaashanmugam/ramyaashanmugam.git +refs/heads/*:refs/remotes/origin/* # timeo
> git rev-parse refs/remotes/origin/main^{commit} # timeout=10

Dashboard
docker_push
#44
Previous Build
> git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision 985d11fdd634de3729ace421d661d4561375115a (refs/remotes/origin/main)
> git config core.sparsecheckout # timeout=10
> git checkout -f 985d11fdd634de3729ace421d661d4561375115a # timeout=10
Commit message: "Update Jenkinsfile"
> git rev-list --no-walk 985d11fdd634de3729ace421d661d4561375115a # timeout=10
[Pipeline] }
[Pipeline] // stage
[Pipeline] withEnv
[Pipeline] {
[Pipeline] withEnv
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Cloning our Git)
[Pipeline] git
Selected Git installation does not exist. Using Default
The recommended git tool is: NONE
using credential github_access
> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/docker_push/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/ramyaashanmugam/ramyaashanmugam.git # timeout=10
Fetching upstream changes from https://github.com/ramyaashanmugam/ramyaashanmugam.git
> git --version # timeout=10
> git --version # 'git version 2.31.1'
using GIT_ASKPASS to set credentials githubaccesws
> git fetch --tags --force --progress -- https://github.com/ramyaashanmugam/ramyaashanmugam.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision 985d11fdd634de3729ace421d661d4561375115a (refs/remotes/origin/main)

Dashboard > docker_push > #44

```
Removing intermediate container f31e997360f1
 ---> 049b8347d07e
Step 7/7 : CMD [ "flask_pjt/weather.py" ]
 ---> Running in 22382d6b97c0
Removing intermediate container 22382d6b97c0
 ---> 3503696620cc
Successfully built 3503696620cc
Successfully tagged ramyaashanmugam/flask_app:44
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Push Image to Dockerhub)
[Pipeline] script
[Pipeline] {
[Pipeline] withEnv
[Pipeline] {
[Pipeline] withDockerRegistry
$ docker login -u ramyaashanmugam -p ******** https://index.docker.io/v1/
WARNING! Using --password via the CLI is insecure. Use --password-stdin.
WARNING! Your password will be stored unencrypted in /var/lib/jenkins/workspace/docker_push@tmp/4142777d-e30d-4ae2-ac7c-118dd57d0dfb/config.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store
```

Result 3:

You can check in the Docker Hub for the latest image version published, which is having the latest repository code

General | Tags | Builds | Collaborators | Webhooks | Settings

Advanced Image Management
View all your images and tags in this repository, clean up unused content, recover untagged images. Available with Pro, Team and Business subscriptions.

View

ramyaashanmugam / **flask_app**

flask_appln

Last pushed: 27 minutes ago

Docker commands

To push a new tag to this repository,

```
docker push ramyaashanmugam/flask_app:tagname
```

Tags and Scans

This repository contains 25+ tag(s).

VULNERABILITY SCANNING - DISABLED
Enable

TAG	OS	PULLED	PUSHED
44		---	27 minutes ago

Automated Builds

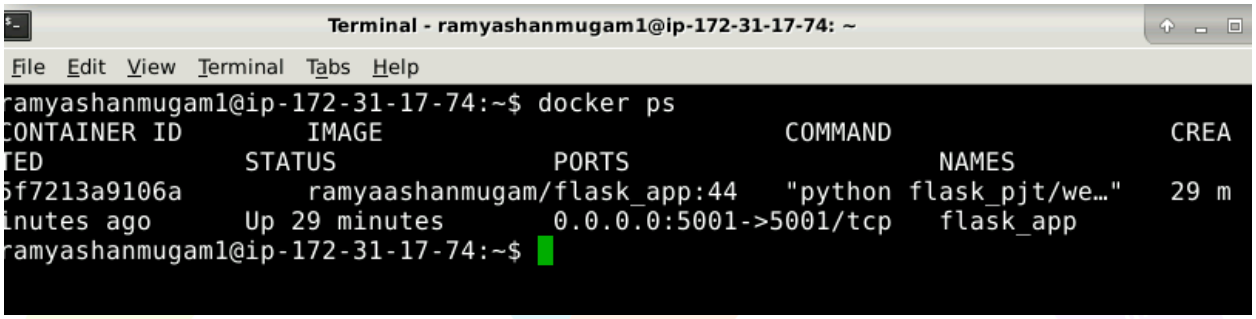
Manually pushing images to Hub? Connect your account to Git
Bitbucket to automatically build and tag new images whenever
code is updated, so you can focus your time on creating.

Available with Pro, Team and Business subscriptions.

Result 4:

You can see that the required application container is started and it is running with the port 5001 exposed, below command can be used to check this

$docker ps

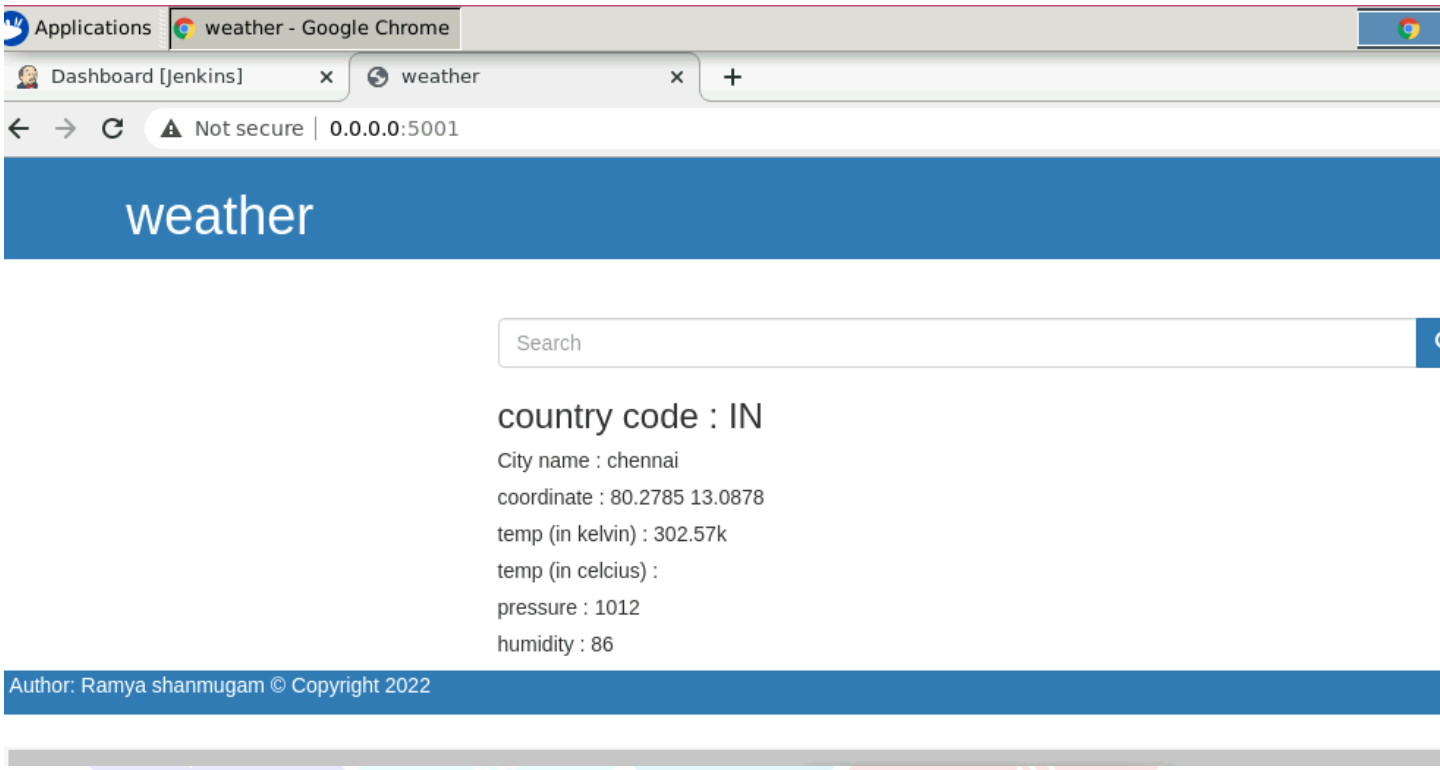


Above result also confirms that, always there is only one container with name **flask_app** is running at any given point in time. New version of changes does not create a new container, old container is first deleted and then the container with latest code version is created.

With below command, we can also see the related version of the image in local system

Result 5:

We can browse our flask application running and exposed in the URL http://0.0.0.0:5001/ Any new changes will be deployed automatically with our CI/CD pipeline with jenkins



5. Conclusion

Results shown is the previous section is the evidence that the Docker Jenkins Pipeline is implemented, enabling the CI/CD workflow.

Without any human intervention, incremental code can be delivered to production with Quality, Governance and Agility.