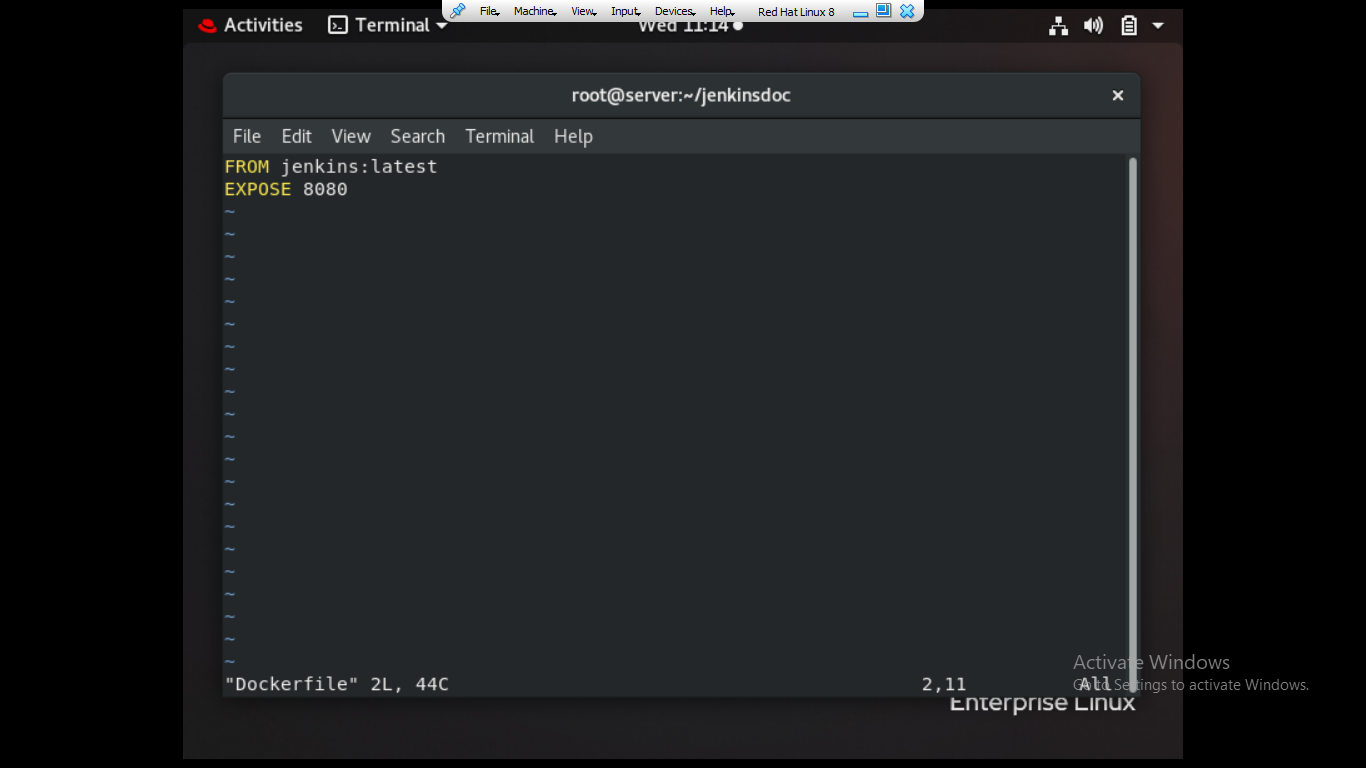
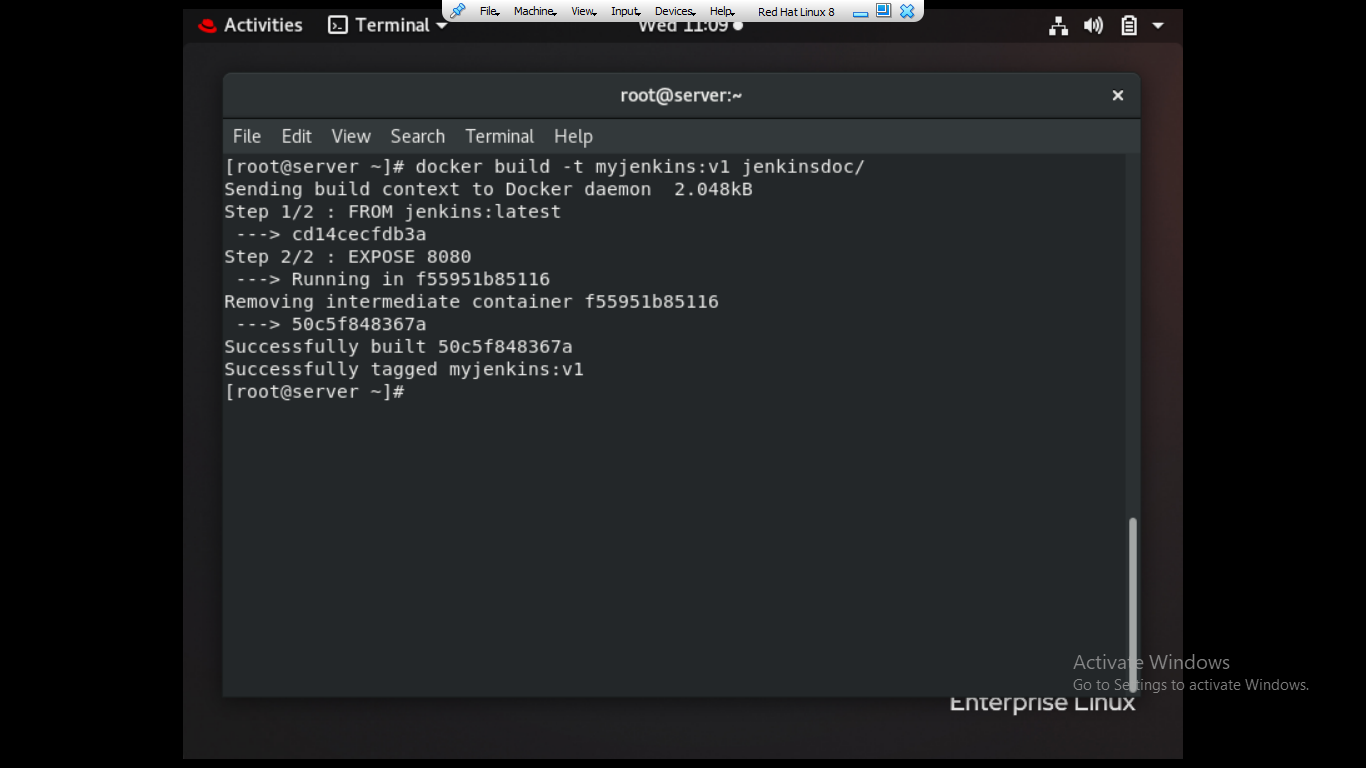
TASK2

$ Create a image using dockerfile

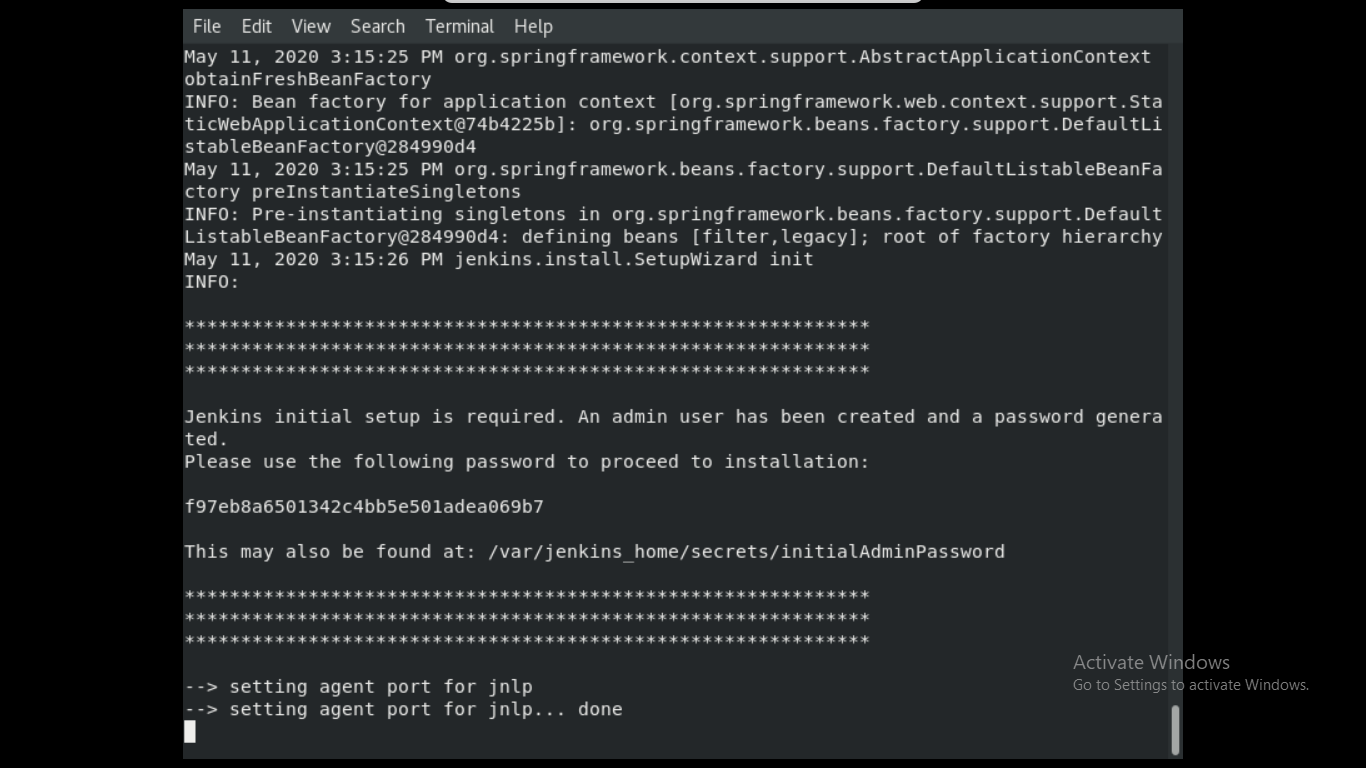
1. Create a directory mkdir jenkinsdoc
2. cd jenkinsdoc/
3. vim Dockerfile
4. FROM jenkins:latest
5. EXPOSE 8080



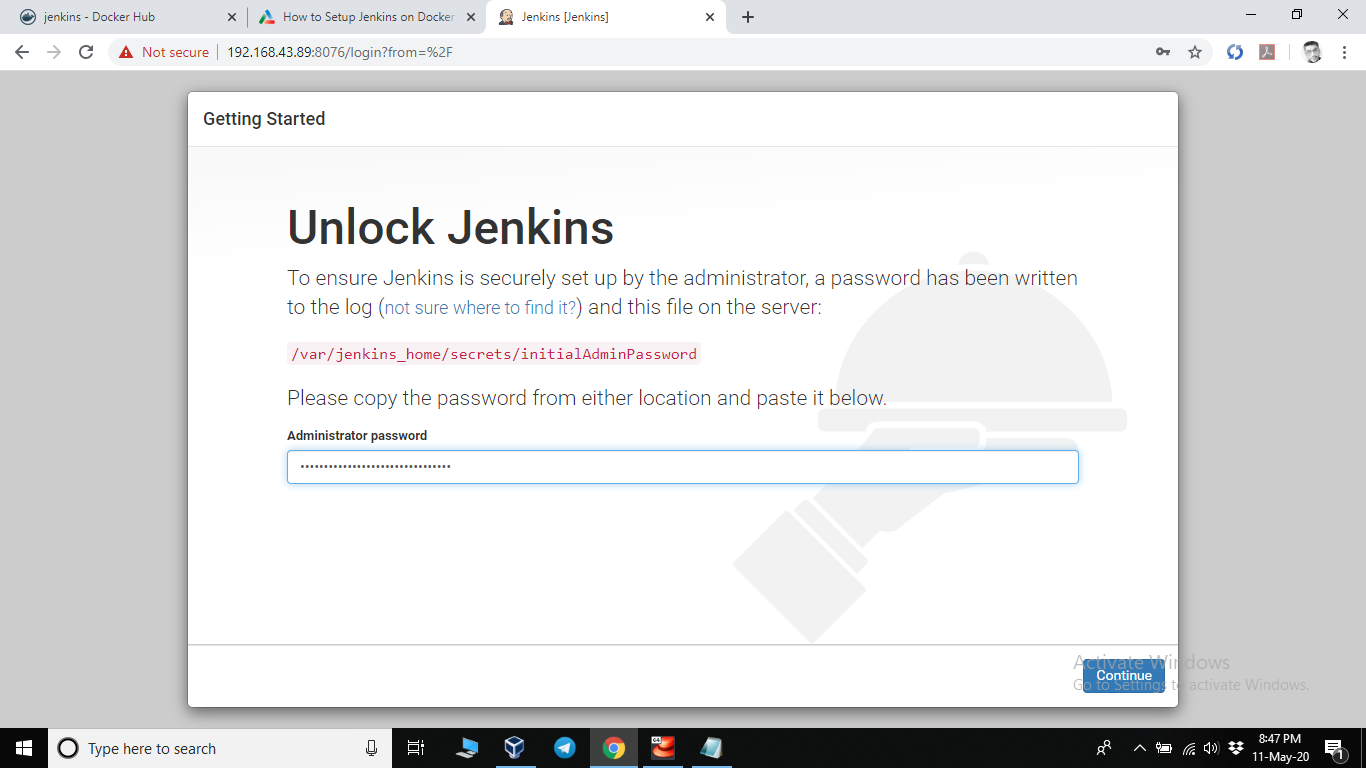
1. Save the file
2. Now build the file: docker build –t myjenkins:v1 jenkinsdoc/



1. Run the image
2. Docker run –dit –name myjenkinsoa –P 8080 myjenkins:v1



1. Open jenkins in browser using ip:8080

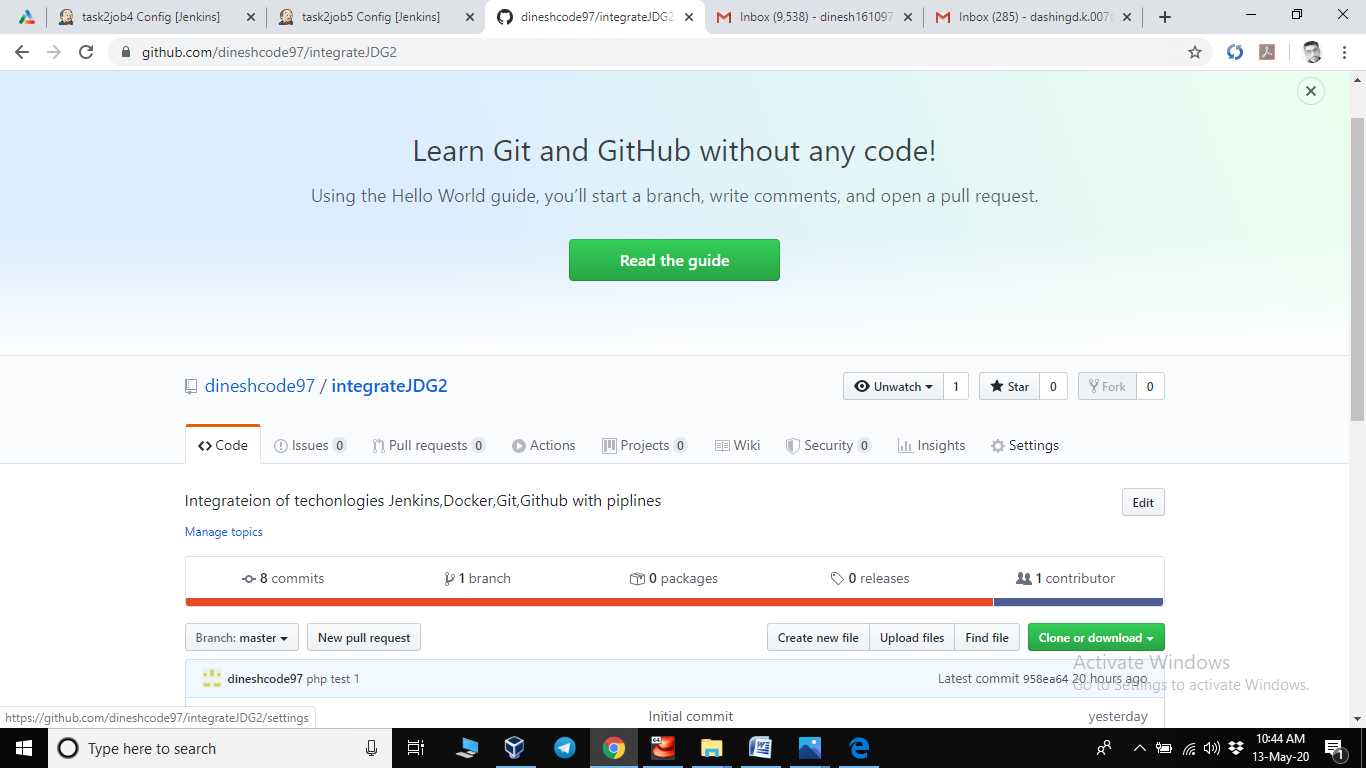


1. Paste initial password here

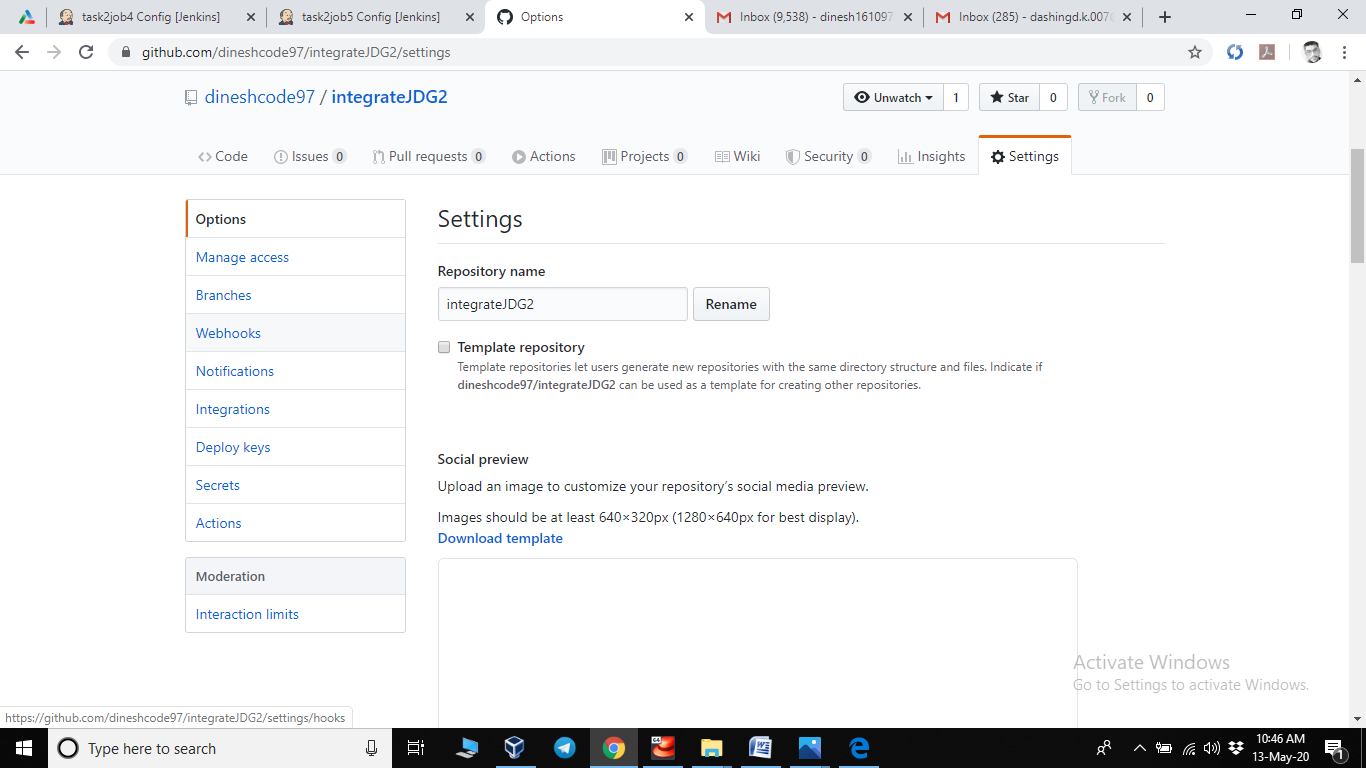
12 .install plugins like github,buildpipline

GITHUB:

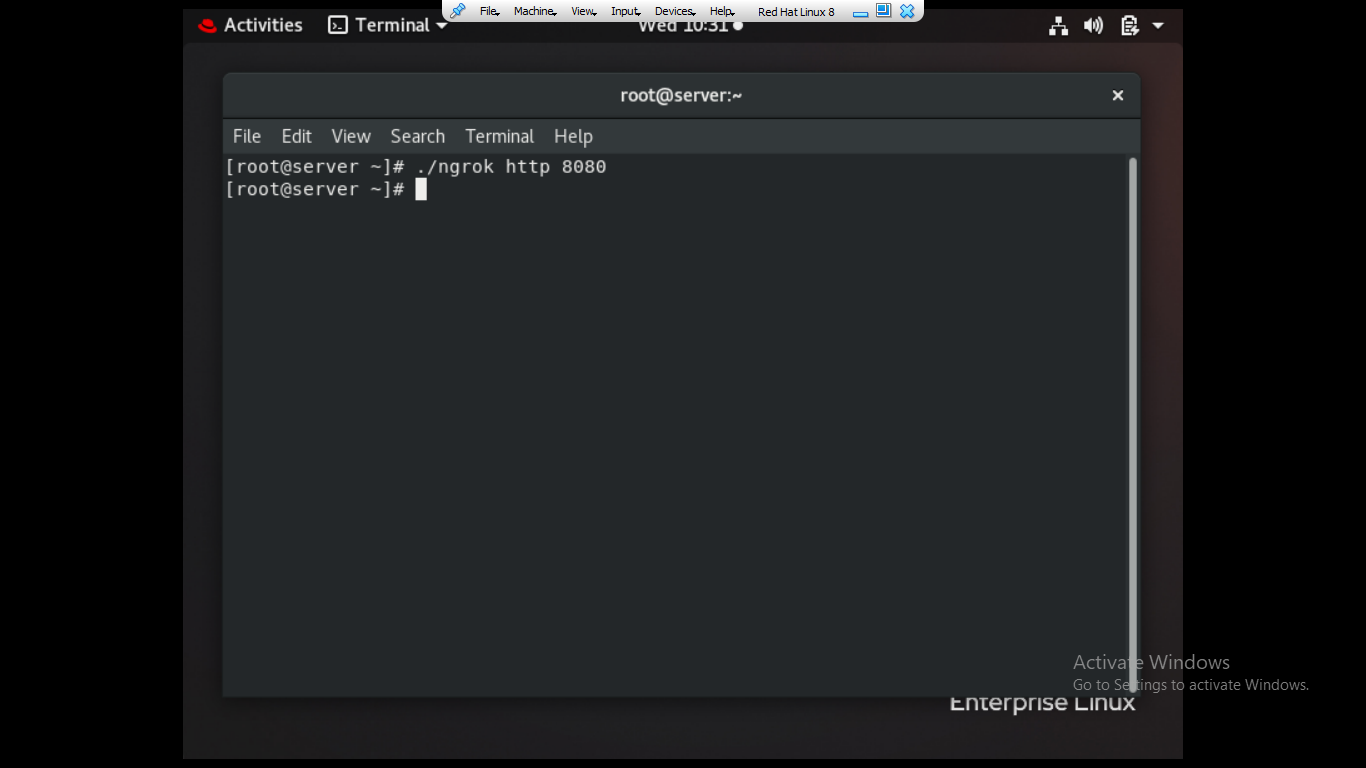
1. Create a github repository
2. Goto setting



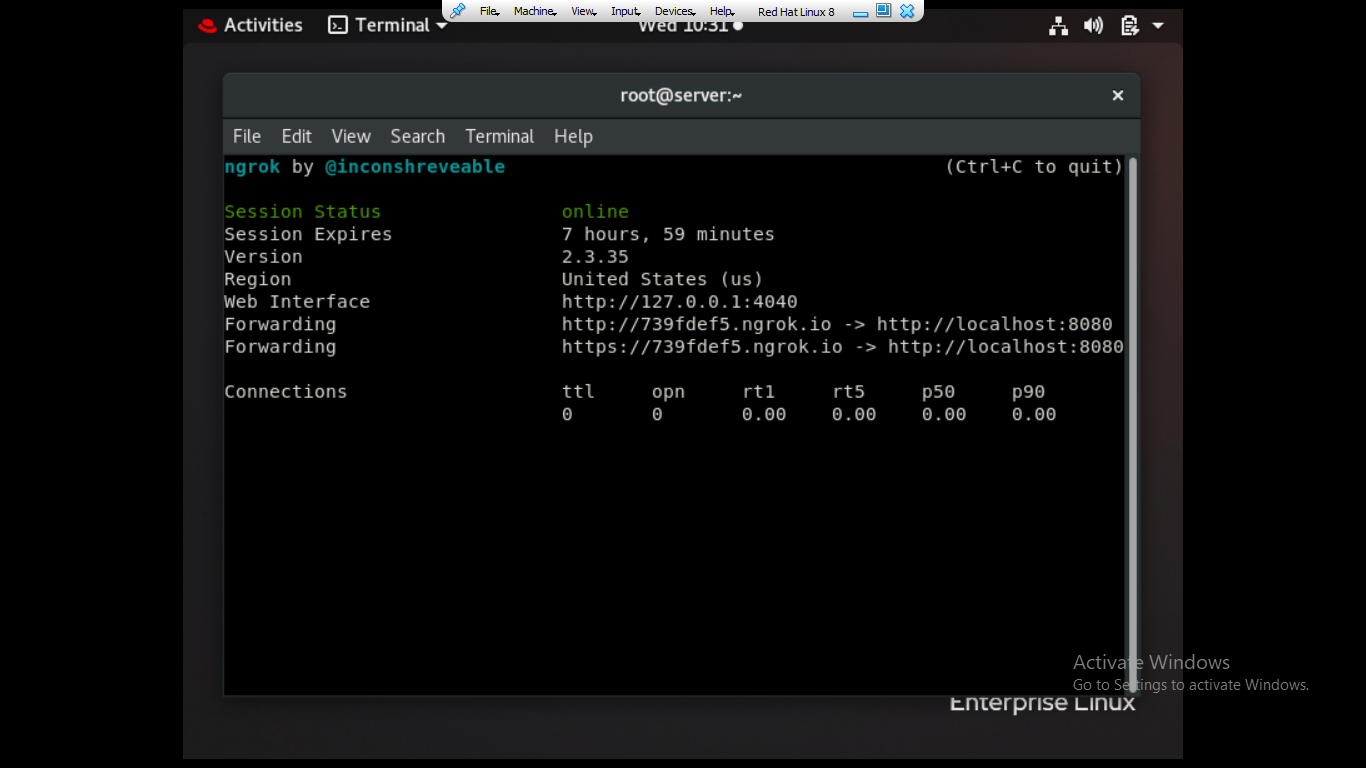
1. Select webooks



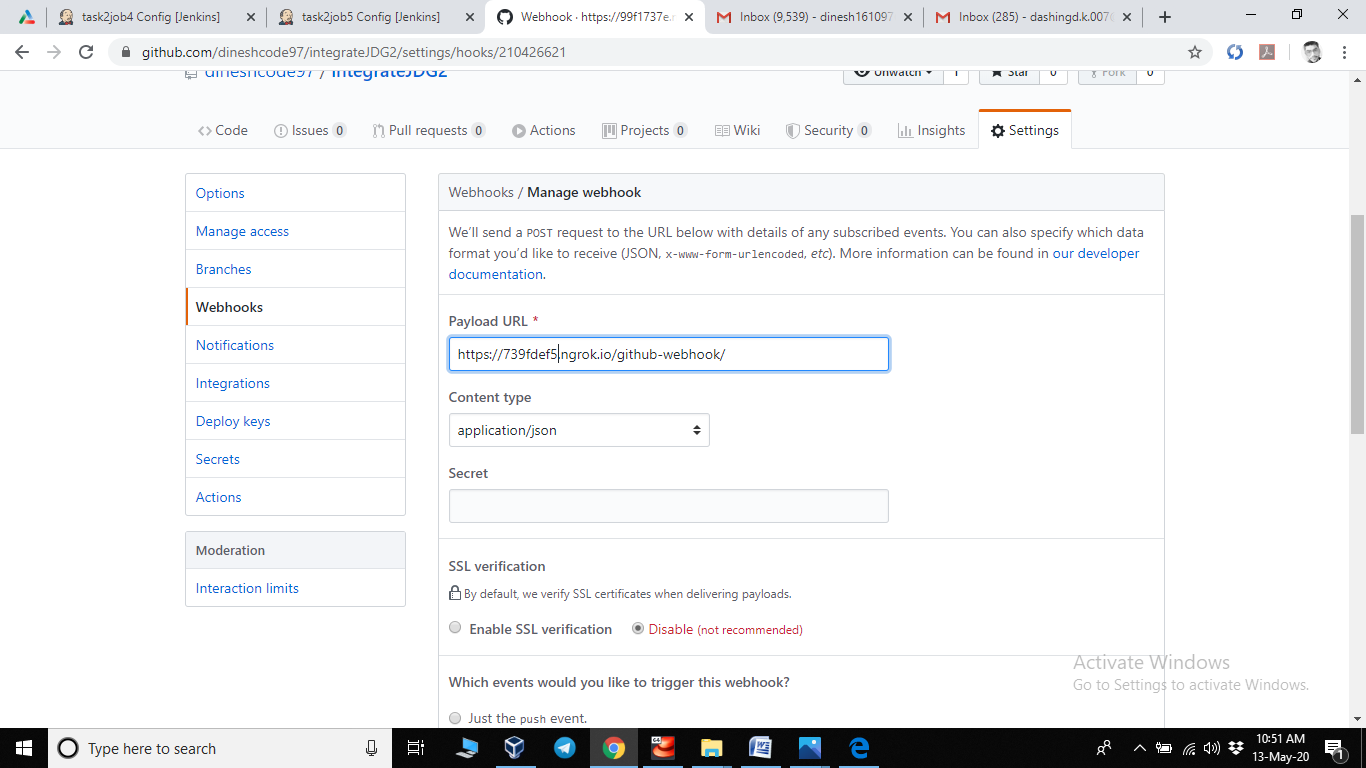
1. In your vm install ngrok and run ./ngrok http 8080



1. Copy the url



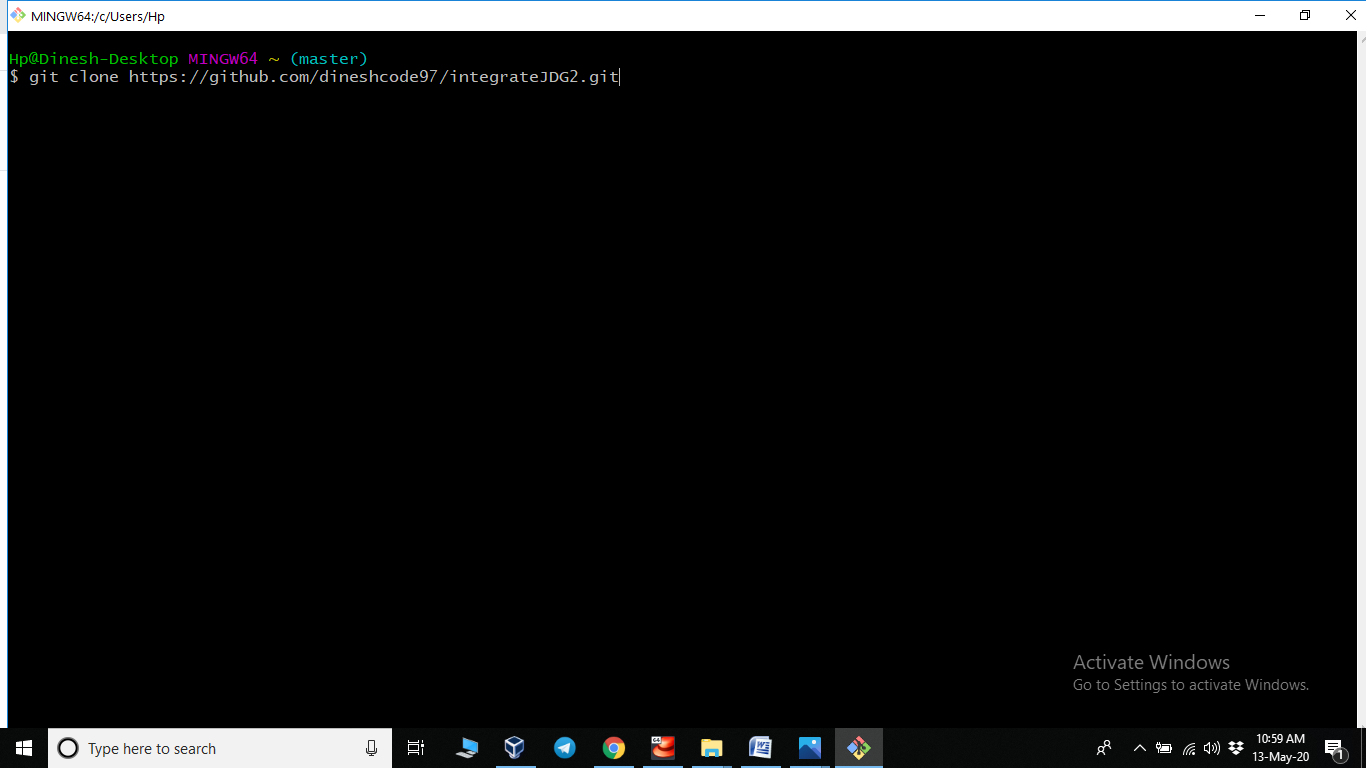
1. Paste url here in webhook in box payload url and type /github-webhook/ ahead url
2. Select content type application json
3. Disale ssl verifivation and select just push event
4. Click create webhook



# Your webhook created

Git:

1. Copy url from github and open git bash then
2. Use git clone githuburl (githuburl is url which you copy from your repo paste after clone)



# After this create files and commit to github

=>Local hook

cd .git/hooks/

cat post-commit

#!bin/bash

git push

(press ctrl + z or ctrl + c)

Note : local hook is created if you commit is push automatically

For example testpage1.html is your html file

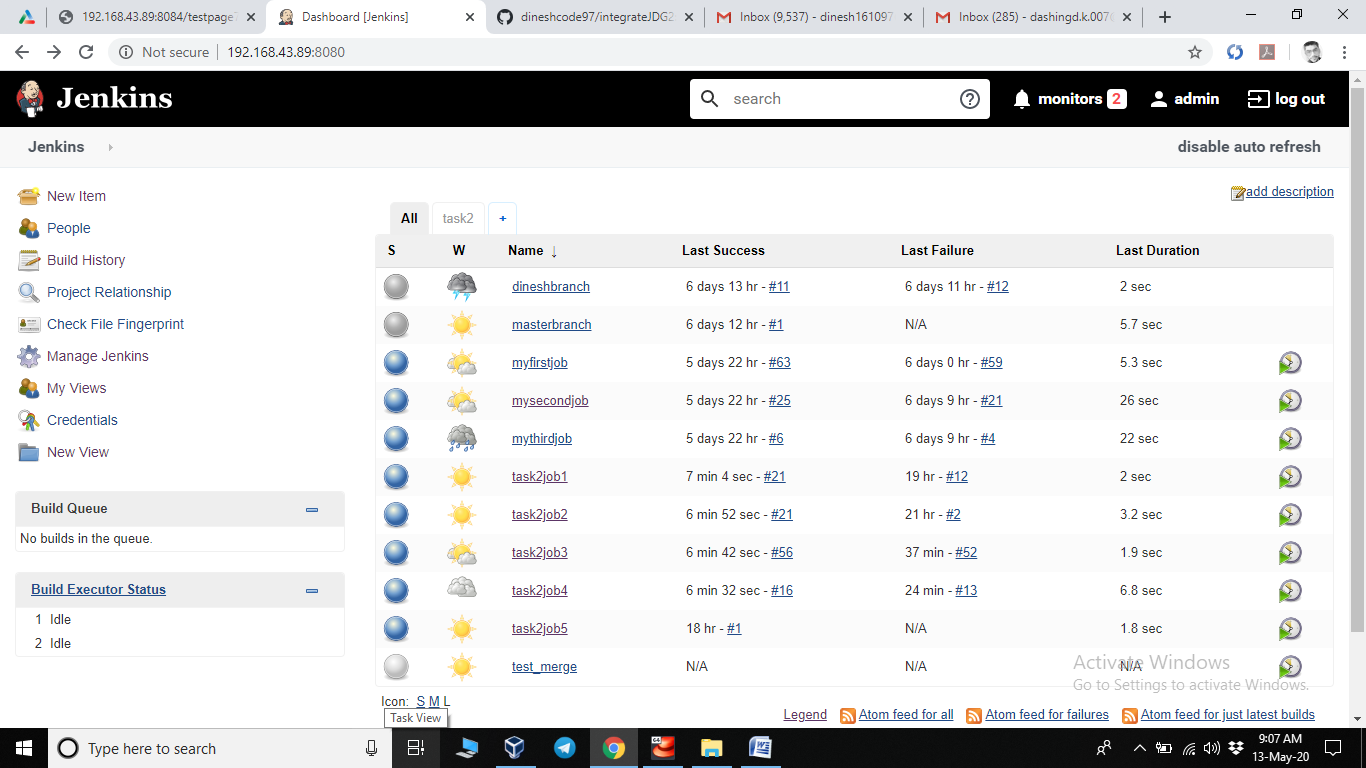
git add testpage1.html

git commit –m “first commit” testpage1.html

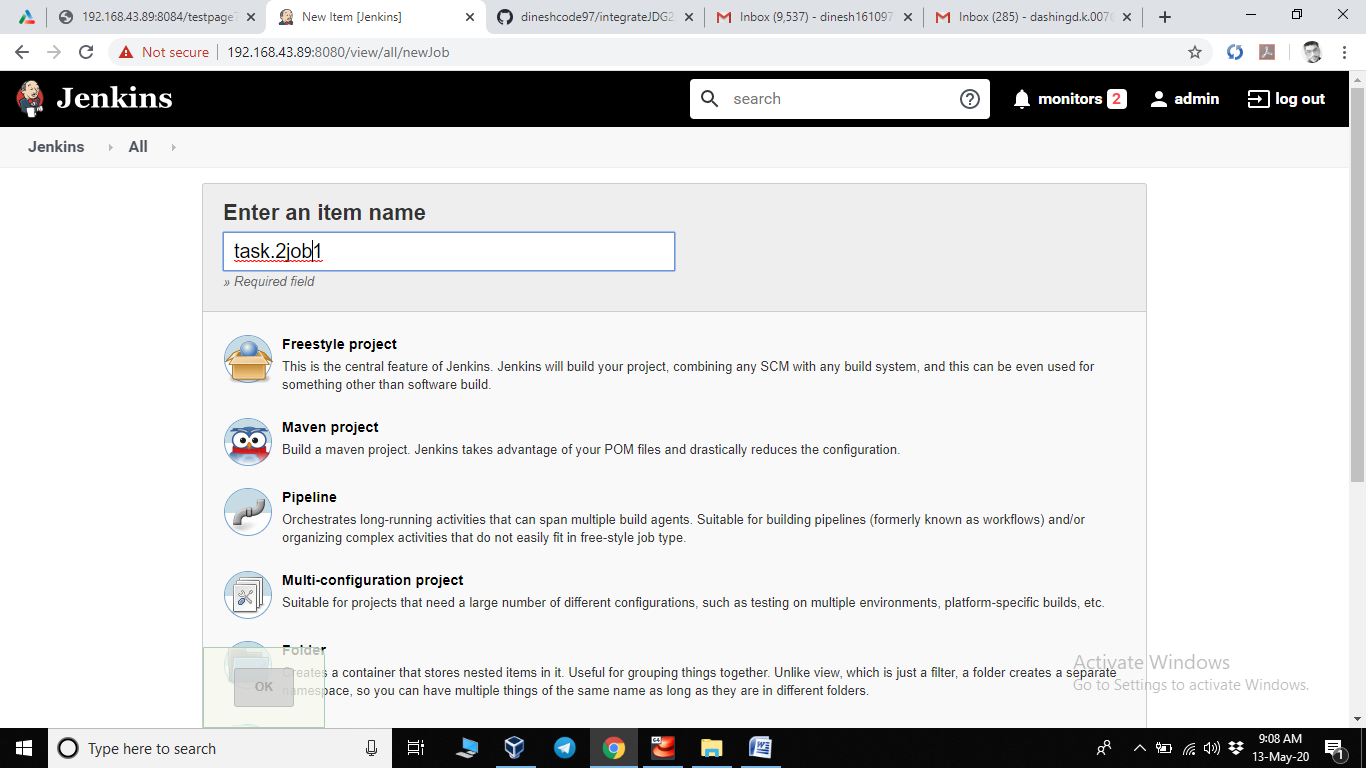
(local hook push this automatically)

#job 1:

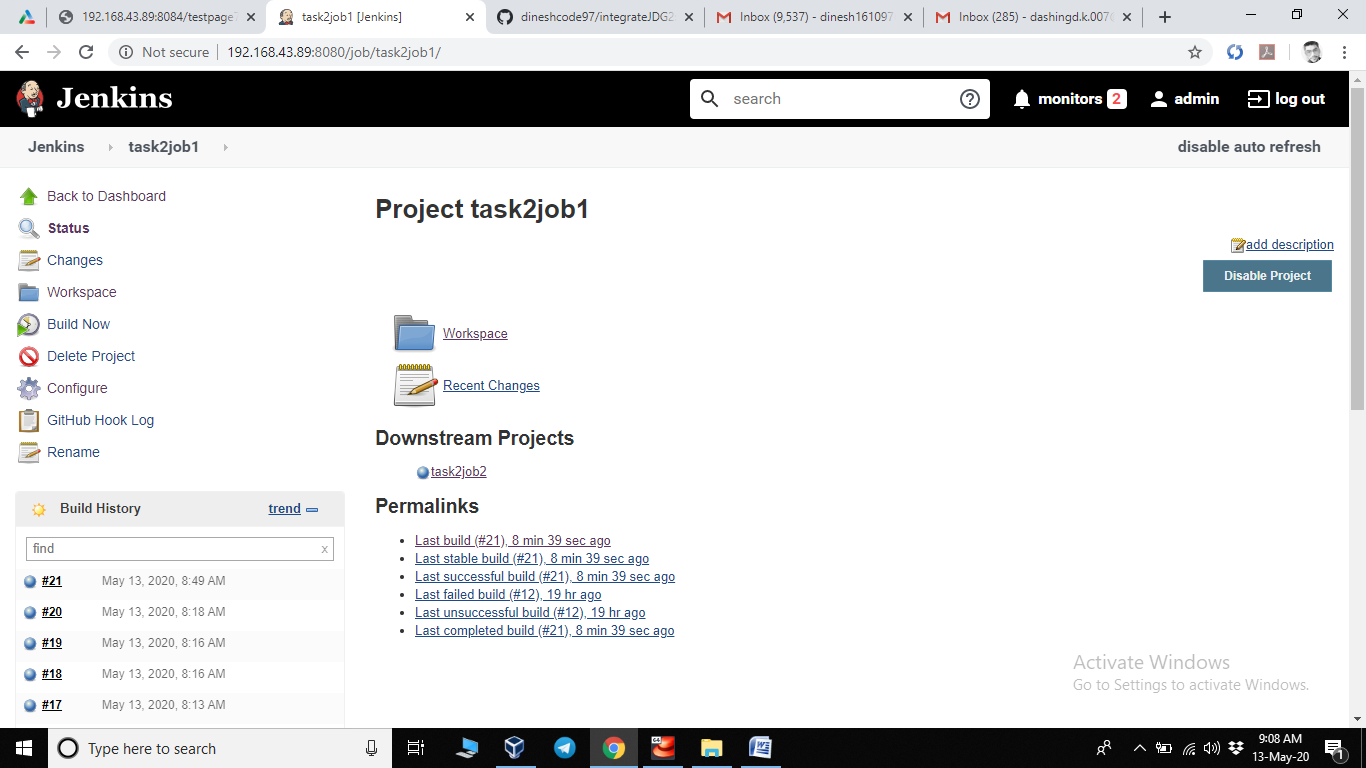
1. Open your Jenkins in browser and your username and password
2. Click new item item and create a job (select freestyle project in it)



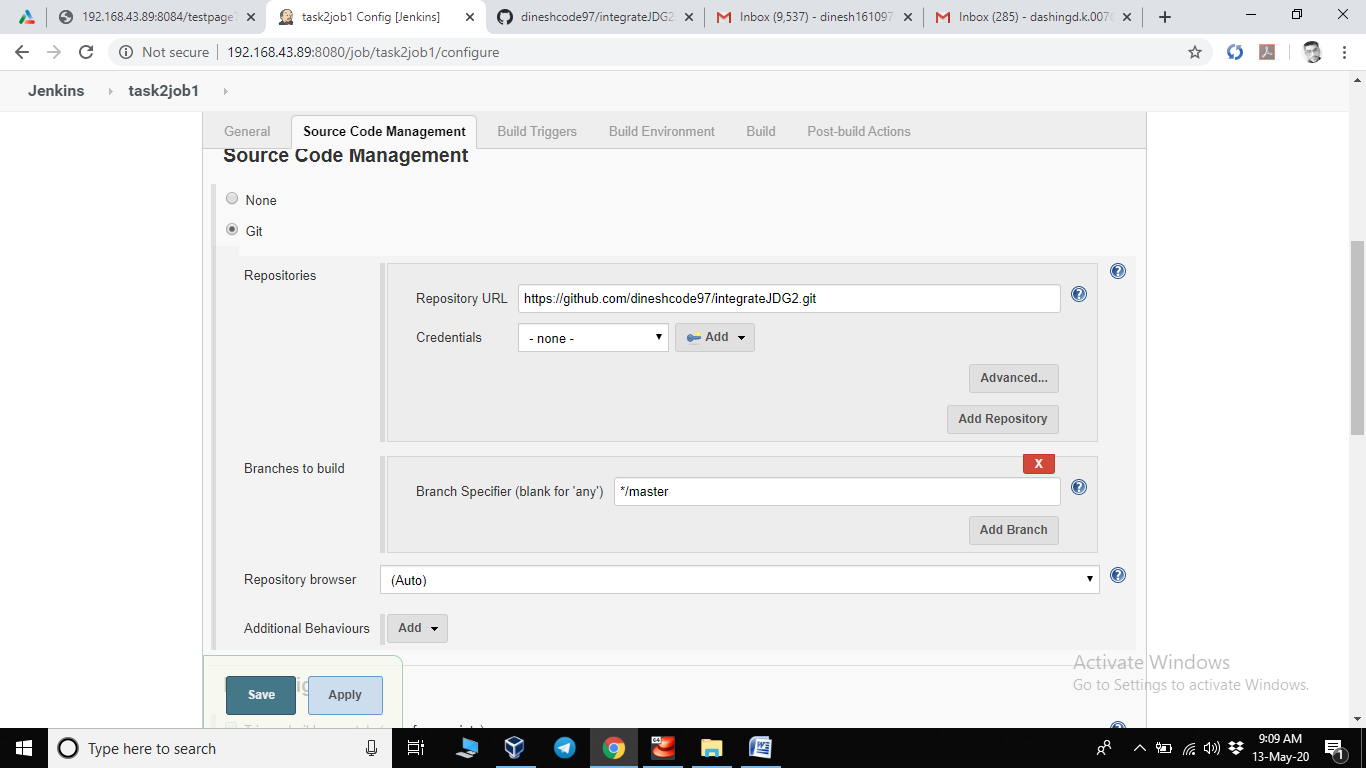
1. Click ok.



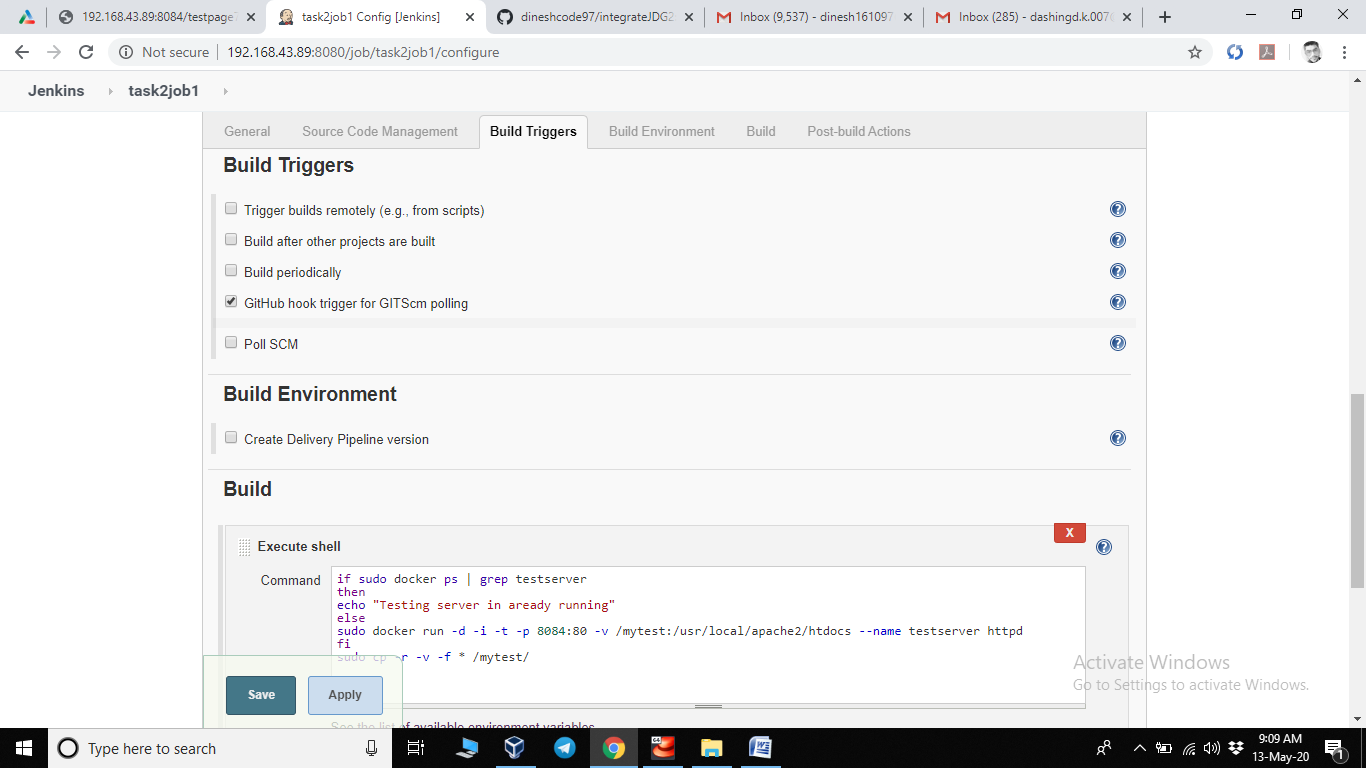
1. Click on configure option to configure the job as you want to do.



1. Select git in SCM category and Paste your github repository url in repository box.



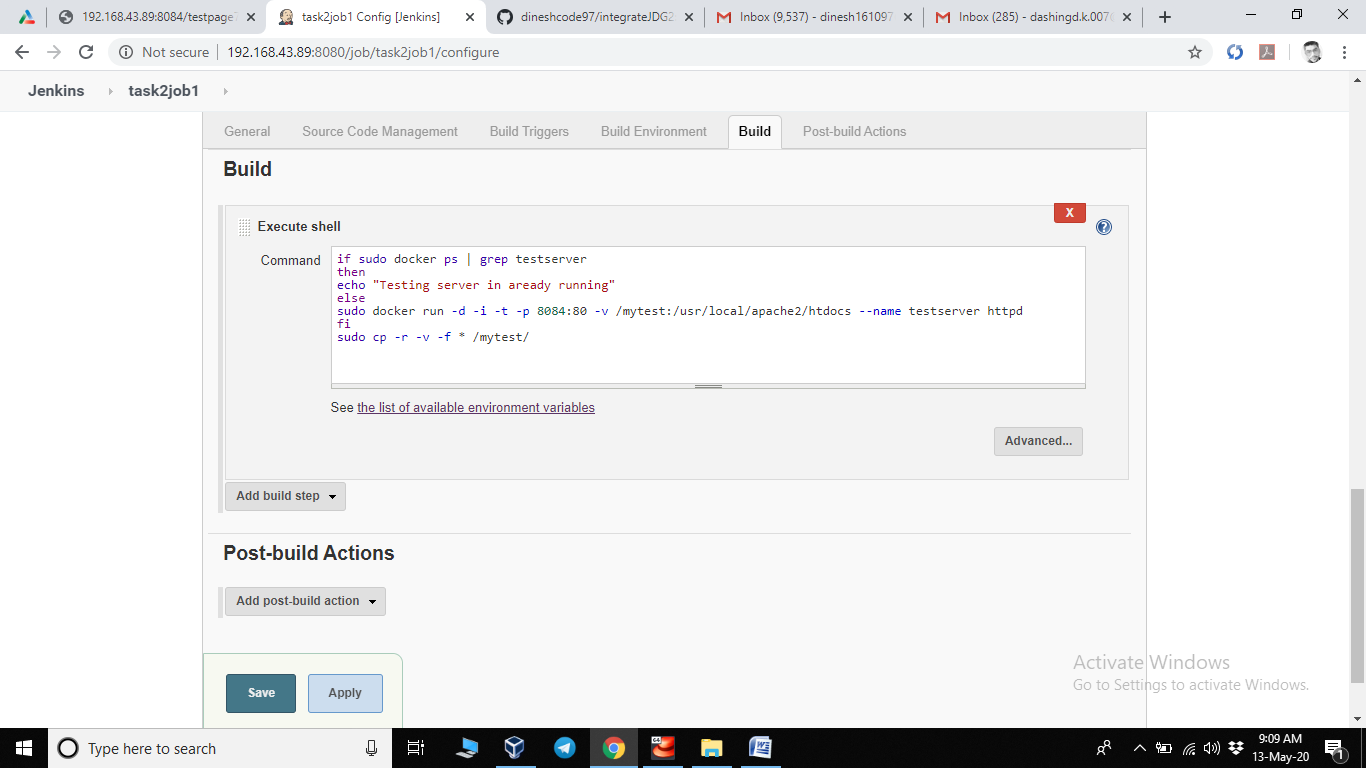
1. In build triggers select Github hook trigger for git Scm Pooling



1. In build option select execute shell and enter this code

Exp. If testserver container is running its show already running

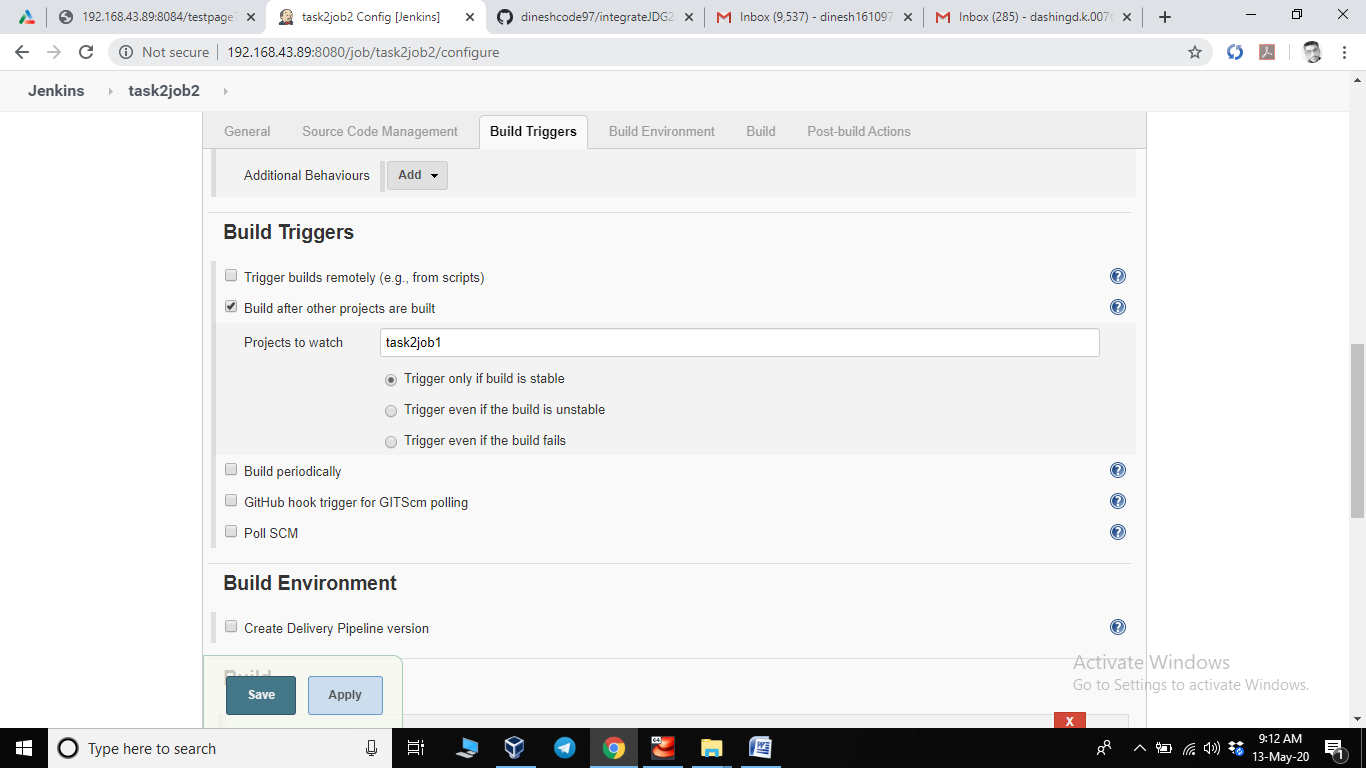
And if it is not running its start testserver container which is a httpd server



1. Save this job by clicking save.

#Job2:

1. Create job 2 as we create job and and goto configure then
2. Copy and paste github url same as job 1
3. In build triggers select build after project is build give name of job you want to run after this job(task2job2)
4. Select trigger only if build is stable



1. in build select execute shell
2. enter this commands

htmlfiles= sudo ls /mytest/ | grep .html | wc –l #check no. of html files

phpfiles= sudo ls /mytest/ | grep .php | wc –l #check no. of php files

if $phpfiles > 0 #if there is a php file in dir.

then

if sudo docker ps -a | grep phpsetup #checks phpsetup is running or not

then

echo "php setup already running" # if running it prints this message on console

else

sudo docker run -d -i -t --name phpsetup -p 8100:80 vimal13/apache-webserver-php # else it starts phpsetup i

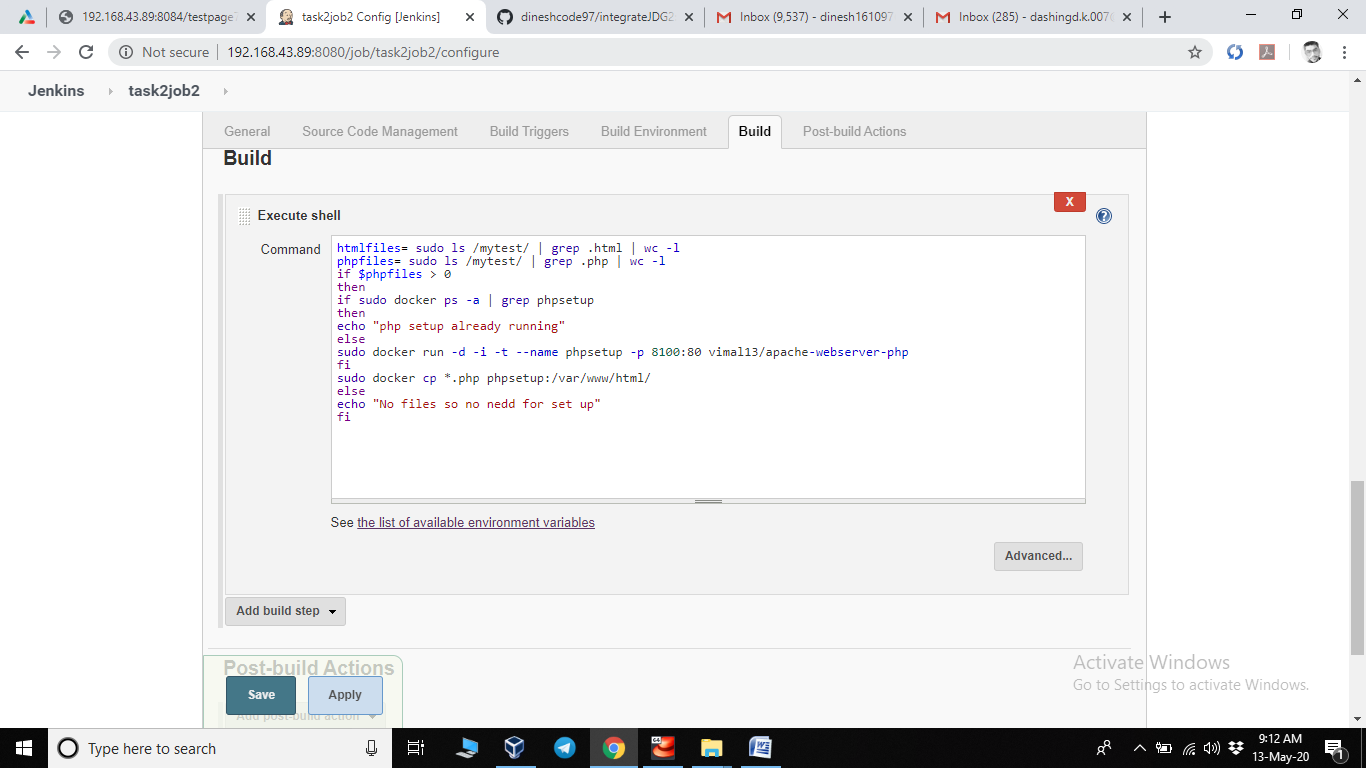
fi

sudo docker cp \*.php phpsetup:/var/www/html/ # copy files in php container

else

echo "No files so no nedd for set up" #if there is no php file it gives this message

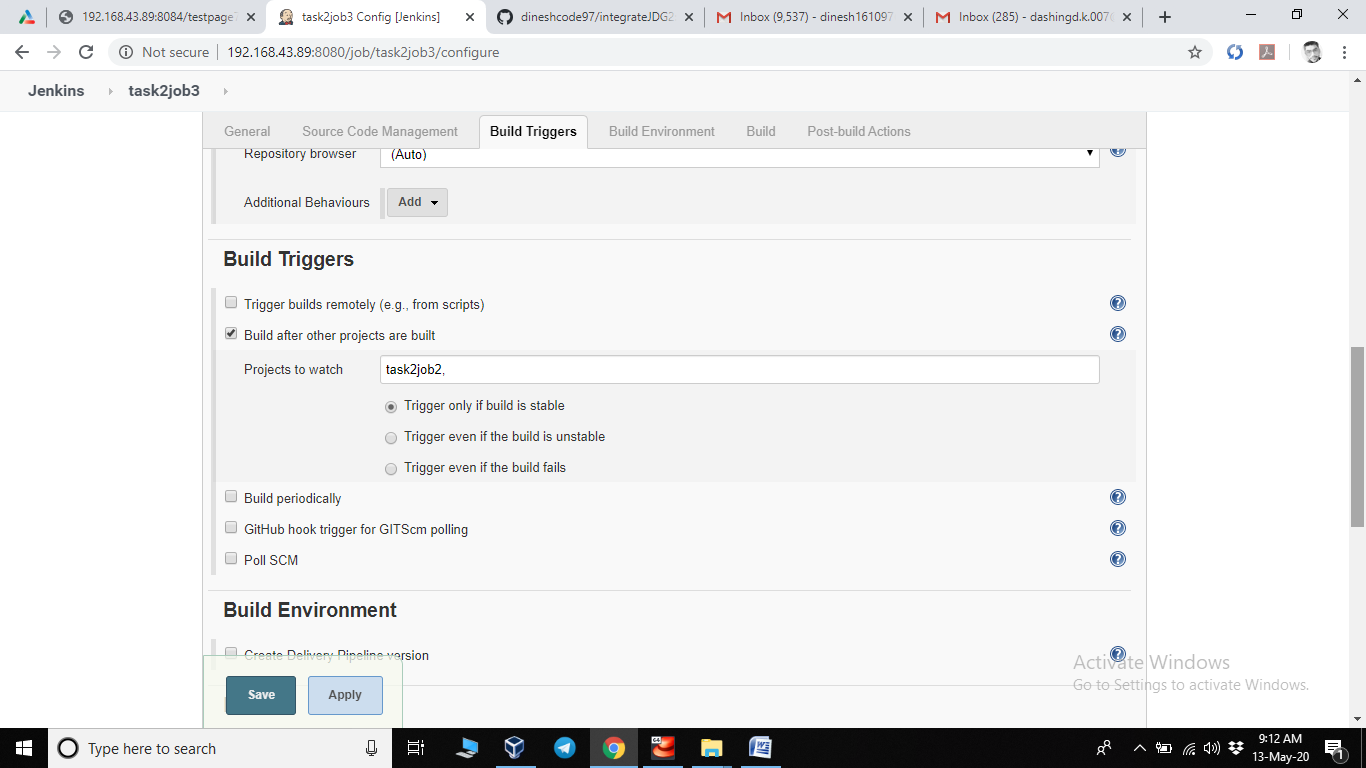
fi



1. click save to save the job

#Job3:

1. Create job 3 as we create job and and goto configure then
2. Copy and paste github url same as job 1
3. In build triggers select build after project is build give name of job you want to run after this job(task2job2)



1. In build select execute shell and enter this commands(code)

allhtmlfiles=$(sudo ls /mytest/\*.html) #assign all html files of dir in variable allhtmlfiles

allphpfiles=$(sudo ls /mytest/\*.php) #assign all php files of dir of variable allphpfiles

for file in $allhtmlfiles; do #tranverse each file

status=$(curl -o /dev/null -s -w "%{http\_code}" 192.168.43.89:8084/$file) #assign file http status to variable status

if [ $status == 200 ] ; then #if http status is 200 file is ok

echo "OK"

else # if it is not 200 there is some error in the file

echo "Error"

fi

done

for file in $allphpfiles; do #tranverse each file

status=$(curl -o /dev/null -s -w "%{http\_code}" 192.168.43.89:8100/$file) #assign file http status to variable status

if [ $status == 200 ] ; then #if http status is 200 file is ok

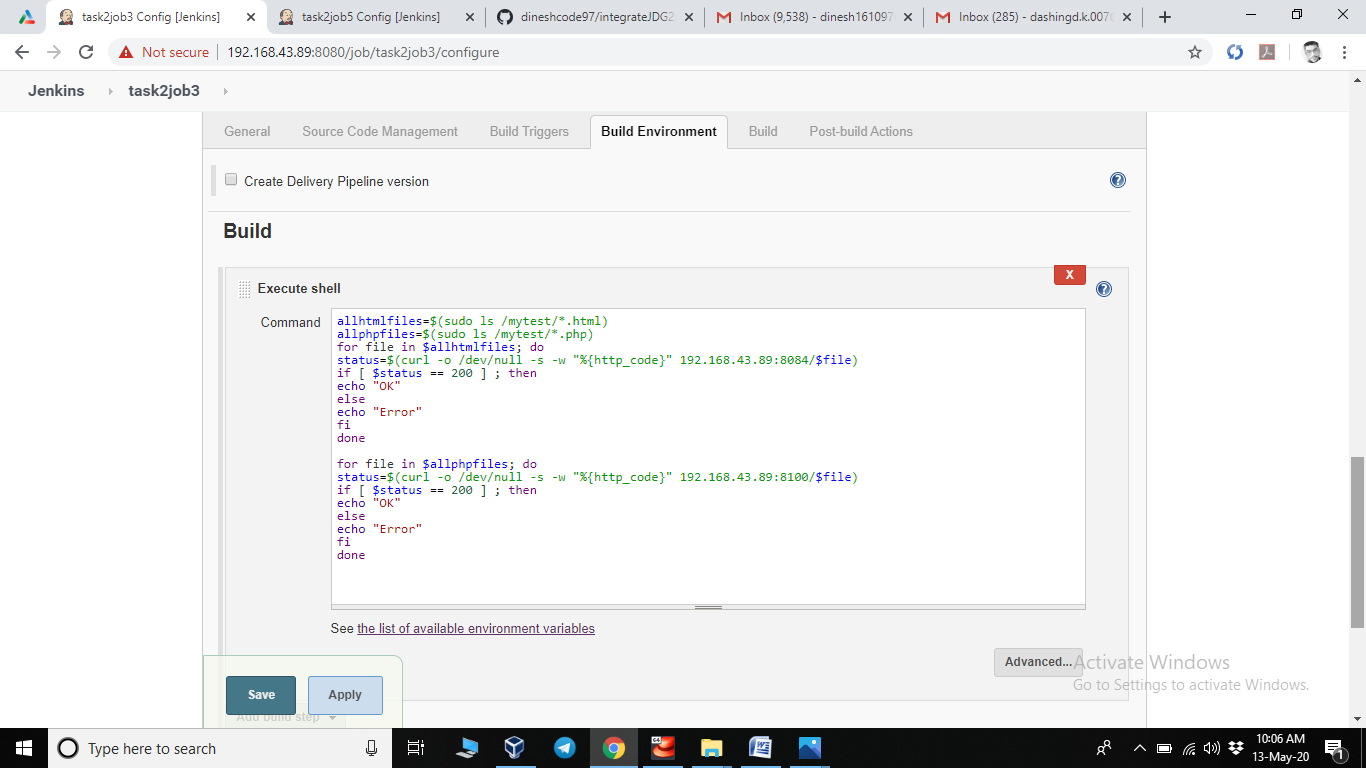
echo "OK"

else # if it is not 200 there is some error in the file

echo "Error"

fi

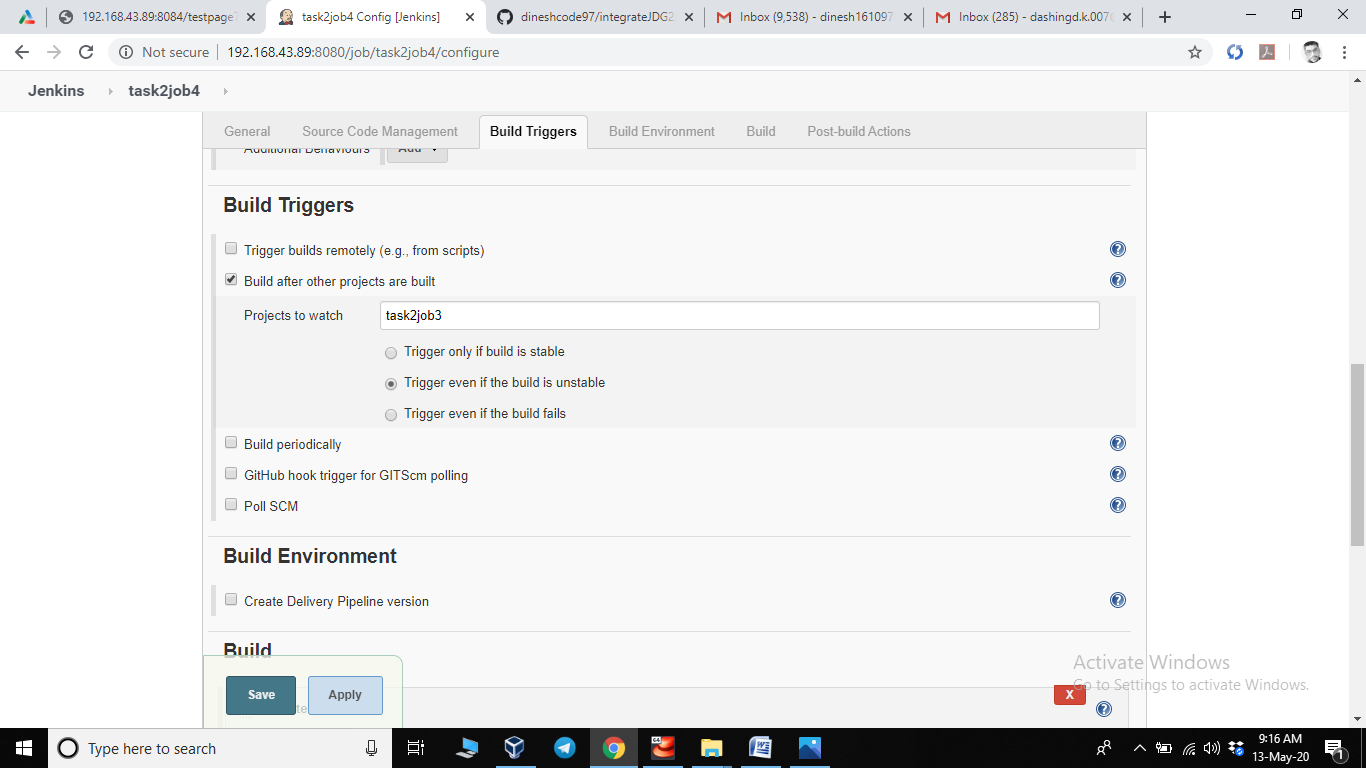
done



1. Save the job by clicking on save

#Job4:

1. Create job 4 as we create job and and goto configure then
2. Copy and paste github url same as job 1
3. In build triggers select build after project is build give name of job you want to run after this job(task2job4)



1. In build select execute shell and enter this commands(code)

sendmail=$? #get last error code from job 3 and assign to sendmail(variable)

if [ $sendmail == 1 ] # if sendmail equals to 1 sends mail to autority

then

sudo python3 /pty/sendmail.py

else

echo "Wroks fine" # if 0 works fine

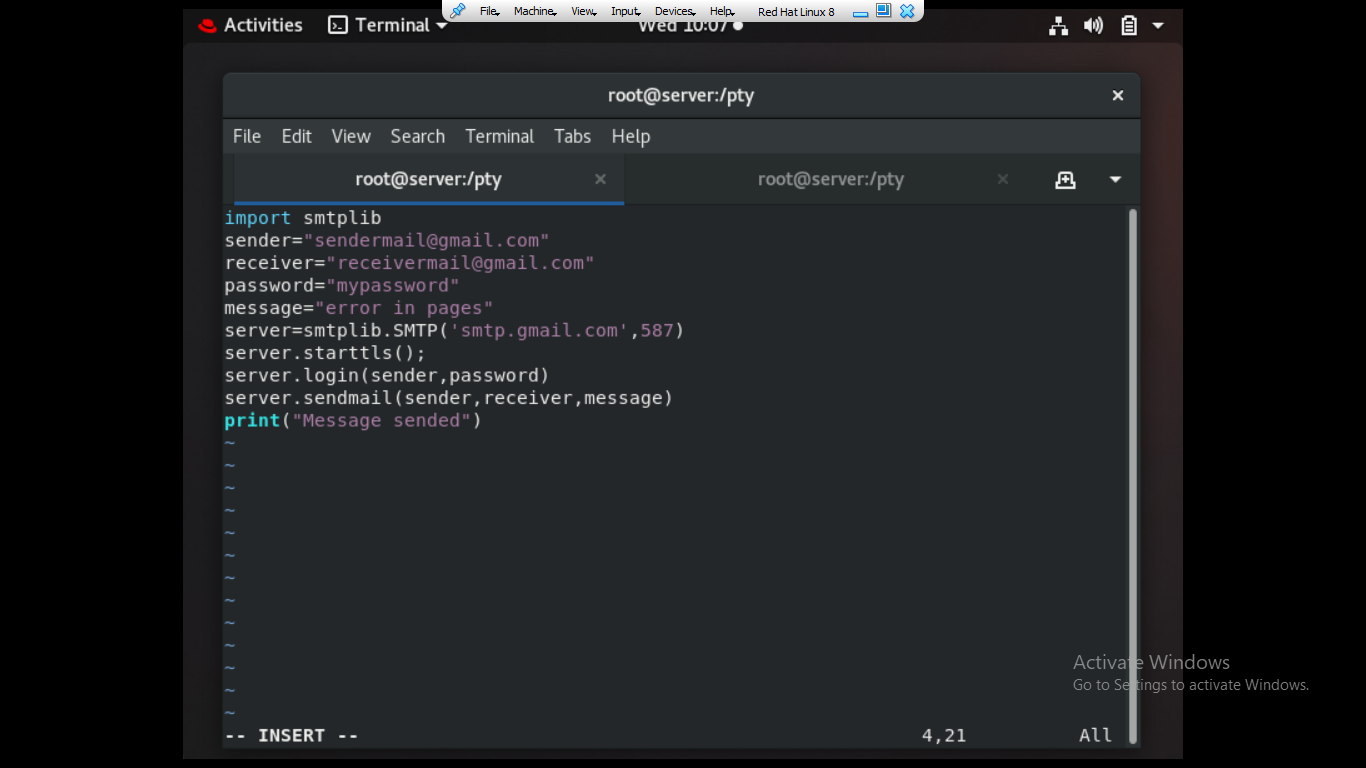
fi

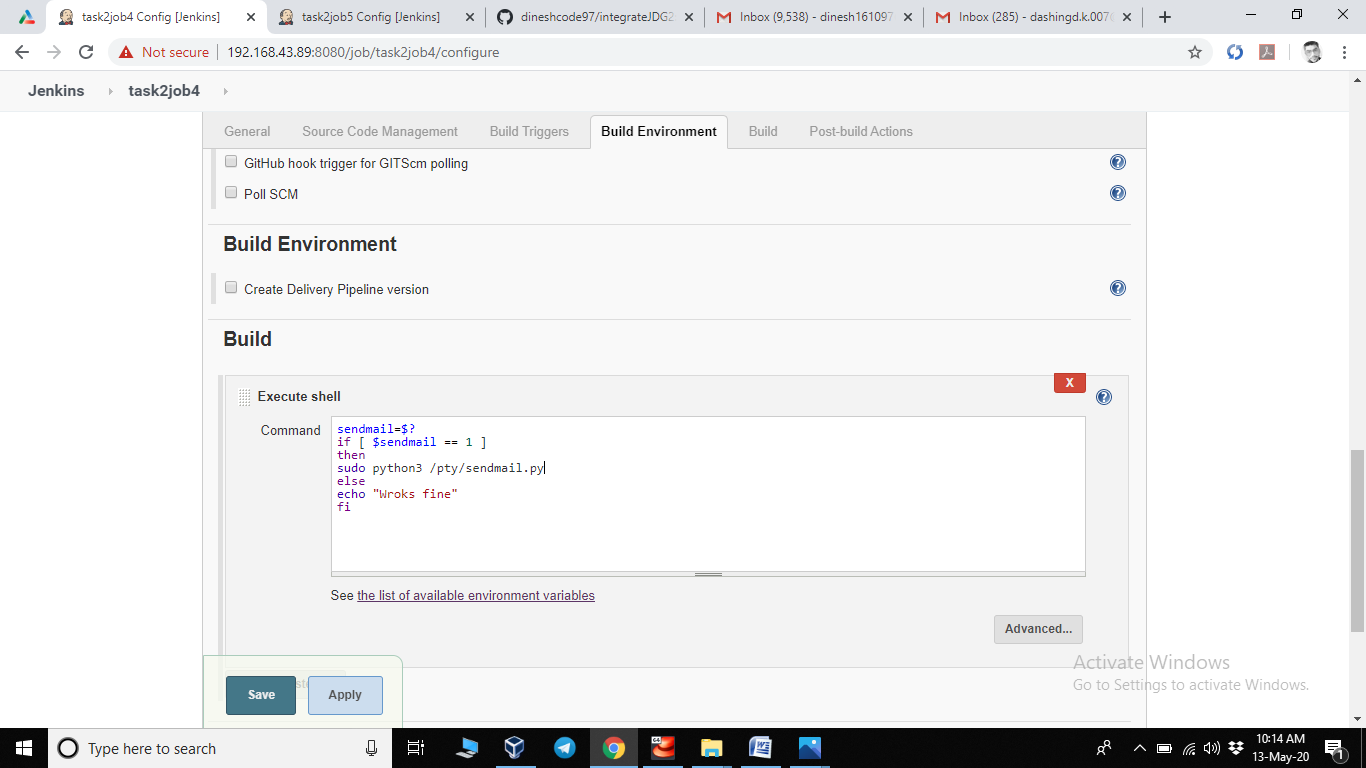
1. Save by click on save

Image or screen shot on next page

Note:===make a script using python to send mail to authority

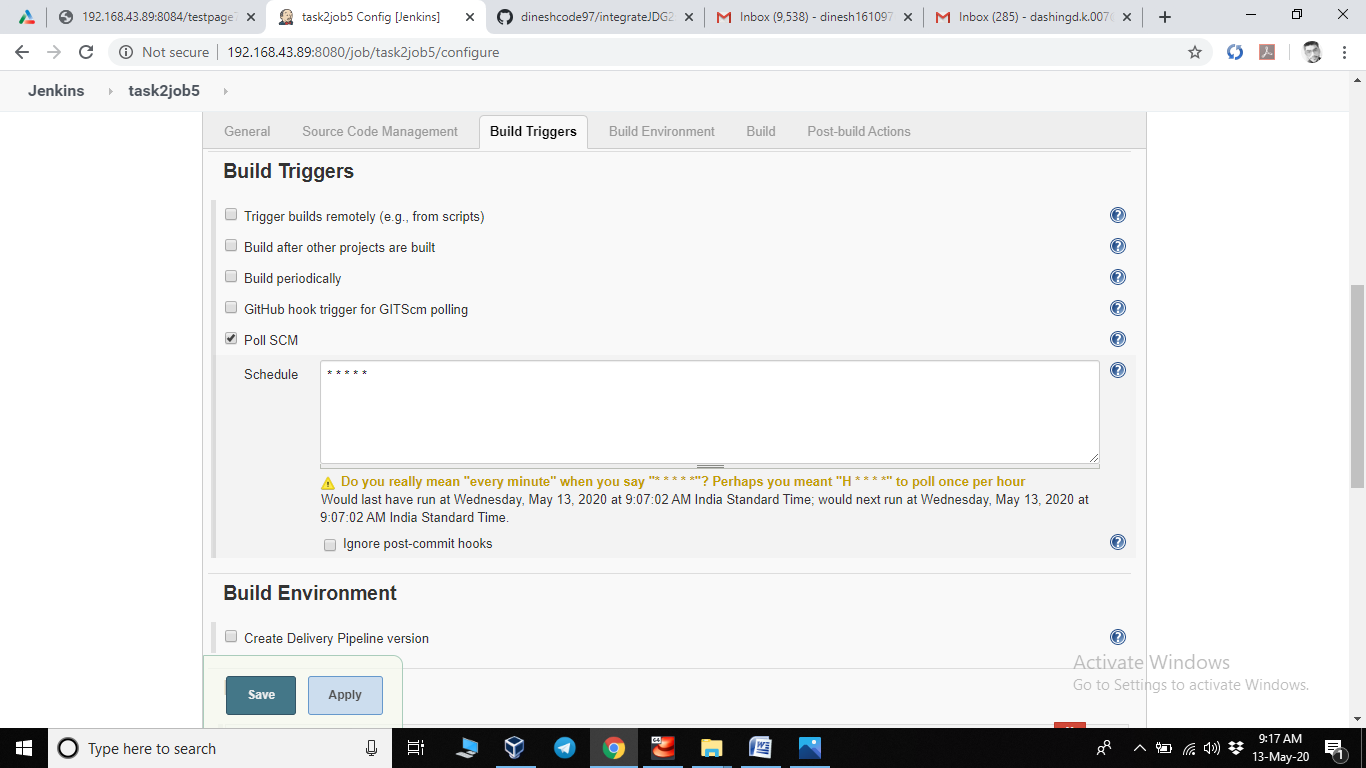
1. import smtplib # module to send mail
2. sender =”xyz@gmail.com” #gives your or sender mail
3. receiver= “abc@gmail.com” #gives receiver or authority mail
4. password= “senderpass” #gives password of sender mail
5. message= “error” #message you want to send
6. server=smtplib.SMTP(“smtp.gmail.com”,587) #use to congiure mail server
7. server.starttls() # to start server
8. server.login(sender,password) #login as sender
9. server.sendmail(sender,receiver,message) #send mail from sender to receiver
10. print(“message sended”)





#Job5:

1. Create job 5 as we create job and and goto configure then
2. Copy and paste github url same as job 1
3. In build trigger select poll scm and give \* \* \* \* \*



1. In build select execute shell

if sudo docker ps -a | grep phpsetup #check every minute if phpsetup is running or not I run ok

then

echo "no need"

else #else it will start php setup

sudo docker run -d -i -t --name phpsetup -p 8100:80 vimal13/apache-webserver-php

fi

if sudo docker ps -a | grep testserver #check httpd server is running if running ok

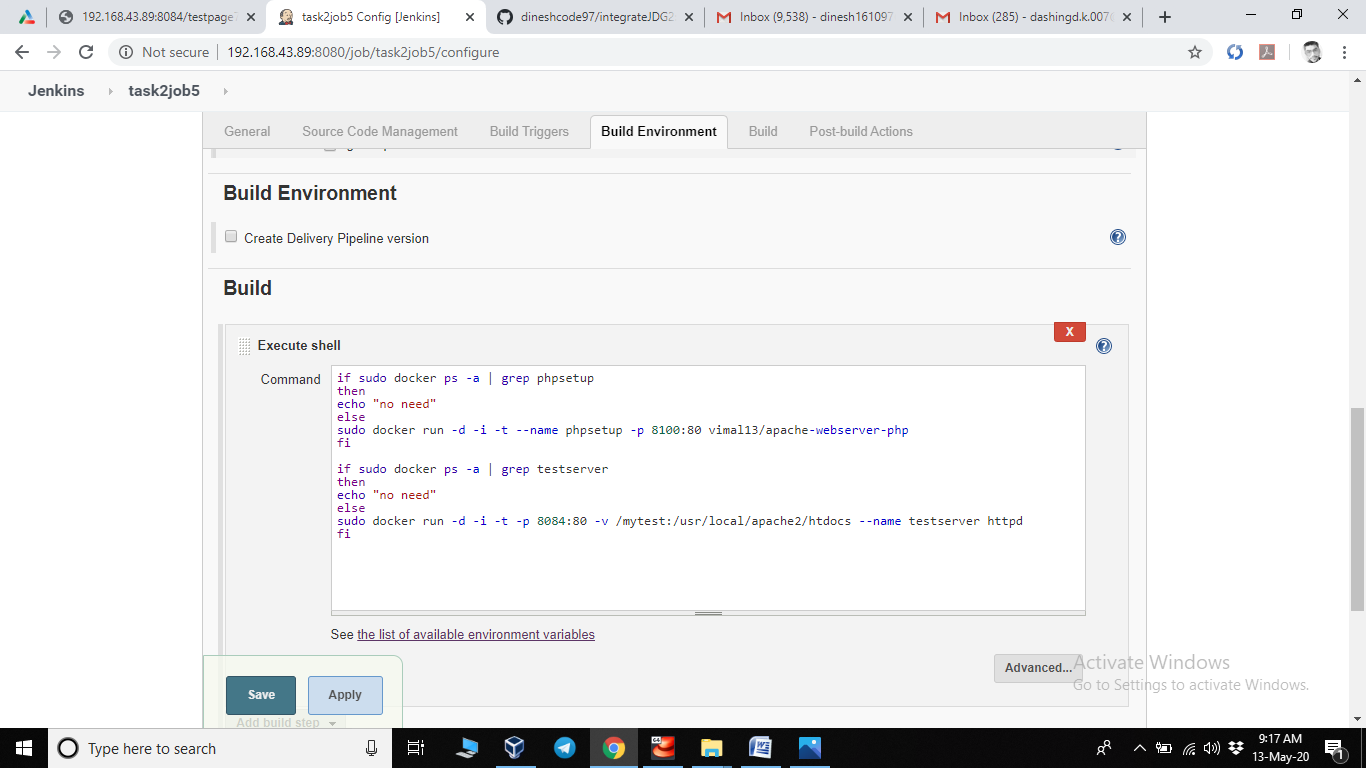
then

echo "no need"

else #else run httpd server

sudo docker run -d -i -t -p 8084:80 -v /mytest:/usr/local/apache2/htdocs --name testserver httpd

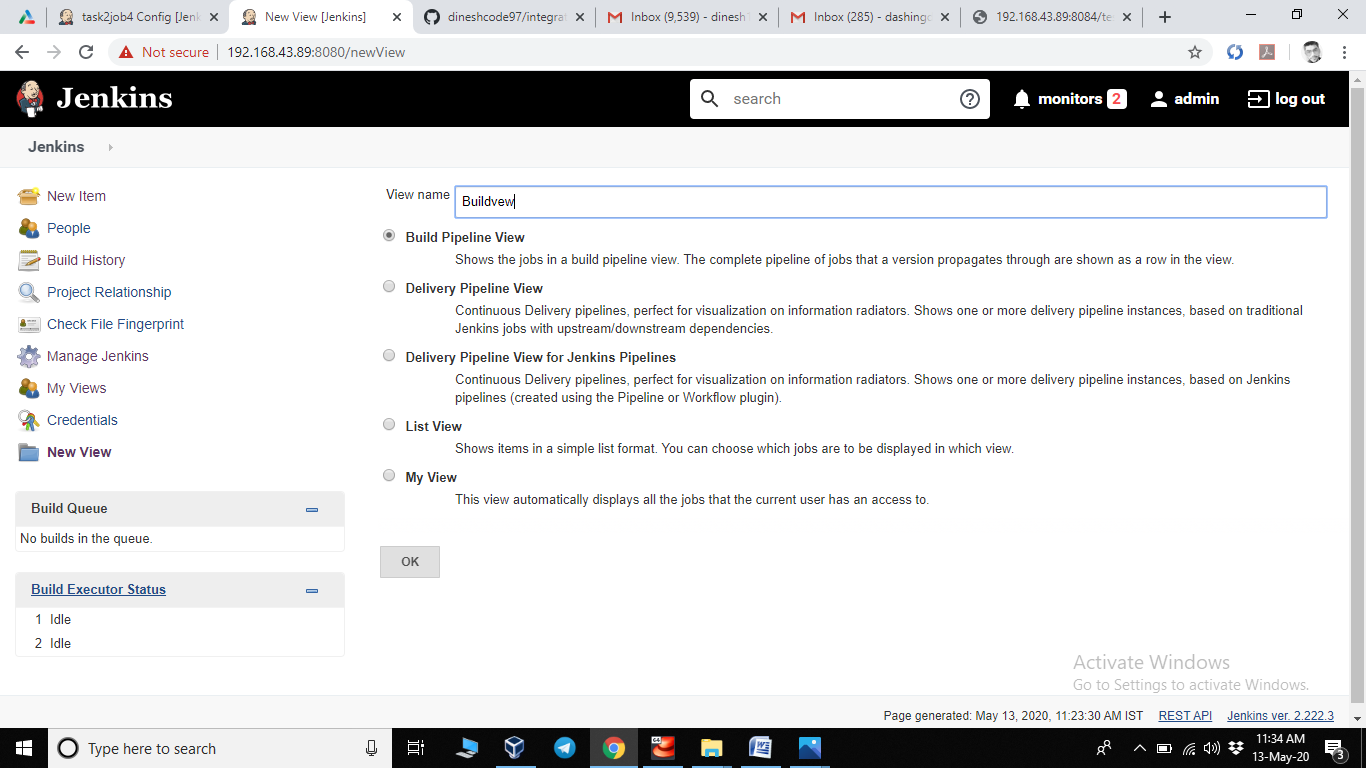
fi



Build Pipeline:

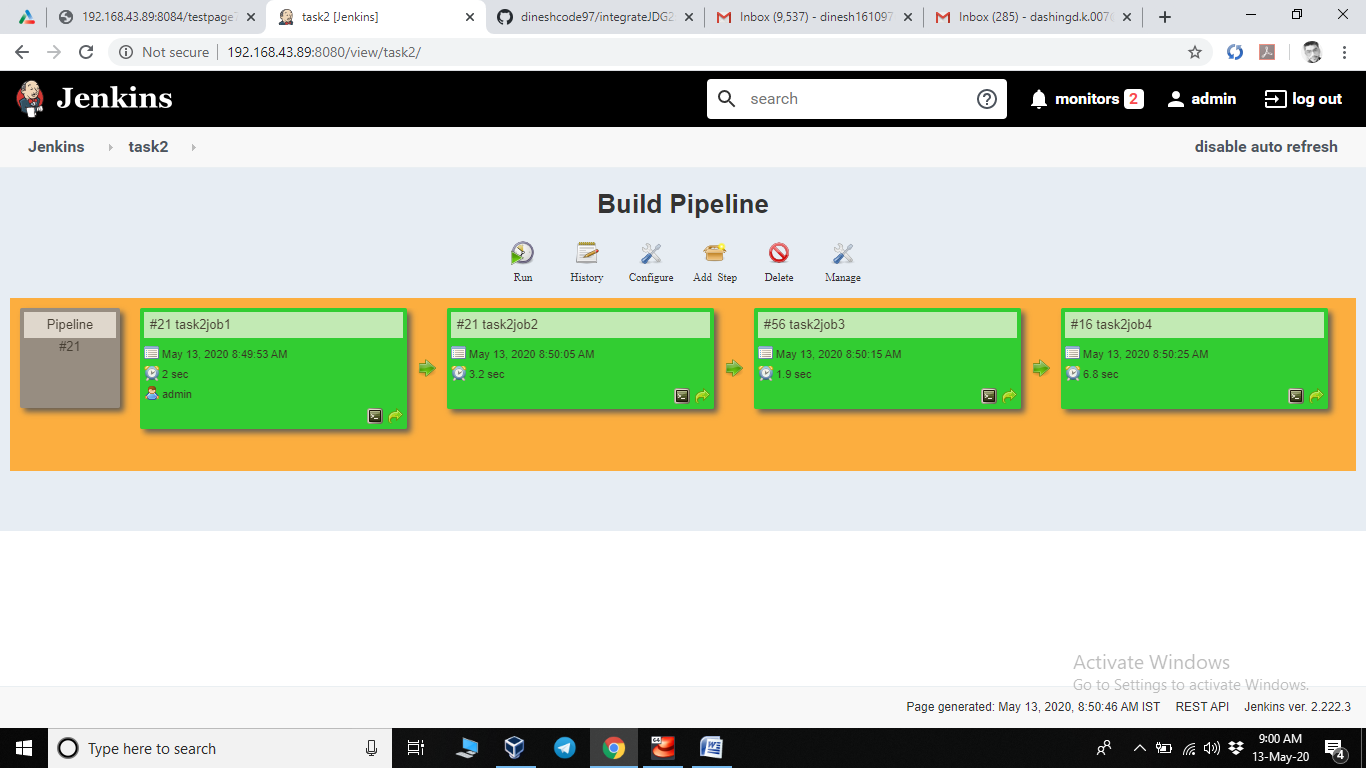
1.Goto new view select build pipeline view

2. give name to view



1. Click ok and in next screen by going down select initial job(task2job)
2. Click ok

In dashboard ypu can see your view is cread go to that view and run its run all job which are pipelined



Note : my job 5 is not depend on any job it every minute and checks both server are running or not