A
Minor Project Report on

“Integrating Machine Learning With Docker And Jenkins On Redhat8 Linux Operating System”

In partial fulfillment of requirements for the degree of

Bachelor of Technology (B. Tech.)

in

Computer Science and Engineering

Submitted by
Ms. Princy Sultania (170319)

Under the Guidance of

Mr. P.K. Bishnoi

Computer Science and Engineering

SCHOOL OF ENGINEERING AND TECHNOLOGY

Mody University and Science and Technology
Lakshmangarh, Distt. Sikar-332311

December 2020

A C K N O W L E D G E M E N T

I sincerely express my gratitude to my guide **Mr. P.K. Bishnoi** for his benevolent guidance in completing the report on “**Integrating Machine Learning With Docker And Jenkins On Redhat8 Linux Operating System”**. His kindness and help have been the source of encouragement for me. I would also like to explicit my private appreciation to all those who supplied me the possibility to complete this report.

I am grateful to my guide for the guidance, inspiration and constructive suggestions that was helpful for me in the preparation of this project.

Princy Sultania (170319)

CERTIFICATE

This is to certify that the minor project report entitled "**Integrating Machine Learning With Docker And Jenkins On Redhat8 Linux Operating System**" submitted by Ms. Princy Sultania, as a partial fulfillment for the requirement of B. Tech. VII Semester examination of the School of Engineering and Technology, Mody University of Science and Technology, Lakshmangarh for the academic session 2019-2020 is an original project work carried out under the supervision and guidance of **Mr. P.K. Bishnoi** has undergone the requisite duration as prescribed by the institution for the project work.

PROJECT GUIDE:	**HEAD OF DEPARTMENT:**
Approval Code: AUT_20_CSE_F17_03	**Signature:**
Name: Mr. P.K. Bishnoi	**Name: Dr. A. Senthil**
Date: 27-12-2020	**Date: 27-12-2020**

EXAMINER-I:	**EXAMINER-II:**
Name: Dr. Sunil Kumar Jangir	**Name: Dr. Ajay Kumar Singh**
Dept.: Computer Science Engineering	**Dept.:Computer Science Engineering**

ABSTRACT

This is an automation pipeline project which aims at implementing an application of DevOps principles to Machine Learning and data science, commonly termed as MLOps.

DevOps combines software development (Dev) and IT operations (Ops), which aims at shortening the software development life-cycle and achieving automation by ensuring continuous integration and delivery of a software.

MLOps combines Machine Learning with DevOps, which applies the principles of DevOps for training the Machine Learning models.

This project aims at achieving an automation which will help train the Machine Learning models faster and with less human intervention, in order to get better results.

In data science field, the main challenge is to process the data, develop and improve the machine learning models with agility.

And achieving better accuracy is one of the main goal, as better accuracy indicates better predictions. But improving the accuracy of machine learning models is a tedious task as it depends on values of hyper-parameters (like no. of epochs), which require many hit and trials to get to a suitable value, which can result in better accuracy.

This hyper-parameter tuning can be done using an automation pipeline which can build the code, run it and analyze the output, and if the accuracy is less than desired, then the automation can tweak the model and rebuild it to achieve better outcomes.

Jenkins is an extremely powerful automation tool that integrates various tools easily. Docker is a tool of containerization technology that builds an environment for the software to run smoothly.

Hence, to implement MLOps, this project integrates GIT, GitHub, Jenkins and Docker on Linux RedHat 8 Operating System.

List of Figures

Table of Contents

Chapter 1: INTRODUCTION

1.1 PRESENT SYSTEM

In Software Industry, to create quality software and to effectively deliver it to the clients is a long and exhausting procedure. As this involves a complete lifecycle of multiple systems to make an effective software and to enhance its ease of use and availability to its clients in the most effective manner possible. The lifecycle involves planning, designing of the software, then implementing it on some platform, then come the rigorous testing part of the complete software and once it is acceptable according to the standards required by the client, the deployment of software starts and then the maintenance and debugging along with updates of software happens time to time. Traditional software management strategies fail to cater with this kind of demand.

Also, there are multiple teams in a project carrying out different roles, and they need to come along for better integration of the software. So, to eliminate most of the repetitive and manual tasks, these days, agile methodologies are used for this complete manufacturing of a software. And one of the ways to implement this is through DevOps.

DevOps combines Software development (Dev) and IT Operations (Ops), which aims at shortening the software development life-cycle and achieving automation by ensuring continuous integration and delivery of a software.

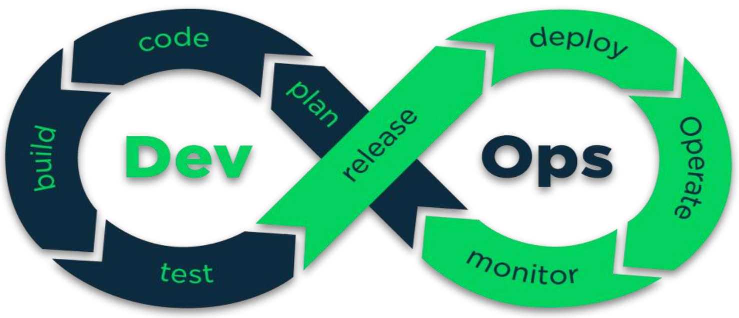


Figure 1: DevOps

DevOps helps to speed up the process using which an idea such as a new software feature goes from development process to deployment in a production environment where it can fulfil the needs of the customer or the client.

Scalability along with flexible provisioning also play a major role. With DevOps, a complete automation pipeline is created so as to fulfil the rising needs and demands in the software market.

Software developers work on the coding part and the IT team provides the software with an environment to execute in the client's system, they make sure to speed up the software build and testing process. Using DevOps, enhances the speed and reliability of the system as it eliminates manual errors.

In data science field, the main challenge is to process the data, develop and improve the machine learning models with agility. Here, achieving better accuracy is one of the main goal, as better accuracy indicates better predictions. But improving the accuracy of machine learning models is a tedious task as it depends on values of hyper-parameters (like no. of epochs), which require many hit and trials to get to a suitable value, which can result in better accuracy.

1.2 PROPOSED SYSTEM

Using DevOps with Machine Learning is the aim of my project.

MLOps combines Machine Learning with DevOps, which applies the principles of DevOps for training the Machine Learning models. This project aims at achieving an automation which will help train the Machine Learning models faster and with less human intervention, in order to get better results

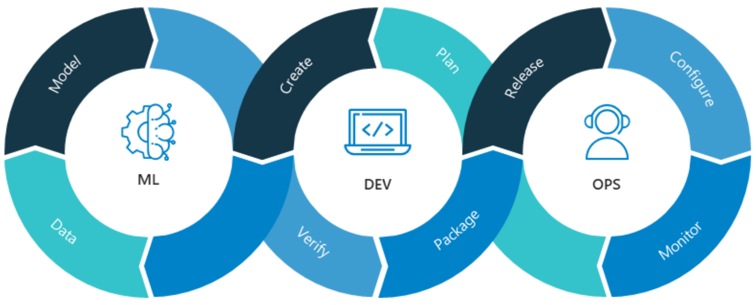


Figure 2: MLOps

This project implements an application of DevOps principles to Machine Learning and data science, commonly termed as MLOps.

The hyper-parameter tuning can be done using an automation pipeline which can build the code, run it and analyse the output, and if the accuracy is less than desired, then the automation can tweak the model and rebuild it to achieve better outcomes.

Jenkins is an extremely powerful automation tool that integrates various tools easily. Docker is a tool of containerization technology that builds an environment for the software to run smoothly. Hence, to implement MLOps, this project integrates Git, Github, Jenkins and Docker on Linux RedHat 8 Operating System.

Chapter 2: SYSTEM DESIGN

2.1 SYSTEM FLOWCHART

As this is an automation pipeline, so all the tools and technologies used are integrated to one another in order to work as a pipeline.

This flowchart gives an idea of the overall flow of different tasks in order to achieve the automation:

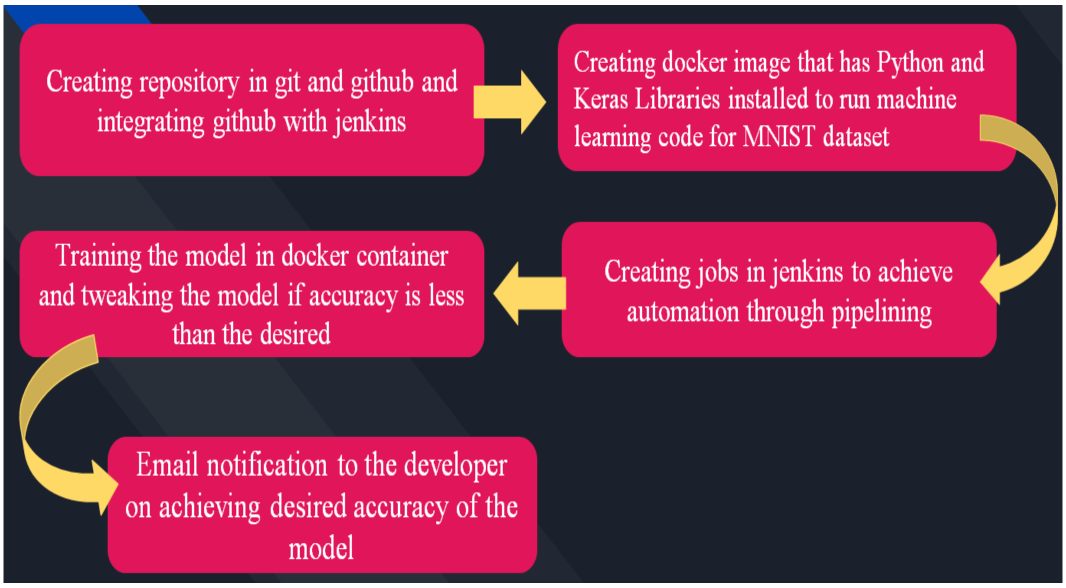

Creating repository in git and github and integrating github with jenkins
Creating docker image that has Python and Keras Libraries installed to run machine learning code for MNIST dataset
Creating jobs in jenkins to achieve automation through pipelining
Training the model in docker container and tweaking the model if accuracy is less than the desired
Email notification to the developer on achieving desired accuracy of the model

Figure 3: Flowchart

So, to understand the project workflow properly, I have designed a detailed pictorial representation about the working of the project.

The figure below shows the entire workflow and the project's design pictorially:

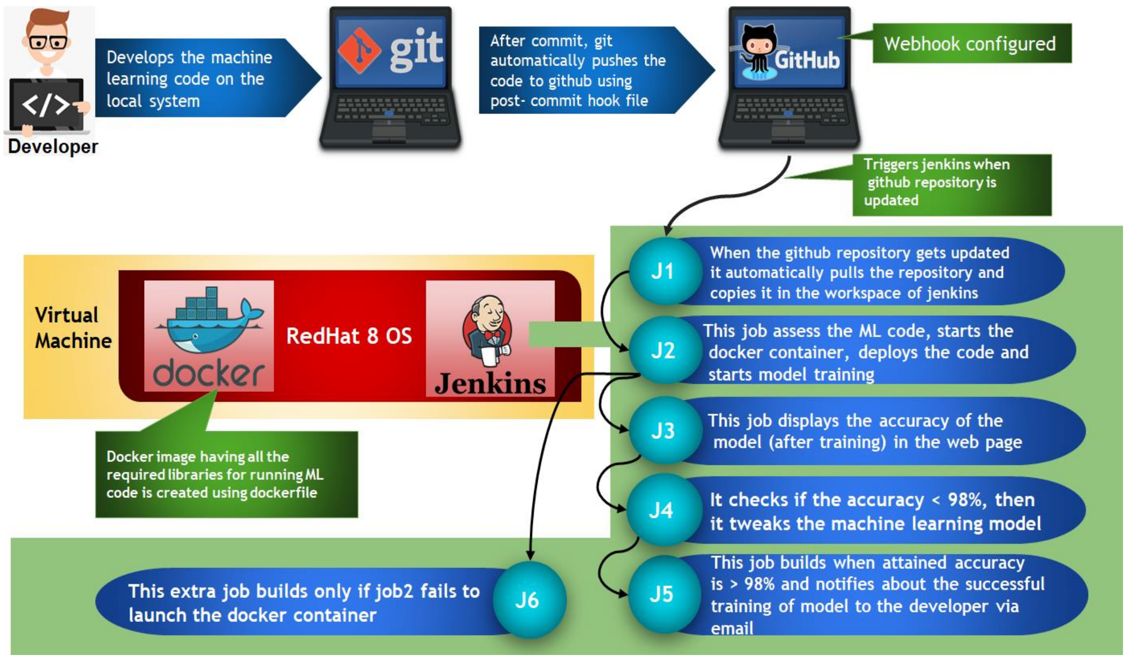

Developer
Develops the machine learning code on the local system
git
After commit, git automatically pushes the code to github using post- commit hook file
GitHub
Webhook configured
Triggers jenkins when github repository is updated
J1
When the github repository gets updated it automatically pulls the repository and copies it in the workspace of jenkins
Virtual Machine
docker
RedHat 8 OS
Jenkins
J2
This job assess the ML code, starts the docker container, deploys the code and starts model training
J3
This job displays the accuracy of the model (after training) in the web page
Docker image having all the required libraries for running ML code is created using dockerfile
J4
It checks if the accuracy < 98%, then it tweaks the machine learning model
This extra job builds only if job2 fails to launch the docker container
J6
J5
This job builds when attained accuracy is > 98% and notifies about the successful training of model to the developer via email

Figure 4: Project Design

2.2 PROJECT DESCRIPTION

The following steps define the workflow of this automation pipeline project:

- ❑ A docker image is created that has environment to run Python3, and other machine learning libraries required (like keras) for running machine learning programs. This docker image is created using Dockerfile.
- ❑ On launching this docker image, the training of the machine learning model starts.
- ❑ A chain of six jobs: jb1, jb2, jb3, jb4, jb5 and jb6 is created in Jenkins and can be viewed using build pipeline plugin in Jenkins.
- ❑ The first job jb1: When program files are pushed to GitHub, it pulls the github repository and copies it in its own workspace.
- ❑ The second job jb2: Jenkins analyses the code to check if it a machine learning code and the launch the docker container to deploy code and run the code.
- ❑ The third job jb3: This job predicts the accuracy or metrics of the trained model.
- ❑ The fourth job jb4: If the accuracy turns out to be lesser than the desired value, then Jenkins runs the code to tweak the machine learning model's architecture.
- ❑ The fifth job jb5: This checks if the desired accuracy is achieved, then it notifies the developer by sending an email.
- ❑ The sixth job jb6: This only executes if the container fails in between training.

Chapter 3: HARDWARE AND SOFTWARE DETAILS

This project uses the following software tools:

- Git
- Github
- Docker
- Jenkins
- RedHat 8 Linux Operating System

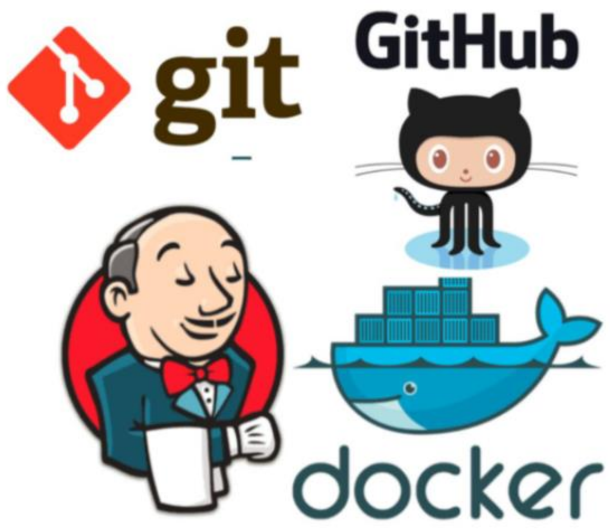


Figure 5: Tools used

3.1 GIT

Git is a distributed version control system, which means that the local copy of code is a complete version control repository. The local repositories make it is easy to work offline or remotely. This helps to commit the developer's work locally, and then sync the copy of the repository with the copy on the server.

Real life projects have multiple developers working together in parallel. So, a version control system like Git is required to ensure there are no code conflicts between the developers.

Also, the requirements in real life projects changes very often. So, a version control system allows developers to revert and go back to an older version of the code and also refer the changes done in the code in case of occurrence of errors.

Some commands in Git which are used in this project are as follows:

1. git init
2. git add
3. git commit
4. git push

3.2 GITHUB

GitHub is a web-based version-control and collaboration platform used by software developers. It is a git repository hosting service, used to put the source code on the internet.

GitHub is used to store the source code for a project and track the complete history of all changes that happened in the program or the code. It helps software developers to collaborate on a project more efficiently by managing conflicting changes done by multiple developers.

Each public or private repository contains all the files of a project, along with the revision history of each file. The github repositories may have many collaborators. These repositories may be public or private.

3.3 JENKINS

Jenkins is an open source automation server which enables developers to build, test and deploy their software.

Jenkins manages the software delivery processes throughout the entire lifecycle, that includes building, documentation, testing, packaging, staging, deployment. Using Jenkins, Software development industries can speed up the software development process by automating it.

In Jenkins, to trigger build of a job, we can use any one of the following methods:

- ❏ Requesting a specific build URL
- ❏ A Webhook that gets triggered upon pushed commits in a version control system
- ❏ After the other builds in the queue have completed
- ❏ Invoked by other builds
- ❏ Scheduling via a cron tab

In this project, the Docker and Jenkins services are hosted on RedHat8 Linux Operating System which is itself installed on Virtual Machine (VM).

3.4 DOCKER

Docker is a tool which uses container technology to give an environment for running programs or applications. Containers allow a developer to package up an application with required libraries and all the other dependencies and deploy it as one complete package. Docker services can be run on Linux, Windows and MacOS as well.

But virtual machines are very heavy as each VM requires its own Operating System, hence it becomes difficult to maintain it.

Containers helps isolate applications execution environments from each other, but the OS kernel remains sharable. They're measured in megabytes, use fewer resources than VMs, and start running immediately.

Since docker containers are very light in weight, so one virtual machine can run multiple containers at the same time.

Here, in this project, docker services are running on the top of RedHat 8 Linux Operating system.

I have created docker image using dockerfile, then build it, to create a docker image.

Some of the docker commands:

1. Docker start- to start the docker services
2. Docker stop- to stop the services
3. Docker ps- to see all the currently running containers
4. Docker build- to build the dockerfile
5. Docker run-to launch the docker container

3.5 REDHAT 8 LINUX OPERATING SYSTEM

The services like jenkins and docker are configured to run on the Linux system here, and the version used is RedHat 8 Linux Operating System.

Chapter 4: IMPLEMENTATION WORK DETAILS

4.1 REAL LIFE APPLICATIONS

The DevOps principles can be applied to various real life problems in the software industry:

- Various docker images can be made which support different languages, and since every other programming language or code requires a different environment and libraries, so a different docker image for each language. And then, the jenkins can be configured to analyze the code pushed by the developer and accordingly launch the docker container which supports that language.
- DevOps can be applied to train Machine learning models as they require hyper-tuning of parameters in order to achieve the desired accuracy. And this is what my project implements.
- Other automation pipelines according to requirement could also be created in order to use this methodology to solve a particular problem or to eliminate too much of manual interference as it leads to errors.

4.2 DATA IMPLEMENTATION AND PROGRAM EXECUTION

As this project aims at training a machine learning model and predicting as well as improving the accuracy, so it needs a dataset. And the dataset used is MNIST Handwritten dataset.

MNIST dataset:

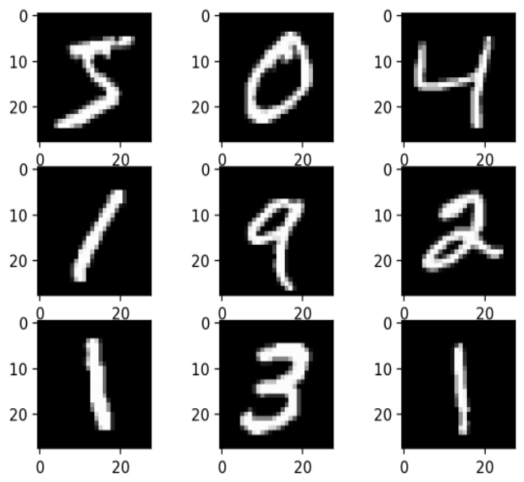


Figure 6: MNIST Handwritten dataset

- In this project, MNIST Handwritten digit classification dataset is used.
- MNIST stands for Modified National Institute of Standards and Technology dataset.
- It is a dataset of 60,000 small square 28×28 pixel grayscale images of handwritten single digits between 0 and 9.

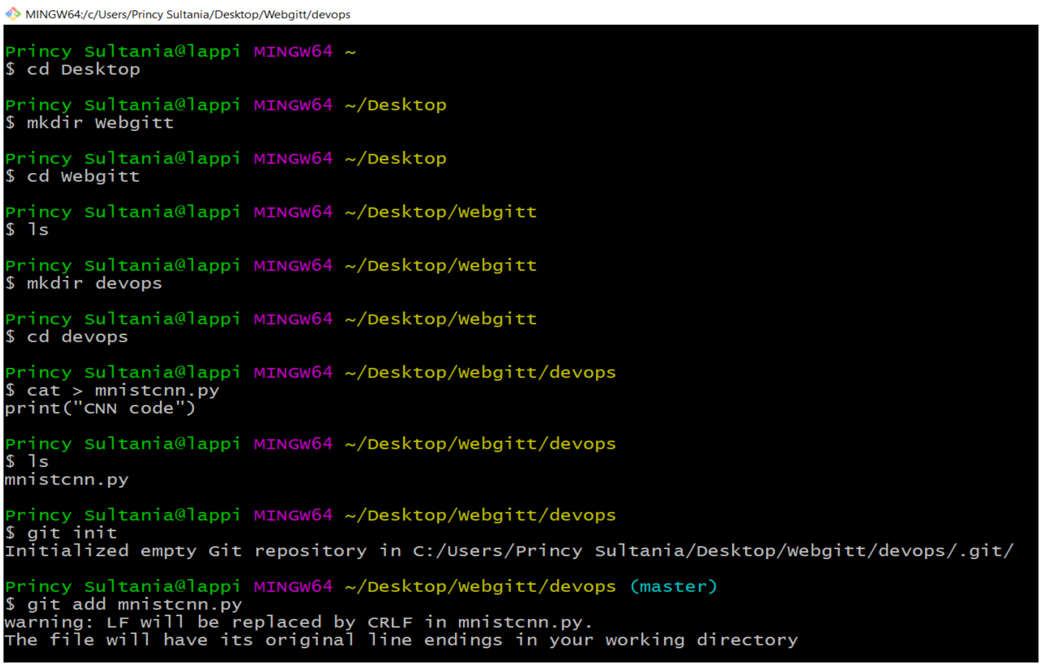
CNN Architecture:

- As this is a classification problem, so Classification technique of supervised learning is used.
- In deep learning, a **Convolutional Neural Network** is a class of deep neural networks most commonly applied to analyzing visual images.
- The name "convolutional neural network" indicates that the network uses a mathematical operation called convolution instead of general matrix multiplication used in other algorithms. Hence, it is considered effective for image analysis applications.
- **CNN architecture** is inspired by the organization and functionality of the visual cortex and designed to mimic the connectivity pattern of neurons within the human brain. The neurons within a CNN are split into a three-dimensional structure, with each set of neurons analyzing a small region or feature of the image.
- The task is to classify a given image of a handwritten digit into one of 10 classes representing integer values from 0 to 9.
- CNN uses multilayer perceptrons to do computational works. CNNs use relatively little pre-processing compared to other image classification algorithms. This means the network learns through filters that in traditional algorithms were hand-engineered. So, for image processing task CNNs are the best-suited option.

Procedure:

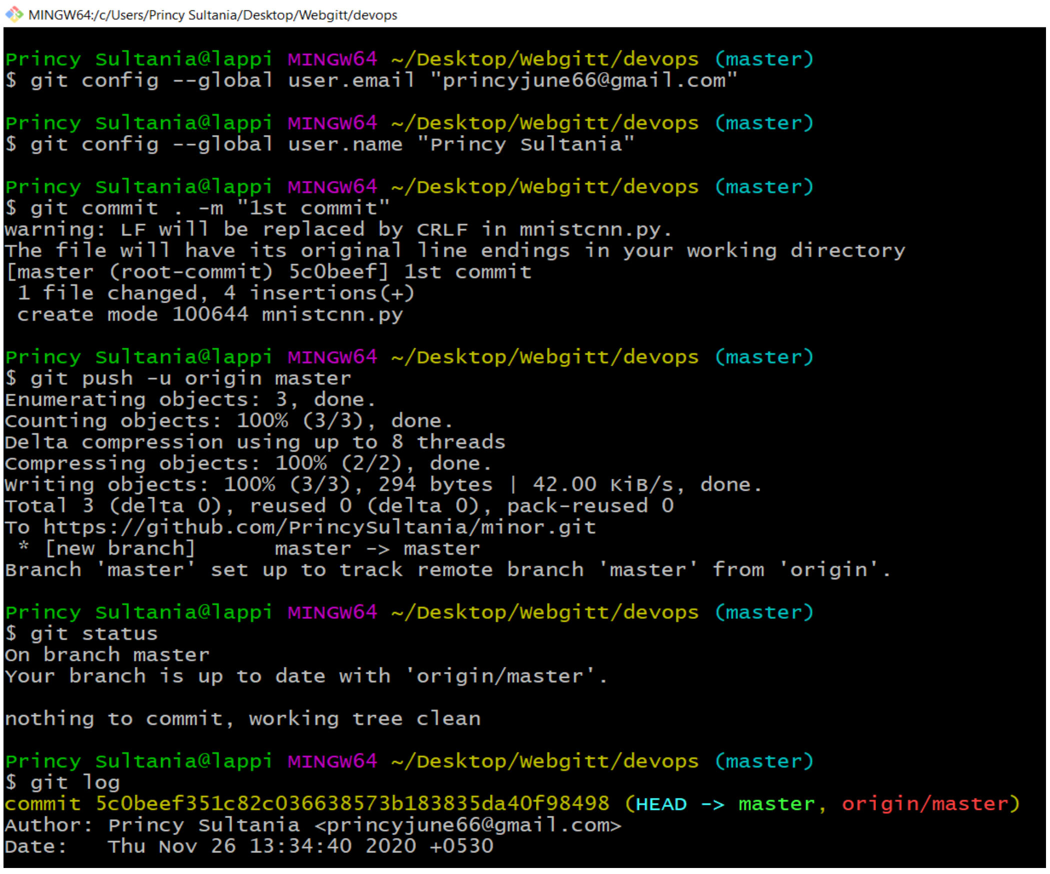
1. **Creating repository and post-commit hook file in git**:

   Git and GitHub repositories are created and linked. An empty github repository named "minor" is created.
   In git terminal, do the following to upload files on github:



```
MINGW64:/c/Users/Princy Sultania/Desktop/Webgitt/devops

Princy Sultania@lappi MINGW64 ~
$ cd Desktop

Princy Sultania@lappi MINGW64 ~/Desktop
$ mkdir Webgitt

Princy Sultania@lappi MINGW64 ~/Desktop
$ cd Webgitt

Princy Sultania@lappi MINGW64 ~/Desktop/Webgitt
$ ls

Princy Sultania@lappi MINGW64 ~/Desktop/Webgitt
$ mkdir devops

Princy Sultania@lappi MINGW64 ~/Desktop/Webgitt
$ cd devops

Princy Sultania@lappi MINGW64 ~/Desktop/Webgitt/devops
$ cat > mnistcnn.py
print("CNN code")

Princy Sultania@lappi MINGW64 ~/Desktop/Webgitt/devops
$ ls
mnistcnn.py

Princy Sultania@lappi MINGW64 ~/Desktop/Webgitt/devops
$ git init
Initialized empty Git repository in C:/Users/Princy Sultania/Desktop/Webgitt/devops/.git/

Princy Sultania@lappi MINGW64 ~/Desktop/Webgitt/devops (master)
$ git add mnistcnn.py
warning: LF will be replaced by CRLF in mnistcnn.py.
The file will have its original line endings in your working directory
```

Figure 7: Configuring git

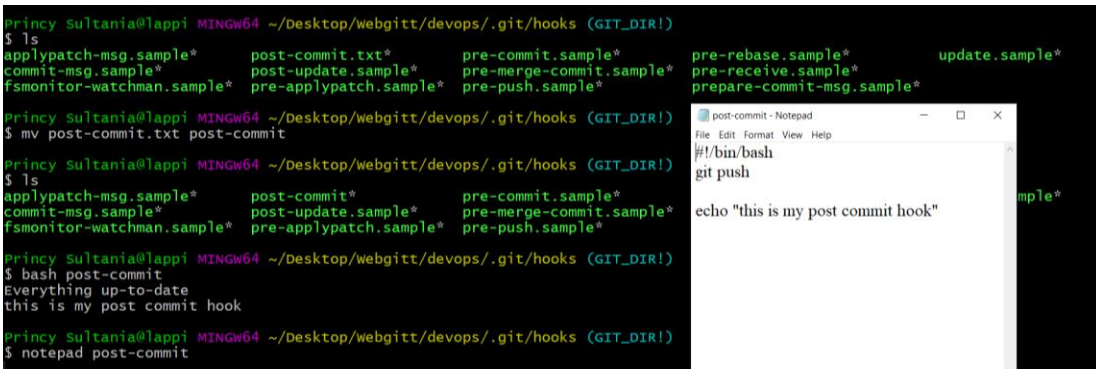


```
MINGW64:/c/Users/Princy Sultania/Desktop/Webgitt/devops

Princy Sultania@lappi MINGW64 ~/Desktop/Webgitt/devops (master)
$ git config --global user.email "princyjune66@gmail.com"

Princy Sultania@lappi MINGW64 ~/Desktop/Webgitt/devops (master)
$ git config --global user.name "Princy Sultania"

Princy Sultania@lappi MINGW64 ~/Desktop/Webgitt/devops (master)
$ git commit . -m "1st commit"
warning: LF will be replaced by CRLF in mnistcnn.py.
The file will have its original line endings in your working directory
[master (root-commit) 5c0beef] 1st commit
 1 file changed, 4 insertions(+)
 create mode 100644 mnistcnn.py

Princy Sultania@lappi MINGW64 ~/Desktop/Webgitt/devops (master)
$ git push -u origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 294 bytes | 42.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/PrincySultania/minor.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.

Princy Sultania@lappi MINGW64 ~/Desktop/Webgitt/devops (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean

Princy Sultania@lappi MINGW64 ~/Desktop/Webgitt/devops (master)
$ git log
commit 5c0beef351c82c036638573b183835da40f98498 (HEAD -> master, origin/master)
Author: Princy Sultania <princyjune66@gmail.com>
Date:   Thu Nov 26 13:34:40 2020 +0530
```

Figure 8: Pushing code to github

Also, a post-commit hook file is created to automatically push the source code from git to github, once the developer commits the code in git. This provides a certain level of automation.

In git terminal, the following commands are run to create a post-commit hook file:

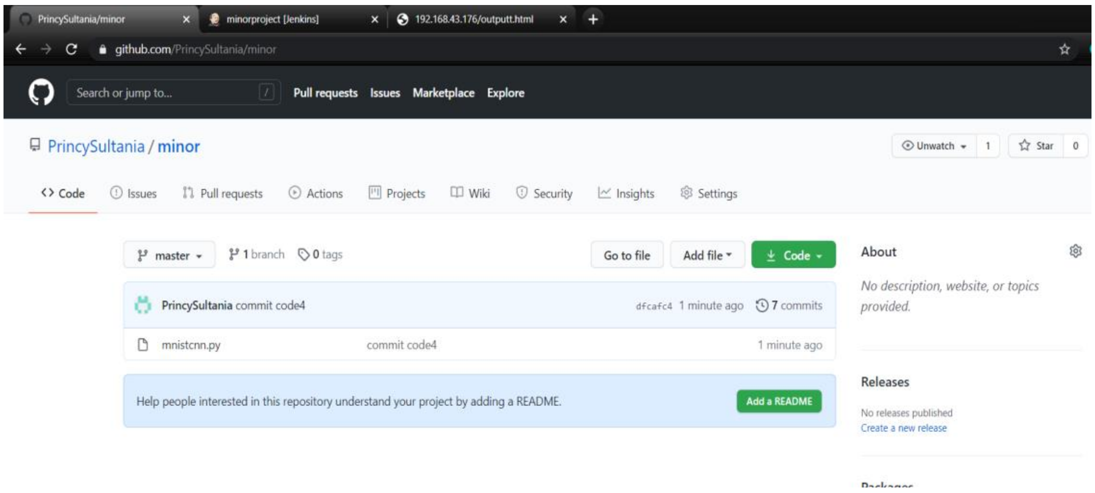


```
Princy Sultania@lappi MINGW64 ~/Desktop/webgitt/devops/.git/hooks (GIT_DIR!)
$ ls
applypatch-msg.sample*      post-commit.txt*        pre-commit.sample*        pre-rebase.sample*           update.sample*
commit-msg.sample*          post-update.sample*     pre-merge-commit.sample*  pre-receive.sample*
fsmonitor-watchman.sample*  pre-applypatch.sample*  pre-push.sample*          prepare-commit-msg.sample*

Princy Sultania@lappi MINGW64 ~/Desktop/webgitt/devops/.git/hooks (GIT_DIR!)
$ mv post-commit.txt post-commit

Princy Sultania@lappi MINGW64 ~/Desktop/webgitt/devops/.git/hooks (GIT_DIR!)
$ ls
applypatch-msg.sample*      post-commit*            pre-commit.sample*
commit-msg.sample*          post-update.sample*     pre-merge-commit.sample*
fsmonitor-watchman.sample*  pre-applypatch.sample*  pre-push.sample*

Princy Sultania@lappi MINGW64 ~/Desktop/webgitt/devops/.git/hooks (GIT_DIR!)
$ bash post-commit
Everything up-to-date
this is my post commit hook

Princy Sultania@lappi MINGW64 ~/Desktop/webgitt/devops/.git/hooks (GIT_DIR!)
$ notepad post-commit
```

```
#!/bin/bash
git push

echo "this is my post commit hook"
```

Figure 9: Post-commit hook file

The repository of github gets updated:



PrincySultania commit code4 dfcafc4 1 minute ago 7 commits
mnistcnn.py commit code4 1 minute ago

Help people interested in this repository understand your project by adding a README. Add a README

No description, website, or topics provided.

Releases
No releases published
Create a new release

Figure 10: Repository updated in github

2. Configuring Webhook in GitHub:

As GitHub requires the public IP (Internet Protocol) of the system where Jenkins automation server is running, so that when a program or code file is pushed to

GitHub in this repository, Jenkins goes to GitHub repository to get the source code in order to start the job1.

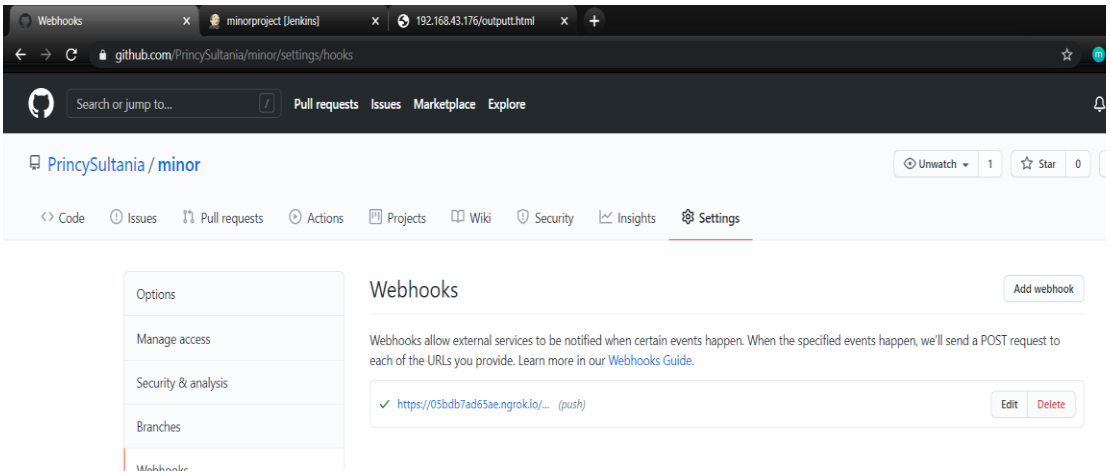
This can be done by adding a Webhook to the GitHub's repository.

The **ngrok software** is used to convert the private Internet Protocol (IP) address into public IP address.

In order to get public Internet Protocol address, run the below command on RedHat8 terminal:

./ngrok http 8080

The address so obtained is mentioned in the *payload URL* field of Webhooks of the GitHub repository.



Webhooks

Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our Webhooks Guide.

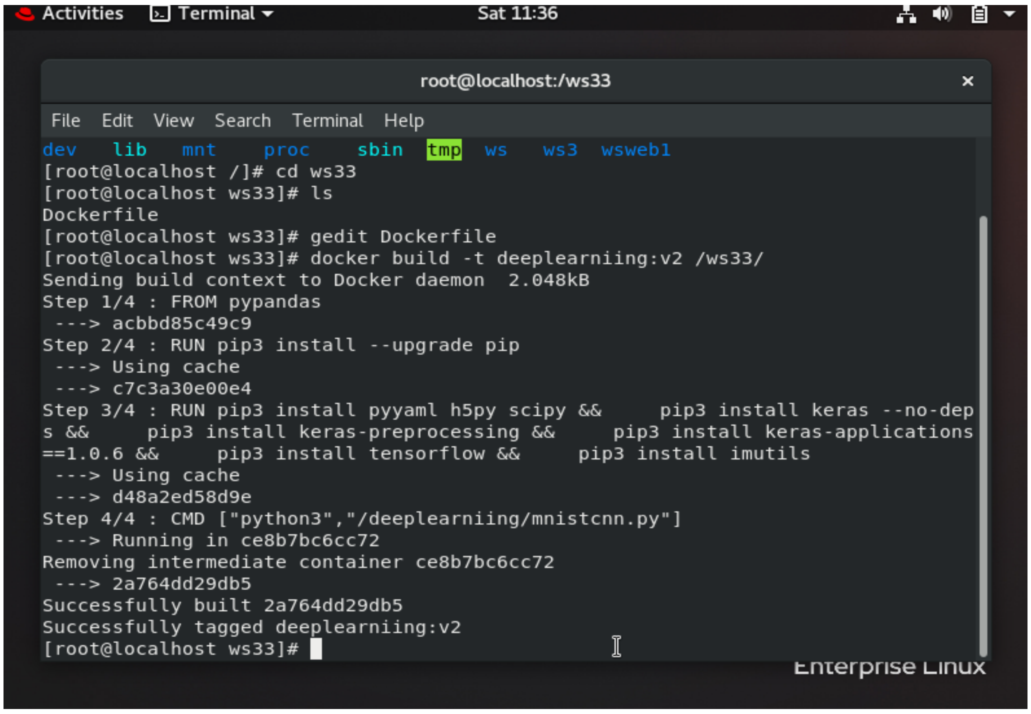
✓ https://05bdb7ad65ae.ngrok.io/... (push)

Figure 11: Configuring Webhook

3. Creating docker images using Dockerfile:

Then, save this Dockerfile and build it using the command:

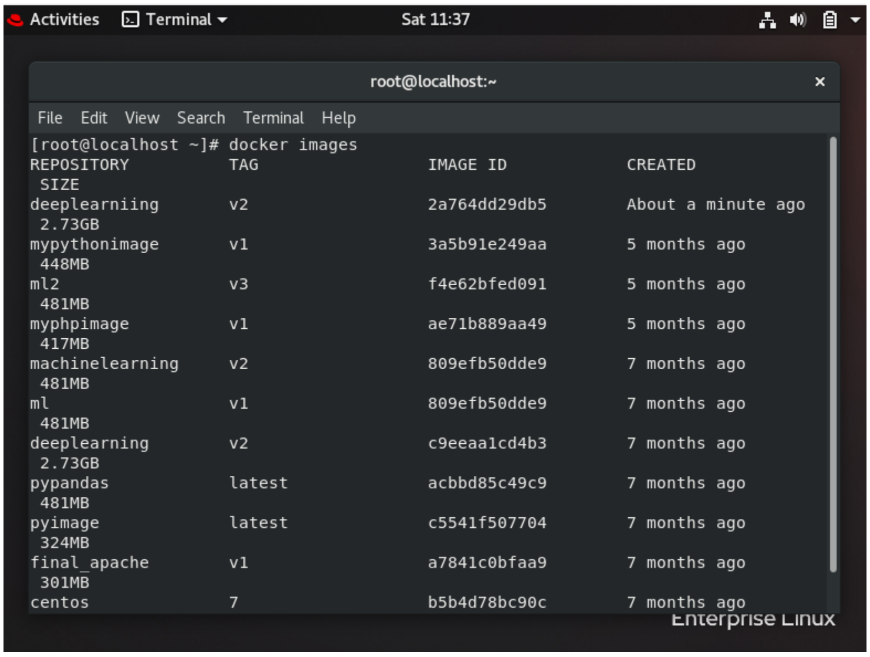
docker build -t deeplearniing:v2 /ws33/



```
dev   lib   mnt    proc    sbin  tmp  ws   ws3  wsweb1
[root@localhost /]# cd ws33
[root@localhost ws33]# ls
Dockerfile
[root@localhost ws33]# gedit Dockerfile
[root@localhost ws33]# docker build -t deeplearniing:v2 /ws33/
Sending build context to Docker daemon  2.048kB
Step 1/4 : FROM pypandas
 ---> acbbd85c49c9
Step 2/4 : RUN pip3 install --upgrade pip
 ---> Using cache
 ---> c7c3a30e00e4
Step 3/4 : RUN pip3 install pyyaml h5py scipy &&     pip3 install keras --no-dep
s &&     pip3 install keras-preprocessing &&     pip3 install keras-applications
==1.0.6 &&     pip3 install tensorflow &&     pip3 install imutils
 ---> Using cache
 ---> d48a2ed58d9e
Step 4/4 : CMD ["python3","/deeplearniing/mnistcnn.py"]
 ---> Running in ce8b7bc6cc72
Removing intermediate container ce8b7bc6cc72
 ---> 2a764dd29db5
Successfully built 2a764dd29db5
Successfully tagged deeplearniing:v2
[root@localhost ws33]#
```

Figure 12: Building the Dockerfile

The docker images can be seen using *docker images* command:

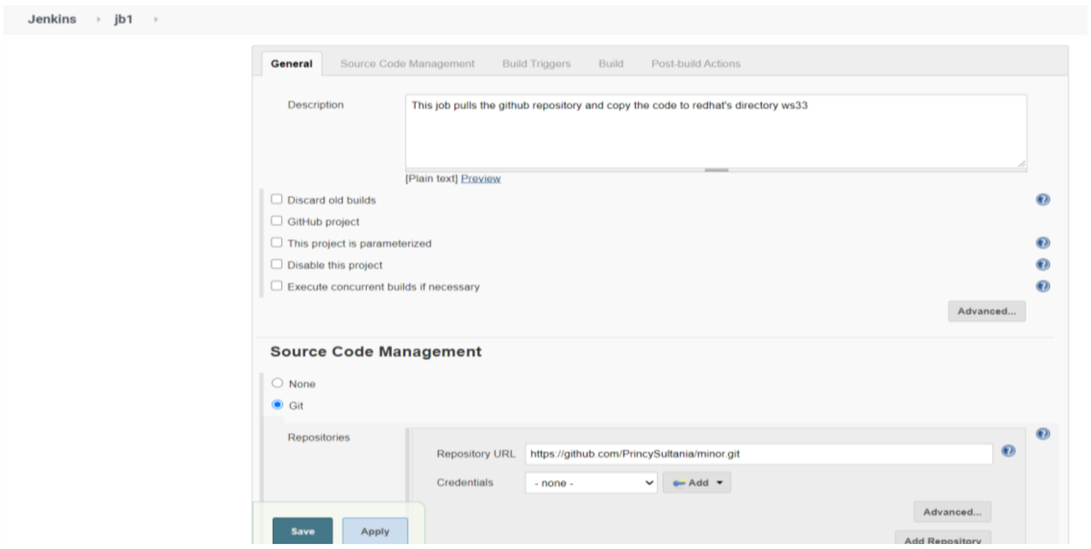


```
[root@localhost ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED
 SIZE
deeplearniing       v2                  2a764dd29db5        About a minute ago
 2.73GB
mypythonimage       v1                  3a5b91e249aa        5 months ago
 448MB
ml2                 v3                  f4e62bfed091        5 months ago
 481MB
myphpimage          v1                  ae71b889aa49        5 months ago
 417MB
machinelearning     v2                  809efb50dde9        7 months ago
 481MB
ml                  v1                  809efb50dde9        7 months ago
 481MB
deeplearning        v2                  c9eeaa1cd4b3        7 months ago
 2.73GB
pypandas            latest              acbbd85c49c9        7 months ago
 481MB
pyimage             latest              c5541f507704        7 months ago
 324MB
final_apache        v1                  a7841c0bfaa9        7 months ago
 301MB
centos              7                   b5b4d78bc90c        7 months ago
```

Figure 13: Docker image created

4. Creating first job jb1 in Jenkins:

When Github informs Jenkins that some code or program files are updated in its repository (using Webhook), this job goes to the GitHub repository and copies the source code/ program into the workspace of Jenkins.


Description: This job pulls the github repository and copy the code to redhat's directory ws33

- [ ] Discard old builds
- [ ] GitHub project
- [ ] This project is parameterized
- [ ] Disable this project
- [ ] Execute concurrent builds if necessary

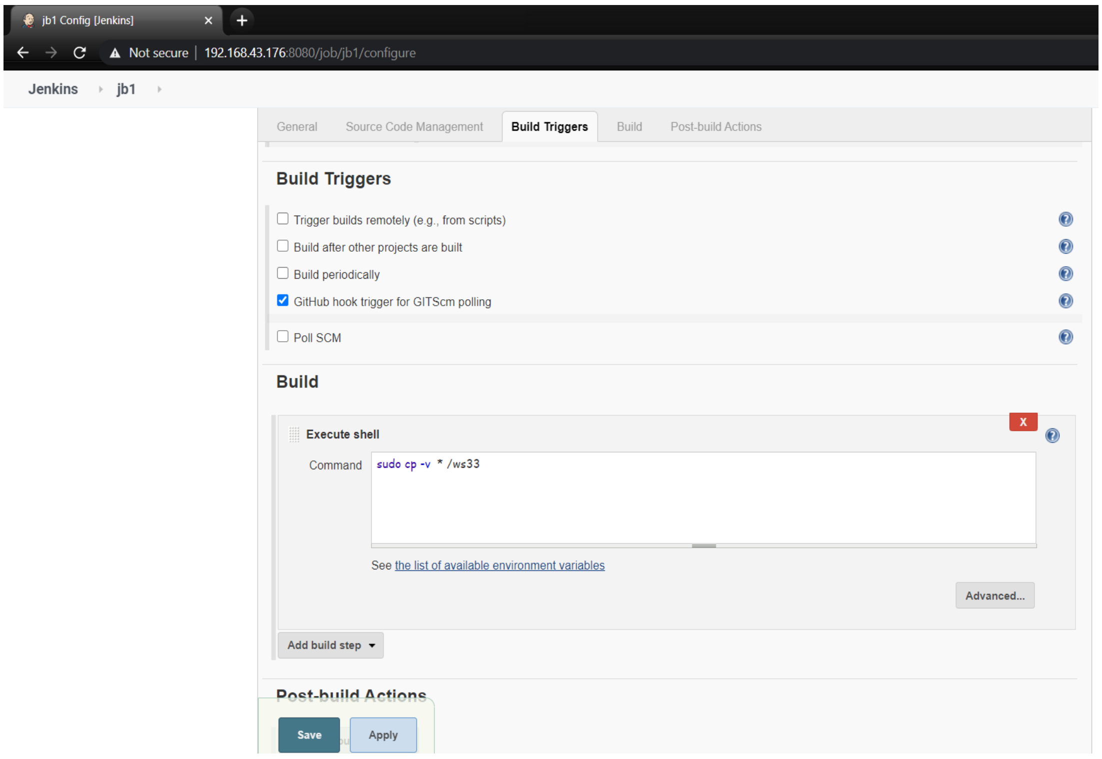
Source Code Management

None [✓ Git]

Repository URL: https://github.com/PrincySultania/minor.git

Credentials: - none -

Figure 14: SCM Configuration of job jb1


192.168.43.176:8080/job/jb1/configure

Build Triggers

- [ ] Trigger builds remotely (e.g., from scripts)
- [ ] Build after other projects are built
- [ ] Build periodically
- [x] GitHub hook trigger for GITScm polling
- [ ] Poll SCM

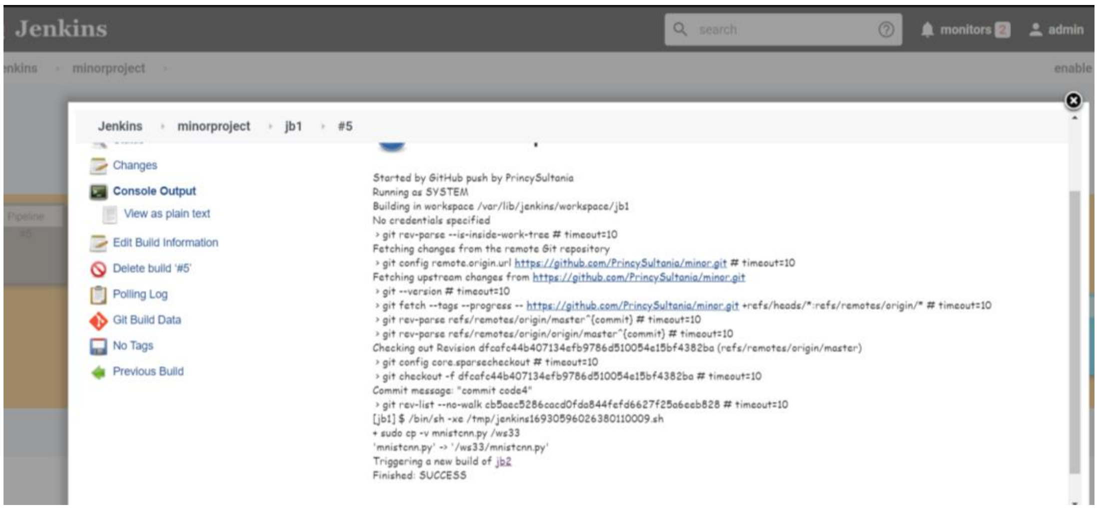
Build

Execute shell

Command:

```
sudo cp -v * /ws33
```

Figure 15: Build Configuration of job jb1



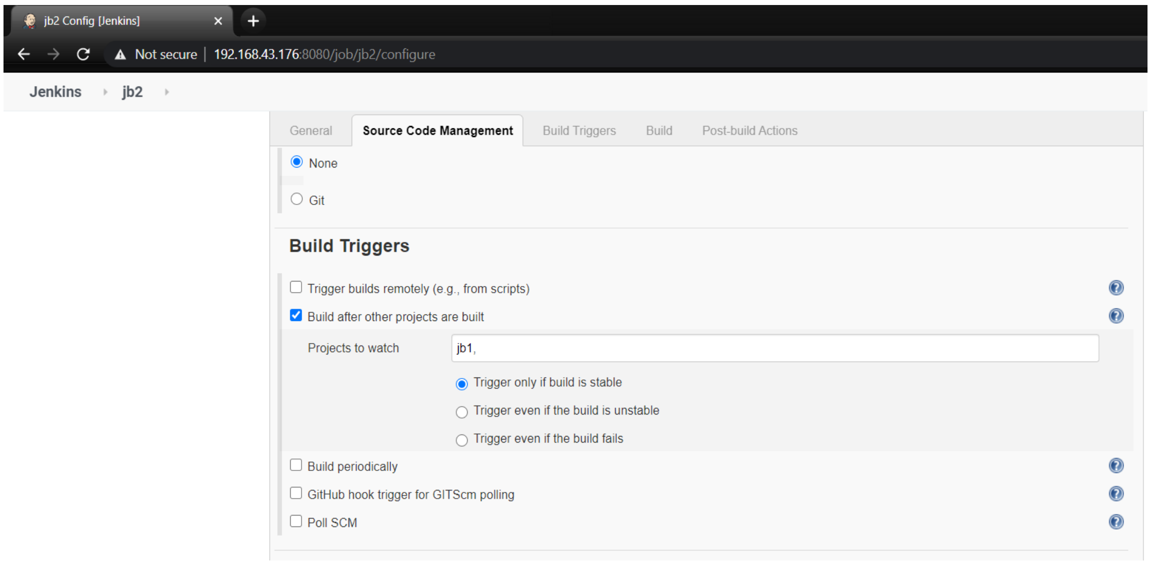
```
Started by GitHub push by PrincySultania
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/jb1
No credentials specified
 > git rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
 > git config remote.origin.url https://github.com/PrincySultania/minor.git # timeout=10
Fetching upstream changes from https://github.com/PrincySultania/minor.git
 > git --version # timeout=10
 > git fetch --tags --progress -- https://github.com/PrincySultania/minor.git +refs/heads/*:refs/remotes/origin/* # timeout=10
 > git rev-parse refs/remotes/origin/master^{commit} # timeout=10
 > git rev-parse refs/remotes/origin/origin/master^{commit} # timeout=10
Checking out Revision dfcafc44b407134efb9786d510054e15bf4382ba (refs/remotes/origin/master)
 > git config core.sparsecheckout # timeout=10
 > git checkout -f dfcafc44b407134efb9786d510054e15bf4382ba # timeout=10
Commit message: "commit code4"
 > git rev-list --no-walk cb5aec5286cacd0fda844fefd6627f25a6eeb828 # timeout=10
[jb1] $ /bin/sh -xe /tmp/jenkins1693059602638011009.sh
+ sudo cp -v mnistcnn.py /ws33
'mnistcnn.py' -> '/ws33/mnistcnn.py'
Triggering a new build of jb2
Finished: SUCCESS
```

Figure 16: Console output of job jb1

5. Creating second job jb2 in Jenkins:

This job analyse the program file to check whether it is a machine learning code or not. Here, as the code pushed to GitHub is a Machine Learning code, so Jenkins launches the docker container made for running this program.

To analyse the code and check it, **grep command** is used to search for a particular keyword, so that the respective docker container can be launched. Like here, keras library is used in the machine learning code, so it is a good way to check for the presence of the word **keras** in the program.

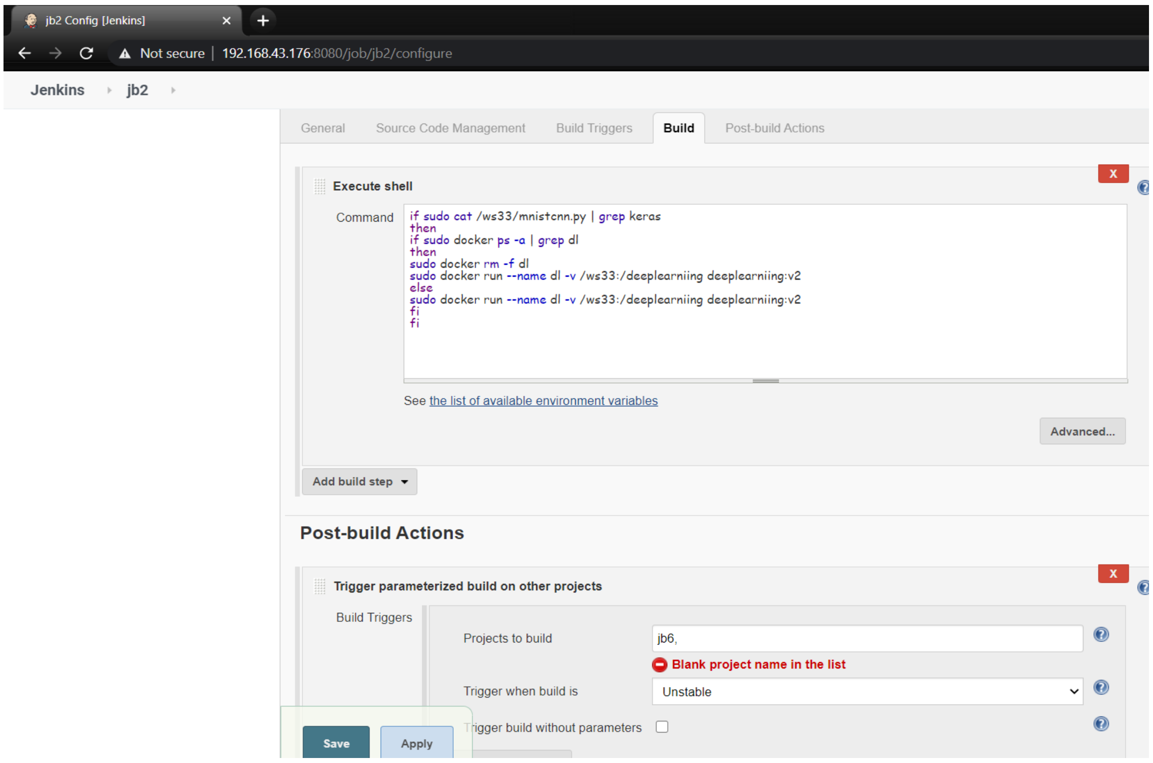


Source Code Management: [✓ None] Git

Build Triggers

- ☐ Trigger builds remotely (e.g., from scripts)
- ☑ Build after other projects are built
  - Projects to watch: jb1,
  - [✓ Trigger only if build is stable] Trigger even if the build is unstable / Trigger even if the build fails
- ☐ Build periodically
- ☐ GitHub hook trigger for GITScm polling
- ☐ Poll SCM

Figure 17: Configuration of job jb2



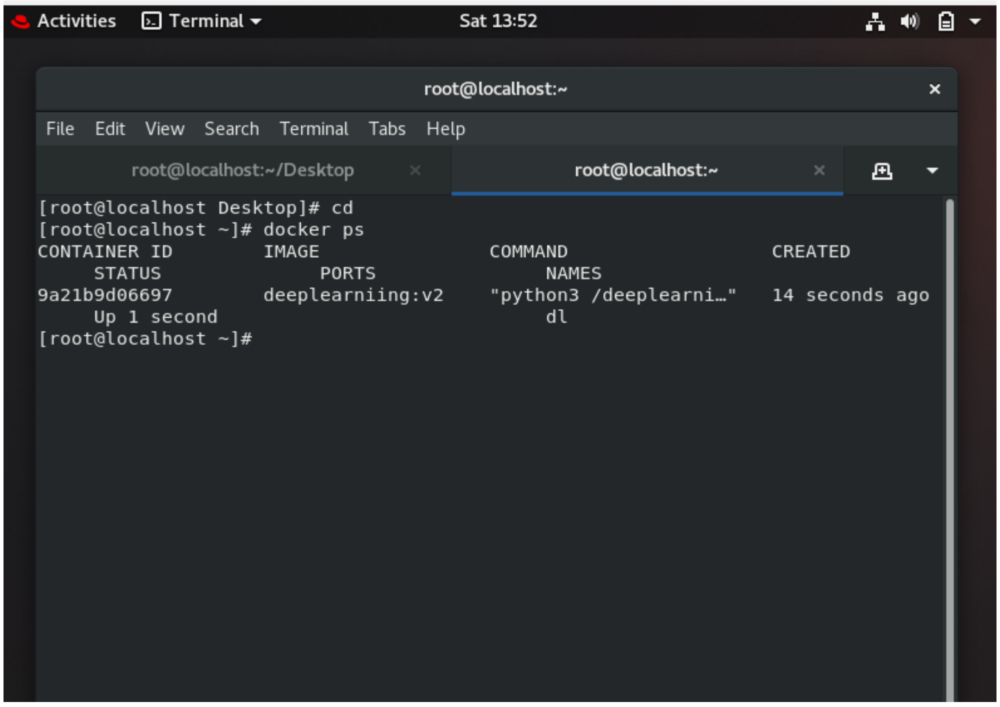
```
if sudo cat /ws33/mnistcnn.py | grep keras
then
if sudo docker ps -a | grep dl
then
sudo docker rm -f dl
sudo docker run --name dl -v /ws33:/deeplearniing deeplearniing:v2
else
sudo docker run --name dl -v /ws33:/deeplearniing deeplearniing:v2
fi
fi
```

See the list of available environment variables

Projects to build: jb6,
Blank project name in the list
Trigger when build is: Unstable
Trigger build without parameters: ☐

Figure 18: Build configuration of job jb2

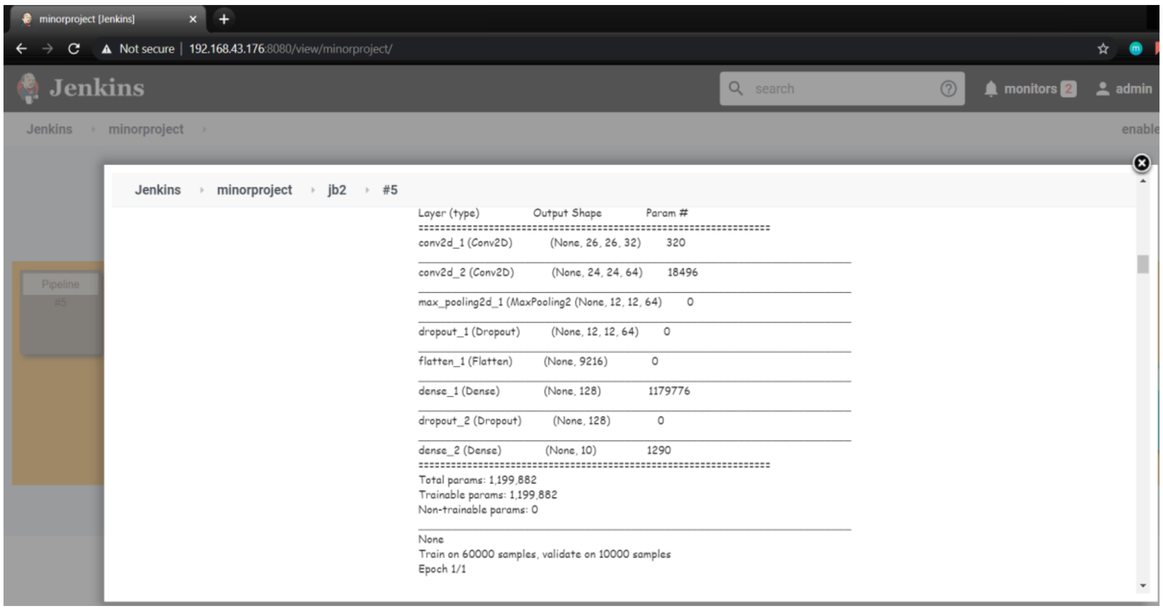
Once the job jb2 starts execution, it launches the docker container, which can be viewed in the Linux terminal, using the command **docker ps**:



```
[root@localhost Desktop]# cd
[root@localhost ~]# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED
     STATUS              PORTS               NAMES
9a21b9d06697        deeplearniing:v2    "python3 /deeplearni…"   14 seconds ago
     Up 1 second                             dl
[root@localhost ~]#
```

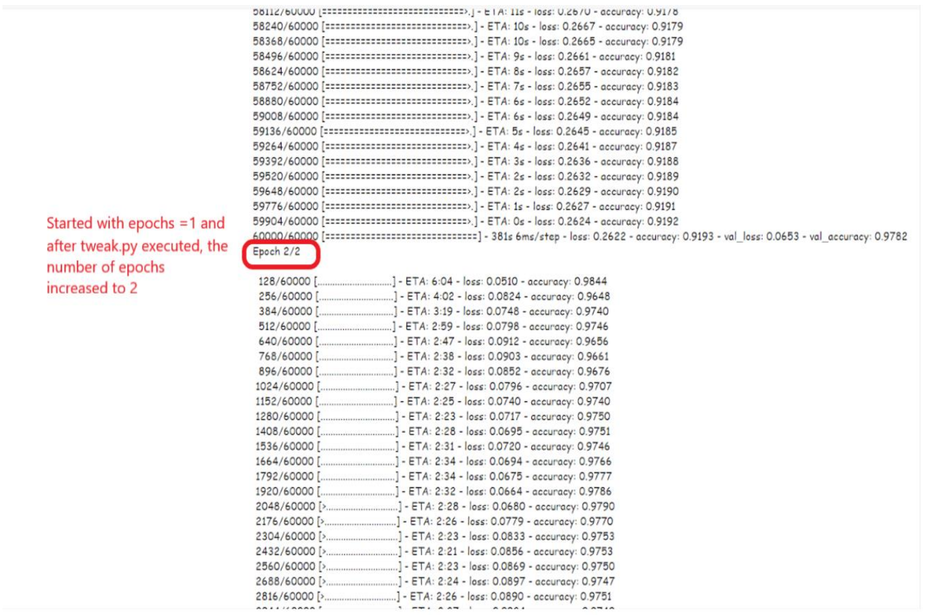
Figure 19: Docker container launched

The console output of the jb2 shows the training summary and the epochs:


minorproject [Jenkins]
Not secure | 192.168.43.176:8080/view/minorproject/
Jenkins
search
monitors 2
admin
Jenkins › minorproject ›
Jenkins › minorproject › jb2 › #5
Pipeline
#5

```
Layer (type)              Output Shape            Param #
=================================================================
conv2d_1 (Conv2D)          (None, 26, 26, 32)      320
_________________________________________________________________
conv2d_2 (Conv2D)          (None, 24, 24, 64)      18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 12, 12, 64)     0
_________________________________________________________________
dropout_1 (Dropout)        (None, 12, 12, 64)      0
_________________________________________________________________
flatten_1 (Flatten)        (None, 9216)            0
_________________________________________________________________
dense_1 (Dense)            (None, 128)             1179776
_________________________________________________________________
dropout_2 (Dropout)        (None, 128)             0
_________________________________________________________________
dense_2 (Dense)            (None, 10)              1290
=================================================================
Total params: 1,199,882
Trainable params: 1,199,882
Non-trainable params: 0
_________________________________________________________________
None
Train on 60000 samples, validate on 10000 samples
Epoch 1/1
```

Figure 20: Console Output of job jb2

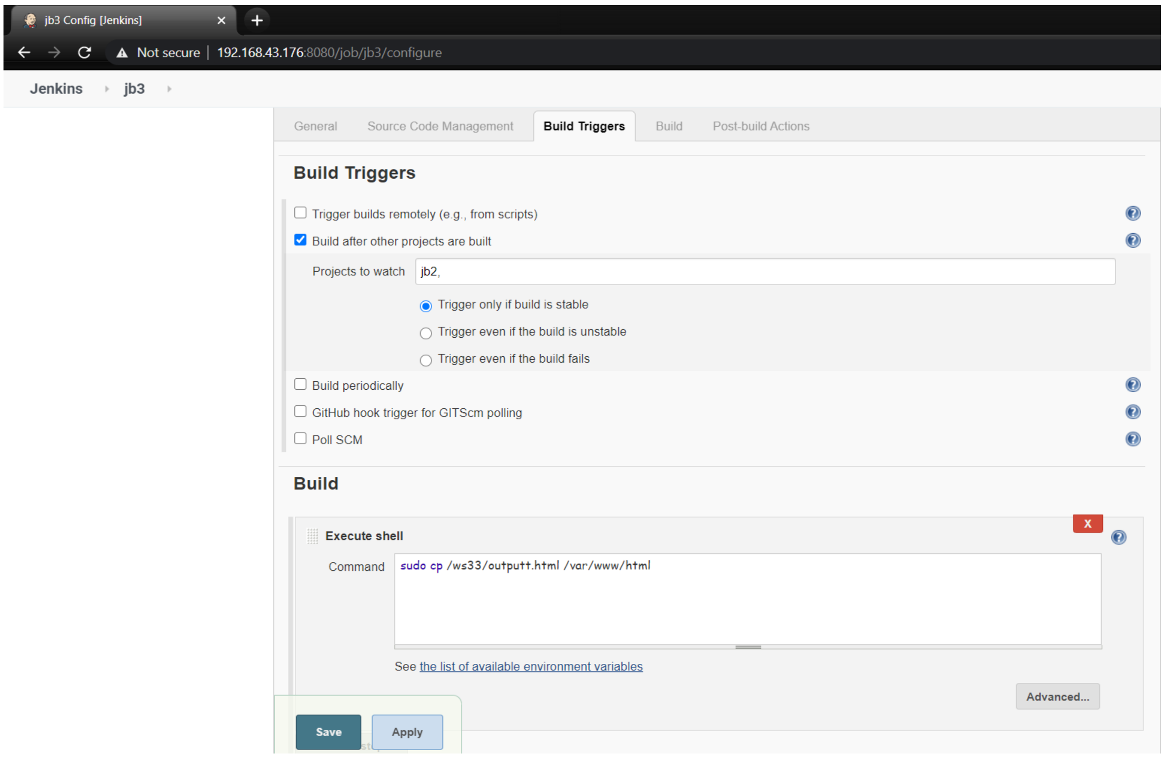

Started with epochs =1 and after tweak.py executed, the number of epochs increased to 2

```
58112/60000 [============================>.] - ETA: 11s - loss: 0.2670 - accuracy: 0.9178
58240/60000 [============================>.] - ETA: 10s - loss: 0.2667 - accuracy: 0.9179
58368/60000 [============================>.] - ETA: 10s - loss: 0.2665 - accuracy: 0.9179
58496/60000 [============================>.] - ETA: 9s - loss: 0.2661 - accuracy: 0.9181
58624/60000 [============================>.] - ETA: 8s - loss: 0.2657 - accuracy: 0.9182
58752/60000 [============================>.] - ETA: 7s - loss: 0.2655 - accuracy: 0.9183
58880/60000 [============================>.] - ETA: 6s - loss: 0.2652 - accuracy: 0.9184
59008/60000 [============================>.] - ETA: 6s - loss: 0.2649 - accuracy: 0.9184
59136/60000 [============================>.] - ETA: 5s - loss: 0.2645 - accuracy: 0.9185
59264/60000 [============================>.] - ETA: 4s - loss: 0.2641 - accuracy: 0.9187
59392/60000 [============================>.] - ETA: 3s - loss: 0.2636 - accuracy: 0.9188
59520/60000 [============================>.] - ETA: 2s - loss: 0.2632 - accuracy: 0.9189
59648/60000 [============================>.] - ETA: 2s - loss: 0.2629 - accuracy: 0.9190
59776/60000 [============================>.] - ETA: 1s - loss: 0.2627 - accuracy: 0.9191
59904/60000 [============================>.] - ETA: 0s - loss: 0.2624 - accuracy: 0.9192
60000/60000 [==============================] - 381s 6ms/step - loss: 0.2622 - accuracy: 0.9193 - val_loss: 0.0653 - val_accuracy: 0.9782
Epoch 2/2

  128/60000 [..............................] - ETA: 6:04 - loss: 0.0510 - accuracy: 0.9844
  256/60000 [..............................] - ETA: 4:02 - loss: 0.0824 - accuracy: 0.9648
  384/60000 [..............................] - ETA: 3:19 - loss: 0.0748 - accuracy: 0.9740
  512/60000 [..............................] - ETA: 2:59 - loss: 0.0798 - accuracy: 0.9746
  640/60000 [..............................] - ETA: 2:47 - loss: 0.0912 - accuracy: 0.9656
  768/60000 [..............................] - ETA: 2:38 - loss: 0.0903 - accuracy: 0.9661
  896/60000 [..............................] - ETA: 2:32 - loss: 0.0852 - accuracy: 0.9676
 1024/60000 [..............................] - ETA: 2:27 - loss: 0.0796 - accuracy: 0.9707
 1152/60000 [..............................] - ETA: 2:25 - loss: 0.0740 - accuracy: 0.9740
 1280/60000 [..............................] - ETA: 2:23 - loss: 0.0717 - accuracy: 0.9750
 1408/60000 [..............................] - ETA: 2:28 - loss: 0.0695 - accuracy: 0.9751
 1536/60000 [..............................] - ETA: 2:31 - loss: 0.0720 - accuracy: 0.9746
 1664/60000 [..............................] - ETA: 2:34 - loss: 0.0694 - accuracy: 0.9766
 1792/60000 [..............................] - ETA: 2:34 - loss: 0.0675 - accuracy: 0.9777
 1920/60000 [..............................] - ETA: 2:32 - loss: 0.0664 - accuracy: 0.9786
 2048/60000 [>.............................] - ETA: 2:28 - loss: 0.0680 - accuracy: 0.9790
 2176/60000 [>.............................] - ETA: 2:26 - loss: 0.0779 - accuracy: 0.9770
 2304/60000 [>.............................] - ETA: 2:23 - loss: 0.0833 - accuracy: 0.9753
 2432/60000 [>.............................] - ETA: 2:21 - loss: 0.0856 - accuracy: 0.9753
 2560/60000 [>.............................] - ETA: 2:23 - loss: 0.0869 - accuracy: 0.9750
 2688/60000 [>.............................] - ETA: 2:24 - loss: 0.0897 - accuracy: 0.9747
 2816/60000 [>.............................] - ETA: 2:26 - loss: 0.0890 - accuracy: 0.9751
```

Figure 21: Output of job jb2 showing model trained

6. Creating third job jb3 in Jenkins:

This job puts the accuracy obtained in the file /var/www/html as this is the default location for the Apache web server, so that the accuracy can be viewed in a webpage.


Build Triggers

- [ ] Trigger builds remotely (e.g., from scripts)
- [x] Build after other projects are built

Projects to watch: jb2,

(●) Trigger only if build is stable
( ) Trigger even if the build is unstable
( ) Trigger even if the build fails

- [ ] Build periodically
- [ ] GitHub hook trigger for GITScm polling
- [ ] Poll SCM

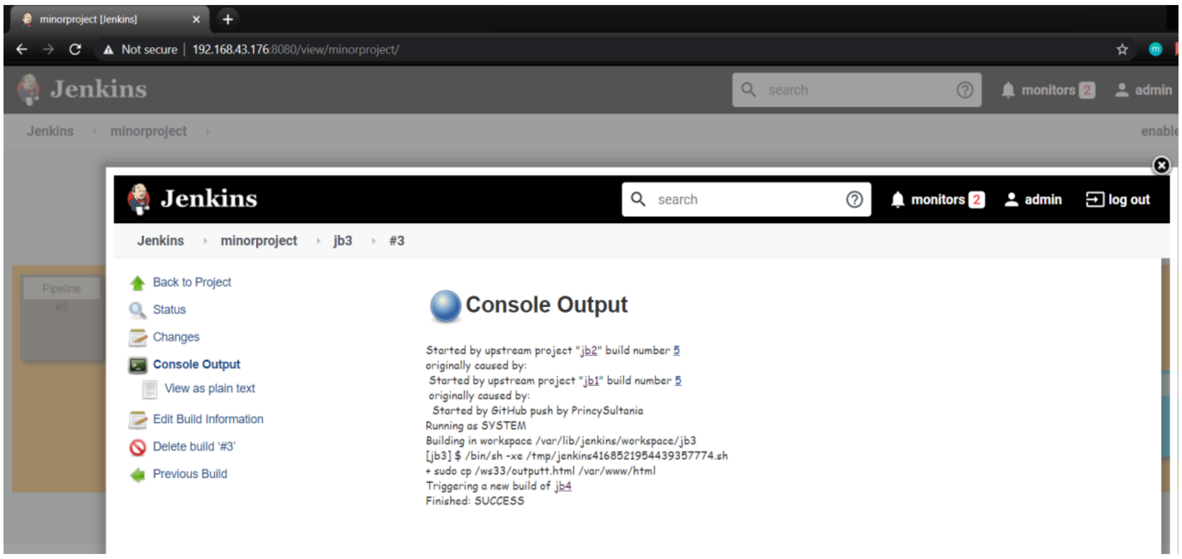
Build

Execute shell

Command:

```
sudo cp /ws33/outputt.html /var/www/html
```

Figure 22: Configuration of job jb3

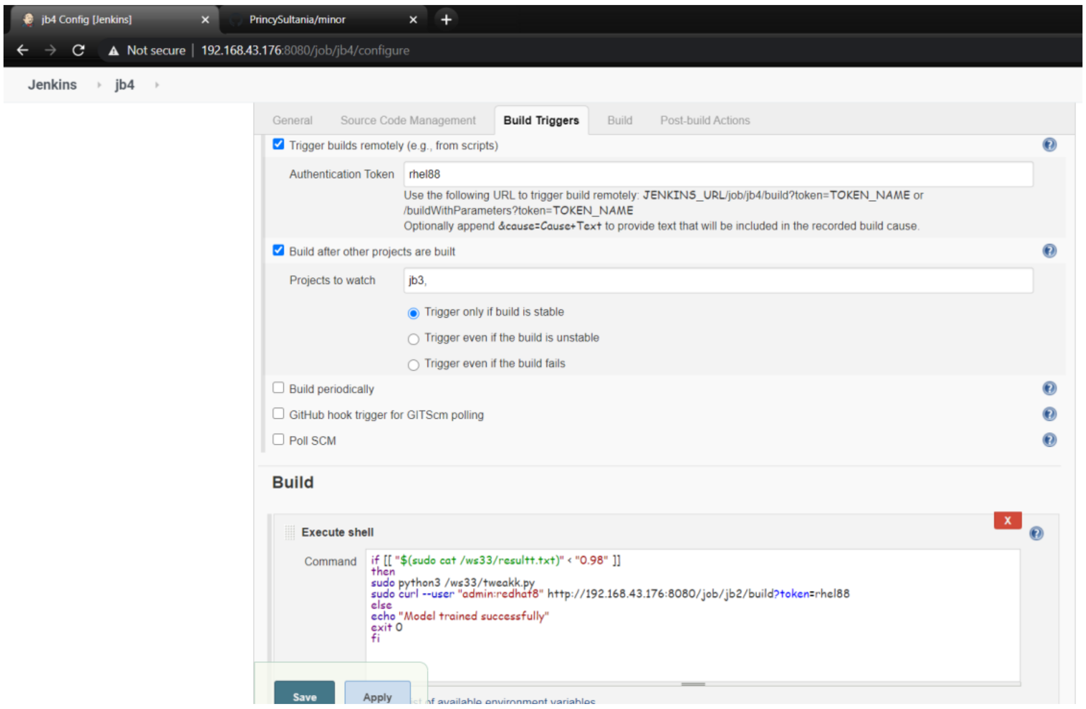

Console Output

```
Started by upstream project "jb2" build number 5
originally caused by:
 Started by upstream project "jb1" build number 5
 originally caused by:
  Started by GitHub push by PrincySultania
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/jb3
[jb3] $ /bin/sh -xe /tmp/jenkins4168521954439357774.sh
+ sudo cp /ws33/outputt.html /var/www/html
Triggering a new build of jb4
Finished: SUCCESS
```

Figure 23: Console output of job jb3

7. Creating fourth job jb4 in Jenkins:

This job analyses if the accuracy obtained in the resultt.py file is less than 98%, then it runs the tweakk.py file in order to do the hyper-tuning of the model by increasing the number of epochs in the original machine learning code. Else, it shows that the model is trained successfully.


Jenkins › jb4

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

☑ Trigger builds remotely (e.g., from scripts)

Authentication Token: rhel88

Use the following URL to trigger build remotely: JENKINS_URL/job/jb4/build?token=TOKEN_NAME or /buildWithParameters?token=TOKEN_NAME
Optionally append &cause=Cause+Text to provide text that will be included in the recorded build cause.

☑ Build after other projects are built

Projects to watch: jb3,

● Trigger only if build is stable
○ Trigger even if the build is unstable
○ Trigger even if the build fails

☐ Build periodically
☐ GitHub hook trigger for GITScm polling
☐ Poll SCM

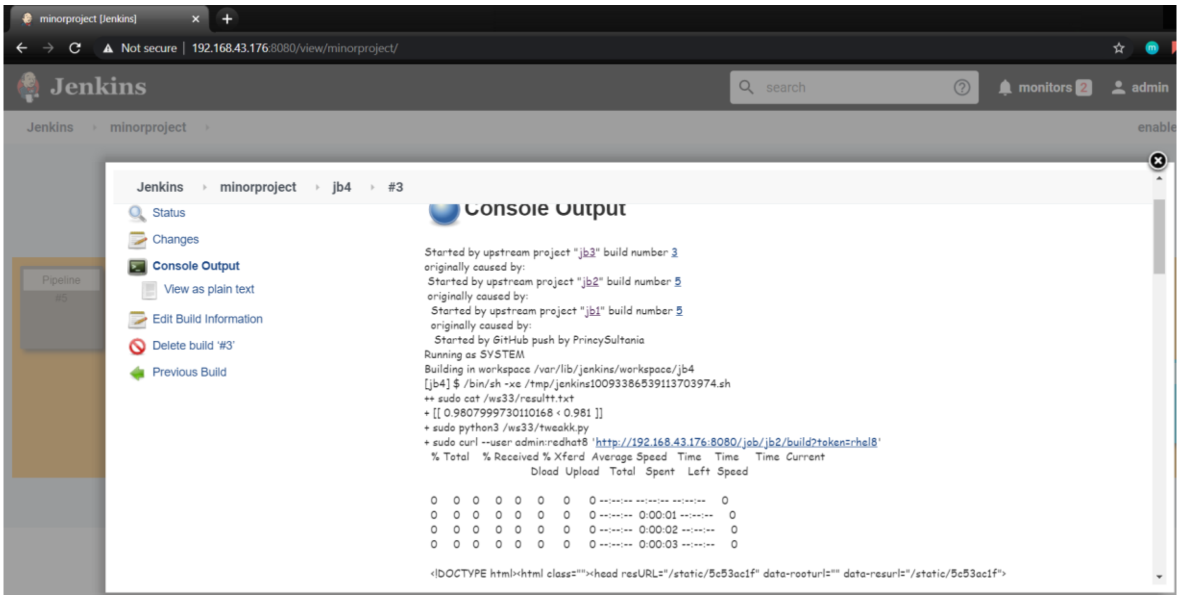
Build

Execute shell

Command:

```
if [[ "$(sudo cat /ws33/resultt.txt)" < "0.98" ]]
then
sudo python3 /ws33/tweakk.py
sudo curl --user "admin:redhat8" http://192.168.43.176:8080/job/jb2/build?token=rhel88
else
echo "Model trained successfully"
exit 0
fi
```

Figure 24: Configuration of job jb4


Jenkins › minorproject › jb4 › #3

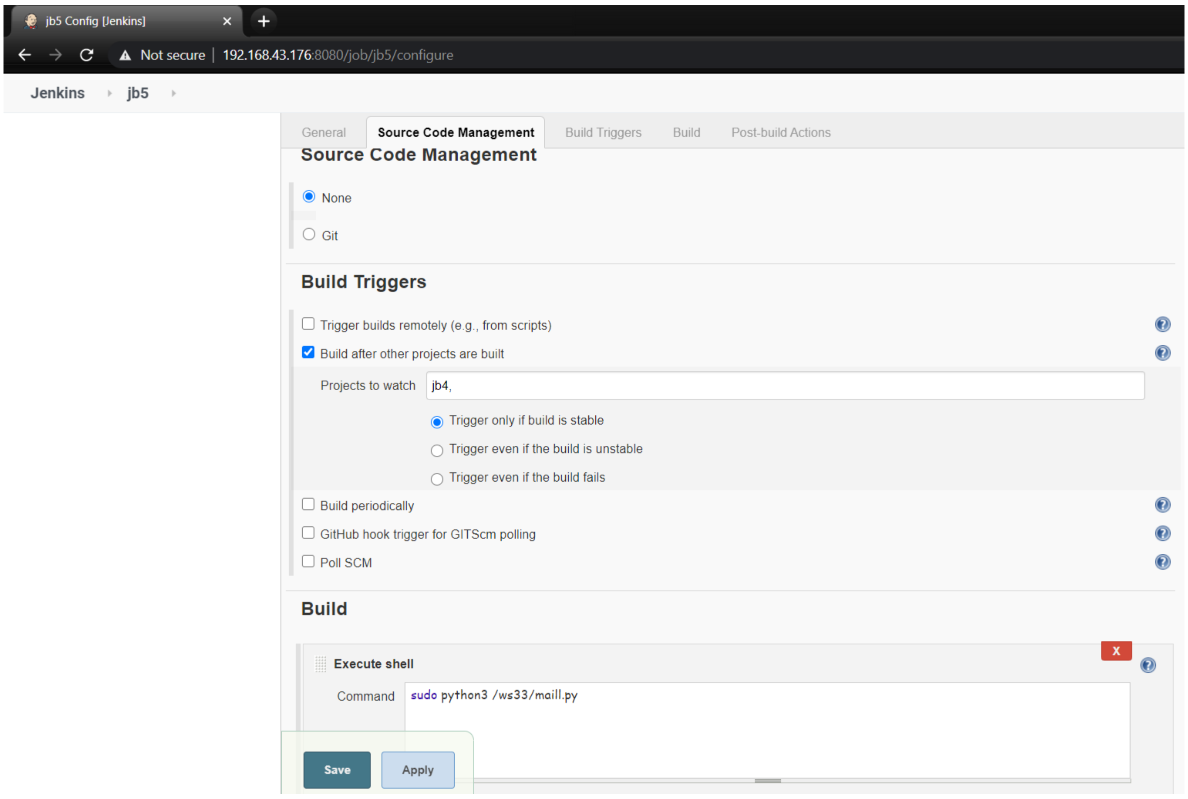
Console Output

```
Started by upstream project "jb3" build number 3
originally caused by:
 Started by upstream project "jb2" build number 5
 originally caused by:
  Started by upstream project "jb1" build number 5
  originally caused by:
   Started by GitHub push by PrincySultania
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/jb4
[jb4] $ /bin/sh -xe /tmp/jenkins10093386539113703974.sh
++ sudo cat /ws33/resultt.txt
+ [[ 0.9807999730110168 < 0.981 ]]
+ sudo python3 /ws33/tweakk.py
+ sudo curl --user admin:redhat8 'http://192.168.43.176:8080/job/jb2/build?token=rhel8'
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed

  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
  0     0    0     0    0     0      0      0 --:--:--  0:00:01 --:--:--     0
  0     0    0     0    0     0      0      0 --:--:--  0:00:02 --:--:--     0
  0     0    0     0    0     0      0      0 --:--:--  0:00:03 --:--:--     0

<!DOCTYPE html><html class=""><head resURL="/static/5c53ac1f" data-rooturl="" data-resurl="/static/5c53ac1f">
```

Figure 25: Console output of job jb4

8. Creating fifth job jb5 in Jenkins:

This job sends a success mail to the developer if the model’s accuracy is achieved more than the desired accuracy i.e. 98%



Source Code Management

(●) None
( ) Git

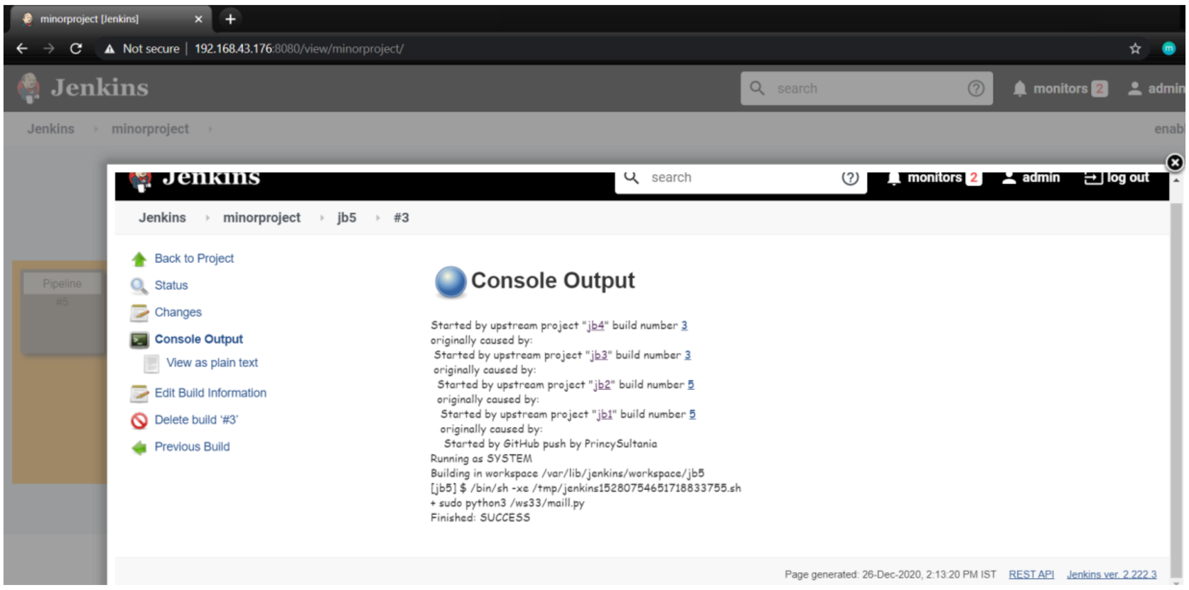
Build Triggers

☐ Trigger builds remotely (e.g., from scripts)
☑ Build after other projects are built
Projects to watch: jb4,
(●) Trigger only if build is stable
( ) Trigger even if the build is unstable
( ) Trigger even if the build fails
☐ Build periodically
☐ GitHub hook trigger for GITScm polling
☐ Poll SCM

Build

Execute shell
Command:
```
sudo python3 /ws33/maill.py
```

Figure 26: Configuration of job jb5



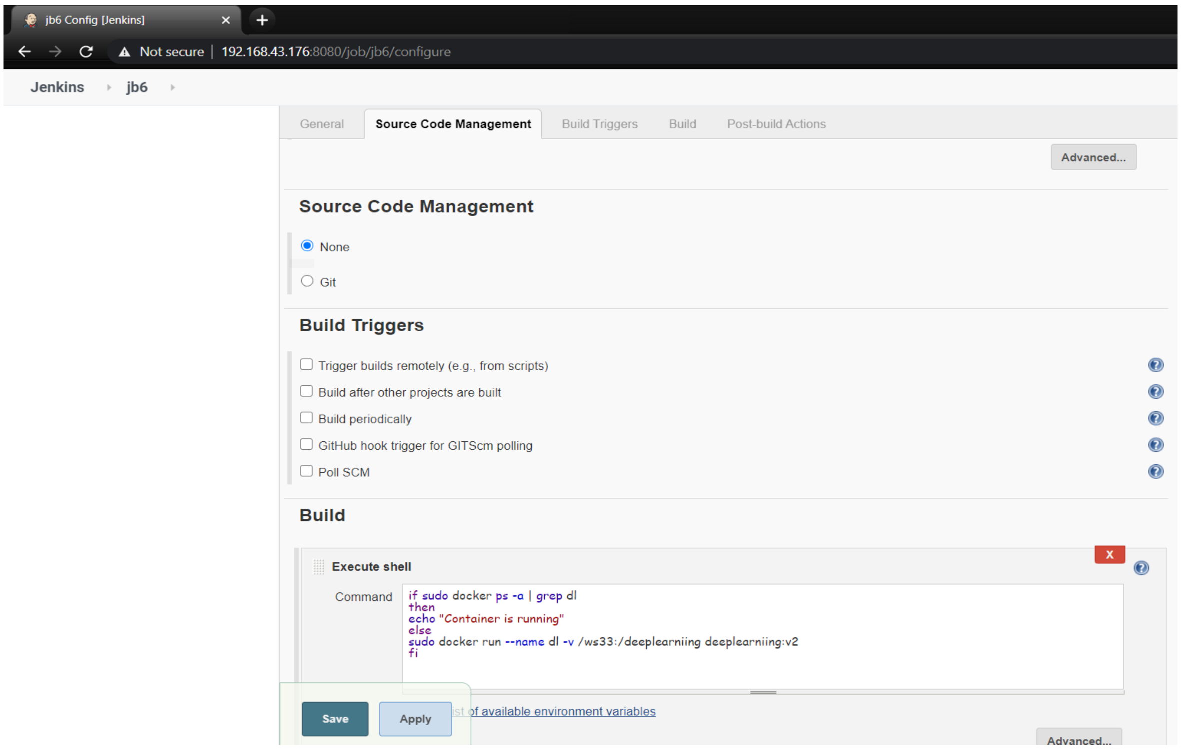
Console Output

```
Started by upstream project "jb4" build number 3
originally caused by:
 Started by upstream project "jb3" build number 3
 originally caused by:
  Started by upstream project "jb2" build number 5
  originally caused by:
   Started by upstream project "jb1" build number 5
   originally caused by:
    Started by GitHub push by PrincySultania
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/jb5
[jb5] $ /bin/sh -xe /tmp/jenkins1528075465171883755.sh
+ sudo python3 /ws33/maill.py
Finished: SUCCESS
```

Figure 27: Console output of job jb5

9. Creating sixth job jb6 in Jenkins:

This job will be launched in case the second job jb2 fails to launch the docker container.


jb6 Config [Jenkins]

Not secure | 192.168.43.176:8080/job/jb6/configure

Jenkins › jb6 ›

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

Advanced...

Source Code Management

- (●) None
- ( ) Git

Build Triggers

- ☐ Trigger builds remotely (e.g., from scripts)
- ☐ Build after other projects are built
- ☐ Build periodically
- ☐ GitHub hook trigger for GITScm polling
- ☐ Poll SCM

Build

Execute shell

Command

```
if sudo docker ps -a | grep dl
then
echo "Container is running"
else
sudo docker run --name dl -v /ws33:/deeplearniing deeplearniing:v2
fi
```

Save | Apply | List of available environment variables

Advanced...

Figure 28: Configuration of job jb6

10. Build pipeline view in Jenkins:

A build pipeline plugin is installed in Jenkins to view all the jobs in a pipeline view.

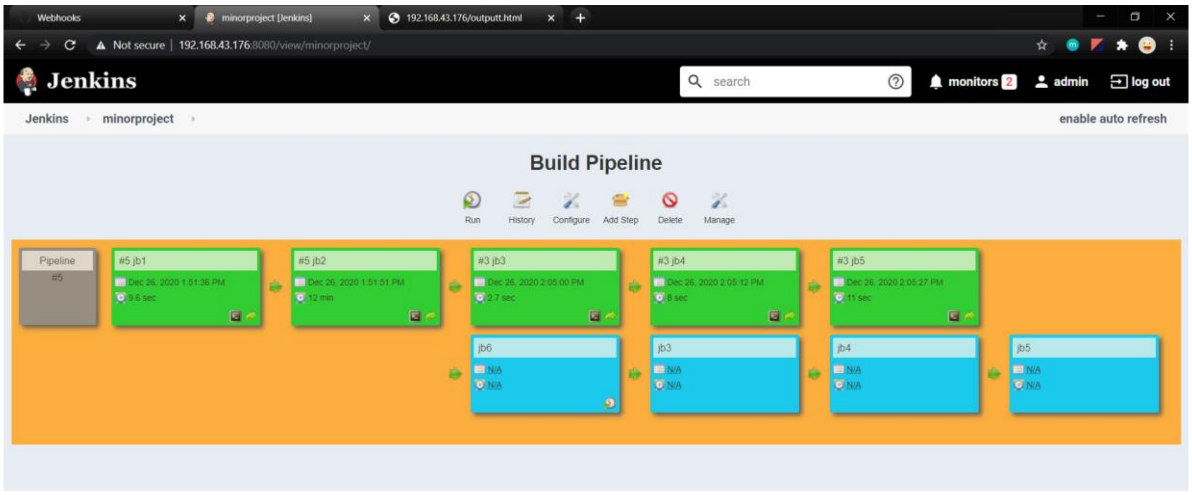

Webhooks
minorproject [Jenkins]
192.168.43.176/outputt.html
Not secure | 192.168.43.176:8080/view/minorproject/
Jenkins
search
monitors 2
admin
log out
Jenkins › minorproject ›
enable auto refresh
Build Pipeline
Run
History
Configure
Add Step
Delete
Manage
Pipeline #5
#5 jb1 Dec 26, 2020 1:51:36 PM 9.6 sec
#5 jb2 Dec 26, 2020 1:51:51 PM 12 min
#3 jb3 Dec 26, 2020 2:05:00 PM 2.7 sec
#3 jb4 Dec 26, 2020 2:05:12 PM 8 sec
#3 jb5 Dec 26, 2020 2:05:27 PM 11 sec
jb6 N/A N/A
jb3 N/A N/A
jb4 N/A N/A
jb5 N/A N/A

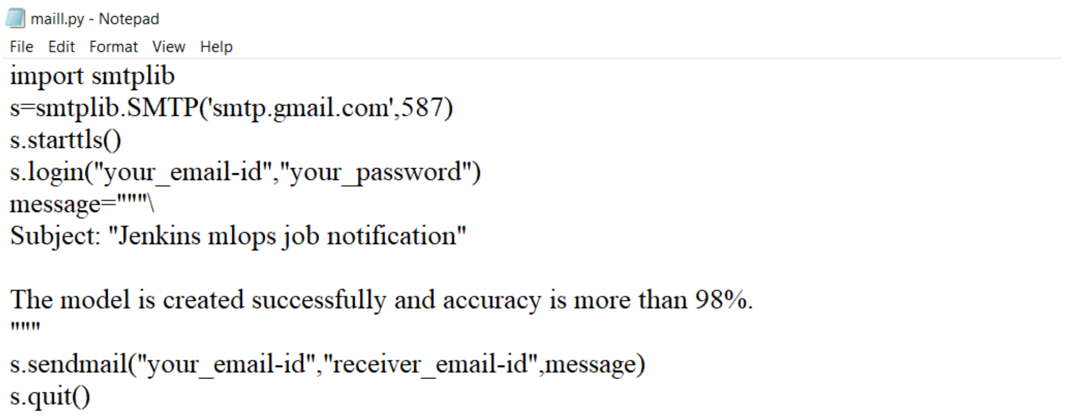
Figure 29: Build pipeline view

Chapter 5: SIMULATION CODE AND COMMANDS

The code of the various files used in the project:

1. maill.py

To send a mail to developer, a python code is executed, which uses a built-in library called **smtplib** as the protocol used for sending mail is Simple Mail Transfer Protocol (SMTP).

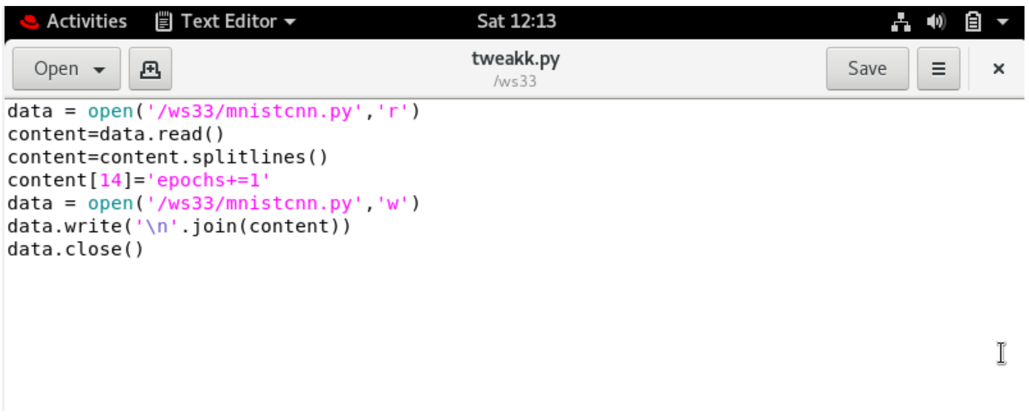


```
import smtplib
s=smtplib.SMTP('smtp.gmail.com',587)
s.starttls()
s.login("your_email-id","your_password")
message="""\
Subject: "Jenkins mlops job notification"

The model is created successfully and accuracy is more than 98%.
"""
s.sendmail("your_email-id","receiver_email-id",message)
s.quit()
```

Figure 30: maill.py file

2. tweakk.py

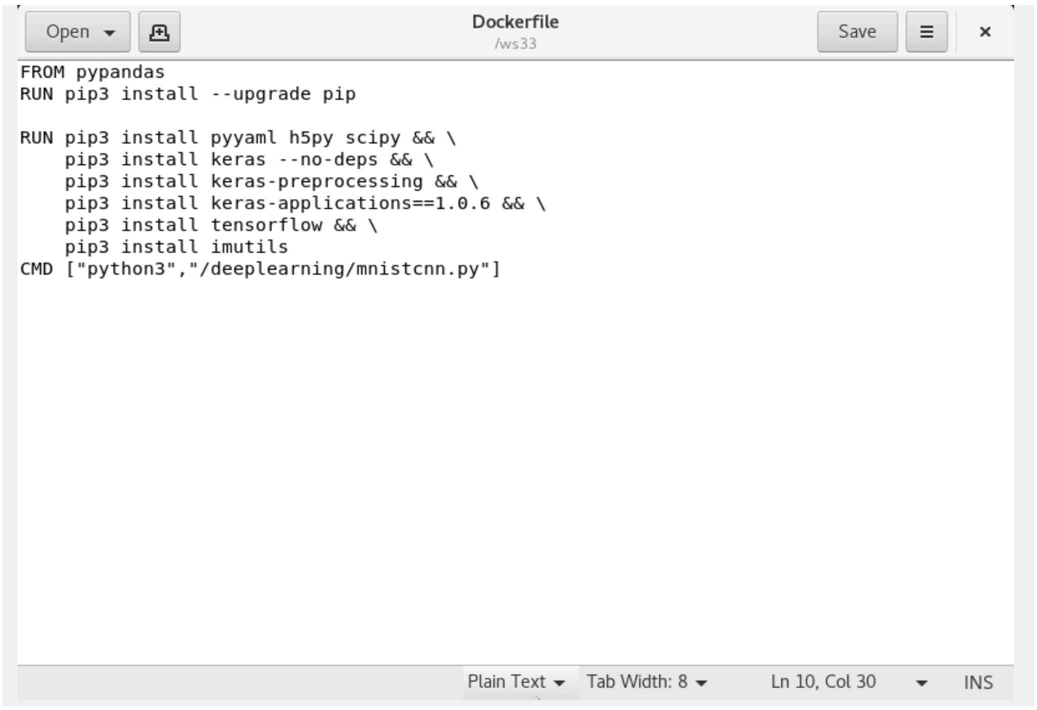
This tweakk.py file is used to hyper-tune the model by increasing the number of epochs. This file is created in the same folder where Dockerfile is created that is inside ws33 folder.



```
data = open('/ws33/mnistcnn.py','r')
content=data.read()
content=content.splitlines()
content[14]='epochs+=1'
data = open('/ws33/mnistcnn.py','w')
data.write('\n'.join(content))
data.close()
```

Figure 31: tweakk.py file

3. Dockerfile for creating the docker image:



```
FROM pypandas
RUN pip3 install --upgrade pip

RUN pip3 install pyyaml h5py scipy && \
    pip3 install keras --no-deps && \
    pip3 install keras-preprocessing && \
    pip3 install keras-applications==1.0.6 && \
    pip3 install tensorflow && \
    pip3 install imutils
CMD ["python3","/deeplearning/mnistcnn.py"]
```

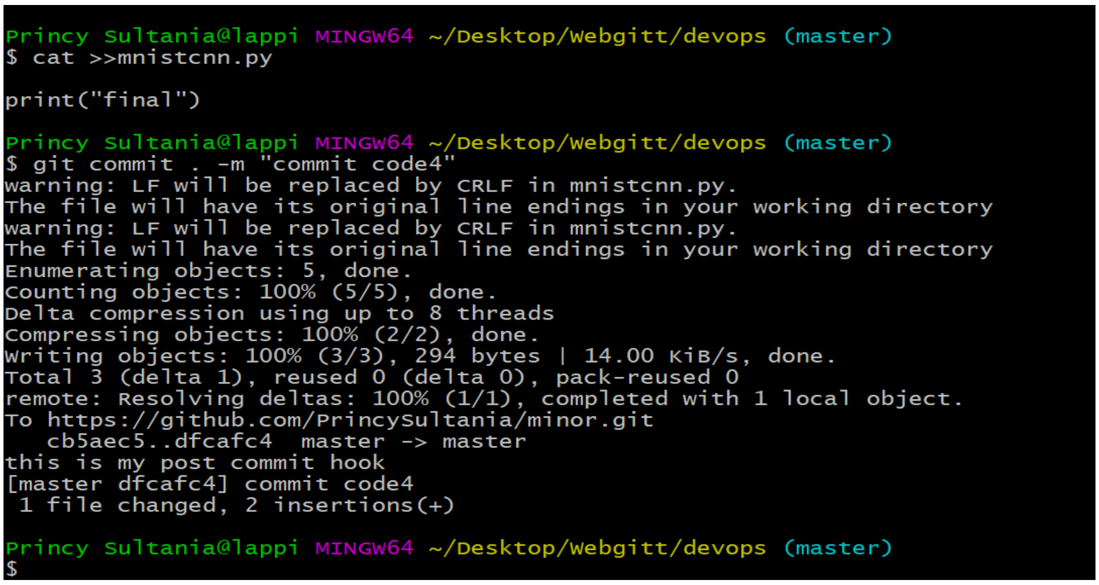
Figure 32: Dockerfile

4. inoutepochh.txt is a file which initially contains 1 inside it, which is for the initial number of epochs. This file is also created inside the ws33 folder in Linux OS.
5. resultt.txt and outputt.html are the empty files created inside ws33 folder. Once the program runs, these files will have some values written inside them.
6. mnistcnn.py is the program file containing the machine learning code for training and testing the MNIST handwritten dataset.

Chapter 6: INPUT / OUTPUT SCREENS

Input:

The program file mnistcnn.py containing the Machine learning code for training the model and getting accuracy is pushed from git to github and then jenkins:



```
Princy Sultania@lappi MINGW64 ~/Desktop/webgitt/devops (master)
$ cat >>mnistcnn.py

print("final")

Princy Sultania@lappi MINGW64 ~/Desktop/webgitt/devops (master)
$ git commit . -m "commit code4"
warning: LF will be replaced by CRLF in mnistcnn.py.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in mnistcnn.py.
The file will have its original line endings in your working directory
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 294 bytes | 14.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/PrincySultania/minor.git
   cb5aec5..dfcafc4  master -> master
this is my post commit hook
[master dfcafc4] commit code4
 1 file changed, 2 insertions(+)

Princy Sultania@lappi MINGW64 ~/Desktop/webgitt/devops (master)
$
```

Figure 33: Pushing source code to github

Output:

The build pipeline view of all the six jobs is created in the jenkins automation server. The green colour means that the task/ job executed successfully.

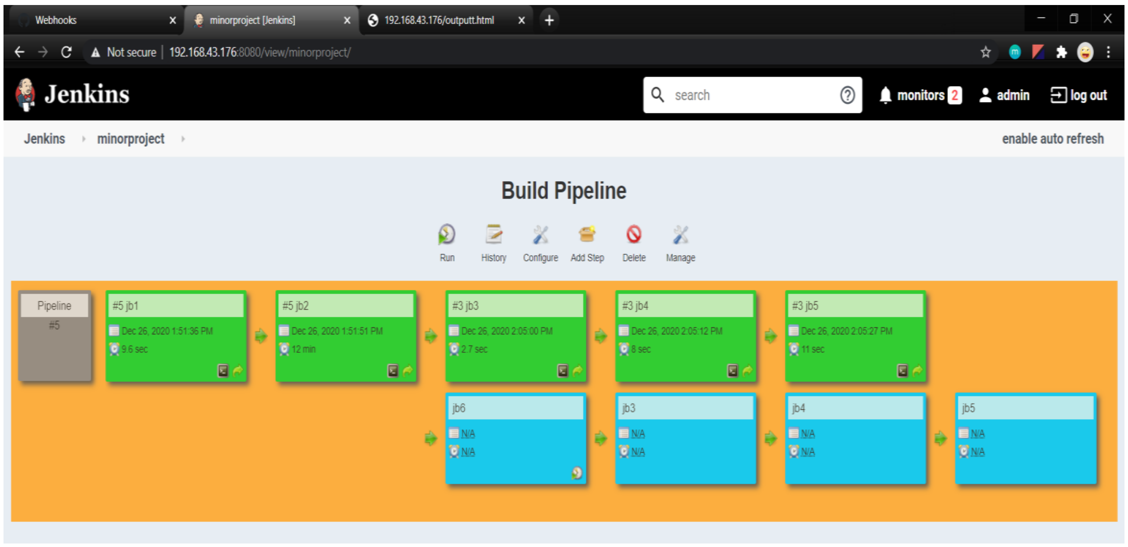
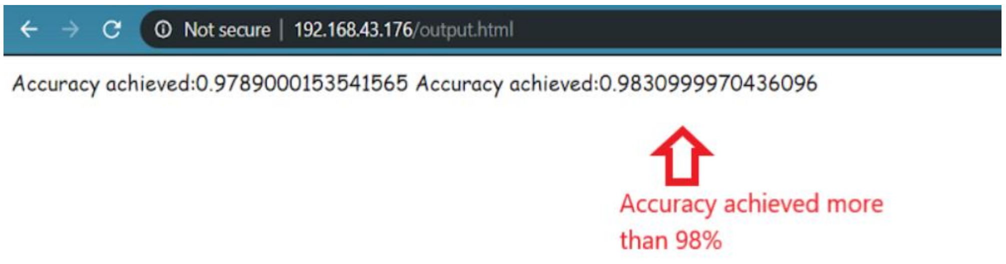


Figure 34: Final Pipeline view

The accuracy achieved can be seen at the url, which is the combination of the IP address of the RedHat 8 Linux Operating system and the outputt.html file.

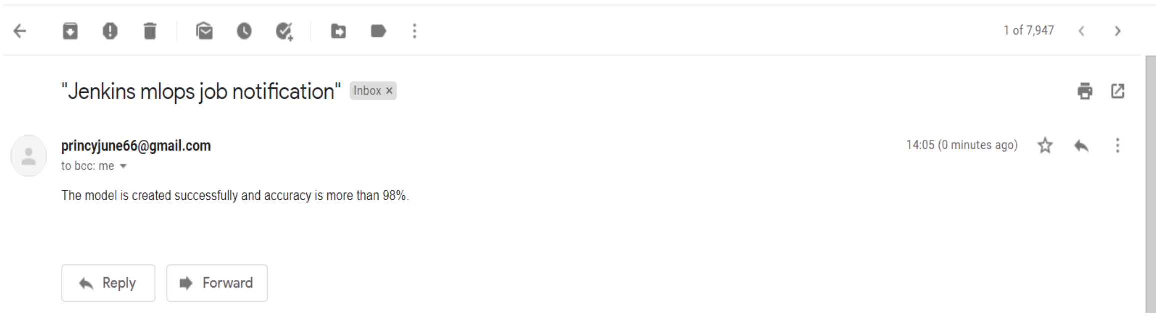

Not secure | 192.168.43.176/output.html

```
Accuracy achieved:0.9789000153541565 Accuracy achieved:0.9830999970436096
```

Accuracy achieved more than 98%

Figure 35: Accuracy achieved

Notification through email received:



"Jenkins mlops job notification"

princyjune66@gmail.com
to bcc: me
14:05 (0 minutes ago)

The model is created successfully and accuracy is more than 98%.

Figure 36: Email notification

Chapter 7: CONCLUSION

7.1 LIMITATIONS

This project was a simple demo of how pipeline and automation are used and helpful in implementing the idea of greater automation of software industry in terms of DevOps along with Machine Learning.

Hyper parameter tuning is a tiring task for a data scientist, as it is simply based on hit and trial methodology. Here, this project only implements tuning of one of the hyper-parameters. But there are many other parameters which can be tuned. Also, as this was a simple demonstration of automating a pipeline for machine learning purposes using one hyper-parameter, it can be enhanced to be able to tune many other hyper-parameters along with the number of epochs.

Also, in this project, a famous already trained dataset was used, so getting a better accuracy was easier.

7.2 SCOPE FOR FUTURE WORK

Using cloud can be the next step to enhance this project further, instead of using Docker container technology, as cloud is much faster and maintainable.

This project can be extended to be able to tune many other parameters on which the accuracy of the model depends.

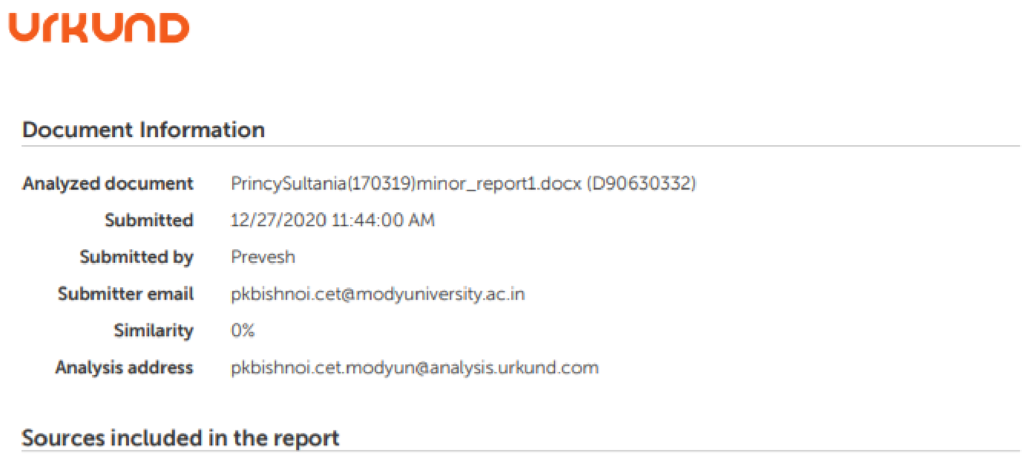
This automation can be applied to many new real life dataset in order to improve the actual performance and accuracy of the model.

Chapter 8: BIBLIOGRAPHY

1. https://www.freecodecamp.org/news/what-is-git-and-how-to-use-it-c341b049ae61/
2. https://www.infoworld.com/article/3204171/what-is-docker-the-spark-for-the-container-revolution.html
3. https://en.wikipedia.org/wiki/Docker_(software)
4. https://en.wikipedia.org/wiki/DevOps
5. https://en.wikipedia.org/wiki/Jenkins_(software)
6. https://searchitoperations.techtarget.com/definition/GitHub

Chapter 9: ANNEXURES

PLAGIARISM REPORT:



urkund

Document Information

Analyzed document	PrincySultania(170319)minor_report1.docx (D90630332)
Submitted	12/27/2020 11:44:00 AM
Submitted by	Prevesh
Submitter email	pkbishnoi.cet@modyuniversity.ac.in
Similarity	0%
Analysis address	pkbishnoi.cet.modyun@analysis.urkund.com

Sources included in the report