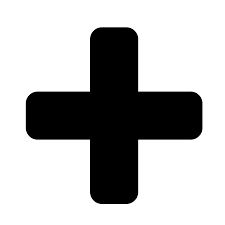
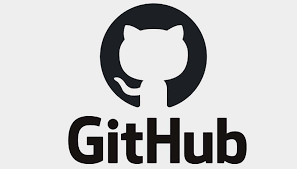
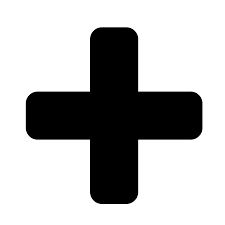
**Testing the Developer branch and merging it to Master branch using Jenkins**











**Task Summary:**

**JOB#1**

If Developer push to dev branch then Jenkins will fetch from dev and deploy on dev-docker environment.

**JOB#2**

If Developer push to master branch then Jenkins will fetch from master and deploy on master-docke environment. Both dev-docker and master-docker environment are on different docker containers.

**JOB#3**

Jenkins will check (test) for the website running in dev-docker environment. If it is running fine then Jenkins will merge the dev branch to master branch

**Task flow summary:**

We have created two branched in GitHub named as ‘master’ branch and ‘dev’ branch. Then we have created four Jenkins jobs JOB1(ojob1), JOB2(ojob2), JOB3(ojob), JOB4(ojob3) .

**JOB1(ojob1):**

To launch a container(devdocker) when the developer pushes anything to ‘dev’ branch. This job will run every hour.

**JOB2(ojob2):**

To launch a container(masterdocker) when the developer pushes anything to ‘master’ branch. This job will run every time when JOB1 will be executed successfully.

**JOB3(ojob):**

This job will test the webpages in the ‘dev’ branch. If they are working properly then we will go for merging the ‘dev’ and ‘master’ branch. This job will run every time when JOB2 will be executed successfully.

**JOB4(ojob3):**

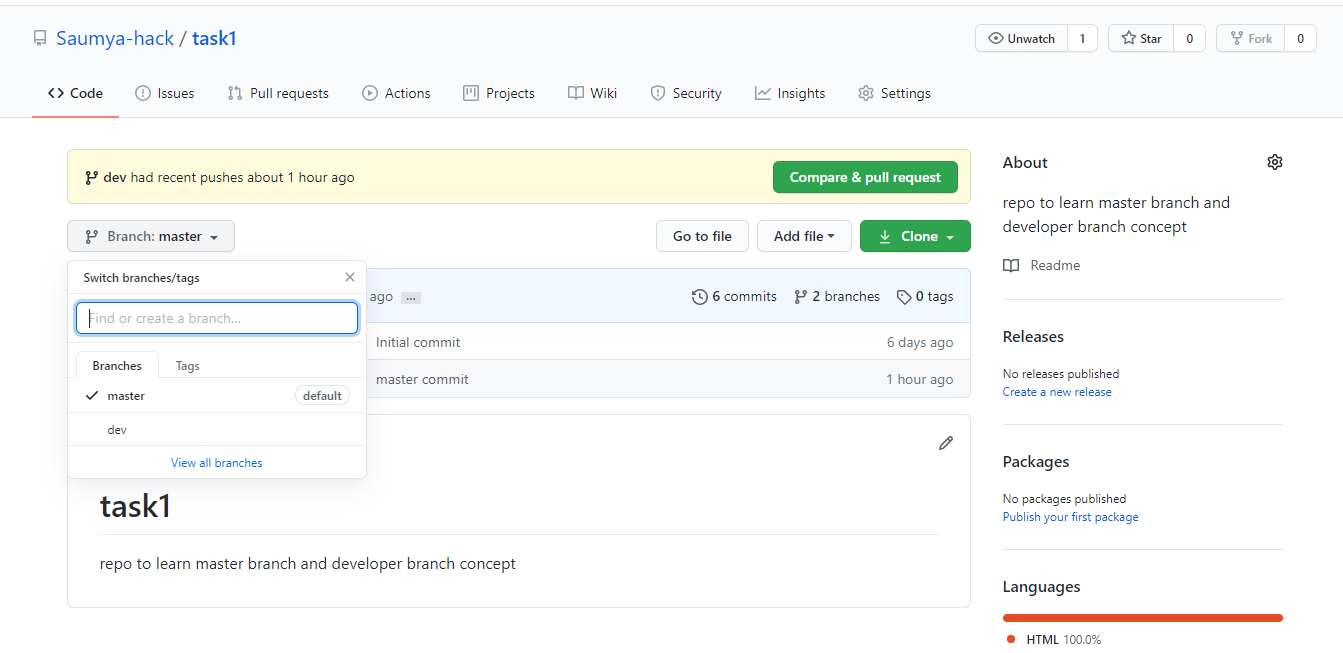
This is the final job of our pipeline. This job will merge the ’dev’ and ‘master’ branch. Also it will merge the data of ‘devdocker’ container to the ‘masterdocker’ container. As in JOB3 we come to know that our webpages are working correctly or not so according to that the data of the ‘devdocker’ container will be copied to ‘masterdocker’ container. This job will run every time when JOB2 will be executed successfully.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

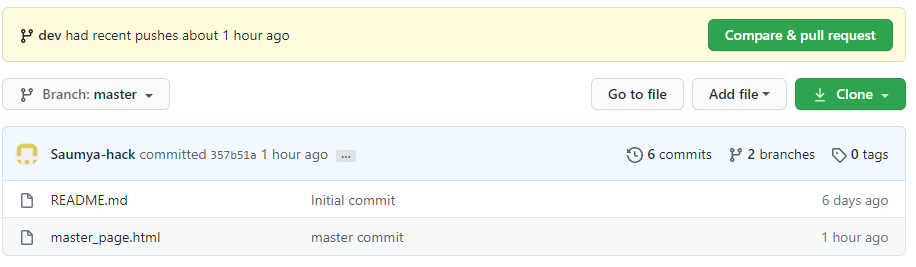
**GitHub Repository:**

Firstly, we have created a repository in GitHub named as ‘task1’.

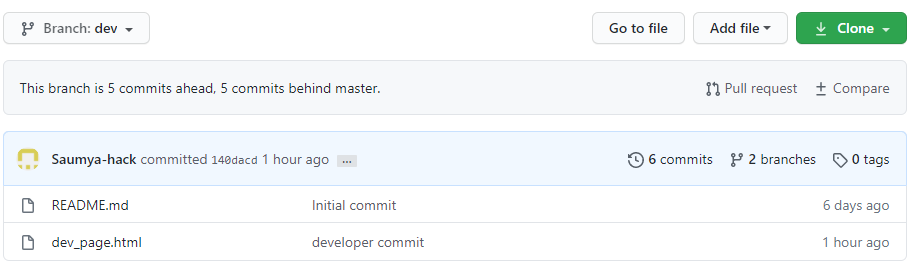
We have created two branches in this repository. First one is ‘master’ branch and another is ‘dev’ branch.



When we are at master branch then we have two files ‘README.md’ and ‘master\_page.html’



When we are at dev branch then we have two files ‘README.md’ and ‘dev\_page.html’



We can also see the branches in our task1 repository and the files committed by these branches by using Git as shown here.

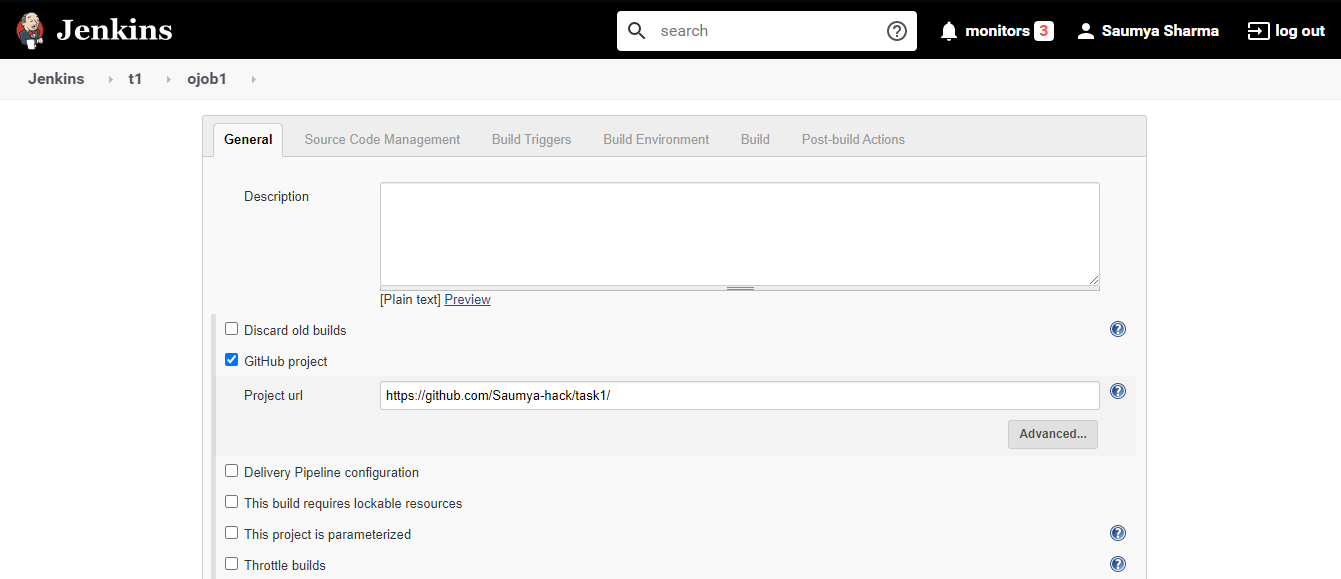


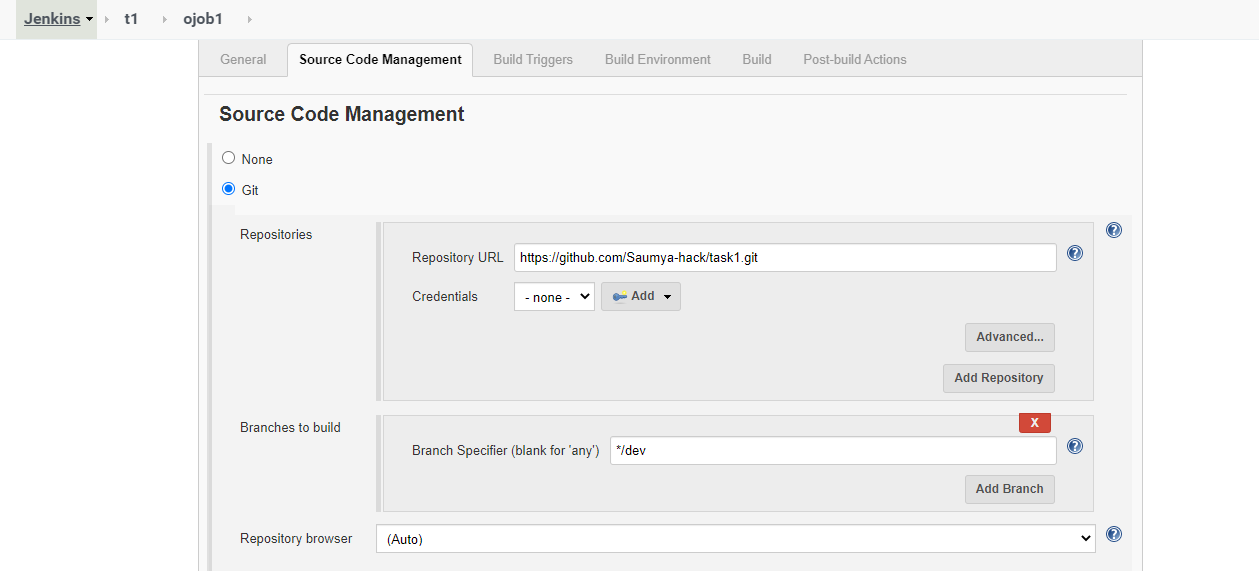
**JENKINS JOBS:**

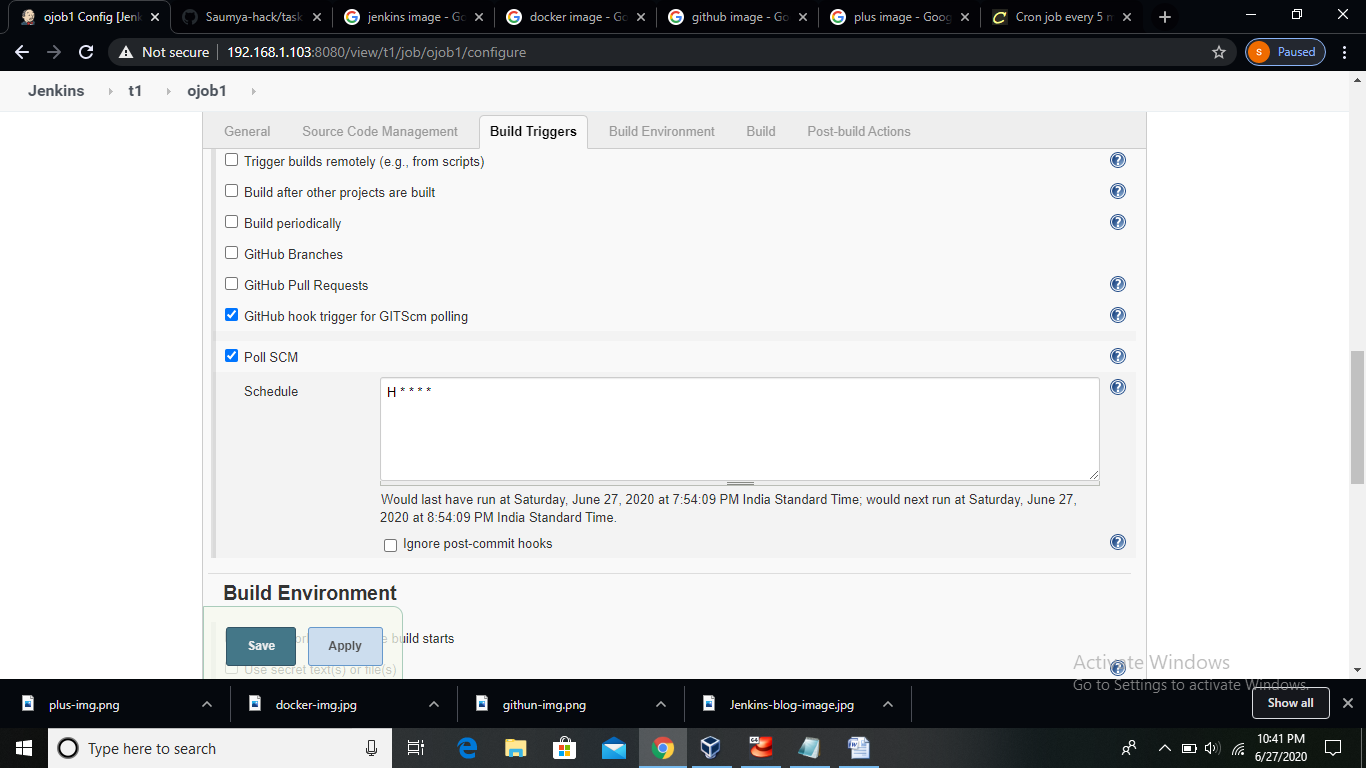
**JENKINS JOB 1(ojob1):**

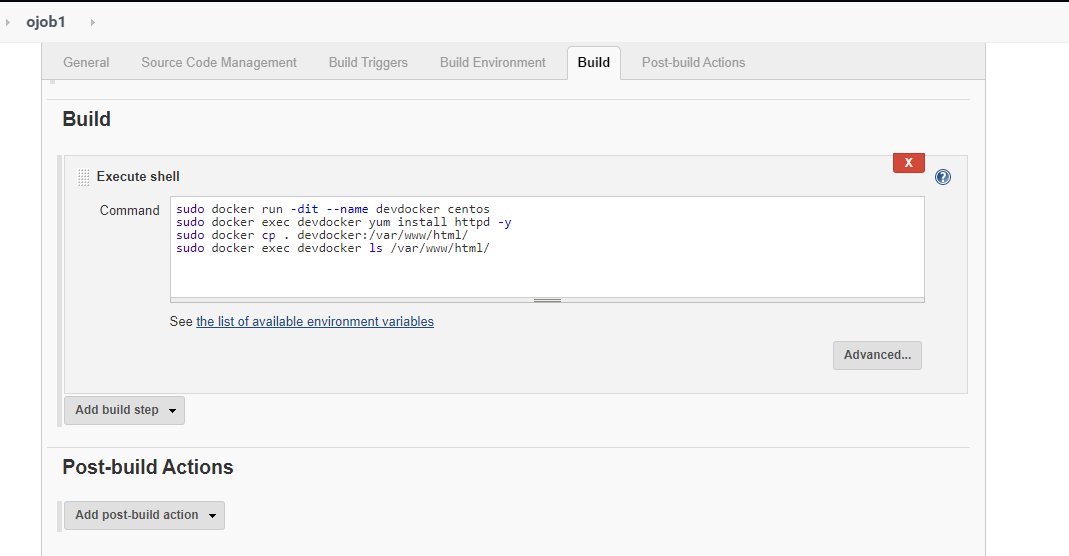
We have given name to JOB1 as ‘ojob1’ in jenkins.

In this job firstly we have provided the url of our GitHub repository to our job and we have selected ‘dev’ branch to trigger in this job.









**sudo docker run -dit --name devdocker centos**

Here we have created a docker container named as ‘devdocker’ using centos image.

**sudo docker exec devdocker yum install httpd -y**

Here we have installed ‘httpd’ in our ‘devdocker’ container so that we can run webpages in our container.

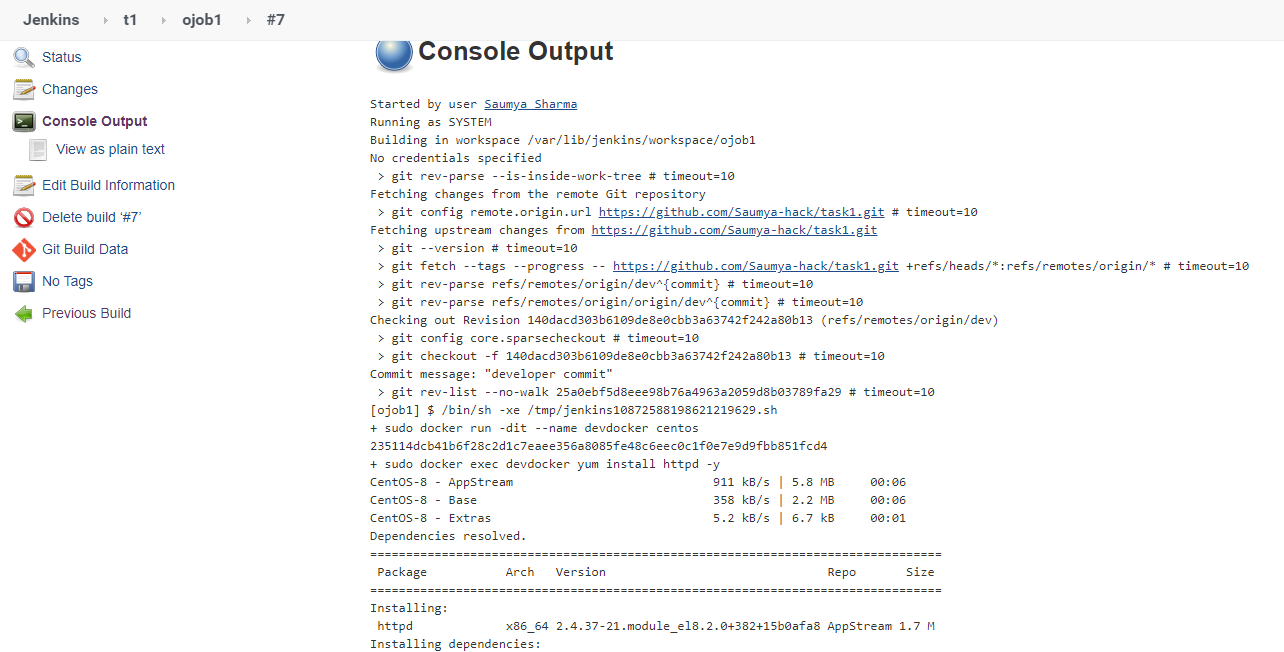
**sudo docker cp . devdocker:/var/www/html/**

Since, we are required to store our webpages in /var/www/html/ directory, so we have copied our ‘dev’ branch data to this directory in ‘devdocker’ container.

**sudo docker exec devdocker ls /var/www/html/**

This is just to check on console that whether our job copied the required data to /var/www/html/ successfully or not.

Console Output of JOB1:



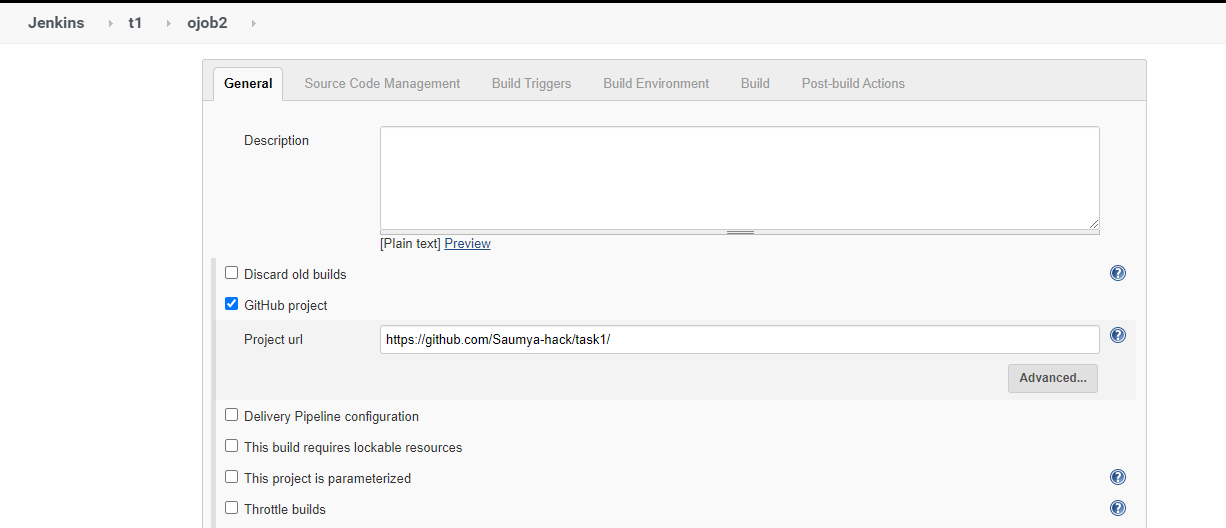


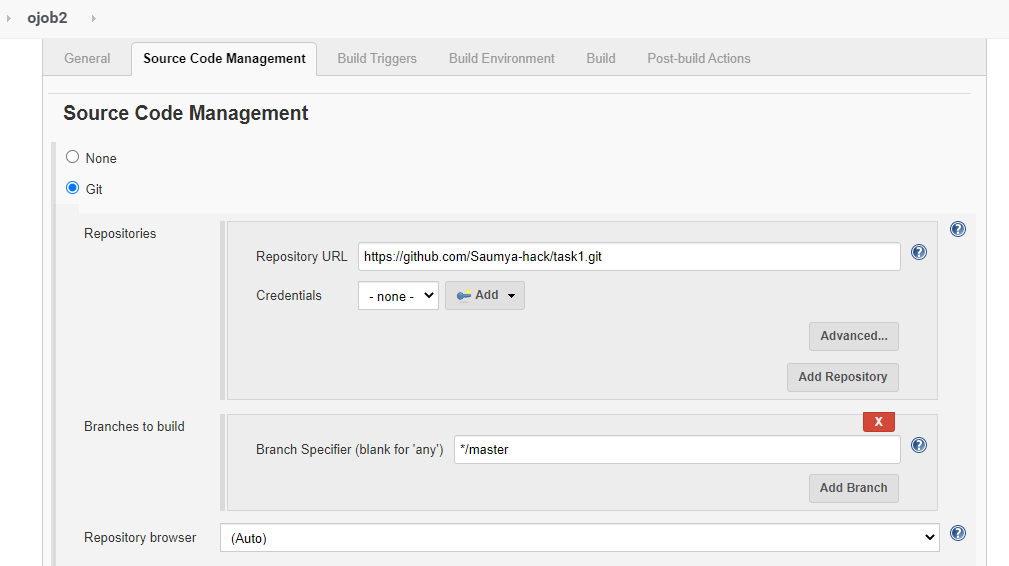
We can see that when we had run this ‘ojob1’ then it has copied the files present in the ‘dev’ branch to ‘devdocker’ container successfully.

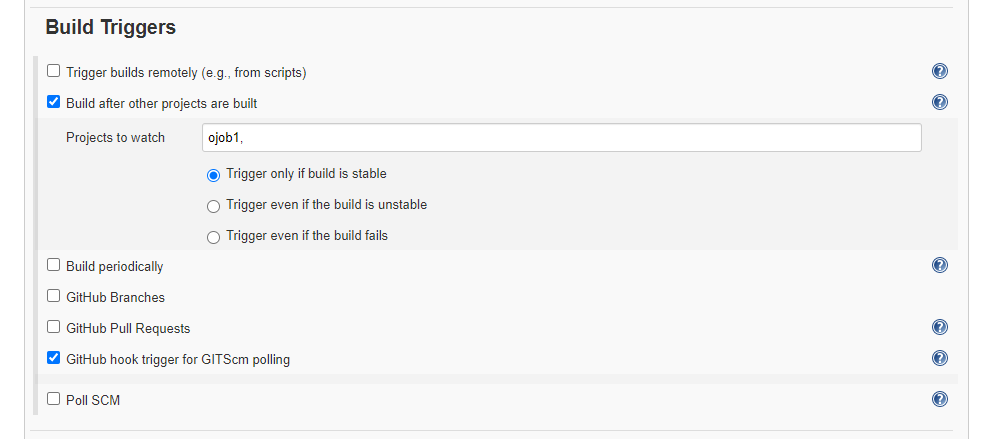
**JENKINS JOB 2(ojob2):**

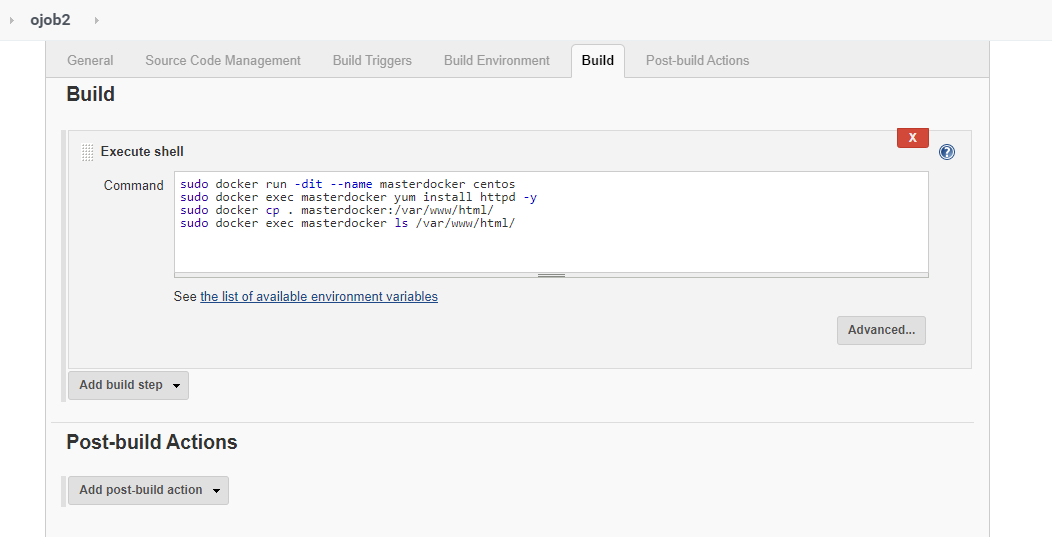
We have given name to JOB2 as ‘ojob2’ in jenkins.

In this job firstly we have provided the url of our GitHub repository to our job and we have selected ‘master’ branch to trigger in this job.









**sudo docker run -dit --name masterdocker centos**

Here we have created a docker container named as ‘masterdocker’ using centos image.

**sudo docker exec masterdocker yum install httpd –y**

Here we have installed ‘httpd’ in our ‘masterdocker’ container so that we can run webpages in our container.

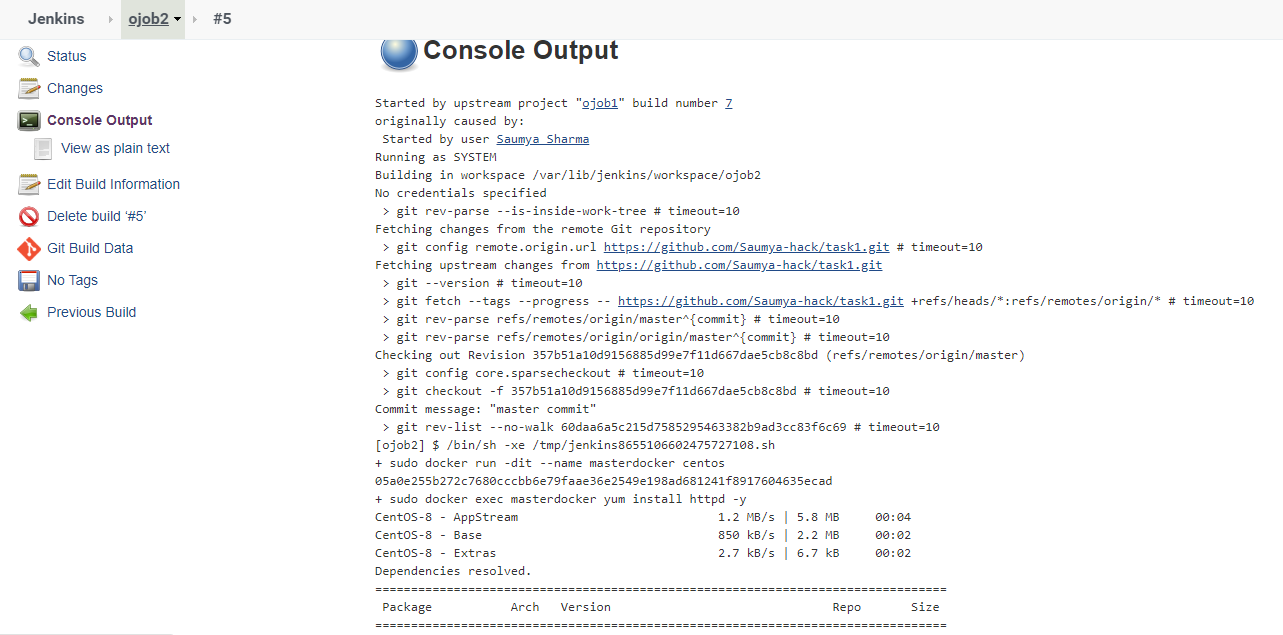
**sudo docker cp . masterdocker:/var/www/html/**

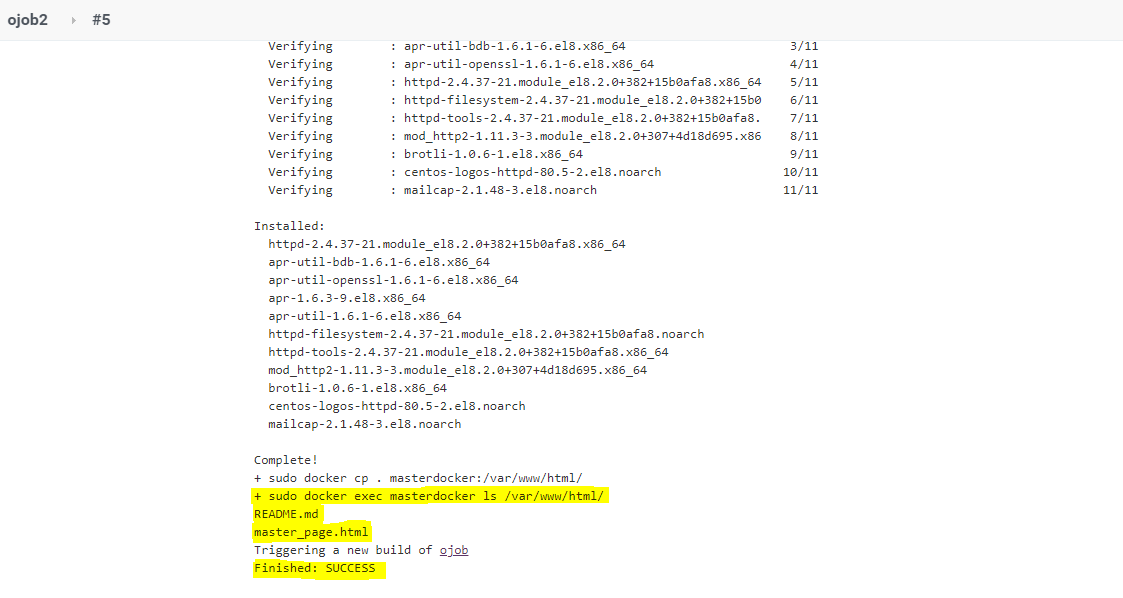
Since, we are required to store our webpages in /var/www/html/ directory , so we have copied our ‘master’ branch data to this directory in ‘masterdocker’ container.

**sudo docker exec masterdocker ls /var/www/html/**

This is just to check on console that whether our job copied the required data to /var/www/html/ successfully or not.

Console Output of JOB2:



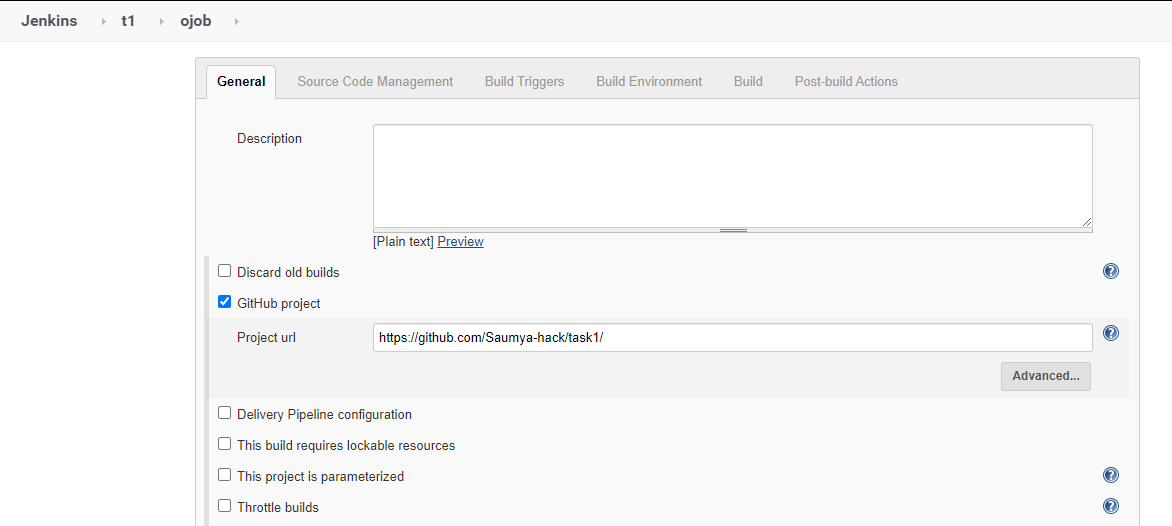


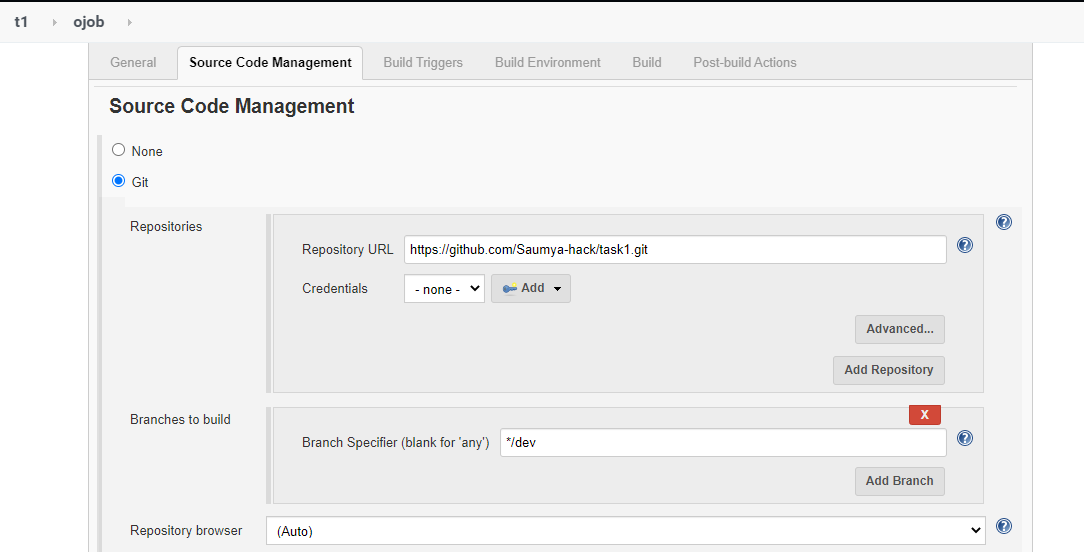
We can see that when we had run this ‘ojob2’ then it has copied the files present in the ‘master’ branch to ‘masterdocker’ container successfully.

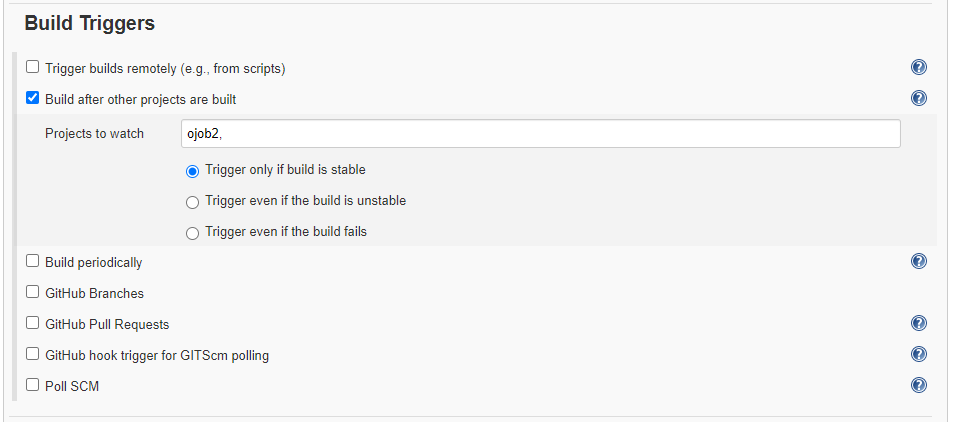
**JENKINS JOB 3(ojob):**

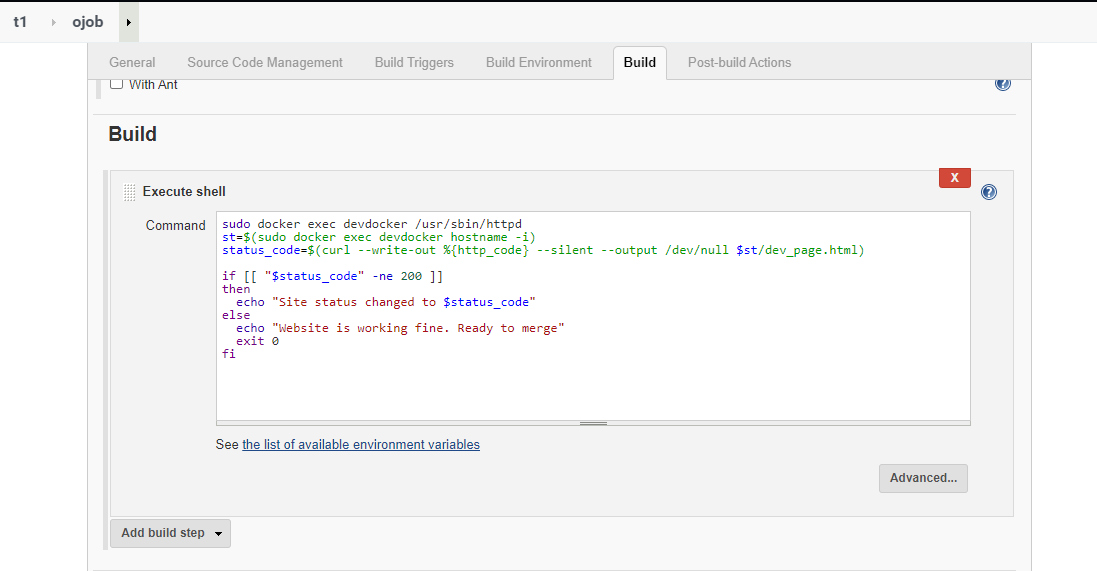
We have given name to JOB3 as ‘ojob’ in jenkins.

In this job firstly we have provided the url of our GitHub repository to our job.









**sudo docker exec devdocker /usr/sbin/httpd**

This command will help us to start the ‘httpd’ service in our ‘devdocker’ container.

‘/usr/sbin/httpd’ starts the ‘httpd’ service.

**st=$(sudo docker exec devdocker hostname -i)**

‘hostname -i’ command is used to get the IP address of the container. So we have extracted the IP address of ‘devdocker’ container and we have stored it in ‘st’ variable.

**status\_code=$(curl --write-out %{http\_code} --silent --output /dev/null $st/dev\_page.html)**

Here we have made a ‘status\_code’ variable and we have stored the status code returned by our ‘dev\_page.html’ page. This will help us to find that whether our page in the ‘dev’ branch is working properly or not. If it will be working properly then it will return status code as ‘200’ and if there will be any problem then it will return any other status code.

**if [[ "$status\_code" -ne 200 ]]**

**then**

**echo "Site status changed to $status\_code"**

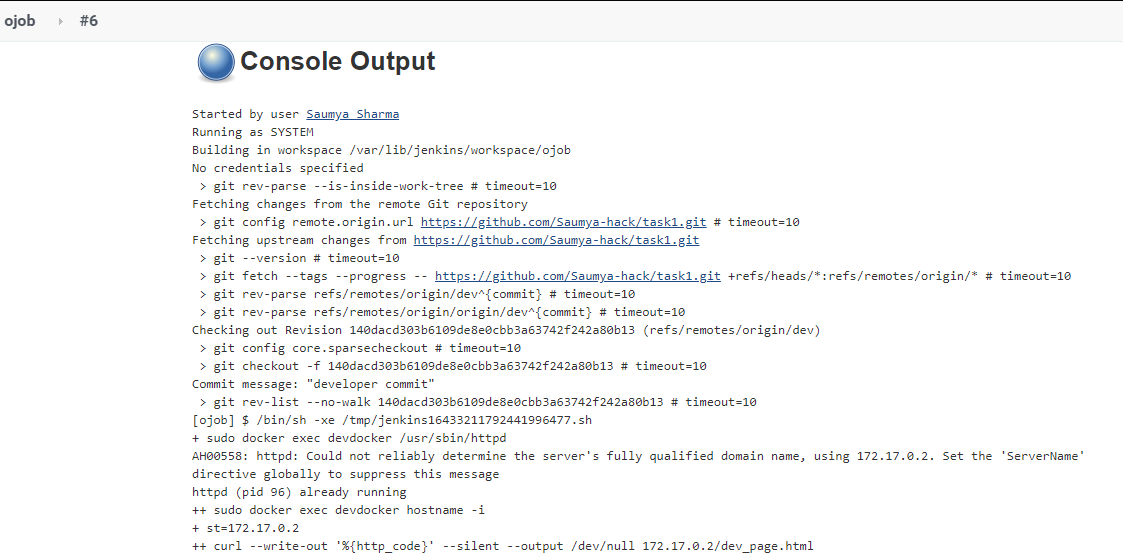
**else**

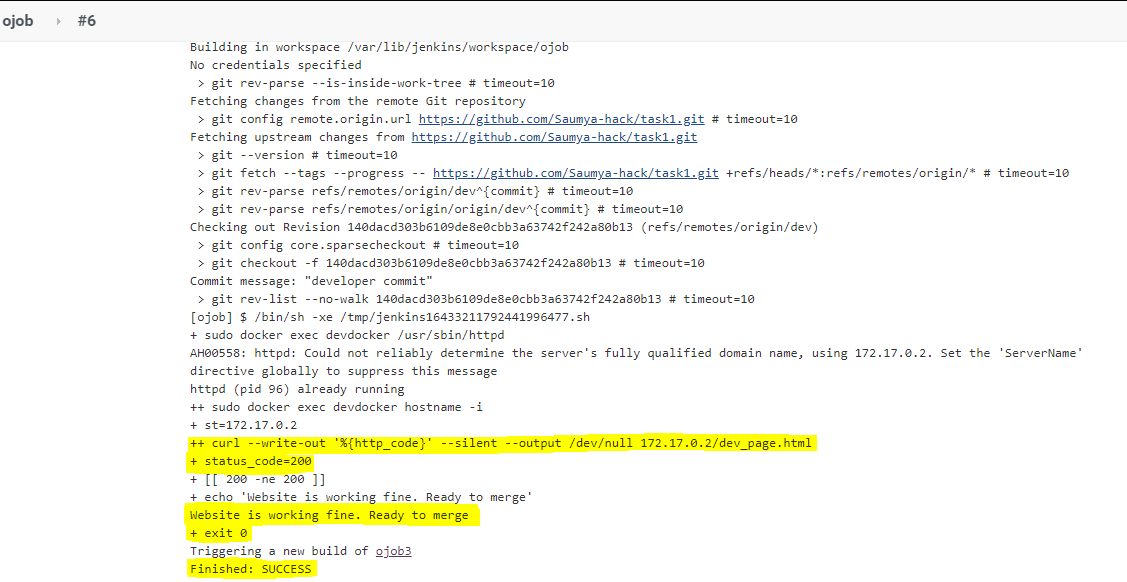
**echo "Website is working fine. Ready to merge"**

**exit 0**

**fi**

Above block of code will help us to determine that whether the status code returned by our webpage is 200 or not. If it is ‘200’ then it means that our webpage is working properly and if it is not equal to ‘200’ then it means that we have any problem with our web server our webpage. So if status code is 200 then we will get message on our console that “Website is working fine. Ready to merge”

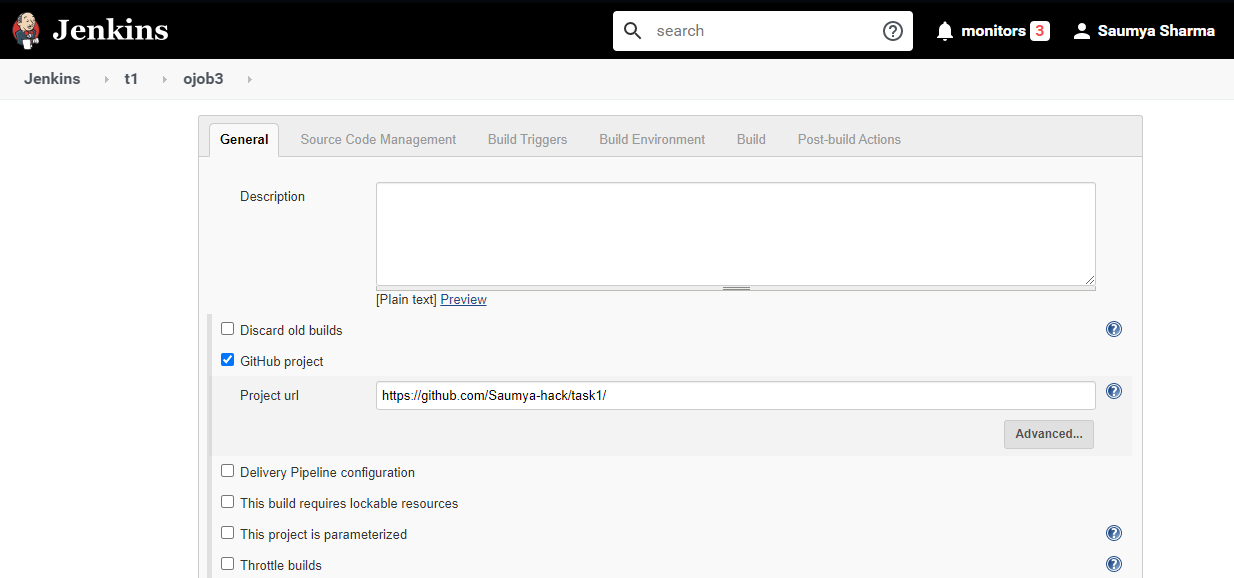


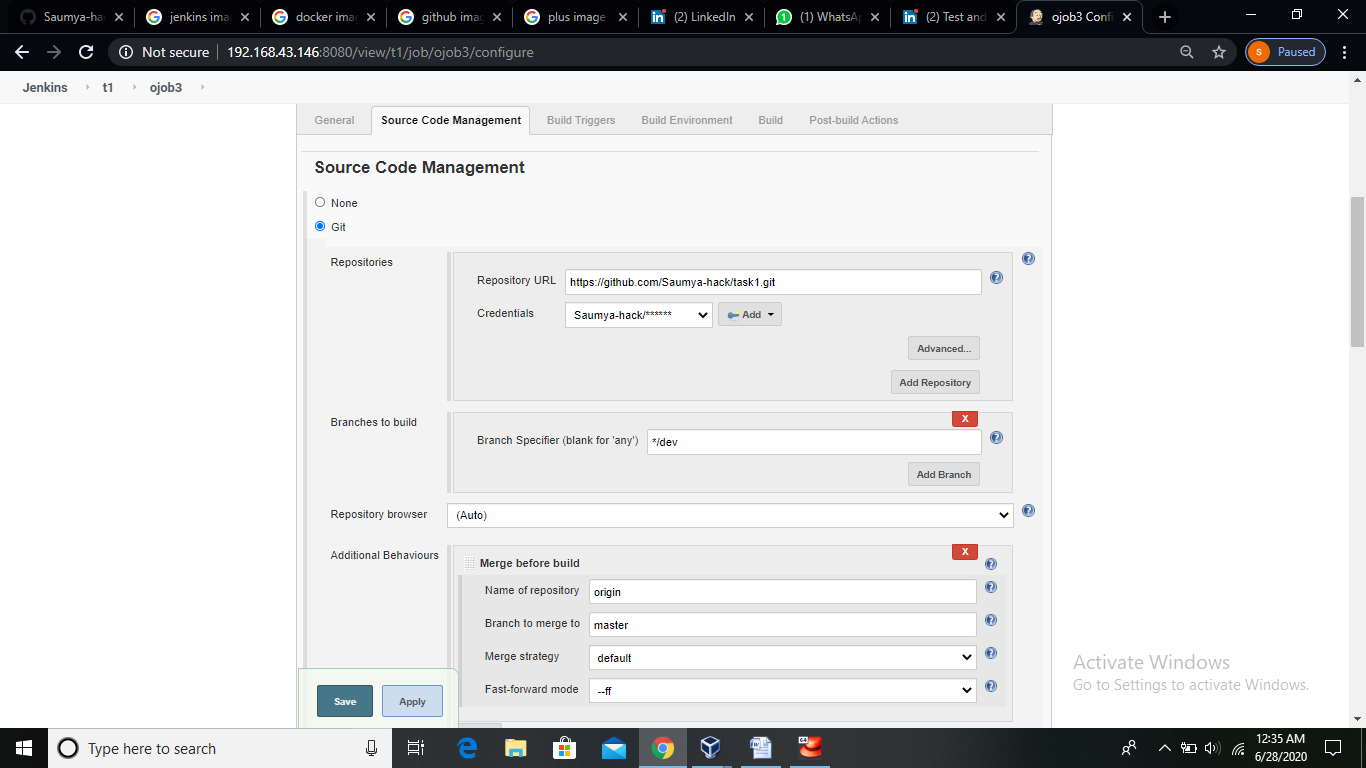


Since, we have got here status code equal to ‘200’. So, it means that our webpage is working properly. So this completes our testing process and now we can merge our ‘dev’ and ‘master’ branch.

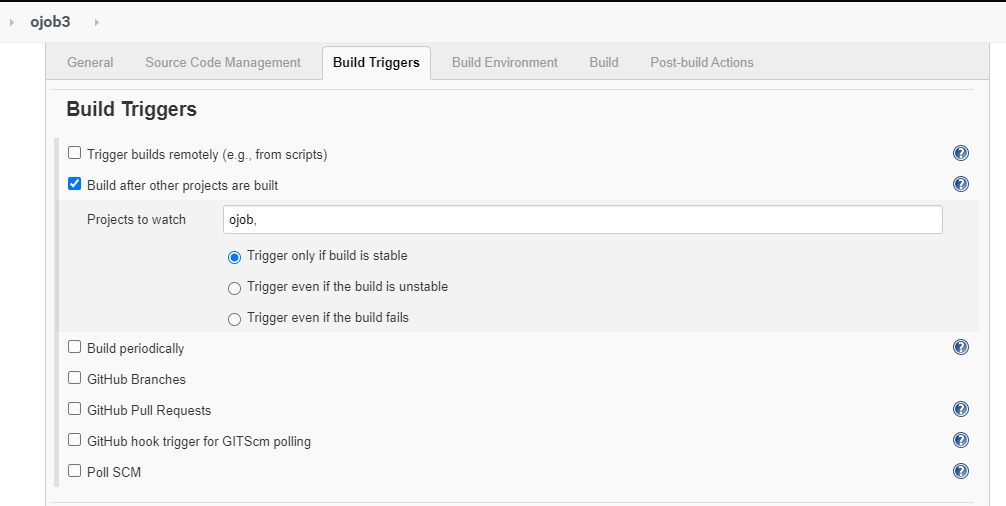
**JENKINS JOB 4(ojob3):**

We have given name to JOB4 as ‘ojob3’ in jenkins.



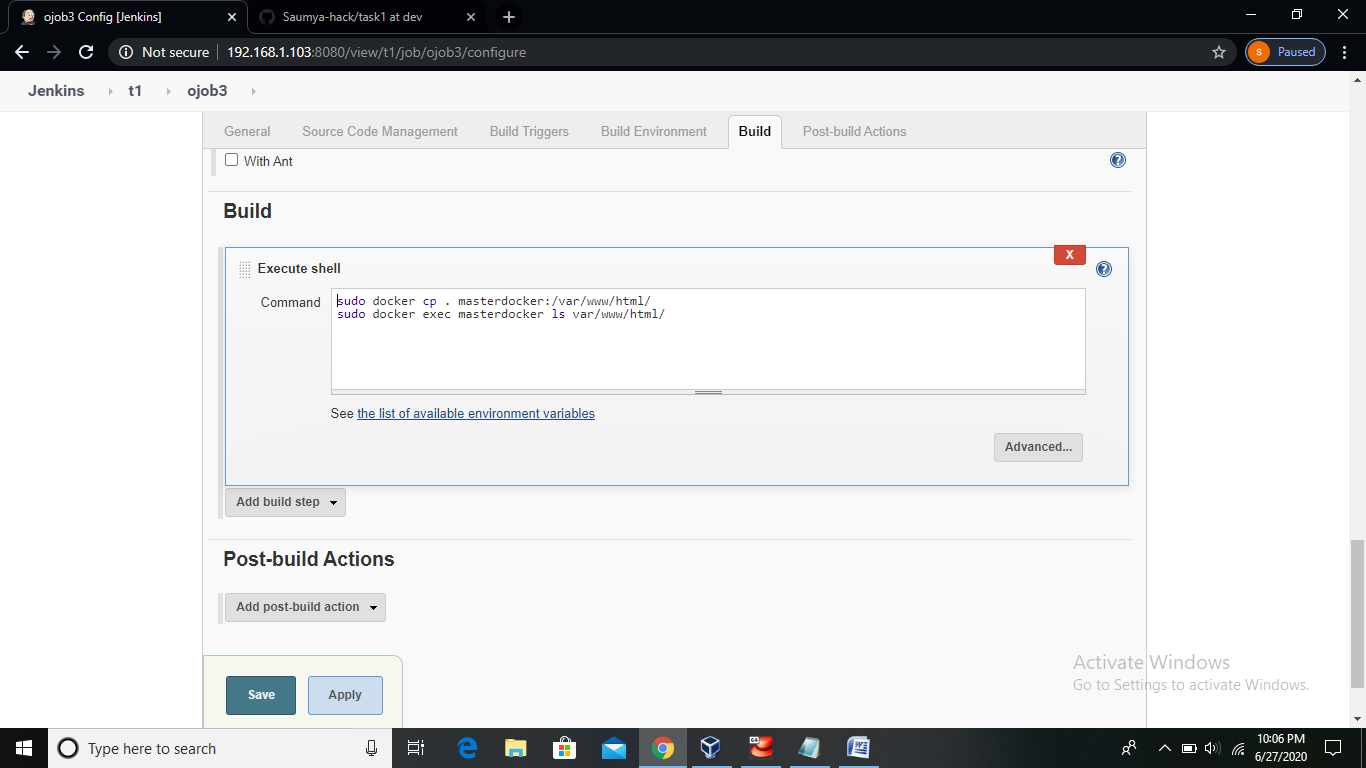


Here we are making setup for merging our dev branch to master branch.



As we have merged our both branches now , so now we are also combining the docker container data of both containers as shown below.

In following image we can see that we have copied data of ‘dev’ branch to ‘masterdocker’ container as in JOB3 our webpages were running properly.



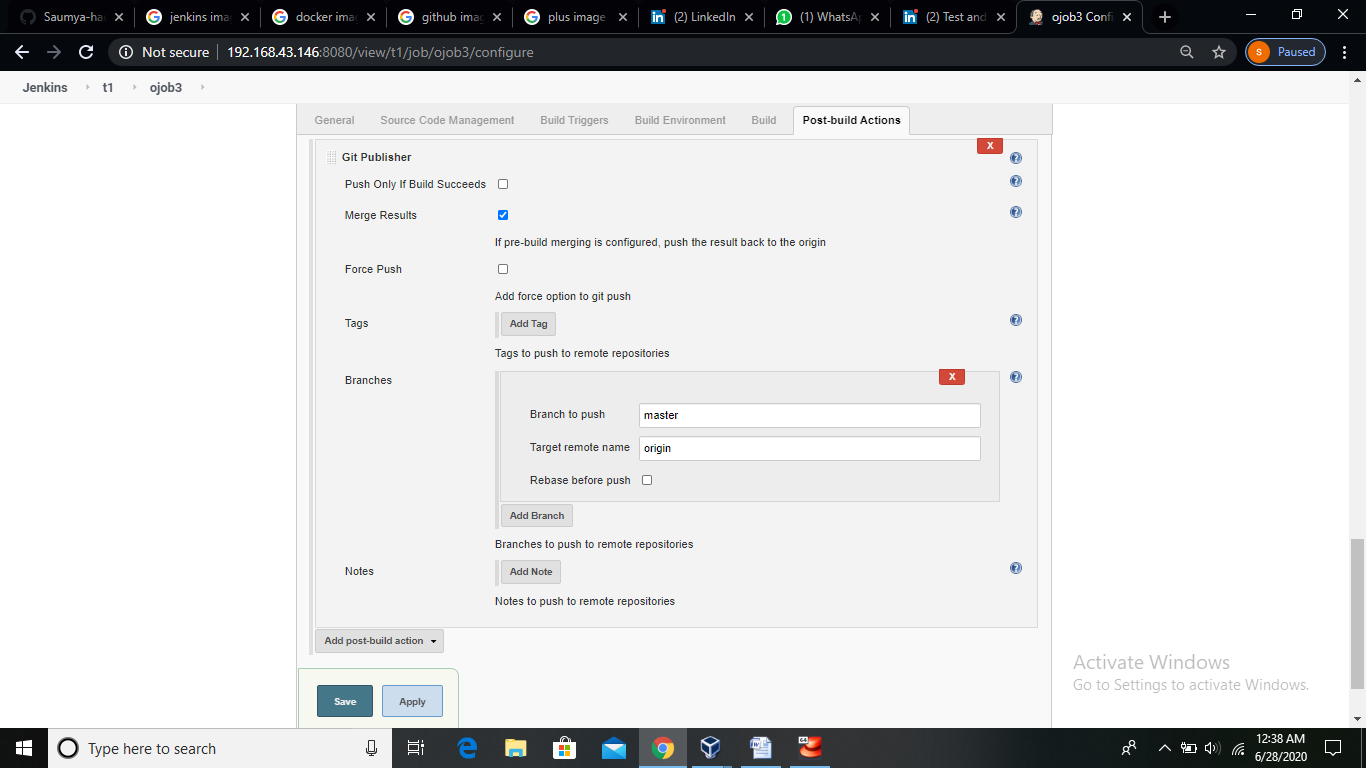
**sudo docker cp . masterdocker:/var/www/html/**

Since, in previous job we got that our ‘dev’ branch webpage is working fine so we have copied our ‘dev’ branch data also in ‘masterdocker’ container.

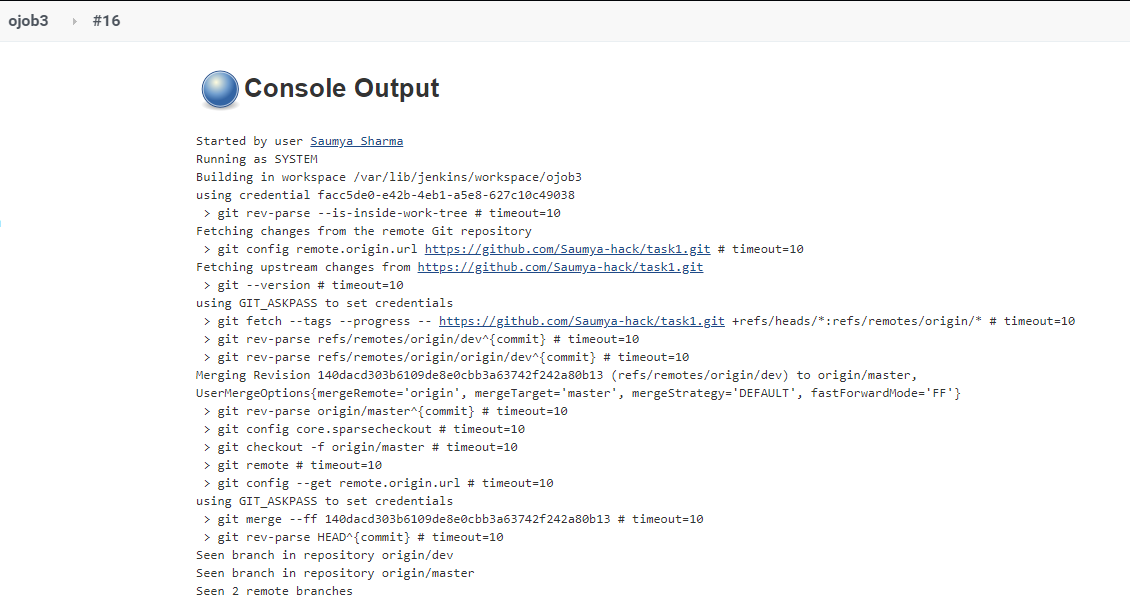
**sudo docker exec masterdocker ls var/www/html/**

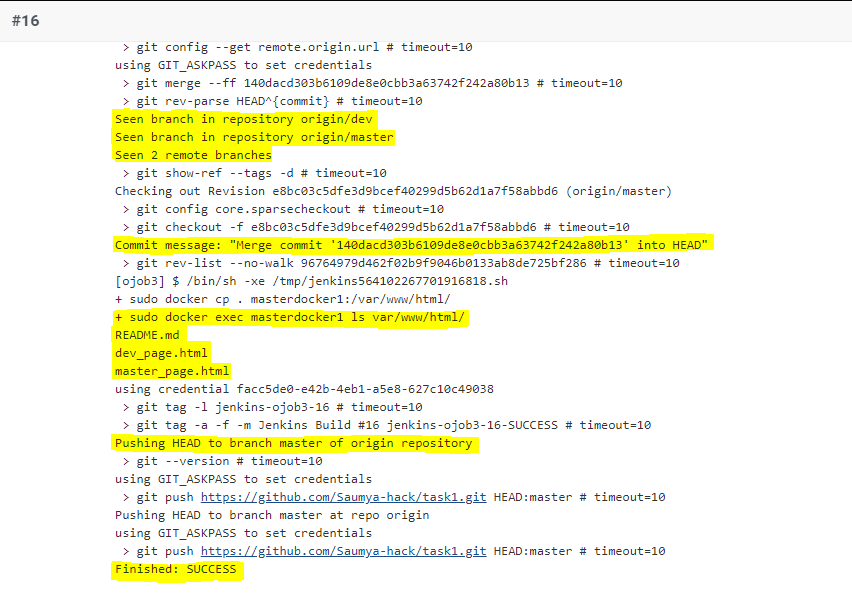
This is just to check on console that whether our job merged the required data to /var/www/html/ successfully or not.

Following image shows some post-build action we are required to do to merge branches.



Console Output of JOB4:

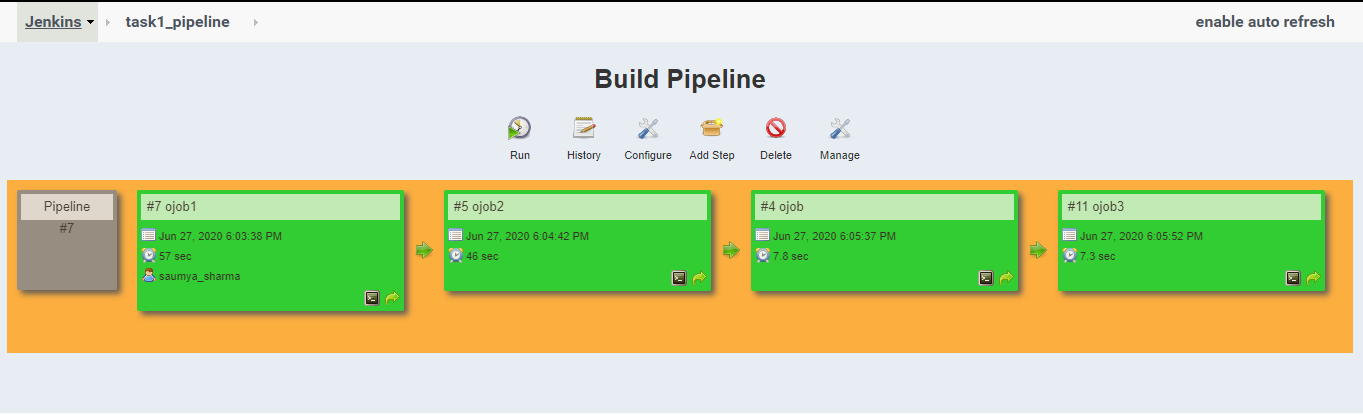




Here it is shown on console that both branches are now merged and also now we have merged the ‘devdocker’ container data with ‘masterdocker’ container data.

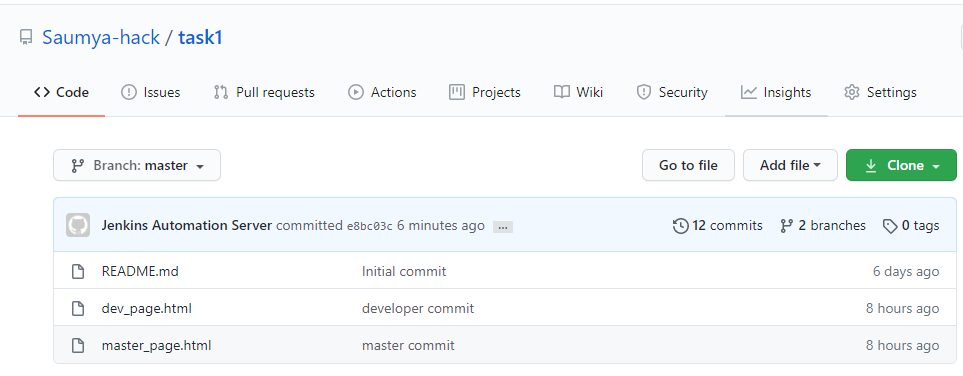
**JENKINS PIPELINE:**

Here we can see through our build pipeline (task1\_pipeline) that all the Jenkins jobs have been executed successfully.

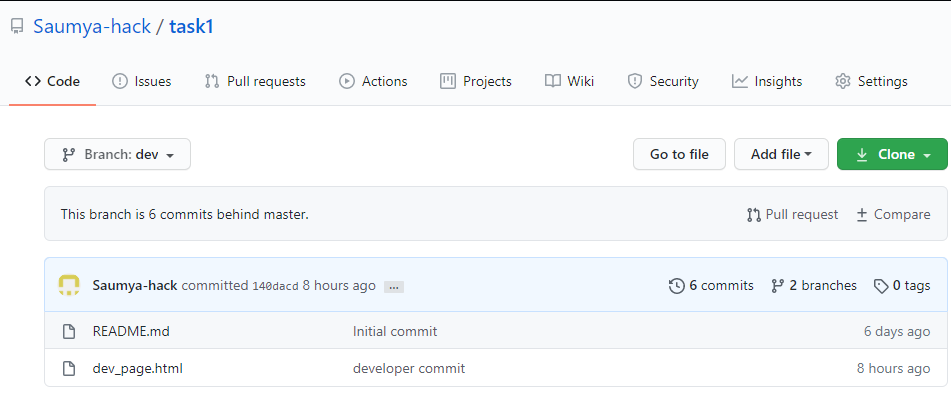


**GitHub repository branches after executing all jobs:**

**Master branch:**

****

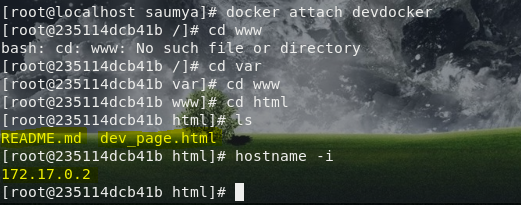
**Dev branch:**

****

**Docker Containers made by Jenkins jobs:**

**devdocker container:**

This is the data of ‘devdocker’ container.

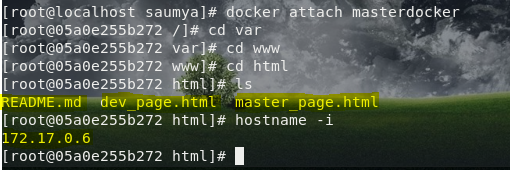


Following image is showing that our webpage in the ‘dev’ branch was working properly.



**masterdocker container:**

This is the data of ‘masterdocker’ container.



So this is the process which we are required to follow to merge the GitHub branches and we can also say to merge docker container data after testing the ‘dev’ branch files.

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

**Thanks for reading.**