MLOPS Pipeline Jenkins :

This article covers various ways of deploying Jenkins on your server. Use our up-to-date guide for manual installation or launching Jenkins in Docker containers. As a bonus, we'll show how to deploy Jenkins using CloudFormation. Deploy Jenkins on your server in 5 minutes using our guidance.

What is Jenkins ?

Jenkins is the most popular free automation server. You can use Jenkins to automate almost any processes in your company. But the traditional market for Jenkins is the automation of the software development processes.

Jenkins benefits:

- Written in Java, and thus you can run it on all platforms
- Has lots of plugins, which can help you solve almost any problem you can imagine.
- Helps developers to integrate code changes and always produce stable software build

Install Jenkins Manually

The installation process is simple and straightforward. It consists of several easy steps:

- Java installation.
- Adding official Jenkins repositories to your package manager.
- Package installation.
- Firewall configuration.

- Initial Jenkins configuration.

Let’s do them one by one.

Java installation

To be able to run Jenkins, you need to install Java first.

First, update the apt package index using this:

```
$ sudo apt update
```

Next, install the default Java OpenJDK package using this:

```
$ sudo apt install openjdk-8-jre
```

Verify the installation using this command:

```
$ java -version
```

Important: you should see that Java version

By the time of this article writing, Jenkins is using the Java 8 version. If you see another version, switch to the correct one:

```
$ sudo update-alternatives --config java
```

Add Jenkins repository

By default, Jenkins is not included in the Ubuntu repositories, and you need to add it.

First, we need to import the repository key:

```
$ wget -q -O - https://pkg.jenkins.io/debian-
stable/jenkins.io.key | sudo apt-key add -
```

Next step is to append the repository address to the server's repositories list:

```
$ sudo sh -c 'echo deb https://pkg.jenkins.io/
debian-stable binary/ > \
    /etc/apt/sources.list.d/jenkins.list'
```

Once that is done, we need to update the information about the software, which we can use from the new repository:

```
$ sudo apt-get update
```

Jenkins Package Installation

Since we have the repository up to date, let's run the following command:

```
$ sudo apt-get -y install jenkins
```

Running this command will prompt you to confirm the download and installation.

By default, Jenkins starts after installation process. If not, use the following command to do it manually:

```
$ sudo systemctl start jenkins
```

Let check if Jenkins service status:

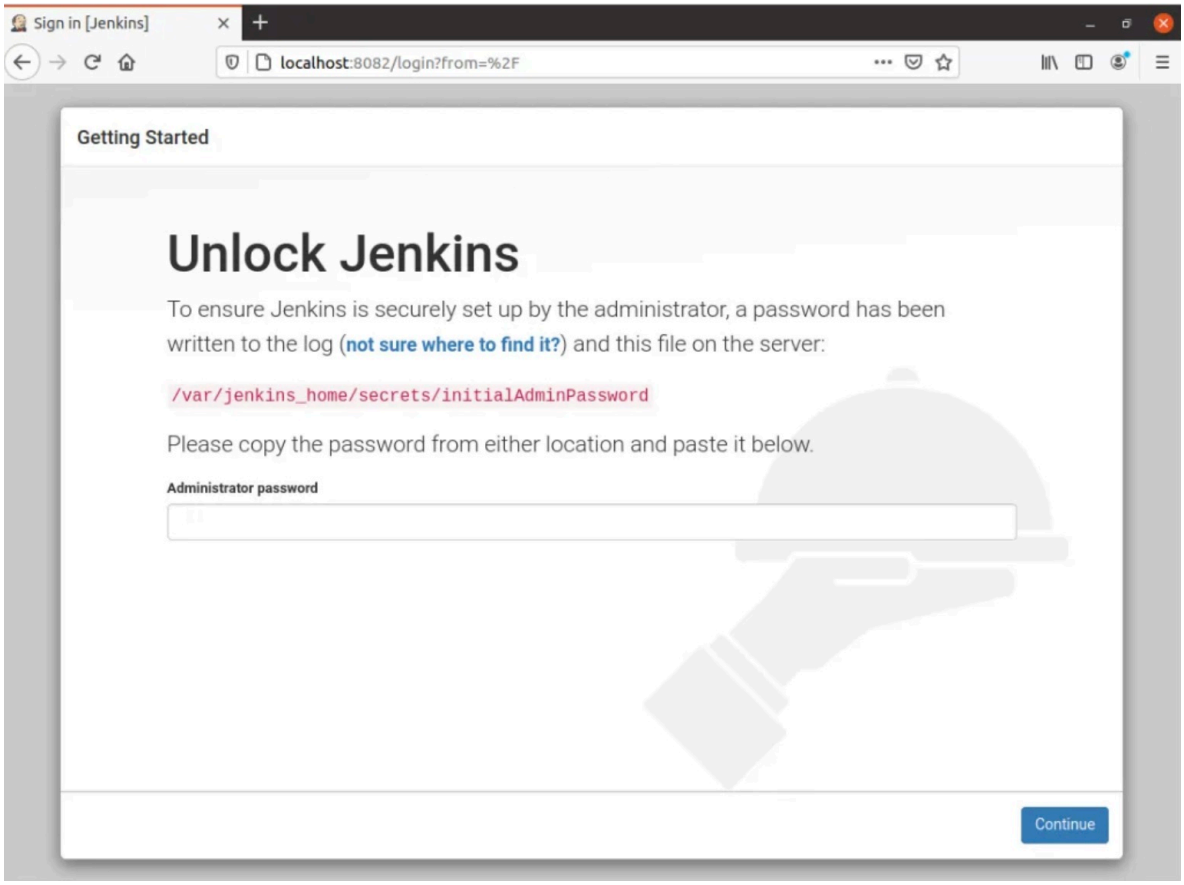
```
$ sudo systemctl status jenkins
```

By default, Jenkins runs on port **8080**.

So this can be done through enabling security group and inbound 8080 port for TCP connection on AWS.

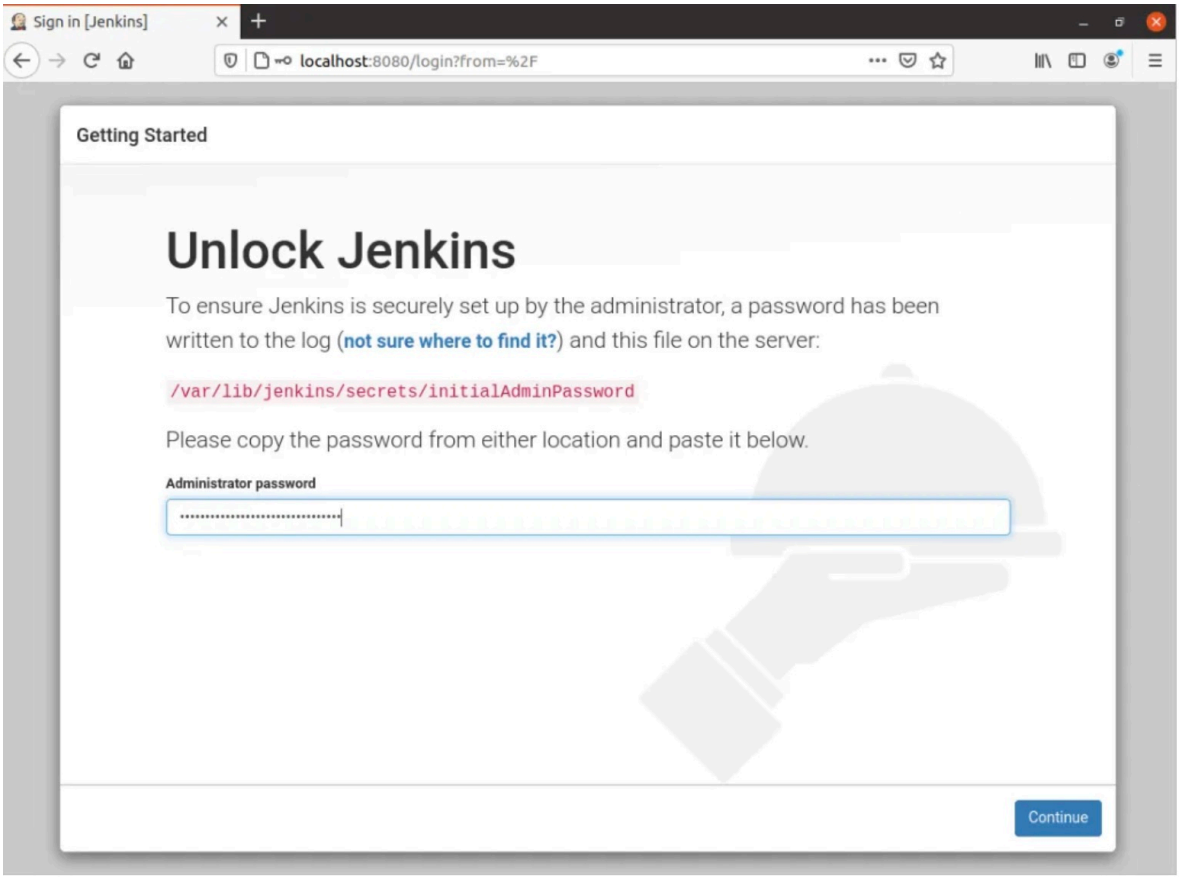
Then copy your AWS Instance IPV4 :
`ubuntu@IP.regrion_name.compute.amazonaws.com:8080`

This command results to opening the Jenkins docker container on port **8080.**



Initial password is available here:

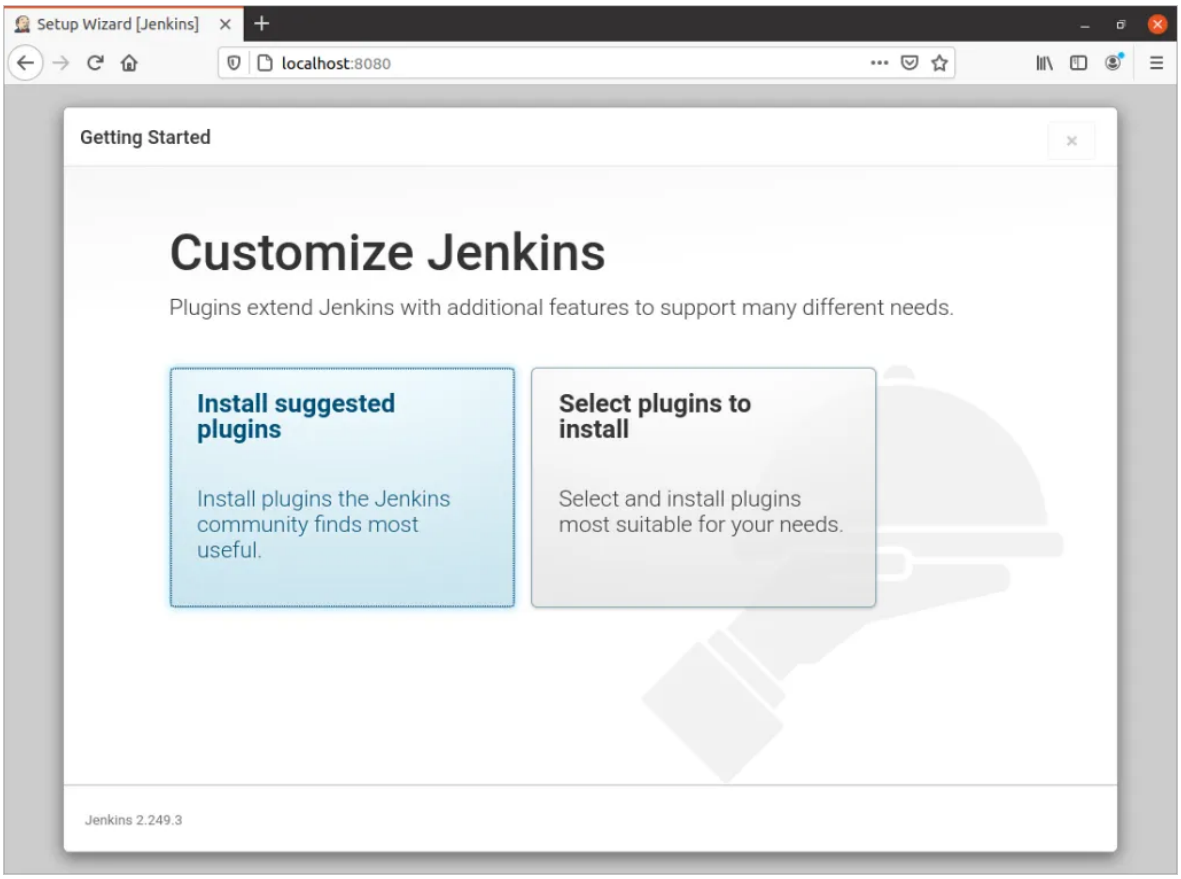
```
$ sudo cat /var/lib/jenkins/secrets/
initialAdminPassword
```



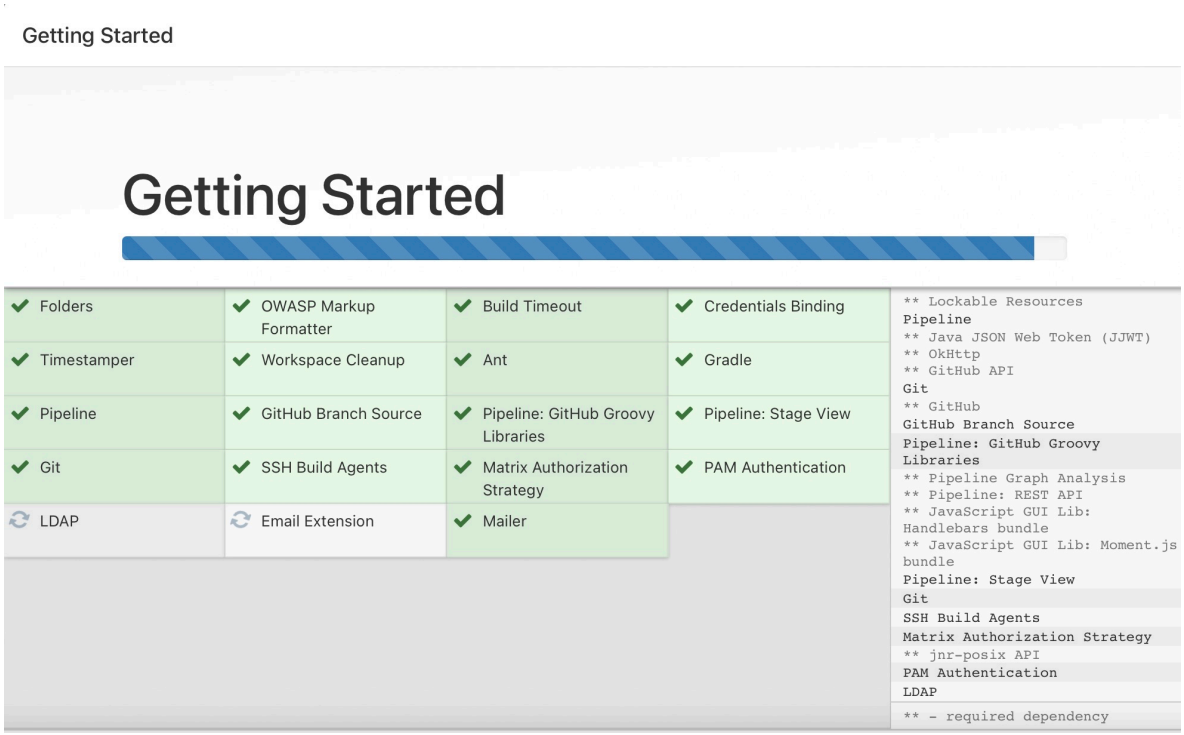
Jenkins Plugins

At the next screen, you will be required to choose plugins for initial installation. If you do not know which plugins to set up

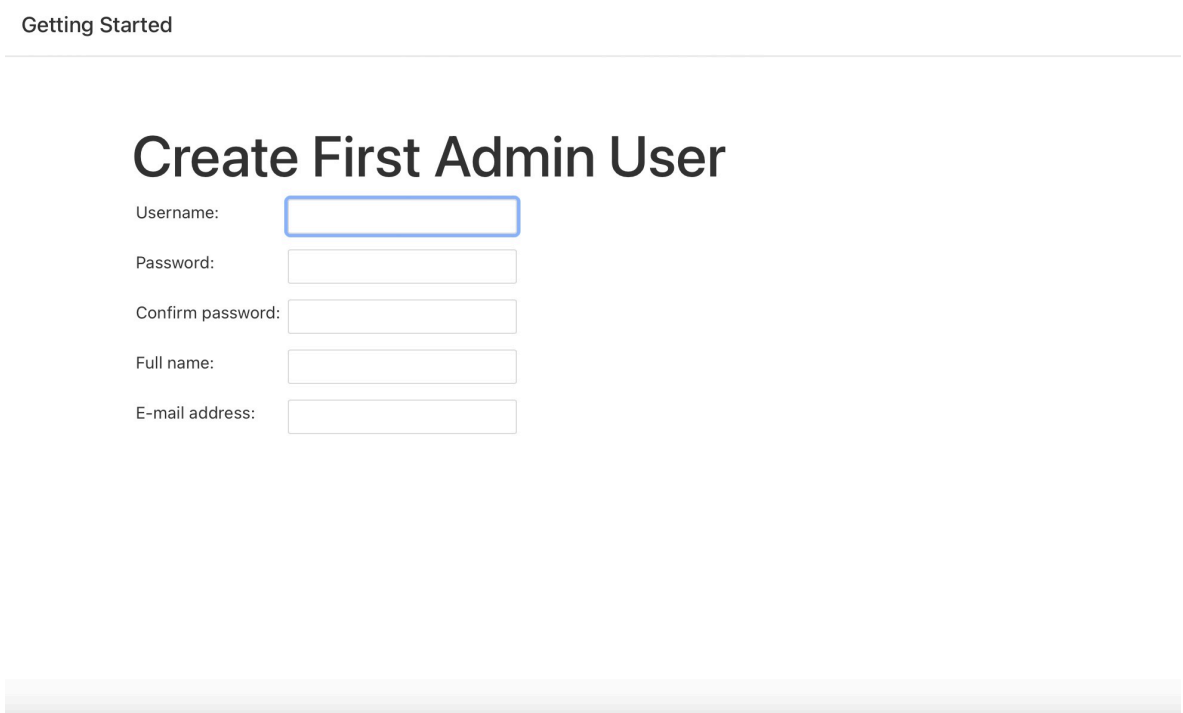
yet, use the “**Install suggested plugins**” option



The below is the screen you will be prompted after clicking on Install Suggested plugins.

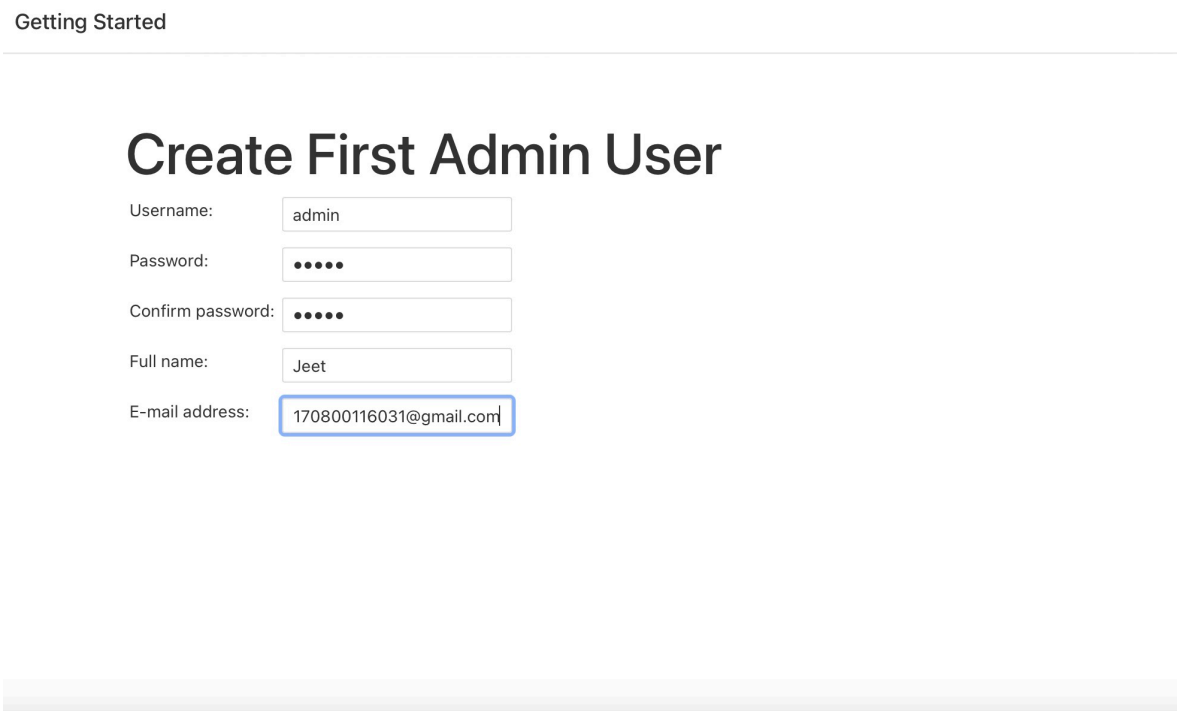


Thereafter you will be prompted with the below screen.

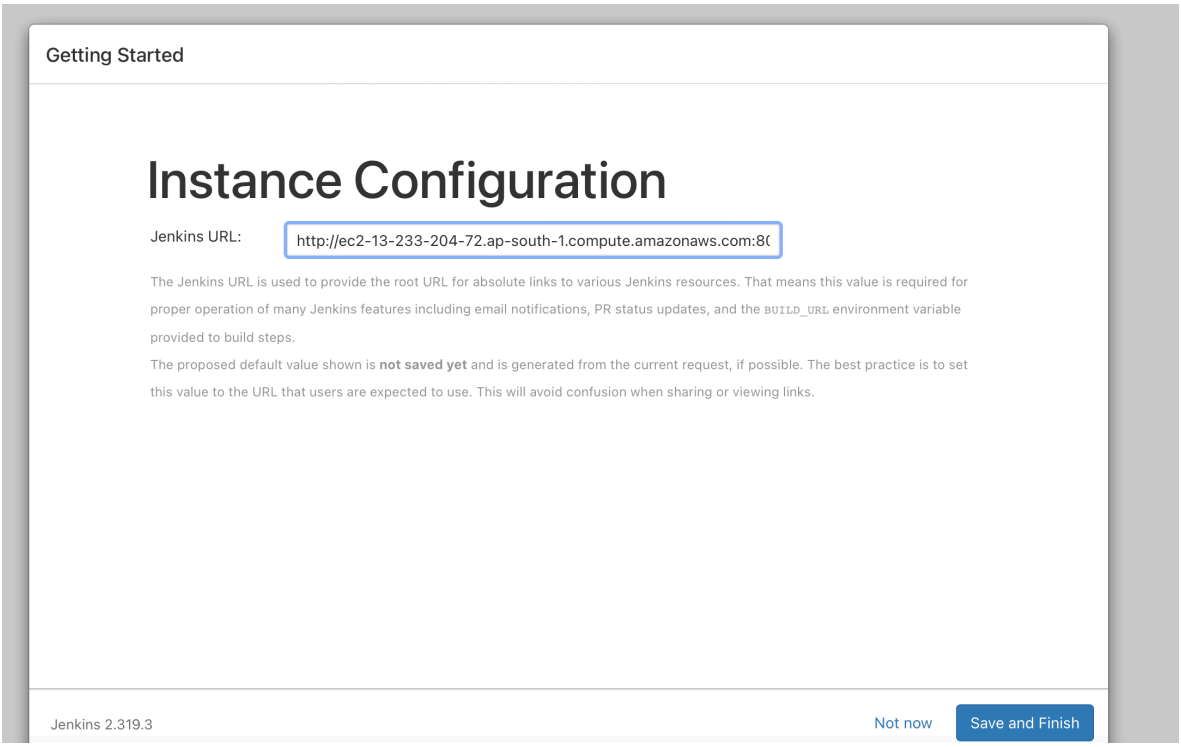


You need to create your Jenkins user

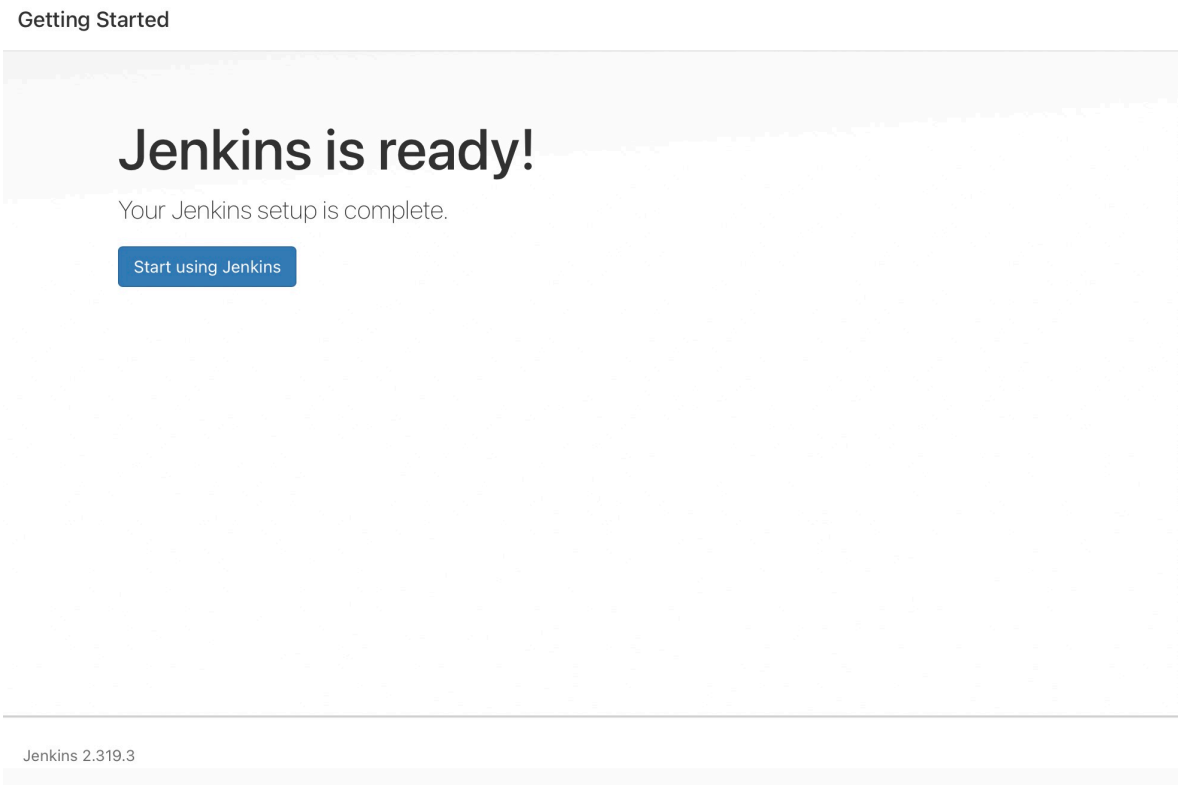
account to get the Jenkins dashboard.



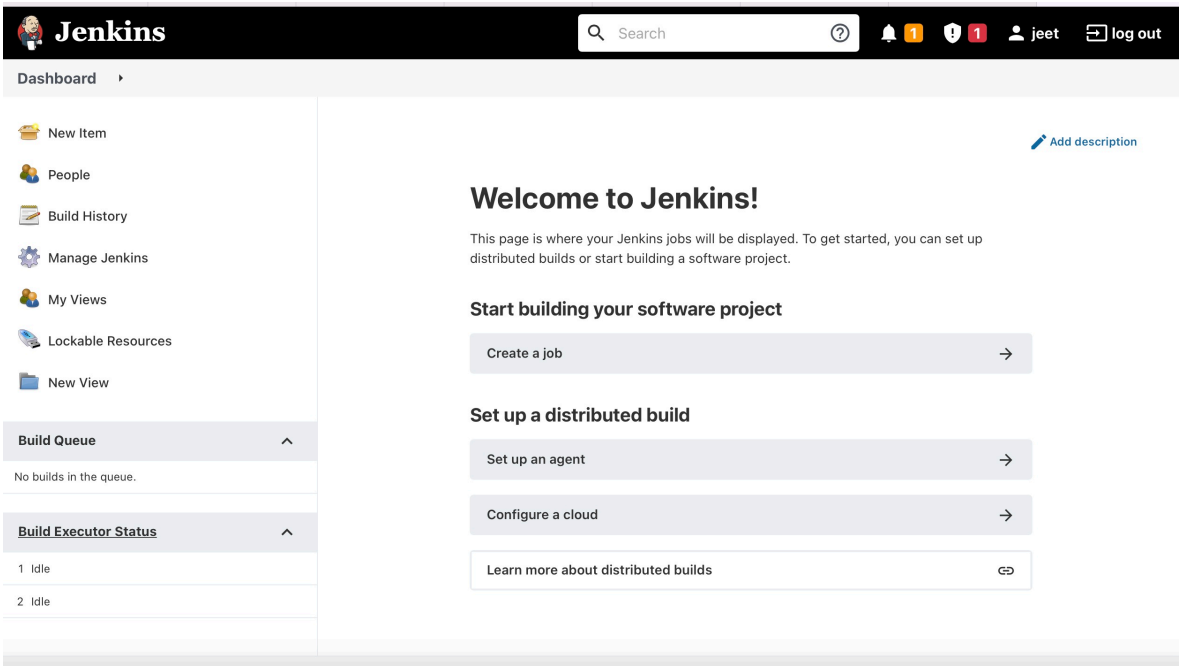
After successfully entering the user details, you will be directed to the instance configuration page to confirm the URL for the Jenkins instance.



There after click on save and finish button. The below screen will appear.



To access the Jenkins main dashboard, click the “**Start using Jenkins**” button. This action will bring you to the main Jenkins interface



To run MLOps pipeline in Jenkins you need to have Docker installed because in docker we have code dependencies.

The complete installation process is described in the Docker official documentation, so we'll provide the necessary commands here:

```
$ sudo apt-get install \
    apt-transport-https \
    ca-certificates \
    curl \
    gnupg-agent \
    software-properties-common


$ curl -fsSL https://download.docker.com/linux/
ubuntu/gpg | sudo apt-key add -

$ sudo add-apt-repository \
   "deb [arch=amd64] https://download.docker.com/
linux/ubuntu \
   $(lsb_release -cs) \
   stable"
$ sudo apt-get update

$ sudo apt-get install docker-ce docker-ce-cli
containerd.io
```

Once Docker is successfully installed, you can run the Docker container.

Todays In Machine Learning world hyperparameter optimization we can say tuning is the problem of choosing a set of optimal Hyperparameter for a machine learning

algorithms. Hyperparameter is a parameter which value is used to control the learning process. By contrast, the values of other parameters (typically node weights) are learned or in other words Hyper Parameters are those parameters which are assigned before the building of the model like number of filters , size of filter , number of neurons , number of epochs etc. The same kind of machine learning model can require different constraints, weights or learning rates to generalize different data patterns. These measures are called hyperparameters, and have to be tuned so that the model can optimally solve the machine learning problem. So to deal with this problem we can automate it where the hyper parameters will change and will make the accuracy of the model to the desire of the developer. So, instead of manually tuning our Hyperparameters, We are giving this job to Jenkins for Automation.

Work flow of Process or we say in term of jenkins automation jobs.

Sudo visudo

```
  GNU nano 2.9.3                                /etc/sudoers.tmp

#
# This file MUST be edited with the 'visudo' command as root.
#
# Please consider adding local content in /etc/sudoers.d/ instead of
# directly modifying this file.
#
# See the man page for details on how to write a sudoers file.
#
Defaults        env_reset
Defaults        mail_badpass
Defaults        secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/bin"

# Host alias specification

# User alias specification

# Cmnd alias specification

# User privilege specification
root    ALL=(ALL:ALL) ALL

# Members of the admin group may gain root privileges
%admin ALL=(ALL) ALL
jenkins ALL=(ALL) NOPASSWD: ALL
# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL
# See sudoers(5) for more information on "#include" directives:

#includedir /etc/sudoers.d

                              [ Read 29 lines ]
^G Get Help     ^O Write Out    ^W Where Is     ^K Cut Text     ^J Justify      ^C Cur Pos      M-U Undo
^X Exit         ^R Read File    ^\ Replace      ^U Uncut Text   ^T To Spell     ^_ Go To Line   M-E Redo
```

sudo vi /etc/default/jenkins

```
# defaults for Jenkins automation server

# pulled in from the init script; makes things easier.
NAME=jenkins

# arguments to pass to java

# Allow graphs etc. to work even when an X server is present
JAVA_ARGS="-Djava.awt.headless=true"

#JAVA_ARGS="-Xmx256m"

# make jenkins listen on IPv4 address
#JAVA_ARGS="-Djava.net.preferIPv4Stack=true"

PIDFILE=/var/run/$NAME/$NAME.pid

# user and group to be invoked as (default to jenkins)
JENKINS_USER="root"
JENKINS_GROUP=$NAME

# location of the jenkins war file
JENKINS_WAR=/usr/share/$NAME/$NAME.war

# jenkins home location
JENKINS_HOME=/var/lib/$NAME

# set this to false if you don't want Jenkins to run by itself
# in this set up, you are expected to provide a servlet container
# to host jenkins.
RUN_STANDALONE=true

# log location.  this may be a syslog facility.priority
"/etc/default/jenkins" 79L, 2776C                                    19,19         Top
```

sudo service jenkins restart

Now we should get into EC2 instance and pull GitHub repository of our Machine Learning project.

```
$ git clone https://github.com/aniforverizon/MLOPS-Pipeline-Jenkins.git
```

Then apply the following commands

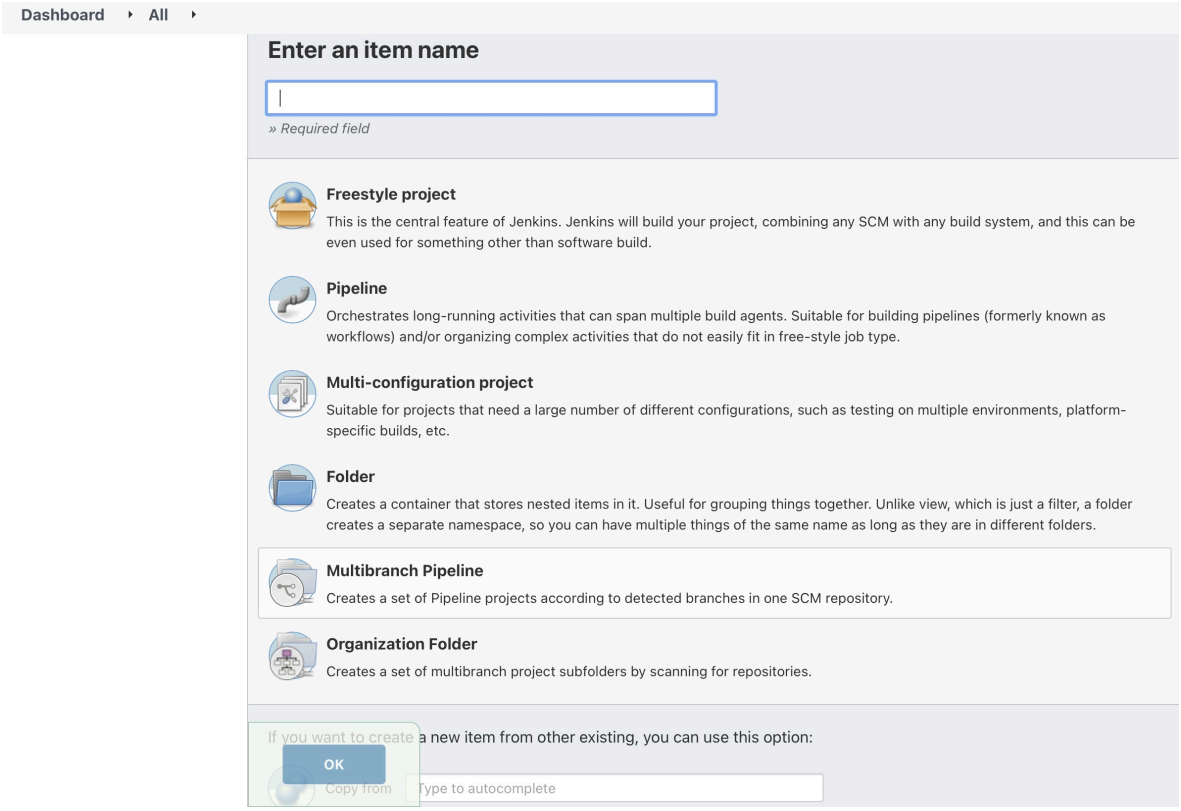
```
$ mkdir MLOPS-project
$ cd MLOPS-project/
$ ls

$ sudo cp -R * /mlops/

$ sudo mkdir /mlops



$ sudo cp -R * /mlops/
$ cd /mlops/
$ ls
$ sudo usermod -aG Docker $USER
$ newgrp docker
$ sudo chmod 777 *
```

First we will create a container image that has python installed (in addition to all the other libraries which will be necessary for a ML/DL code to run) using Dockerfile.

```
FROM centos

RUN cd /etc/yum.repos.d/

RUN sed -i 's/mirrorlist/#mirrorlist/g' /etc/yum.repos.d/CentOS-*
RUN sed -i 's|#baseurl=http://mirror.centos.org|baseurl=http://vault.centos.org|g' /etc/yum.repos.d/CentOS-*
RUN yum update -y

RUN yum install -y python36
RUN yum install -y epel-release
RUN yum groupinstall -y "development tools"
RUN yum install -y python36-devel
RUN pip3 install numpy
RUN pip3 install pandas
RUN pip3 install tensorflow
RUN pip3 install keras
RUN pip3 install --upgrade pip
RUN pip3 install setuptools
RUN pip3 install pillow
RUN pip3 install opencv-python
~
~
~
~
~
~
~
~
~
~
~
~
-- INSERT --                                                    17,31         All
```

Now build this dockerfile with new image name. so we run this command on ubuntu:18.04 ” docker build -t your_imagename:your_tagname .”

Thereafter go to Jenkins dashboard and click on “**New Items**” then below screen will be prompted.



Here enter the name of the job and click on **Freestyle project** then click on **OK**

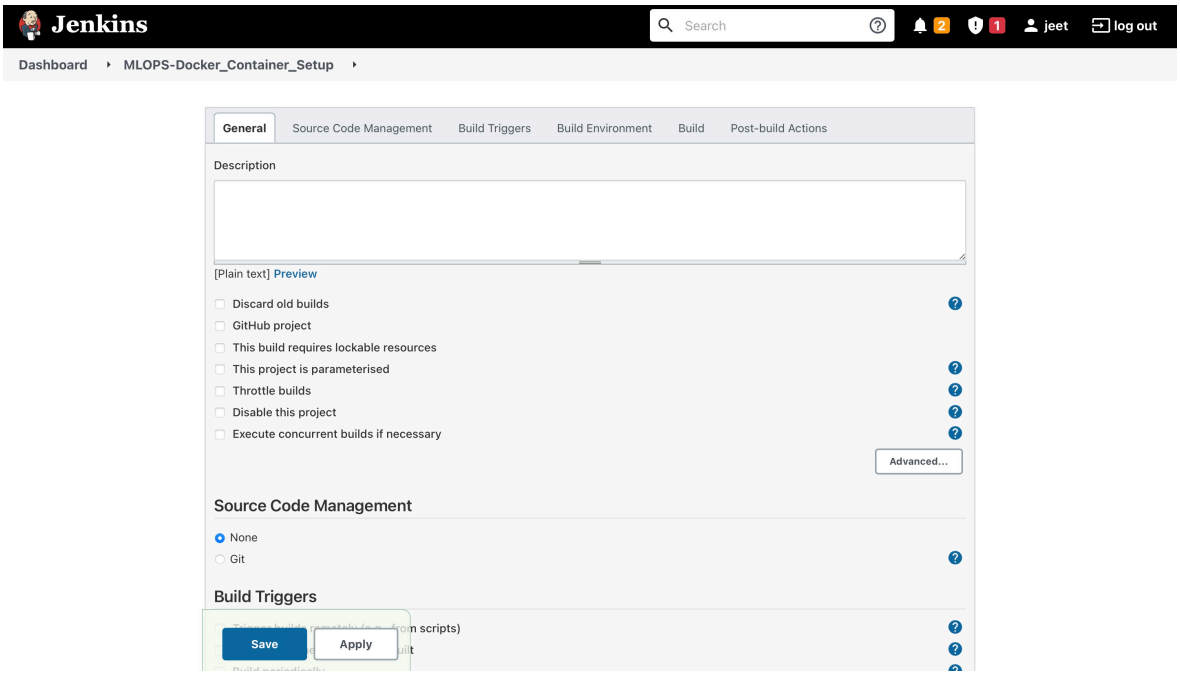
Now you need to execute docker container through Jenkins.

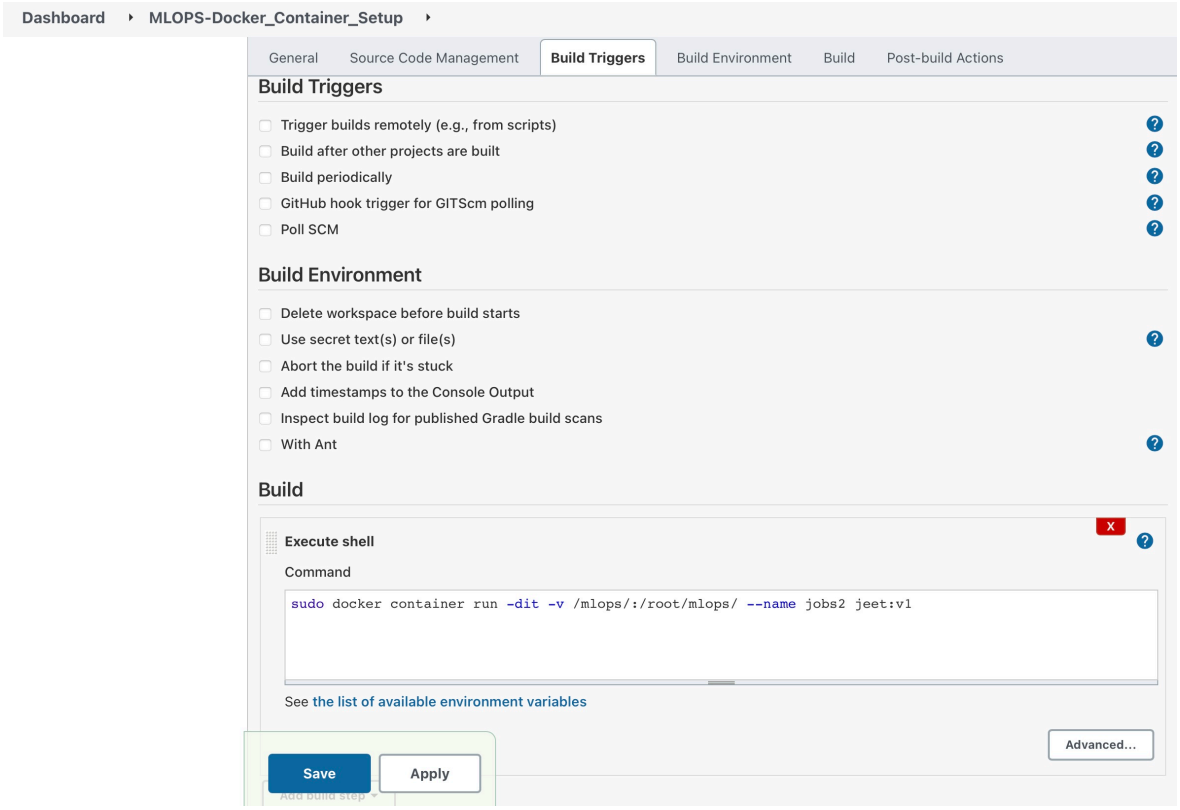
JOB 1 :-

Jenkins should automatically start the respective container on pre install libraries to deploy code and start training.

For that below is the process. Move to the Build Environment and select "**Execute shell**" enter the

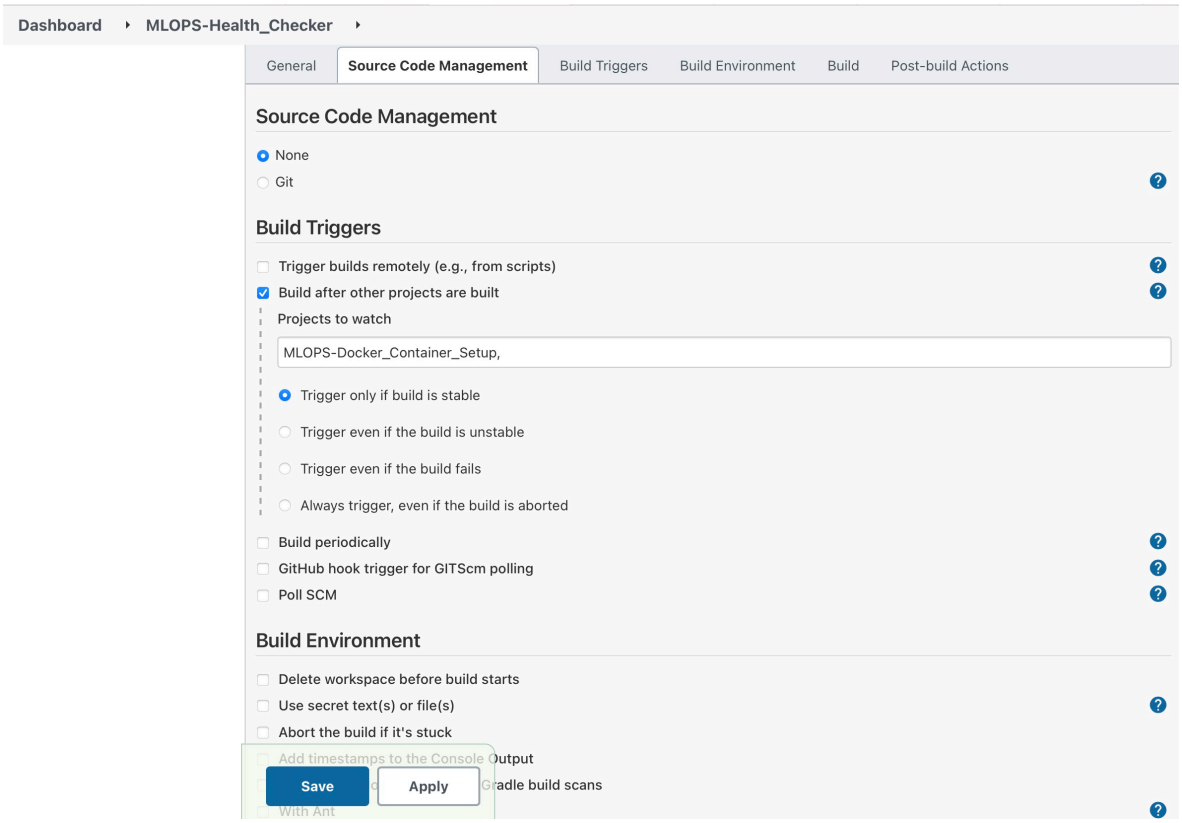
command to execute container

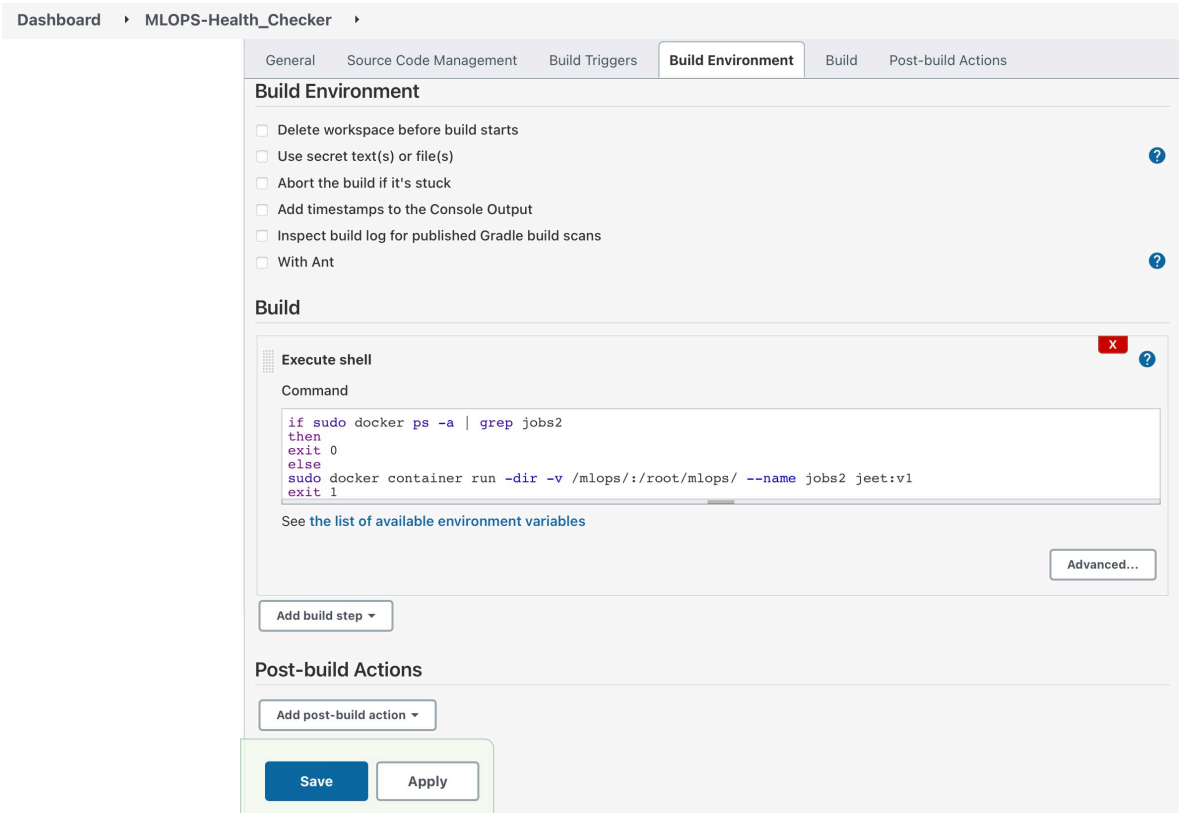




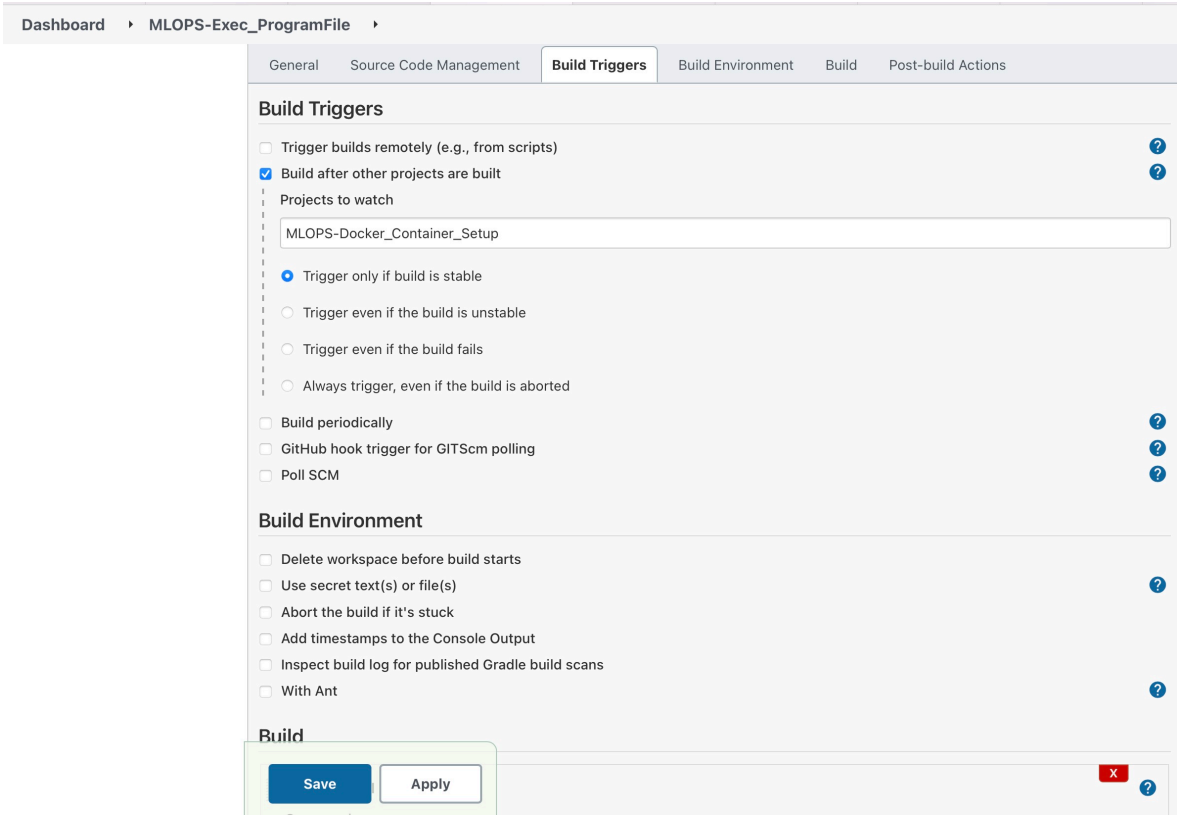
Job 2: This job will check the health of the container, if somehow container is stopped then it will restart the container.

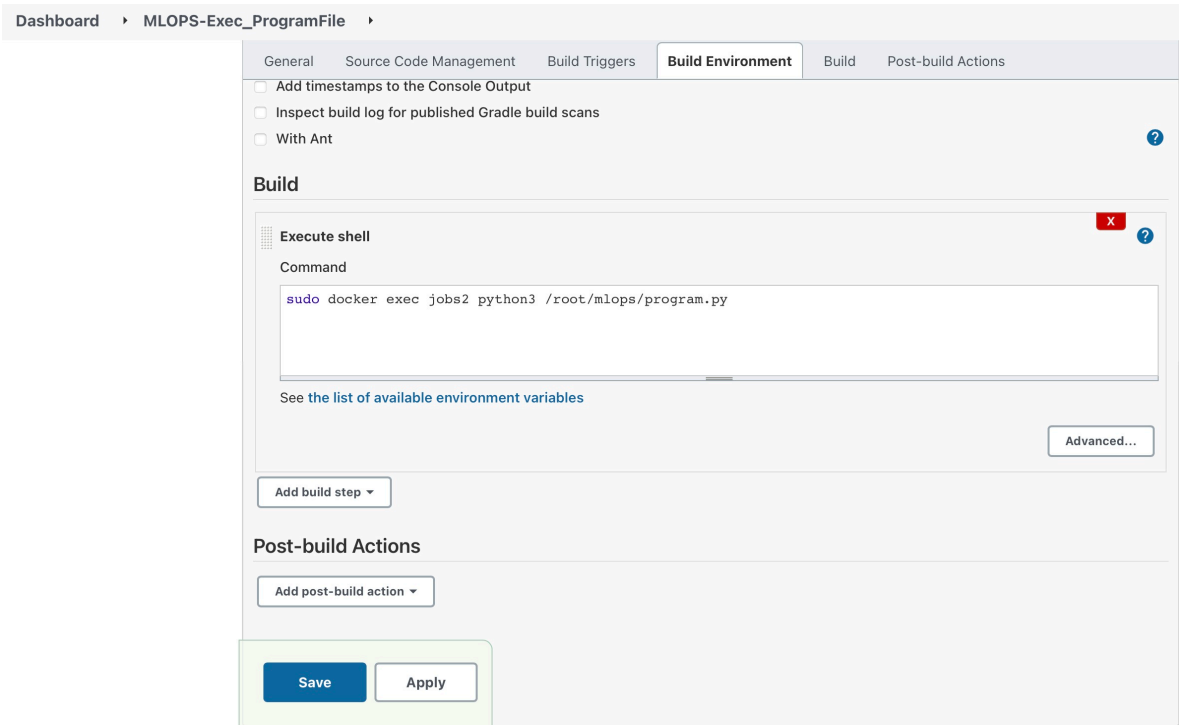
To do this again create a new item that will check the health of the container from Job1(MLOPS-Docker_Container_Setup). For that


Dashboard
MLOPS-Health_Checker
General
Source Code Management
Build Triggers
Build Environment
Build
Post-build Actions
Source Code Management
None
Git
Build Triggers
Trigger builds remotely (e.g., from scripts)
Build after other projects are built
Projects to watch
MLOPS-Docker_Container_Setup,
Trigger only if build is stable
Trigger even if the build is unstable
Trigger even if the build fails
Always trigger, even if the build is aborted
Build periodically
GitHub hook trigger for GITScm polling
Poll SCM
Build Environment
Delete workspace before build starts
Use secret text(s) or file(s)
Abort the build if it's stuck
Add timestamps to the Console Output
Save
Apply

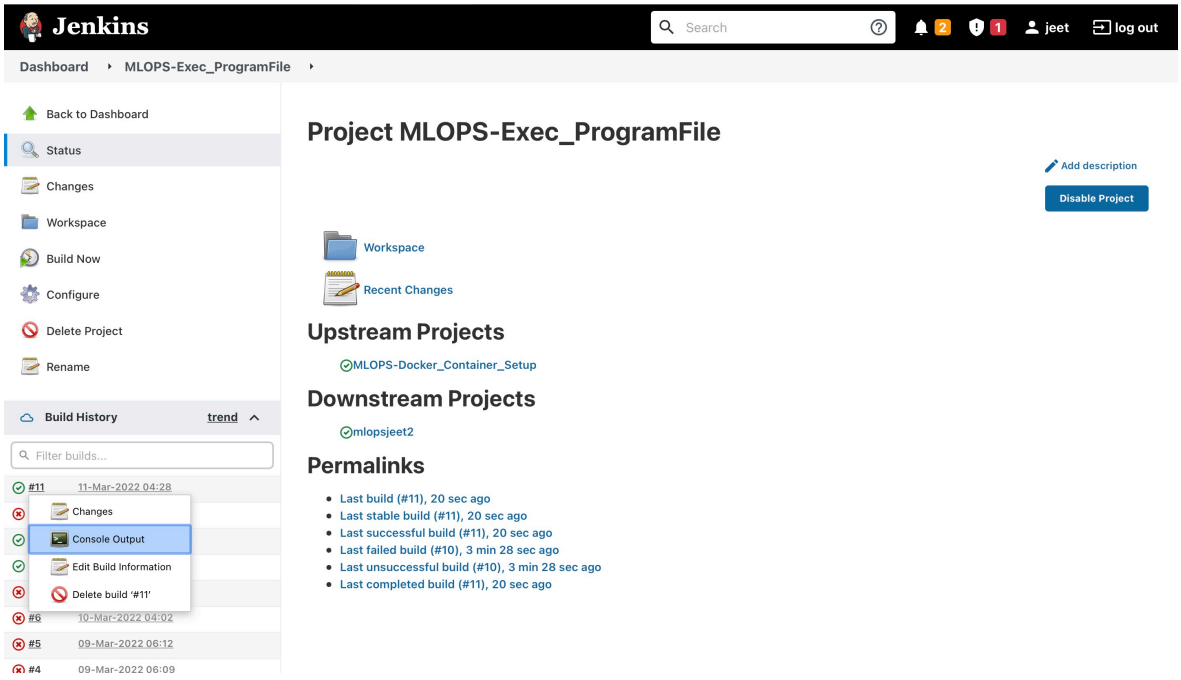

Dashboard
MLOPS-Health_Checker
General
Source Code Management
Build Triggers
Build Environment
Build
Post-build Actions
Build Environment
Delete workspace before build starts
Use secret text(s) or file(s)
Abort the build if it's stuck
Add timestamps to the Console Output
Inspect build log for published Gradle build scans
With Ant
Build
Execute shell
Command
if sudo docker ps -a | grep jobs2
then
exit 0
else
sudo docker container run -dir -v /mlops/:/root/mlops/ --name jobs2 jeet:v1
exit 1
See the list of available environment variables
Advanced...
Add build step
Post-build Actions
Add post-build action
Save
Apply

Job3 : This job will start the training for the Machine Learning model if and only if the docker container is in running mode.





You can check the output for the training process by the following process.



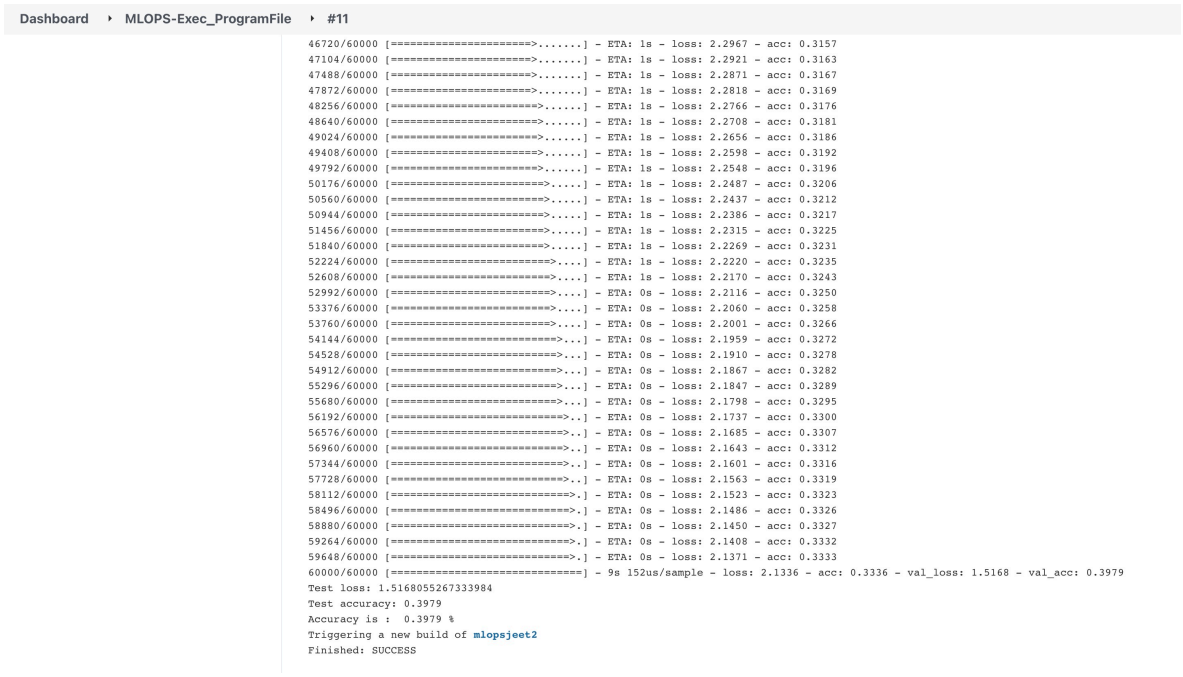
Back to Project
Status
Changes
Console Output
View as plain text
Edit Build Information
Delete build '#11'
Previous Build

Console Output

```
Started by user jeet
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/MLOPS-Exec_ProgramFile
[MLOPS-Exec_ProgramFile] $ /bin/sh -xe /tmp/jenkins6989810909729746277.sh
+ sudo docker exec jobs2 python3 /root/mlops/program.py
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz

    8192/11490434 [..............................] - ETA: 0s
 4202496/11490434 [=========>....................] - ETA: 0s
 9723904/11490434 [========================>.....] - ETA: 0s
11493376/11490434 [==============================] - 1s 0us/step
/usr/local/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:516: FutureWarning: Passing (type, 1) or '1type' as a
synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
/usr/local/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:517: FutureWarning: Passing (type, 1) or '1type' as a
synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
/usr/local/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:518: FutureWarning: Passing (type, 1) or '1type' as a
synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
/usr/local/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:519: FutureWarning: Passing (type, 1) or '1type' as a
synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
/usr/local/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:520: FutureWarning: Passing (type, 1) or '1type' as a
synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
/usr/local/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:525: FutureWarning: Passing (type, 1) or '1type' as a
synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
/usr/local/lib/python3.6/site-packages/tensorboard/compat/tensorflow_stub/dtypes.py:541: FutureWarning: Passing (type, 1) or 'ltype'
as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
/usr/local/lib/python3.6/site-packages/tensorboard/compat/tensorflow_stub/dtypes.py:542: FutureWarning: Passing (type, 1) or 'ltype'
```



```
2022-03-11 04:28:34.266052: I tensorflow/compiler/xla/service/service.cc:175]   StreamExecutor device (0): <undefined>, <undefined>
2022-03-11 04:28:34.347023: W tensorflow/compiler/jit/mark_for_compilation_pass.cc:1412] (One-time warning): Not using XLA:CPU for
cluster because envvar TF_XLA_FLAGS=--tf_xla_cpu_global_jit was not set.  If you want XLA:CPU, either set that envvar, or use
experimental_jit_scope to enable XLA:CPU.  To confirm that XLA is active, pass --vmodule=xla_compilation_cache=1 (as a proper
command-line flag, not via TF_XLA_FLAGS) or set the envvar XLA_FLAGS=--xla_hlo_profile.
size of image (28, 28)
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 26, 26, 8)         80
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 13, 13, 8)         0
_________________________________________________________________
flatten (Flatten)            (None, 1352)              0
_________________________________________________________________
dense (Dense)                (None, 10)                13530
_________________________________________________________________
dense_1 (Dense)              (None, 10)                110
=================================================================
Total params: 13,720
Trainable params: 13,720
Non-trainable params: 0
_________________________________________________________________
None
Train on 60000 samples, validate on 10000 samples

  128/60000 [..............................] - ETA: 1:57 - loss: 44.0682 - acc: 0.0859
  512/60000 [..............................] - ETA: 35s - loss: 31.1298 - acc: 0.0820
  896/60000 [..............................] - ETA: 23s - loss: 24.2630 - acc: 0.1083
 1280/60000 [..............................] - ETA: 19s - loss: 18.9010 - acc: 0.1266
 1664/60000 [..............................] - ETA: 16s - loss: 15.2765 - acc: 0.1460
 2048/60000 [>.............................] - ETA: 14s - loss: 12.9203 - acc: 0.1489
 2432/60000 [>.............................] - ETA: 13s - loss: 11.2406 - acc: 0.1530
 2816/60000 [>.............................] - ETA: 12s - loss: 10.0148 - acc: 0.1499
 3200/60000 [>.............................] - ETA: 12s - loss: 9.0761 - acc: 0.1541
 3584/60000 [>.............................] - ETA: 11s - loss: 8.3391 - acc: 0.1554
 3968/60000 [>.............................] - ETA: 11s - loss: 7.7470 - acc: 0.1552
 4352/60000 [=>............................] - ETA: 10s - loss: 7.2563 - acc: 0.1560
 4736/60000 [=>............................] - ETA: 10s - loss: 6.8467 - acc: 0.1582
 5120/60000 [=>............................] - ETA: 10s - loss: 6.4913 - acc: 0.1621
```



Once the training process is done then here comes mailing service in the picture.

Job 4: This job will trigger the mail regarding your model’s accuracy. It will check the accuracy of the model, if it is below 90% then it will send mail to the developer that the accuracy doesn't match with the expected accuracy. And then it will trigger next job for tuning hyperparameters. It will also depend on the previous job.

Dashboard › MLOPS-Triggering_Mail ›

General | Source Code Management | **Build Triggers** | Build Environment | Build | Post-build Actions

Build Triggers

- [ ] Trigger builds remotely (e.g., from scripts)
- [x] Build after other projects are built
  - Projects to watch: MLOPS-Exec_ProgramFile
  - (•) Trigger only if build is stable
  - ( ) Trigger even if the build is unstable
  - ( ) Trigger even if the build fails
  - ( ) Always trigger, even if the build is aborted
- [ ] Build periodically
- [ ] GitHub hook trigger for GITScm polling
- [ ] Poll SCM

Build Environment

- [ ] Delete workspace before build starts
- [ ] Use secret text(s) or file(s)
- [ ] Abort the build if it's stuck
- [ ] Add timestamps to the Console Output
- [ ] Inspect build log for published Gradle build scans
- [ ] With Ant

Build

Save | Apply

Dashboard › MLOPS-Triggering_Mail ›

General | Source Code Management | Build Triggers | Build Environment | **Build** | Post-build Actions

- [ ] With Ant

Build

Execute shell

Command

```
if [[ "$(sudo cat /mlops/accuracy.txt)" < "0.9000"]]
then
echo "Accuracy not matched!!!"
sudo docker exec jobs1 python3 /root/mlops/mail.py
else
echo "Accuracy matched !!!"
sudo docker exec jobs1 python3 /root/mlops/mail_success.py
fi
```

See the list of available environment variables

Advanced...

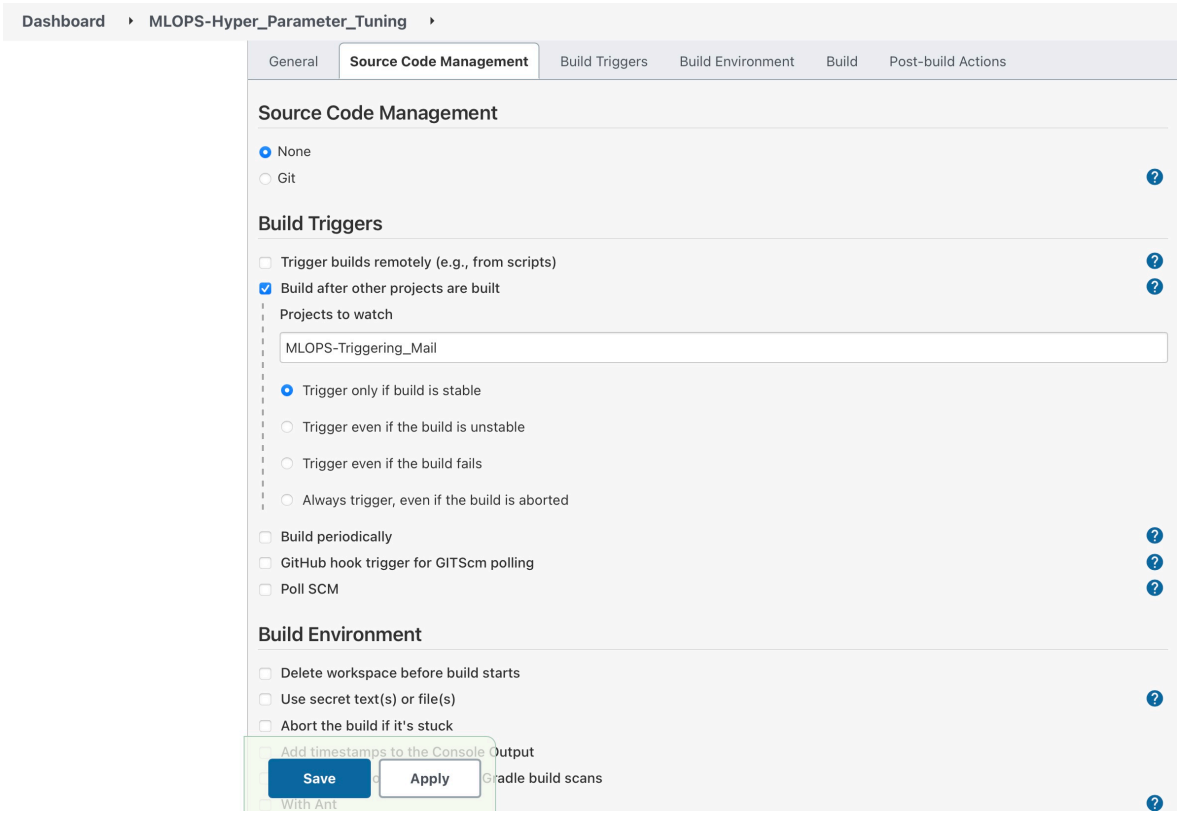
Add build step

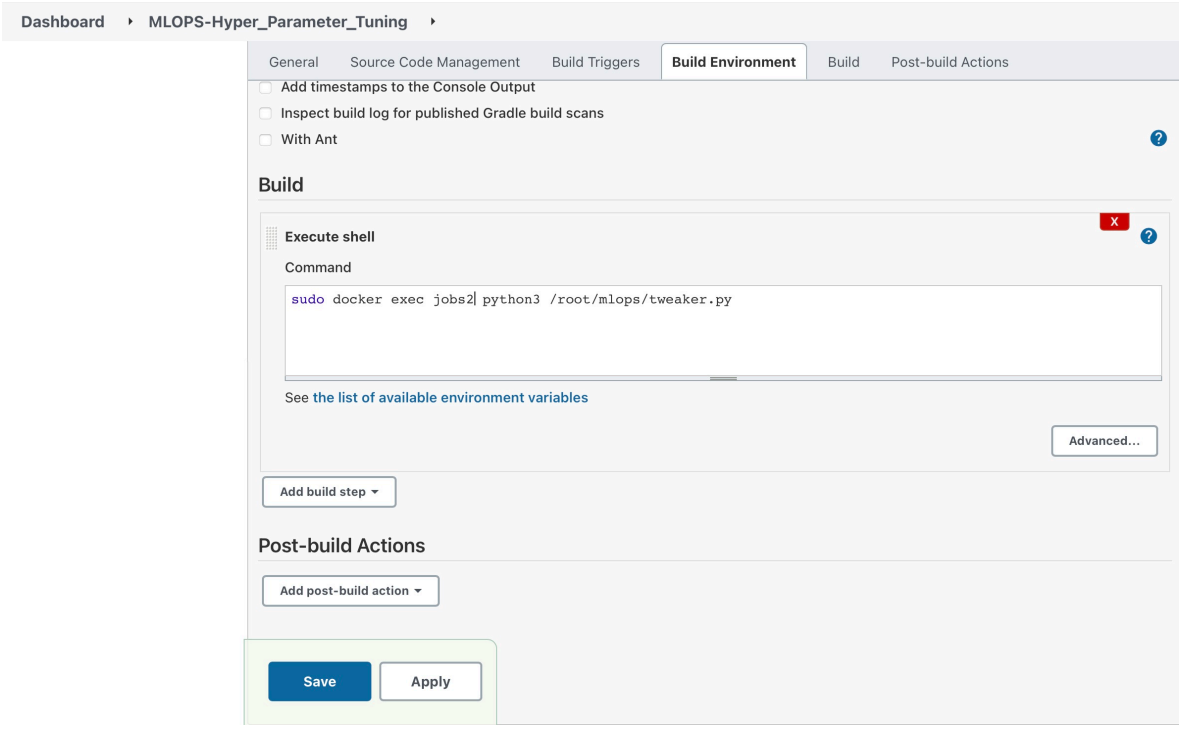
Post-build Actions

Add post-build action

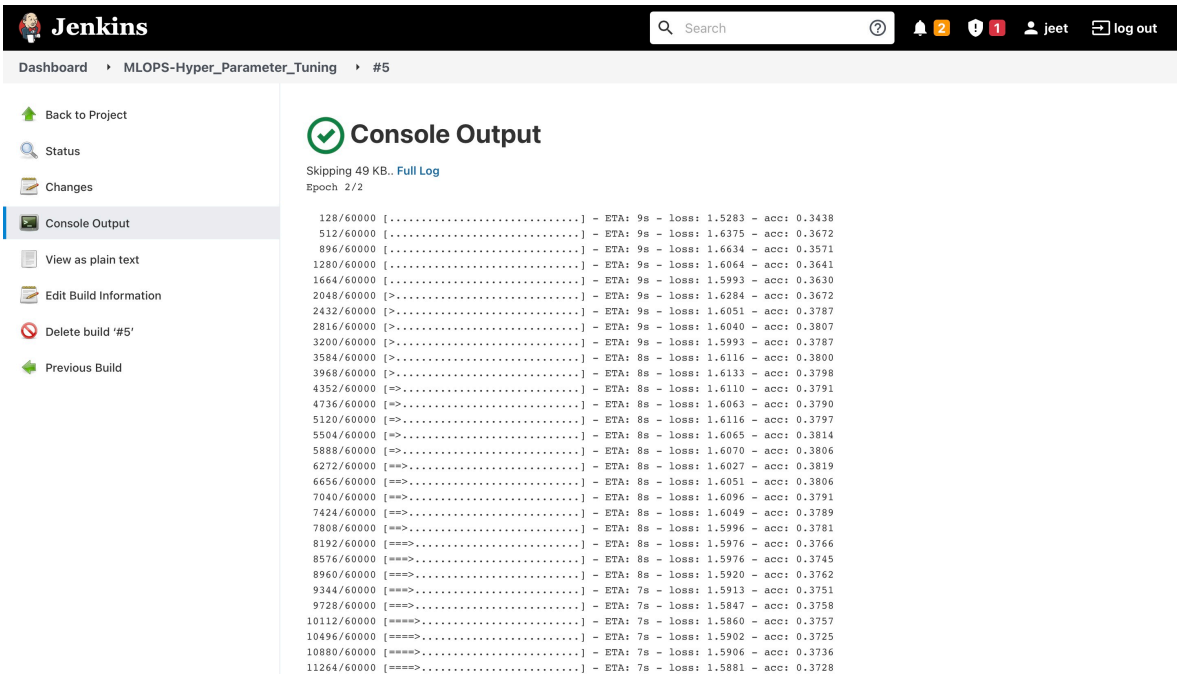
Save | Apply

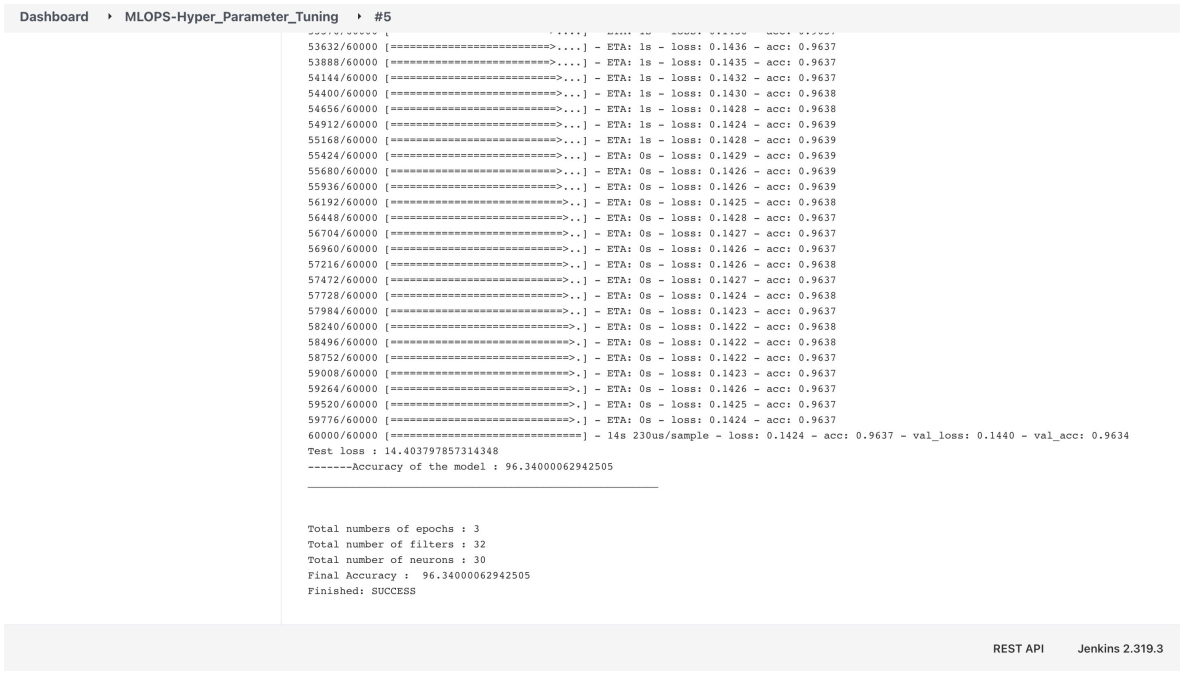
JOB 5 : This job will be responsible for hyper parameter tuning of the model and will make changes in the code in order to achieve the required accuracy. Then send the mail to developer regarding the accuracy.



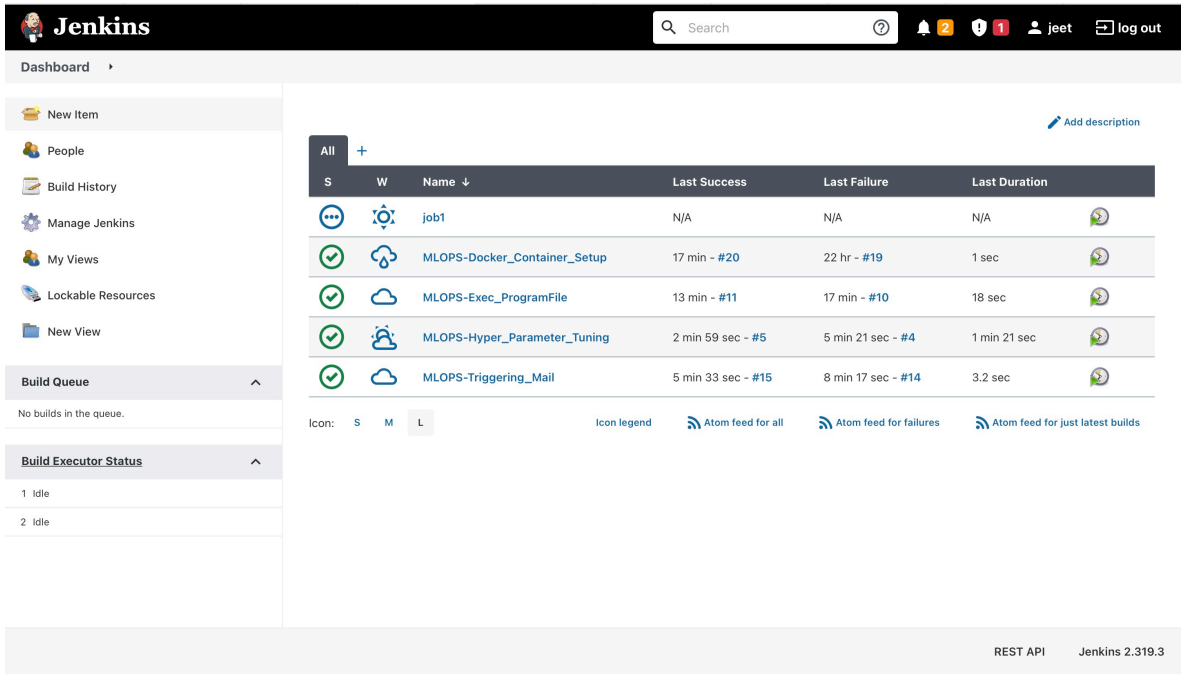


To check the accuracy of tuned model you can check the below process





Once the pipeline is done then you will get the below related dashboard .



Keep learning !!!!