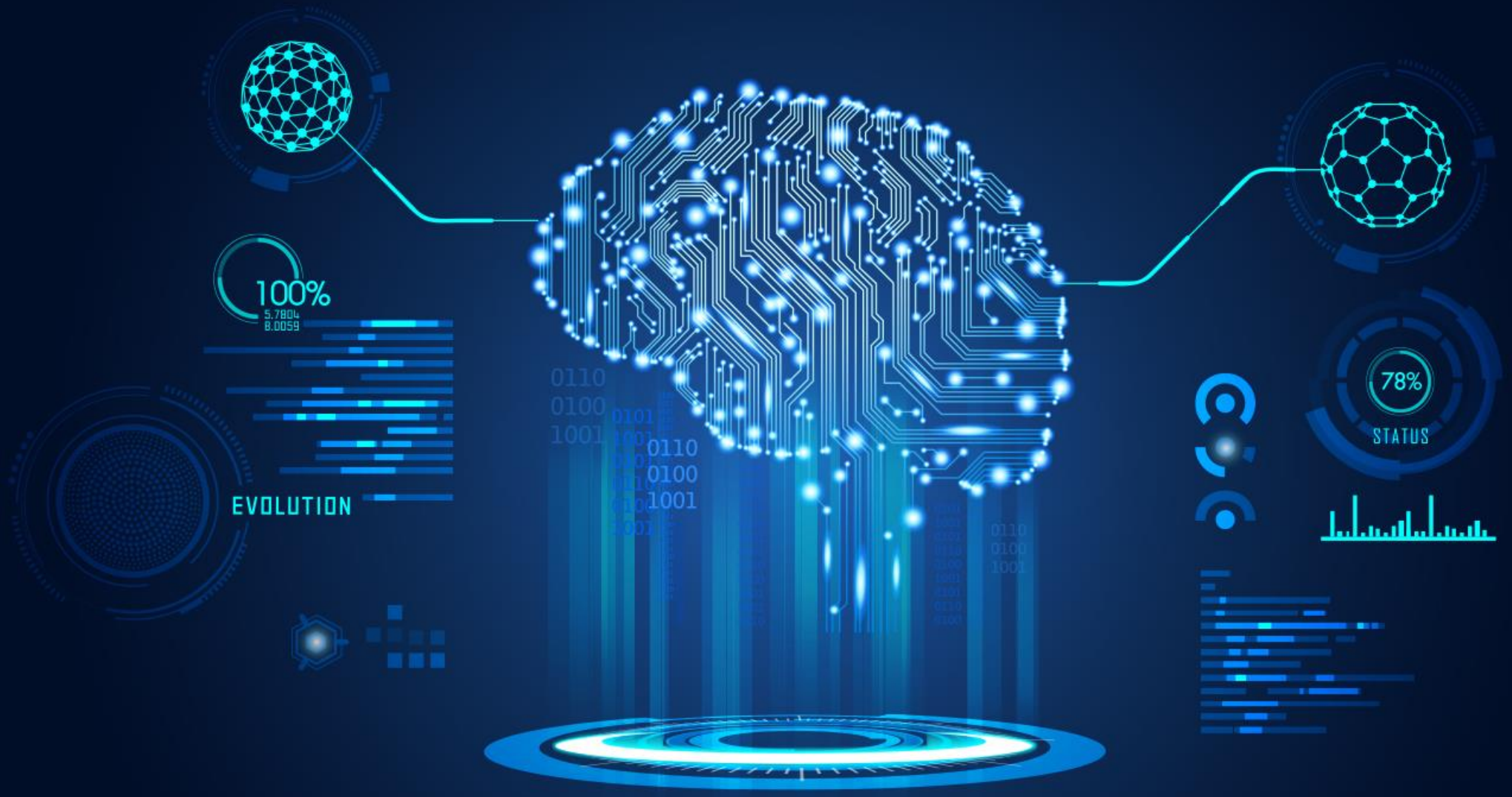
MLOPS TASK#3



TASK DESCRIPTION:

1. CREATE A CONTAINER IMAGE THAT HAS PYTHON3 AND KERAS OR NUMPY INSTALLED USING DOCKER FILE.

2. WHEN WE LAUNCH THIS IMAGE, IT SHOULD AUTOMATICALLY START TO TRAIN THE MODEL IN THE CONTAINER.

3. CREATE A JOB CHAIN OF JOB1, JOB2, JOB3, JOB4 AND JOB5 USING BUILD PIPELINE PLUGIN IN JENKINS

4. JOB1: PULL THE GITHUB REPO AUTOMATICALLY WHEN SOME DEVELOPERS PUSH THE REPO TO GITHUB.

5. JOB2: BY LOOKING AT THE CODE OR PROGRAM FILE, JENKINS SHOULD AUTOMATICALLY START THE RESPECTIVE MACHINE LEARNING SOFTWARE INSTALLED INTERPRETER INSTALL IMAGE CONTAINER TO DEPLOY CODE AND START TRAINING(EG. IF CODE USES CNN, THEN JENKINS SHOULD START THE CONTAINER THAT HAS ALREADY INSTALLED ALL THE SOFTWARE REQUIRED FOR THE CNN PROCESSING).

6. JOB3: TRAIN YOUR MODEL AND PREDICT ACCURACY OR METRICS.

7. JOB4: IF METRICS ACCURACY IS LESS THAN 80%, THEN TWEAK THE MACHINE LEARNING MODEL ARCHITECTURE.

8. JOB5: RETRAIN THE MODEL OR NOTIFY THAT THE BEST MODEL IS BEING CREATED.

9. CREATE ONE EXTRA JOB JOB6 FOR MONITORING: IF THE CONTAINER WHERE THE APP IS RUNNING, FAILS DUE TO ANY REASON THEN THIS JOB SHOULD AUTOMATICALLY START THE CONTAINER AGAIN FROM WHERE THE LAST TRAINED MODEL LEFT.

CREATING DOCKER IMAGE USING DOCKERFILE WITH PYTHON LIBRARIES INSTALLED:

Open ▾ | Dockerfile ~/TASK3 | Save

```
FROM centos:7

RUN yum install python36 -y

RUN pip3 install --upgrade pip
RUN pip3 install --upgrade pip setuptools
RUN pip3 install --upgrade pip setuptools wheel
RUN pip3 install --upgrade pip tools
RUN pip3 install --upgrade pip tools wheel

RUN pip3 install keras
RUN pip3 install tensorflow
RUN pip install opencv-python
RUN pip3 install numpy
RUN pip3 install pandas
RUN pip3 install scikit-learn

RUN mkdir task3code
```

- After creating the Dockerfile, save it in the folder named /TASK3
- Then, use #docker built –t task3image:v2 /TASK3/ and it will automatically download all the required libraries and create docker image.

FOLLOWING IS MY PYTHON CODE THAT WILL RUN ON THE DOCKER CONTAINER AND TRAIN MY MODEL

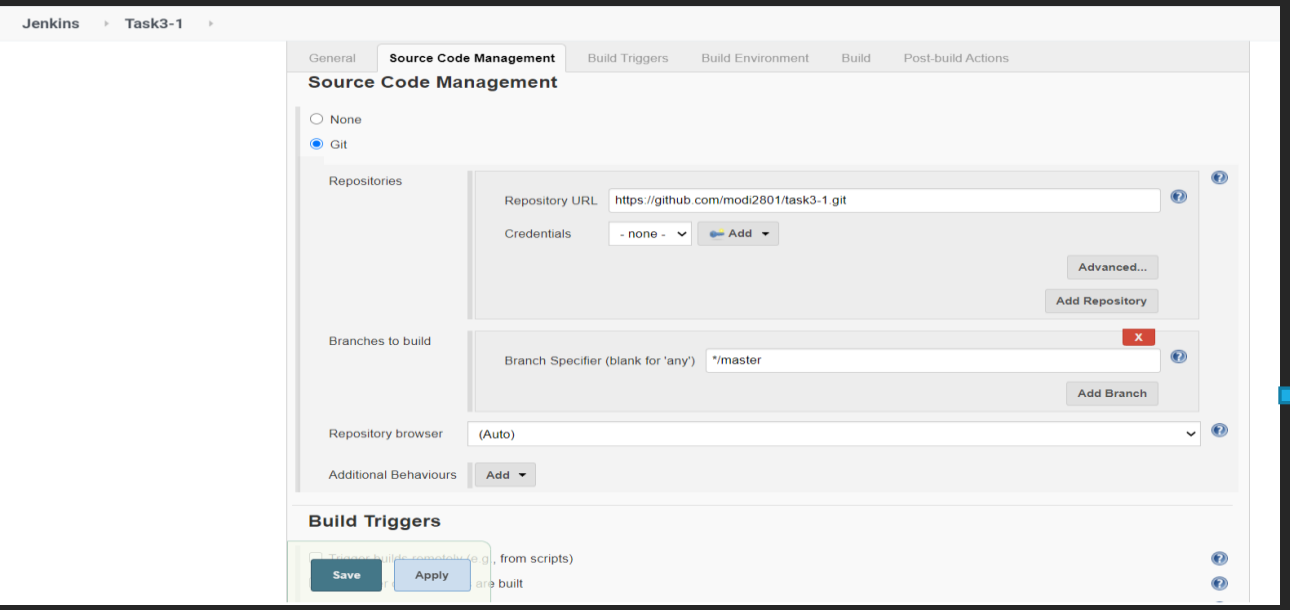
```python
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense
from keras.utils.np_utils import to_categorical
from keras.optimizers import RMSprop
from sklearn.metrics import accuracy_score
import numpy as np

(X_train, y_train) , (X_test, y_test) = mnist.load_data()

X_train = X_train.reshape(-1, 784)
X_test = X_test.reshape(-1, 784)
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')

y_train = to_categorical(y_train)
y_test = to_categorical(y_test)

model = Sequential()
model.add(Dense(units=10, input_shape=(784,), activation='relu'))
i=1
n=50
for i in range(i):
    model.add(Dense(units=n, activation='relu'))
    n+=50
model.add(Dense(units=10, activation='softmax'))

model.compile(optimizer=RMSprop(), loss='categorical_crossentropy', metrics=['accuracy'])
epoch = model.fit(X_train, y_train, epochs=1, validation_data=(X_test , y_test))

model.save('mnist_model_trained.h5')

accuracy = model.history.history.get('accuracy')
accuracy=accuracy[-1]*100

print(accuracy)

f = open("accuracy.txt", "w")
f.write(np.array2string(accuracy))
f.close()
```

- After creating the code, save it as mnist_model.py and push it in the Github.

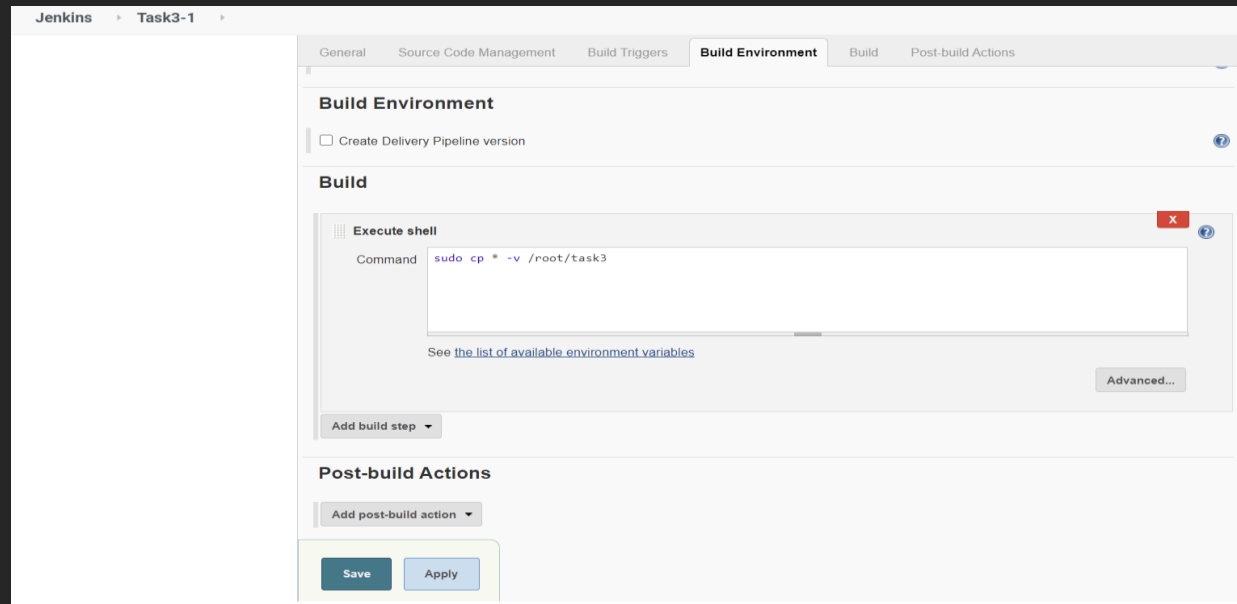
CONFIGURE YOUR JENKINS AND CREATE FIRST JOB:

FOR CONFIGURING THE JENKINS AND INTEGRATING IT WITH GITHUB, WE MUST HAVE GITHUB PLUGIN INSTALLED IN THE JENKINS.

AFTER INSTALLING THE GITHUB PLUGIN, CREATE 'JOB-1' THAT WILL DOENLOAD THE CODE FROM THE GITHUB AND COPY IT TO THE REDHAT OS.



- This Job will copy the code from the Github and paste it to the /root/task3 folder in Redhat.



CREATE SECOND JOB IN JENKINS THAT WILL BE TRIGGERED AFTER THE COMPLETION OF THE FIRST JOB.

Jenkins › Task3-2 ›

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

☐ Create Delivery Pipeline version

Build

Execute shell

Command

```
if sudo cat /root/task3/mnist_model.py | grep keras
then
    if sudo docker ps -a | grep mnist_container
    then
    sudo docker rm -f mnist_container
    sudo docker run -dit --name mnist_container task3image:v2
    sudo docker cp /root/task3/mnist_model.py mnist_container:/task3code/
    sudo docker exec mnist_container python3 /task3code/mnist_model.py
    sudo docker cp mnist_container:/accuracy.txt /root/task3/
    elif sudo docker ps | grep mnist_container
    then
    sudo docker rm -f mnist_container
    sudo docker run -dit --name mnist_container task3image:v2
    sudo docker cp /root/task3/mnist_model.py mnist_container:/task3code/
    sudo docker exec mnist_container python3 /task3code/mnist_model.py
    sudo docker cp mnist_container:/accuracy.txt /root/task3/
    else
    sudo docker run -dit --name mnist_container task3image:v2
    sudo docker cp /root/task3/mnist_model.py mnist_container:/task3code/
    sudo docker exec mnist_container python3 /task3code/mnist_model.py
    sudo docker cp mnist_container:/accuracy.txt /root/task3/
    fi
else
echo "This python program is not suitable for the created image"
fi
```

Save | Apply

- This Job will launch the docker container with the name mnist_container and launch mnist_model.py code in train that will train the model and save the accuracy of the model in accuracy.txt file. It will also copy accuracy.txt file in /root/task3/ directory of REDHAT.

CREATE THIRD JOB IN JENKINS THAT WILL BE TRIGGERED AFTER THE COMPLETION OF THE SECOND JOB.

Jenkins › Task3-3 ›

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

☐ Create Delivery Pipeline version

Build

Execute shell

Command

```
accuracy=$(sudo cat /root/task3/accuracy.txt)
x=2
filter=(i=$x)
while [ $accuracy -lt 95 ]
do
 sudo sed -i '21s/.*/'"$filter"'/' /root/task3/mnist_model.py
 sudo docker cp /root/task3/accuracy.txt mnist_container:/task3code/
 sudo docker cp /root/task3/mnist_model.py mnist_container:/task3code/
 sudo docker exec mnist_container python3 /task3code/mnist_model.py
 sudo docker cp mnist_container:/accuracy.txt /root/task3/
 accuracy=$(sudo cat /root/task3/accuracy.txt)
 x=$((x+1))
 filter=(i=$x)
done
```

See the list of available environment variables

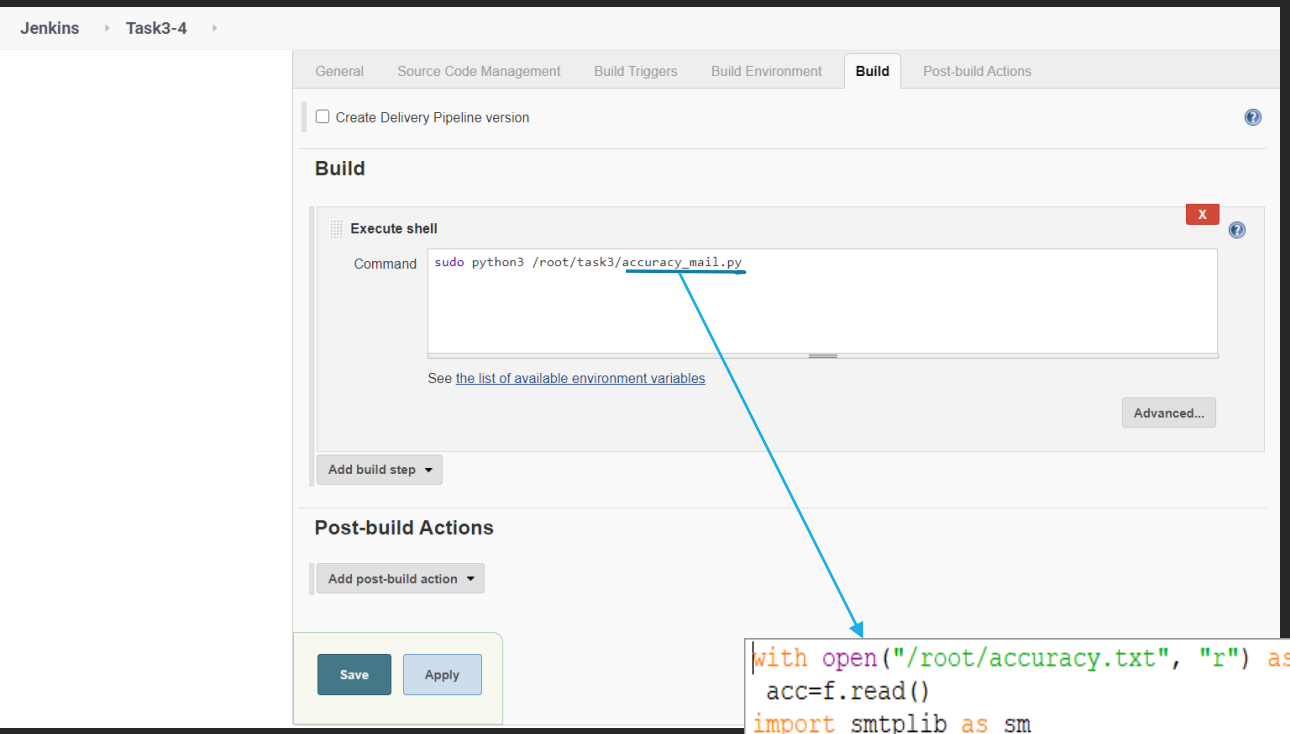
Advanced...

Add build step

Save | Apply

- This Job will check the accuracy of model and if the accuracy is less than 95%, then it will automatically change the number of filters and epochs in the model and train it again till 95% accuracy is achieved.

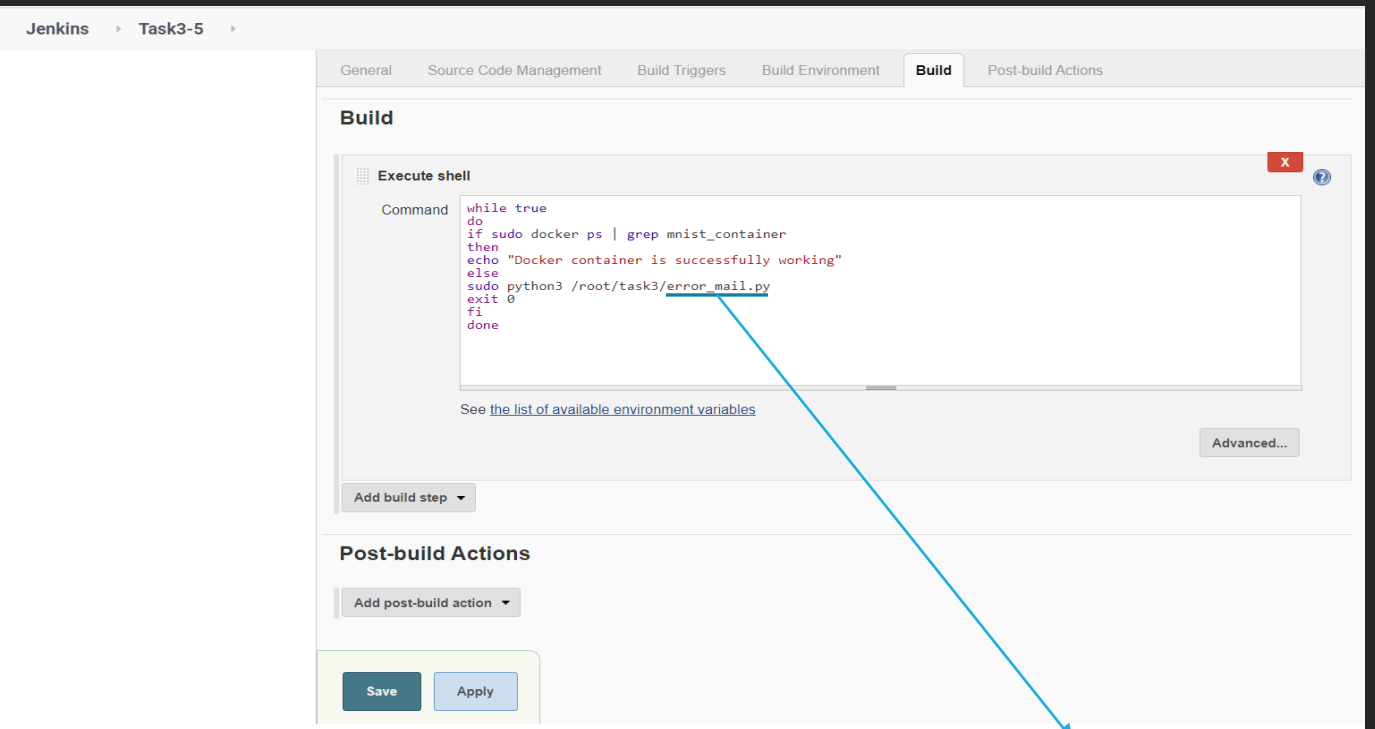
CREATE FOURTH JOB IN JENKINS THAT WILL BE TRIGGERED AFTER THE COMPLETION OF THE THIRD JOB.




```
with open("/root/accuracy.txt", "r") as f:
 acc=f.read()
import smtplib as sm
msg="The model has been successfully created with the desired accuracy!! \nAccuracy of the model is: "+str(acc)+" percentage"
server=sm.SMTP_SSL("smtp.gmail.com", 465)
server.login("modifearless@gmail.com","[redacted])
server.sendmail("modifearless@gmail.com","shhhhubhammmm@gmail.com", msg)
server.quit()
```

- As 95% accyracy is achieved, this job will automatically send a mail to your gmail account saying that desired accuracy is achieved!

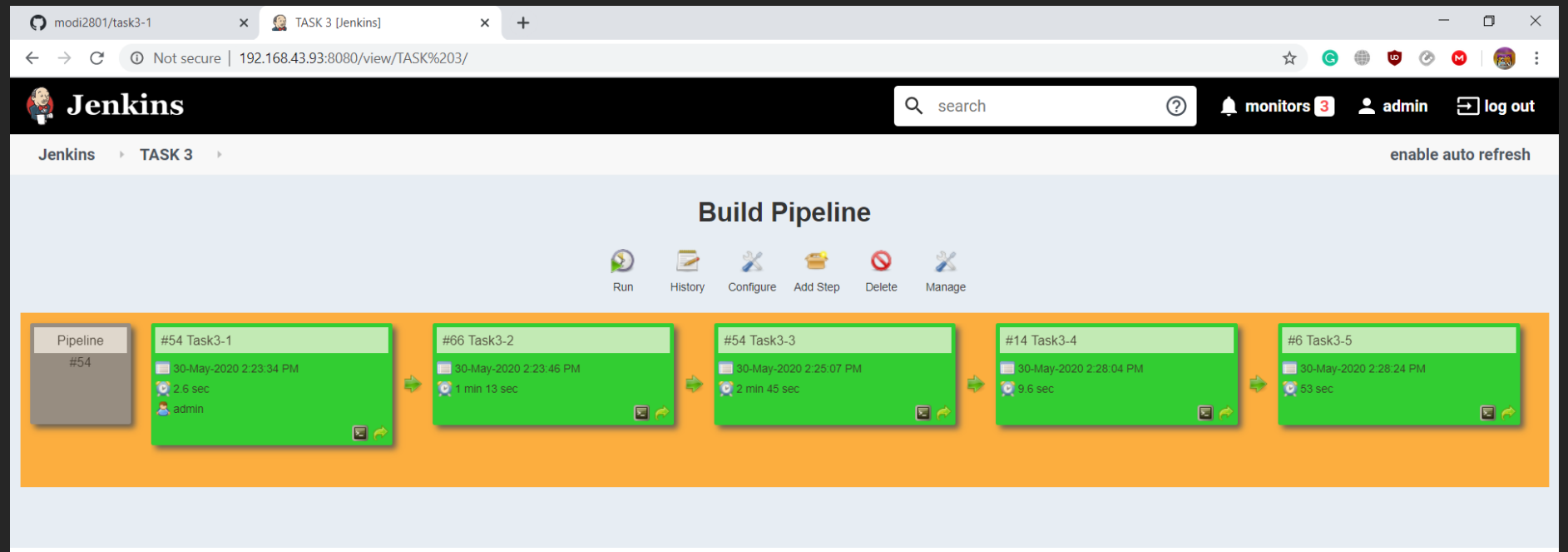
CREATE FIFTH JOB IN JENKINS THAT WILL BE TRIGGERED AFTER THE COMPLETION OF THE FOURTH JOB.




```
import smtplib as sm
msg="The model is not working properly!!!"
server=sm.SMTP_SSL("smtp.gmail.com", 465)
server.login("robertjr.2801@gmail.com","[redacted]")
server.sendmail("robertjr.2801@gmail.com","modifearless@gmail.com", msg)
server.close()
```

- This job will look over the container and as the container is failed, it will send an error mail to you gmail account.

AFTER CREATING ALL THE JOBS, IT WILL BE COMPLETELY AUTOMATED AND ALL THE PROCESS WILL RUN JUST ON A SINGLE CLICK !



Following is the Github link in which all the Console outputs and the codes are posted for further reference.

Github: https://github.com/modi2801/task3-1