Mini Project Report

On

"Machine Learning for CyberSecurity"

Submitted in partial fulfillment for the award of the degree

of

Third Year

In

INFORMATION TECHNOLOGY

Submitted by

1. **Nehal Janardhan Ingole (19141228)**
2. **Gaurav Bhausheb Pagare (19141245)**

Under the Guidance of

Prof. R. S. Mawale

Government College of Engineering, Karad, Maharashtra, India

(An Autonomous Institute of Government of Maharashtra)

Academic Year 2021 - 2022

Government College of Engineering,

Karad, Maharashtra, India

(An Autonomous Institute of Government of Maharashtra)

Department of Information Technology

CERTIFICATE

This is to certify that the project entitled "Machine Learning for CyberSecurity" has been carried out by the team:

1. Nehal Janardhan Ingole (19141228)

2. Gaurav Bhausheb Pagare (19141245)

of T. Y. B. Tech IT class under the guidance of Prof. R. S. Mawale during the

The academic year 2021 -22 (Sem- VI).

Prof. R. S. Mawale **Dr. S. J. Wagh Head**

Project Guide **Head**
Information Technology Department

External Examiner

ACKNOWLEDGEMENT

Apart from individual efforts, the success of any project depends largely on the encouragement and guidelines of many others. We take this opportunity to express our gratitude to the people who have been instrumental throughout the project work.

It is our privilege to express our gratitude towards our project guide, Prof. R. S. Mawale for her valuable guidance, encouragement, inspiration and whole-hearted cooperation throughout the project work. We thank him for being a motivation through all our highs and importantly, our lows.

We deeply express our sincere thanks to our Head of Department Dr. S. J. Wagh for encouraging and allowing us to present the project on the topic “Machine Learning for CyberSecurity” and providing us with the necessary facilities to enable us to fulfill our project requirements as best as possible. We take this opportunity to thank all faculty members and staff of the Department of Information Technology, who have directly or indirectly helped our project.

We pay our respects to honorable Principal Dr. A. T. Pise for their encouragement. Our thanks and appreciations also go to our family and friends, who have been a source of encouragement and inspiration throughout the duration of the project

ABSTRACT

This age is the age of Automation, the machines are doing our work, a lot of calculations and managing the big task from general household to big corporates. As the world is developing with the technologies. These all stuff are good but it also comes with great risk of security and one of the most common things which we used on a daily basis is the websites. On the web, the most common types of attacks are DOS-based attacks. This kind attacks is very common and anyone with good knowledge can do these kinds of attacks. These kinds of attack can heavely impact our resource consumption and might take our website down resulting in loss of business

In order to prevent these things, we have developed a project in which we will secure webserver from DoS attacks using Machine Learning. In this project we are using K-mean clustering algorithm to detect the suspicious IP's which are trying to perform such attacks. We will automate the complete flow via the jenkins

Once the suspicious IP's are detected and found we can either hand over them to our security teams, take some action on them or block them from using our website

LIST OF FIGURES

Table of Content

Chapter 1

UnderStanding the Project

1.1 Background

A Denial-of-Service (DoS) attack is an attack meant to shut down a machine or network, making it inaccessible to its intended users. DoS attacks accomplish this by flooding the target with traffic, or sending it information that triggers a crash. In both instances, the DoS attack deprives legitimate users (i.e. employees, members, or account holders) of the service or resource they expected.Victims of DoS attacks often target web servers of high-profile organizations such as banking, commerce, and media companies, or government and trade organizations. Though DoS attacks do not typically result in the theft or loss of significant information or other assets, they can cost the victim a great deal of time and money to handle.

1.2 Importance of the project :

With the implementation of these kinds of projects in the industry environment these DOS attack can be prevented or stoped. Currently the project been designed is capable of detecting the DOS attack but using similar approach models can be created for different kinds of attacks too (tipically for pattern based attacks). The project reduces the task of the security team in the organization as we have automated most of the stuff using jenkins.

1.3 Motivation :

Security have been an major issue in todays digital world organizations does there business via the websites/ webapp. But here we observer there are many kinds of attack / threats on them. Most common is the DOS- attack. This not only consumes resources and energy of the organization but also lead in loss of business. For Security and monitoring team via the manual ways and methodology its very challenging to observe and protect our websites 24 * 7 from threats and hackers. So here automation comes into picture we can use automation tools and its integration with machine learning to solve these cyber security requirements

1.4 Scope :

=> The approach used for creating machine learning model for the DOS attack using similar approach model can be developed for other patter based attacks too.

=> Internally for creation of the model we have use the K-mean clustering algorithm (Discussed further in detail). And in this algorithms we need to decide the hyper parameter i.e number of cluster to be formed this is decided by analysis over the data (elbow method). Now we can provide this report of analysis frequently to our datascientest team so that if any changes need in code in future it can be done. This will reduce task of datascientest as well as there will be continous improvement in the project

1.5 Expected Outcome :

On successful implementation of the project we will be detecting the Suspecious IP as well as blocking those from accessing the website by triggering the firewalld rules.

Chapter 2

Literature Survey

2.1 K-Means Clustering Approach to Analyze NSL-KDD Intrusion Detection Dataset (LINK)

In this paper the author have present the complete analysis of NSL-KDD Intrusion Detection Dataset using clustering approach. Simple K-Means clustering algorithm is used to identify the different clusters and group them in four major attack categories. They have also provide a complete analysis of different kind of attacks present in training and testing dataset. Testing dataset was used to validate the results. Performance of the algorithm is investigated during different execution of the program on the input data set. The execution time for the algorithm is also analyzed.

Chapter 3

Releated Theory and Problem Defination

3.1 Problem Definition:

Our Final Goal is to an automated system which will be very useful for a server in following ways:

=> This system will keep log of the information about the clients hit or request to the server.

=> This log data of clients will be used for finding the unusual pattern of a client request for example if a client is sending request repeatedly. (DOS -attack)

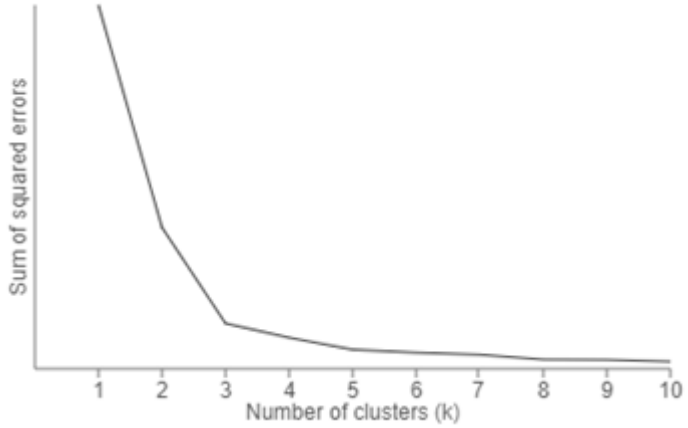
=> If any kind of unusual pattern we got then we can use Jenkins to perform certain task for example it will trigger the firewall to block that IP which is causing this trouble/problem.

3.2 Related Theory:

Here for detecting the suspecious IP we are using the K-mean clustering algorithim. In the K-means clustering algorithm data points are assigned into K groups, where K represents the number of clusters based on the distance from each group's centroid. The data points closest to a given centroid will be clustered under the same category. A larger K value will be indicative of smaller groupings with more granularity whereas a smaller K value will have larger groupings and less granularity. K-means clustering is commonly used in market segmentation, document clustering, image segmentation, and image compression.

Now in this K-means clustering algorithm one question arises how to decide the value of K One of the methods is called "Elbow" method can be

used to decide an optimal number of clusters. Here you would run K-mean clustering on a range of K values and plot the “percentage of variance explained” on the Y-axis and “K” on X-axis.



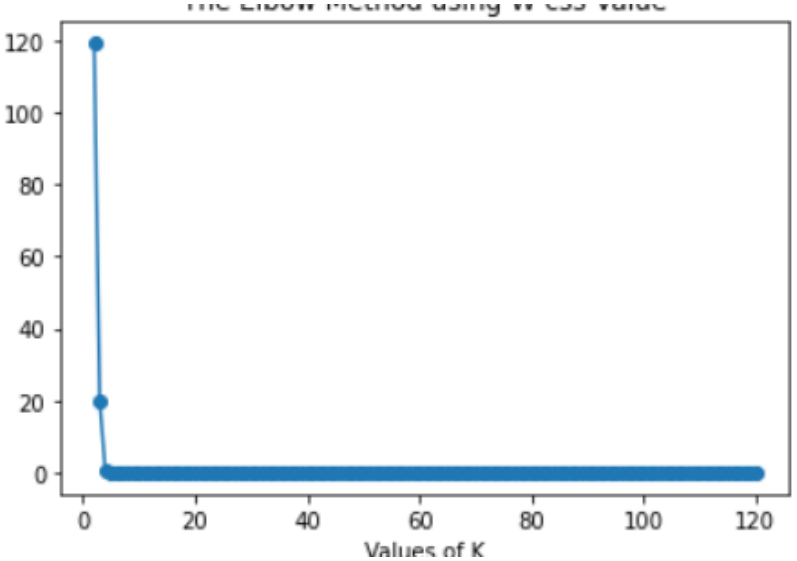
3.1 Elbow method Theory 1

In the picture above we can see that as we add more clusters after 3 it doesn't give much better modeling on the data. The first cluster adds much information, but at some point, the marginal gain will start dropping.

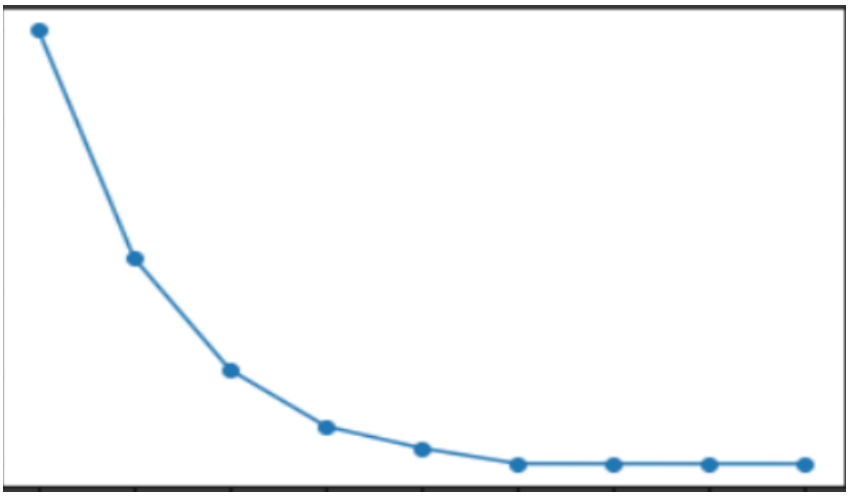
The sum of squared distance is also called as wcss value and in python we can find this using the inertia_ keyword. For our requirement each time we can get variety of results but we know there can be at least 2 kind of output suspicious or non suspicious so minimum value of K is 2.

3.2.1 Code for finding value of K:

```
inertias = []

K = range(2,len(training_data)+1)

for k in K:

    kmeanModel = KMeans(n_clusters=k)

    kmeanModel.fit(data_scaled)

    inertias.append(kmeanModel.inertia_)

plt.plot(K, inertias, 'o-')

plt.xlabel('Values of K')

plt.ylabel('W-css')

plt.title('The Elbow Method using W-css Value')

plt.show()
```



3.2 Elbow method Theory 2

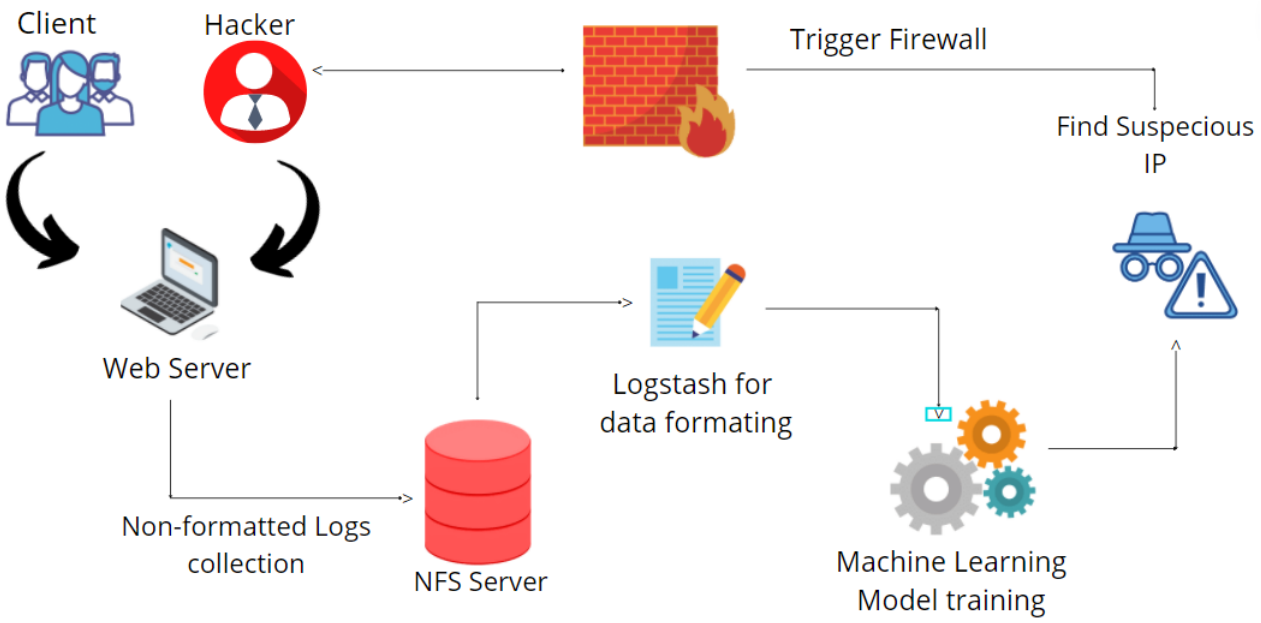


3.3 Elbow method Theory 3

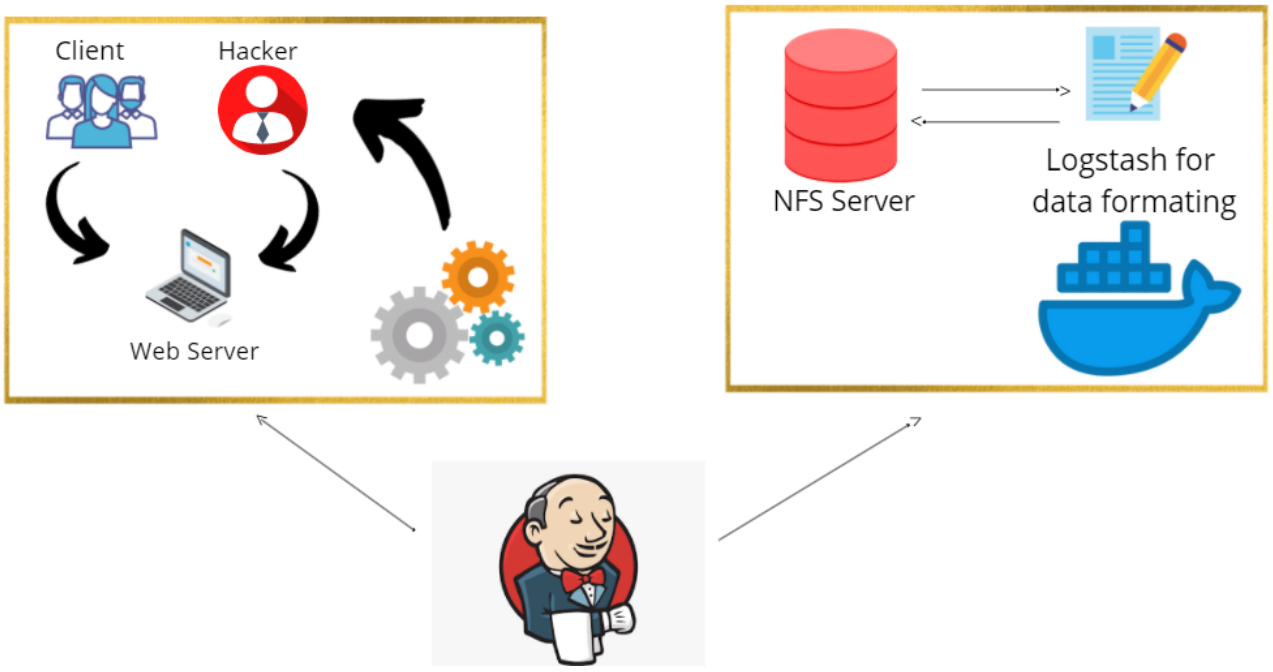
Chapter 4

Design Methodology

4.1 Proposed system Architecture:



4.1 Arcticture of Dos attack prevention system



4.2 Using Jenkins to Establish Same setup

4.2 Internal Logic of the System:

=> At the most we will be hosting our demo website using the apache httpd web server

=> In order to replicate the real world senerio we will hitting our webserver as genuin client and with one of the virtual machine we will try to perform DOS attack over the web server

=> Now apache web server generates the logs which we are storing in the NFS server so that those log can be access by other teams in the organization

=> The logs collected and present in the NFS can't be directly given to the machine learning model so for this purpose we will use logstach for this transformation opration

=> Now we will give the formatted logs (csv file) to the machine learning code. Now the model will be created with this code

=> further over the data depending on the number of hits done by the client in a particular time span i.e approx >3000 request within an hour. If such cases are identified they will be considered to be suspicious

=> List of this suspicious IP will be used and firewalld rule will be created to block those IP from using the website.

=> Ok now we know the final architecture which we need to achieve. Now we can do the same this with automation using jenkins

=> In the jenkins we have one master node setup and 2 node i.e one is web server and other is the one with docker and nfs running

=> In the jenkins we will be configuring the multiple jobs i.e one job to get the raw data from webserver to the nfs server. Further once that job is

done other job will be automatically get triggered i.e to launch a logstash docker container.

=> Further we will be training our model via another job getting trigged after logstash job is done. The output generated will be the the list of suspicious IP. this will be stored in BlockedIP.txt file (discussed further)

=> Now using the suspicious IP we will be triggering the firewall rule

4.3 Technical Specifications:

Hardware Requiements: (Note all the nodes are virtual machines)

=> Jenkins-Master Node

RAM	8GB and Above
HDD	50 GB
Image	RHEL-8
Processor	Intel® Core™ i5

=> Web-Server (Ec2- instance)

Image (AMI)	RHEL-8
Instance-type	T2.micro (1 CPU, 1 GB ram)

=> Docker-NFS-server (Ec2- instance)

Image (AMI)	RHEL-8
Instance-type	T2.large (2 CPU, 8 GB ram)

Software Requiements (dependencies for the softwares and some common s/w eg. vim, git, yum-utils, etc if not available is installed further):

1. Jenkins - Master
    a. Java-OpenJDK
    b. Jenkins package
    c. Python3.6
2. Web-Server
    a. Httpd package
    b. Java-OpenJDK
    c. Python3.6
        i. Os-sys module
        ii. Time module
3. Docker-NFS-server
    a. NFS-server package
    b. Docker-ce package
    c. Java-OpenJDK
    d. Python3.6
        i. Skit learn module
        ii. Numpy
4. Windows Machine
    a. Putty
    b. HTML/CSS
    c. Browser (Google Chrome)

Chapter 5

Implementation

5.1 Understanding the tools and technologies used:

Python :

Python is an interpreted high-level general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

HTML , CSS :

HTML (Hypertext Markup Language) is responsible for the structure of a web page. For example, you can define headings, paragraphs, texts, and images in HTML.CSS (Cascading Style Sheets) is responsible for the style and layout of a web page. You can define styles, such as colors, fonts, margins, and you can even create simple animations in CSS.Both

languages, HTML and CSS, are independent and should be saved in separate files.

Putty:

Putty is a free implementation of SSH (and telnet) for PCs running Microsoft Windows (it also includes an xterm terminal emulator). You will find PuTTY useful if you want to access an account on a Unix or other multi-user system from a PC. Here we have used it to connect to our jenkins master as well as the instances running in the AWS cloud

Amazon Web Server

AWS (Amazon Web Services) is a comprehensive, evolving cloud computing platform provided by Amazon that includes a mixture of infrastructure as a service (IaaS), platform as a service (PaaS) and packaged software as a service (SaaS) offerings. AWS services can offer organization tools such as compute power, database storage and content delivery services.

Jenkins

Jenkins is used to build and test your product continuously, so developers can continuously integrate changes into the build. Jenkins is

the most popular open source CI/CD tool on the market today and is used in support of DevOps, alongside other cloud native tools.

Docker

Docker is an open source containerization platform. It enables developers to package applications into containers—standardized executable components combining application source code with the operating system (OS) libraries and dependencies required to run that code in any environment. Containers simplify delivery of distributed applications, and have become increasingly popular as organizations shift to cloud-native development and hybrid multicloud environments.

Httpd

httpd is the Apache HyperText Transfer Protocol (HTTP) server program. It is designed to be run as a standalone daemon process. When used like this it will create a pool of child processes or threads to handle requests.

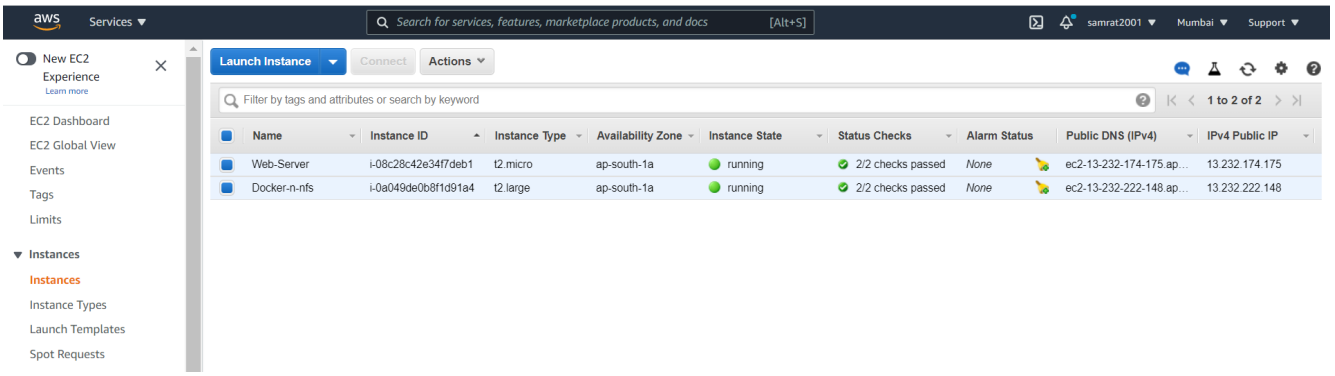
NFS Services:

NFS services uses Network File Sharing (NFS) is a protocol that allows you to share directories and files with other Linux clients over a network. Shared directories are typically created on a file server, running

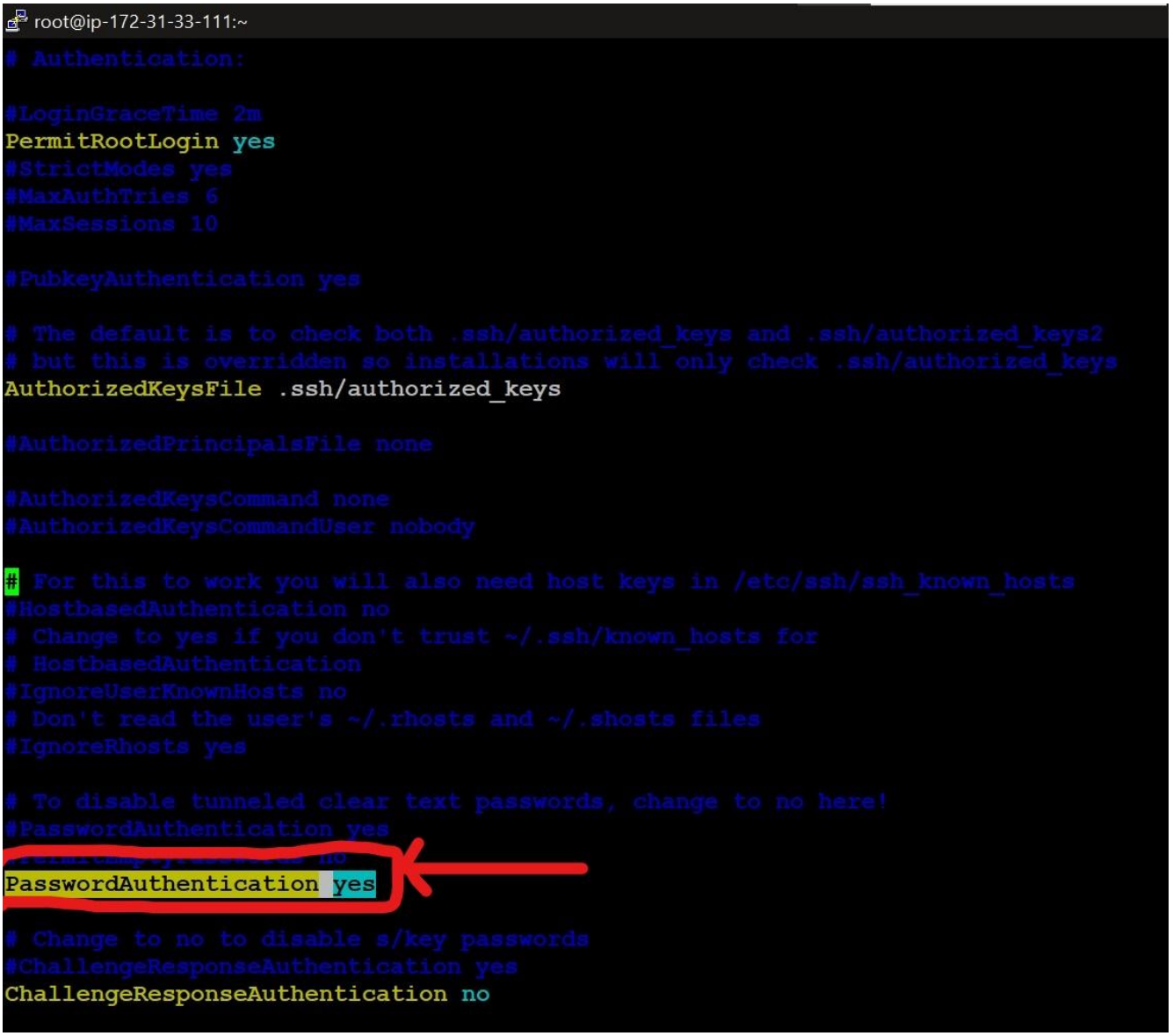
the NFS server component. Users add files to them, which are then shared with other users who have access to the folder. An NFS file share is mounted on a client machine, making it available just like folders the user created locally. NFS is particularly useful when disk space is limited and you need to exchange public data between client computers

5.2 Setting up the requied Environment:

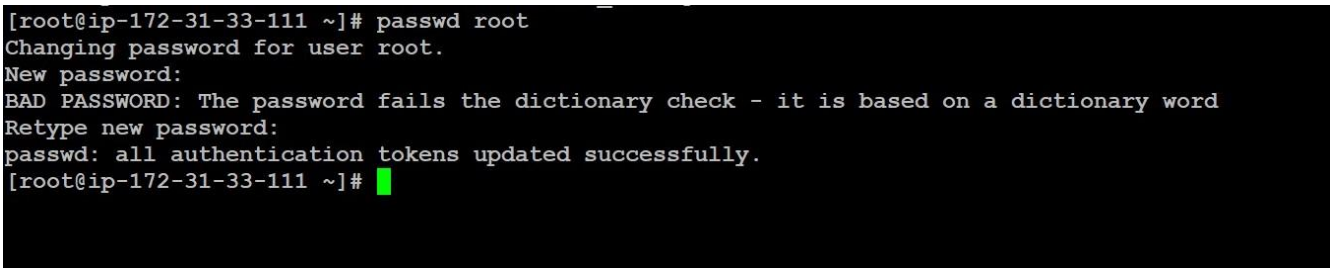
Step 1: Launch 2 ec2-instances on AWS instance with RHEL-8 AMI. Also one instance with t2.micro machine-type and other with t2.large machine-type . Here we are going to use instance with t2.micro machine-type for web-services and Instance for t2.large machine-type for Docker and NFS server. We have also tagged them with same names too.

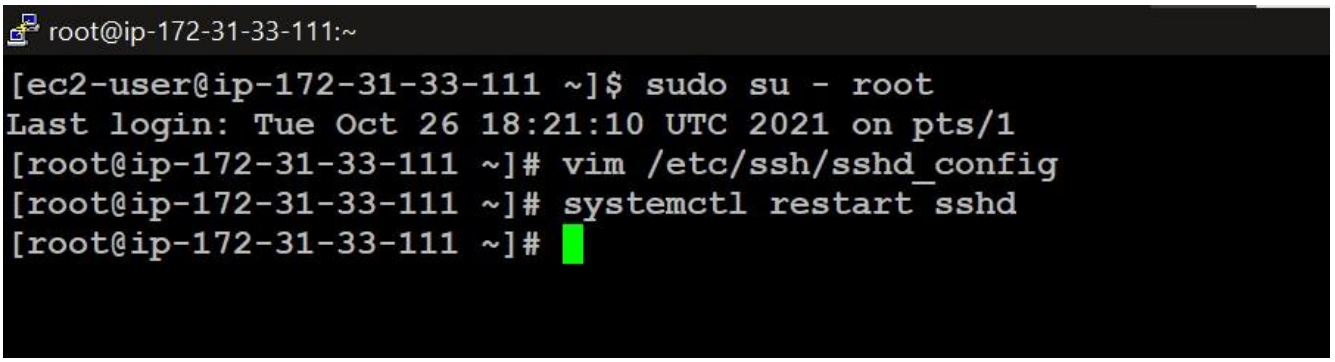


Step 2: Login to the instances with the ec2-user as username and authentication key. Now for easy access we will first allow the root login via password in both the instances for this we need to do changes in /etc/ssh/sshd_config file. Here we need to mention PasswordAuthincation yes by default its no.



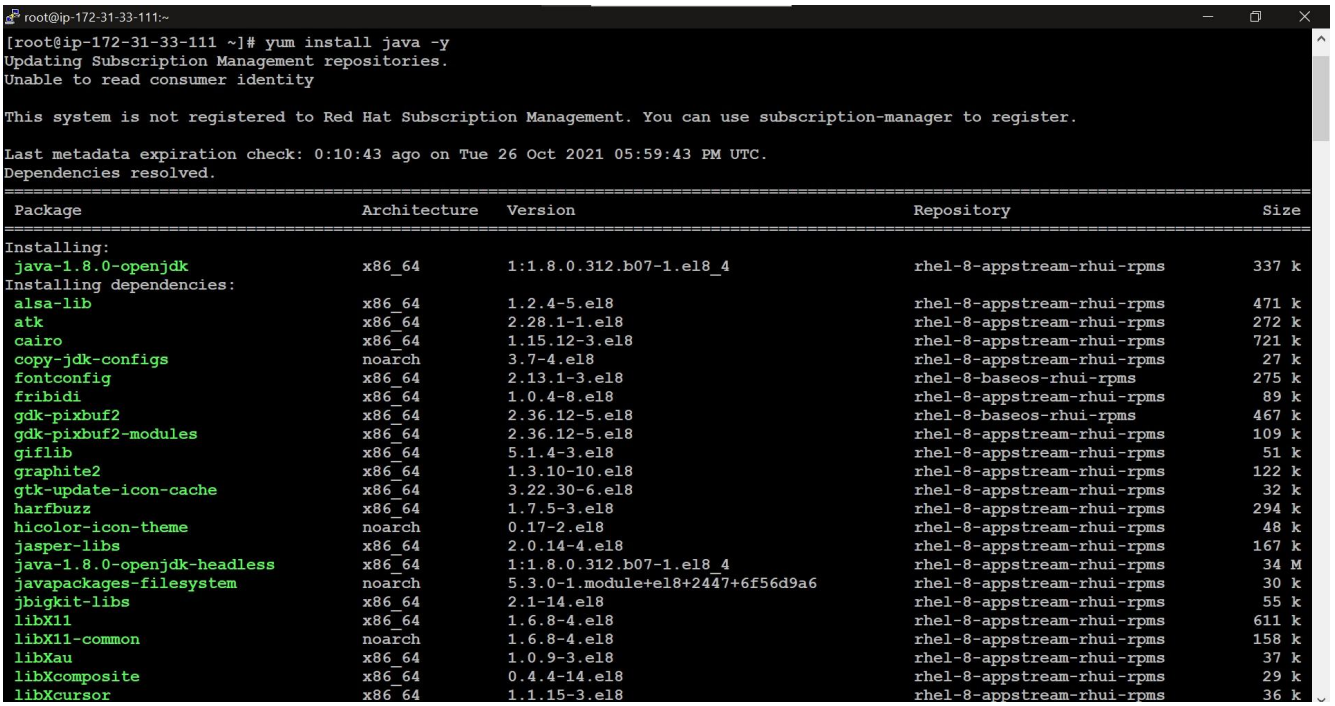
Step 3: Now we also need to set new root password for this we can use the passwd command as shown below. Further Restart the sshd service

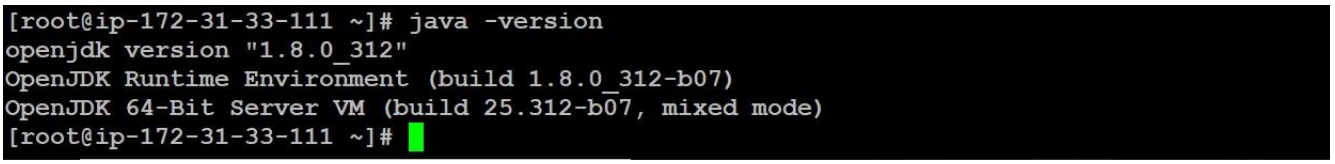




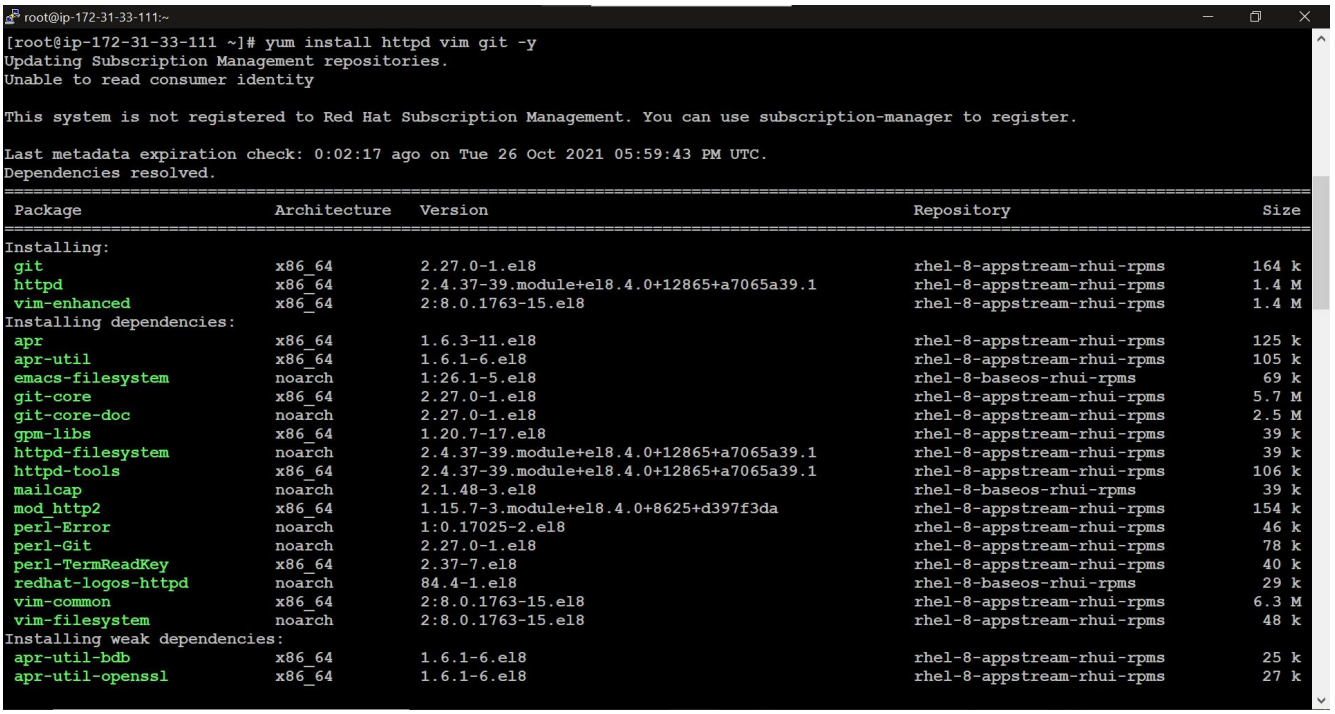
Note: We need to perform step 2 and 3 in both of the ec2-instances

Step 4: Now as these instances are going to be jenkins worker nodes so we need java installed in both the instance.





Step 5: Now lets move toward configuration of the web server. For this we need to install the httpd software. Here I have installed more packages also like git and vim

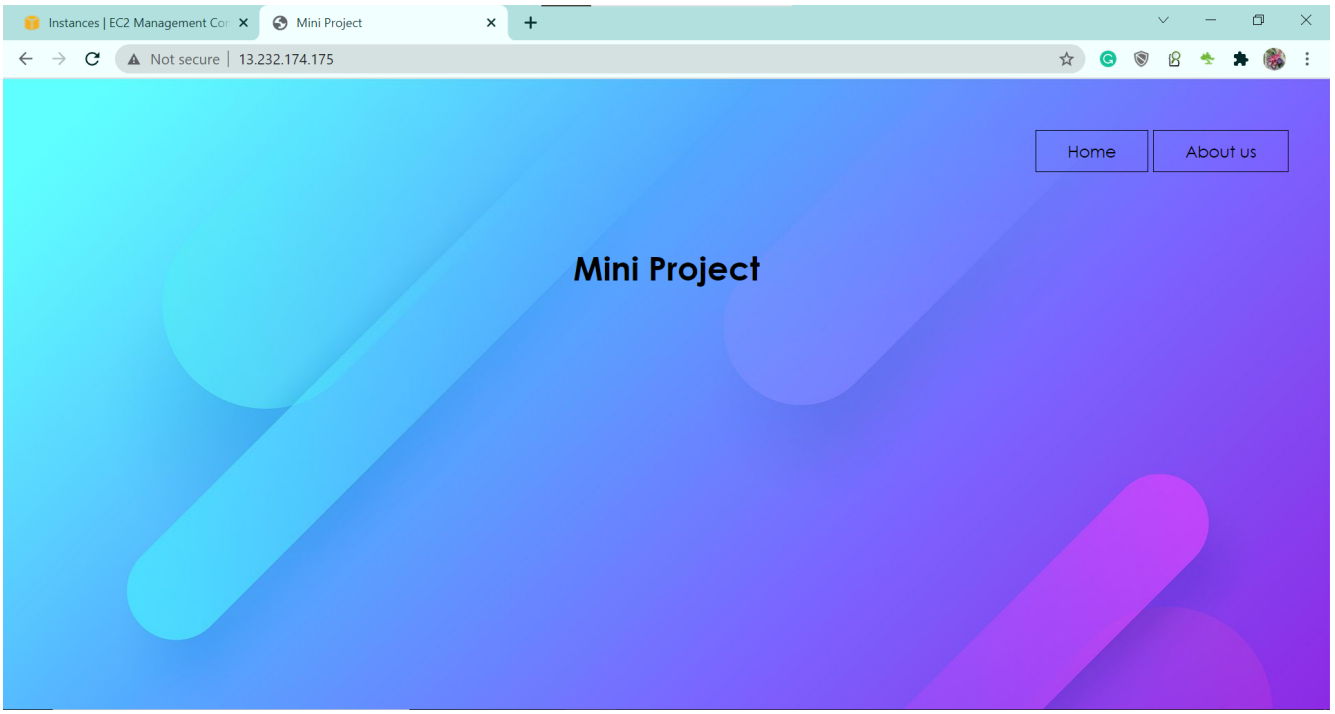


Step 6: Now we want our demo website to be configured for this we used our git repo which we created (https://github.com/Ingole712521/Industrial-Training). Here we have cloned this repo in /var/www/html i.e document root of httpd

Step 7: Further we have started and enabled the services for httpd. Also in httpd we have the logs been collected at /etc/httpd/logs. Root user have the permission to view them.

```
root@ip-172-31-33-111:~
[root@ip-172-31-33-111 ~]# systemctl start httpd
[root@ip-172-31-33-111 ~]# systemctl enable httpd
Created symlink /etc/systemd/system/multi-user.target.wants/httpd.service → /usr/lib/systemd/system/httpd.service.
[root@ip-172-31-33-111 ~]#
```

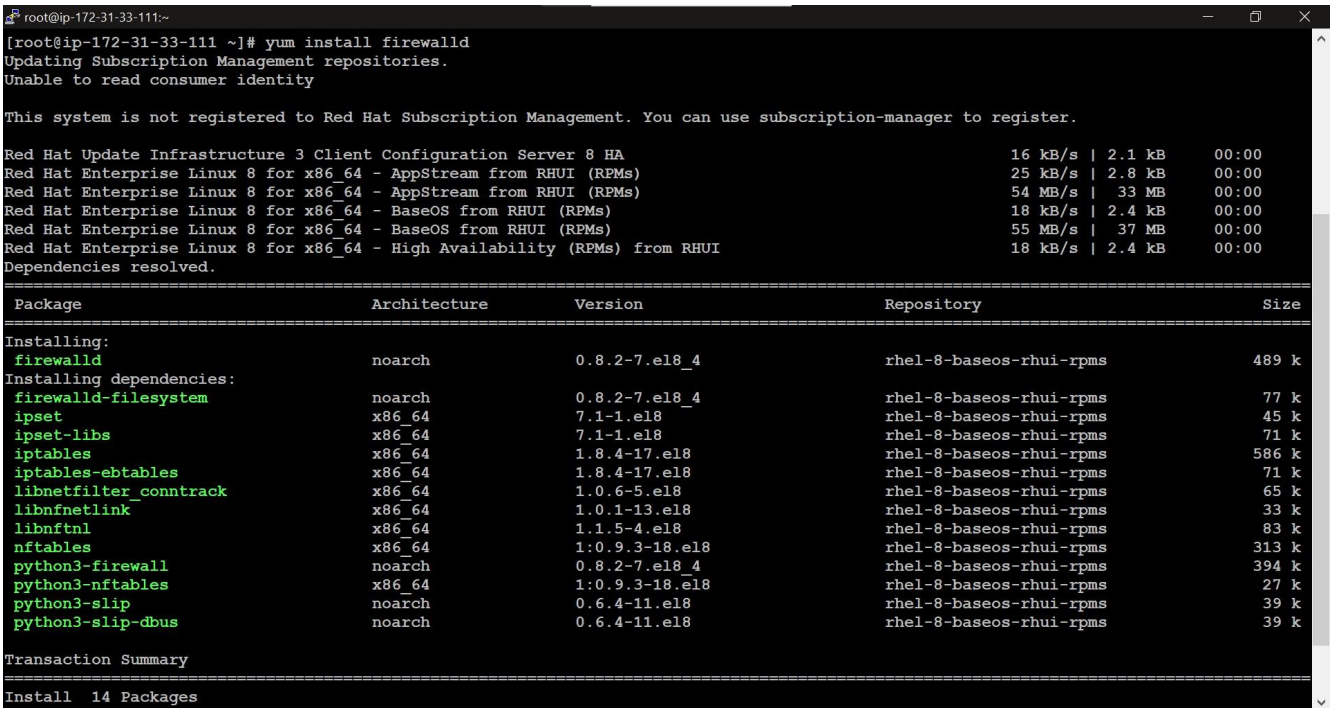
Webpage:

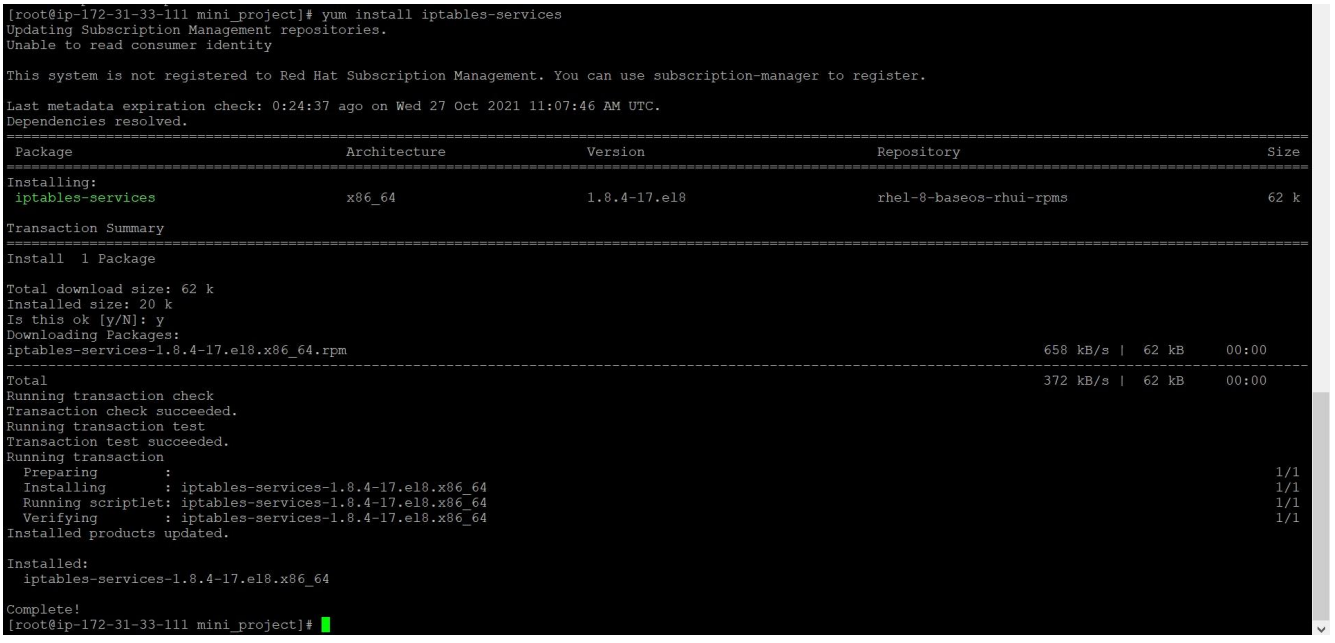


Logs:

```
root@ip-172-31-33-111:/etc/httpd/logs
[root@ip-172-31-33-111 ~]# cd /etc/httpd/logs/
[root@ip-172-31-33-111 logs]# ls
access_log  error_log
[root@ip-172-31-33-111 logs]# cat access_log
152.57.170.125 - - [26/Oct/2021:18:09:15 +0000] "GET / HTTP/1.1" 200 514 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537
.36 (KHTML, like Gecko) Chrome/95.0.4638.54 Safari/537.36"
152.57.170.125 - - [26/Oct/2021:18:09:15 +0000] "GET /style.css HTTP/1.1" 200 612 "http://13.235.103.159/" "Mozilla/5.0 (Windows NT 10.
0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/95.0.4638.54 Safari/537.36"
152.57.170.125 - - [26/Oct/2021:18:09:15 +0000] "GET /iamge.jpg HTTP/1.1" 200 43505 "http://13.235.103.159/" "Mozilla/5.0 (Windows NT 1
0.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/95.0.4638.54 Safari/537.36"
152.57.170.125 - - [26/Oct/2021:18:09:16 +0000] "GET /favicon.ico HTTP/1.1" 404 196 "http://13.235.103.159/" "Mozilla/5.0 (Windows NT 1
0.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/95.0.4638.54 Safari/537.36"
[root@ip-172-31-33-111 logs]# cat access_log | wc -l
4
[root@ip-172-31-33-111 logs]#
```

Step 8: Here we have launch the web-instance in the aws and aws uses its security group as the firewalld. So in the os we need to separately install and start the firewall





Further we also need to all hits over port 80 for this we can use the command :

```
# firewall-cmd --zone=public --add-port=80/tcp
--permanent

# firewall-cmd --reload
```

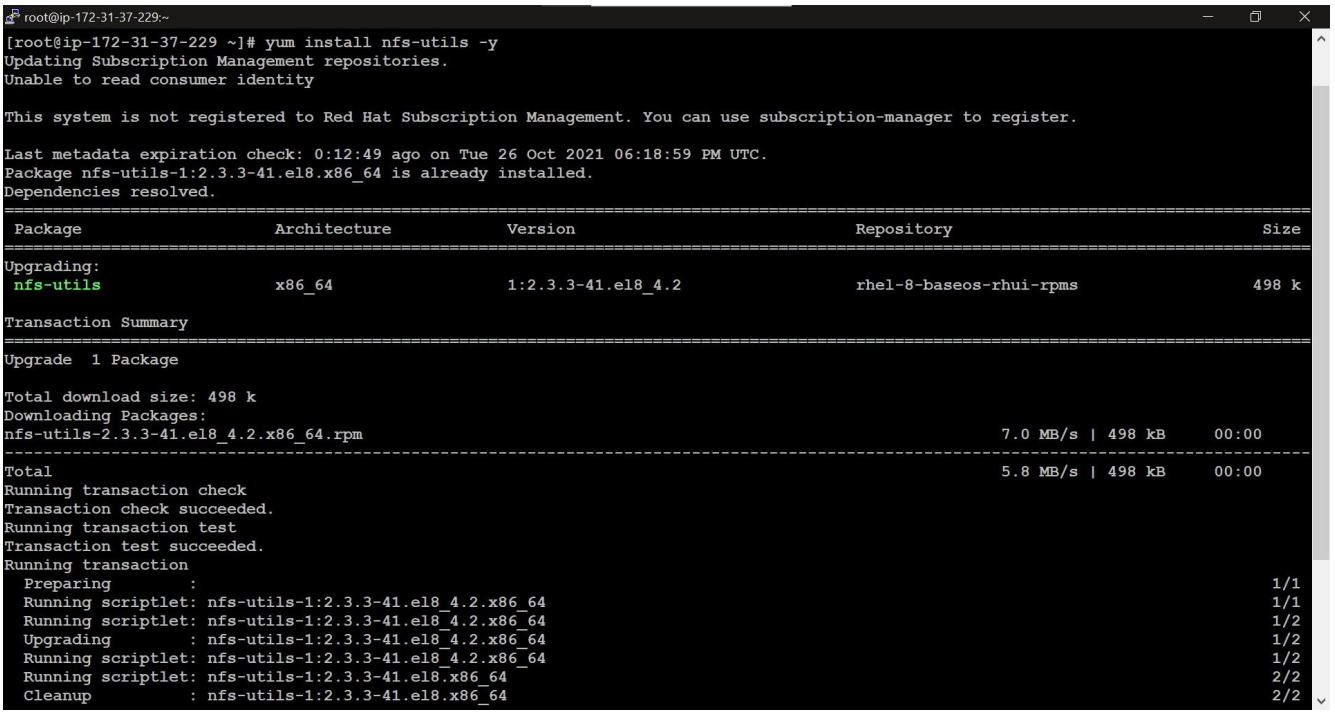
Now lets move toward configuring the Docker and nfs servers in other ec2-instance of t2.large machine-type

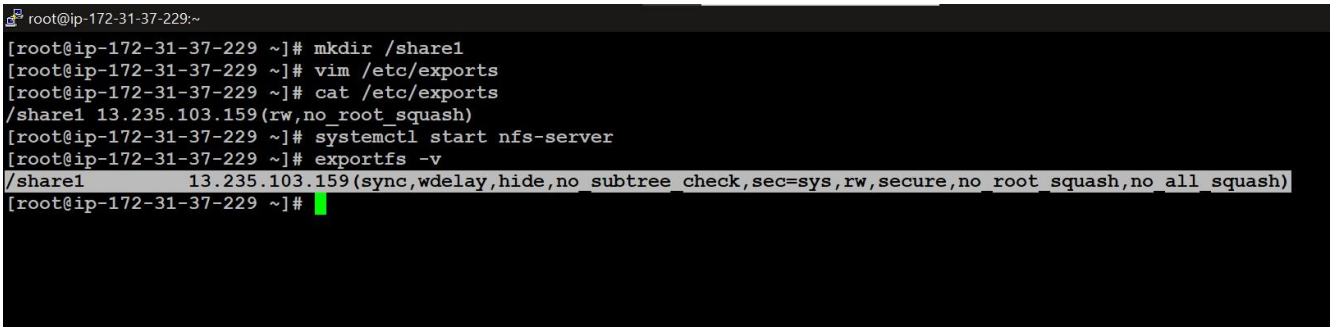
Step 9: For having the nfs setup we need to install nfs-utils package. Now we want to have one folder /share1 where my nfs client can put there data and we can also access it directly simply like accessing a normal directory. For this we need to first create a folder say /share1 now we need to mention in the nfs configuration file (/etc/exports) which folder is the shared one. Format for mentioning this is

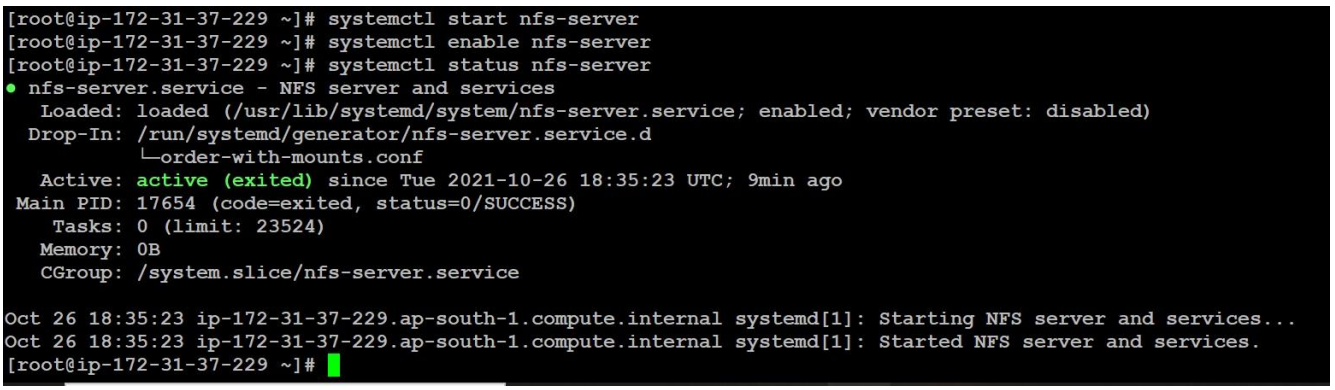
 # <folder name> <client_ip>(permisions)

Further we also need to start the nfs services. We can view our shared folders using the command

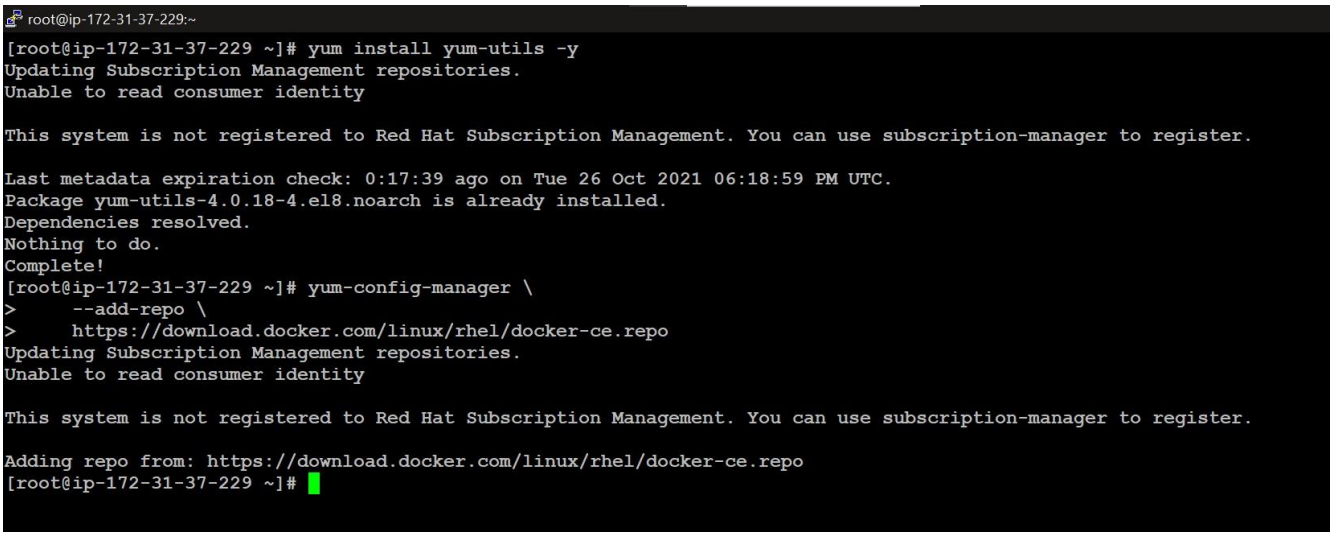
 # exposefs -v

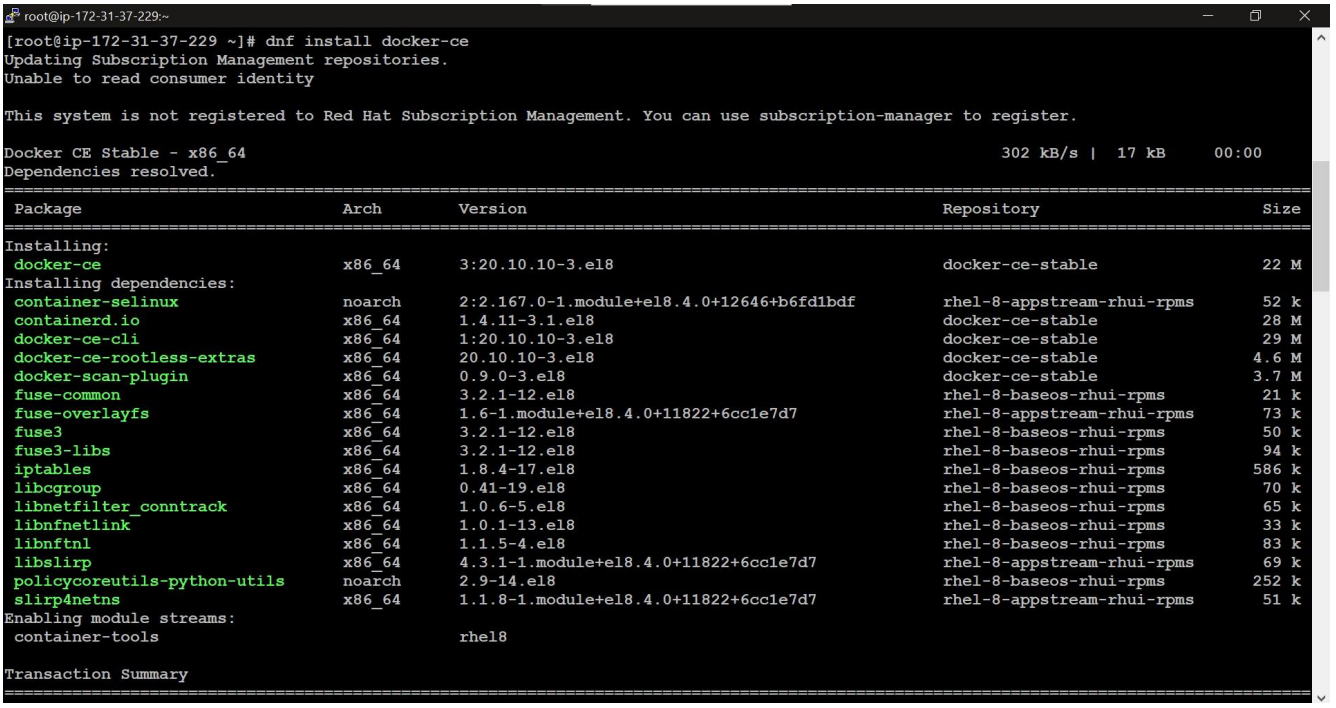






Step 10: Now lets setup the docker in the system. For this we need to first get the repository for the docker added in the system (for this we need yum-utils package too). And then we can install the docker-ce package





Exposing the docker port (2376) so that we can access it from outside world too.

```
root@ip-172-31-37-229:~
[Unit]
Description=Docker Application Container Engine
Documentation=https://docs.docker.com
After=network-online.target firewalld.service containerd.service
Wants=network-online.target
Requires=docker.socket containerd.service

[Service]
Type=notify
# the default is not to use systemd for cgroups because the delegate issues still
# exists and systemd currently does not support the cgroup feature set required
# for containers run by docker
ExecStart=/usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock -H tcp://0.0.0.0:2376
ExecReload=/bin/kill -s HUP $MAINPID
TimeoutSec=0
RestartSec=2
Restart=always

# Note that StartLimit* options were moved from "Service" to "Unit" in systemd 229.
# Both the old, and new location are accepted by systemd 229 and up, so using the old location
# to make them work for either version of systemd.
StartLimitBurst=3

# Note that StartLimitInterval was renamed to StartLimitIntervalSec in systemd 230.
# Both the old, and new name are accepted by systemd 230 and up, so using the old name to make
# this option work for either version of systemd.
StartLimitInterval=60s

# Having non-zero Limit*s causes performance problems due to accounting overhead
# in the kernel. We recommend using cgroups to do container-local accounting.
LimitNOFILE=infinity
LimitNPROC=infinity
LimitCORE=infinity

# Comment TasksMax if your systemd version does not support it.
-- INSERT --
```

Starting the docker services

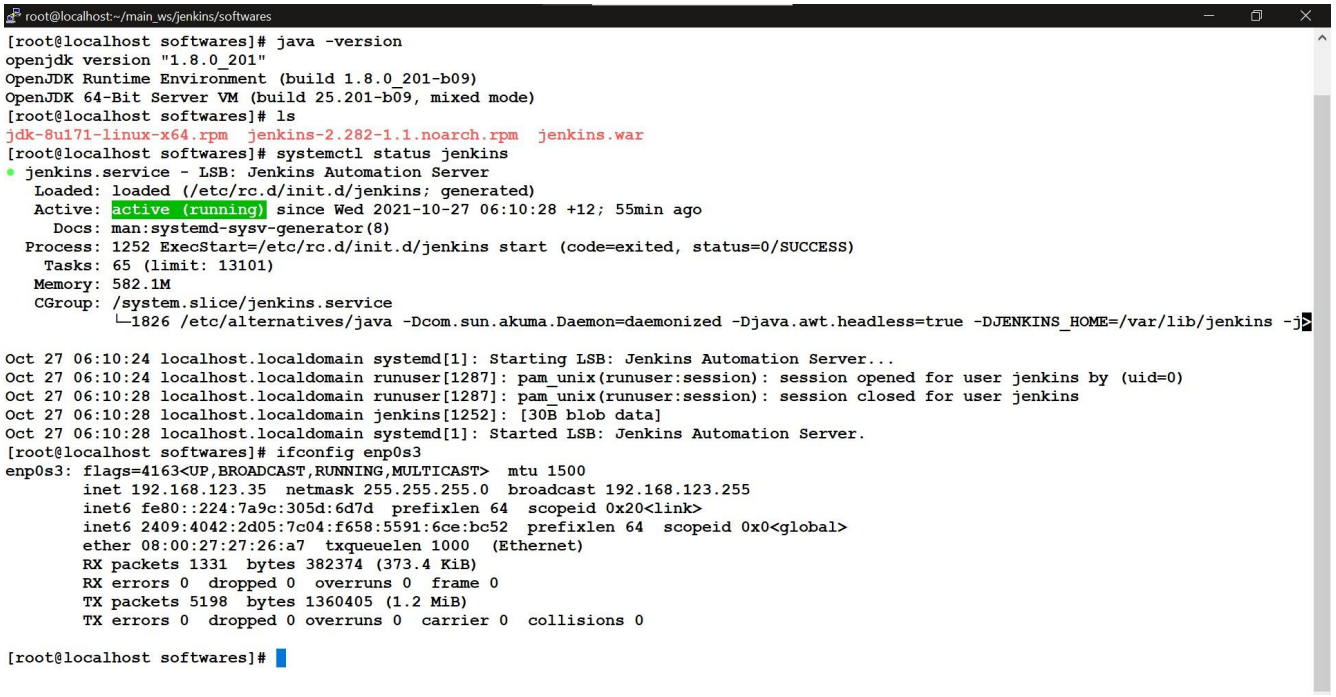
```
root@ip-172-31-37-229:~
[root@ip-172-31-37-229 ~]# vim /usr/lib/systemd/system/docker.service
[root@ip-172-31-37-229 ~]# systemctl start docker
[root@ip-172-31-37-229 ~]# systemctl enable docker
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /usr/lib/systemd/system/docker.service.
[root@ip-172-31-37-229 ~]# systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; vendor preset: disabled)
   Active: active (running) since Tue 2021-10-26 18:43:45 UTC; 13s ago
     Docs: https://docs.docker.com
 Main PID: 20128 (dockerd)
    Tasks: 8
   Memory: 33.6M
   CGroup: /system.slice/docker.service
           └─20128 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock -H tcp://0.0.0.0:2376

Oct 26 18:43:45 ip-172-31-37-229.ap-south-1.compute.internal dockerd[20128]: time="2021-10-26T18:43:45.213907231Z" level=warning msg="
Oct 26 18:43:45 ip-172-31-37-229.ap-south-1.compute.internal dockerd[20128]: time="2021-10-26T18:43:45.213954688Z" level=warning msg="
Oct 26 18:43:45 ip-172-31-37-229.ap-south-1.compute.internal dockerd[20128]: time="2021-10-26T18:43:45.214190241Z" level=info msg="Loa
Oct 26 18:43:45 ip-172-31-37-229.ap-south-1.compute.internal dockerd[20128]: time="2021-10-26T18:43:45.430759226Z" level=info msg="Def
Oct 26 18:43:45 ip-172-31-37-229.ap-south-1.compute.internal dockerd[20128]: time="2021-10-26T18:43:45.513010725Z" level=info msg="Loa
Oct 26 18:43:45 ip-172-31-37-229.ap-south-1.compute.internal dockerd[20128]: time="2021-10-26T18:43:45.542410870Z" level=info msg="Doc
Oct 26 18:43:45 ip-172-31-37-229.ap-south-1.compute.internal dockerd[20128]: time="2021-10-26T18:43:45.542685676Z" level=info msg="Dae
Oct 26 18:43:45 ip-172-31-37-229.ap-south-1.compute.internal systemd[1]: Started Docker Application Container Engine.
Oct 26 18:43:45 ip-172-31-37-229.ap-south-1.compute.internal dockerd[20128]: time="2021-10-26T18:43:45.597130616Z" level=info msg="API
Oct 26 18:43:45 ip-172-31-37-229.ap-south-1.compute.internal dockerd[20128]: time="2021-10-26T18:43:45.607080116Z" level=info msg="API
[root@ip-172-31-37-229 ~]#
```

Step 11: Now lets configure the jenkins here in our case it is running over the locally. Here we need java to run jenkins further for the installation we

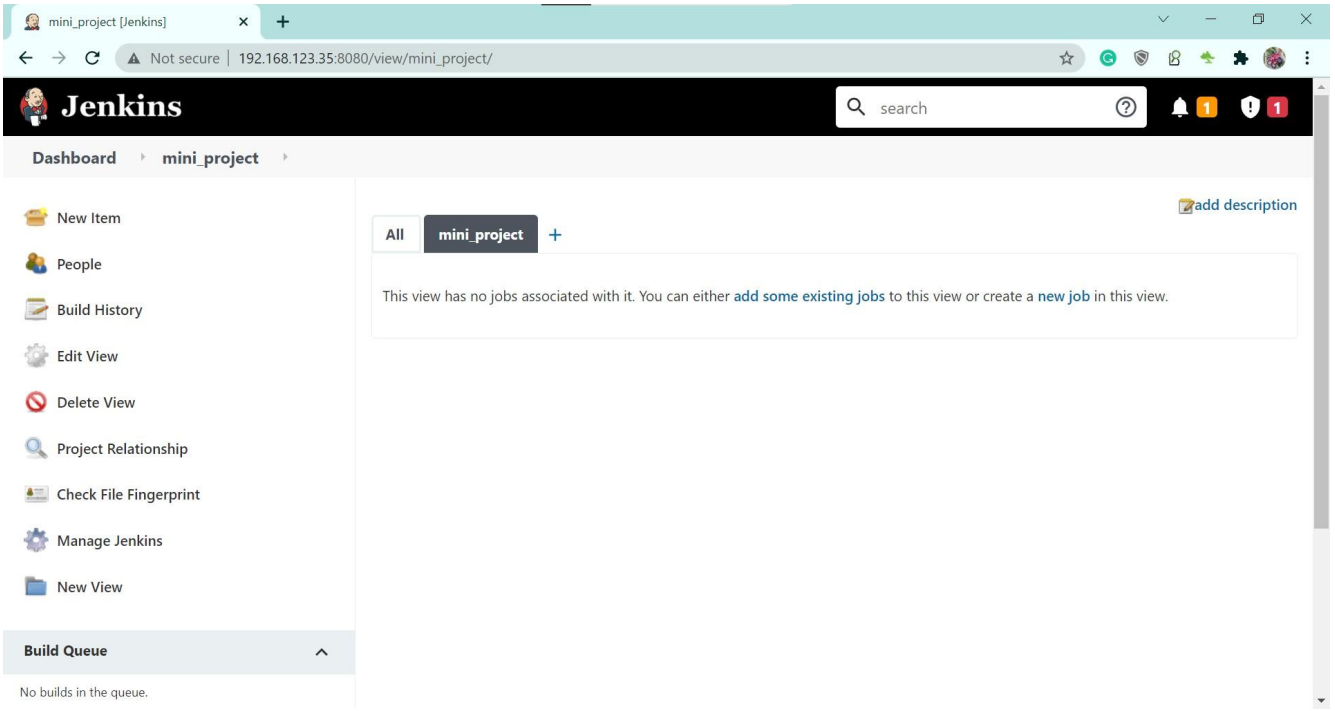
also need the jenkins rpm which we can get over the internet for the installation we can then simply use the command

 # rpm -ivh <package_name>

Further we need to start and enable the services of jenkins

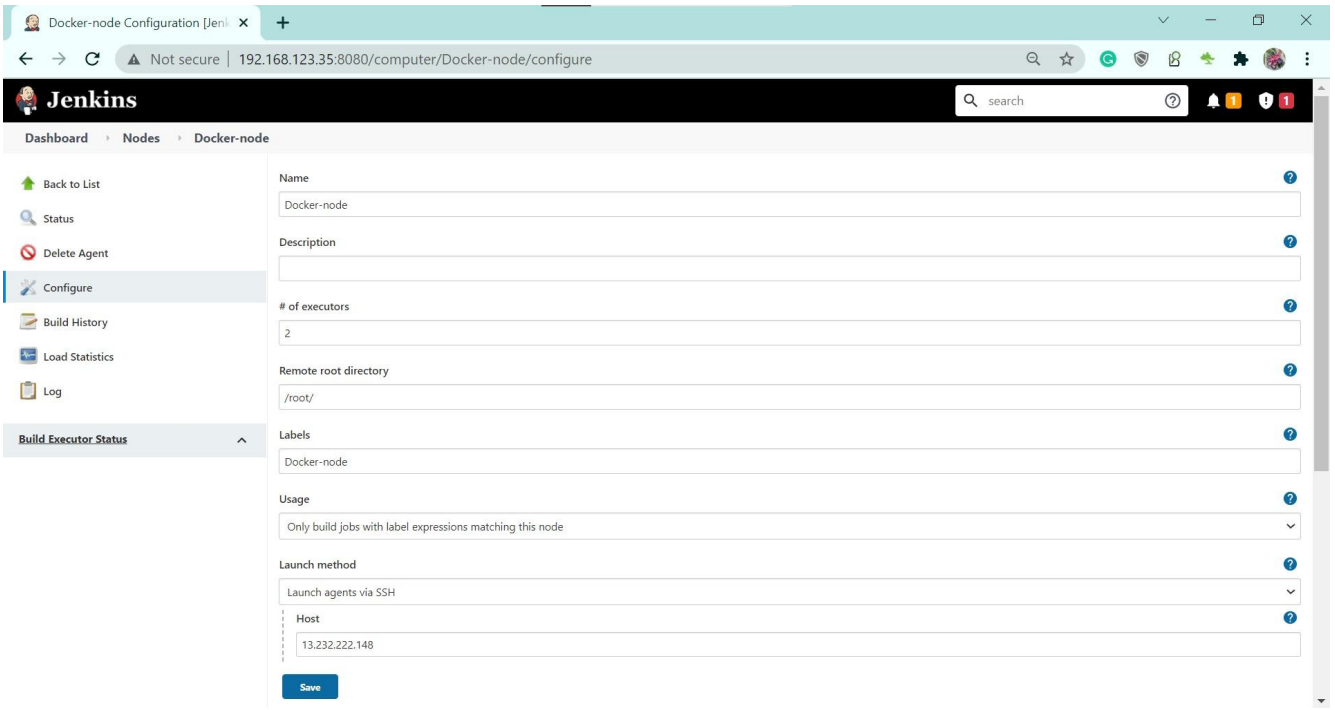


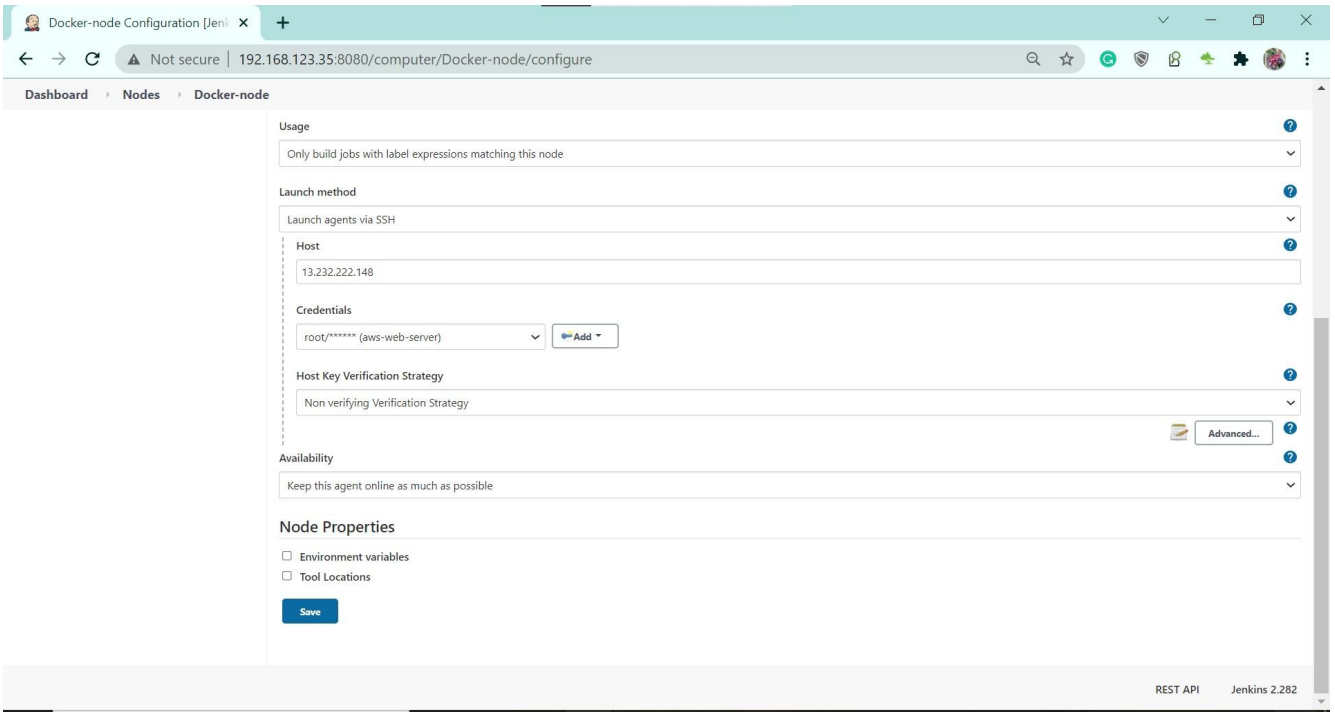
Jenkins by default runs on port number 8080.



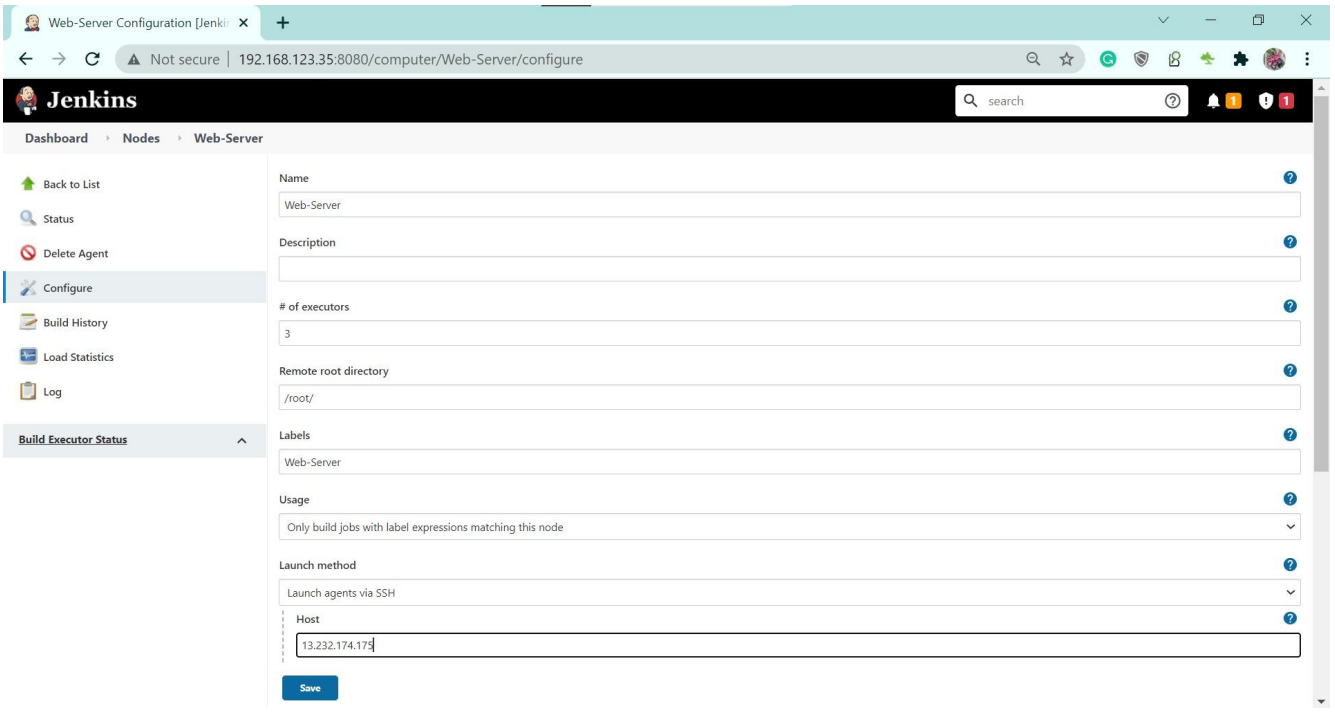
Steps 12: Now lets configure the nodes (worker nodes for the jenkins)

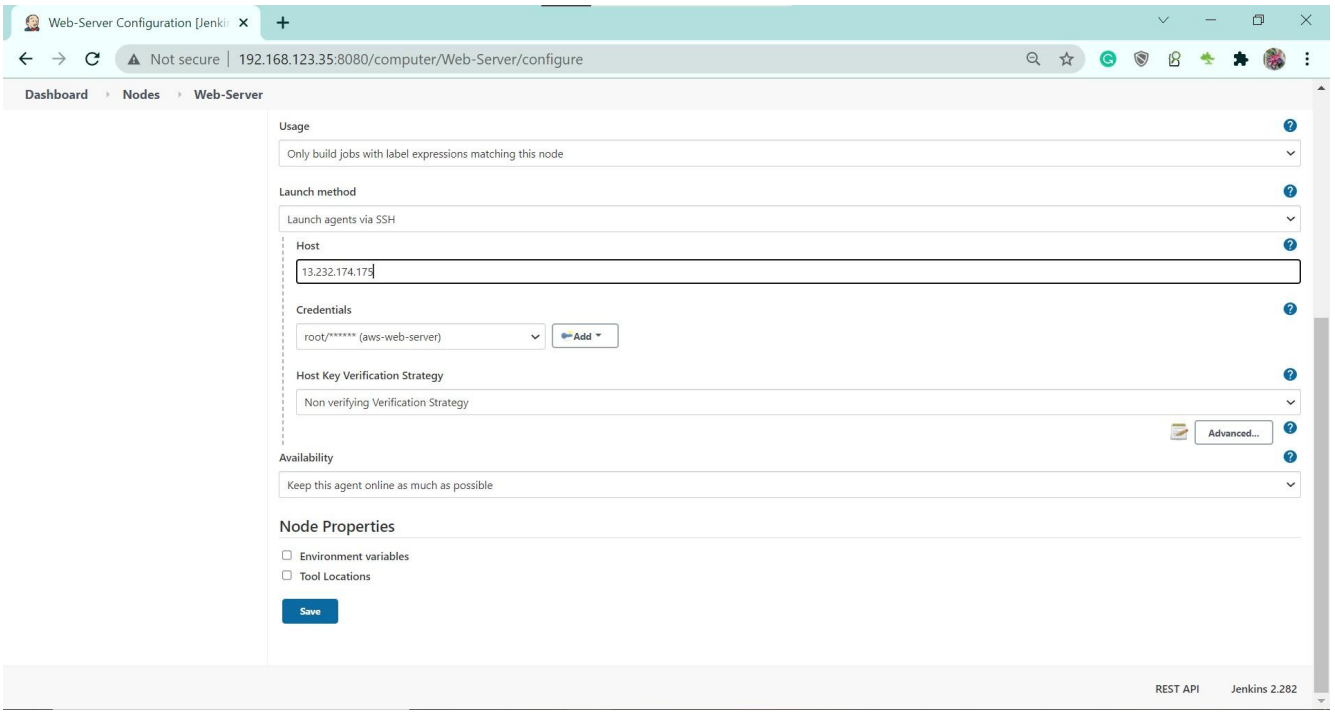
Docker-nfs node:



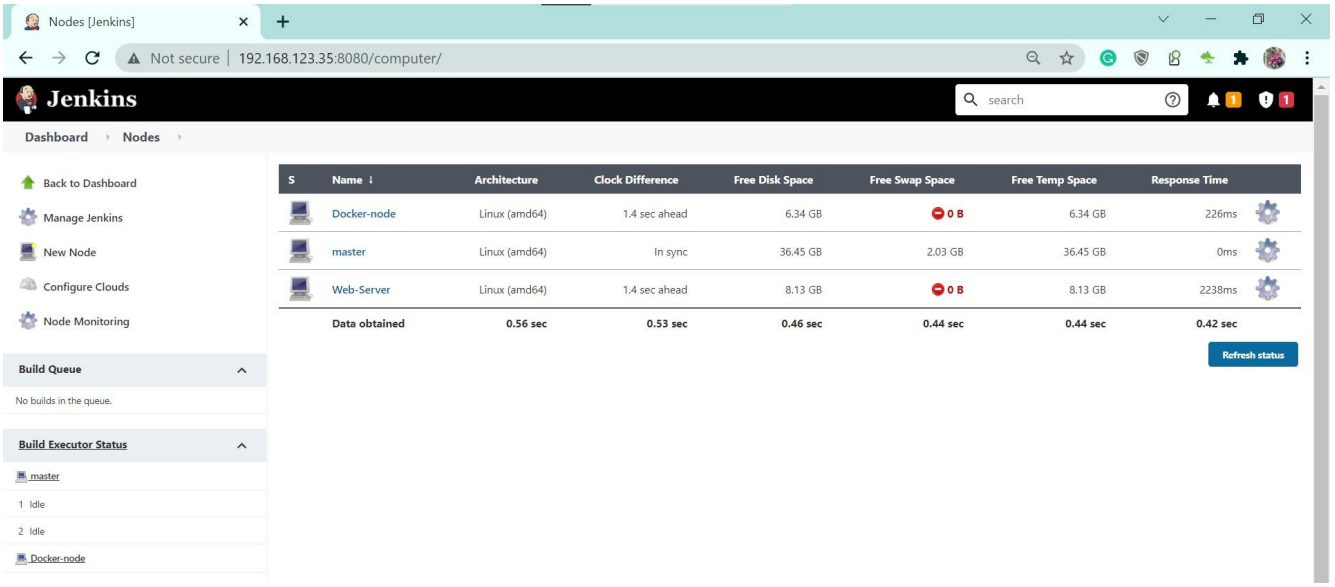


Web-Server Node:





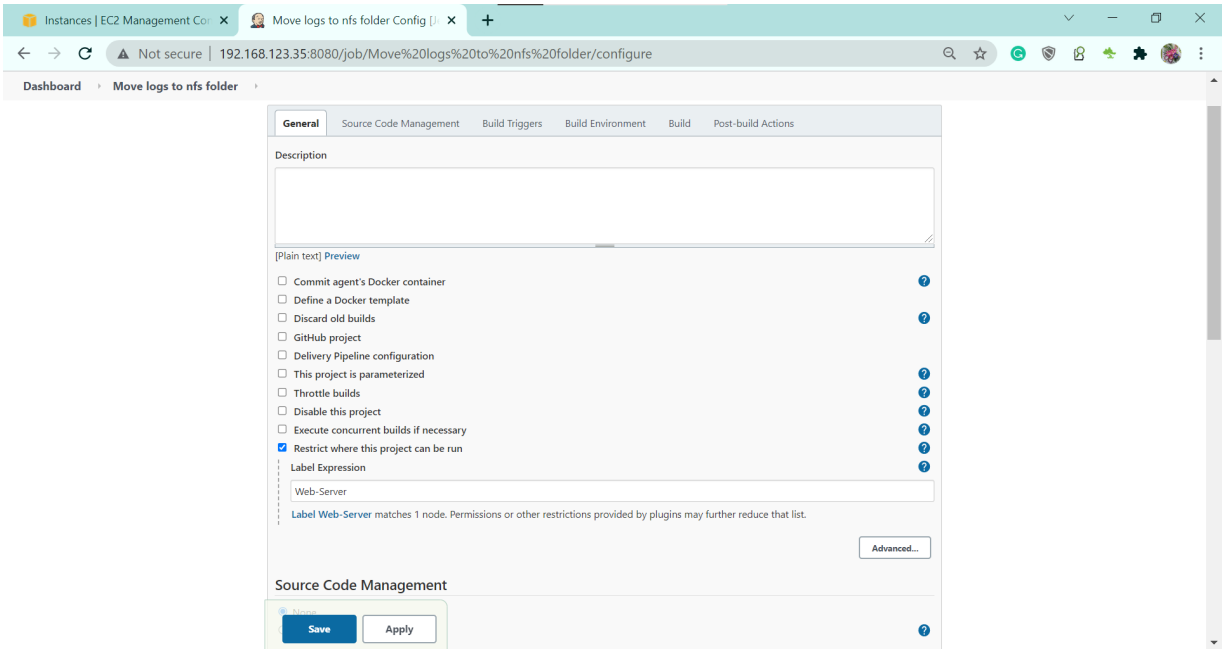
Status of the nodes:

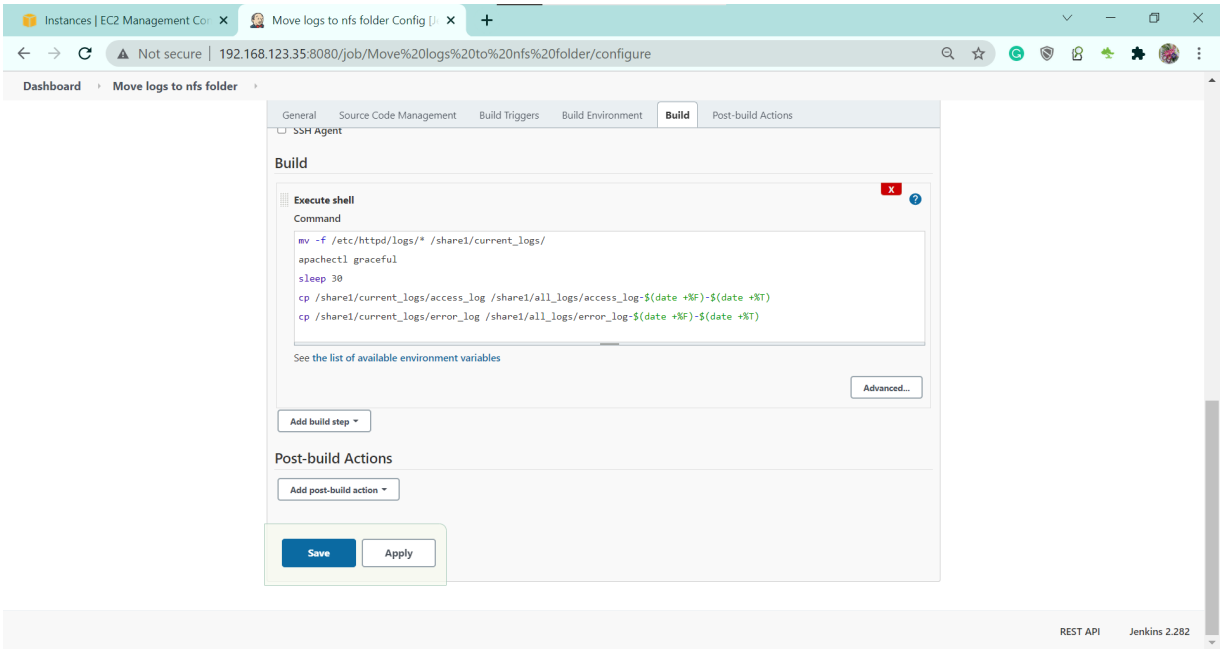


5.3 Implementaion:

Step 1: At the most we will be configuring the jenkins jobs i.e Move logs to nfs folder. (The Job is restricted to be running on web-server node only) Now via this job we are moving the log file from /etc/http/logs to

/share1/current_logs. Further we are also reloading the httpd services. Once this is done we will copy the logs in /share1/current_logs to /share1/all_logs with appropriate time-stamp. So that in future if we need to see past logs we can refer these logs



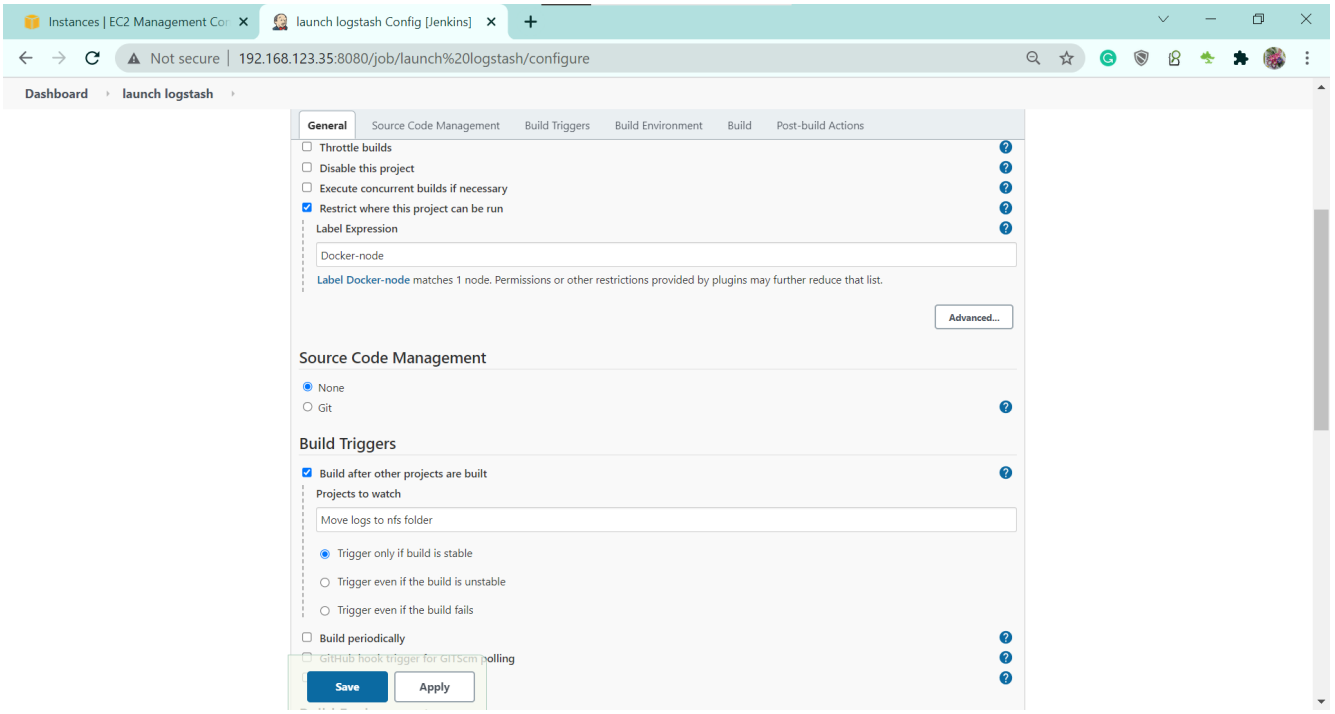


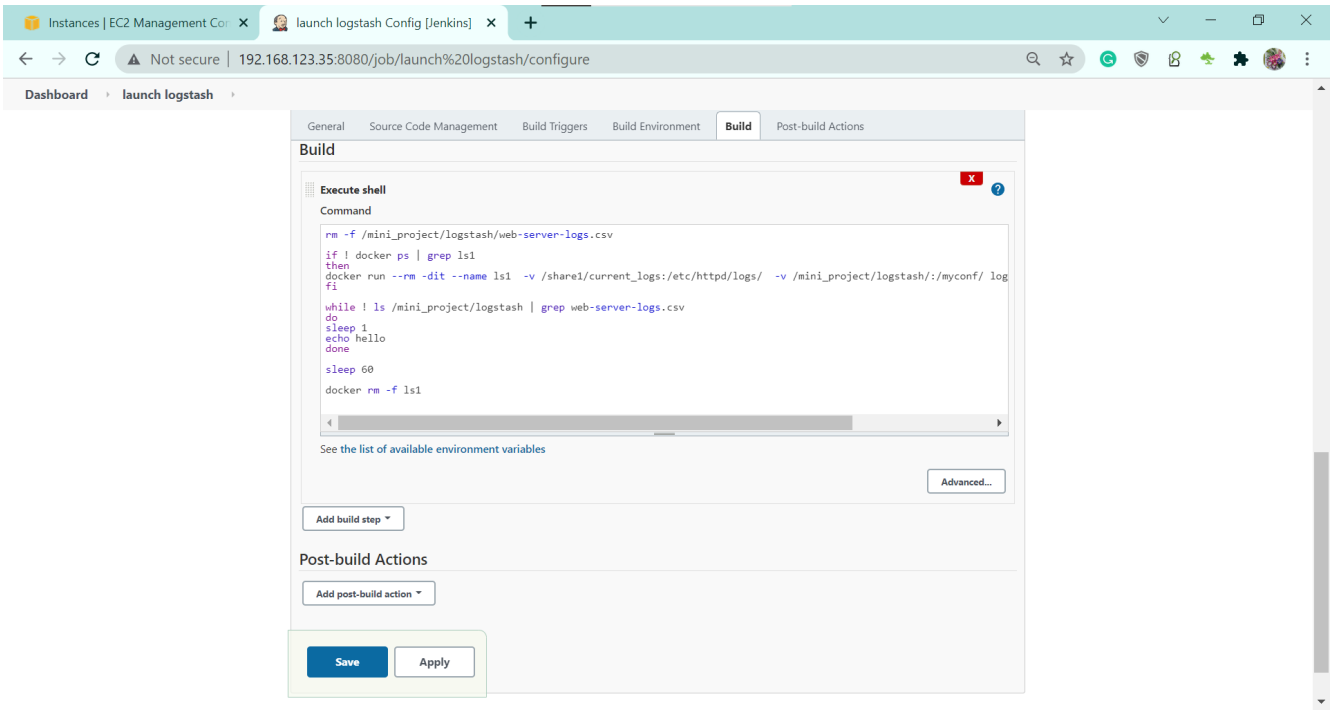
Step 2: Now we will be configuring our second job i.e launch logstash via docker (This job is restricted to be ran on the Docker-nfs-node only). The job will be triggered only if previous job (Move logs to nfs folder) was build successful Via this job we will first check if already formatted old logs are present if yes it will remove those then proceed. Further the job will lauch an logstash container. Now logstash takes a min or two to generate its output i.e (web-server-logs.csv) file. For this we will wait till its there once the file is generated we don't need the container so we will remove the logstash container

(Note: make shure that the volume mounted /mini_project/logstash have read write and execute powers for all or docker user if need we can give those via

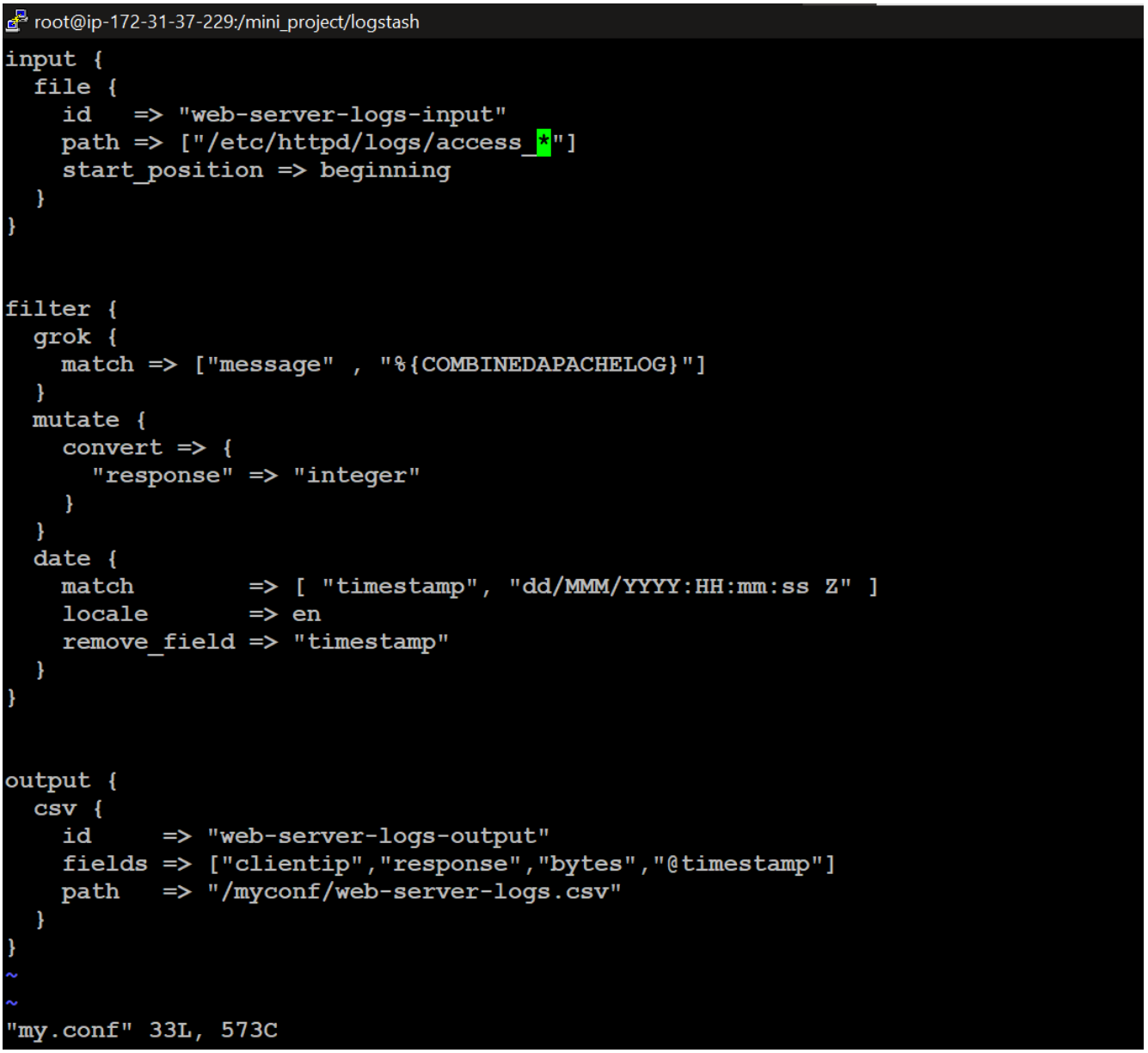
```
# chmod ugo+rwx /mini_project/logstash
```

)





Here in the logstash we are having our custom requirements i.e we want logstash to take input from a particular file we want logstash to give output in particular destination. Here we need to use custom configuration file. (hence we have used -f option to pass that custom conf file)



In the logstash for almost any kind of requirement we use the plugins here we had the requirement to filer apache logs. We used grok plugin for this grok internally uses the regular expression to find and filter required data from raw data. Grok have many other keyword too eg. if we need to find IP in the file we can use %IP:client.

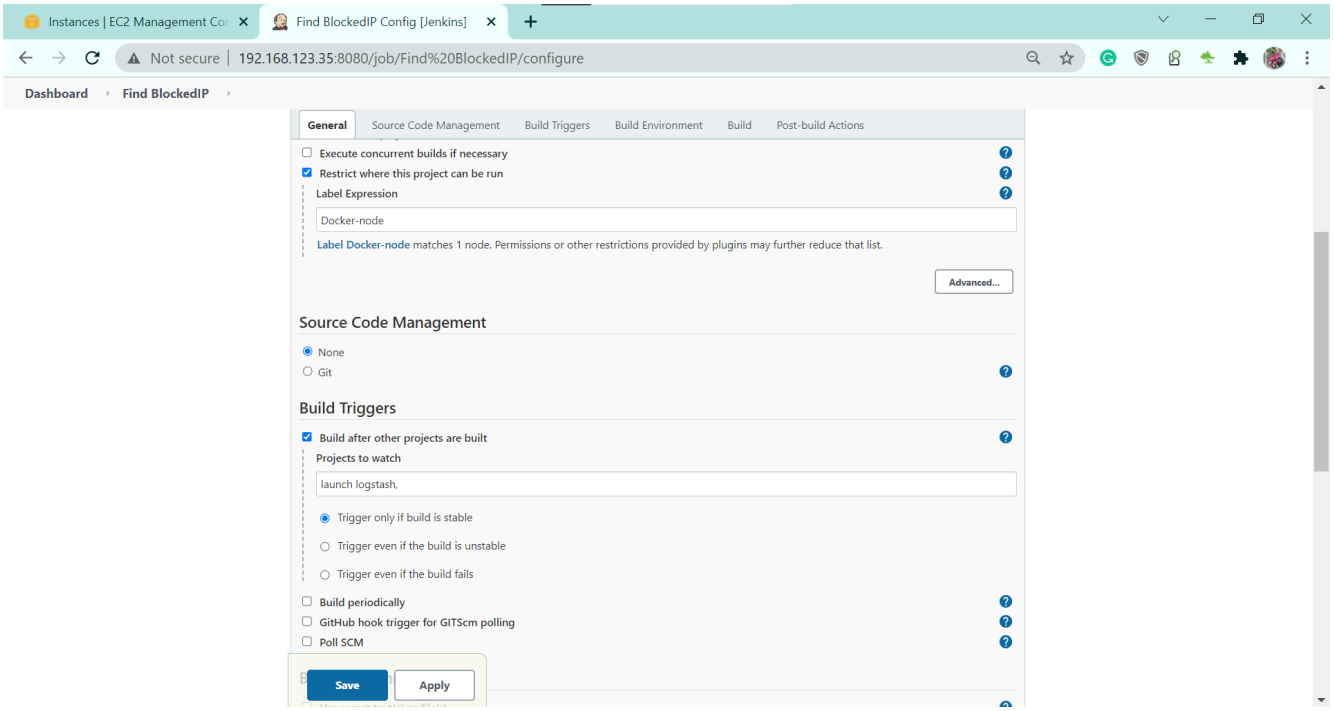
Further in similar way to retrieve the status code and timestamp we used the mutete and data plugin. Also via the input and output we have mention the input destination and the output destination with required format

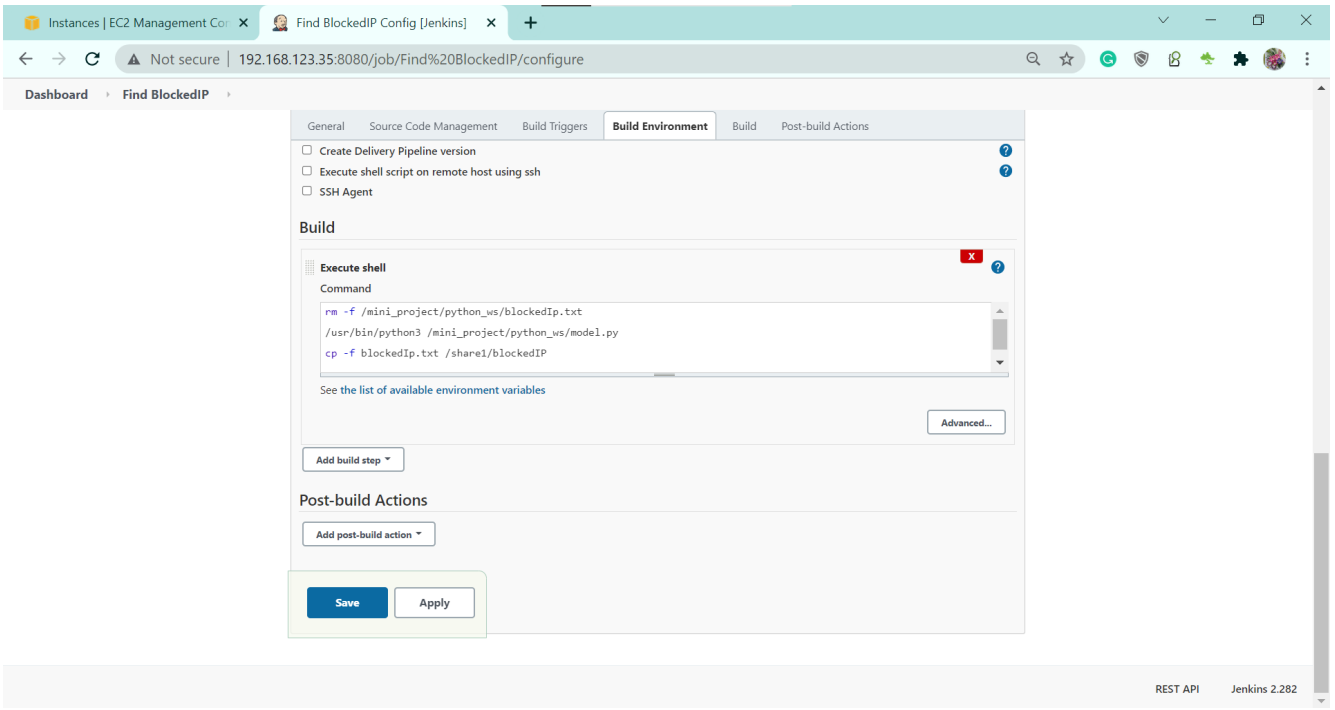
Launching logstash via docker:

```
# docker run --rm -dit --name ls1
-v /share1/current_logs:/etc/httpd/logs/
-v /mini_project/logstash/:/myconf/
logstash:7.7.1 -f /myconf/my.conf
```

With the use of -f option we are telling location of custom conf file. Also the output file generated will be in /myconf/ but we have mounted it to /mini_project/logstash so at the end we can see that in this folder too.

Step 3: Now here we will be configuring our next job i.e Find BlockedIP. Via this job we will be getting the suspicious IP. (This job is restricted to be ran in Docker-nfs-node only). This job will be triggered once the previous job i.e launch logstash will be build successfully. In this job once the blockedIP.txt file is created it is moved to /share1/BlockedIP folder so that my web server can use this to block those IP





Model.py

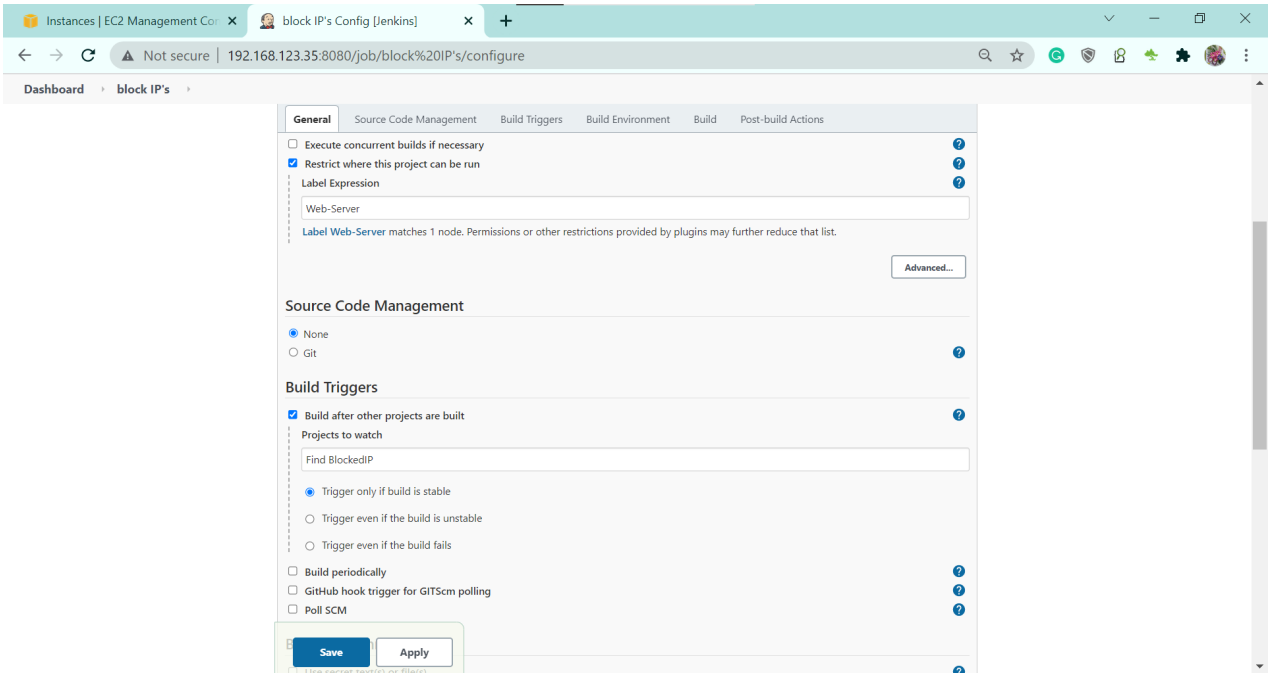
```
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

ds =
pd.read_csv('/mini_project/logstash/web-server-logs.c
sv',header=None)
ds.columns =
['IP','Status_code','Bytes_trf','TimeStamp']
ds = ds[['IP','Status_code','TimeStamp']]
ds.info()

ds = ds[['IP','Status_code']]
ds = ds.groupby(['IP',
'Status_code']).Status_code.agg('count').to_frame('Co
unt').reset_index()
ds = ds.sort_values(['Count'],ascending=False)
ds.head(20)
```

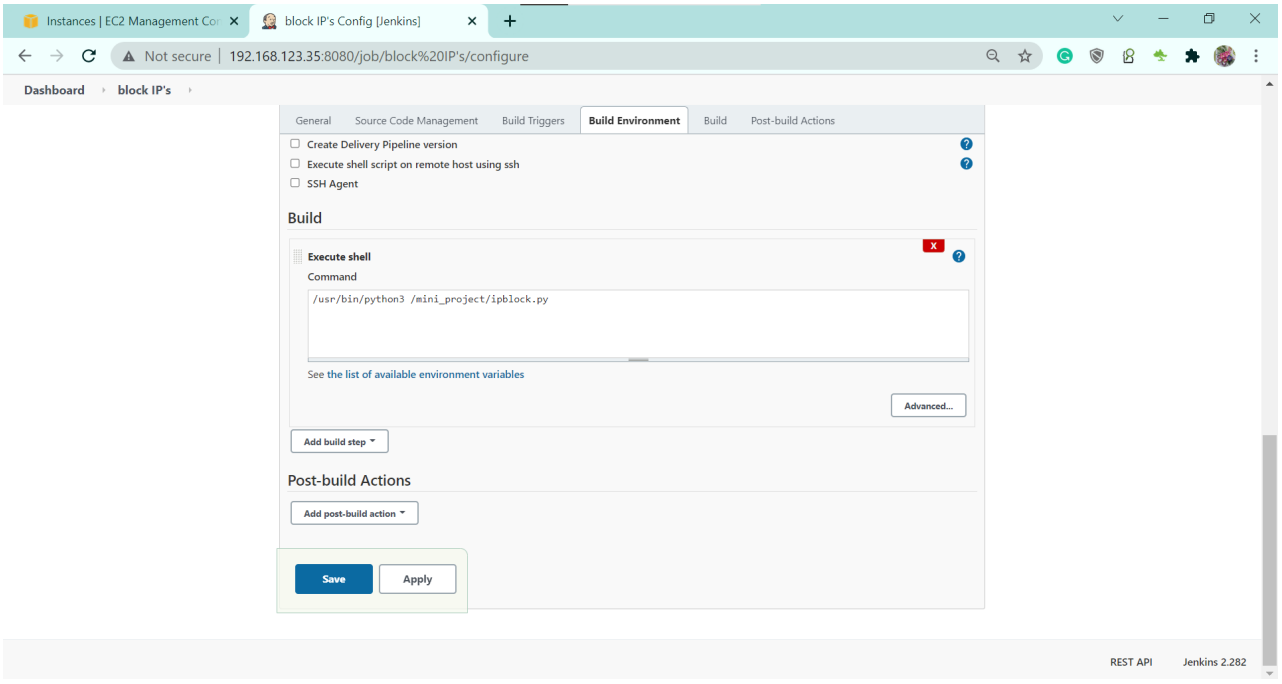
```
training_data = ds.drop(['IP'],axis=1)
training_data.head(20)

sc = StandardScaler()
data_scaled = sc.fit_transform(training_data)

model=KMeans(n_clusters=2)
pred=model.fit_predict(data_scaled)

final_dataset = ds
final_dataset["Cluster"] = pred
final_dataset.head()

a = []
for index,row in ds.iterrows():
    if row['Count'] > 500:
        a.append(row['Cluster'])
blockedCluster = max(set(a), key = a.count)

blockedIP = []
for index,row in ds.iterrows():
    if row['Cluster'] == blockedCluster:
        blockedIP.append(row['IP'])

for b_ip in blockedIP:
    print("Blocked IP: {}".format(b_ip))

blockedIP=set(blockedIP)

with open('blockedIp.txt', 'w') as filehandle:
            for listitem in blockedIP:
             filehandle.write('%s\n' % listitem)
```

Now in this code we have use the k-mean clustering algorithm after multiple test we have found that depending on the number of hits its good to have at 2 clusters as static for now.

Once the model is trained we can use it and using it we have created the blockedIP.txt file compromising of the BlockedIP

Step 4: Now we come to our last job (this job is restricted to be ran in web-server node only). Via this job we are only asking the internal firewall running in the webserver to block the suspicious IP.



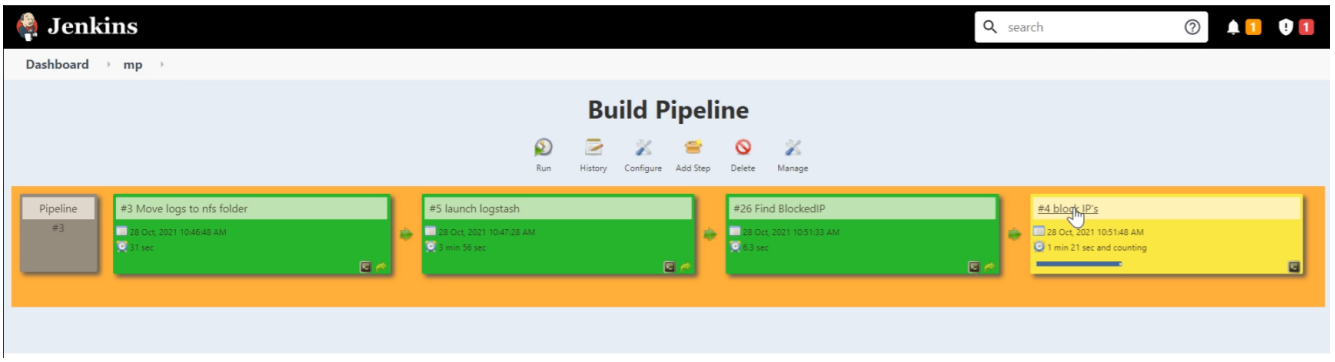


Ipblock.py

```
from os import system
with open("/share1/blockedIP/blockedIp.txt","r") as
file1:
  for i in file1:
    system('iptables -A INPUT -s {} -j
DROP'.format(i.rstrip('\n')))
    system('service iptables save')
```

Here after this packets receiving from these suspicious IP will be dropped and not hit the server.

Step 5: Now lets create the view in jenkins for the build pipeline



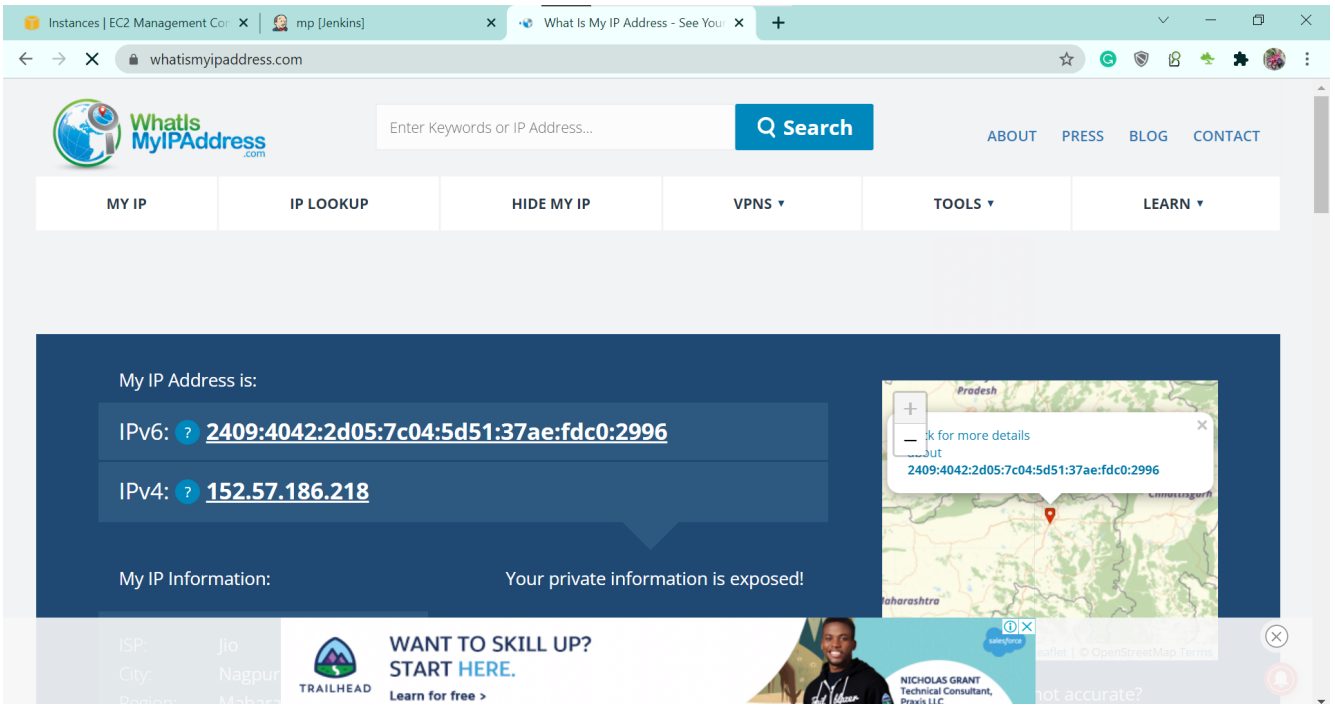
This is created because we have set in the proper way for each job which one is there upstream job and which one is there down stream job

Step 6: Now from my windows machine with the help of a bash script we will be hitting the web server with multiple fake/ dummy request and try to increase the load over the server. Before or during this process many genuin clients might have also hit the webserver

```
#!/bin/bash
while true
do
```

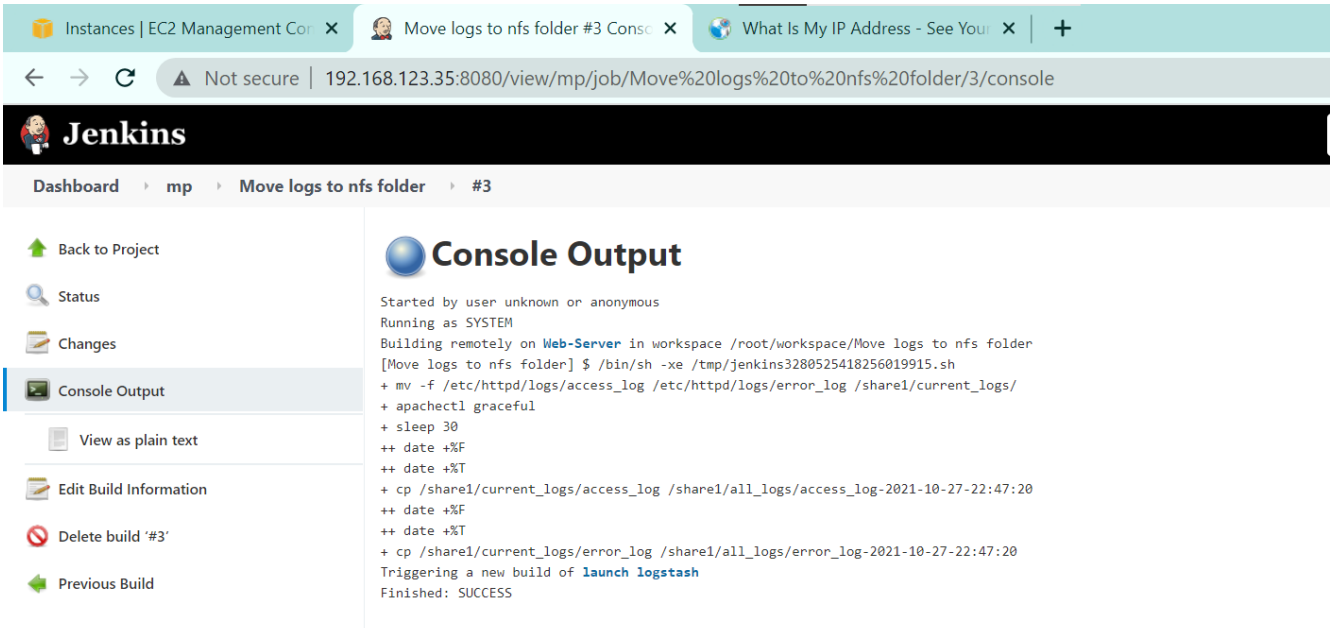
```
        curl http://13.232.174.175/gaurav.html
        curl http://13.232.174.175/gaurav2001.html
        curl http://13.232.174.175/jack.html
        curl http://13.232.174.175/nehal.html
        curl http://13.232.174.175/asdasfadfsf.html
        curl http://13.232.174.175/asdadw.html
done
```

IP address of attacker:

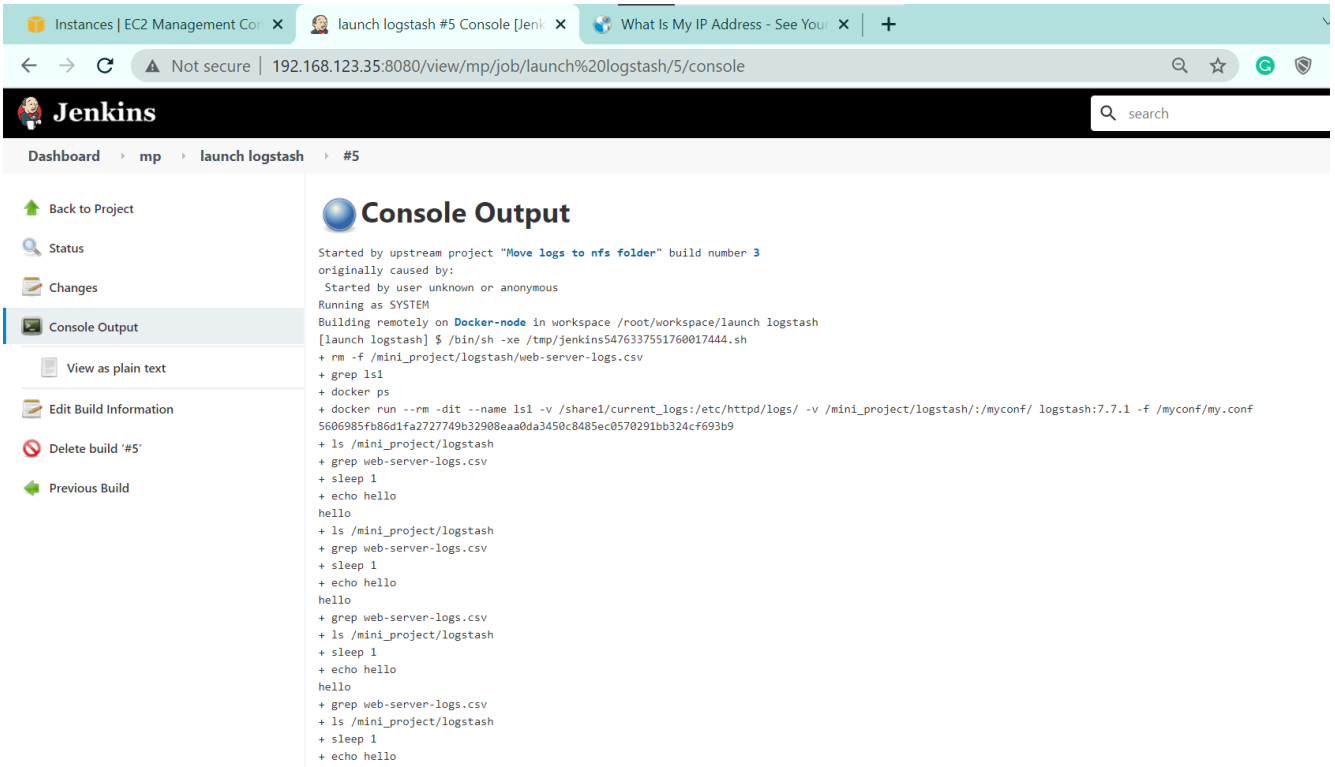


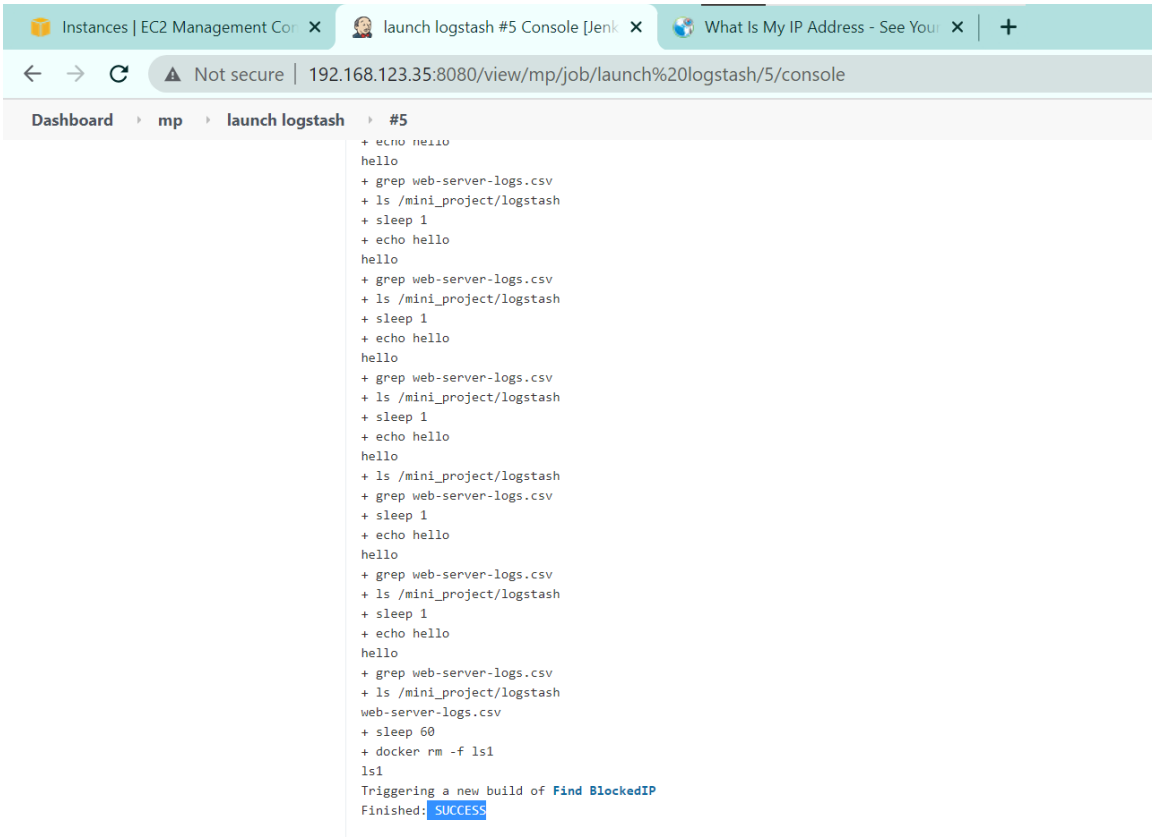
Now Lets trigger the pipeline and see the console outputs of the job

Move logs to nfs folder:

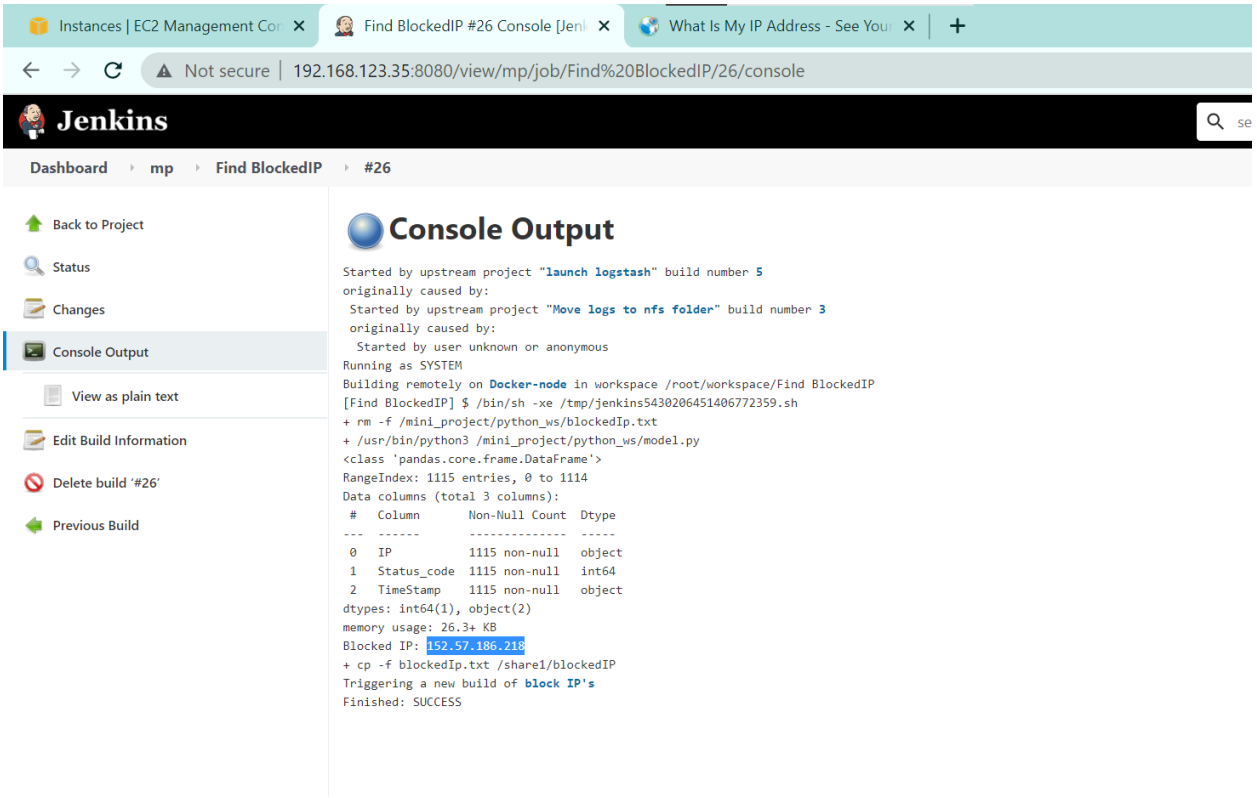


launch logstash:



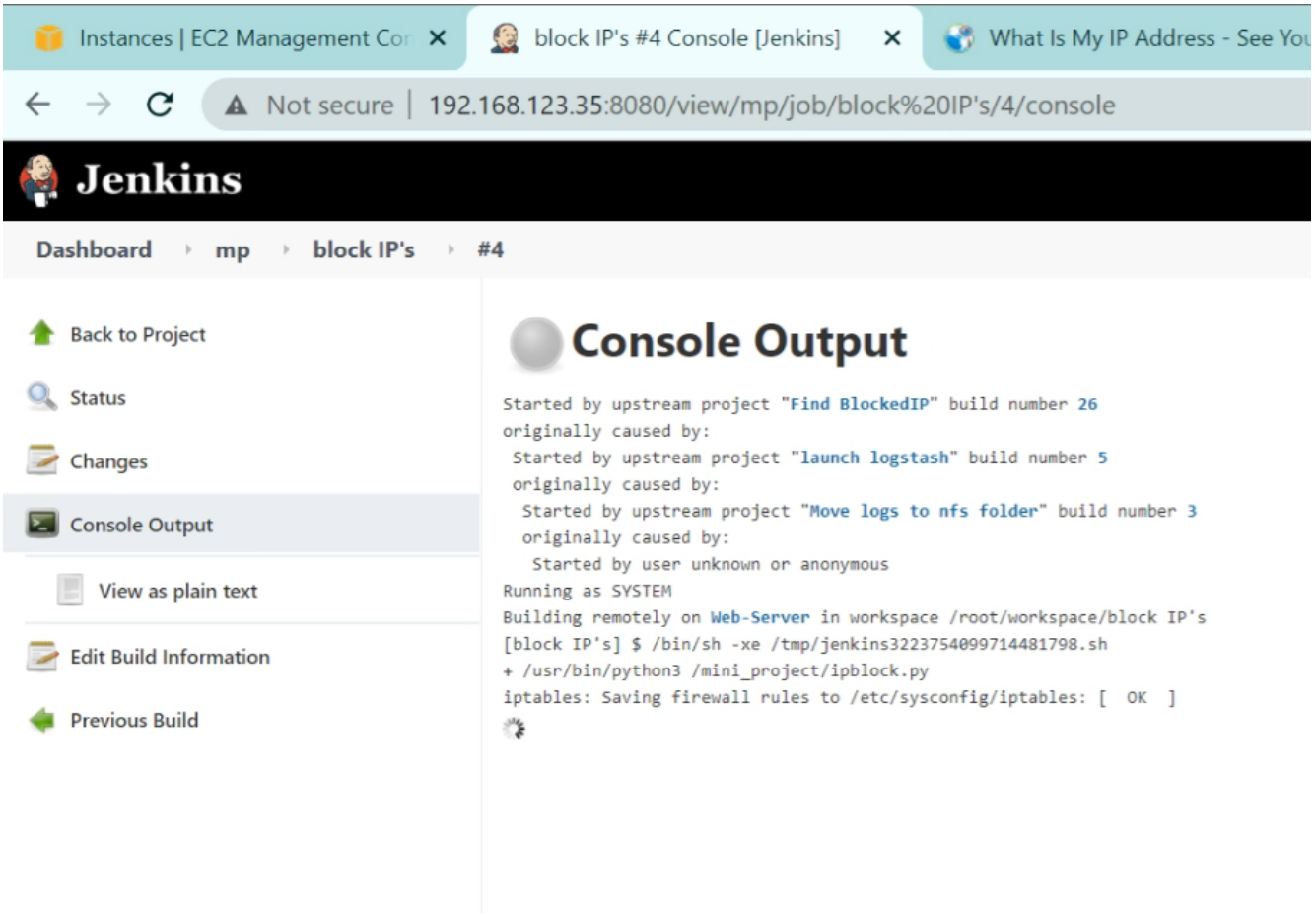


Find BlockedIP:

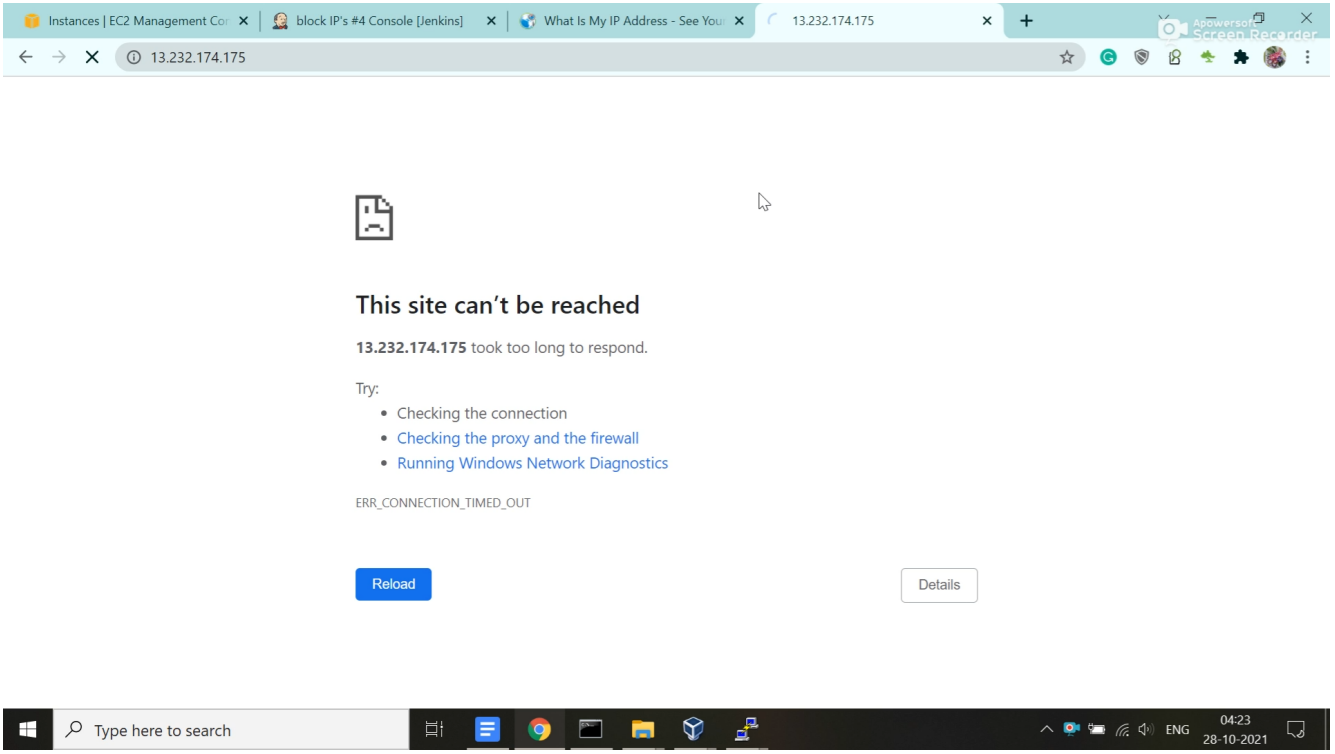


From here we can see it have detected the windows machine IP as suspicious IP.

block IP's:

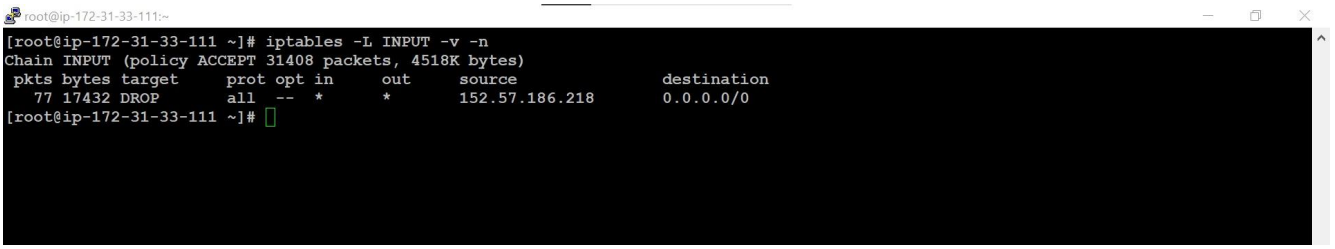


Now as the firewall rules are saved so lets check the connectivity via the windows machine



Now windows can't connect

We can also check the rules via going inside the web-server



Now at this same time we have also tried to connect to the webserver by different device and its works absolutely fine

Chapter 6

Result and Declaration

6.1 Result:

1. Our Environment was configured Successfully (Jenkins-master node, webserver, docker-nfs node)
2. The Machine learning model working internally gave the correct output i.e suspicious IP
3. The Whole pipeline worked successfully and at the end the suspicious IP was not able to connect to the web server while other genuin client can connect

6.2 Discussion:

1. Inspite the project aim was achieved but the project have many scopes of future improvement resulting in reduction of the manual efforts of the datascients and the security team
2. Manual work is reduced as the pipeline have the facility to be triggered by unique url so our monitoring tools can do this job as an when the load increase the first job is triggered and the series start

Chapter 7

Conclusion and Future Scope

7.1 Conclusion

In this project we have successfully implemented the k-mean clustering algorithm for the suspicious IP detection. We did the data processing using the logstash for which we setup the docker lauch logstash over it created our own custom configuration file. We used the jenkins to automate the stuff on a single click the all the jobs can be run in such a way the pipeline were designed. Finally the implementation also gave a good response

7.2 Future Scope

1. The approach used for creating machine learning model for the DOS attack using similar approach model can be developed for other patter based attacks too.
2. Internally for creation of the model we have use the K-mean clustering algorithm (Discussed further in detail). And in this algorithms we need to decide the hyper parameter i.e number of cluster to be formed this is decided by analysis over the data (elbow method). Now we can provide this report of analysis frequently to our datascientest team so that if any changes need in code in future it can be done. This will reduce task of datascientest as well as there will be continous improvement in the project

Refrences

1. K-Means Clustering Approach to Analyze NSL-KDD Intrusion Detection Dataset
2. https://hub.docker.com/_/logstash
3. https://www.elastic.co/guide/en/logstash/current/input-plugins.html
4. https://www.elastic.co/guide/en/logstash/current/filter-plugins.html
5. https://www.elastic.co/guide/en/logstash/current/output-plugins.html
6. https://us-cert.cisa.gov/ncas/tips/ST04-015