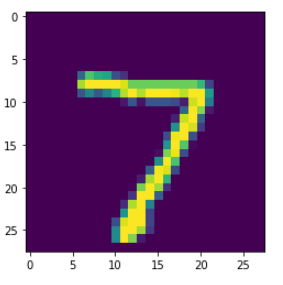
**MLOps Task**

**Task for integrating ML AND Devops**

1. Firstly, we will create two containers using Dockerfile. One container will be launched when we find a code of ML and another container will be launched if we find code of DL.
2. Then we will create a python code of DL in which we have used mnist dataset. We will use CNN to build our model. Mnist dataset is available on internet which contains thousands of images containing different ways in which a person can write a number. These are hand-written images. Like following is one of the mnist dataset image which have a way in which ‘7’ is written.



1. Then we will create a Jenkins jobs to perform different tasks. Here we will create 5 jobs
2. **JOB 1:**

We have given name ‘**djob1**’ to our JOB1.

The task of this job is to trigger github repo when a commit is done to our repo. As soon as the person will commit the code to github our job will keep track on the github as we have already set webhooks, so our job will trigger the repo from github.

1. **JOB 2:**

We have given name ‘**djob2**’ to our JOB2.

The task of this job is to launch the container based on our code. If the container is of ML code then our ML container will be launched and if we find code of DL then DL container will be launched.

1. **JOB 3:**

We have given name ‘**djob3**’ to our JOB3.

This job will be the most important job in all of the jobs. This job will check whether accuracy of our model is above 80%. If yes, then execution of our pipeline will be stopped with exit signal 1. If our model accuracy is less than 80% then JOB4 will be executed. Another thing is that this job will again execute after JOB5 if our model accuracy is found to be less than 80% because this job will again check that after modifying our code in JOB4 whether our model accuracy is now above 80% or not. If yes then execution will be terminated else it will continue. So we have actually performed cyclic job chaining here. You will understand this all in a better way with this diagram shown at last of this page.

1. **JOB4:**

We have given name ‘**djob4**’ to our JOB4.

This job is very much interesting. In this job we will modify our python code. We will modify number of Dense layers in our code through this job. We have used ‘sed’ shell script command for this. This command will help us to increase the number of dense layers in the job. We know that if Dense layers will increase then there are very good chances that our model accuracy will be increased.

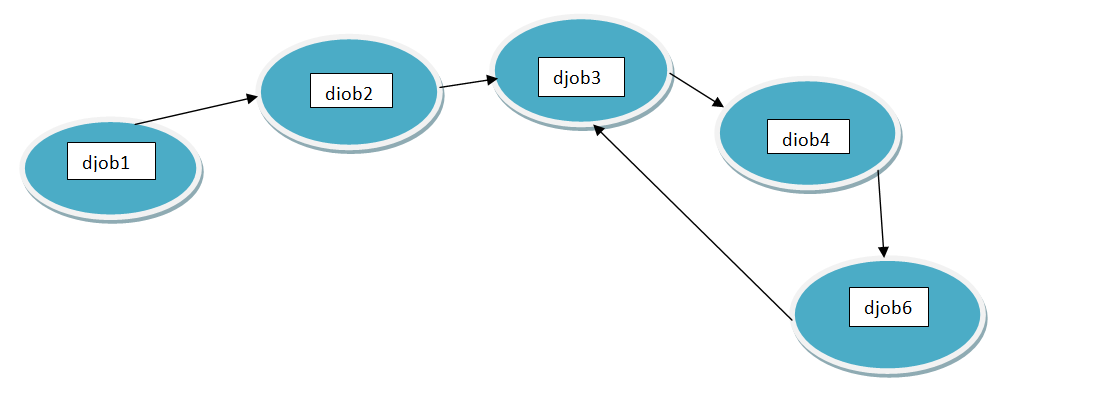
1. **JOB5:**

We have named ‘**djob6**’ to our job JOB5.

This job will check that whether our container is running properly or not. If by chance due to any reason our container will get stopped then this job will again start our same container.

1. **Job Chaining:**

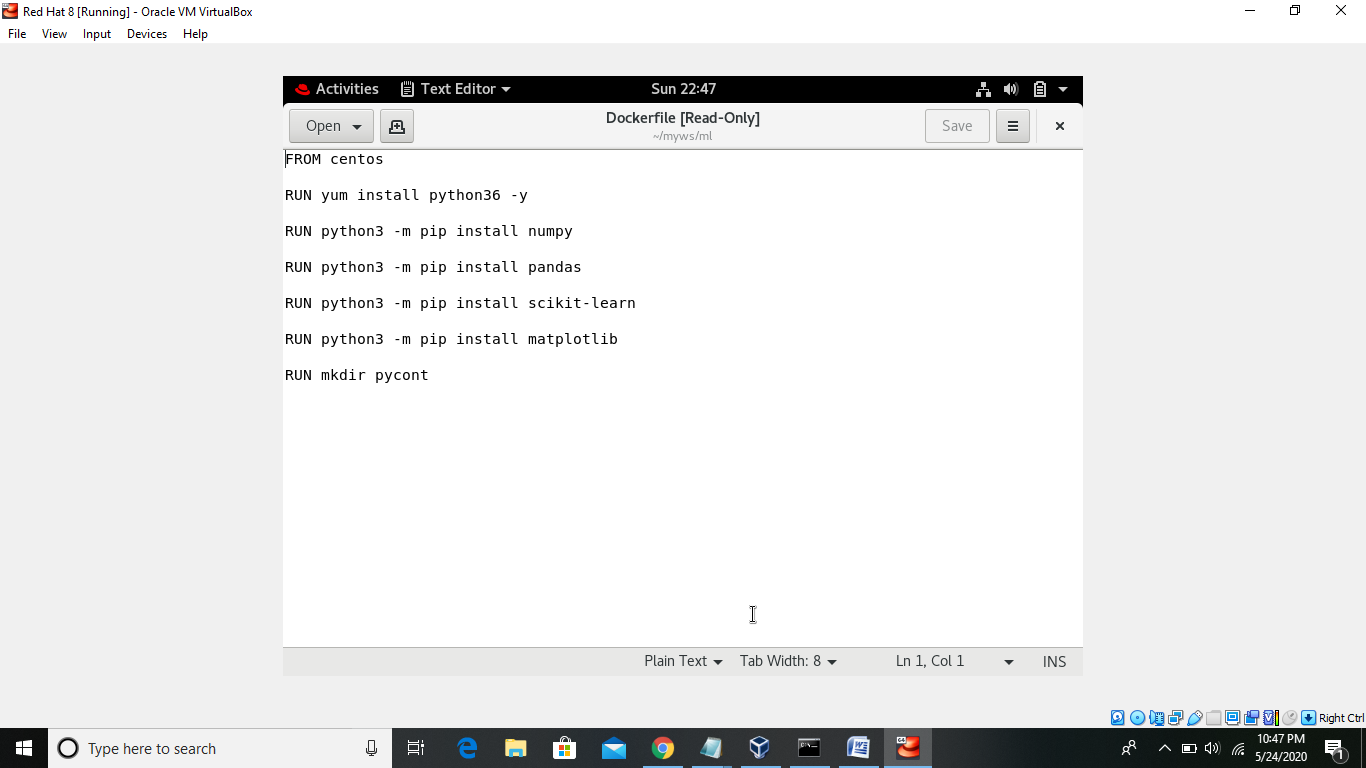
Actually our jobs are going to be chained in this way:



**Creation of Docker containers:**

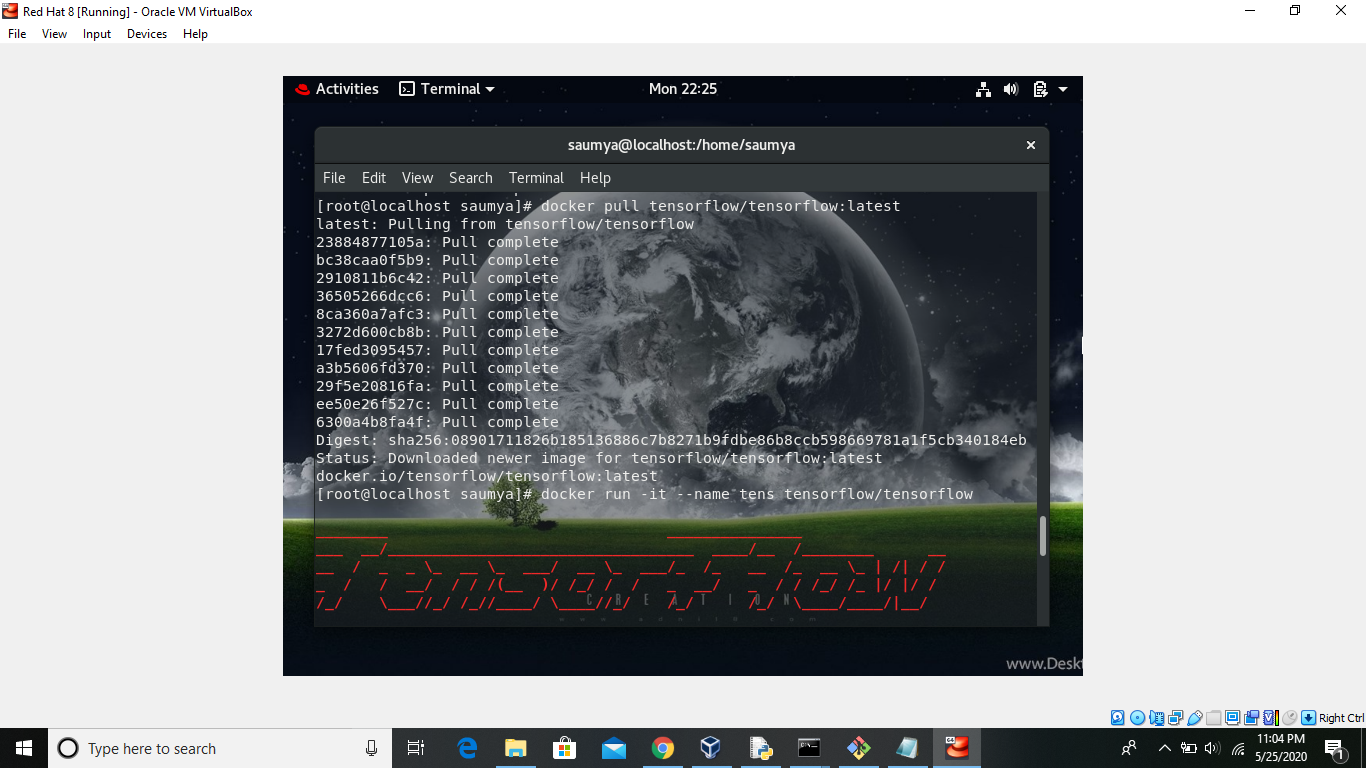
Here we have two docker images one for ML and another for DL

**Dockerfile for ML container:**



**Docker image for DL container:**

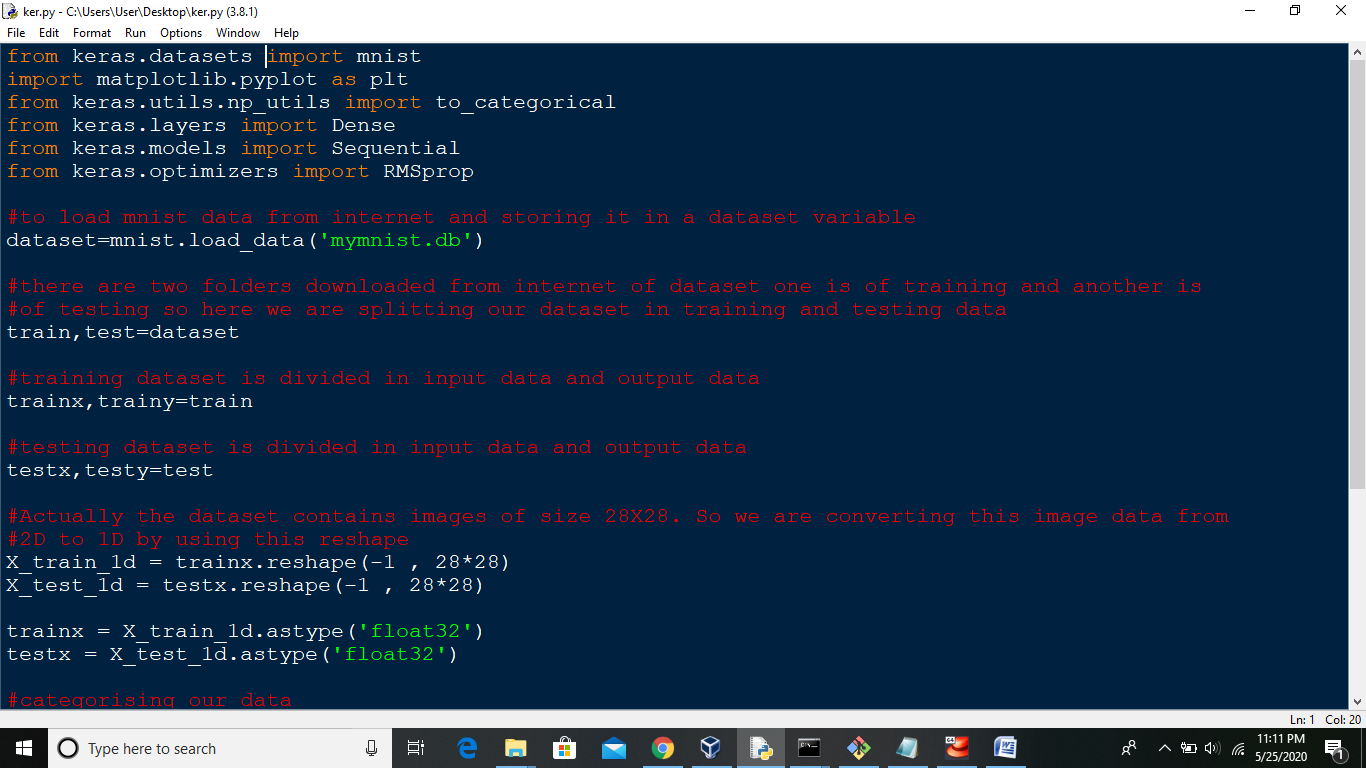
I have pulled docker image tensorflow/tensorflow from hub.docker.com because I was having a issue of downloading tensorflow library from Dockerfile. We will install keras library in this image when we will launch this container from Jenkins job.

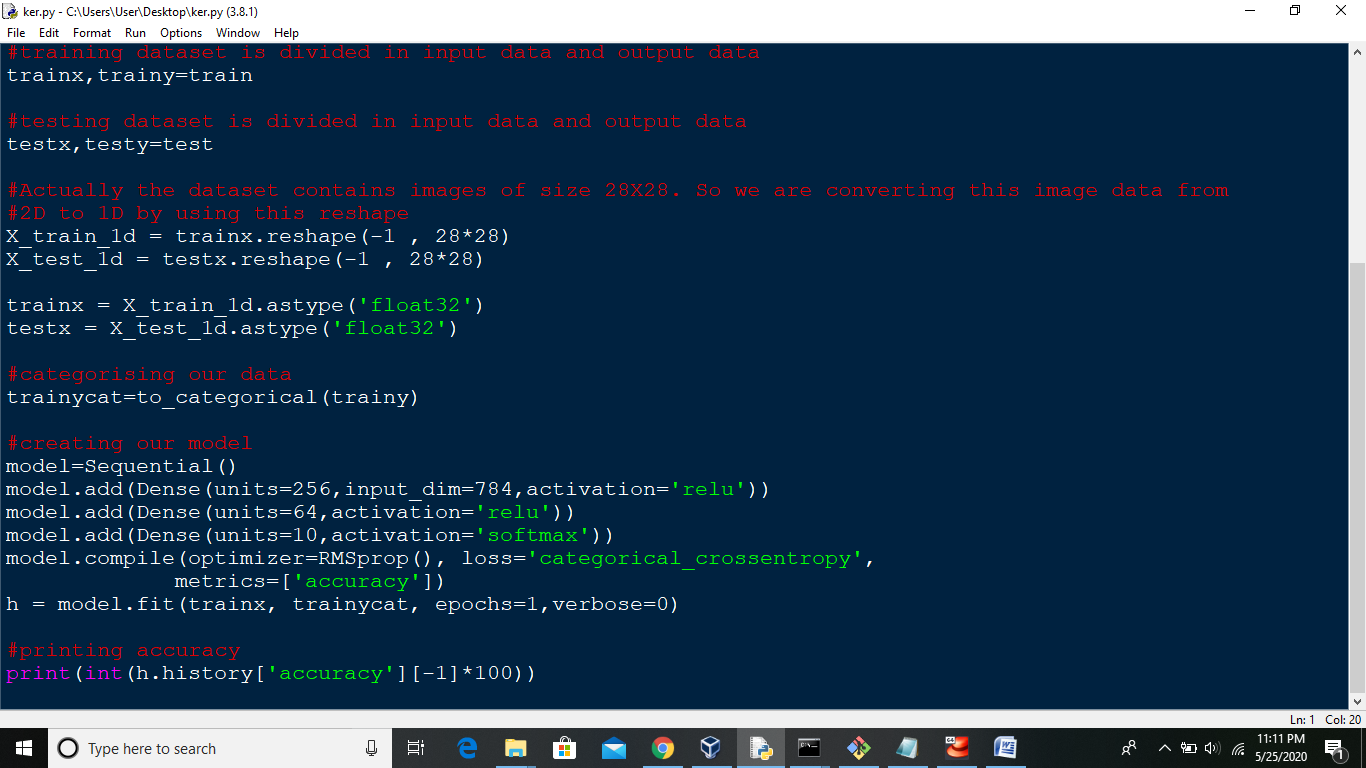


**Creating a python program based on CNN:**

**Our Program:**

This is our initial program which we are going to push to github





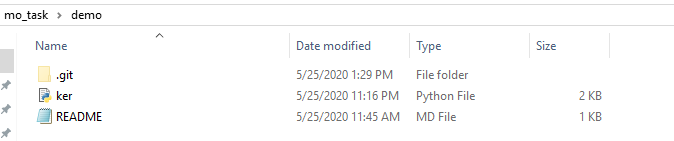
**Pushing our CNN program on GitHub:**

**Creating GITHUB Repo:**

After making program we will create a GITHUB repository. We have named it as ‘**demo**’.

Now we will clone it using Git.

Then we will copy our ‘**ker.py**’ file to this repo as shown here.



**Creating post-commit file in hooks :**

Then on GIT we have made a post-commit file



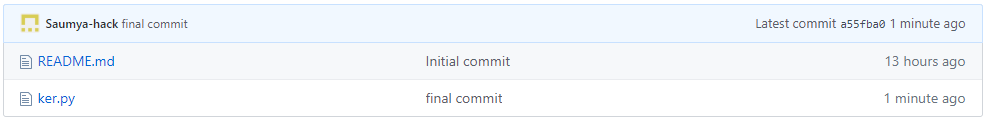
**Committing and pushing file to Github:**

Then on GIT we will add this file and then we will make a commit





After committing, our repo will contain these files.

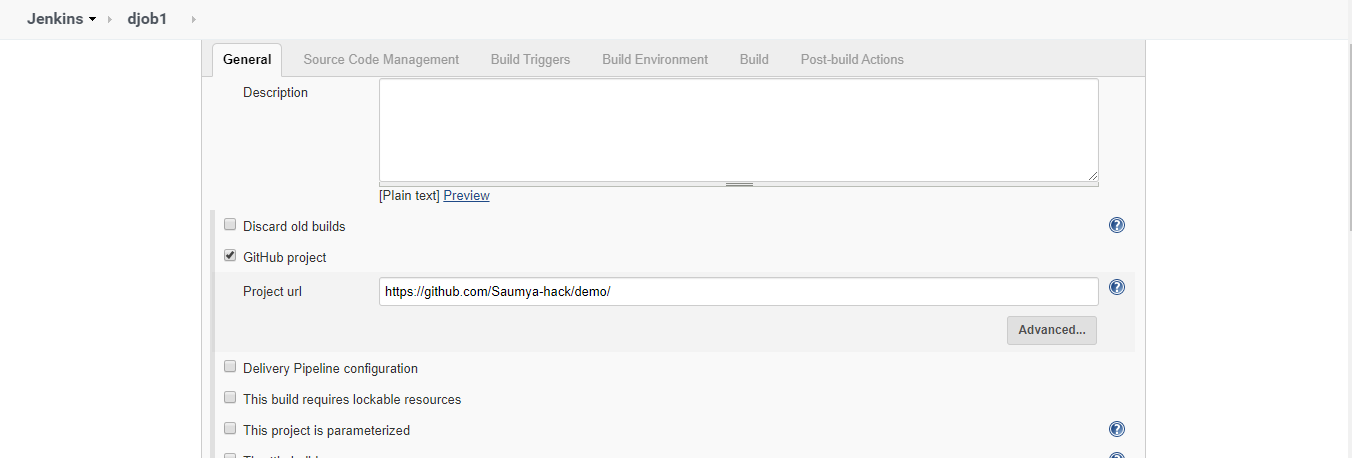


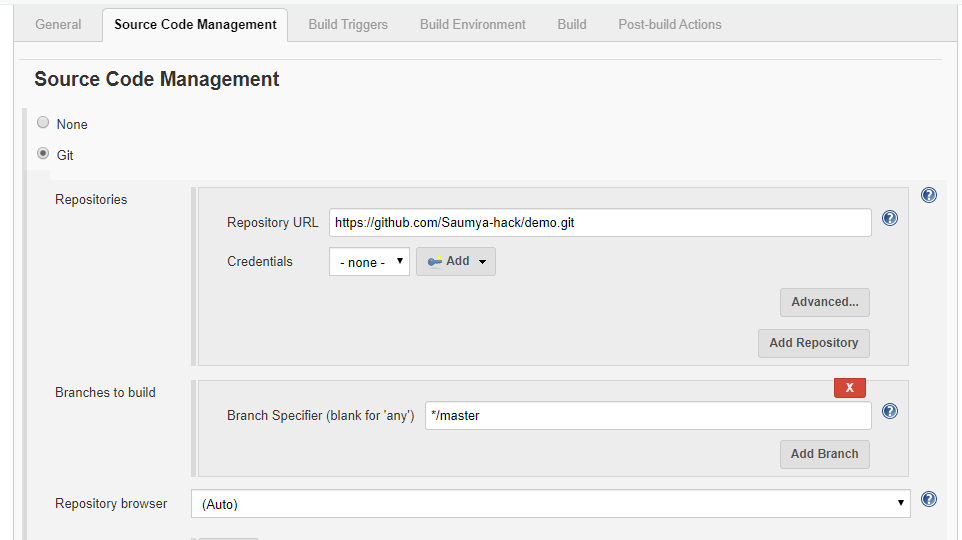
**Jenkins jobs configuration:**

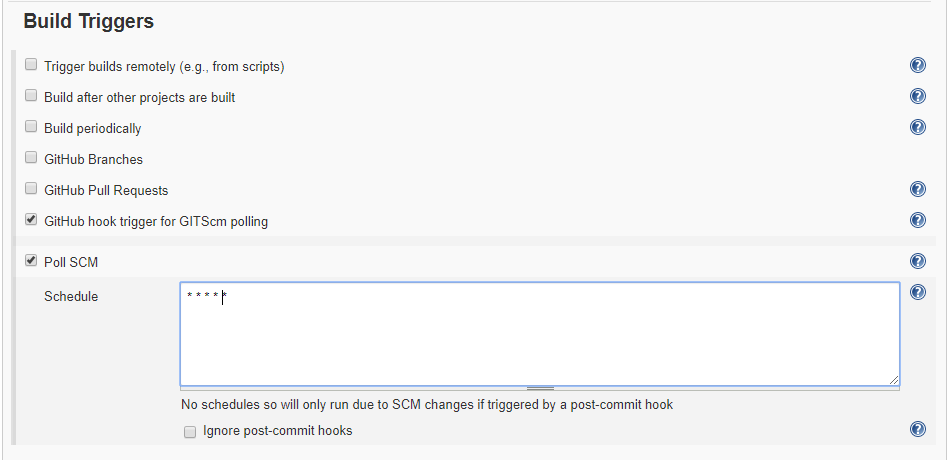
**JOB 1:**

We have given name ‘**djob1**’ to our JOB1.

The task of this job is to trigger github repo when a commit is done to our repo. As soon as the person will commit the code to github our job will keep track on the github as we have already set webhooks, so our job will trigger the repo of github from this job and code will get downloaded automatically.



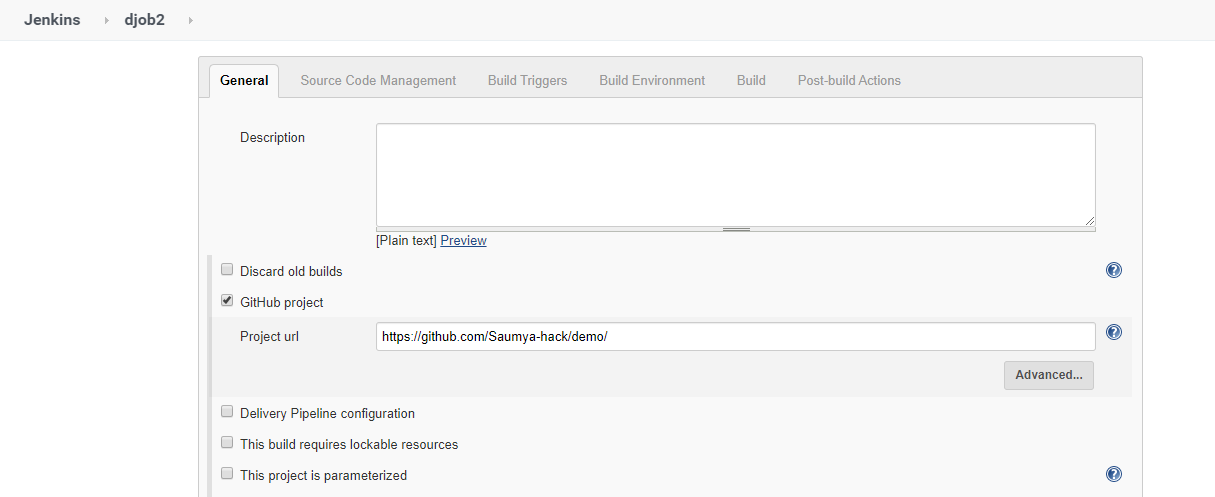


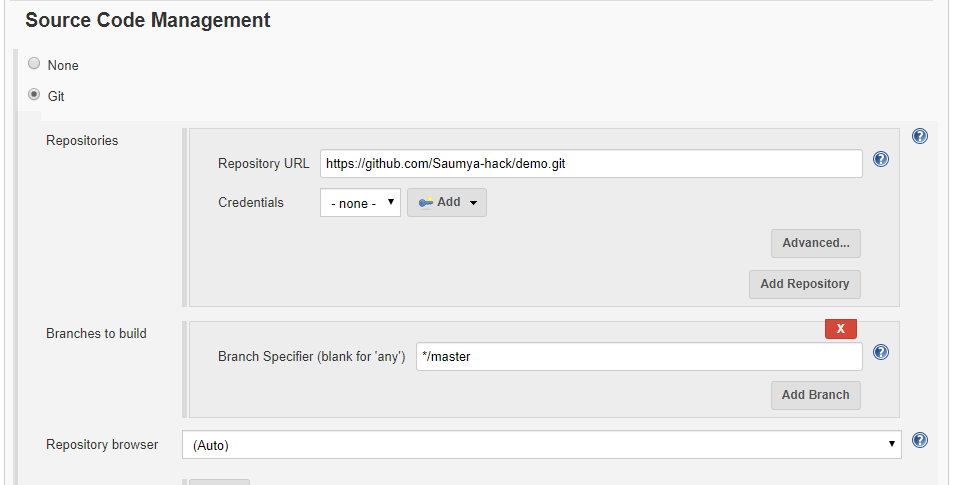


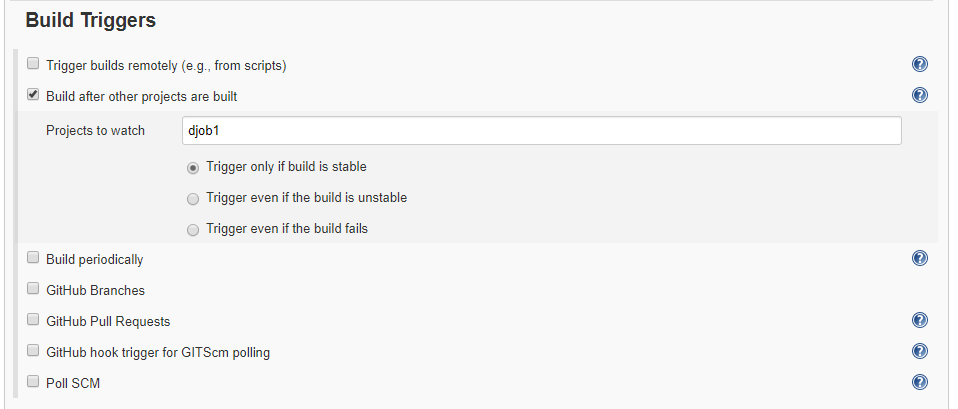
**JOB 2:**

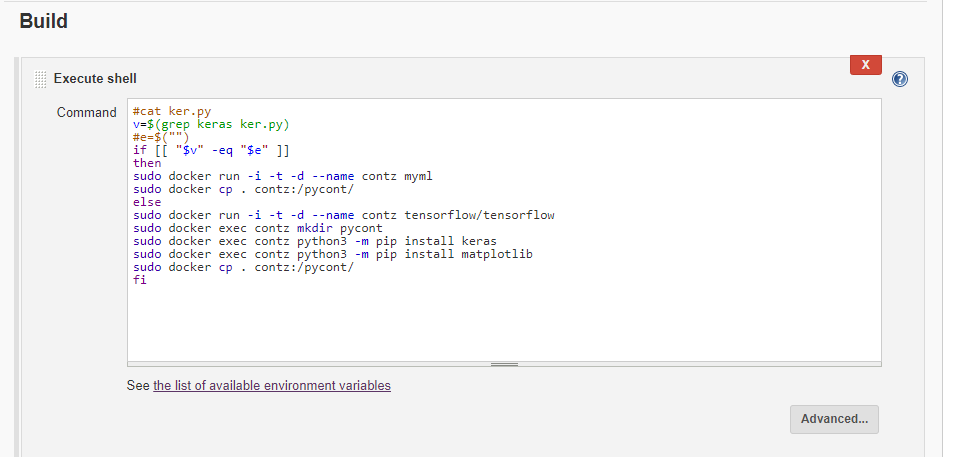
We have given name ‘**djob2**’ to our JOB2.

The task of this job is to launch the container based on our code. If the container is of ML code then our ML container will be launched and if we find code of DL then DL container will be launched. This job will be executed as soon as JOB1 will finish it’s execution.





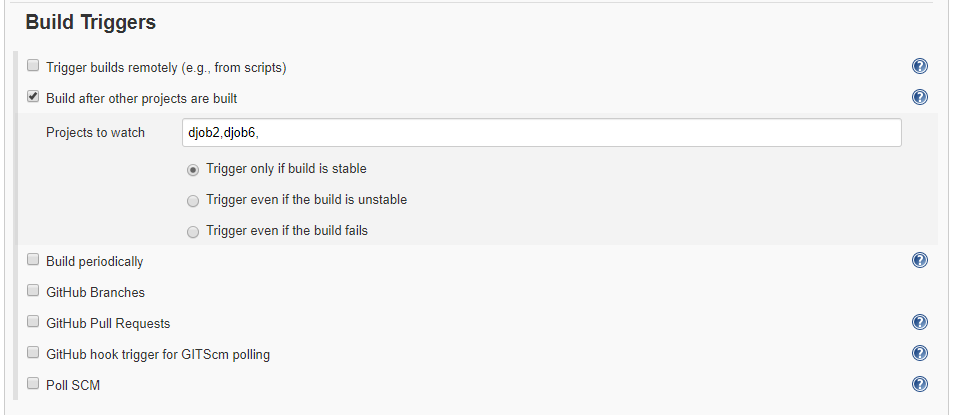


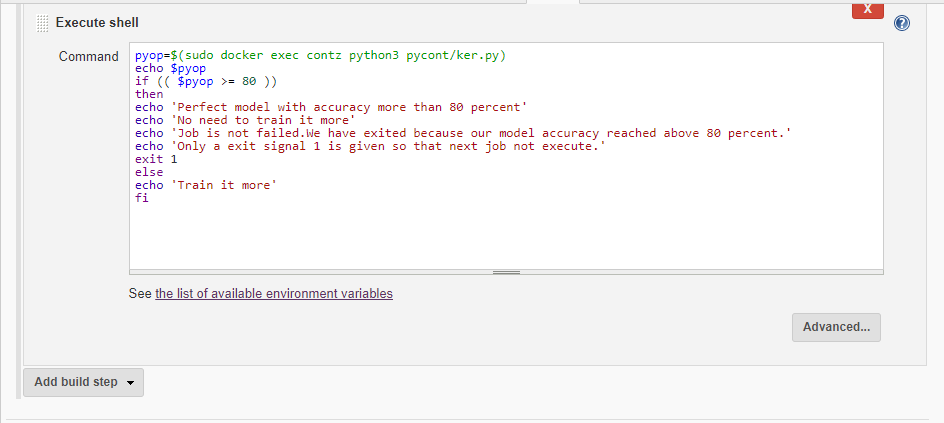


**JOB 3:**

We have given name ‘**djob3**’ to our JOB3.

This job will check whether accuracy of our model is above 80%. If yes, then execution of our pipeline will be stopped with exit signal 1. If our model accuracy is less than 80% then JOB4(djob4) will be executed. Another thing is that this job will again execute after JOB5(djob6) if our model accuracy is found to be less than 80% because this job will again check that after modifying our code in JOB4 whether our model accuracy is now above 80% or not. If yes then execution will be terminated else it will continue. So we have actually performed cyclic job chaining here as shown previously.

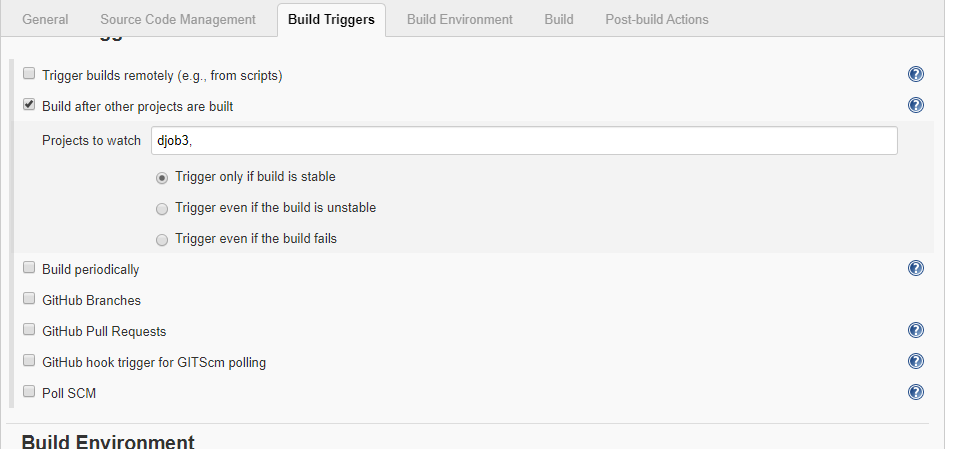




**JOB 4:**

We have given name ‘**djob4**’ to our JOB4.

This job is very much interesting. In this job we will modify our python code. We will modify number of Dense layers in our code through this job. We have used ‘sed’ shell script command for this. This command will help us to increase the number of dense layers in the job. We know that if Dense layers will increase then there are very high chances that our model accuracy will be increased.

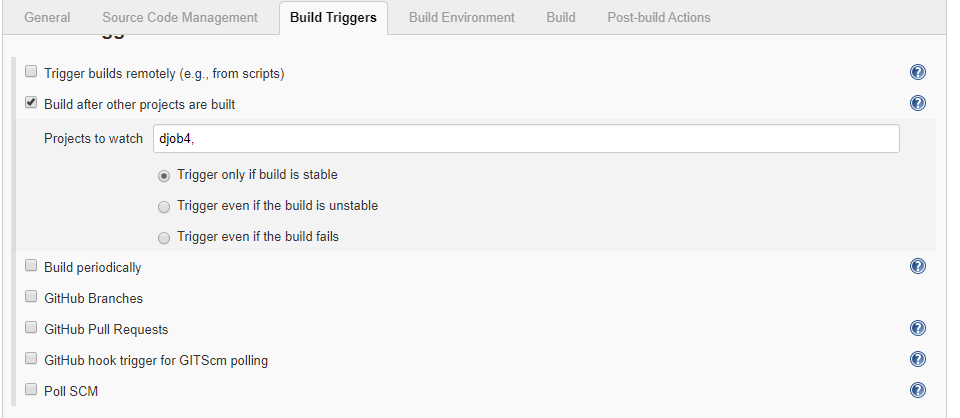


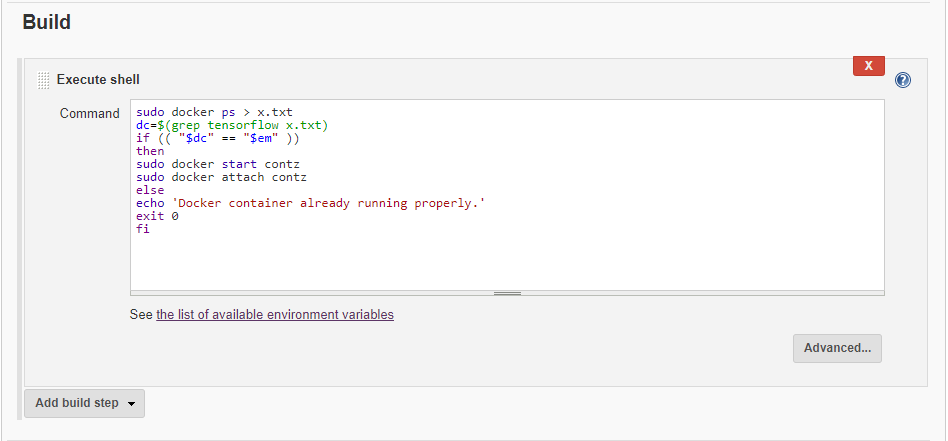


**JOB 5:**

We have named ‘**djob6**’ to our job JOB5.

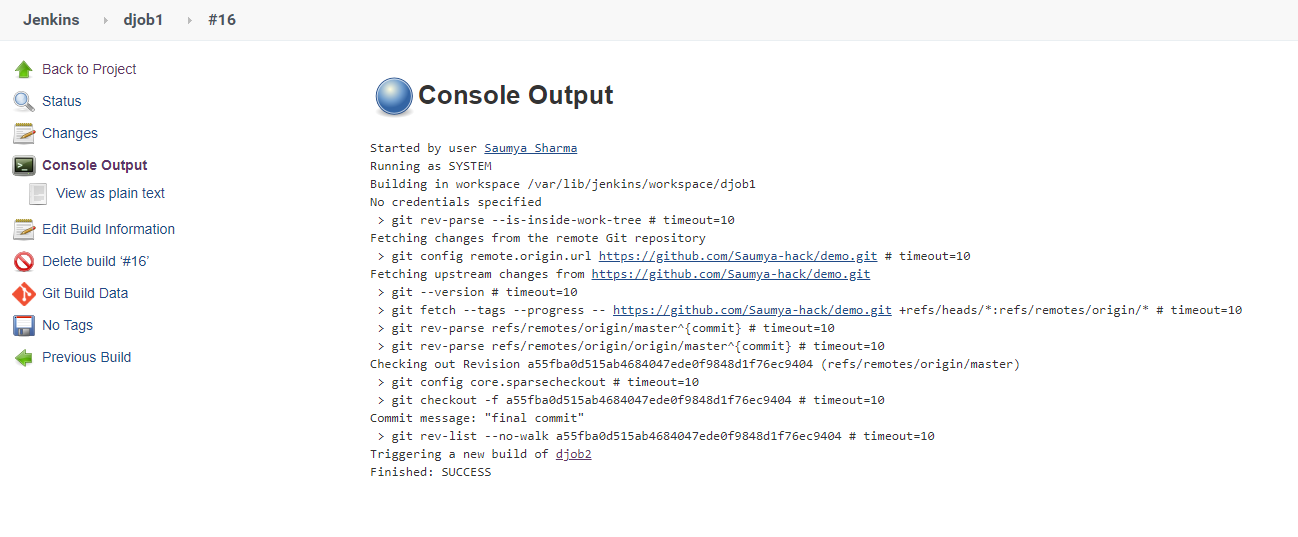
This job will check that whether our container is running properly or not. If by chance due to any reason our container will get stopped then this job will again start our same container.



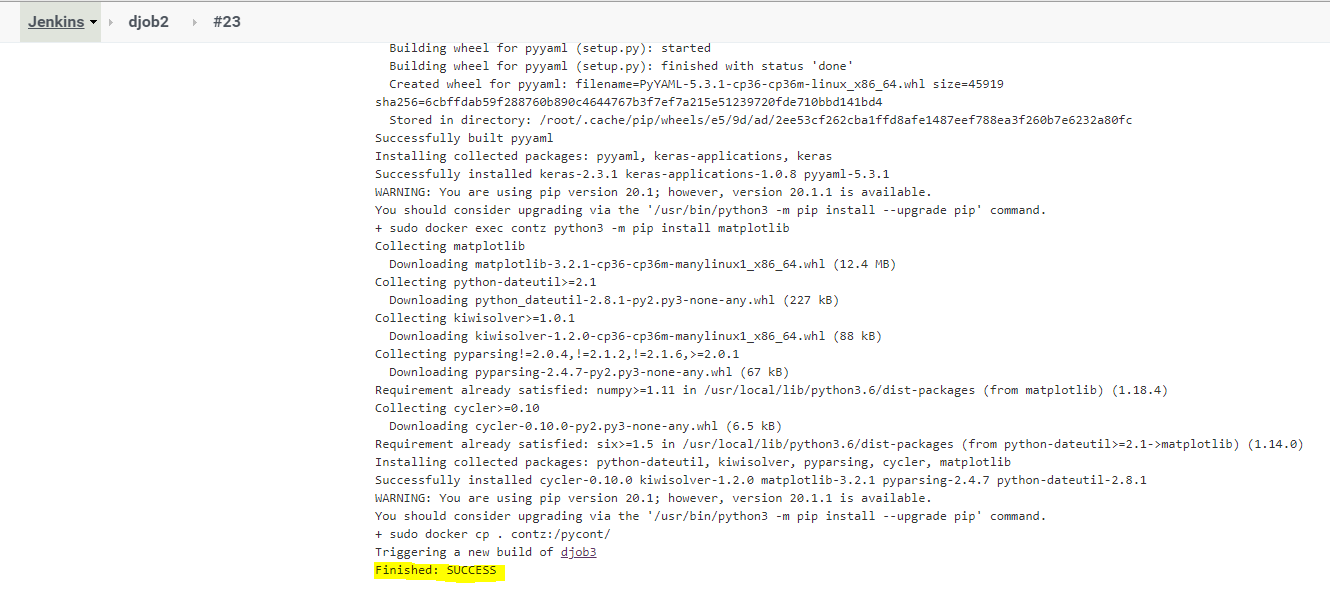


**Console outputs when our jobs will execute**

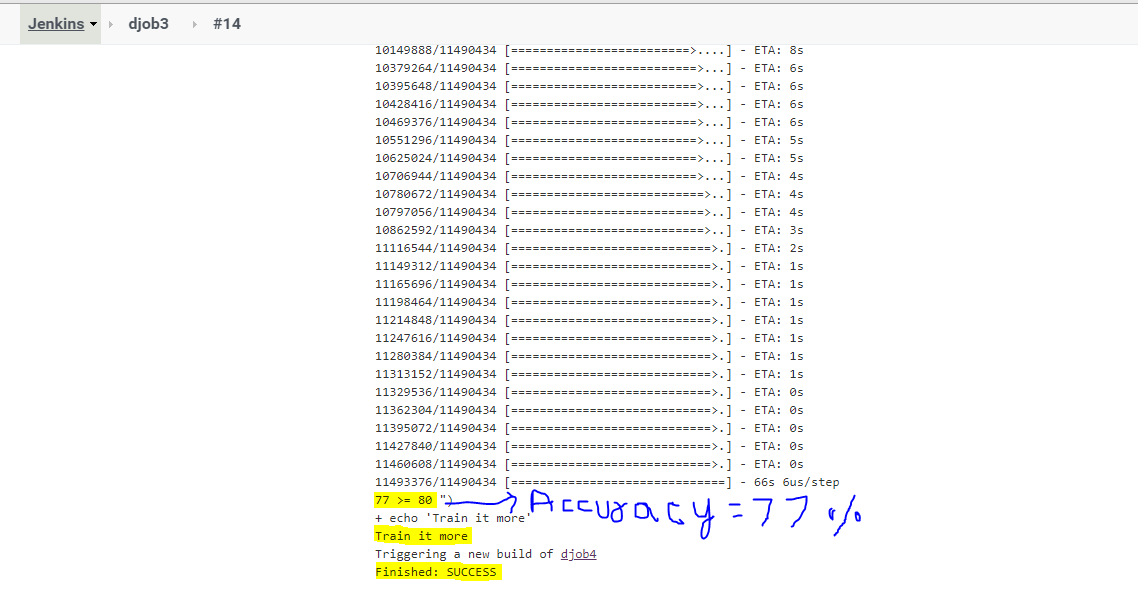
**JOB 1 executes:**



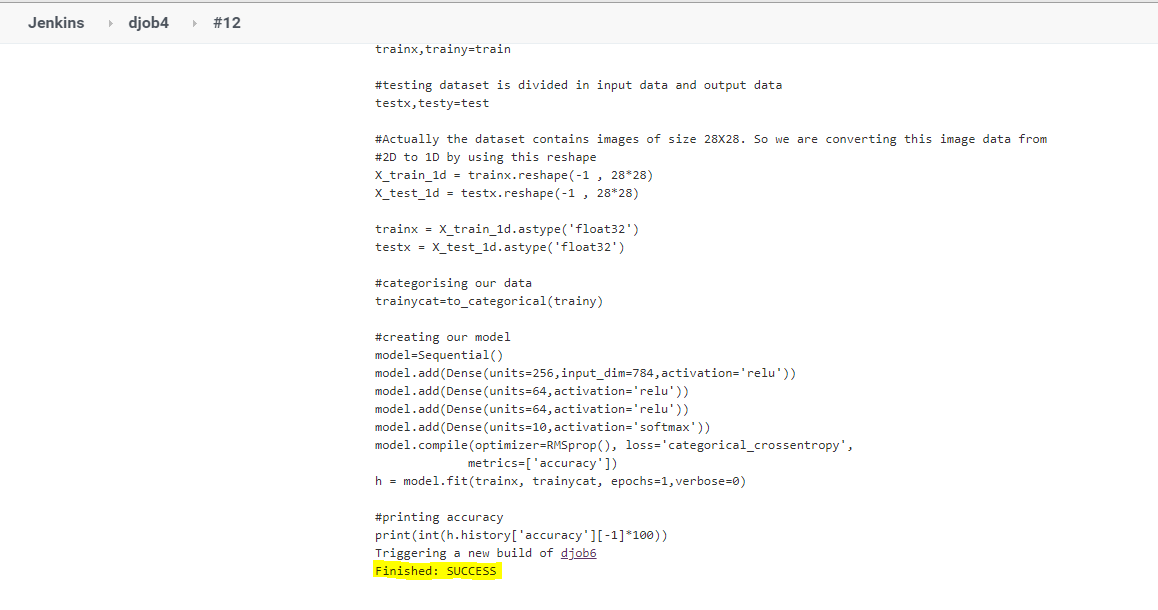
**JOB 2 executes:**



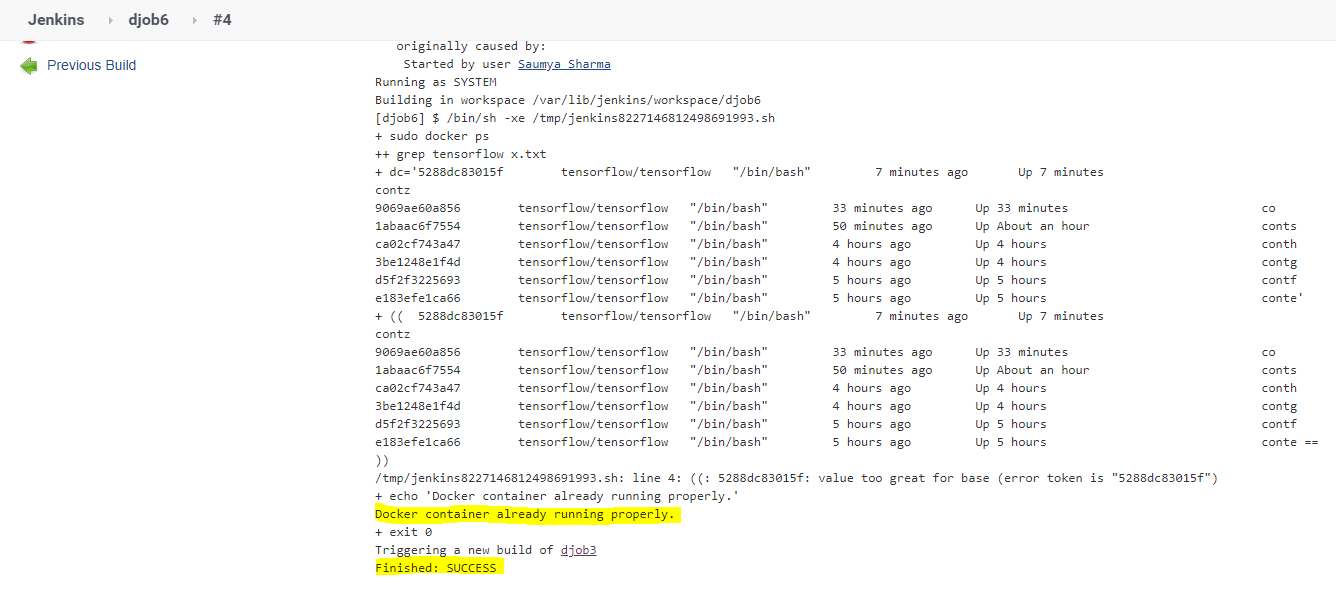
**JOB 3 executes:**



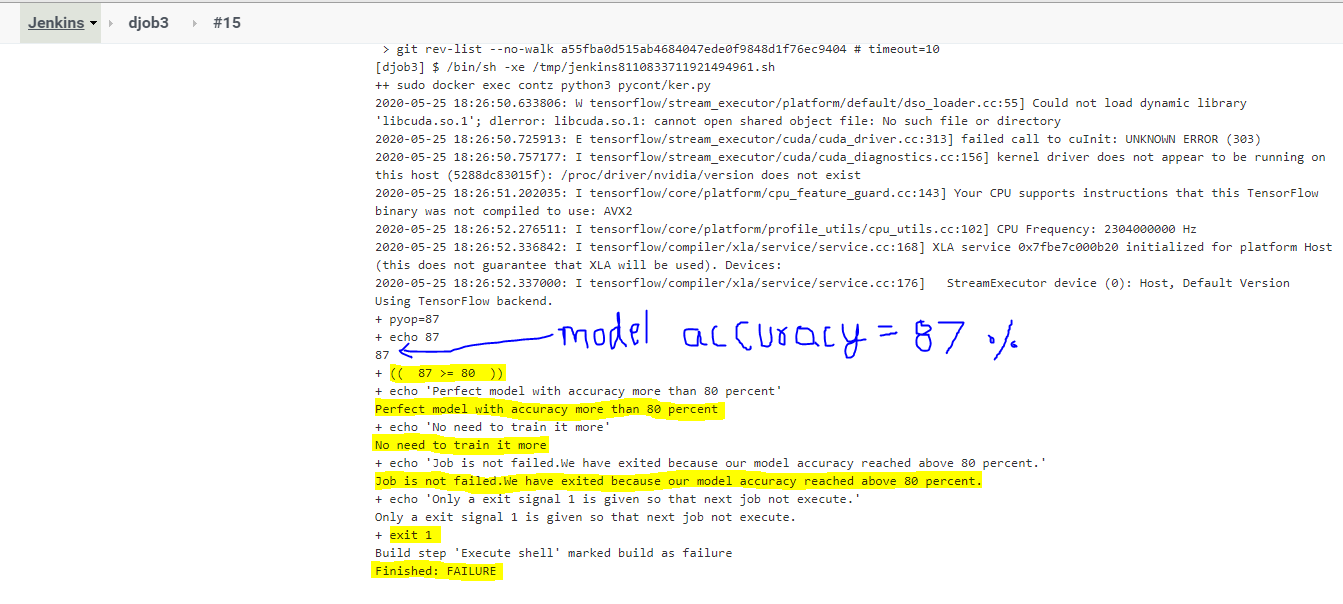
**JOB 4 executes:**



**JOB 5 executes:**



**JOB 3 executes again:**



**Our build pipeline:**

