Department of Computing

Academic Year 2022/2023

Infrastructure to Big Data (CA3)

Implementation of Apache Spark Cluster on Hadoop using Vagrant

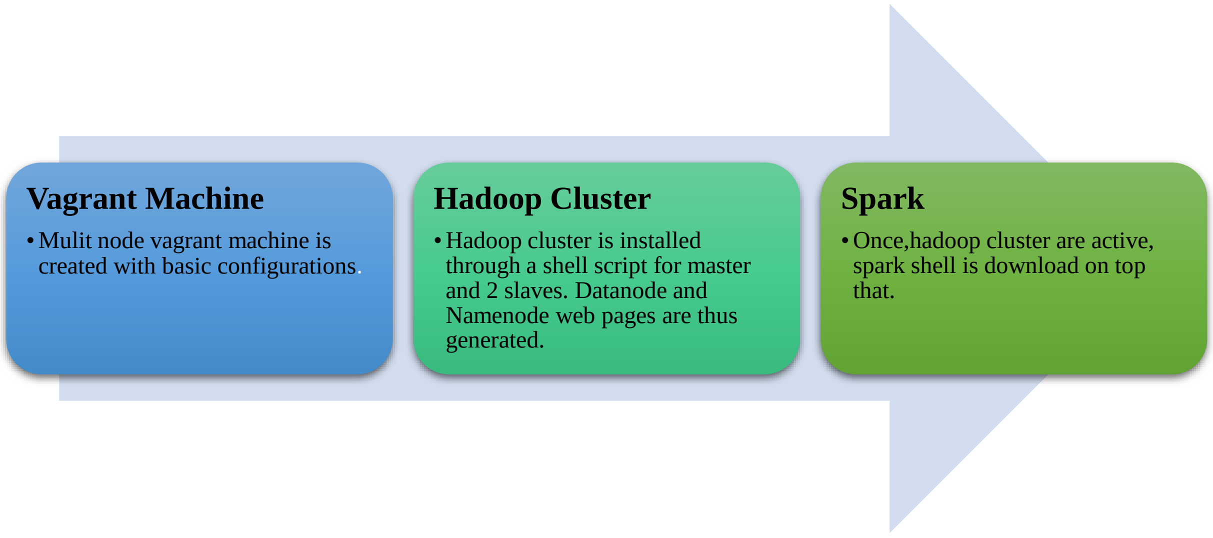
STUDENT NAME:	Beksy Saji George
STUDENT NUMBER:	C00290800
COURSE NAME:	Master’s in Data Science
DEPARTMENT:	Department of Computing
COURSE CODE:	DATAH5R04
SUPERVISOR:	Michael Gleeson
DATE OF SUBMISSION:	23rd December 2022

CONTENTS

Implementation of Apache spark on Hadoop cluster using Vagrant

Through this project, I had built apache spark on top of Hadoop cluster with the help of vagrant. Here the idea is to generate an automated multi-node machines through vagrant. By this way, we could avoid manual procedure of downloading Hadoop cluster on the number of nodes we utilizes.

1. Use Case for Big data architecture


Vagrant Machine
• Mulit node vagrant machine is created with basic configurations.
Hadoop Cluster
• Hadoop cluster is installed through a shell script for master and 2 slaves. Datanode and Namenode web pages are thus generated.
Spark
• Once,hadoop cluster are active, spark shell is download on top that.

2. Steps for creating Multi node Hadoop Cluster using Apache Spark

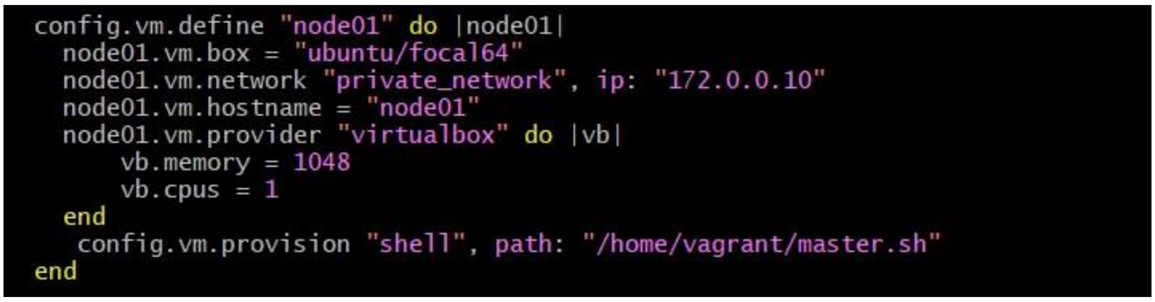
Whenever we create a virtual machine it is necessary to configure IP addresses to each nodes. The steps that I followed to create vagrant file is as follows: -
A Vagrant file is used to create 3 VMs through vagrant. Below are the machines created along with their IP addresses.

node01 172.16.0.10

node02 172.16.0.20

node03 172.16.0.30

- Create a directory called Hadoop under */c/vagrant* directory through git bash
- Use `*vagrant init*` command to generate a default vagrantfile.
- Add the basic configuration details for each VM inside the vagrant file. Below is the code for the node01.Similarly we give for node02 and node03.

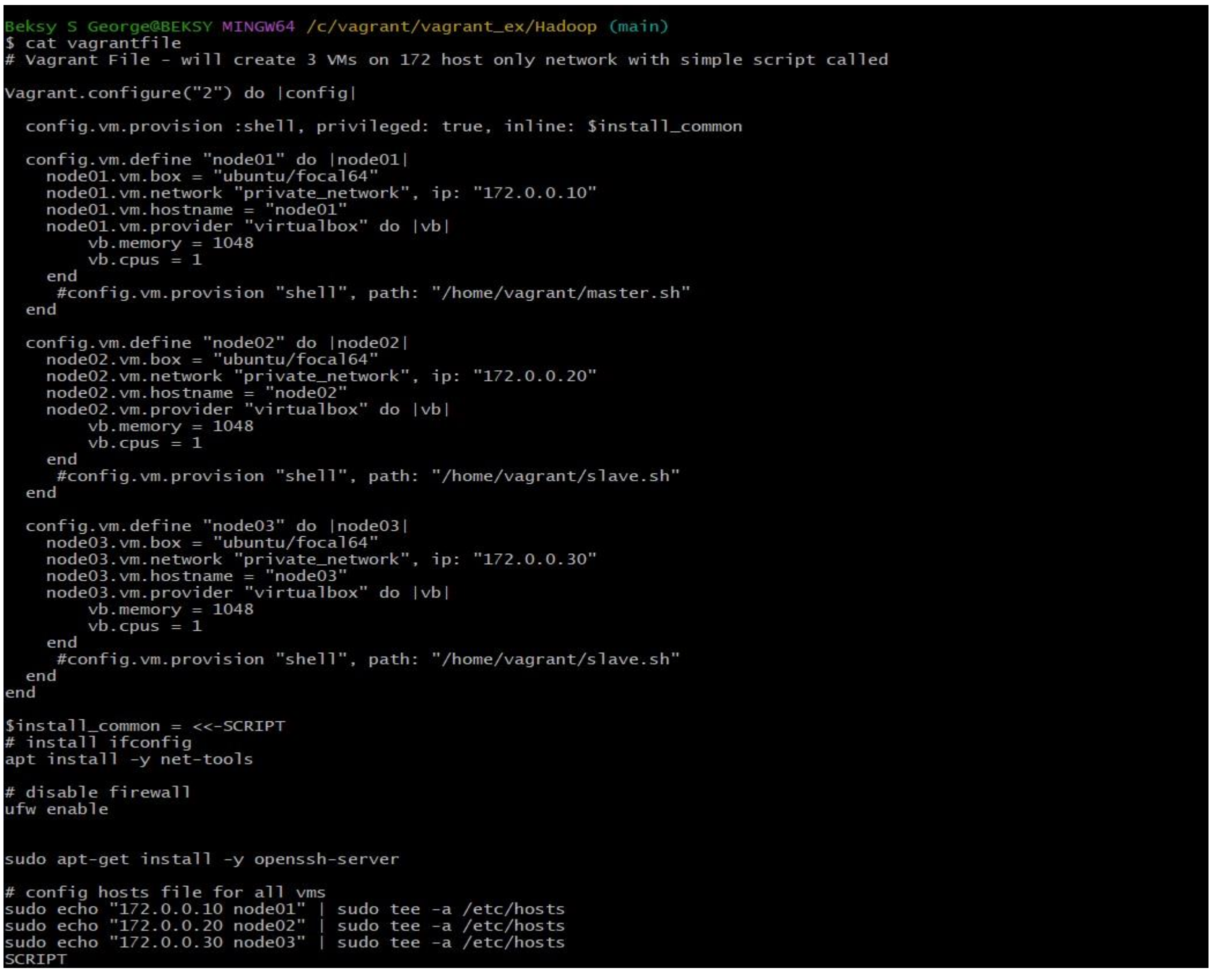


```
config.vm.define "node01" do |node01|
  node01.vm.box = "ubuntu/focal64"
  node01.vm.network "private_network", ip: "172.0.0.10"
  node01.vm.hostname = "node01"
  node01.vm.provider "virtualbox" do |vb|
      vb.memory = 1048
      vb.cpus = 1
  end
   config.vm.provision "shell", path: "/home/vagrant/master.sh"
end
```

- Using the shell provision, we also add the master node script which contains Hadoop cluster setup. For slaves, we use a different script called slave.sh
- Once every script ran successfully, use start-all.sh command to start all Hadoop daemons in Master node. Thereafter, check the job status using jps command in both master and slaves.
- If the Hadoop cluster is running fine, we could render the web page using the format-*MasterIP:HostIP*(Vagrant machines done up to here).
- After this, we install Apache spark on Hadoop cluster. Once all the installations are done, we can start the nodes from */usr/local/spark* directory. (This step is done through virtual box).

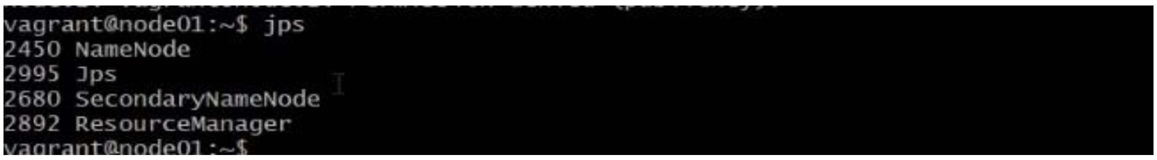
3.Screencasts

3.1 Hadoop cluster creation through vagrant



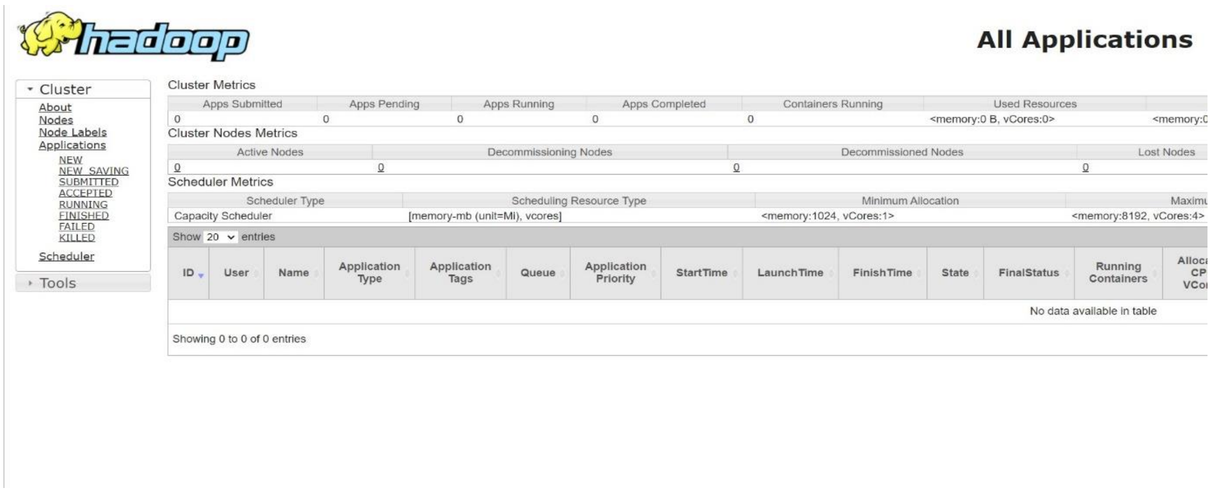
```
Beksy S George@BEKSY MINGW64 /c/vagrant/vagrant_ex/Hadoop (main)
$ cat vagrantfile
# Vagrant File - will create 3 VMs on 172 host only network with simple script called

Vagrant.configure("2") do |config|

  config.vm.provision :shell, privileged: true, inline: $install_common

  config.vm.define "node01" do |node01|
    node01.vm.box = "ubuntu/focal64"
    node01.vm.network "private_network", ip: "172.0.0.10"
    node01.vm.hostname = "node01"
    node01.vm.provider "virtualbox" do |vb|
        vb.memory = 1048
        vb.cpus = 1
    end
     #config.vm.provision "shell", path: "/home/vagrant/master.sh"
  end

  config.vm.define "node02" do |node02|
    node02.vm.box = "ubuntu/focal64"
    node02.vm.network "private_network", ip: "172.0.0.20"
    node02.vm.hostname = "node02"
    node02.vm.provider "virtualbox" do |vb|
        vb.memory = 1048
        vb.cpus = 1
    end
     #config.vm.provision "shell", path: "/home/vagrant/slave.sh"
  end

  config.vm.define "node03" do |node03|
    node03.vm.box = "ubuntu/focal64"
    node03.vm.network "private_network", ip: "172.0.0.30"
    node03.vm.hostname = "node03"
    node03.vm.provider "virtualbox" do |vb|
        vb.memory = 1048
        vb.cpus = 1
    end
     #config.vm.provision "shell", path: "/home/vagrant/slave.sh"
  end
end

$install_common = <<-SCRIPT
# install ifconfig
apt install -y net-tools

# disable firewall
ufw enable


sudo apt-get install -y openssh-server

# config hosts file for all vms
sudo echo "172.0.0.10 node01" | sudo tee -a /etc/hosts
sudo echo "172.0.0.20 node02" | sudo tee -a /etc/hosts
sudo echo "172.0.0.30 node03" | sudo tee -a /etc/hosts
SCRIPT
```

3.1.1 Vagrant node IPs



```
vagrant@node01:~$ jps
2450 NameNode
2995 Jps
2680 SecondaryNameNode
2892 ResourceManager
vagrant@node01:~$
```

3.1.2 Rendered Hadoop clusters through vagrant



All Applications

- Cluster
  - About
  - Nodes
  - Node Labels
  - Applications
    - NEW
    - NEW_SAVING
    - SUBMITTED
    - ACCEPTED
    - RUNNING
    - FINISHED
    - FAILED
    - KILLED
  - Scheduler
- Tools

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Used Resources	
0	0	0	0	0	<memory:0 B, vCores:0>	<memory:0

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	Decommissioned Nodes	Lost Nodes
0	0	0	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximu
Capacity Scheduler	[memory-mb (unit=Mi), vcores]	<memory:1024, vCores:1>	<memory:8192, vCores:4>

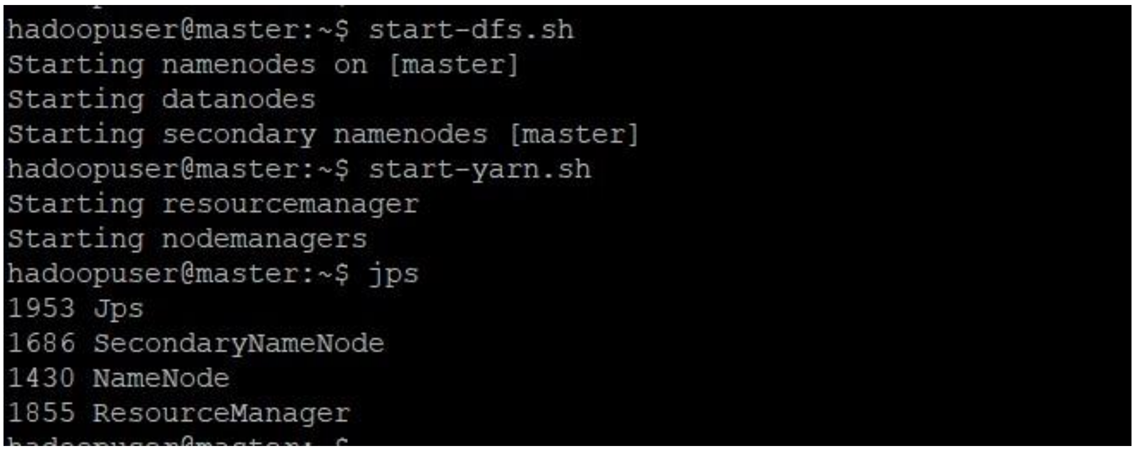
Show 20 entries

ID	User	Name	Application Type	Application Tags	Queue	Application Priority	StartTime	LaunchTime	FinishTime	State	FinalStatus	Running Containers	Alloca CP VCo
No data available in table													

Showing 0 to 0 of 0 entries

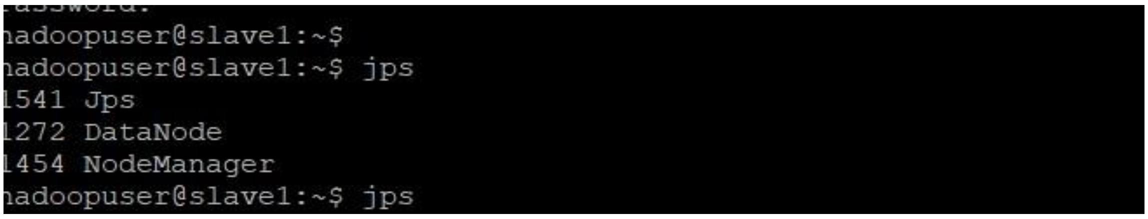
3.2 Apache Spark through Virtual Box

3.2.1 Master node up



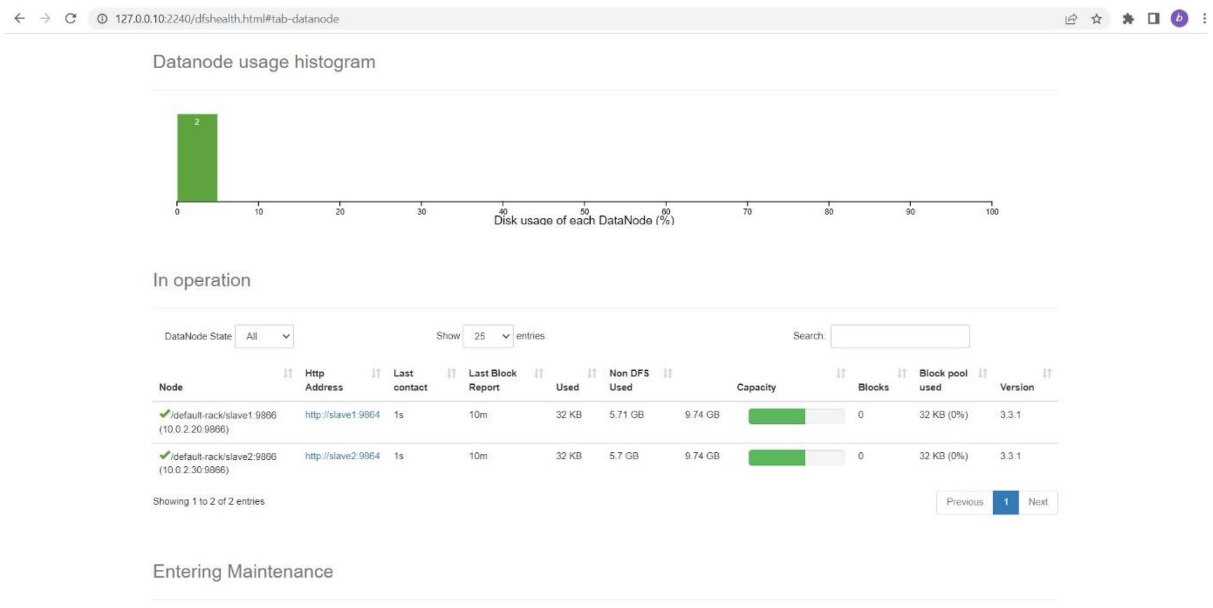
```
hadoopuser@master:~$ start-dfs.sh
Starting namenodes on [master]
Starting datanodes
Starting secondary namenodes [master]
hadoopuser@master:~$ start-yarn.sh
Starting resourcemanager
Starting nodemanagers
hadoopuser@master:~$ jps
1953 Jps
1686 SecondaryNameNode
1430 NameNode
1855 ResourceManager
```

3.2.2 Slave nodes



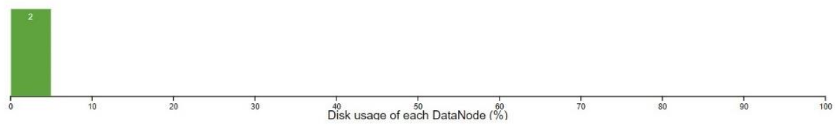
```
hadoopuser@slave1:~$
hadoopuser@slave1:~$ jps
1541 Jps
1272 DataNode
1454 NodeManager
hadoopuser@slave1:~$ jps
```

3.2.3 Data node



127.0.0.10:2240/dfshealth.html#tab-datanode

Datanode usage histogram



In operation

DataNode State All Show 25 entries Search:

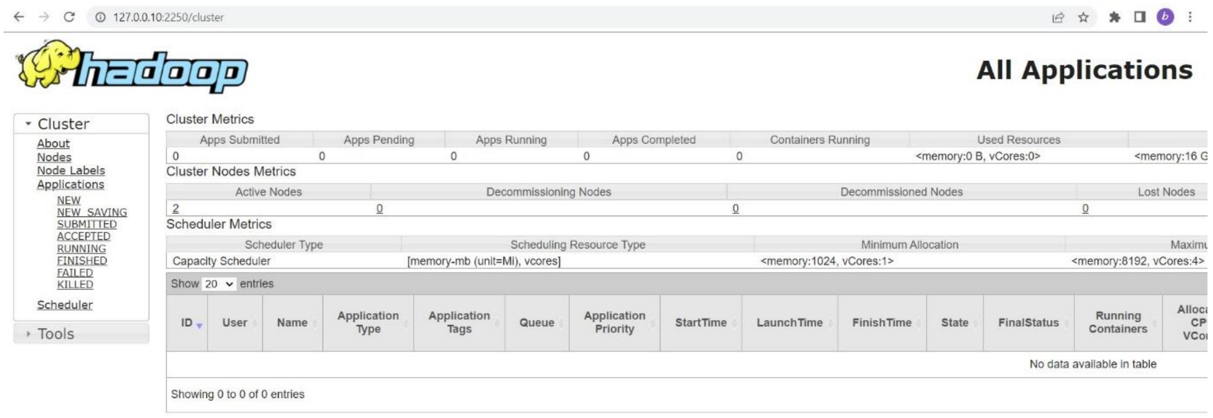
Node	Http Address	Last contact	Last Block Report	Used	Non DFS Used	Capacity		Blocks	Block pool used	Version
✓/default-rack/slave1:9866 (10.0.2.20:9866)	http://slave1:9864	1s	10m	32 KB	5.71 GB	9.74 GB	🟩	0	32 KB (0%)	3.3.1
✓/default-rack/slave2:9866 (10.0.2.30:9866)	http://slave2:9864	1s	10m	32 KB	5.7 GB	9.74 GB	🟩	0	32 KB (0%)	3.3.1

Showing 1 to 2 of 2 entries

Previous 1 Next

Entering Maintenance

3.2.4 All Applications



127.0.0.10:2250/cluster

All Applications

- Cluster
  - About
  - Nodes
  - Node Labels
  - Applications
    - NEW
    - NEW_SAVING
    - SUBMITTED
    - ACCEPTED
    - RUNNING
    - FINISHED
    - FAILED
    - KILLED
  - Scheduler
- Tools

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Used Resources	
0	0	0	0	0	<memory:0 B, vCores:0>	<memory:16 G

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	Decommissioned Nodes	Lost Nodes
2	0	0	0

Scheduler Metrics

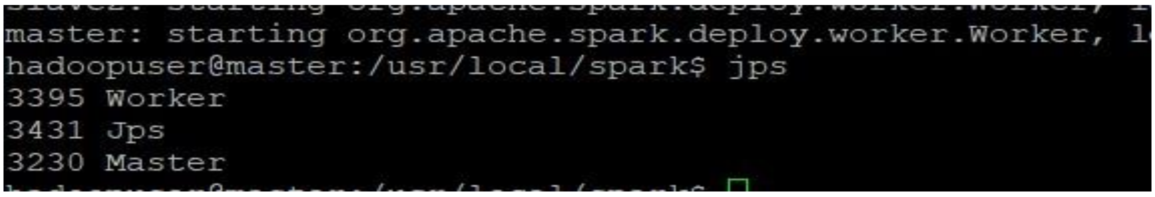
Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximu
Capacity Scheduler	[memory-mb (unit=Mi), vcores]	<memory:1024, vCores:1>	<memory:8192, vCores:4>

Show 20 entries

ID	User	Name	Application Type	Application Tags	Queue	Application Priority	StartTime	LaunchTime	FinishTime	State	FinalStatus	Running Containers	Alloca CP VCo
											No data available in table		

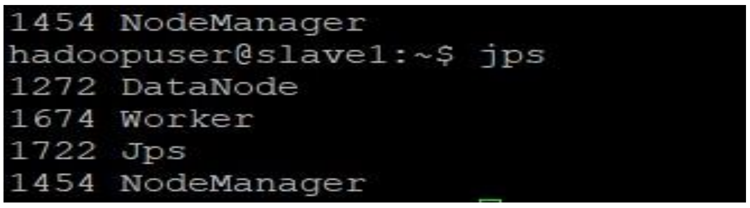
Showing 0 to 0 of 0 entries

3.2.5 Spark Master



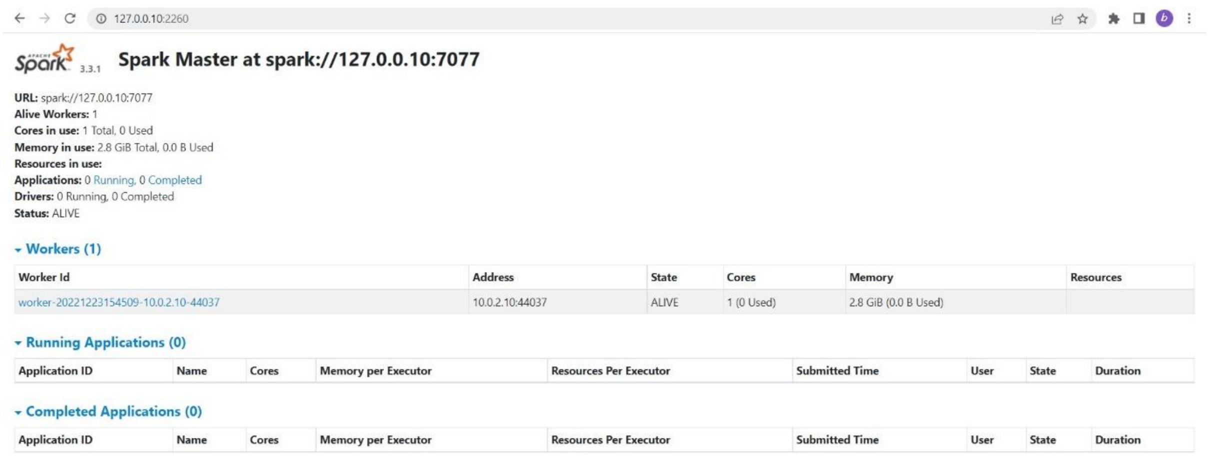
```
master: starting org.apache.spark.deploy.worker.Worker, l
hadoopuser@master:/usr/local/spark$ jps
3395 Worker
3431 Jps
3230 Master
```

3.2.6 Spark Slave



```
1454 NodeManager
hadoopuser@slave1:~$ jps
1272 DataNode
1674 Worker
1722 Jps
1454 NodeManager
```

3.2.7 Spark Rendered Page



127.0.0.10:2260

3.3.1 **Spark Master at spark://127.0.0.10:7077**

URL: spark://127.0.0.10:7077
Alive Workers: 1
Cores in use: 1 Total, 0 Used
Memory in use: 2.8 GiB Total, 0.0 B Used
Resources in use:
Applications: 0 Running, 0 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers (1)

Worker Id	Address	State	Cores	Memory	Resources
worker-20221223154509-10.0.2.10-44037	10.0.2.10:44037	ALIVE	1 (0 Used)	2.8 GiB (0.0 B Used)	

Running Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration

Completed Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration

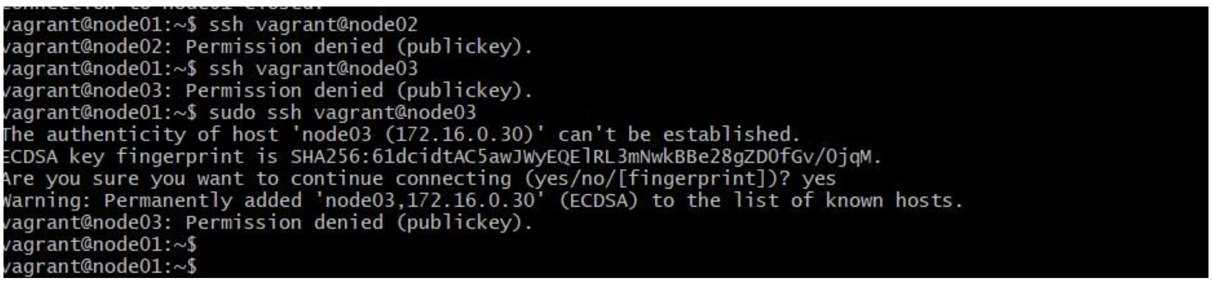
4. Issues Faced in Implementation

During Vagrant machine installation, there was some issues connecting with virtual boxes.

- While connecting with one node to another we used the command
  ssh vagrant@node01 -p 2222

  Here when I tried, got the permission denied issue.
  Checked all the /etc/hosts of all nodes. But the nodes are not getting connected with each other.

  Due to the above issue, while starting master node, the web pages are rendering but the slave nodes are not in sync with it. The above screenshots shows the webpages rendered for this.



```
vagrant@node01:~$ ssh vagrant@node02
vagrant@node02: Permission denied (publickey).
vagrant@node01:~$ ssh vagrant@node03
vagrant@node03: Permission denied (publickey).
vagrant@node01:~$ sudo ssh vagrant@node03
The authenticity of host 'node03 (172.16.0.30)' can't be established.
ECDSA key fingerprint is SHA256:61dcidtAC5awJWyEQElRL3mNwkBBe28gZD0fGv/0jqM.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'node03,172.16.0.30' (ECDSA) to the list of known hosts.
vagrant@node03: Permission denied (publickey).
vagrant@node01:~$
vagrant@node01:~$
```

5. References

https://medium.com/@jootorres_11979/how-to-set-up-a-hadoop-3-2-1-multi-node-cluster-on-ubuntu-18-04-2-nodes-567ca44a3b12

https://medium.com/@jootorres_11979/how-to-install-and-set-up-an-apache-spark-cluster-on-hadoop-18-04-b4d70650ed42

https://iceburn.medium.com/basic-knowledge-of-ssh-95cb53d55c08

https://www.linode.com/docs/guides/install-configure-run-spark-on-top-of-hadoop-yarn-cluster/

https://sysadmins.co.za/setup-hadoop-2-7-multinode-cluster-on-ubuntu/