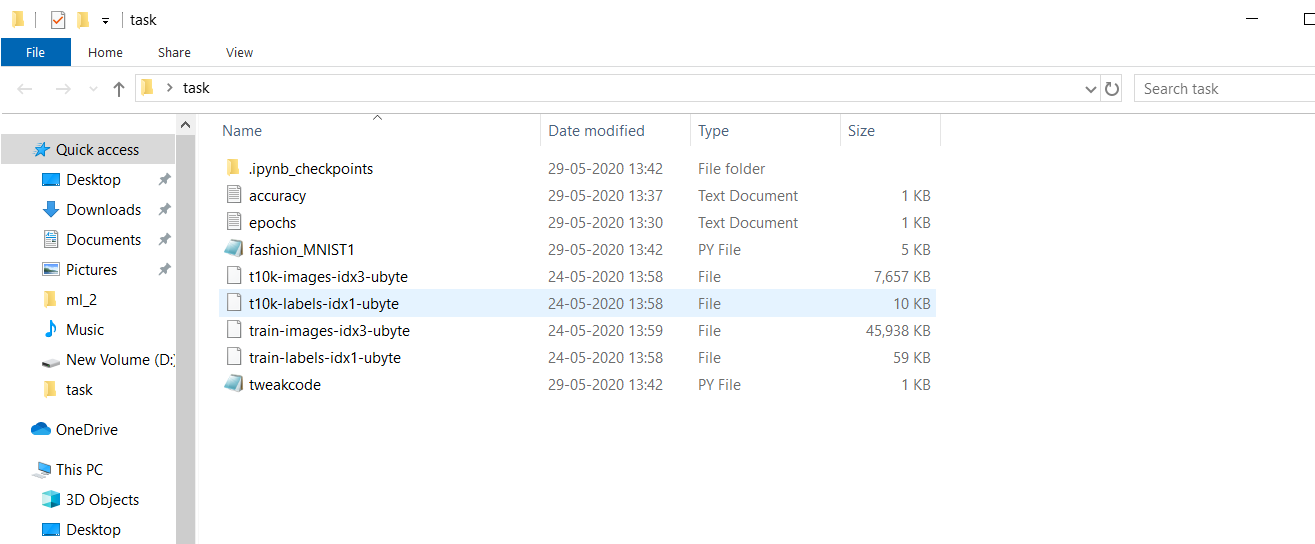
**PROJECT TO INTEGRATE ML(MACHINE LEARNING) ,**

**GIT , GITHUB , JENKINS , DOCKER….**

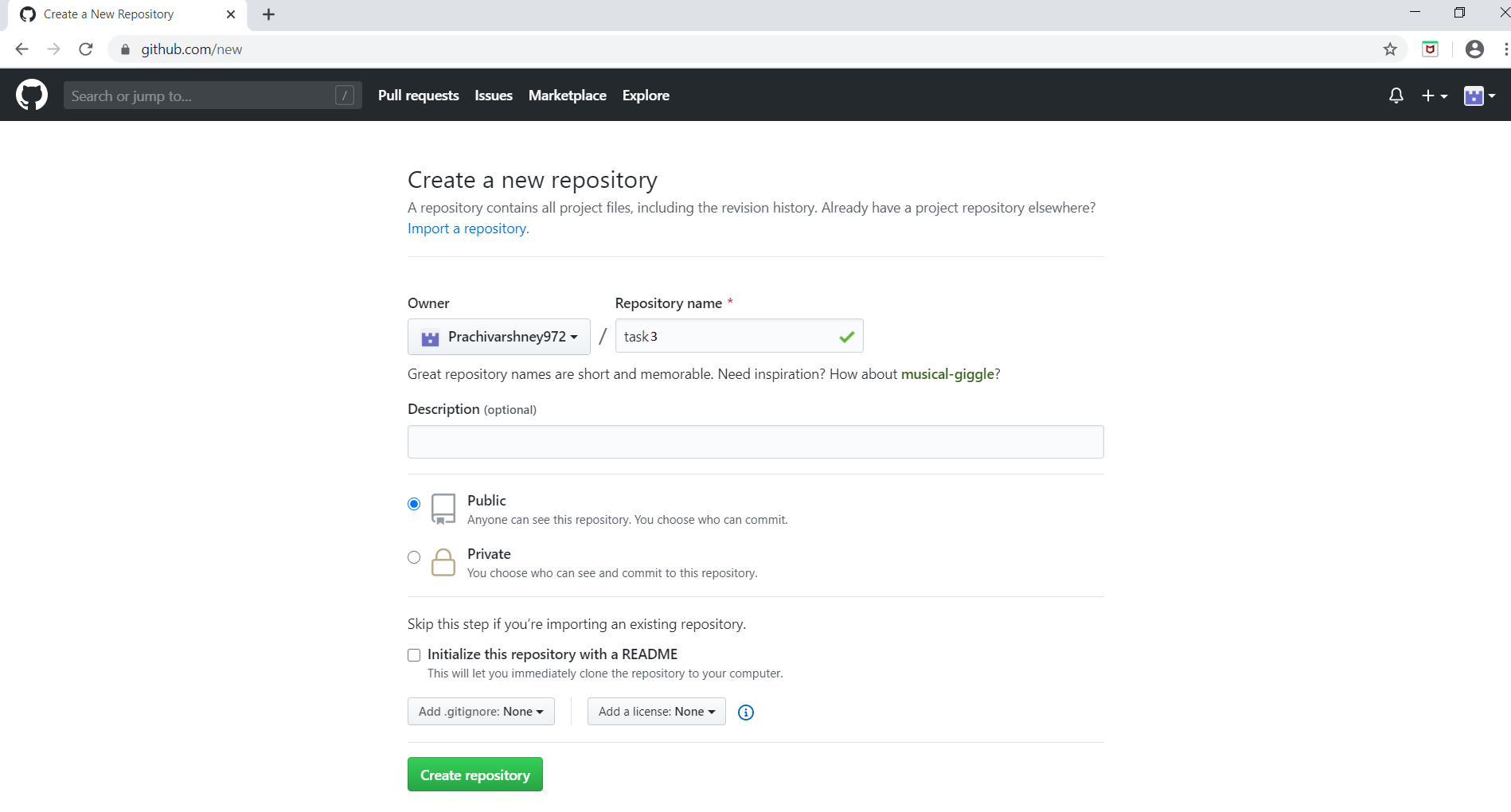
**Let’s start our project**………

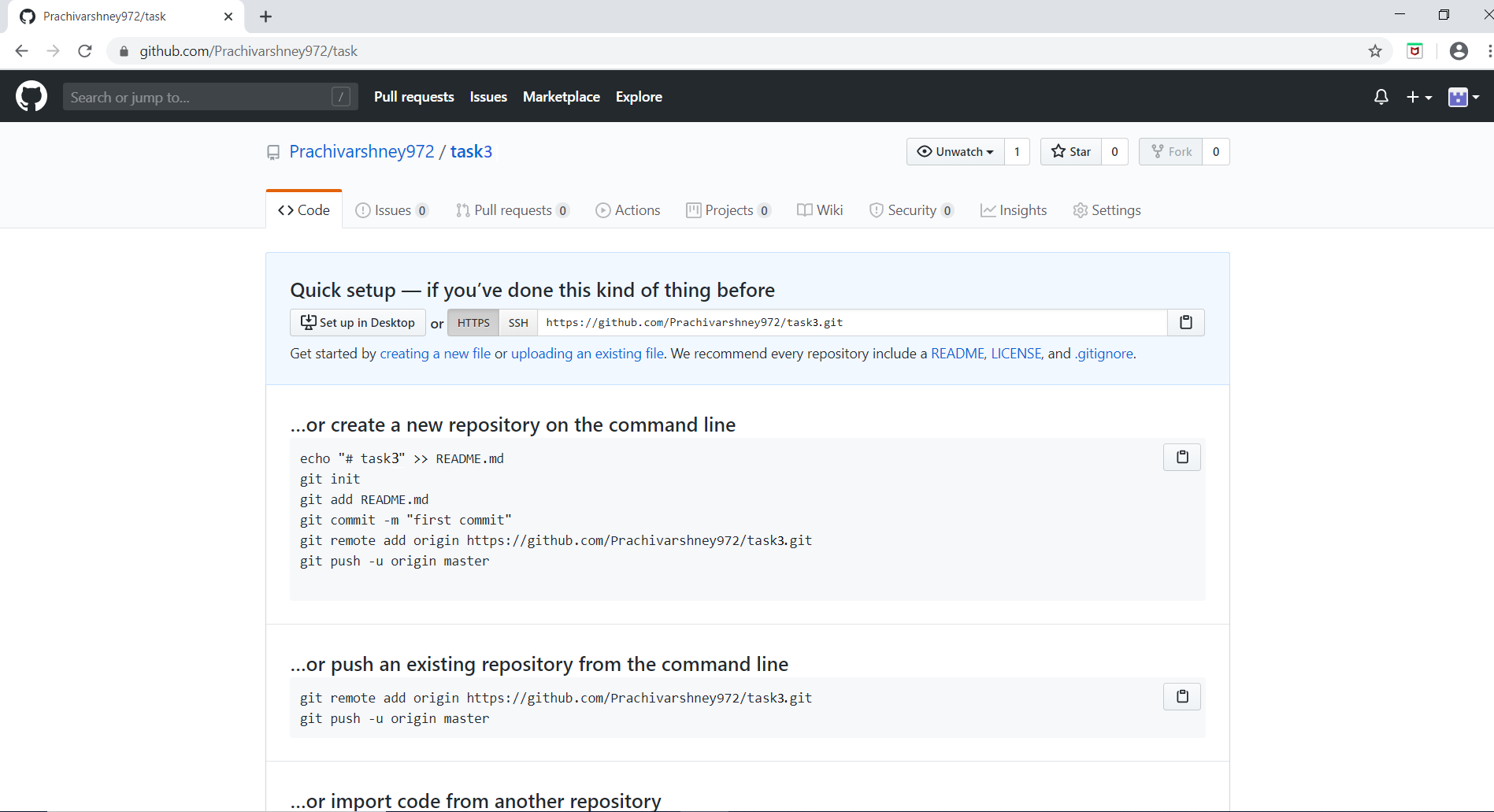
First , make a folder in your system where is your ml model (code) and datasets for training and testing present …..

And go inside the folder…by using “GIT BASH” and push these files on “GITHUB” using “GIT BASH”…

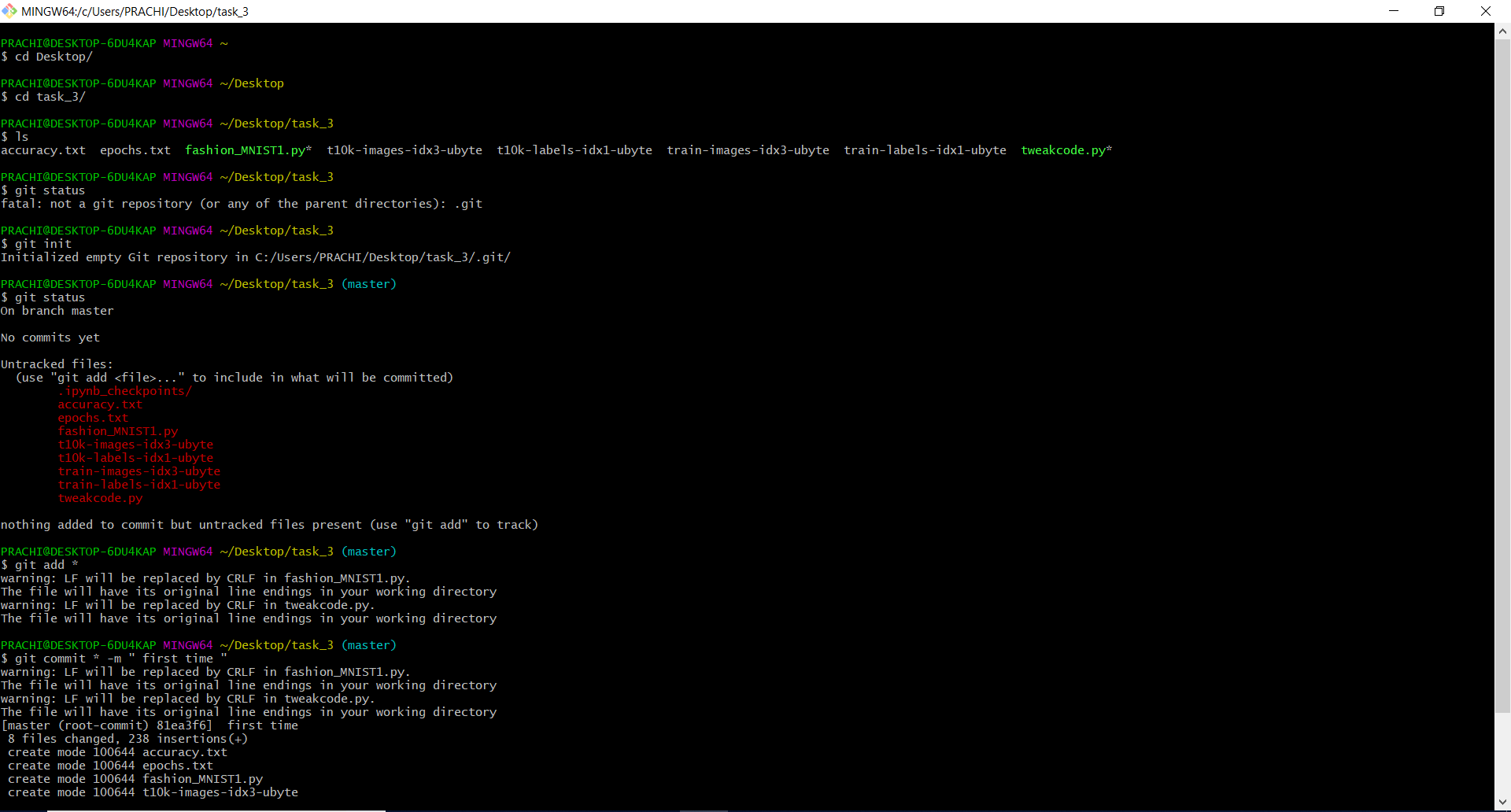


Create new repo on github …..and it should look like this :





Now using “GIT BASH” , push the files on this repo ,

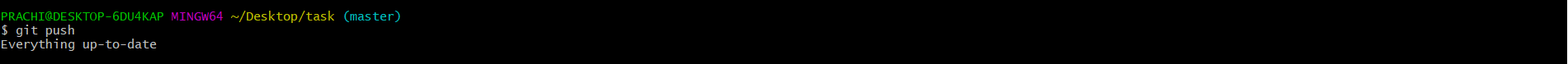


I have already push this code on github ….so push command shows….everything up to date….

But on first time pushing to a newly created repository :

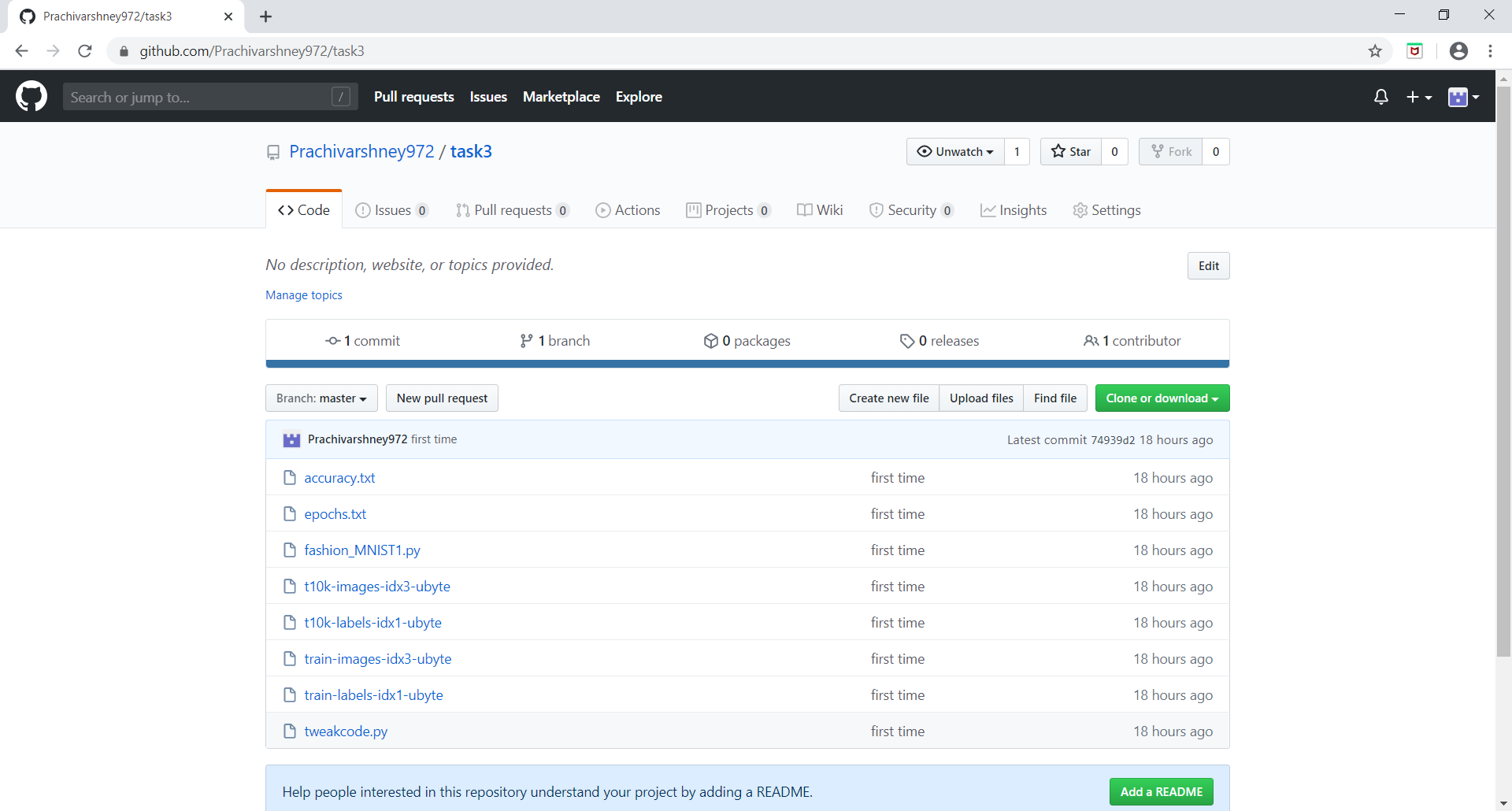
Two commands you have to use :

* git remote add origin “github\_repo link”
* git push –set-upstream origin master..



After pushing your files on your particular repo:

This type of frame shown to you:



Here fashion\_MNIST1.py is my “ML” code……



And these four are my training and testing datasets…..

**So WHAT My aim is :**

**I WANT;**

As soon as my files are push to the github …

Jenkins start a “job” to download the code from github and after that another Jenkins “job” start to launch a environment to interpret my “ML” code and there is 3rd job also …..

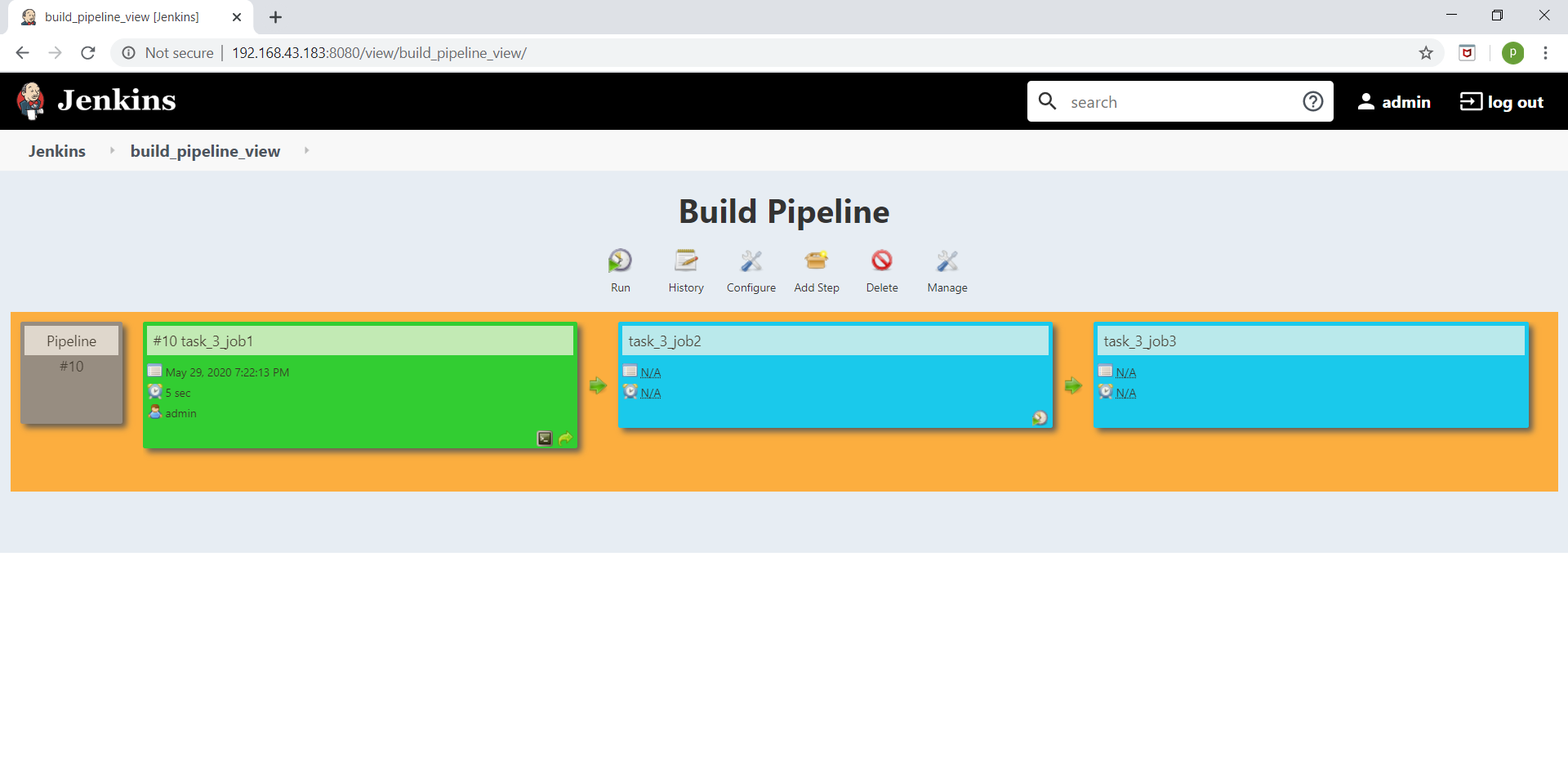
Which checks the accuracy and if the accuracy of model (code) is less than 91% , it increase the no of epochs until……..model accuracy reached to 91 % ……

As soon as accuracy received is > 91% , mail is send to the developer that your demanding accuracy is achieved……..

To link these jobs , chaining is done :

And you can see graphical view of chaining using build pipeline….(plugin)

Something like this ….



BRIEF DESCRIPTION , WHAT “JOB1” DO ….

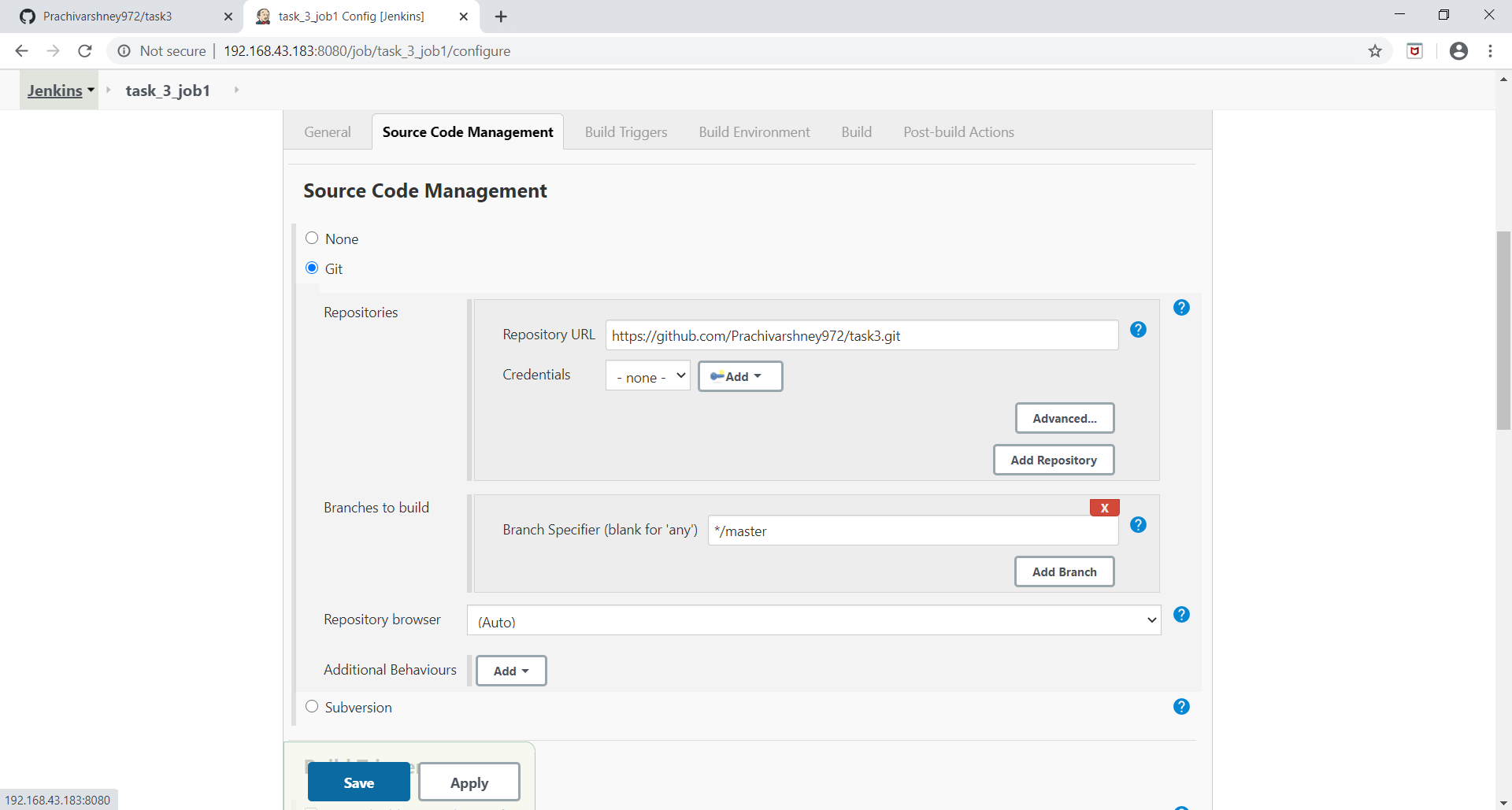
[“JOB1” is created to download the code from github ]

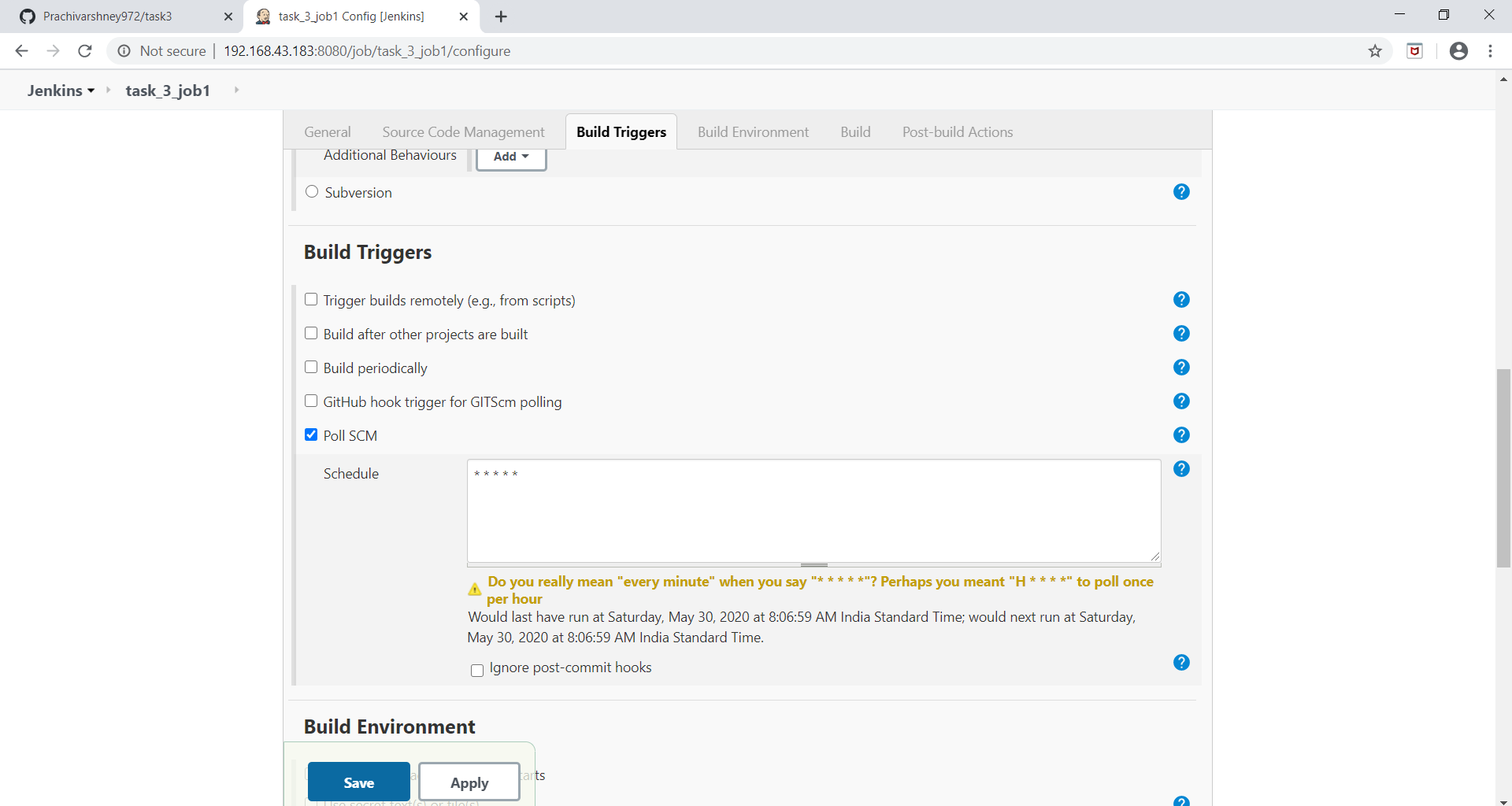
As soon as developer push the code on github…..

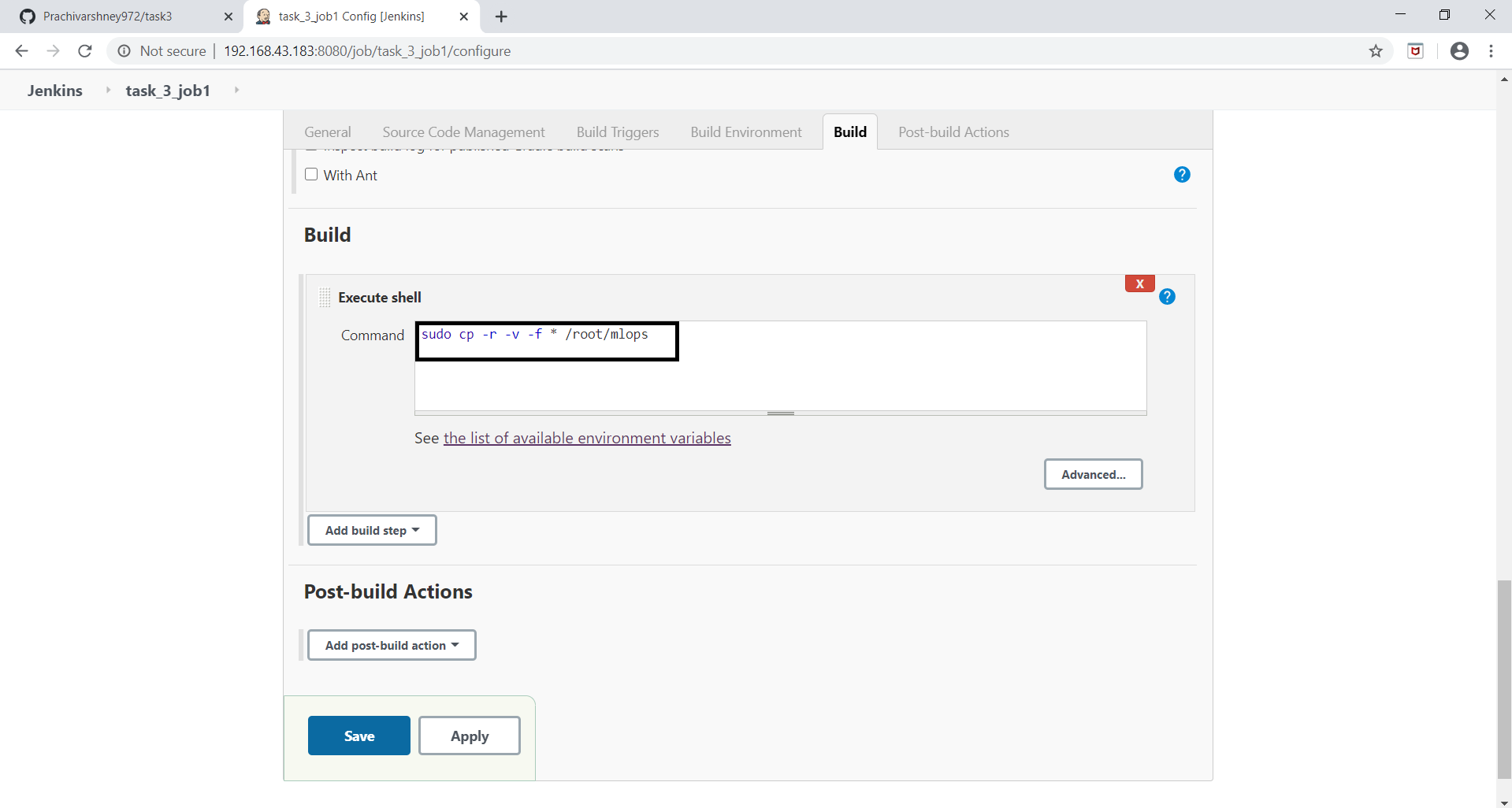
This job go the github , in every minute , if Jenkins feel there are changes on repo or something new added on , it download that change code / download the new code.

And after this , copy the code to folder inside the server (RHEL8) on which docker is running.

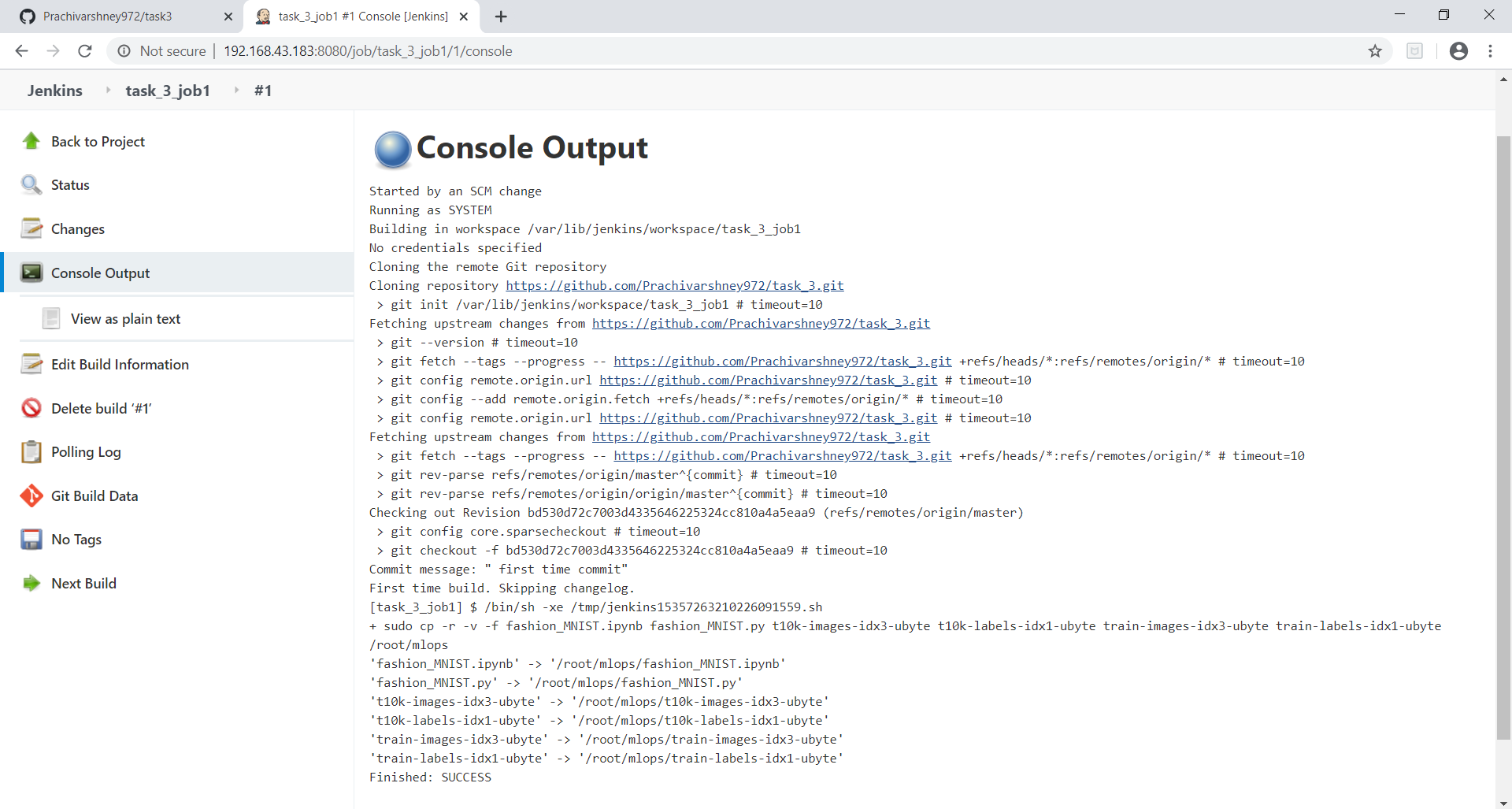
**“CONFIGURATION OF JOB1”**





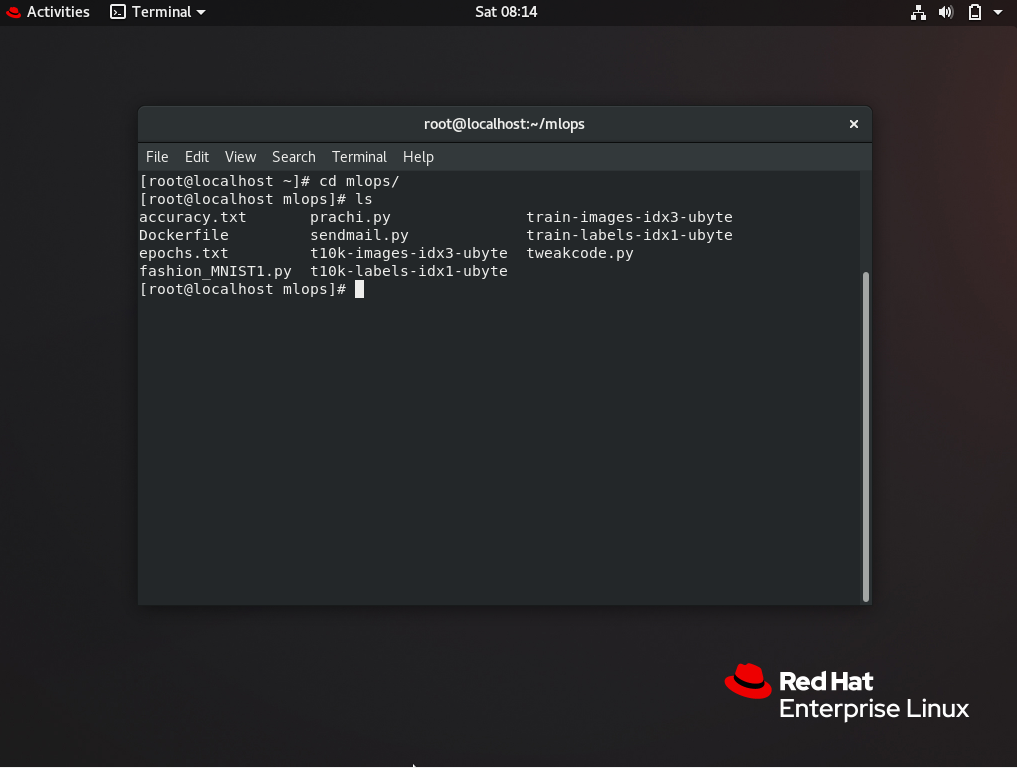


And output of this job…. shown in below figure.



After job 1 runs ,

We can see all files on RHEL8 ( because of copy command we use in “EXECUTE SHELL”)

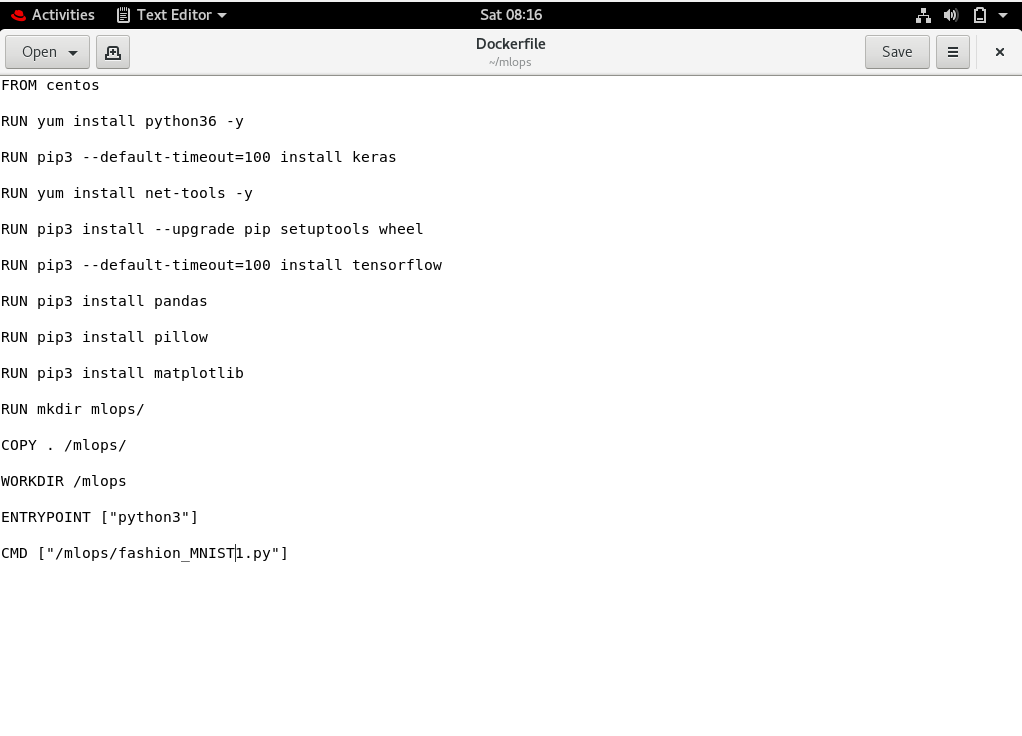


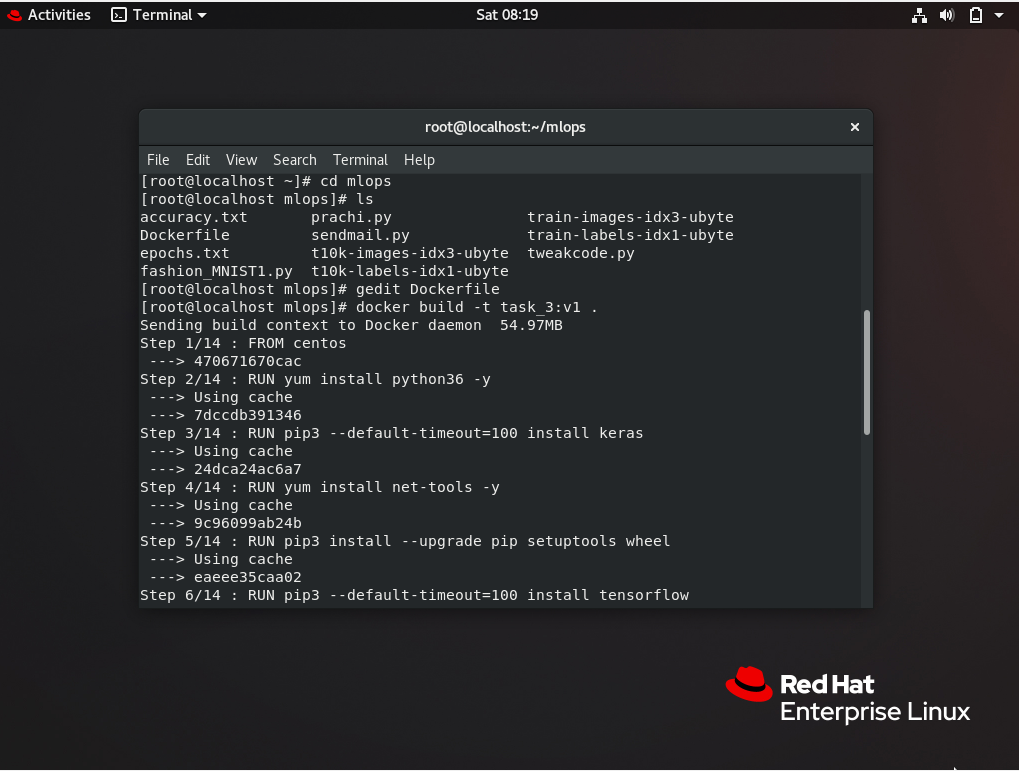
Now “JOB2” starts ….and its work is to launch the container (environment ) to run ML code which is present inside the “fashion\_MNIST1.py”.

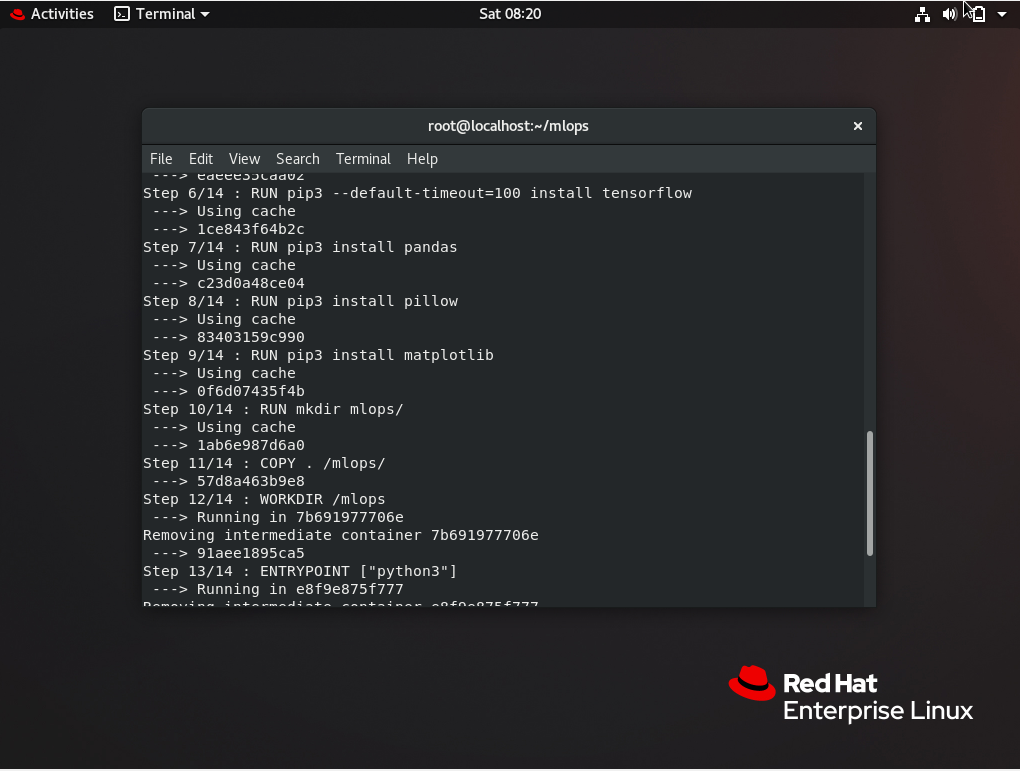
And before this , to launch the container ,

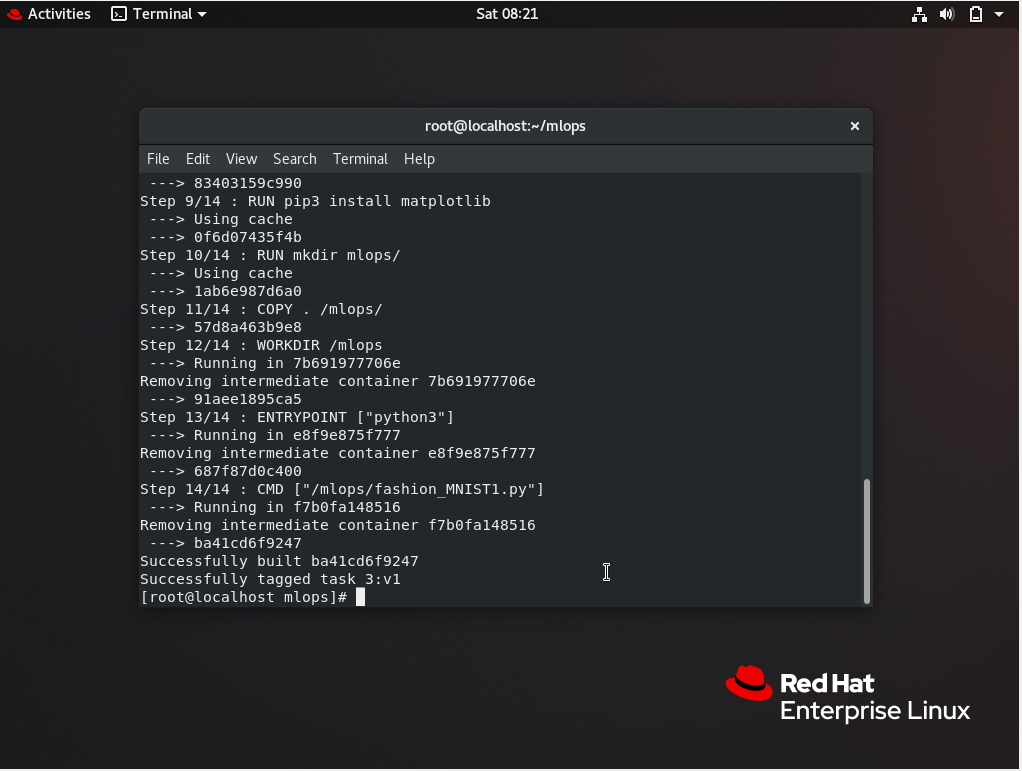
a dockerfile is made in mlops folder on RHEL8 and this command **“docker build -t task\_3:v1 . “** is used to create container image , then only Jenkins job is able to launch the container …

**DOCKERFILE**

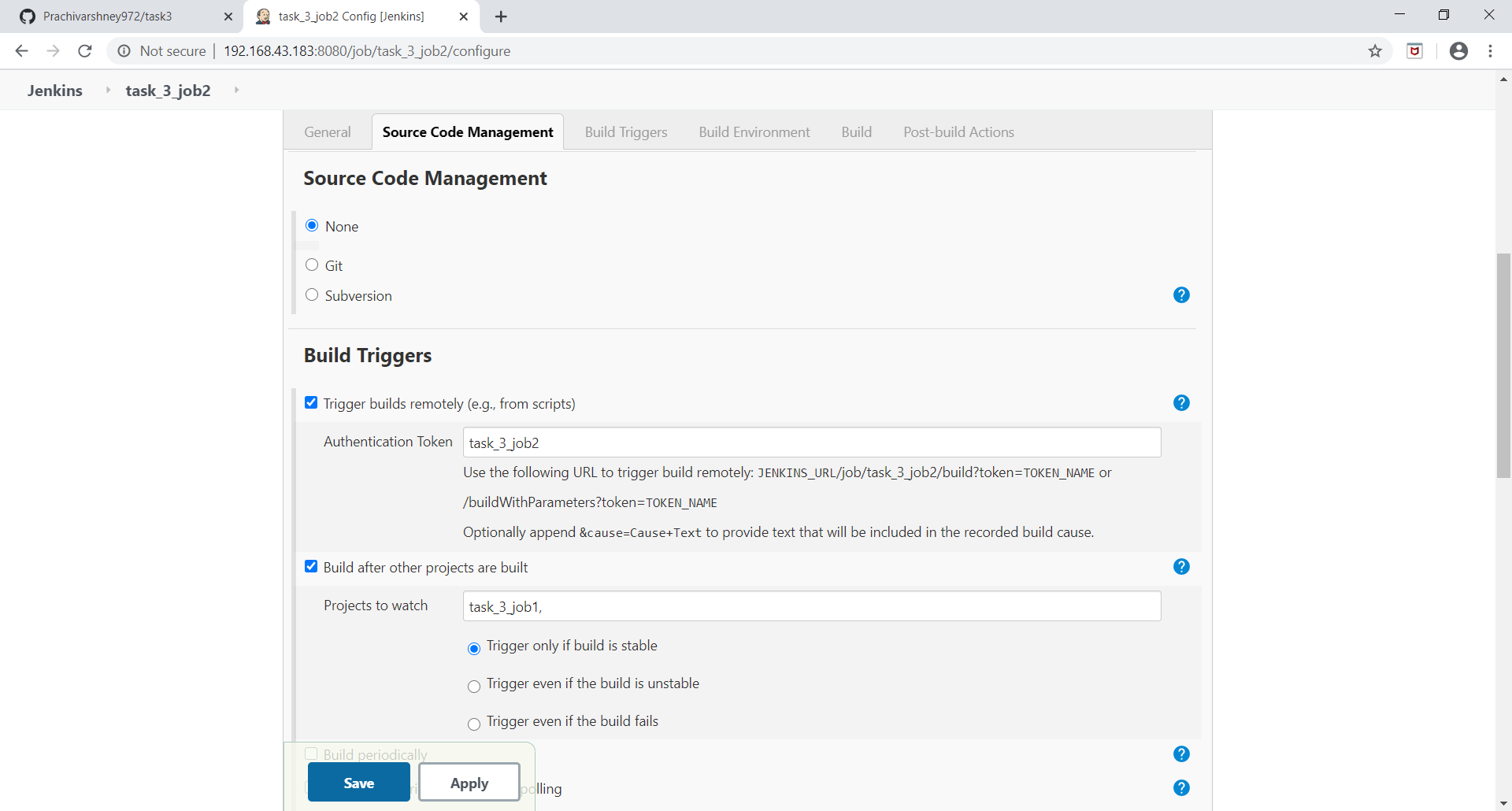
****

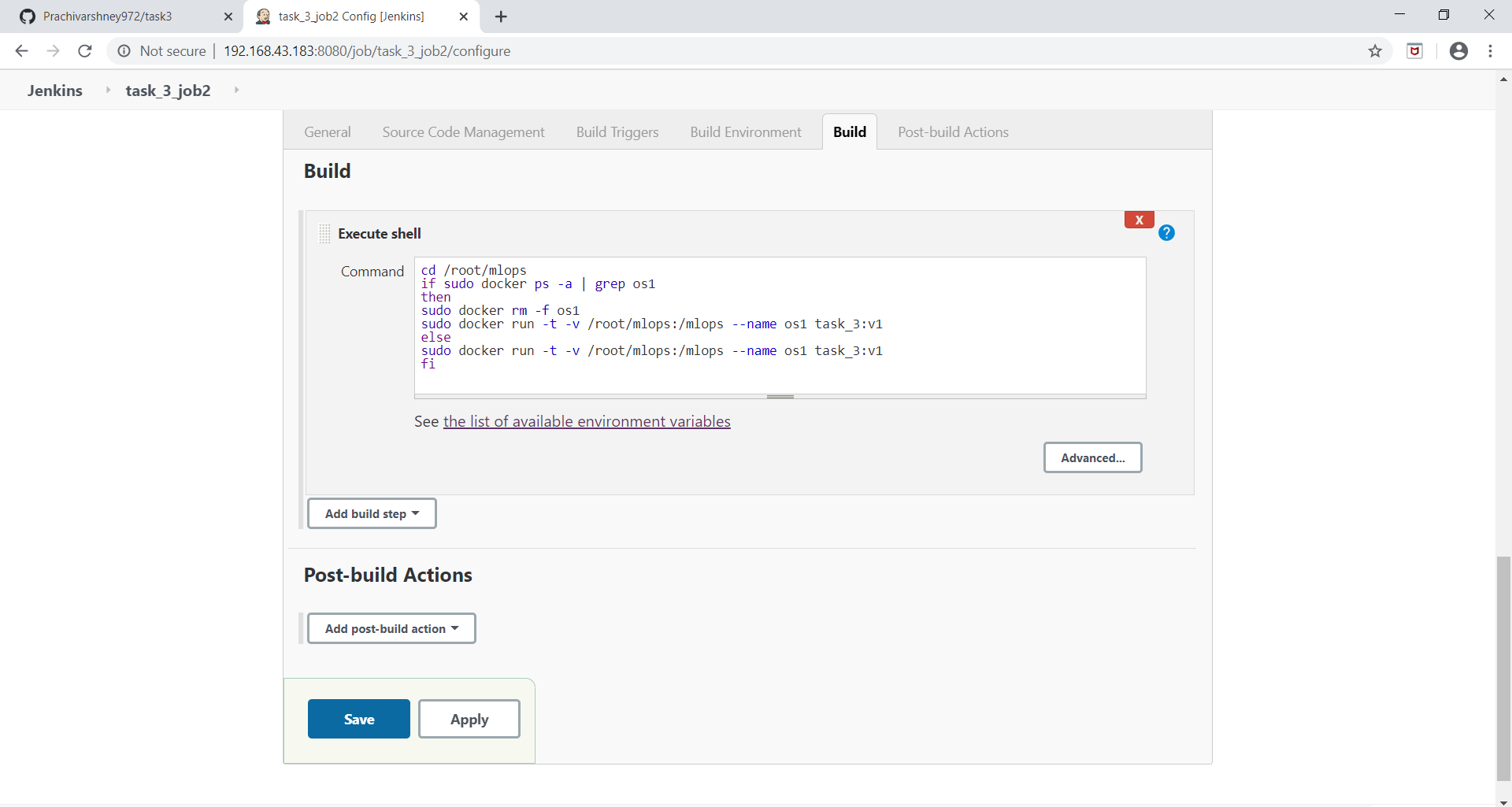






NOW “JOB2” configuration shown below:



one more job is also there i.e., “JOB3” to check accuracy

What it do ??

if accuracy is less than 91% , it increase no of epochs

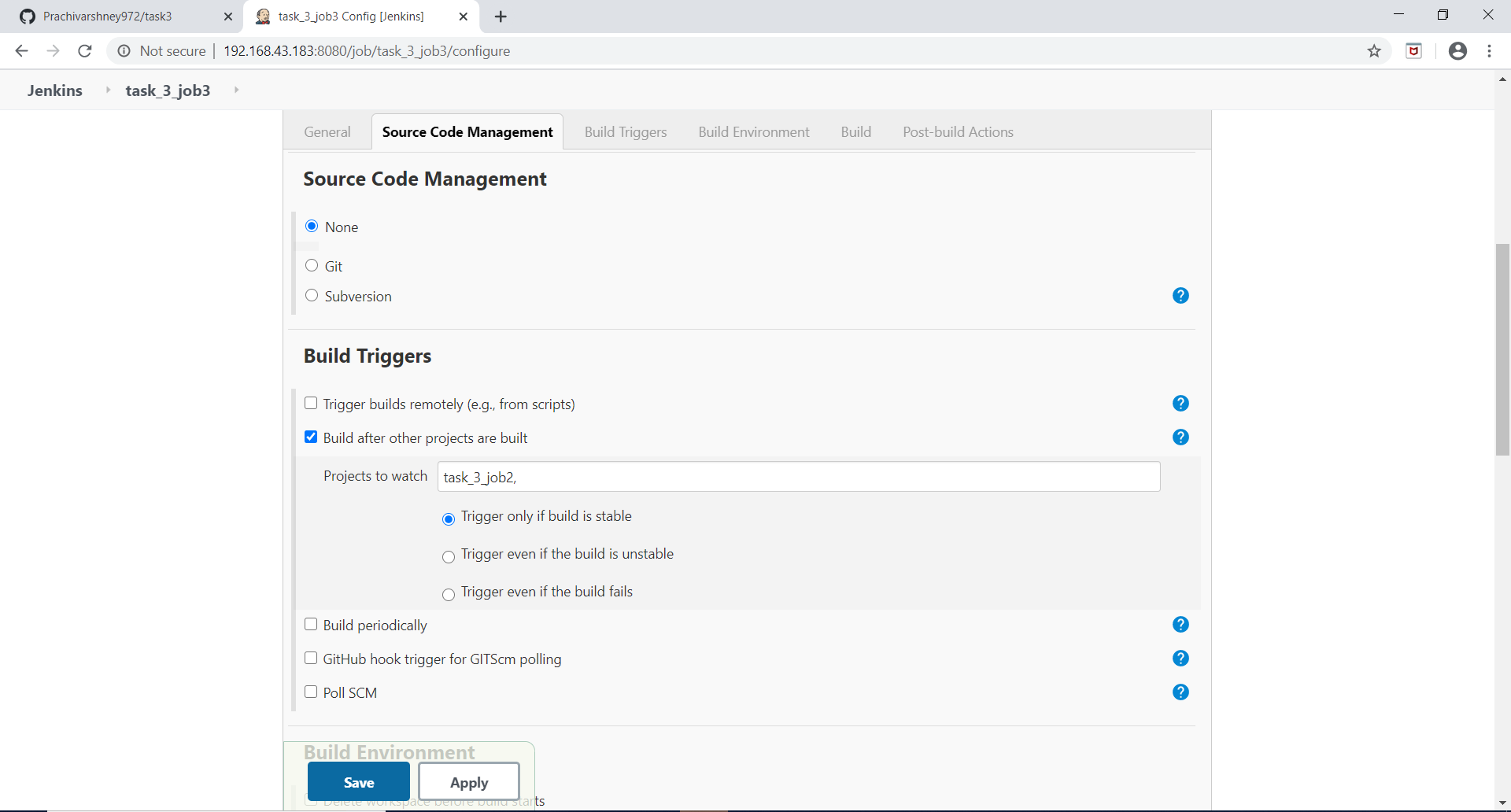
by running “tweakcode.py” file.

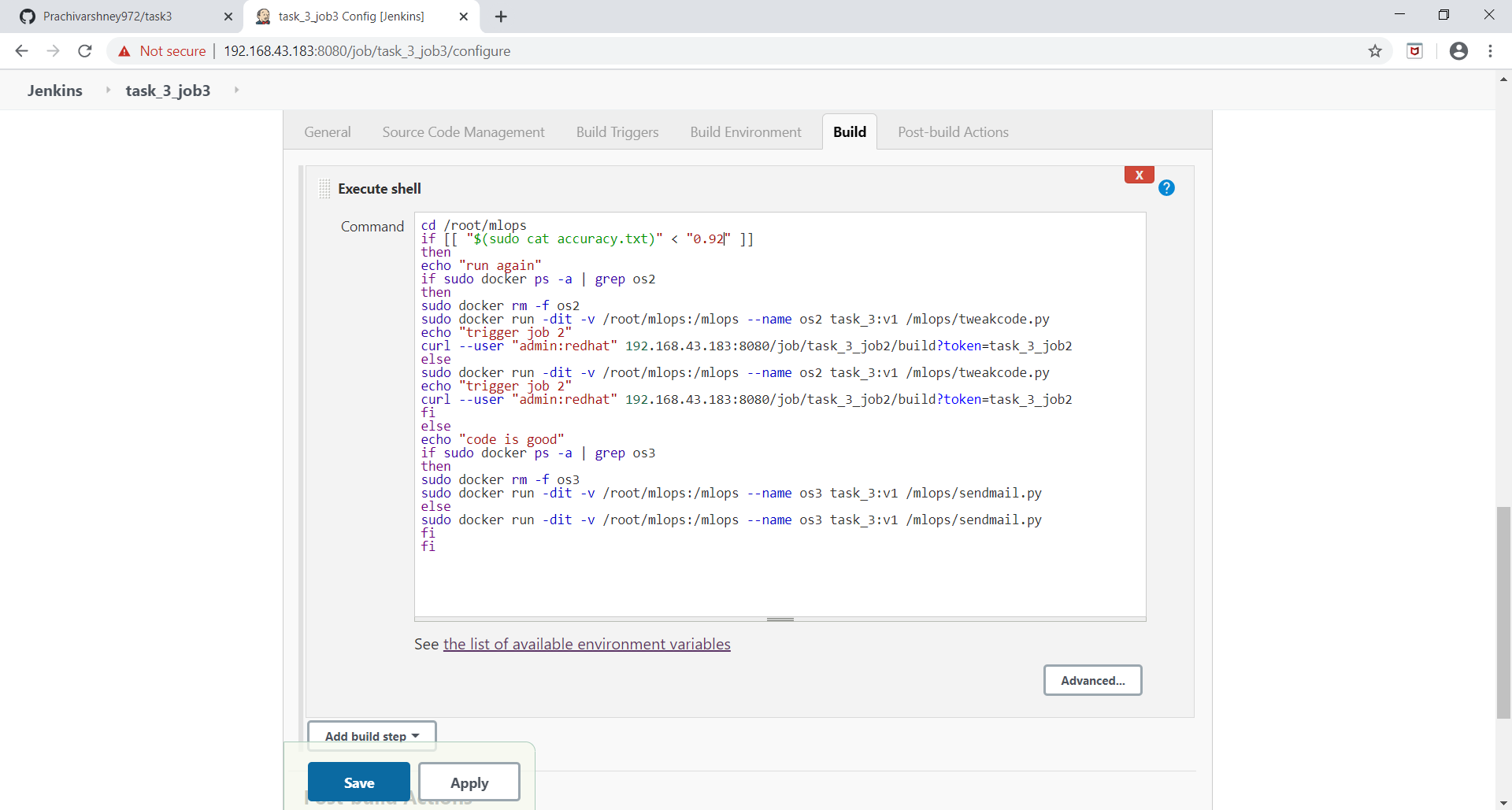
And again run job2 and then job3 , process goes on….until 91% accuracy model achieved….

As soon as model achieved accuracy > 91% , it send mail by running “sendmail.py” file.

***NOW SEE ,***

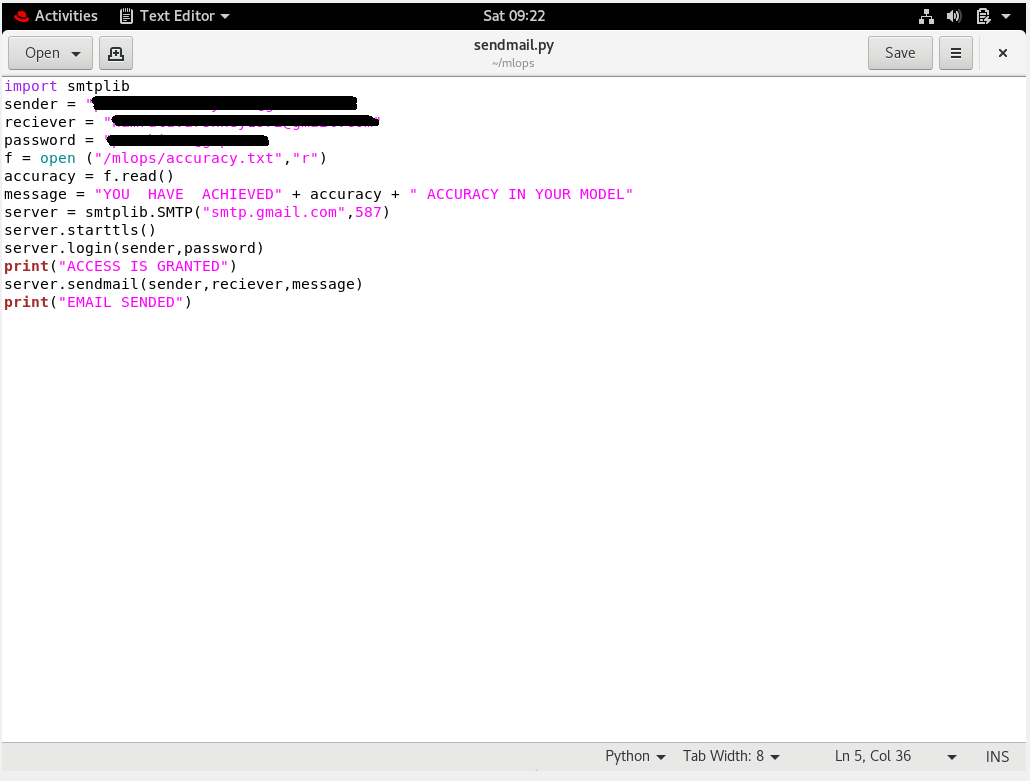
**“JOB3” configuration**





**SENDMAIL.PY**

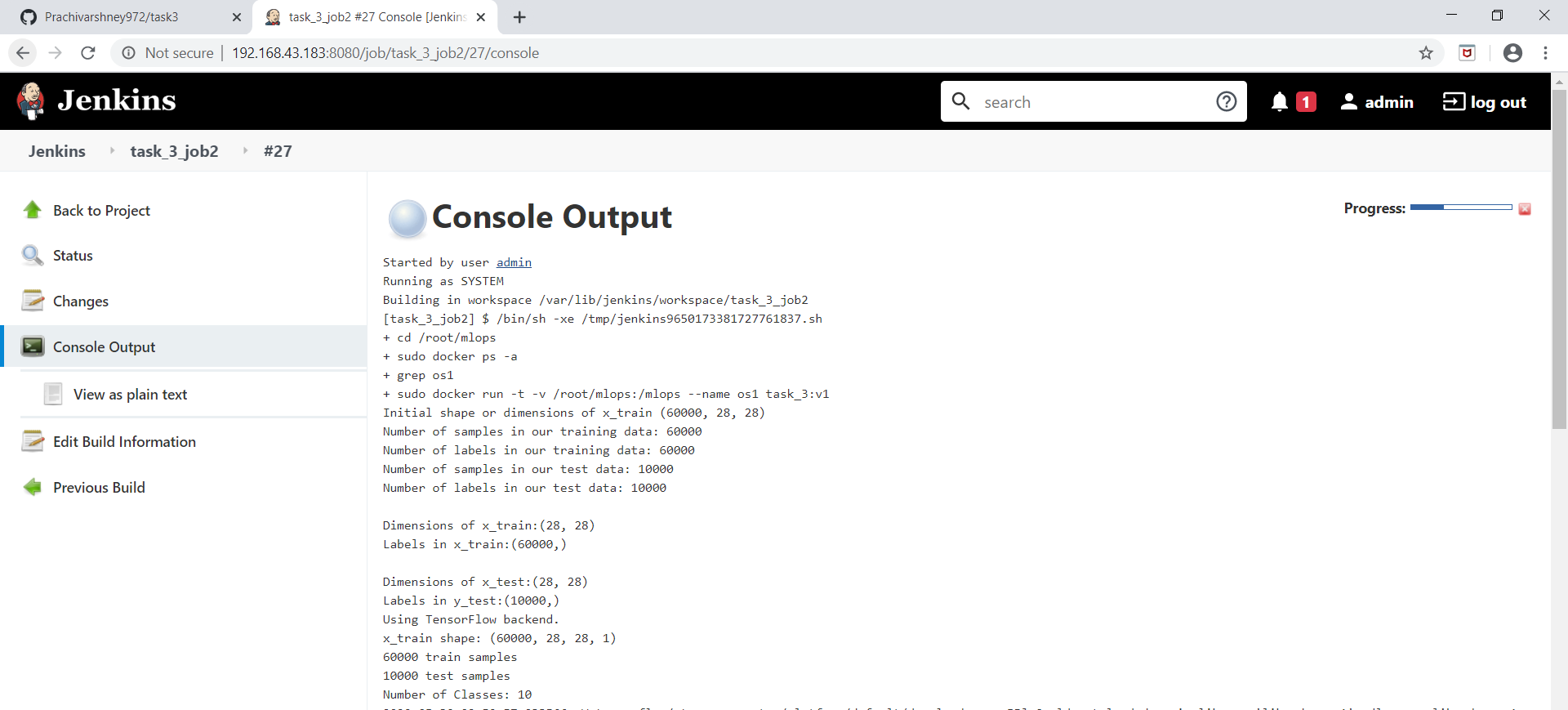
**FILE**

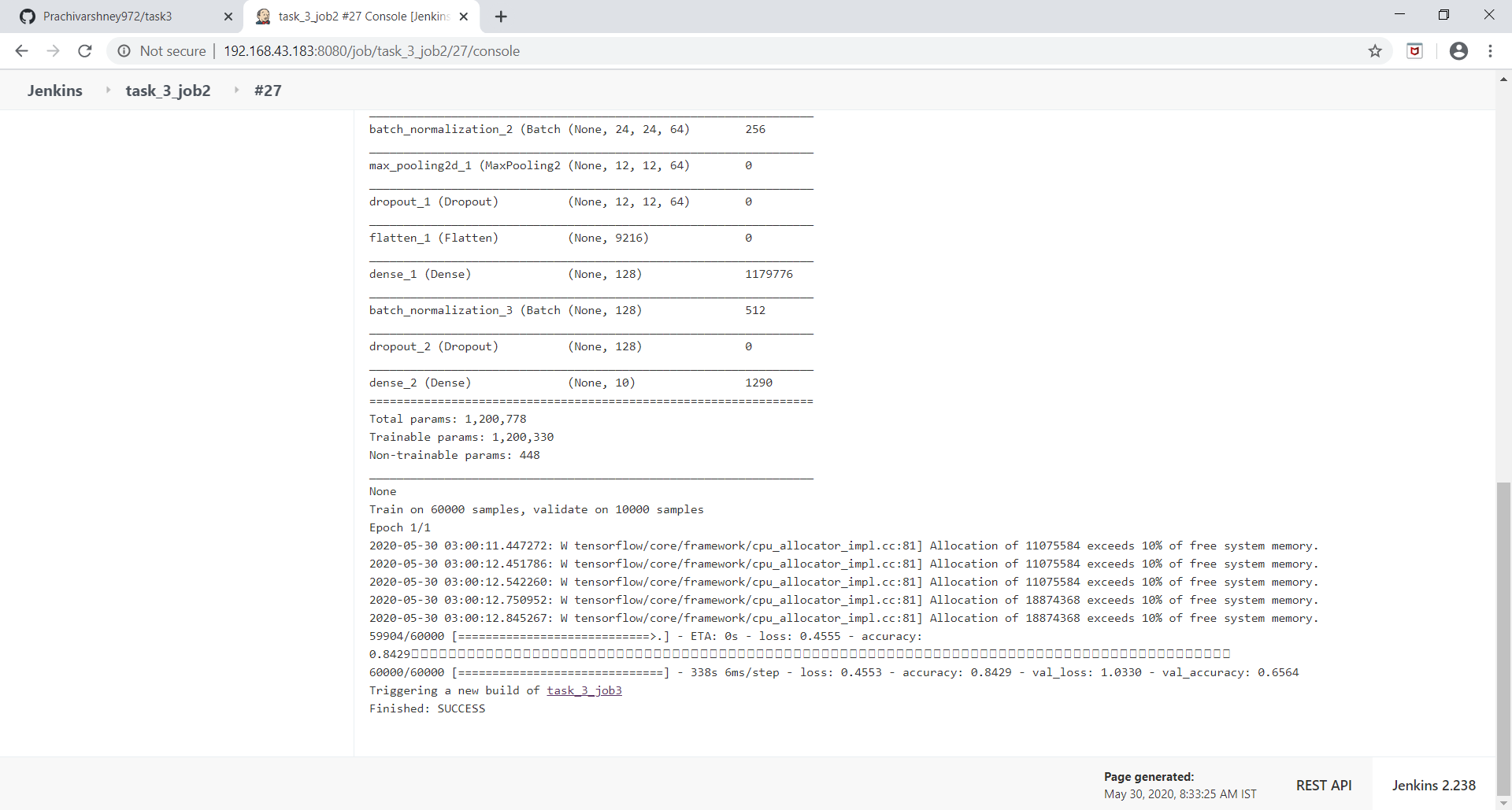


NOW ,

OUTPUT OF JOB2 AND JOB3, HOW THESE JOB WORK CONTINOUSLY /AUTOMATICALLY, TO GET 91% ACCURACY , SHOWN BELOW:

JOB2



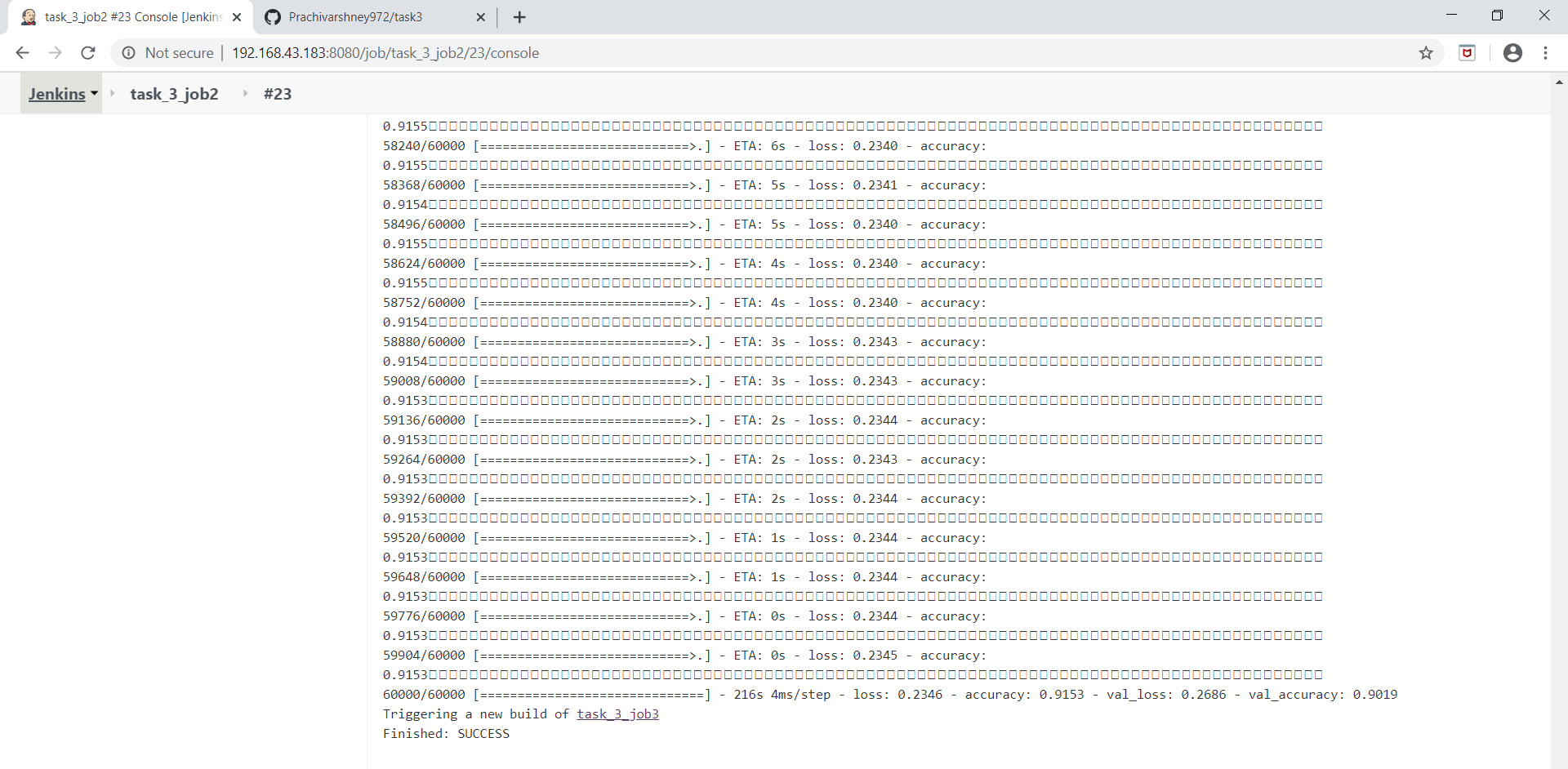


NOW JOB3 RUNS:

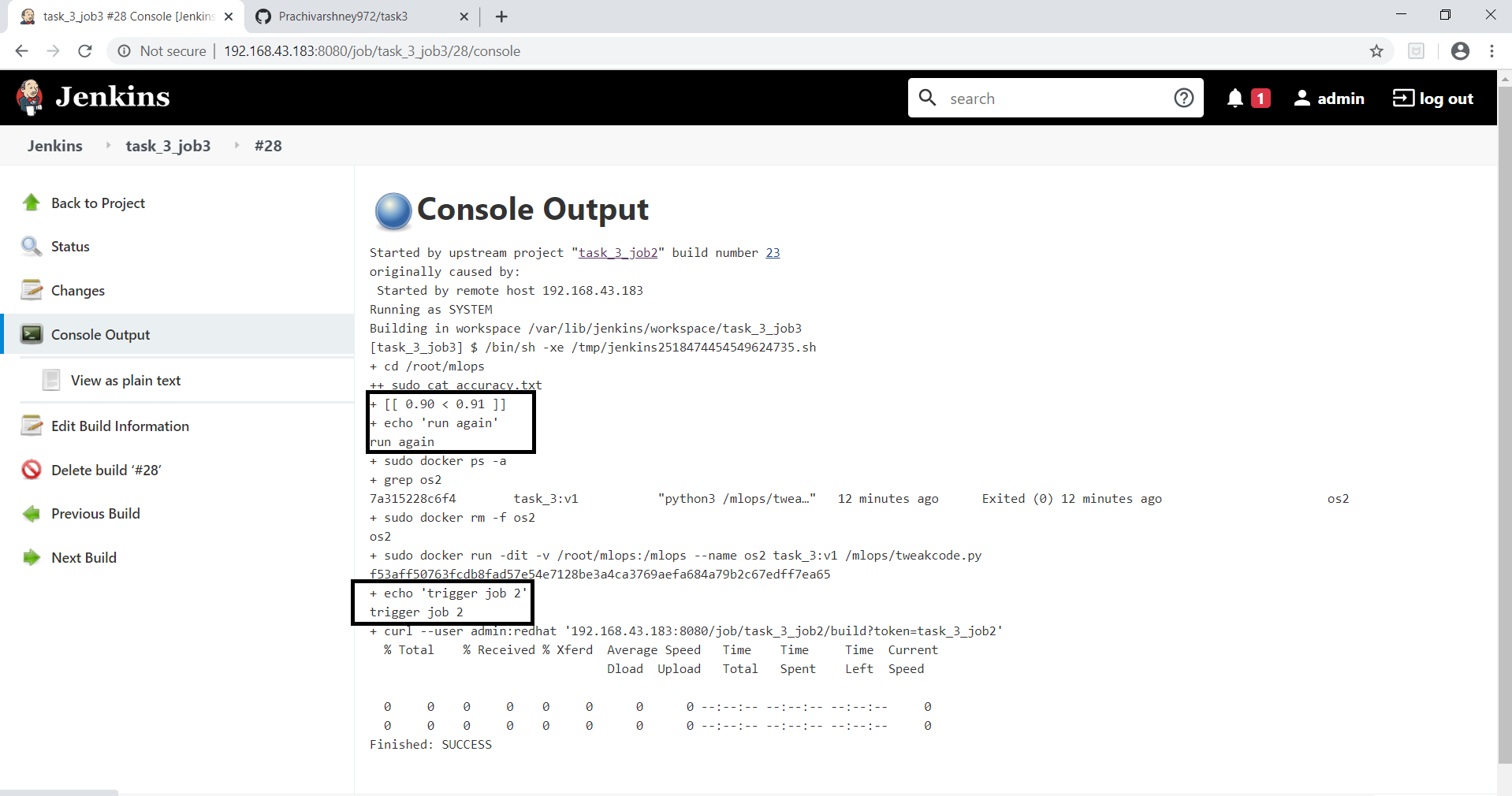


AGAIN JOB2 RUNS , but this time with 3 epochs

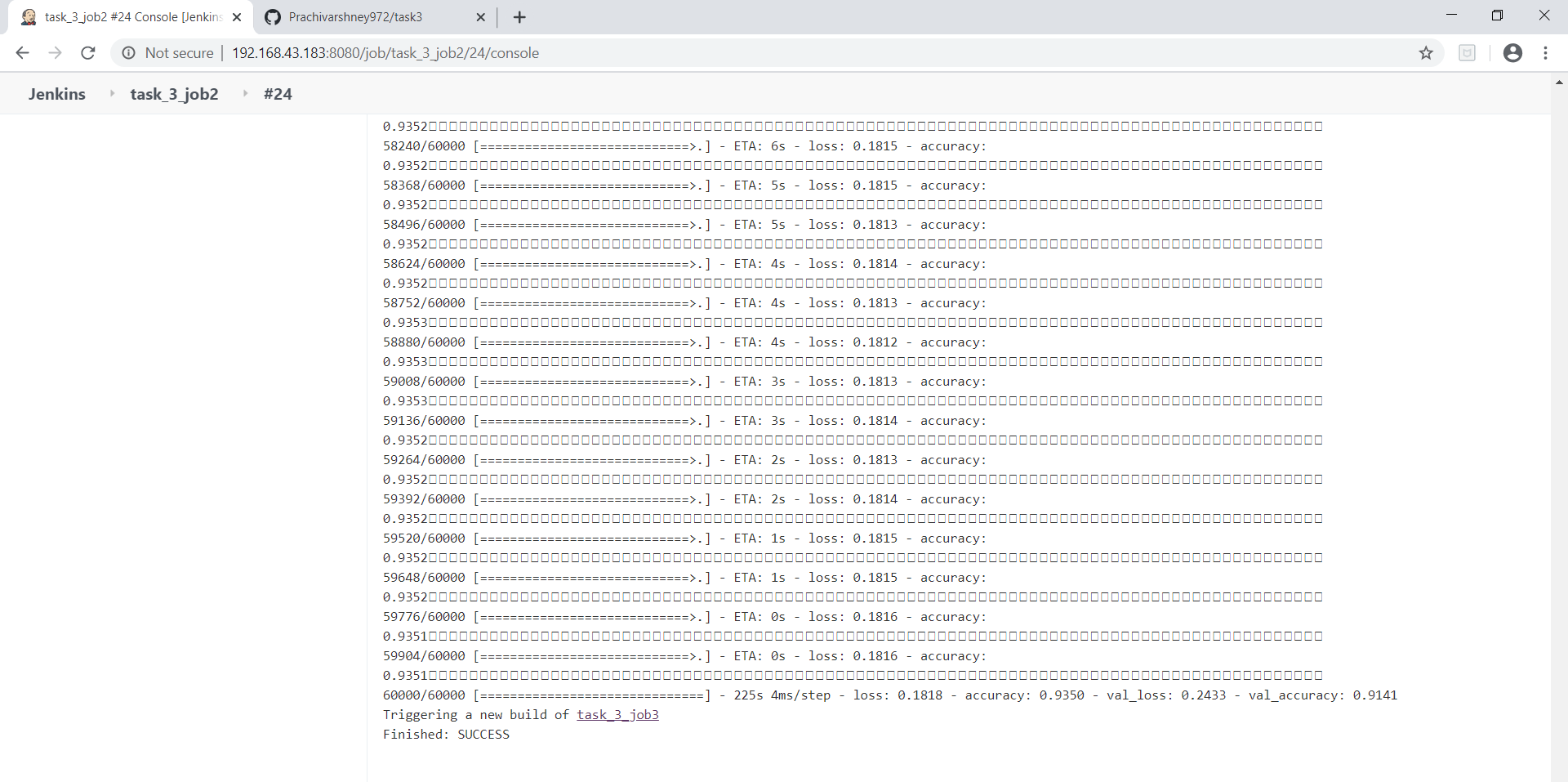


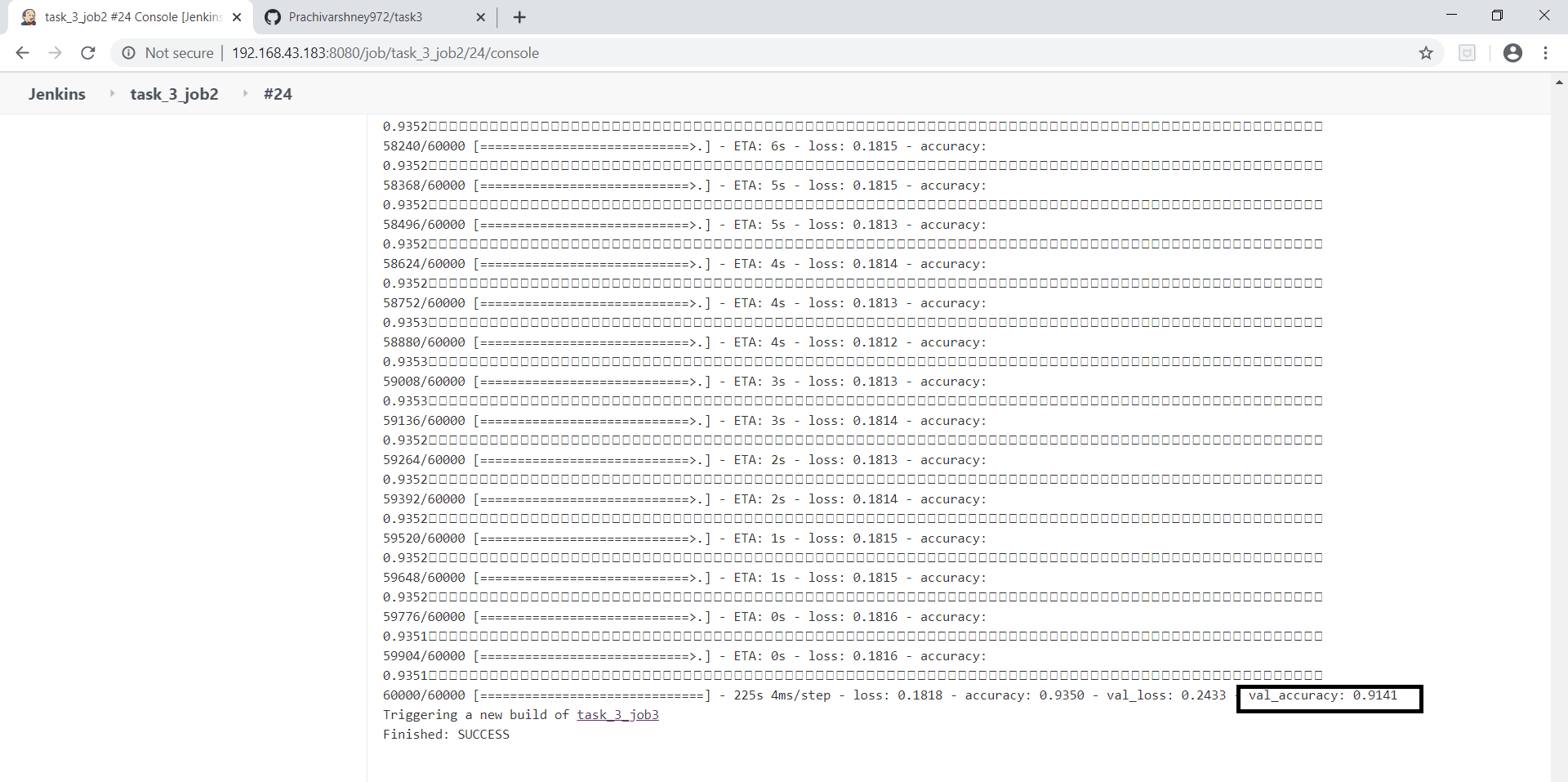


AGAIN JOB 3 RUNS ,



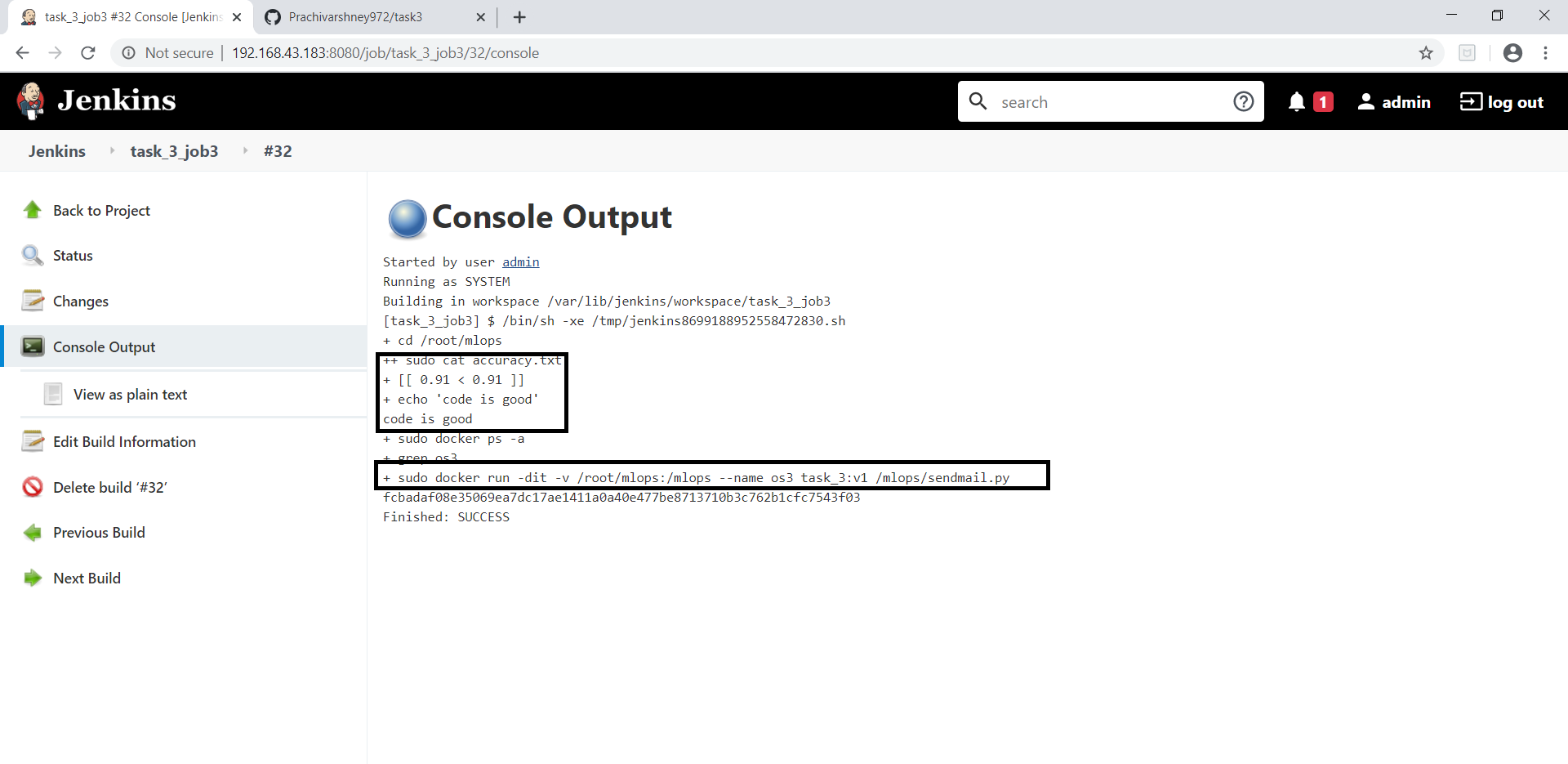
AGAIN JOB2 RUNS , because condition not satisfied till yet , and this time with 5 epochs.





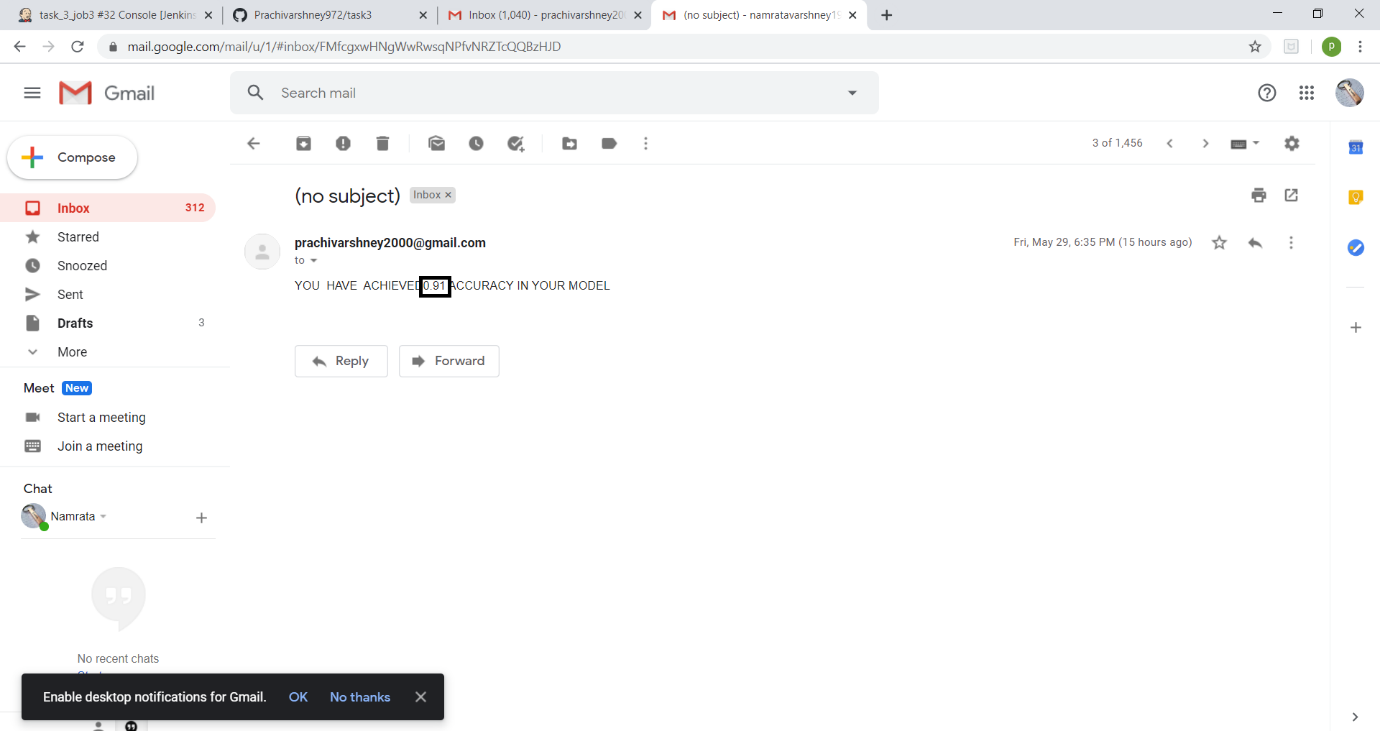
Now this time , my model achieved 91% accuracy ,

See what happens when job 3 run .



So, what happens here ,

Model achieved 91 % accuracy , so it send mail to developer….



In this way , we achieved automation , for increasing accuracy of model ( ML ) by using GIT , GITHUB , JENKINS , DOCKER …………