Jenkins Dockerfile Execution from Git SOP

Table of Contents

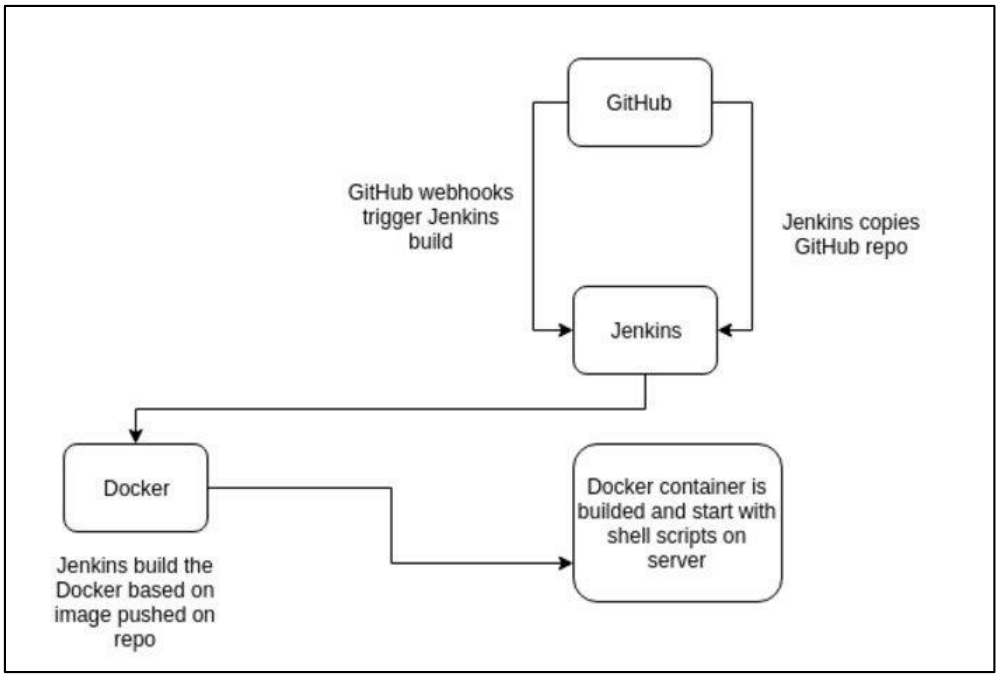
1. Jenkins Dockerfile Execution from Git

1.1 Description

In this lab exercise, we are creating docker build using the Dockerfile method and taking the sample application through git repo.

1.2 Architecture Diagram

The diagram below displays a visual representation of the underlying application architecture:



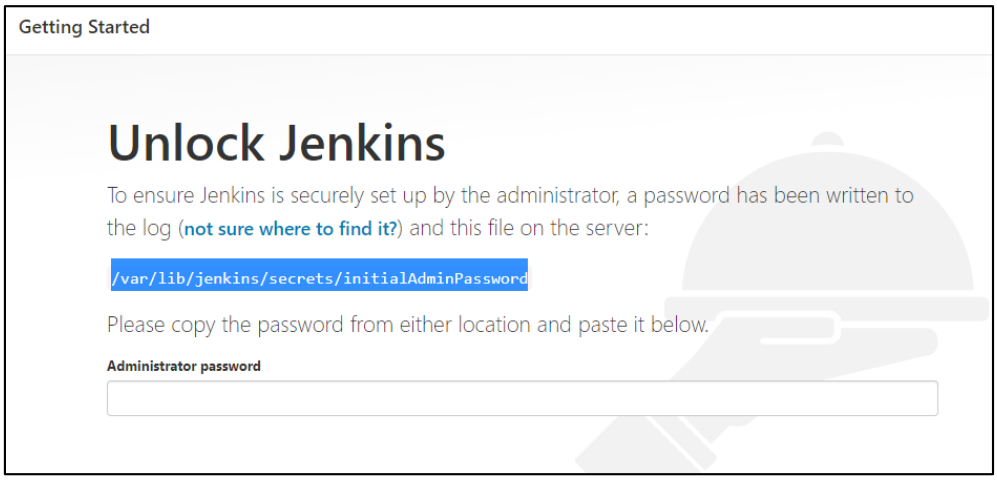
1.3 Lab Steps

Follow the steps outlined below to achieve the objectives of this lab exercise:

Prerequisites:

- Sign in to the AWS Management Console.
- An Amazon EC2 key pair, if you do not have one.

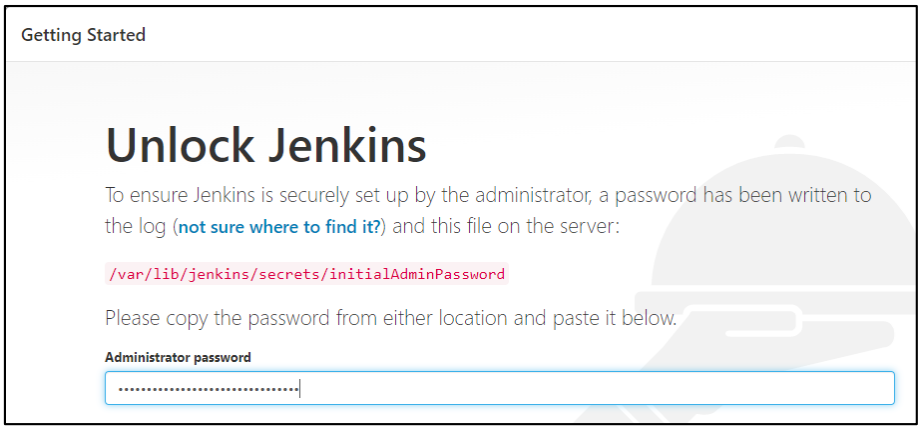
1. Install the Jenkins and docker in AWS machine and execute the build steps
    a. Execute the following steps to install the Jenkins in Centos 7 machine:
        i. Once you open your AWS machine or any other cloud instance get into root mode.
        ii. Enter the following address from your web browser to test Jenkins:
        http://< your_public_ip_address >:8080
        Once you enter the browser, the below screen displays:



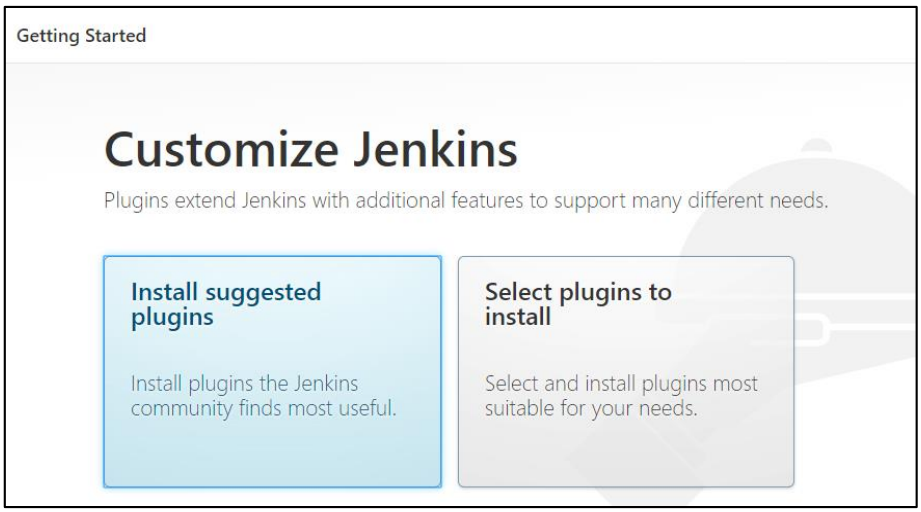
        iii. Type the following command in your Jenkins server and get the password and paste it in the Administrator password section in the above screen:

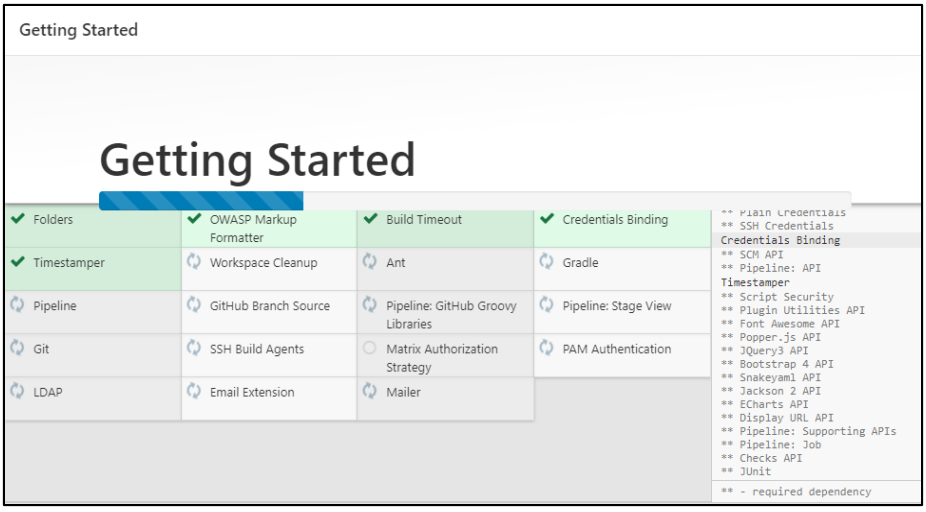
```
cat /var/lib/jenkins/secrets/initialAdminPassword
```

```
root@centos-jenkins ~]# cat /var/lib/jenkins/secrets/initialAdminPassword
2a2af0961044a57860dd977629db501
root@centos-jenkins ~]#
```

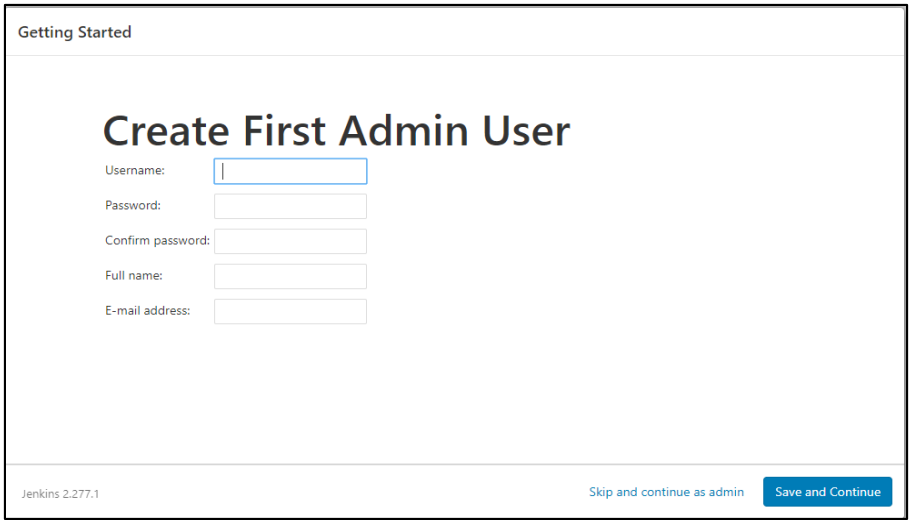


iv. Select the **Install suggested plugins** option:

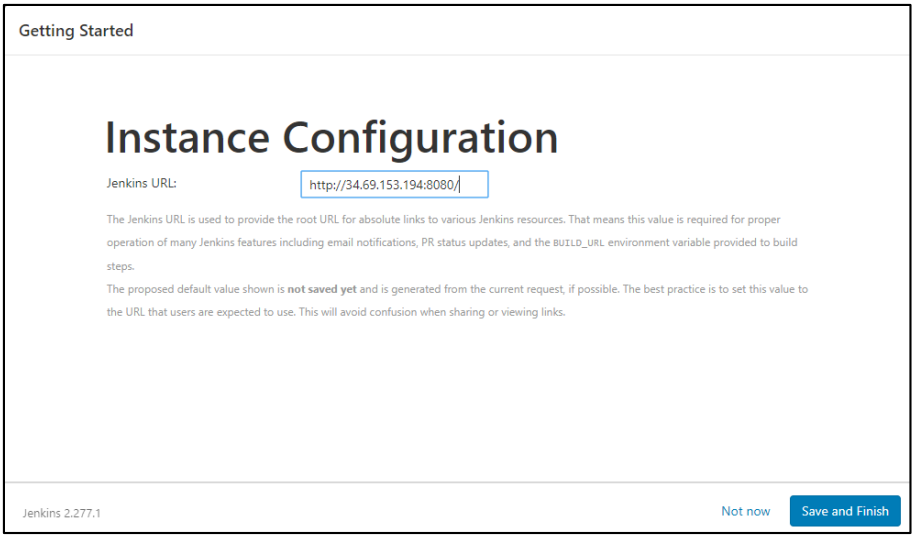


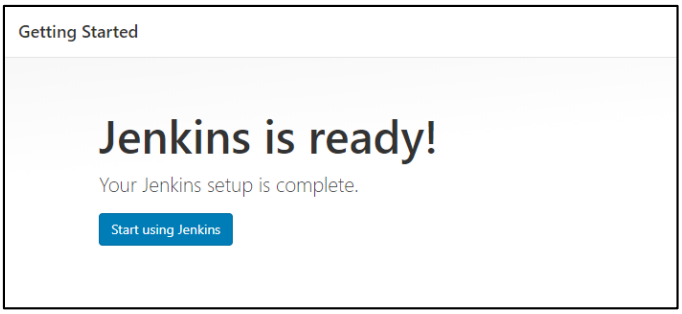


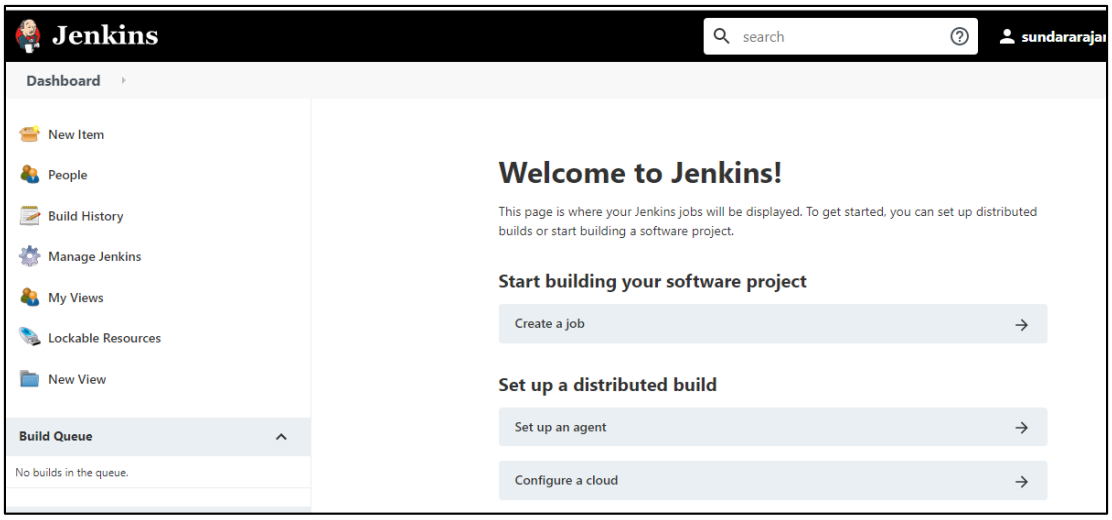
v. Click **Save and Continue** once you provided the required info in the screen:



vi. Click **Save and Finish** button:







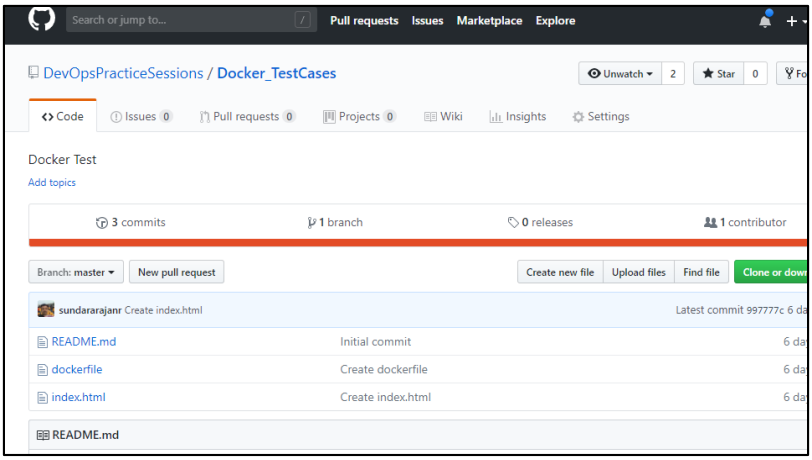
2. Install the docker in AWS VM machine using the command in the following attachment. It should support Centos 7 OS.

3. Follow the below steps to execute the Dockerfile through Jenkins.
   Task details: Jenkins job should perform the following steps.
   Git path: https://github.com/DevOpsPracticeSessions/Docker_TestCases.git

   a. Checkout the source code from GitHub repo.
      i. Repo contains the original source code (here in Practice-1, we are taking an example, one web application in terms of HTML pages only).
      ii. The same repo (where source code available) contains the Dockerfile which will create an environment for our web application as to run and test.
         1. Environment details to run & test our application: Ubuntu OS, apache web server installed on ubuntu, and then our web app will be deployed into this container/environment created by docker.
         2. Assume these are the same exact similar details as to compare with production. So that, we can test our apps which is exactly similar to the prod environment. Many issues will be solved during development stage.
   b. Build the project (compiling source code, packaging etc) if required. But in this example, we are using only HTML static files as our web app (to practice and understand how the docker will work). Build is not required for this example.
   c. Create an environment/container using docker.
      i. Build the Dockerfile, it will create an image.
      ii. Run the image, it will create a container.
      iii. Try to manually access the web application in any browser once the environment is fully setup and is up and running.

      Note: Jenkins is connecting and performing all the above three steps (three different configurations).

d. Create a New repo in GitHub:



e. Checkout the repo to local machine and then add Dockerfile and index.html files.

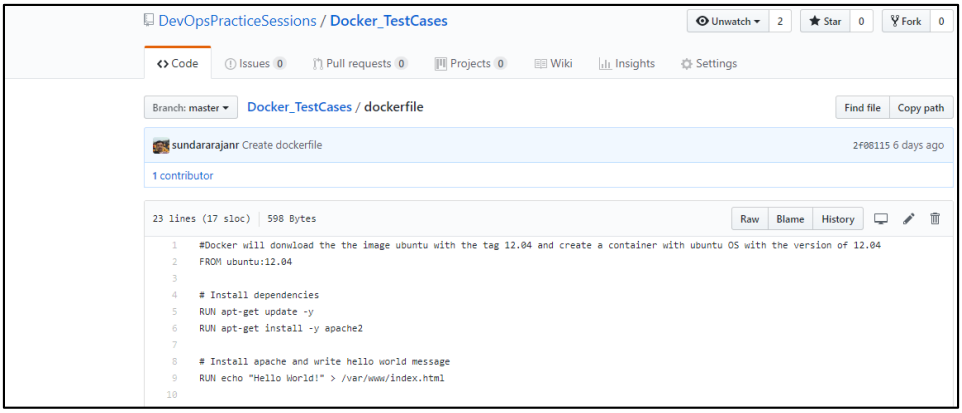
f. Content of **Dockerfile**:

i. FROM **ubuntu** – **image**: Docker will download the image **ubuntu** from docker hub (docker repo) with the latest tag/version. We can specify the image tag as "FROM **ubuntu:12.04**".

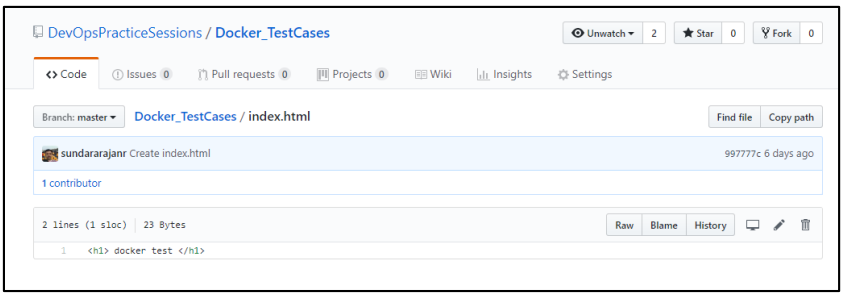
ii. RUN: Running the ubuntu commands as to update and install the required apps.

iii. ADD: Adding the local **index.html** file to container (where ubuntu installed) in to particular path (/var/www/html/) where apache can execute the **index.html**.

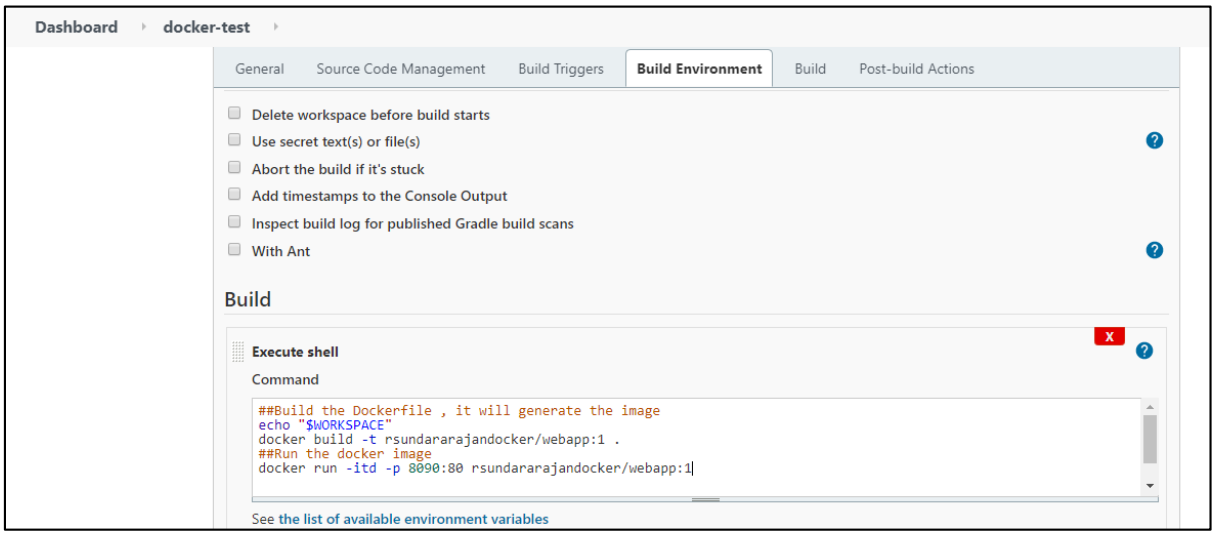
iv. CMD: Execute the command apachectl –D FOREGROUND: This means that systemd is capable of automatically restarting Apache if it does crash.



g. **Index.html** file content:



h. Build the docker file: It will generate image on out local machine > run the image > generate the container and run it on 8080 0r 8090 port.



i. Save the Jenkins job configuration.
Note: Make sure that there is no docker container running on 8080 or 8090 port: Better case, remove all existing images and containers.

```
[root@instance-jenkins ~]# docker ps
CONTAINER ID        IMAGE                          COMMAND                  CREATED             STATUS
 PORTS                    NAMES
a5d2f6eeb0a1        rsundararajandocker/webapp:1   "apachectl -D FORE..."   18 seconds ago      Up 17 seconds
 0.0.0.0:8090->80/tcp   affectionate_kowalevski
[root@instance-jenkins ~]#
```

   i. Step 1/4: FROM ubuntu
   ii. Step 2/4: RUN apt-get update && apt-get install apache2 -y && service apache2 restart
   iii. Step 3/4: ADD index.html /var/www/html/
   iv. Step 4/4: CMD apachectl -D FOREGROUND

j. Run the application.

k. Perform another exercise: Add a new html file with the name **sample.html**.
Output: **Index.html** is the default file which will be detected by apache server. http://52.14.5.172 and http://52.14.5.172/index.html both will run the default html file i.e., index.html.

I. To run the sample html file, and specify the name: http://52.14.5.172/sample.html

1.4 Troubleshooting

S. No	Problem	Solution

1	Received an error message stating **docker group not found then follow the steps to create group**.	Execute the following commands:
		`sudo groupadd docker`
		`sudo usermod -aG docker $USER`
		`newgrp docker`
		
		`sudo usermod -aG docker jenkins`
		`sudo systemctl restart jenkins`