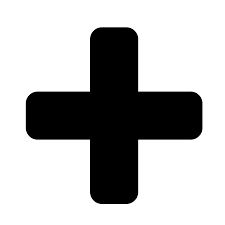
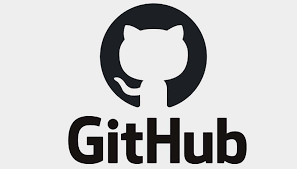
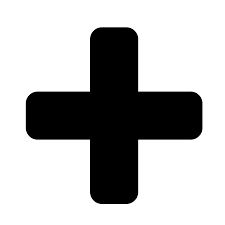
**Starting Jenkins Service using Docker Container and then testing our application**











**Task to do:**

1. Create container image that has Jenkins installed in it using dockerfile

2. When we launch this image, it should automatically start Jenkins service in the container.

3. Create a job chain of job1, job2, job3 and job4 using build pipeline plugin in Jenkins

4. Job1 : Pull the Github repo automatically when some developers push repo to Github.

5. Job2 : By looking at the code or program file, Jenkins should automatically start the respective language interpreter install image container to deploy code ( eg. If code is of PHP, then Jenkins should start the container that has PHP already installed ).

6. Job3 : Test your app if it is working or not.

7. Job4 : if app is not working , then send email to developer with error messages.

8. Create one extra job job5 for monitor. If container where app is running fails due to any reason then this job should automatically start the container again.

**Execution of Task:**

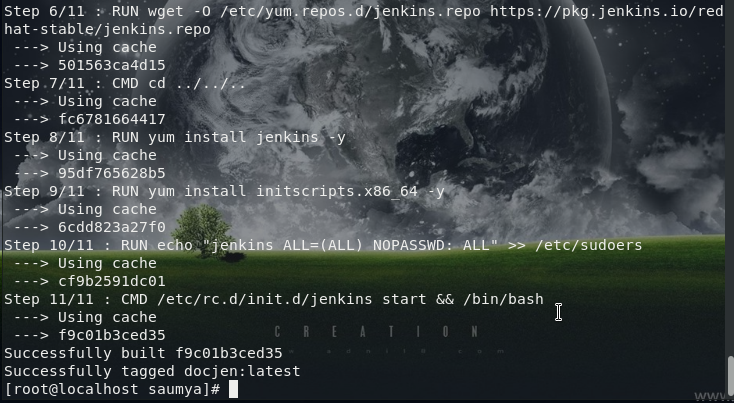
1. Firstly we will create a Dockerfile which will help us to start Jenkins service automatically when we will launch container through this dockerfile.
2. Then we will create a GitHub repo which will contain some files to test.
3. Then we will create respective containers for different programming language codes (In our case for simplicity we had made only a single container ‘webcontainer’ in which it will launch httpd services)
4. Then we will create Jenkins jobs for testing the GitHub repo files and sending mail to the developer if our files are not working properly.
5. We will also create an extra job to check that whether our container is running properly or not. If by chance our container gets switched off then we will again start the container.

**Dockerfile to start Jenkins service automatically:**



We have build the Dockerfile using the following command

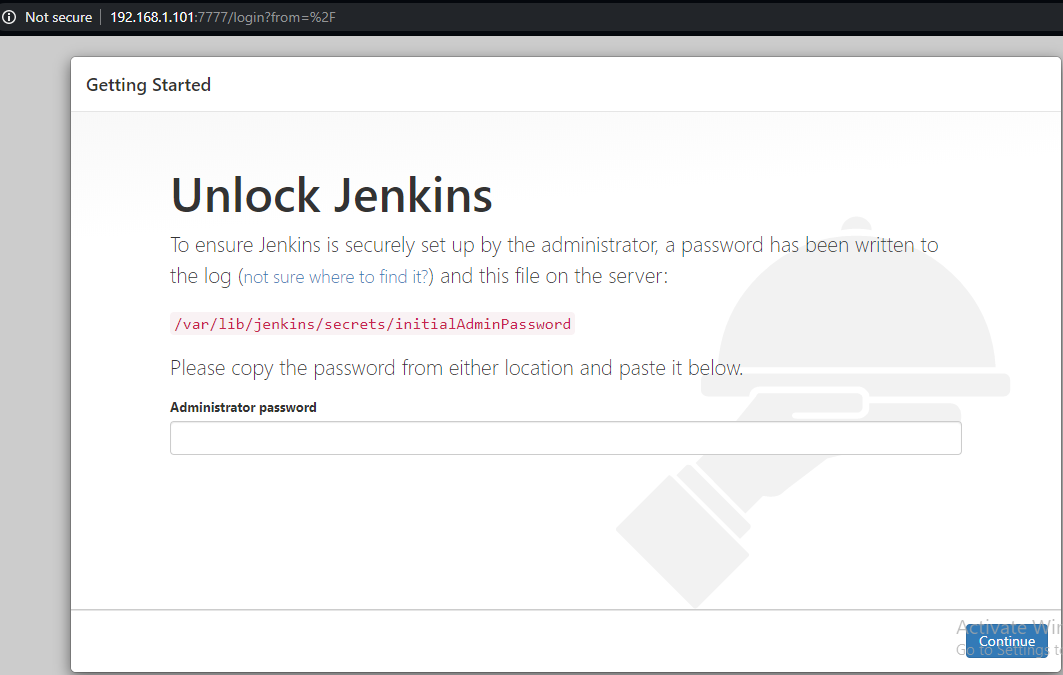
**docker build –t jencontainer ws1**



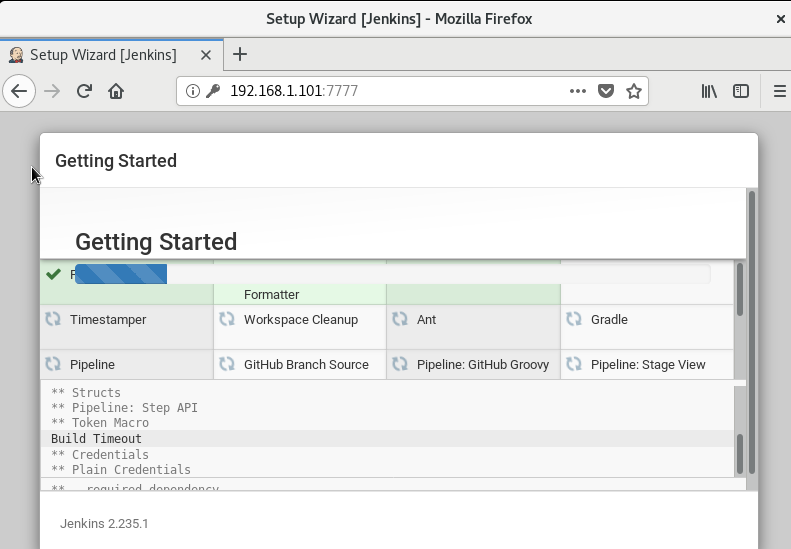
Then, we will use following command to launch the container for automatically starting Jenkins on port no. 7777

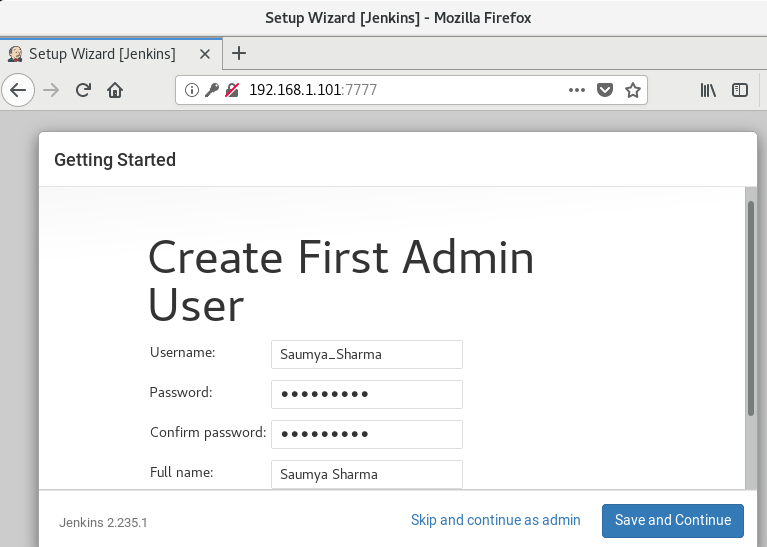


When we will start Jenkins in our browser on port 7777 then it will ask for password to us which we are required to provide it. So then we will provide it initial Admin Password.

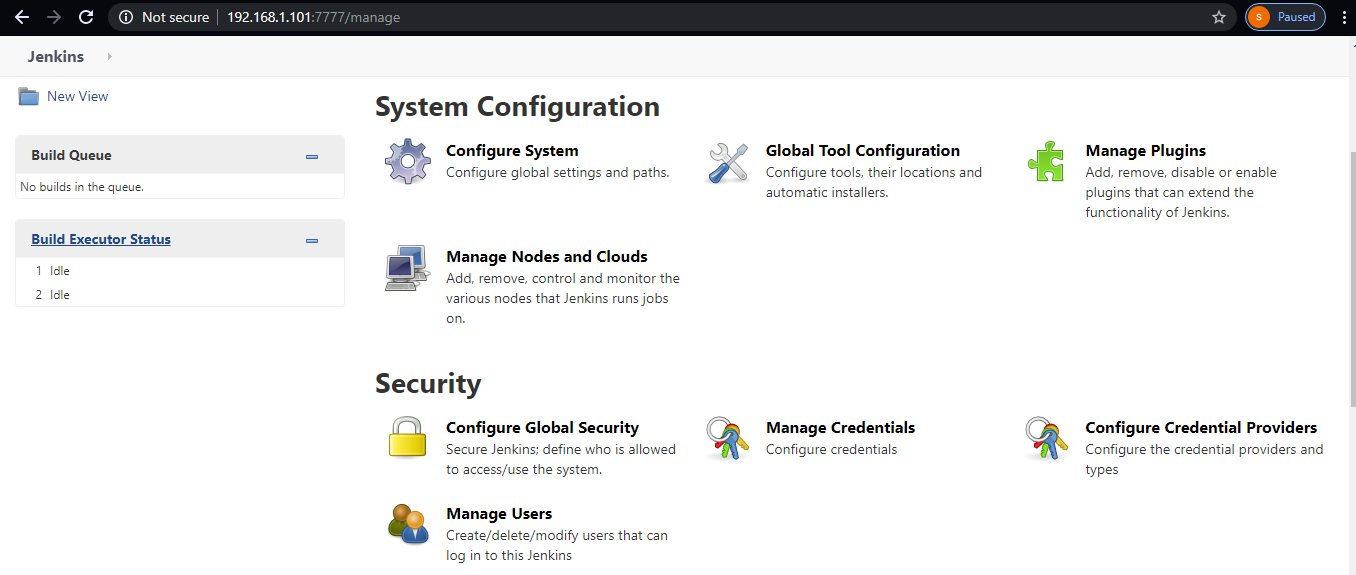


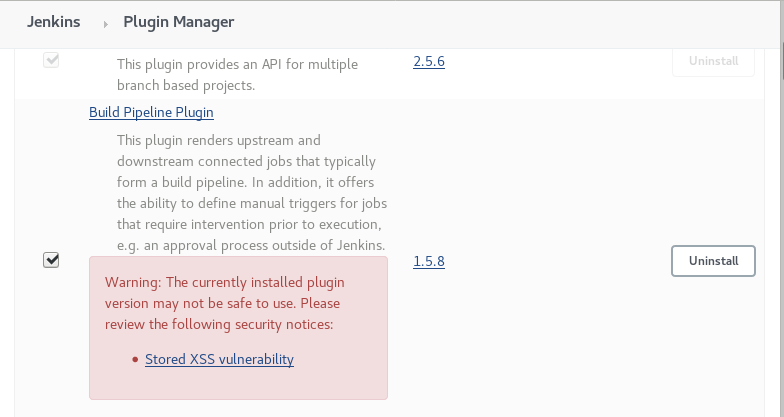
Then we will install the required plugins suggested by Jenkins in beginning and we will also setup our Jenkins account

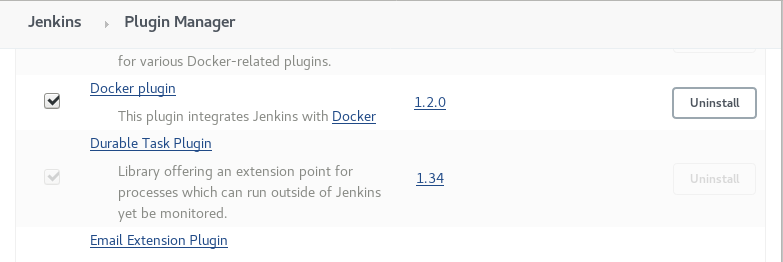


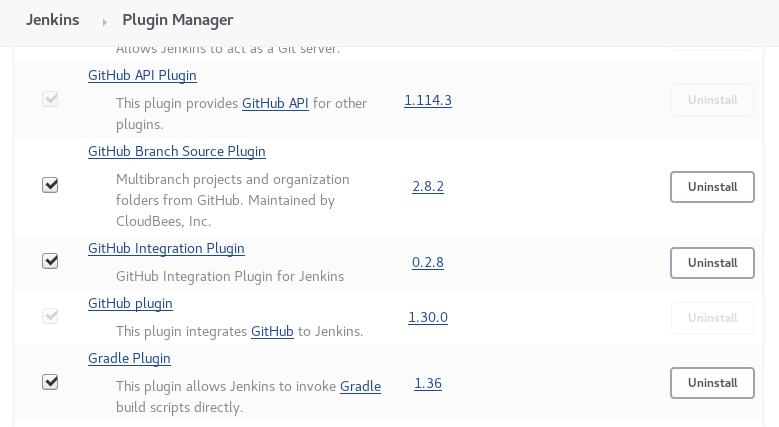


Then in system configuration we will go to manage plugins and then we will install required plugins for our project like build pipeline plugin, docker plugin, GitHub plugin, etc.

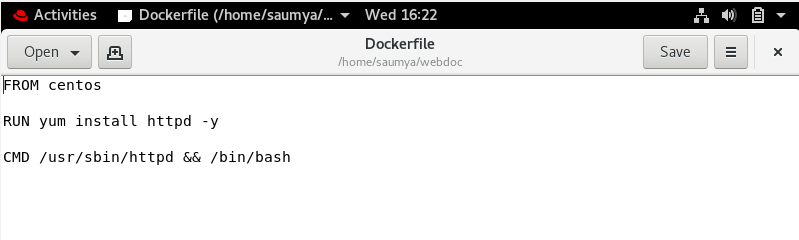






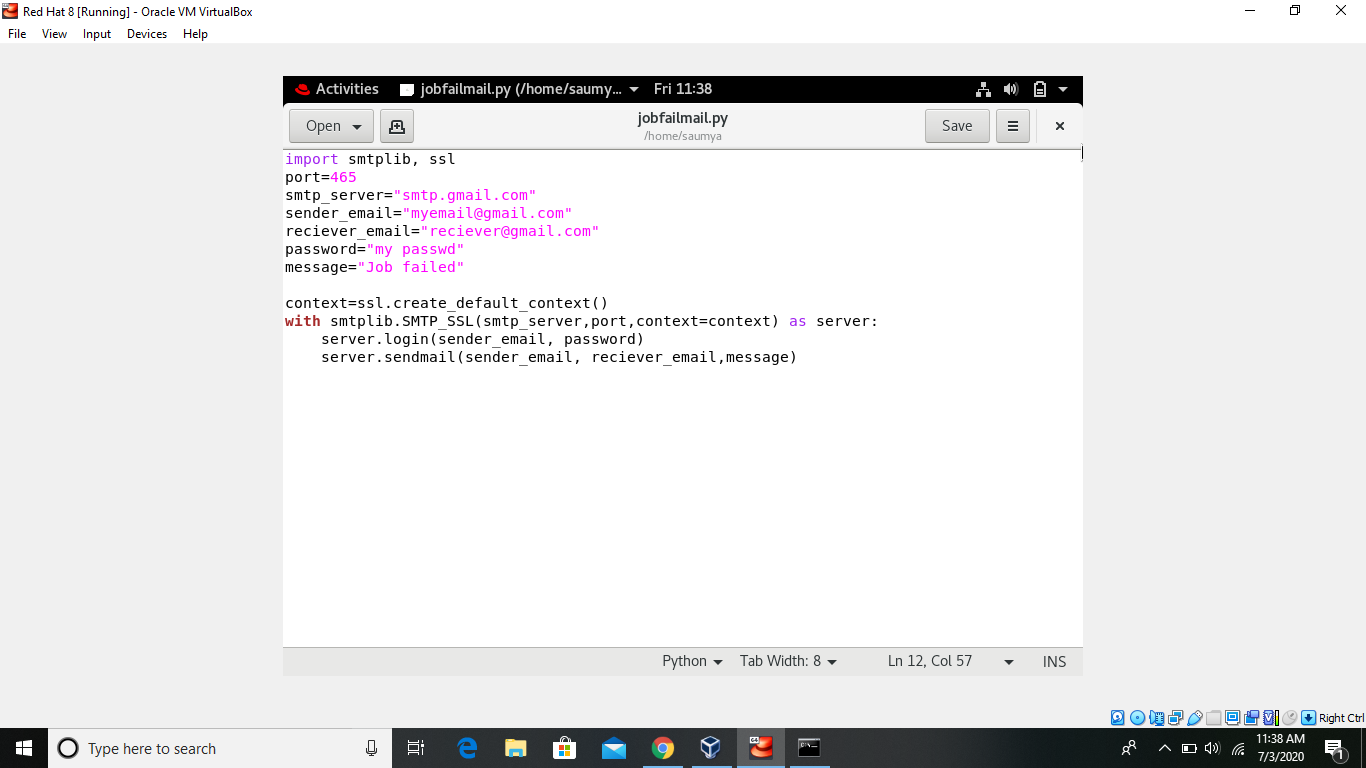


**Dockerfile for running httpd:**



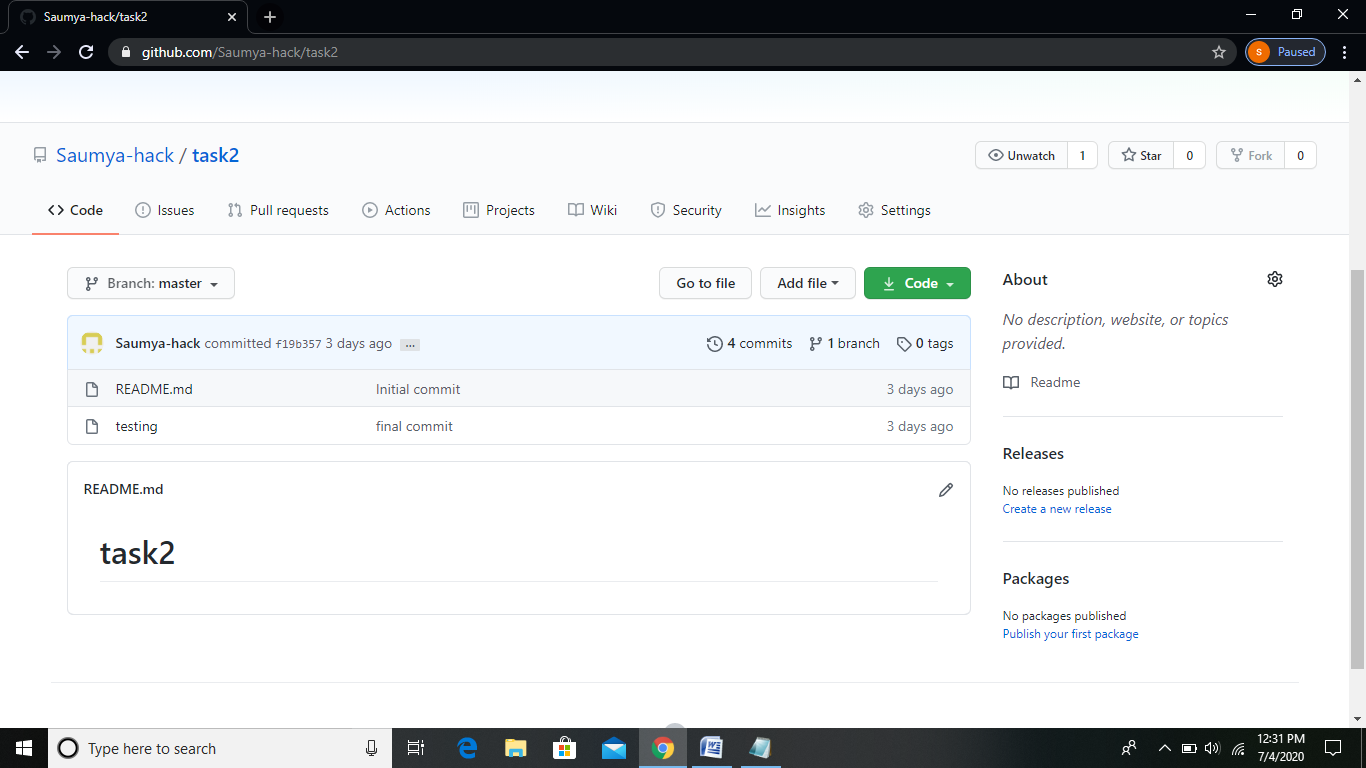
**Python file to mail:**

This is a python file which we had used in our jenkins job to mail the user in case when the testing of files gives us wrong result

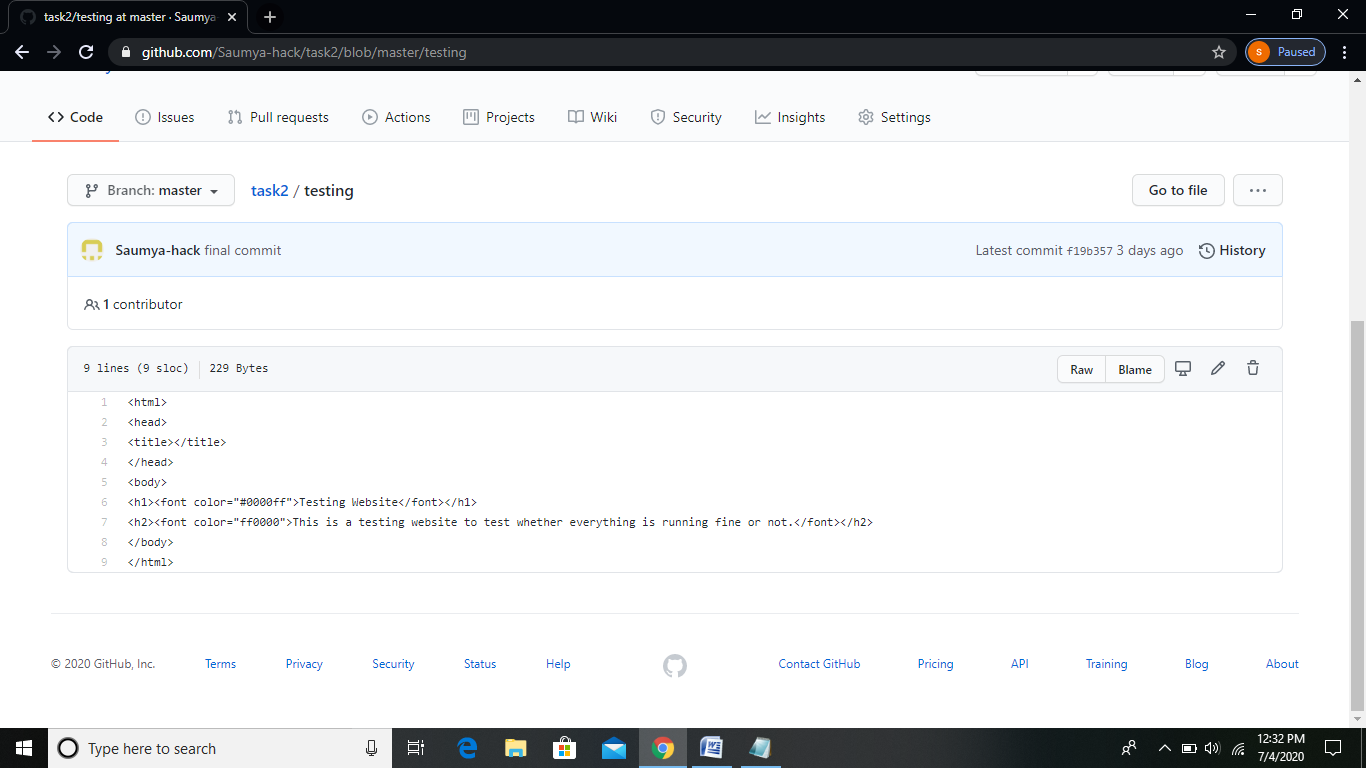


**GitHub repo:**

This is GitHub repo named ‘task2’. We have created a testing file here.



This is the ‘testing’ file content.

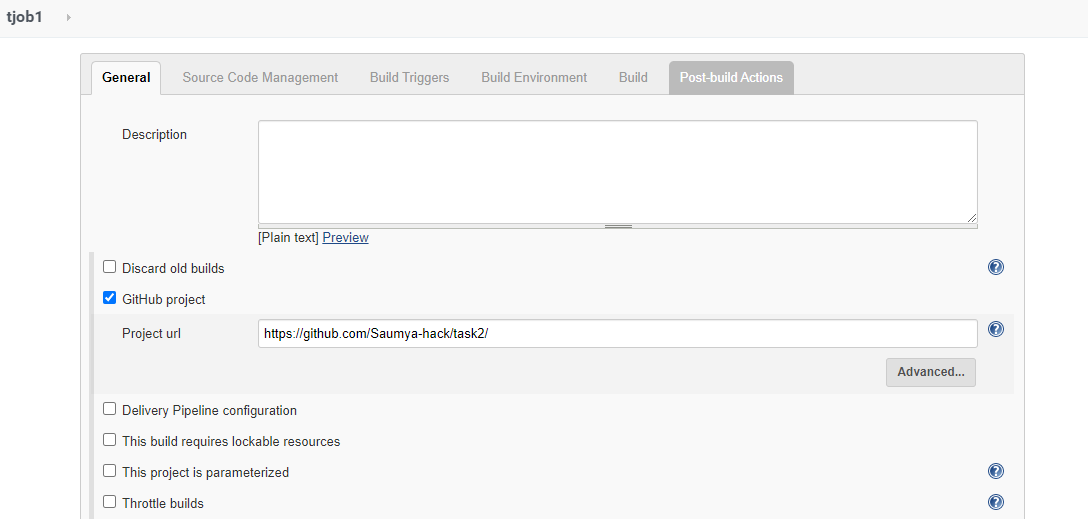


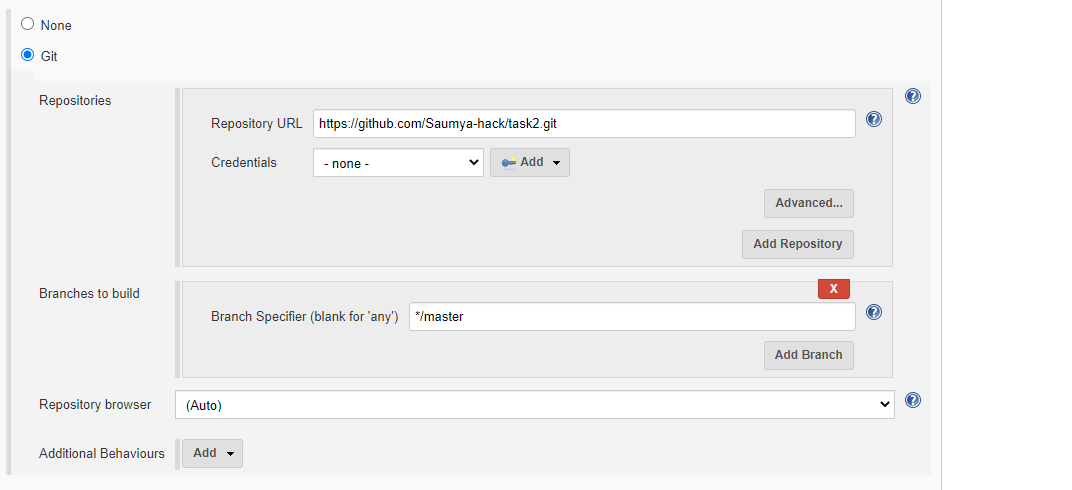
**Jenkins Jobs:**

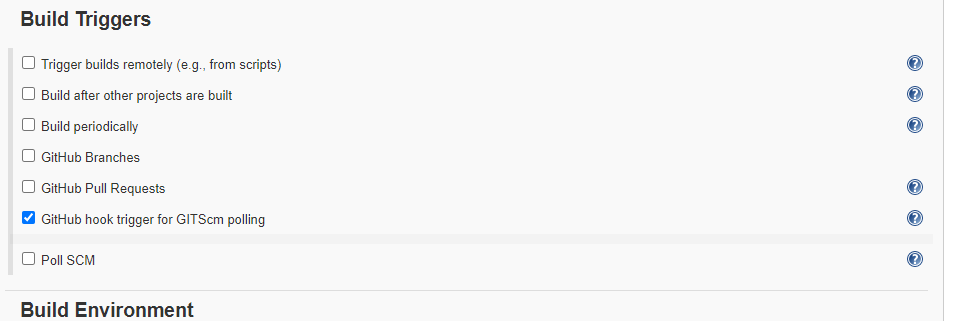
So now we have done the initial setup required and now we will create and run our Jenkins job.

JOB1 (tjob1):

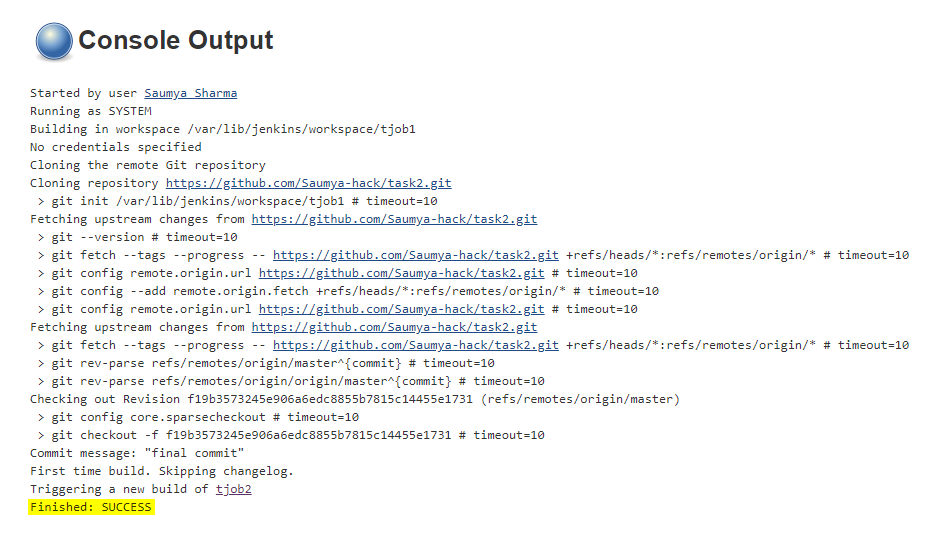
This job will simply trigger the GitHub repository (task2) from GitHub.





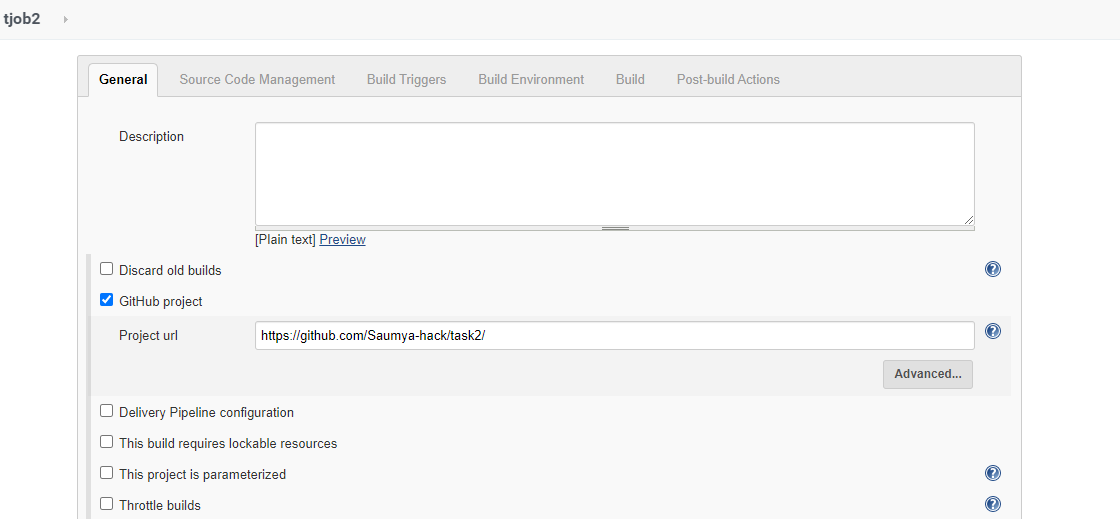


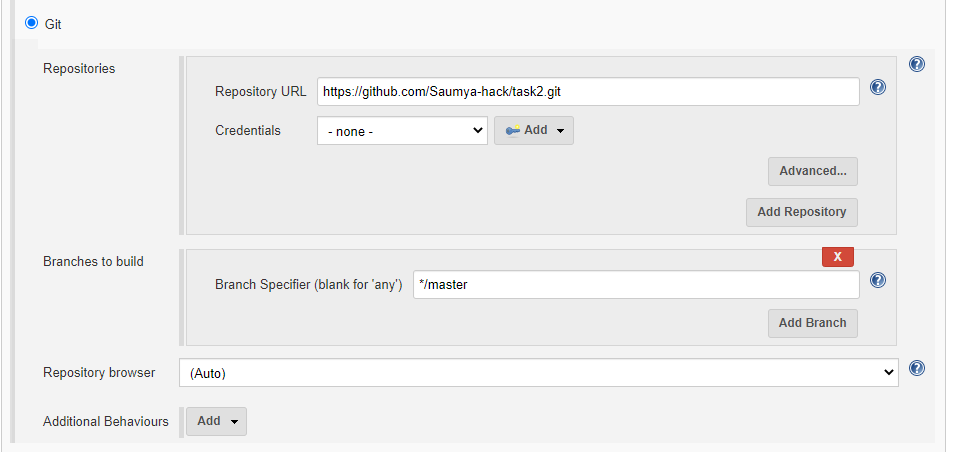
Console output shows that our GitHub repository is triggered successfully.

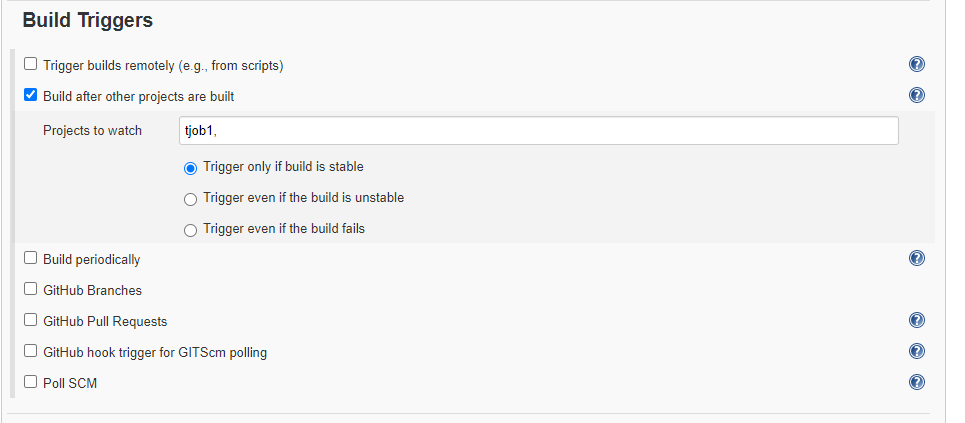


JOB2 (tjob2):

This job will check the files in our triggered GitHub repository that whether the file is of HTML or any other language. If the file is of HTML then it will simply launch a container of httpd service image which we had made earlier else it will launch a simple container of centos image. We can improve this job by making other container images one installed with java, one installed with python and then can launch those container if our repo contains code of those languages. But for simplicity we have just considered HTML for now.



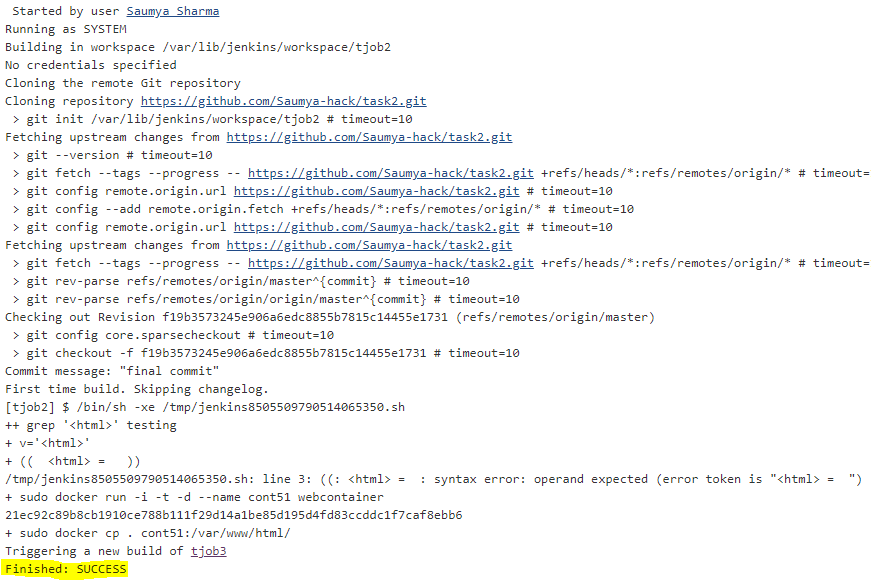


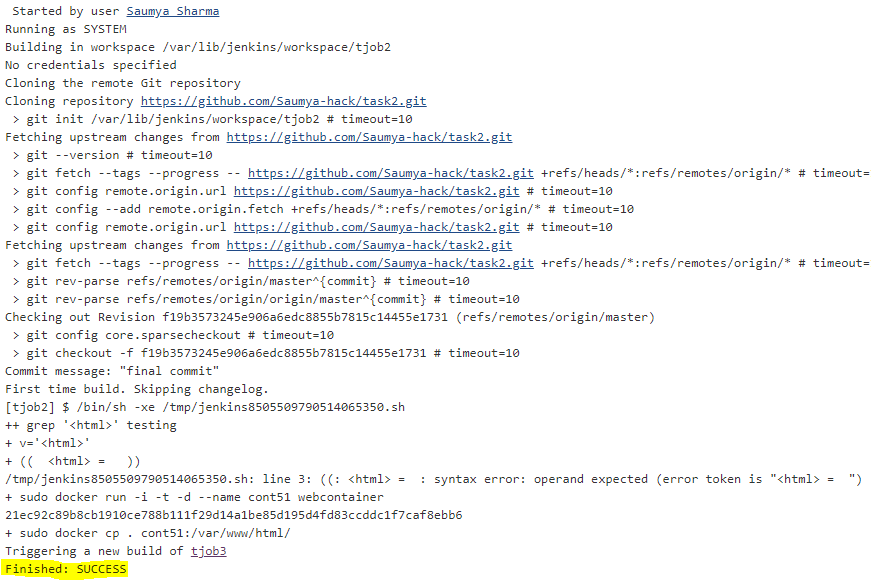


This following shell script will check that whether the file in repo is of HTML or not. If the file is of HTML then webcontainer will be launched and httpd service will be started automatically.



This console output shows that our job is executed successfully



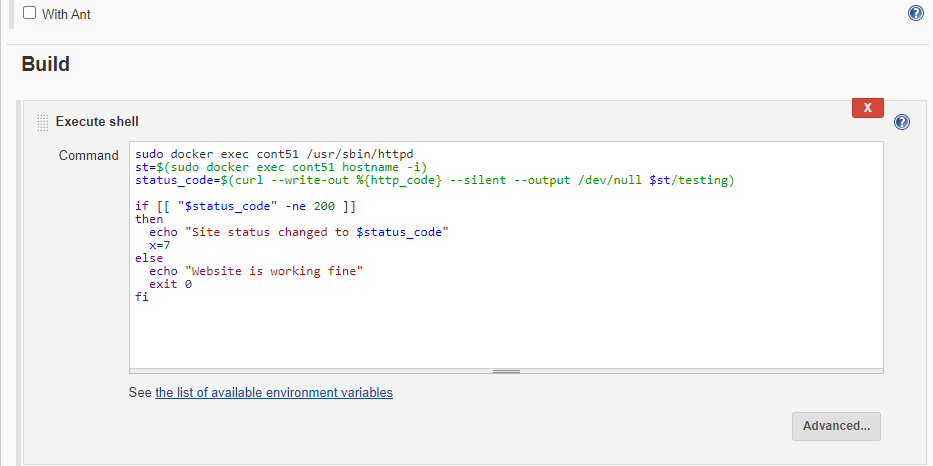


JOB3 (tjob3):

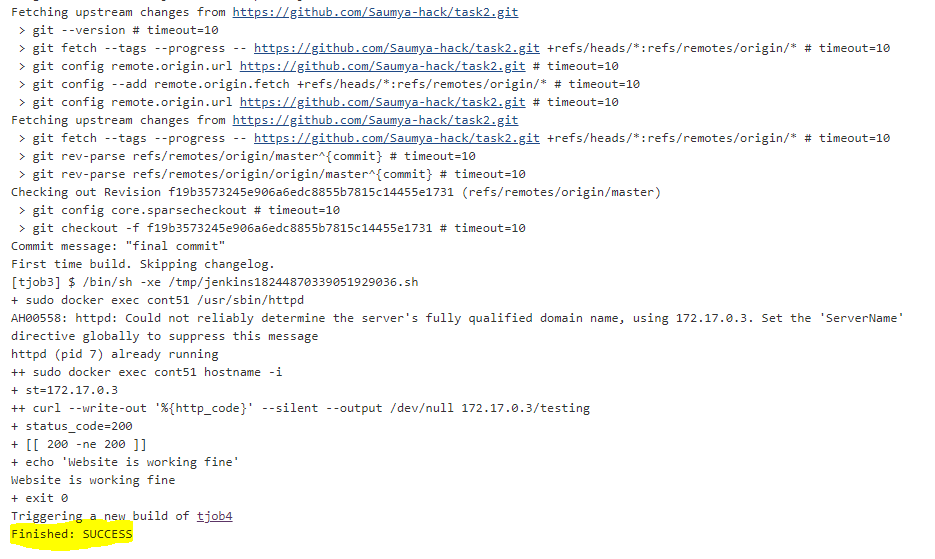
This job will check that whether our website is working properly or not. If the website will be working properly then status code 200 will be returned else any other status code will be returned.



Here, in this script we are checking the status code returned by webpage. If it will be working fine then console output will show “Website is working fine” and the job will be executed successfully, else a variable ‘x’ will be assigned value ‘7’. This ‘x’ is just working as a flag variable. We will use it in next job.

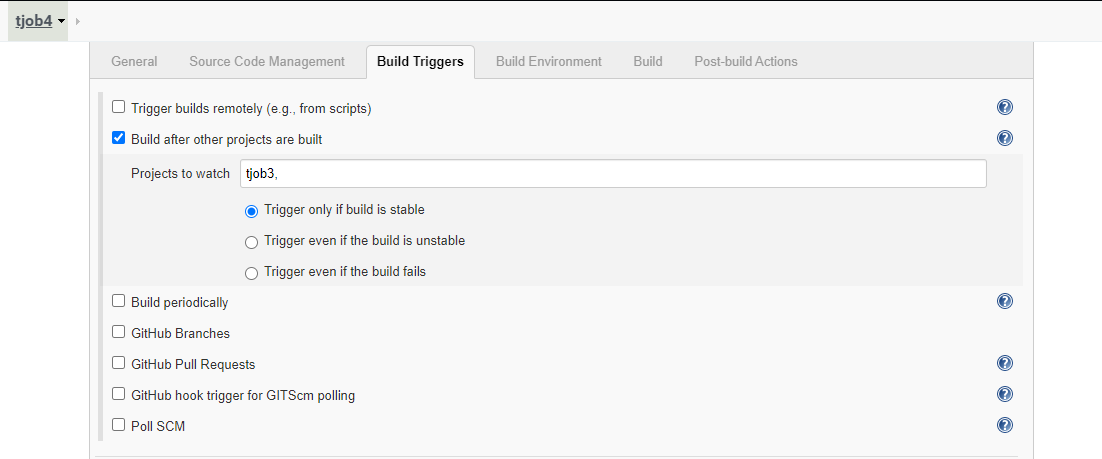


Here, our job is finished successfully.



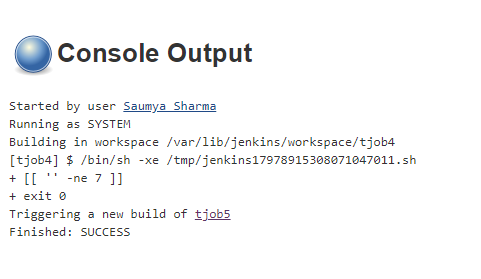
JOB4 (tjob4):

This job will check the value of ‘x’ variable which we had made in last job. If the value of x is 7 it means that our webpage was not working properly and if it has no value means that our webpages were working properly. So if value of ‘x’ will be equal to ‘7’ then we will exit our Job with status ‘1’ and a mail will be sent to the developer otherwise the job will exit with ‘0’ status.



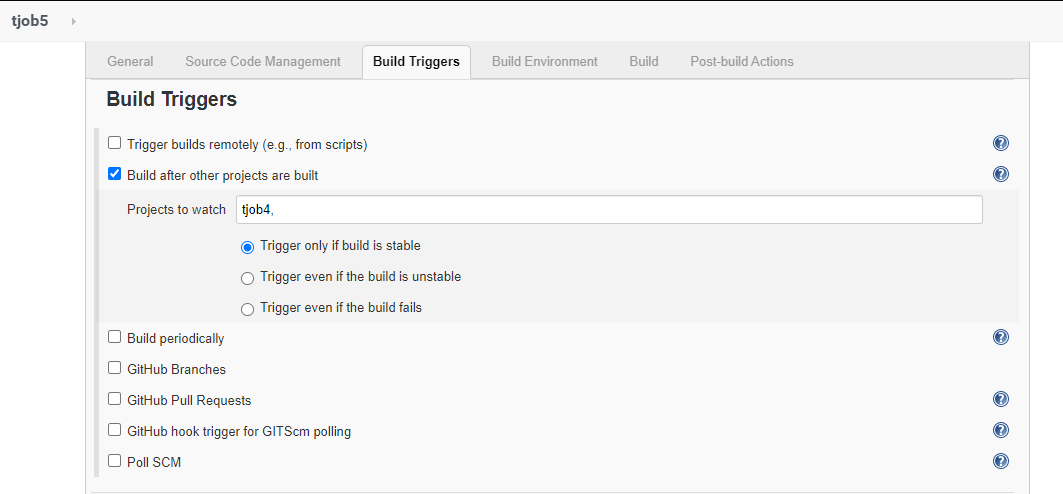


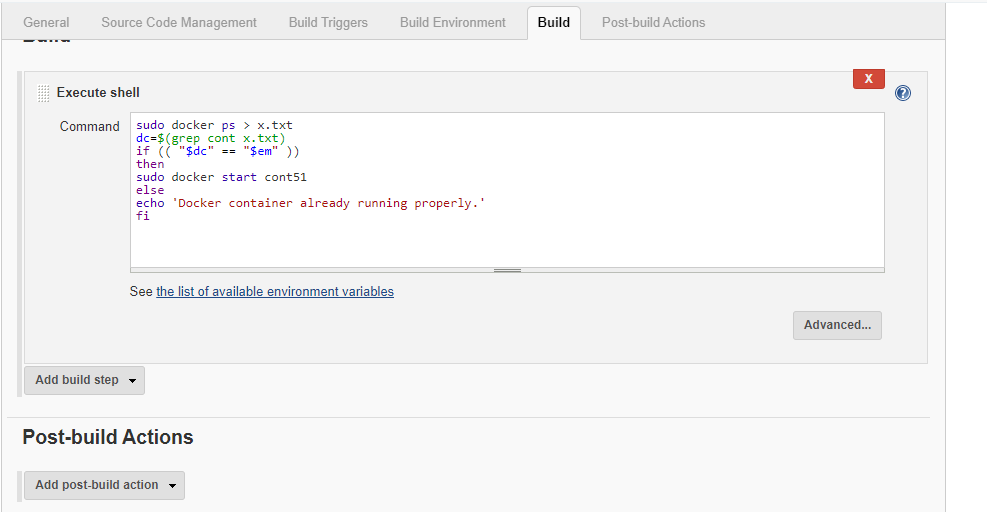
Console output shows that our Job was successfully executed and our webpages are working fine.



JOB5 (tjob5):

This job will check that whether our container is working properly or not. If it is not working properly then the container will be launched again else Console output will display that ‘Docker container already running properly’.



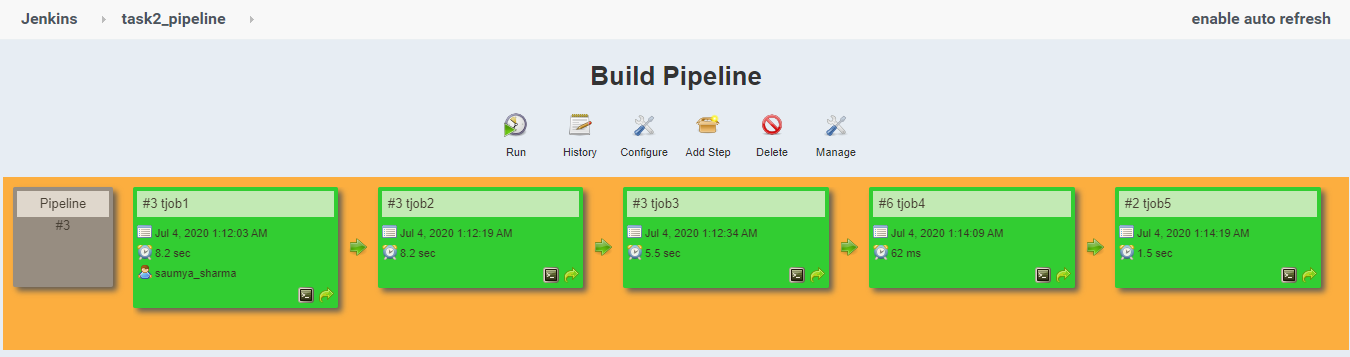


Here our Job is executed successfully and it is being displaying on Console output that our Docker Container is already running properly.

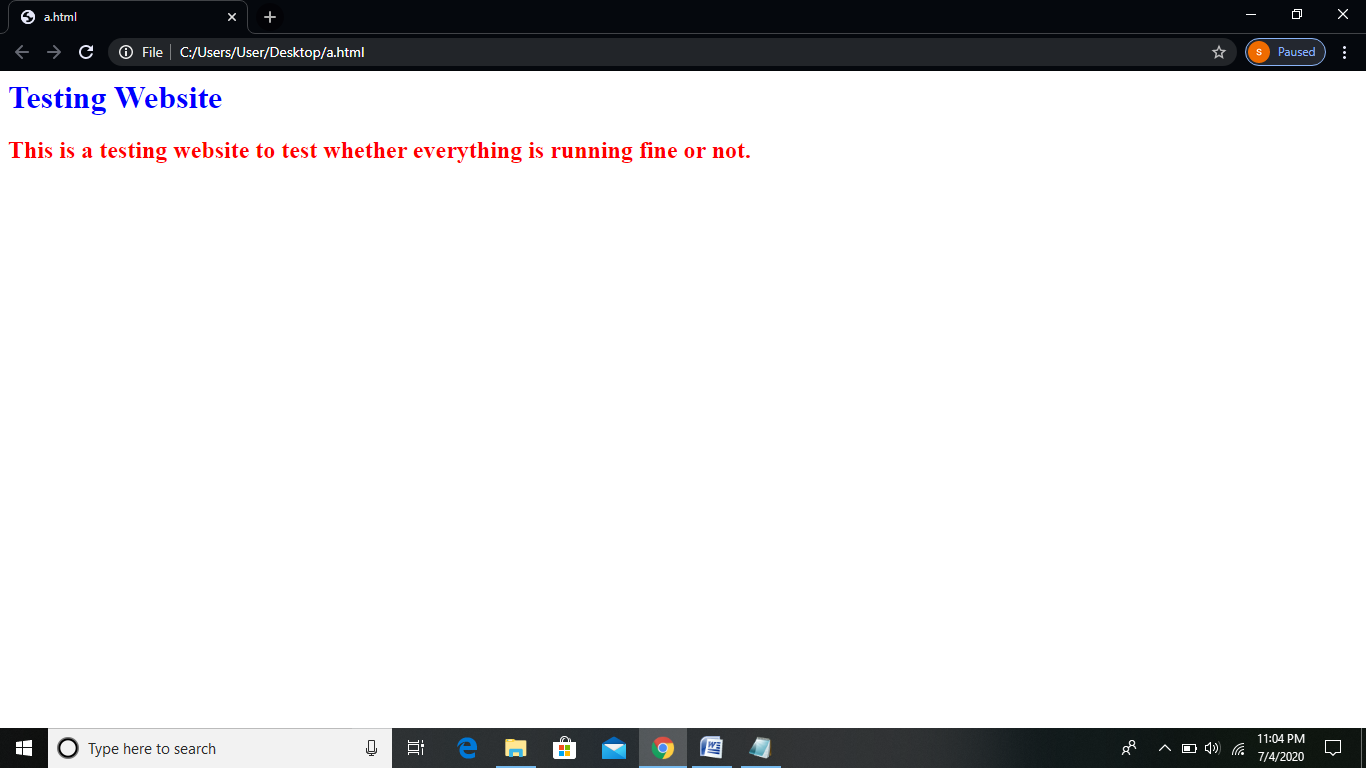


**Build Pipeline:**

Following is the build pipeline which is showing that all our Jobs are running properly.



Following image shows our website page. So we can see that our webpage in the GitHub repo was working properly. That’s why our all jobs are executed successfully which is shown in above image by build pipeline.



I learnt these things in my MLOps training organized by Linux World under the mentorship of Mr. Vimal Daga Sir.

Thanks for reading.