**Complete end-to end Automation integrating Docker, Github with Jenkins CI/CD**

**Integration of Jenkins, Docker and Github**

**Pre-Requisites**

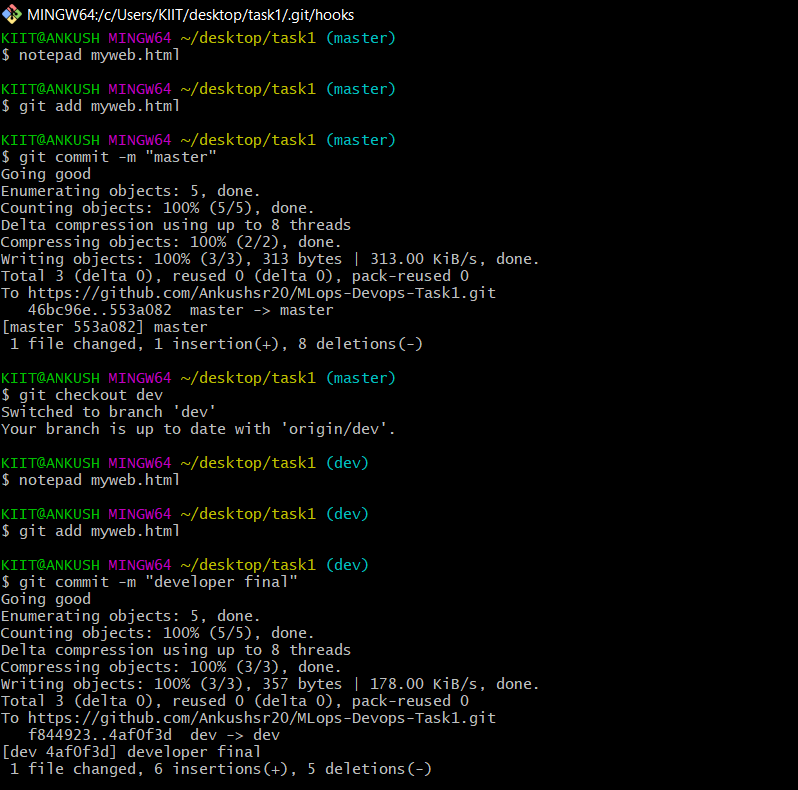
1. RHEL 8, Docker, Jenkins and Git Bash.
2. Make sure you have Jenkins and Docker installed on rhel8 and Git bash installed on windows and also in Jenkins make sure that you have github plugin installed.
3. First disable the firewall in rhel8 using systemctl stop firewalld.
4. Now Start Docker and jenkins using the command **systemctl start docker** and **systemctl start jenkins**.
5. Now open jenkins in your windows using your local Ip address.  
    Also start Git bash and configure git remote in it.
6. To do automatic push once commit is made in git remote then go to **.git/hooks** directory and create a post-commit.sample file and inside it write

**#! /bin/bash**

**echo “Going good”**

**git push**

Firstly, we need to create a [github repository](https://github.com/Ankushsr20/MLops-Devops-Task1" \t "https://medium.com/@srank2000/_blank).  
Then, we are going to create a branch dev. After creating a html webpage, we will upload it there. We just have to commit to the repository, it will automatically get pushed!

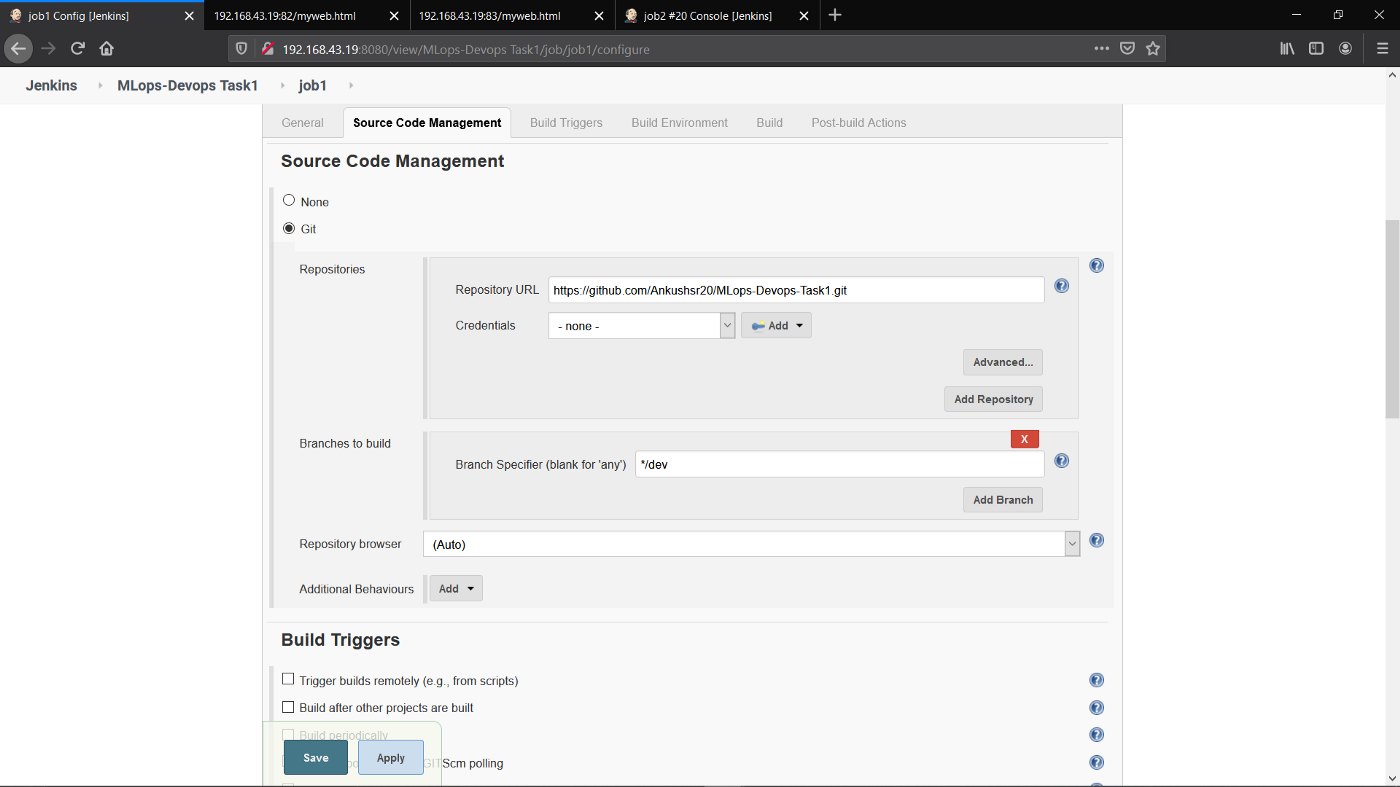


Commit to github

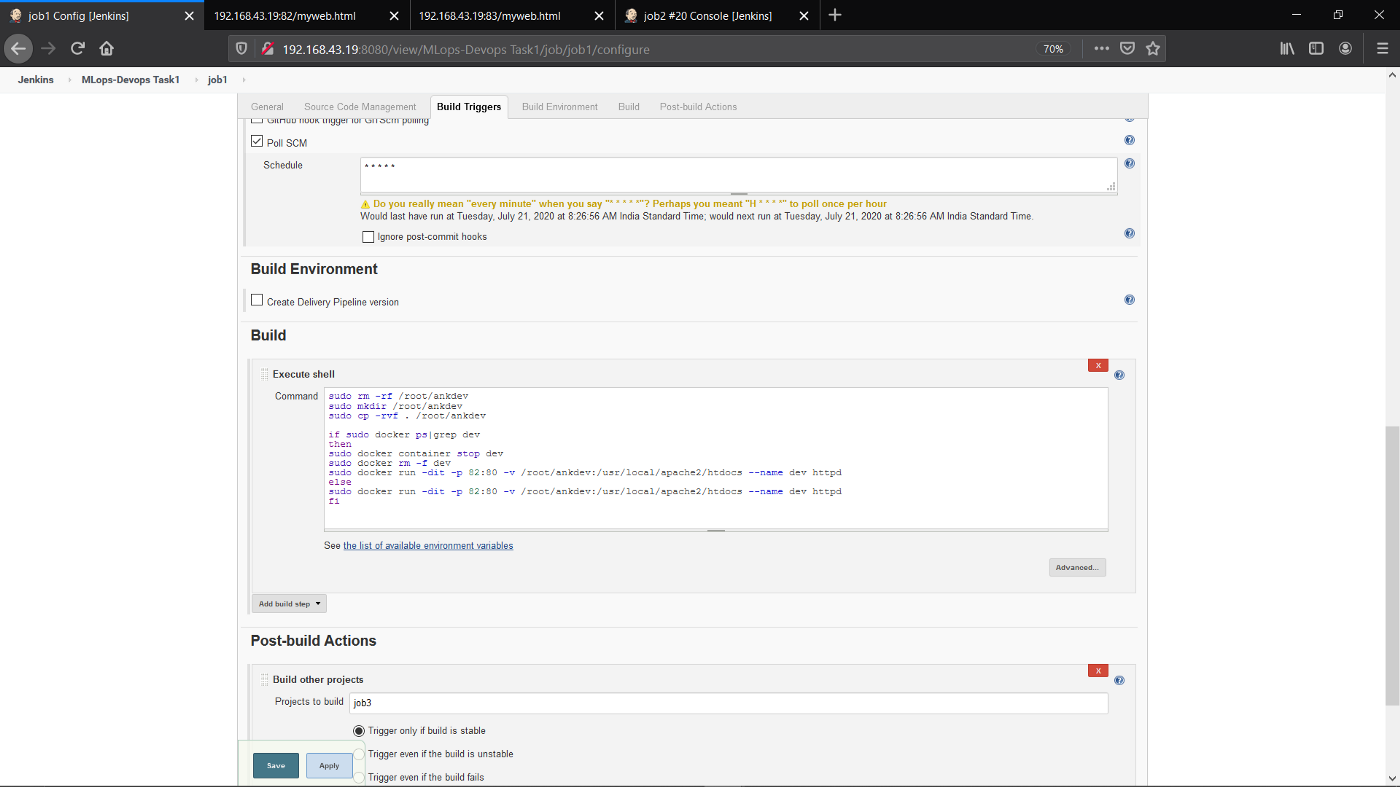
Now, let’s take a look at the Jenkins Jobs we are going to build!

**JOB 1**

If Developer push to dev branch then Jenkins will fetch from dev and deploy on testing\_environment.



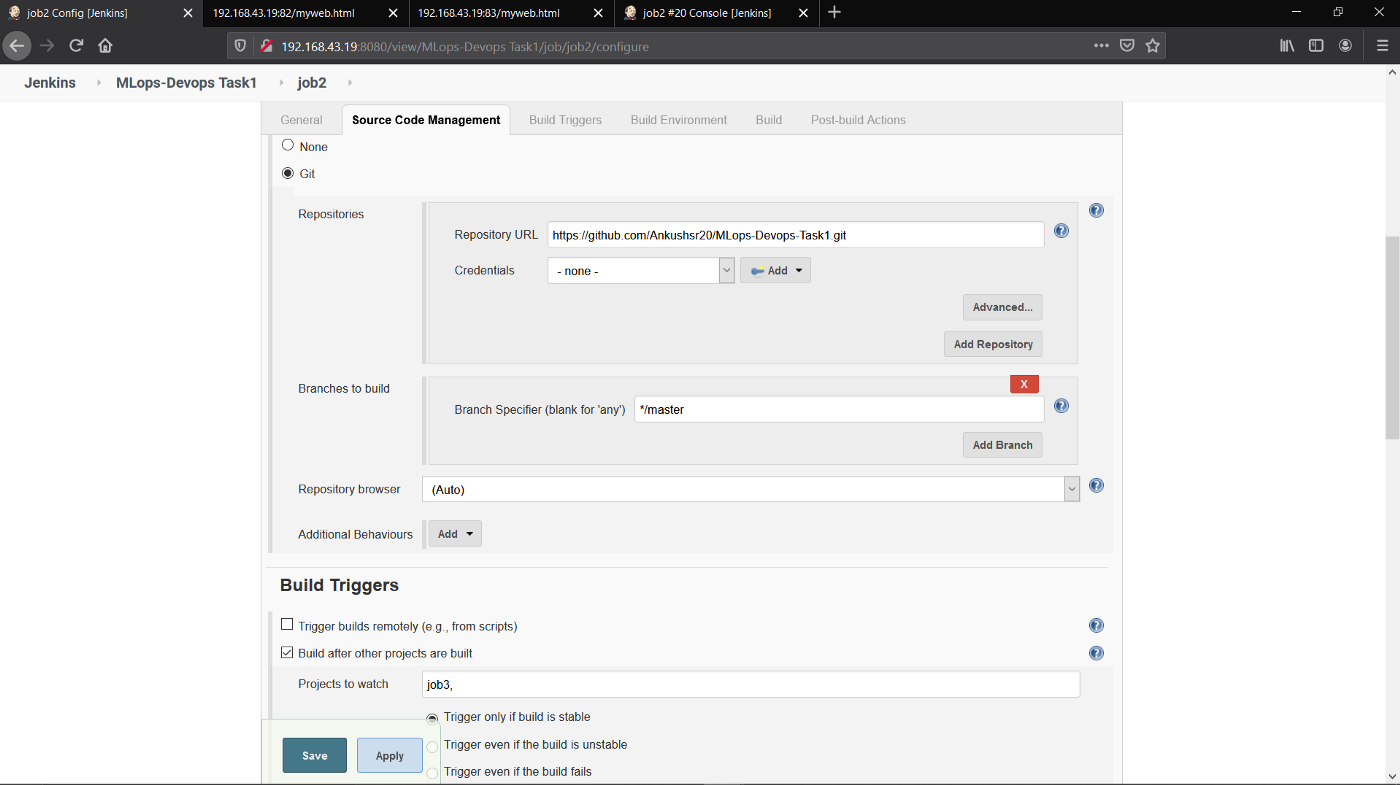
Job 1.1



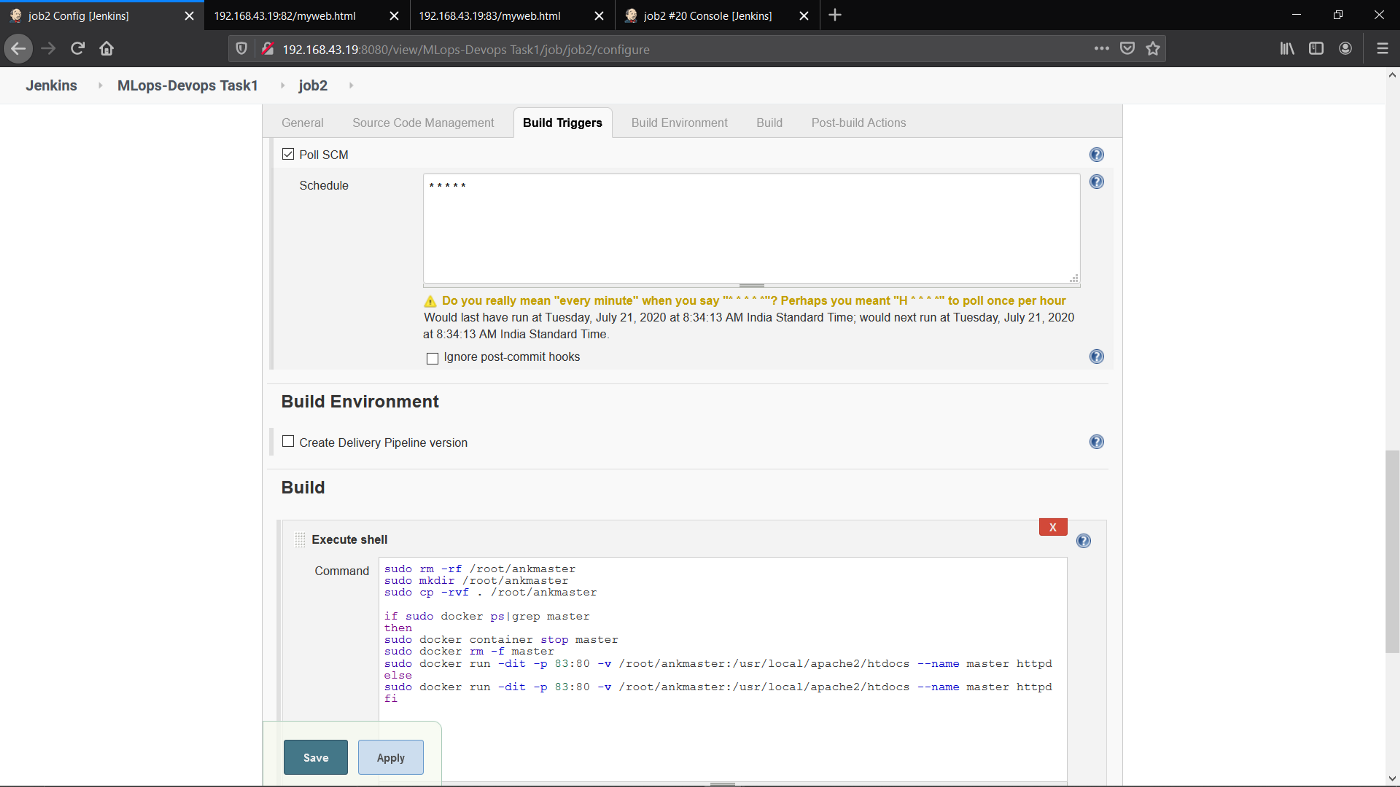
Job 1.2

**JOB 2**

If Developer push to master branch then Jenkins will fetch from master and deploy on production\_environment. Both testing\_environment and production\_environment are on different docker containers.



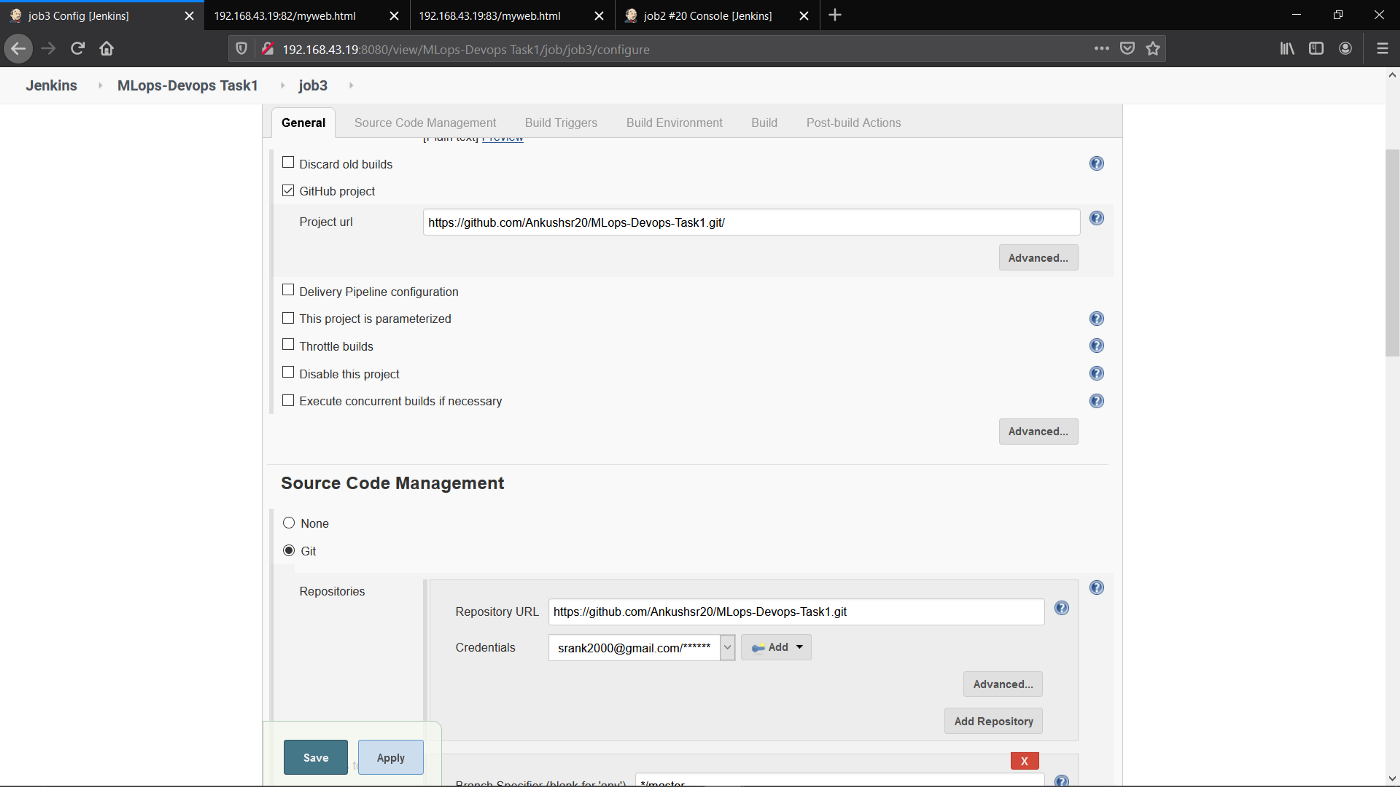
Job 2.1



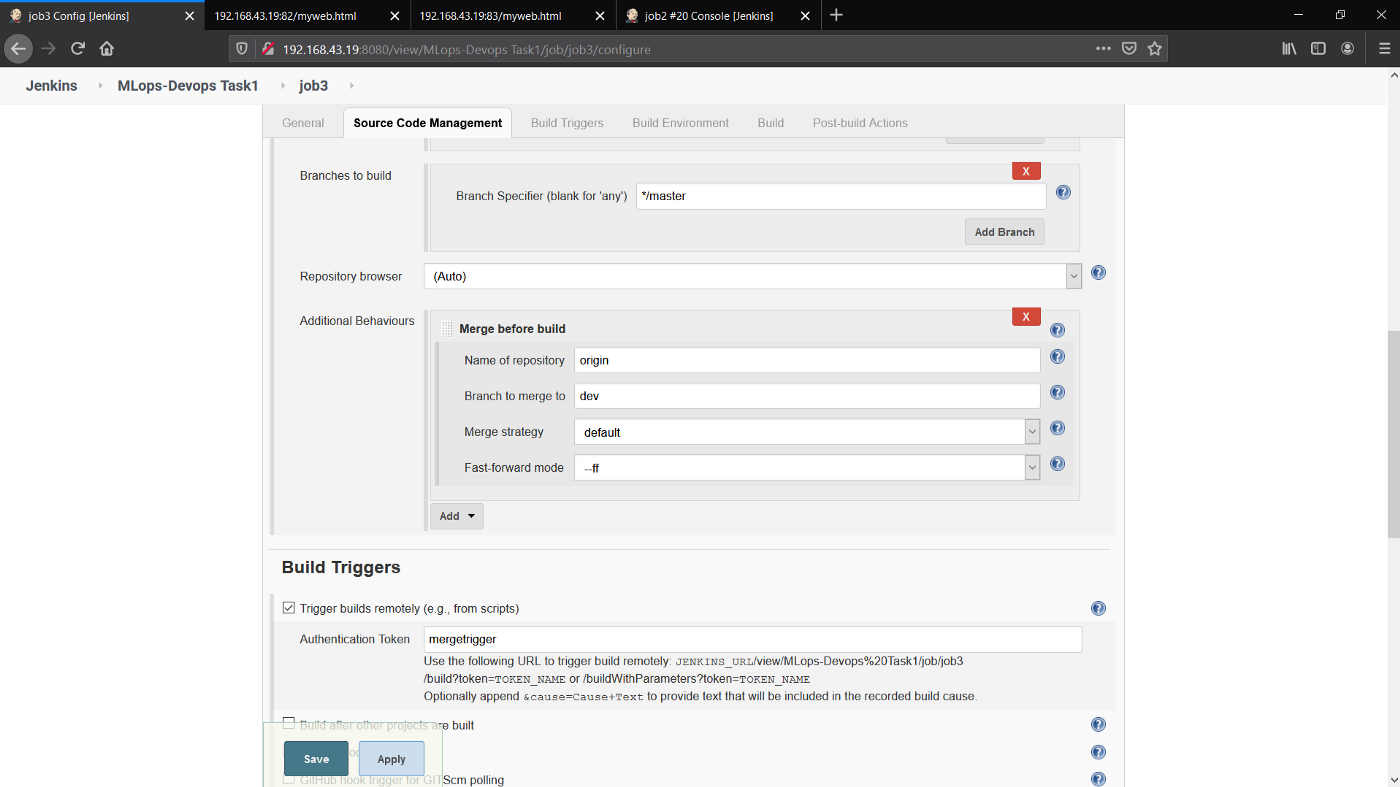
Job 2.2

**JOB 3**

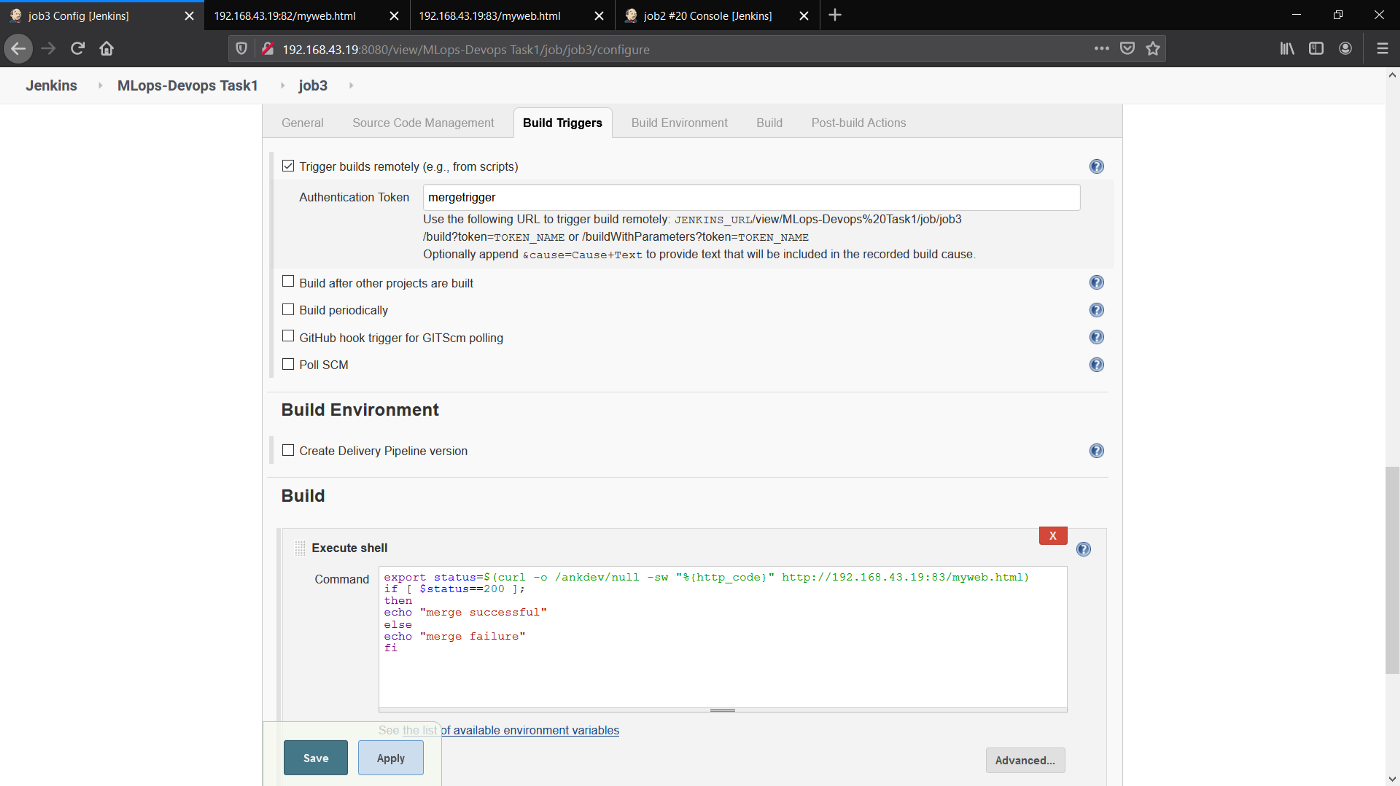
Manually the Quality Assurance team will check (test) for the website running in testing\_environment. If it is running fine then Jenkins will merge the dev branch to master branch and trigger **[job](http://tg//search_hashtag?hashtag=job" \t "https://medium.com/@srank2000/_blank) 2.**



Job 3.1

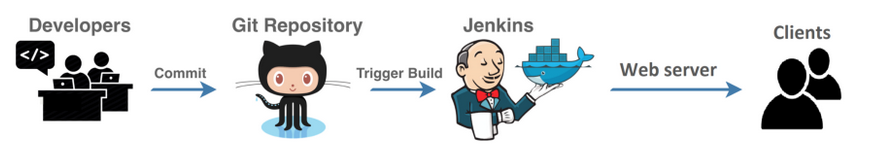


Job 3.2

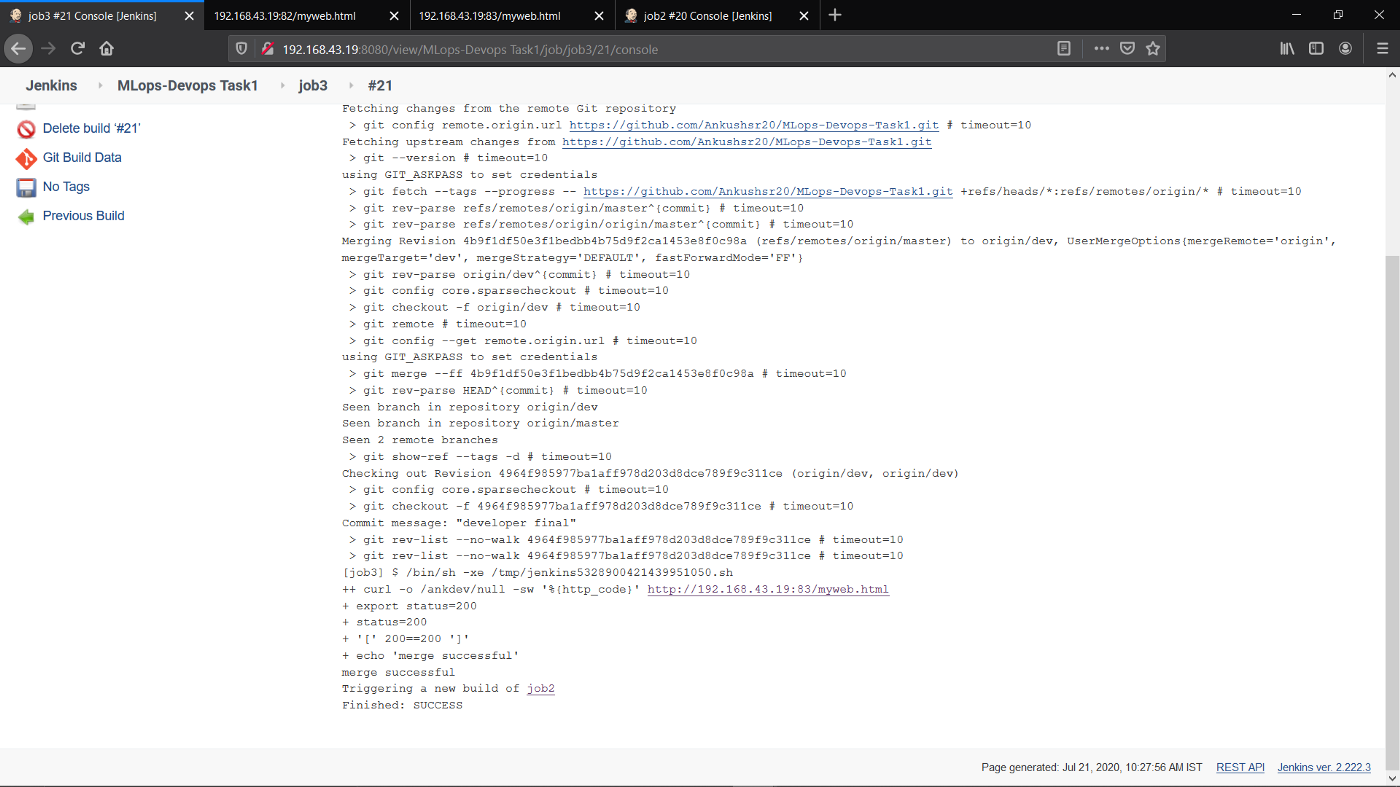


Job 3.3

**Final Workflow:**



As soon as we commit, by just one click, the jobs will get triggered, thus achieving an end-to-end automation.



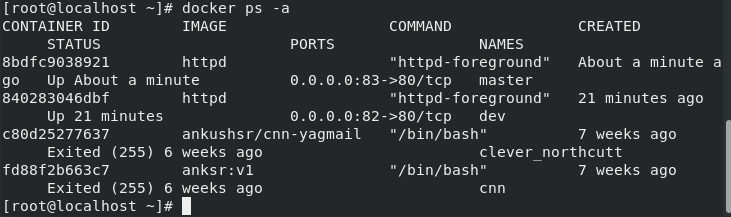
Job3 merge successful.

So let’s what figure out what actually is going on.

Behind the scene Docker service is running on the machine and that have httpd image preinstalled and Jenkins is downloading the git files locally and run the docker container on port 82 and 83 with placing the respective files in the container and whenever it receive new files updated on the GitHub it automatically download files locally and replacing the previously running docker containers .

Thus we achieve a zero-downtime setup.

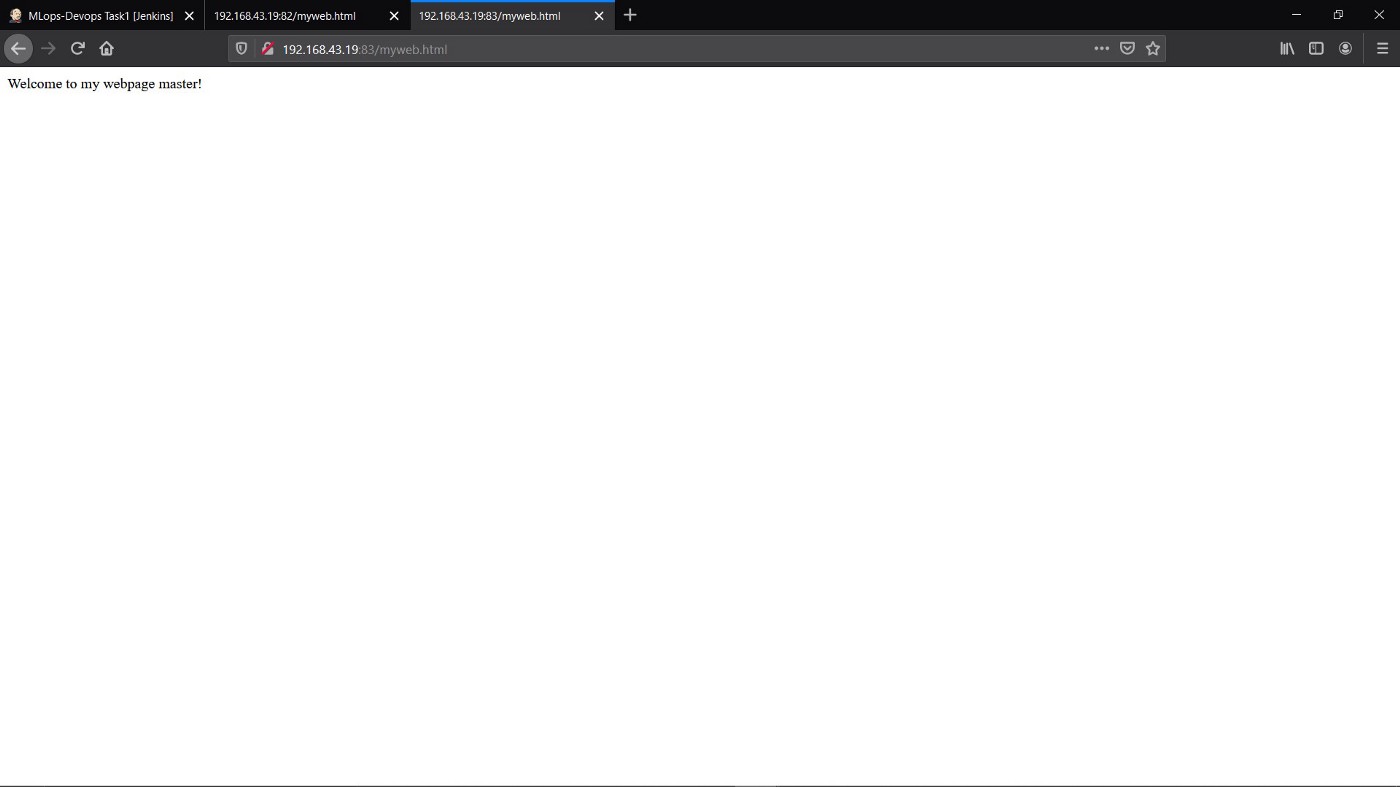
We can check the running containers using docker ps -a command.



Docker containers running

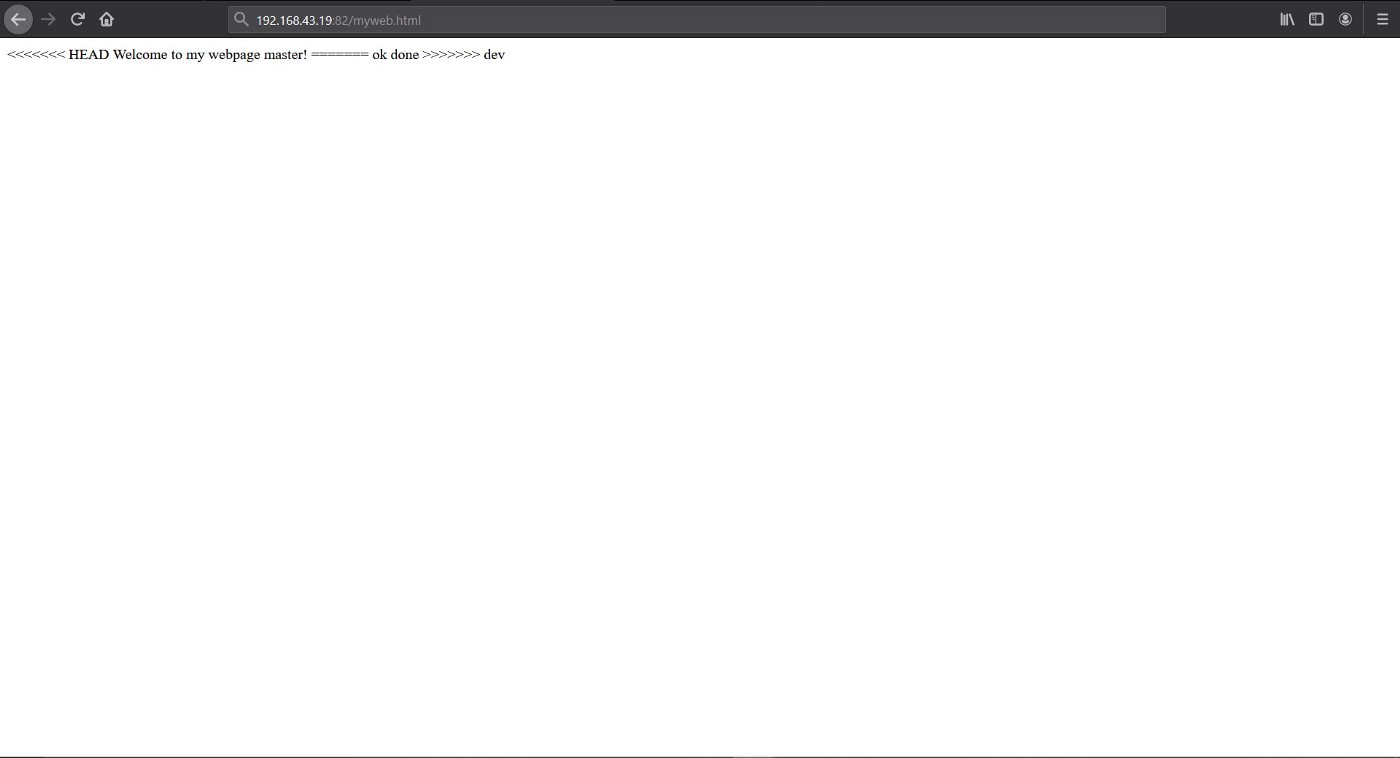
Now, If the Developer Push the file in the master branch then it will automatically download and directly update the main site running on port 82 . If Developer push and change the file in dev branch, then it deployed in another site running on port 83 which the quality assurance team will verify manually and after visiting that URL will trigger the Job 3 on Jenkins . And Job 3 will merge to the dev branch with master branch and also trigger the Job 2 and the main site is updated without doing anything .

Let’s check http://IP:83/myweb.html // master



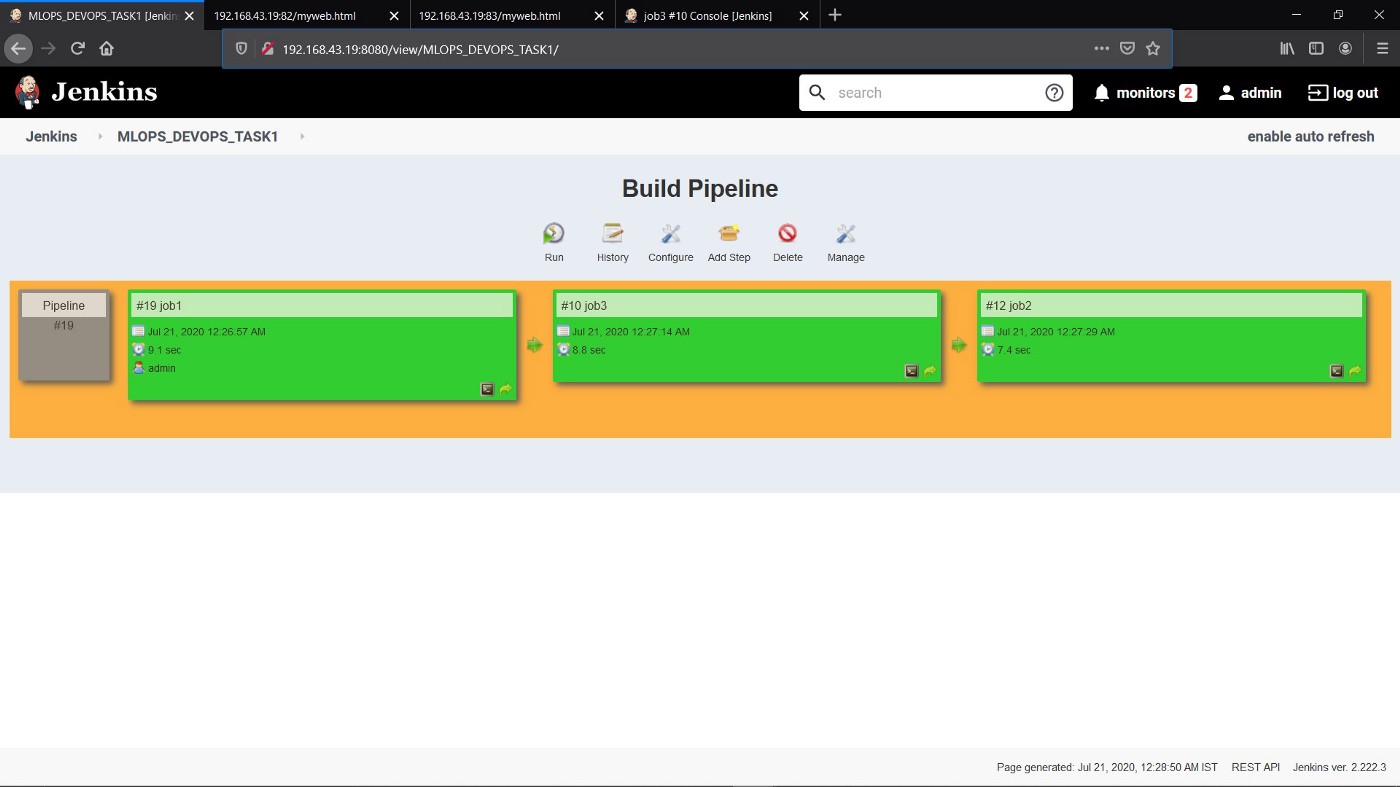
Port 83 (Master)

Now let’s check [http://IP:82/myweb.html](http://ip:83/myweb.html" \t "https://medium.com/@srank2000/_blank) // dev



Port 82 (dev)

Now the Jenkins Jobs should run!



Build Pipeline view of all the 3 jobs

Finally let’s check http://IP:83/myweb.html again and check the result.

