CUTLER Platform Deployment Documentation

**Thuy Truong, DELL EMC**

# Platform Deployment

This section covers topic of i) preparing the cutler-orchestrator, ii) preparing Ubuntu-base image, and iii) platform deployment

1. **Preparing the cutler-orchestrator**
   1. On vSphere vCenter, prepare a VM “cutler-orchestrator” with 2vCPU, 2GB, 20GB, Ubuntu Linux (64-bit)
   2. Download Ubuntu iso (either desktop or server image depending on the requirement)
      1. Ubuntu live server 18.04.3 LTS is downloaded for this orchestrator
   3. Connect Ubuntu ISO to the VM and boot the VM up with iso for Ubuntu installation.
   4. Follow the steps in the GUI to install Ubuntu OS.
   5. Connect to the correct network.
   6. Now VM with Ubuntu OS should be ready.
   7. Create user orchestrator/cutler (your choice)
      1. Add orchestrator user to sudoer

sudo usermode -aG sudo orchestrator

confirm the user is in sudo group by

su - orchestrator

and run any sudo command

* 1. Install VMware tools on the machine (optional?)
  2. For software installation,
     1. install python

sudo apt-get install python -y

apt-get install python-jmespath

* + 1. install terraform (download latest zip file, unzip and copy the bin file terraform to /usr/local/bin)

You might have to install unzip first if the tool is not in the system:

$ sudo apt install unzip -y

Terraform is distributed as a single binary. Install Terraform by unzipping it and moving it to a directory included in your system's PATH.

$ curl [https://releases.hashicorp.com/terraform/0.12.10/terraform\_0.12.10\_linux\_amd64.zip --output terraform\_0.12.10\_linux\_amd64.zip](https://releases.hashicorp.com/terraform/0.12.10/terraform_0.12.10_linux_amd64.zip%20--output%20terraform_0.12.10_linux_amd64.zip)

$ unzip terraform\_0.12.10\_linux\_amd64.zip

sudo mv terraform /usr/local/bin/

* + 1. install ansible-playbook, latest version

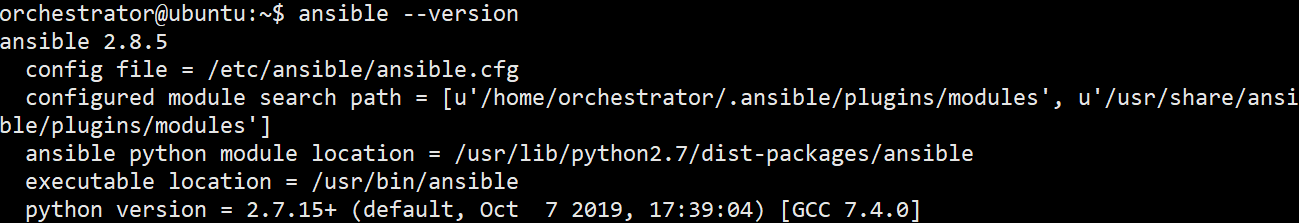
sudo apt-get update

sudo apt-get upgrade -y

sudo apt-add-repository ppa:ansible/ansible

sudo apt-get update

sudo apt-get install ansible -y



The below are only if you want to use global ansible config file:

* Good link: <https://www.techrepublic.com/article/how-to-install-ansible-on-ubuntu-server-18-04/>
* Create a new directory (on the cutler-orchestrator)

sudo mkdir /etc/ansible/group\_vars

* Create a new file

sudo nano /etc/ansible/group\_vars/servers

and then add the following line:

ansible\_ssh\_user: USERNAME

Where USERNAME is the username on the remote node, e.g., *cutleradmin*). This can be done on the command line as well, with option -u. In the project, we do -u option to reduce the dependencies.

1. **Ubuntu image**
   1. VM Preparation
      1. On vSphere vCenter, prepare a minimum VM “cutler-vm” with 2vCPU, 2GB, 10GB, Ubuntu Linux (64-bit)
      2. Download Ubuntu iso (either desktop or server image depending on the requirement)
         1. Ubuntu Desktop 18.04.3 LTS is downloaded for this project.
      3. Connect Ubuntu ISO to the VM and boot the VM up with iso for Ubuntu installation.
      4. Follow the steps in the GUI to install Ubuntu OS.
      5. Now VM with Ubuntu OS should be ready.
   2. User Preparation – need to be consistent in the ansible scripts (service deployment)
      1. Add “cutleradmin” user

$sudo adduser cutleradmin

* + 1. Add user into sudo group

$sudo usermod -aG sudo cutleradmin

* + 1. Test it out with $su – cutleradmin
  1. Install VMware tools on the VM
     1. Mount the vmtools iso onto the cdrom
     2. Copy the taz file to a tmp folder
     3. Untar the file
     4. Sudo ./install…. -d
  2. Pre-Packages Installation – follow Baseline-VM-config.txt for installing all the required packages (the content of the file can be found below).
  3. sudo rm /etc/netplan/cloud-xyz.cfg (to remove duplicate configuration, vm-network will be in /etc/network/interface)
  4. Now Shut down the VM, switch the network adapter back to “VM Network” and set CD device back to Client ISO.
  5. Clone/Migrate “cutler-vm” into a template “cutler-template” – roughly 11GB

Good link: <https://blog.inkubate.io/create-a-ubuntu-18-04-terraform-template-for-vmware-vsphere/>

1. **Terraform for VM Deployment**
   1. Terraform scripts are provided to run either by each VM creations for each cluster or one script to deploy all VMs in Cutler platform.
   2. Terraform folder structure:

terraformhome

vSphere-pilotX.tfvars – info of VMs, data center/host/resource pool/network

vSphere-config.tf – vsphere configuration

variables.tf – contains all the defined variables

* 1. To run terraform

$cd path/to/terraform/platform/folder

$terraform init -> to initiate the terraform environment in the folder

$terraform apply -var-file=”vSphere-P2.var” -> to create VMs specified in the var-file. This step will generate the terraform state file “terraform.tfstate” for all VMs created in the step

$terraform destroy -var-file=”vSphere-P2.var” -> this will destroy all the created VMs that were deployed earlier and the states have been stored in the terraform state file terraform.tfstate.

# Service Deployment

There are three clusters in Cutler Platform, description + figure to be added in as in WP2.

Ansible scripts are developed for automated service deployment.



Where

cutler-cluster-1 is for Hadoop cluster deployment

cutler-cluster-2 is for Kafka cluster deployment

cutler-cluster-3 is for Elasticsearch + Kibana (ELK stack) cluster deployment

to deploy all-in-one step: run

$./cutler-service-deployment.sh

to deploy one cluster at a time: run

$./cutler-install-cluster-1/2/3.sh

There are some post-requisites that partners required, those can be deployed via ansible as well.

$./cutler-install-post-requisites.sh

**Baseline-VM-config.txt**

# Baseline configuration of the CUTLER VMs

The following installation/configuration steps were applied to the c1, c2, c3, c4 baseline VM images:

- Updated the resolution to 1440x900

- sudo apt update

- sudo apt upgrade

Base packages/tools:

- sudo apt install net-tools

- sudo apt install ifupdown

- sudo mkdir /etc/dhcp3

- sudo apt install default-jdk

- sudo apt install python3.6

- sudo apt install python3-pip

- sudo apt install python3-pandas

- sudo apt install python3-bs4

- sudo apt install python3-dateutil

- sudo apt install python3-requests

- sudo apt install python-pip

- apt install openjdk-11-jdk-headless

- sudo apt-get install nmap

- pip install pyexcel-ezodf

- sudo apt install curl

- pip3 install -U selenium

Firewall:

- sudo ufw enable

- sudo ufw default deny incoming

- sudo ufw default allow outgoing

- sudo ufw allow 22

- sudo ufw allow 80

- sudo ufw allow 443

- sudo ufw allow 21/tcp

IPS:

- sudo apt install fail2ban

- sudo cp /etc/fail2ban/jail.conf /etc/fail2ban/jail.local

- edit the .local file

- "5m" ban time

- uncomment "enabled = true"

- everything else default

- sudo systemctl restart fail2ban

Add non-root user:

- sudo adduser cutler-user

Google Chrome:

- wget -q -O - https://dl-ssl.google.com/linux/linux\_signing\_key.pub | sudo apt-key add -

- sudo sh -c 'echo "deb [arch=amd64] http://dl.google.com/linux/chrome/deb/ stable main" >> /etc/apt/sources.list.d/google.list'

- sudo apt update

- sudo apt install google-chrome-stable

- wget https://chromedriver.storage.googleapis.com/73.0.3683.68/chromedriver\_linux64.zip

- unzip chromedriver\_linux64.zip

- sudo mv chromedriver /usr/bin/

- chmod +x /usr/bin/chromedriver

GeckoDriver:

- wget https://github.com/mozilla/geckodriver/releases/download/v0.24.0/geckodriver-v0.24.0-linux64.tar.gz

- tar -xvf geckodriver-v0.24.0-linux64.tar.gz

- sudo mv geckodriver /bin

Crawlers (for ES nodes):

- pip3 install virtualenv

- sudo add-apt-repository ppa:alex-p/tesseract-ocr

- sudo apt update

- sudo apt install tesseract-ocr

- sudo apt install python3-xlrd

- sudo apt install python3-openpyxl

node.js:

- curl -s -O https://deb.nodesource.com/node\_10.x/pool/main/n/nodejs/nodejs\_10.13.0-1nodesource1\_amd64.deb

- sudo dpkg -i nodejs\_10.13.0-1nodesource1\_amd64.deb

Enable ssh server:

sudo apt update

sudo apt install openssh-server

sudo systemctl status ssh

**Debug Issues**

* Change the cutler admin account: cutleradmin, avoid special character in the account, Ansible complains and not continue the script if “-“
* IP\_address\_start is not defined correctly
* Running command with option -k for ask for ssh password, and -u for remote\_user to avoid permission denied(publickey,password)

ansible-playbook -i cutler-cluster-2/inventory.ini cutler-cluster-2/cc2.yml --ask-become-pass **-u cutleradmin -k**

* When getting into issue, and don’t know what it is, try to run with to have the debug mode on for better understanding of the issue.

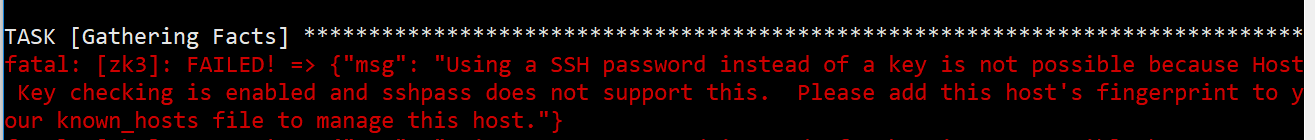
$ ANSIBLE\_DEBUG=1 ansible-playbook -i cutler-cluster-2/inventory.ini cutler-cluster-2/cc2.yml --ask-become-pass -u cutleradmin -k

**Issue#1 – Host’s fingerprints**

TASK [Gathering Facts]

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

fatal: [zk3]: FAILED! => {"msg": "Using a SSH password instead of a key is not possible because Host Key checking is enabled and sshpass does not support this. Please add this host's fingerprint to your known\_hosts file to manage this host."}



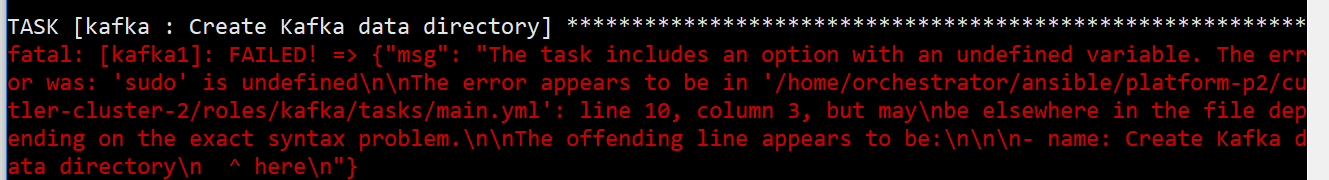
For this issue, just add the fingerprint to the running host. For example, the running host (the host that is currently running the ansible script, e.g., cutler-orchestrator) when first ssh to the ansible host, it will be asked to add the fingerprint of the ansible host into its known-host file, so just try ssh to the ansible hosts for saving the fingerprints.

**Issue#2**

TASK [kafka : Create Kafka data directory]

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

fatal: [kafka1]: FAILED! => {"msg": "The task includes an option with an undefined variable. The error was: 'sudo' is undefined\n\nThe error appears to be in '/home/orchestrator/ansible/platform-p2/cutler-cluster-2/roles/kafka/tasks/main.yml': line 10, column 3, but may\nbe elsewhere in the file depending on the exact syntax problem.\n\nThe offending line appears to be:\n\n\n- name: Create Kafka data directory\n ^ here\n"}



This is just a syntax error.

Go into the file to check the error: /home/orchestrator/ansible/platform-p2/cutler-cluster-2/roles/kafka/tasks/main.yml'

**Issue#3 – stuck at some task**

TASK stuck at Gathering Facts/or something

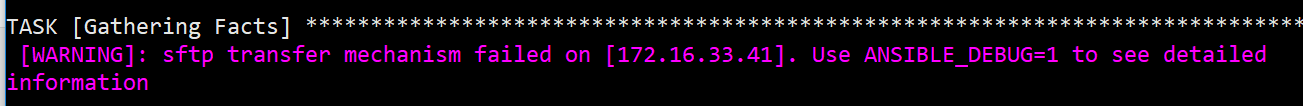
$rm -rf ~/.ansible

**Rebooting the ansible\_hosts actually help!!!**

**Issue#4 – stuck at stfp file transfer**

[WARNING]: scp transfer mechanism failed on [172.16.33.41]. Use ANSIBLE\_DEBUG=1 to see detailed information

fatal: [kafka2]: UNREACHABLE! => {"changed": false, "msg": "Data could not be sent to remote host \"172.16.33.41\". Make sure this host can be reached over ssh: packet\_write\_wait: Connection to 172.16.33.41 port 22: Broken pipe\r\n", "unreachable": true}





The ansible scripts use sftp to transfer files to the host, make sure the subsystem is running (/etc/ssh/sshd-config)

Subsystem sftp /usr/lib/ssh/sftp-server

Or if necessary, change the file transferring protocol to scp by modifying the ansible.config (/etc/ansible/ansible.cfg or the local ansible.cfg) file

[ssh\_connection]

scp\_if\_ssh=True

and in the host file (/etc/ansible/hosts or inventory.ini) use the ipaddress as "cutleradmin@172.16.33.41" instead of just "172.16.33.41".

if only one host is affected, there’s no need to change the config for the whole setup, this can be remedied in the hosts file:

alias ansible\_host=172.16.33.40 ansible\_ssh\_transfer\_method=scp

**Here are some facts regarding scp vs sftp:**

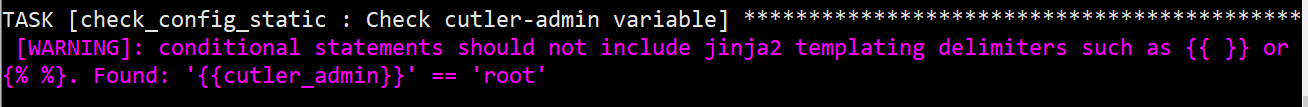
Both SCP and SFTP run over TCP port 22 by default, and both are also used for transferring files.

* Speed - SCP is generally much faster. SFTP has to ACK (acknowledge) every tiny packet, while SCP does not. That's why the disparity becomes more evident in high latency networks. This will happen when the cutler-orchestrator is from different data center and orchestrating the service deployment onto ansible hosts located in different data centers in a high latency network.
* Security – even because they are both operating on SSH and providing the same security feature that SSH can offer. That includes **data-in-motion encryption** and **public key authentication**. When combined with password authentication (which SSH also supports), public key authentication allows SCP and SFTP logins to be secured by two-factor authentication. Data-in-motion encryption and 2-factor authentication are required in data security and privacy laws/regulations like HIPAA and PCI-DSS.
* Functionality – SFTP as it is designed for detail file transferring and offer file operations such as generating directory listings (to view files within a dir), create directories, delete directories, delete files, etc.
* Even in file size - Neither SCP nor SFTP have any file size limitations.
* Support for resume file transfers - SFTP

**Issue#5 – Warning**

[WARNING]: conditional statements should not include jinja2 templating delimiters such as {{ }} or

{% %}. Found: '{{cutler\_admin}}' == 'root'



This warning was produced on purposed since ansible 2.3.

Change to when: cutler\_admin == ‘root’

**Issue#6 – Root access for some tasks**

Some tasks require root access:

- name: Add Elasticsearch apt key.

become: true

- name: Add Kibana repository.

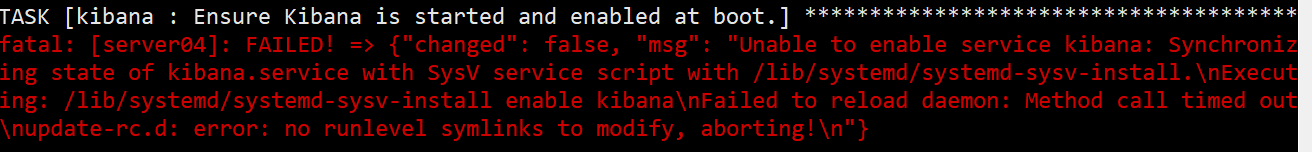
become: true

become\_method: sudo

**Issue#7 – Enable Kibana failed at timeout with /lib/systemd/systemd-sysv-install enable kibana**

TASK [kibana : Ensure Kibana is started and enabled at boot.] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

fatal: [server04]: FAILED! => {"changed": false, "msg": "Unable to enable service kibana: Synchronizing state of kibana.service with SysV service script with /lib/systemd/systemd-sysv-install.\nExecuting: /lib/systemd/systemd-sysv-install enable kibana\nFailed to reload daemon: Method call timed out\nupdate-rc.d: error: no runlevel symlinks to modify, aborting!\n"}



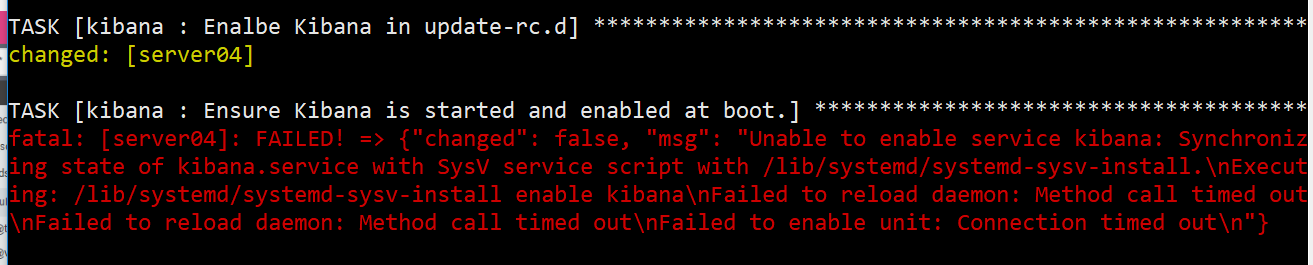
After enable Kibana in update-rc.d, it has the symlinks, but still failed to load the daemon.

TASK [kibana : Enalbe Kibana in update-rc.d] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

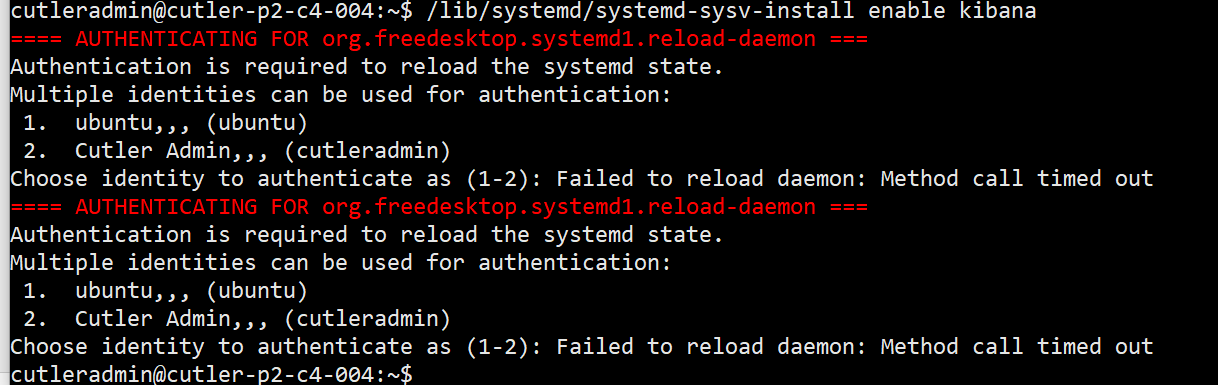
changed: [server04]

TASK [kibana : Ensure Kibana is started and enabled at boot.] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

fatal: [server04]: FAILED! => {"changed": false, "msg": "Unable to enable service kibana: Synchronizing state of kibana.service with SysV service script with /lib/systemd/systemd-sysv-install.\nExecuting: /lib/systemd/systemd-sysv-install enable kibana\nFailed to reload daemon: Method call timed out\nFailed to reload daemon: Method call timed out\nFailed to enable unit: Connection timed out\n"}



**This is due to the Kibana host has multiple users (sudo users).**



**Suggestion: Try to change the command to in the role**

sudo /bin/systemctl enable kibana.service

**instead of** /lib/systemd/systemd-sysv-install enable kibana

or remove the other user.

use become : yes to become root.

Check status with

**sudo** journalctl -u kibana.service

**Issue#8 – Elastic does not listen on the interface (only loopback interface).**

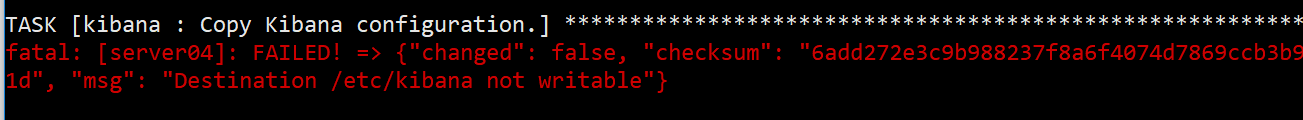
**Set in the var:**

network.host: 0.0.0.0

discovery.seed\_hosts:

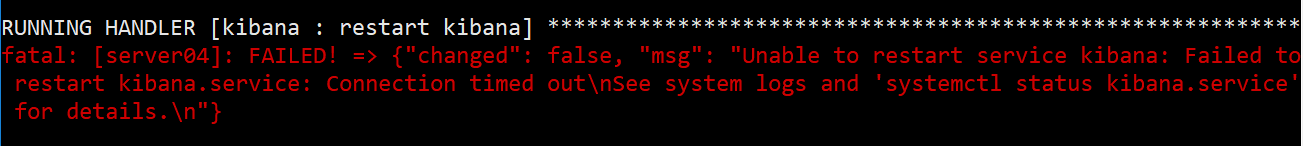
- "172.16.32.43:9300"

**Issue#9** – accessing to /etc/kibana require sudo



sudo : yes

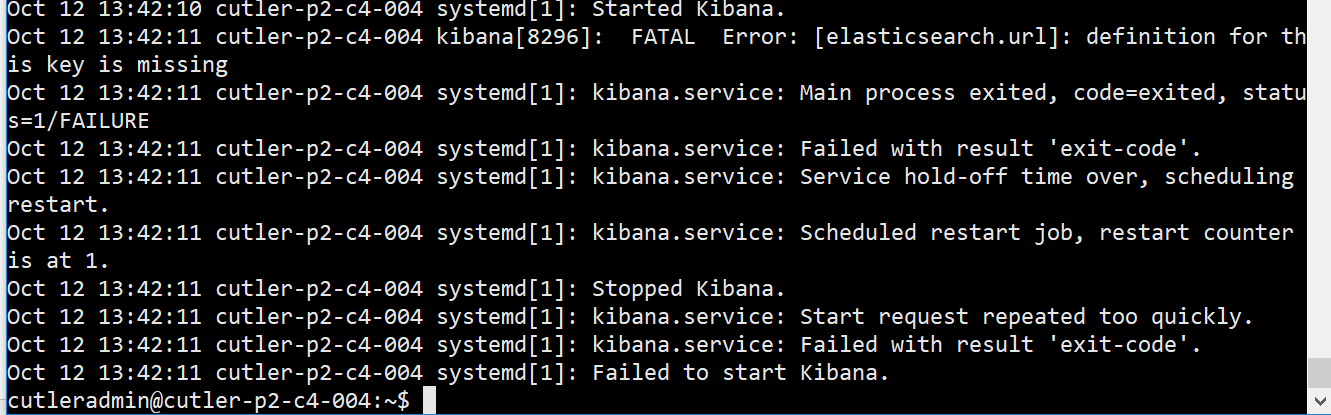
**Issue#10**



**Issue#11 – Kibana failed to start (start request repeated too quickly) - Error: [elasticsearch.url]: definition for this key is missing**

$ sudo journalctl -u kibana.service > status

$ tail status -n 30



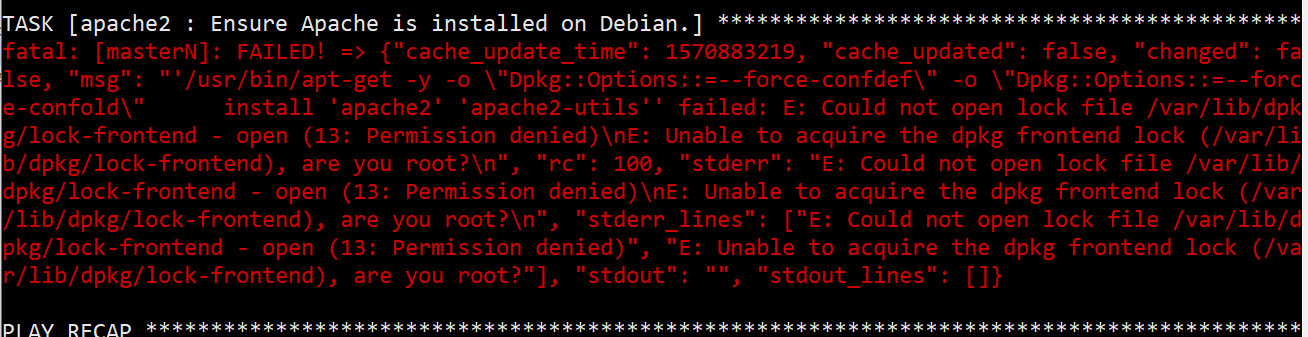
Kibana does not understand the elasticsearch.url, in the kibana template, try to use this:

~~# The URL of the Elasticsearch instance to use for all your queries.~~

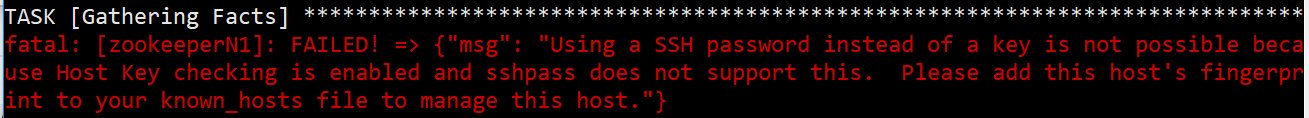
~~elasticsearch.url: {{ kibana\_elasticsearch\_url }}~~

elasticsearch.hosts: ["http://localhost:9200"]

**Issue#12 – su require**



**Issue#13 – host\_key\_checking**

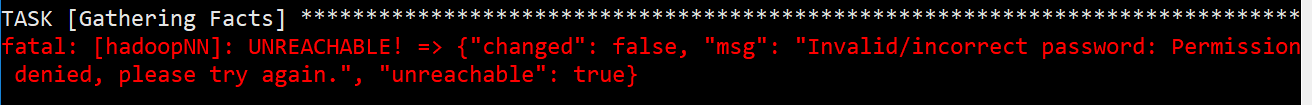


Set in the ansible.cfg:

[defaults]

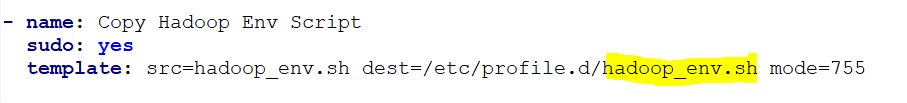
host\_key\_checking = false

**Issue#14 – Permission Denied**

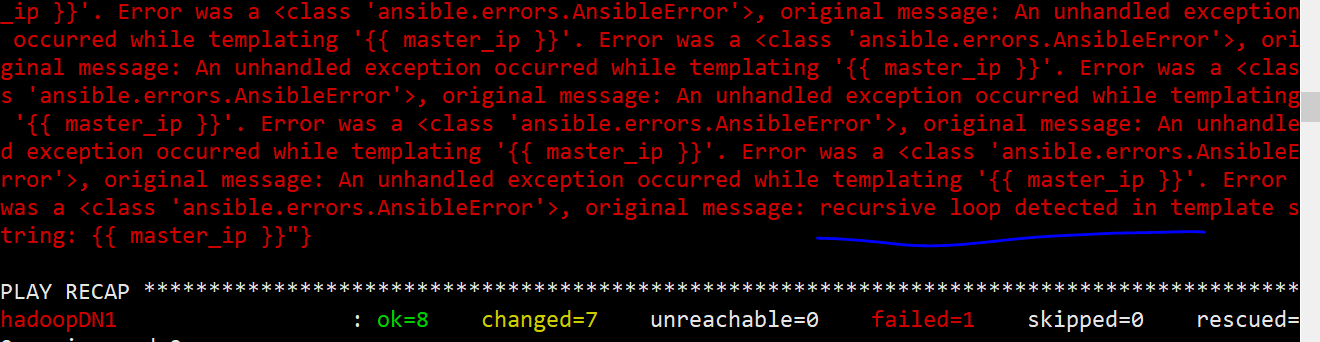


Use this option in the command:

--ask-become-pass -k -e 'ansible\_ssh\_user=cutleradmin'

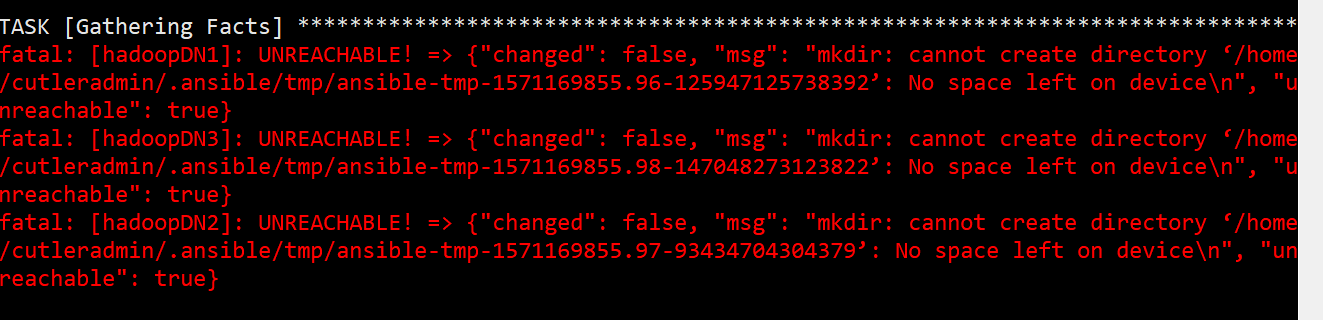


**Recursive Loop**



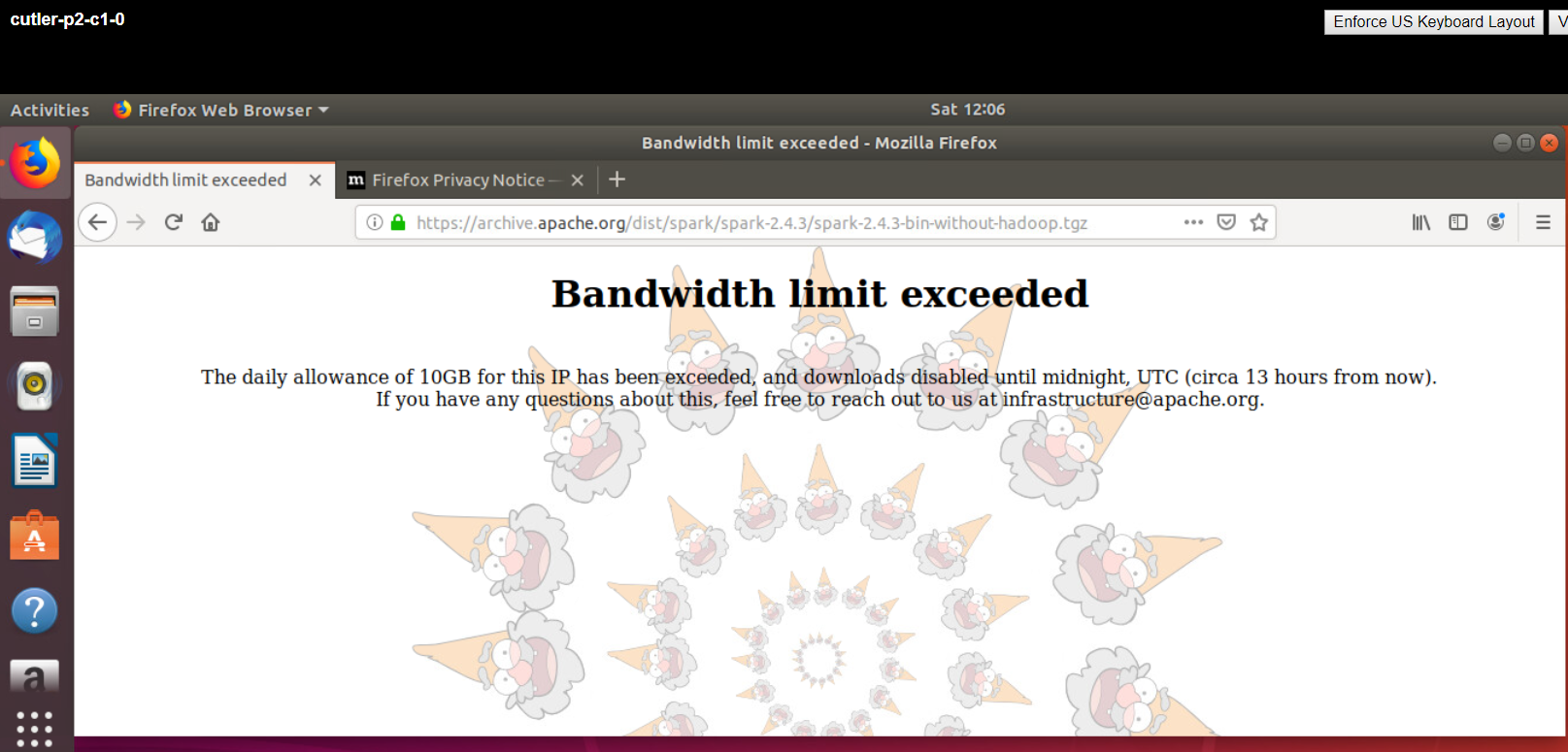
**Issue# No space left on device**

Migrating the existing VMs to different datastore to free up the space in the host.



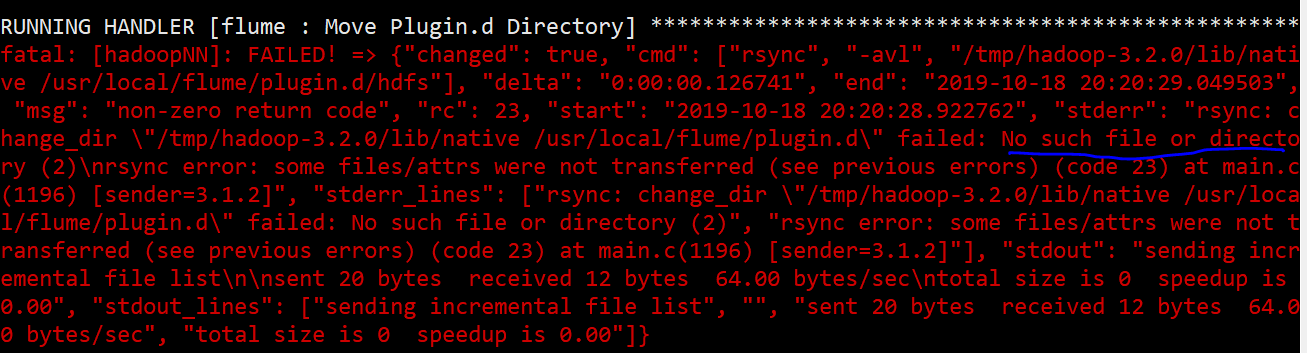
**Issue# - get\_url returns SERVICE\_UNAVAILABLE**

Check if the link is still there. If so, this is because the host already exceeds 10GB download allowance from this website.

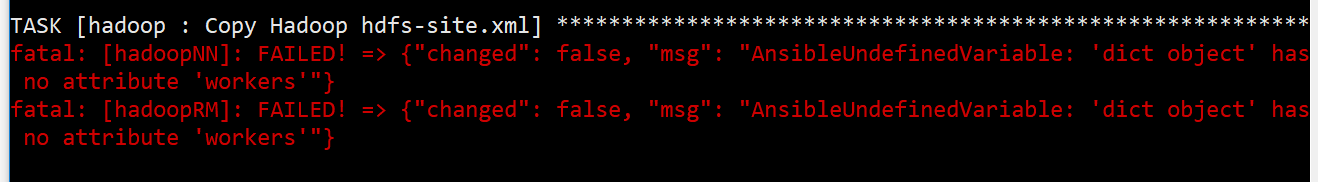


**Issue# Flume**

**Wrong directory, the first time it wasn’t fast enough to create the folder. Usually need to run the 2nd time, the folder will be there.**



**Issue#**



# Testing Phase

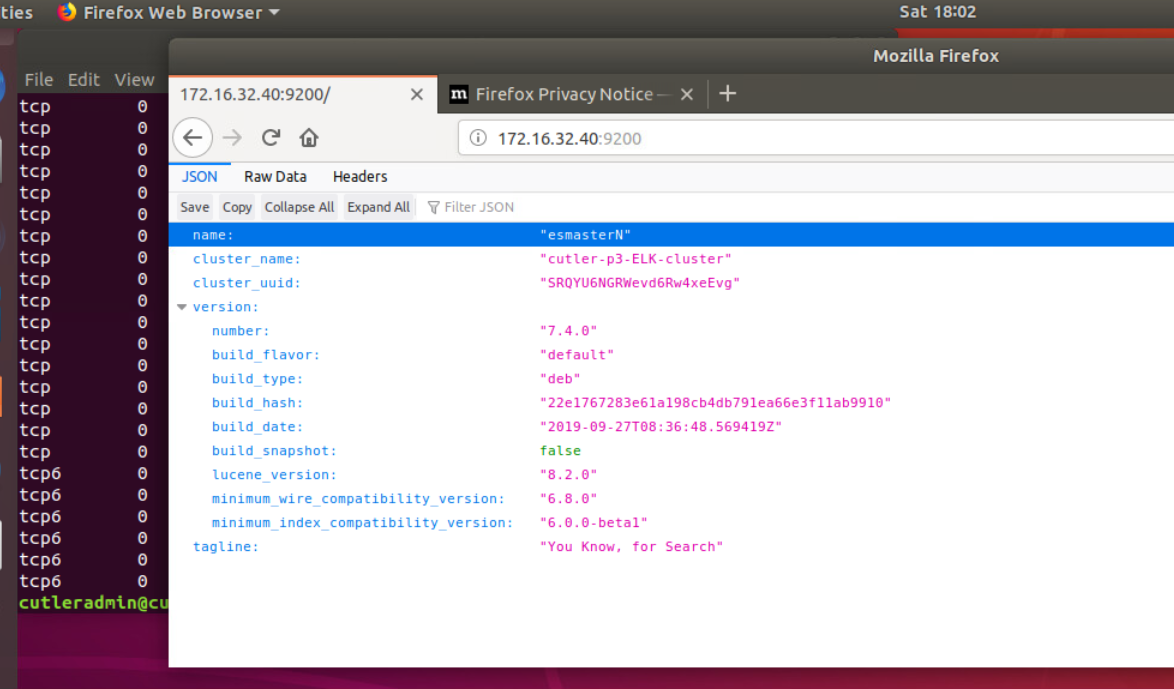
**Test Elastic Search**

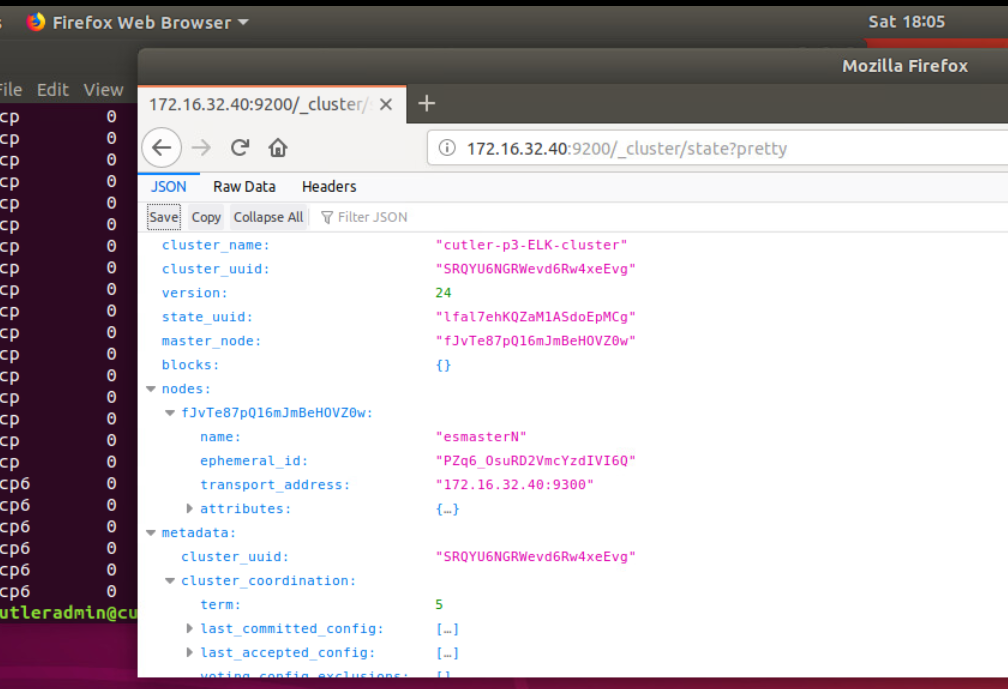
Test to see if elastic cluster is formed correctly (no split brain, etc.) <http://(master_node_ip):9200/_cluster/state?pretty>

curl -X GET <http://(master_node_ip):9200/_cluster/health?pretty>

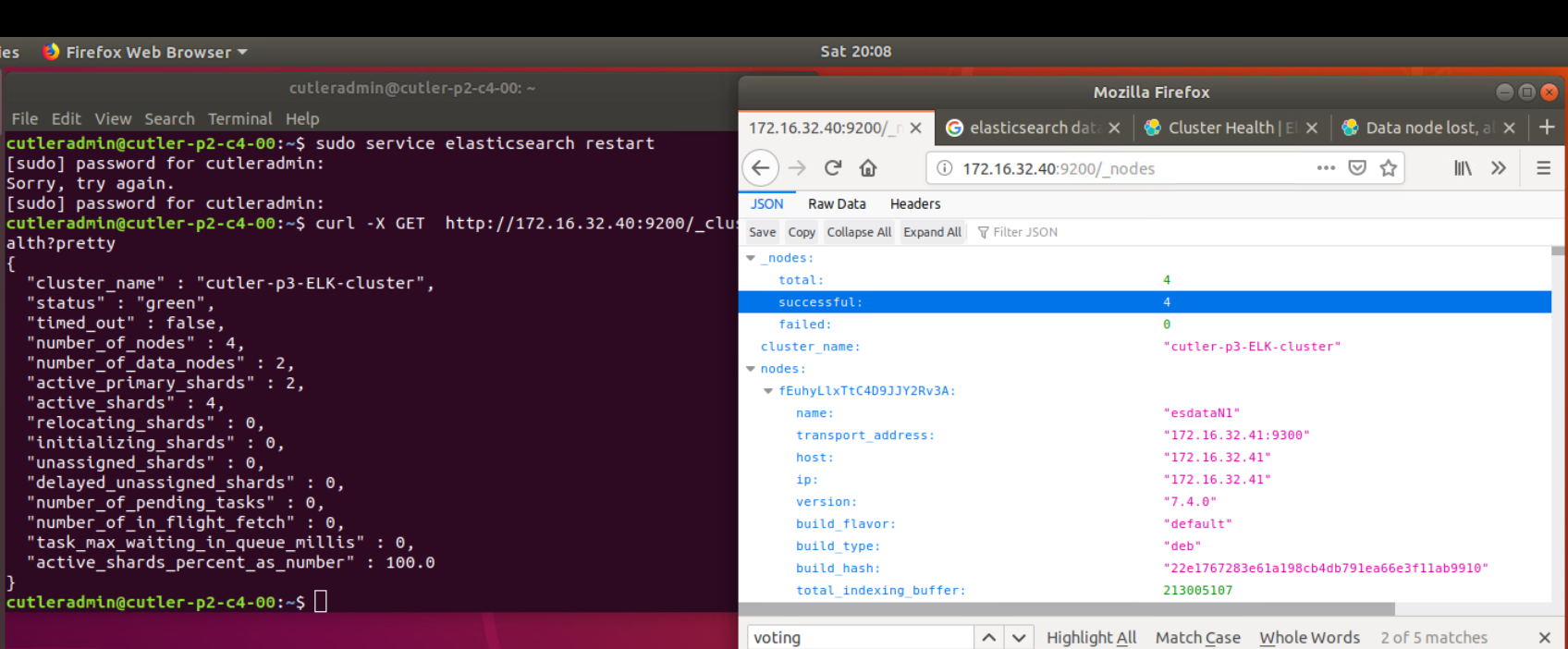
**ElasticSearch Master Node**

Format





**Get node list:** curl -X GET http://172.16.32.40:9200/\_nodes



**Check log file:**

sudo nano /var/log/elasticsearch/cutler-p2-ELK-cluster.log

**Useful links:**

<https://tecadmin.net/setup-elasticsearch-on-ubuntu/>

**Firewall:**

sudo ufw allow 9200

sudo ufw allow 9300

for all nodes

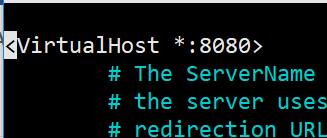
**Testing Apache:**



**Issue#**

Mismatch the port number apache2 setup to listen on 8080,

$ sudo nano /etc/apache2/sites-available/000-default.conf



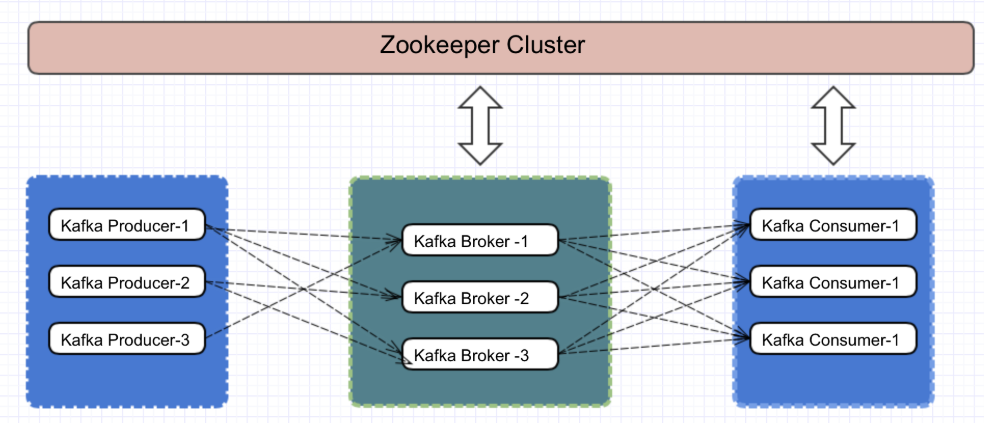
sudo mkdir /var/www/vhosts

sudo mkdir /var/www/vhosts/cutler\_com

sudo systemctl reload apache2

sudo journalctl -u apache2

**Test Kafka Cluster:**



**Zookeeper:**

**+ Fix in templating zoo.cfg**

{% for host in groups['zookeeper\_servers'] %}

server.{{ hostvars[host].myid }}={{ hostvars[host].ansible\_host }}:2888:3888

{% endfor %}

**+ Start zk:**

cd /opt/zookeeper/zk

make sure the server.id is correct in conf/zoo.cfg

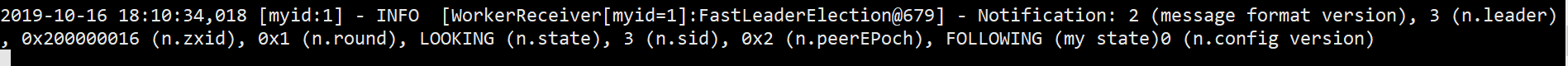
sudo bin/zkServer.sh start-foreground

$ sudo bin/zkServer.sh start (to start at background)

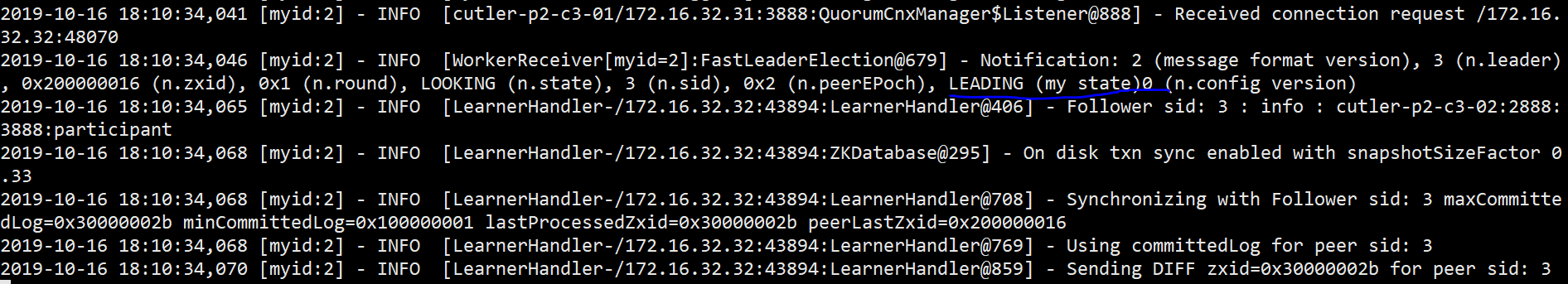
**Make sure all three zookeepers up and running and has correct leader election**

**Server.1:**

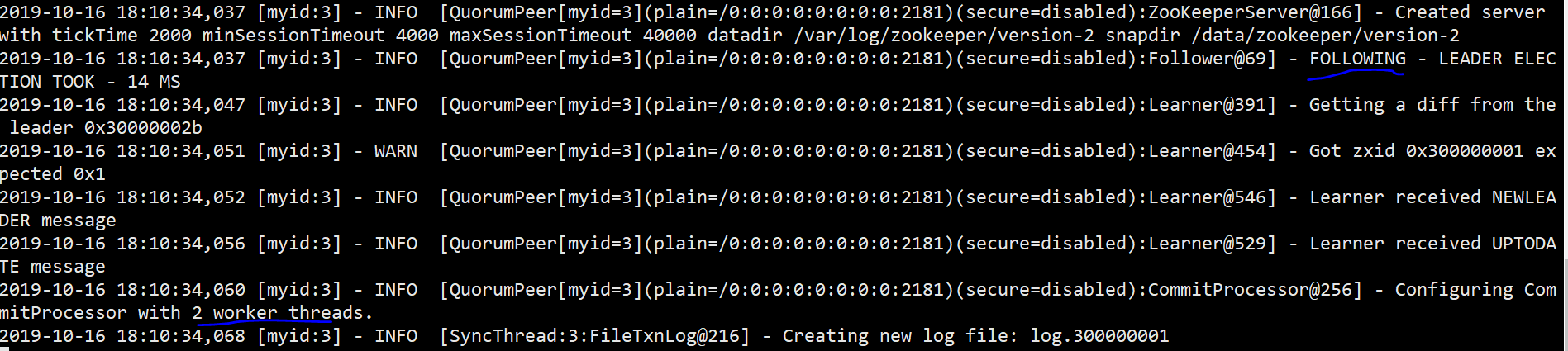




**Server.2: - LEADER**



**Server.3:**



**Kafka:**

Firstly, make sure the config file is correct:

$sudo nano /opt/kafka/config/server-3.properties

broker.id=1-3

listeners=PLAINTEXT://172.16.32.30-32:9092

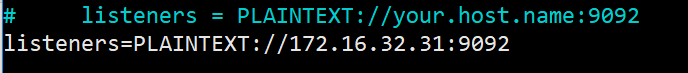
log.dirs=/var/lib/kakfa/data/server-1

If you would like to delete the topics then you need to make edits to delete.topic.enable setting. By default, kafka doesn’t allow you to delete it. It needs to enable in configuration to do it. Please find the line and change it.

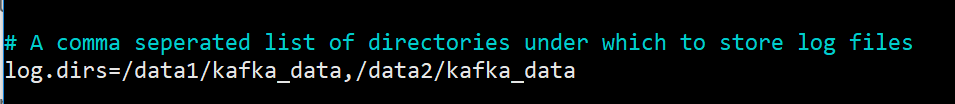
delete.topic.enable = true

Ansible setting currently:

Broker.id is correct.



Change to 0.0.0.0 for localhost interface (lo) as well.



Missing templating for consumer.properties and producer.properties to have fix IP address of the local broker.

Producer/Consumer Basics

bootstrap.servers=172.16.33.30:9092

**+ Start the brokers** (or check if they’re started by default, should be from the ansible setting) (netstat -ant or check systemctl status -u kafka.service)

*$bin/kafka-server-start.sh config/server-1.properties*

*$bin/kafka-server-start.sh config/server-2.properties*

*$bin/kafka-server-start.sh config/server-3.properties*

As in ansible: bin/kafka-server-start.sh -daemon config/server.properties

**+Test the installation:**

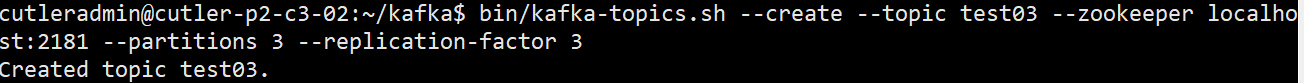
*bin/kafka-topics.sh --create --topic testingtopic --zookeeper localhost:2181 --partitions 3 --replication-factor 3*

partition allows how many brokers you want data to be split. As, we have 3 brokers, we can set this to 3.

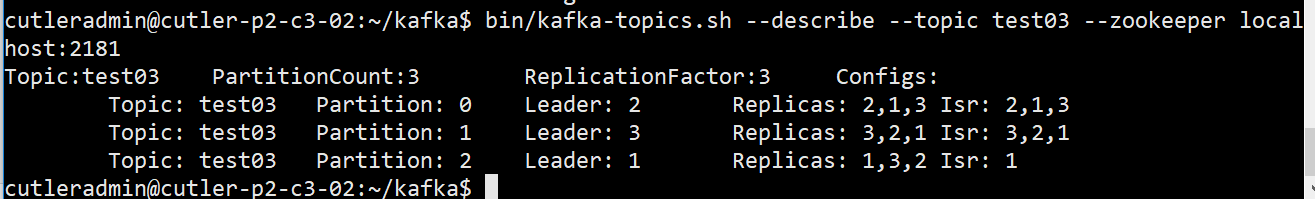
replication factor allows how many copies of data you need. This is helpful when any broker down other brokers can handle the job.

**What we did:**

$ bin/kafka-topics.sh --create --topic test01 --zookeeper localhost:2181 --partitions 3 --replication-factor 3



$ bin/kafka-topics.sh --describe --topic test01 --zookeeper localhost:2181

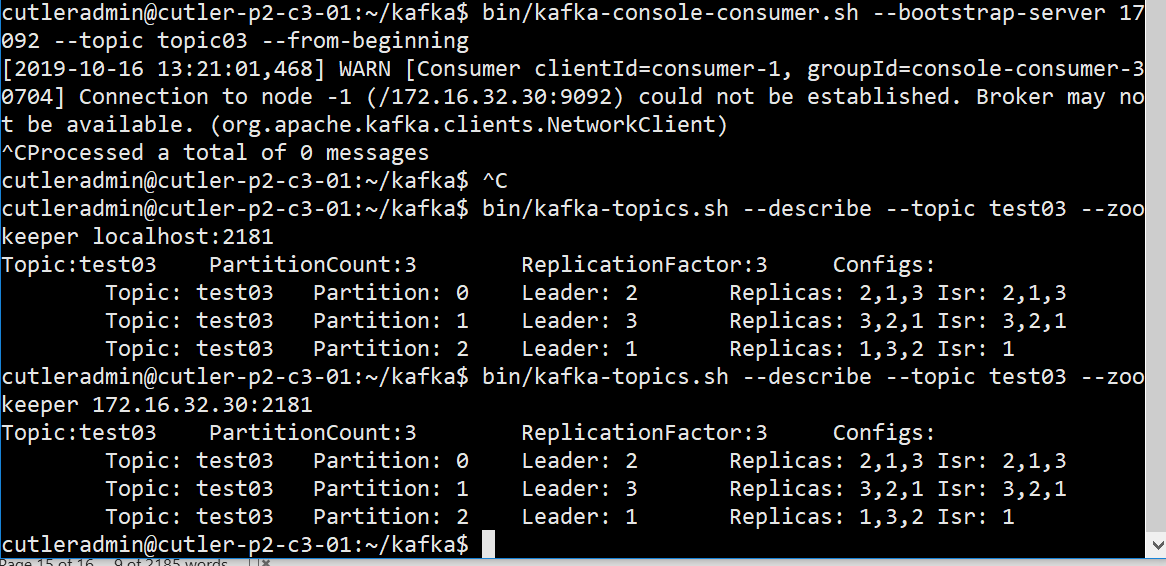


**What we did:**

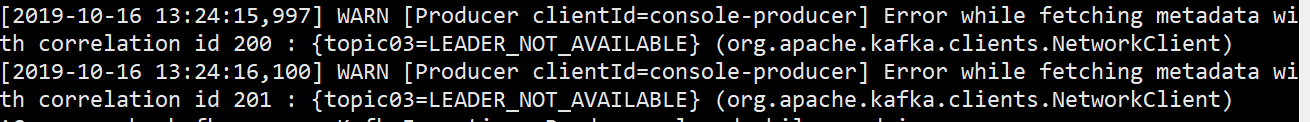
$ bin/kafka-console-producer.sh --broker-list 172.16.32.30:9092,172.16.32.31:9092,172.16.32.32:9092 --topic topic01

$ bin/kafka-console-consumer.sh --bootstrap-server 172.16.32.32:9092 --topic topic01 --from-beginning

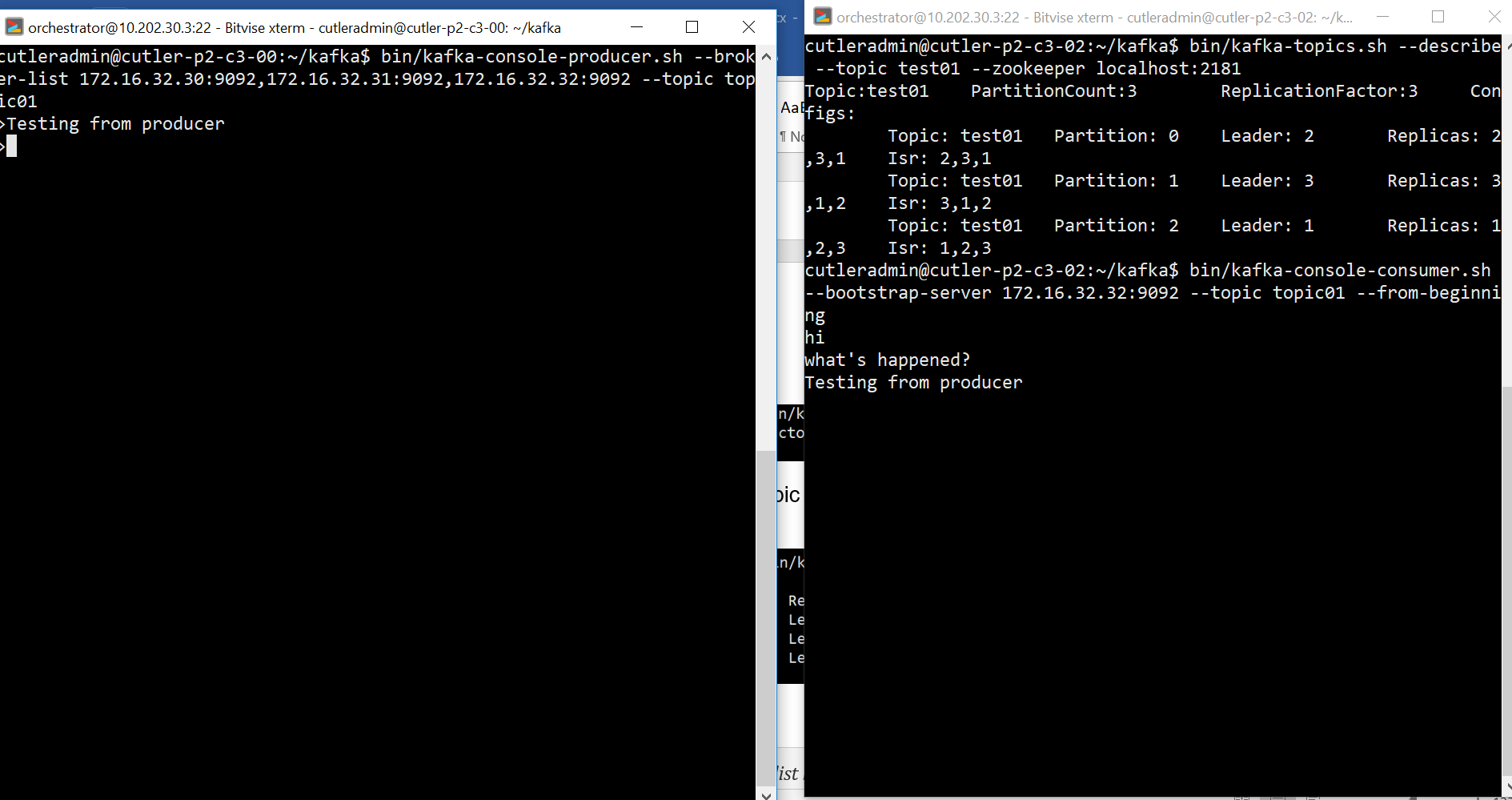
We can talk to the zookeeper but not the kafka on broker-1 172.16.32.30



Also receiving message: LEADER\_NOT\_AVAILABLE - fixed



Working now:



sudo ufw allow 9092

sudo ufw allow 2181

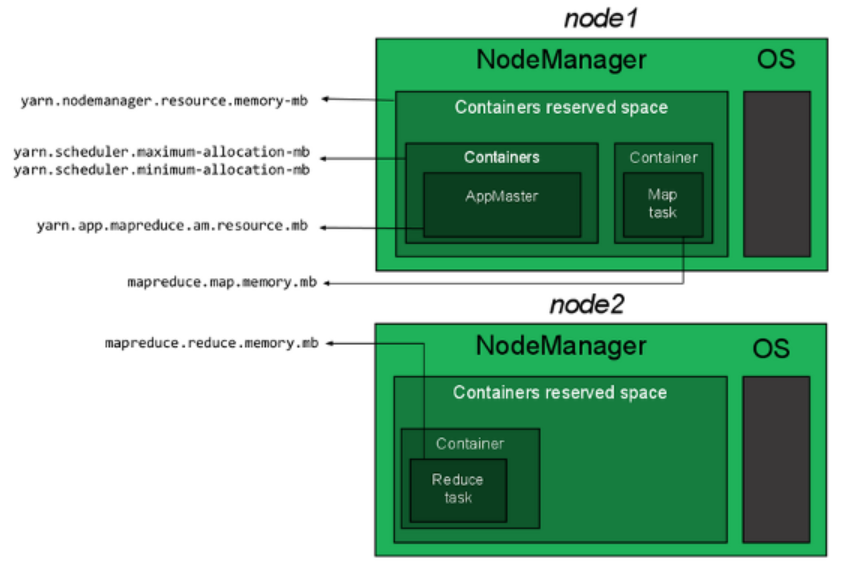
**Testing Hadoop Cluster:**

1. Starting the services:

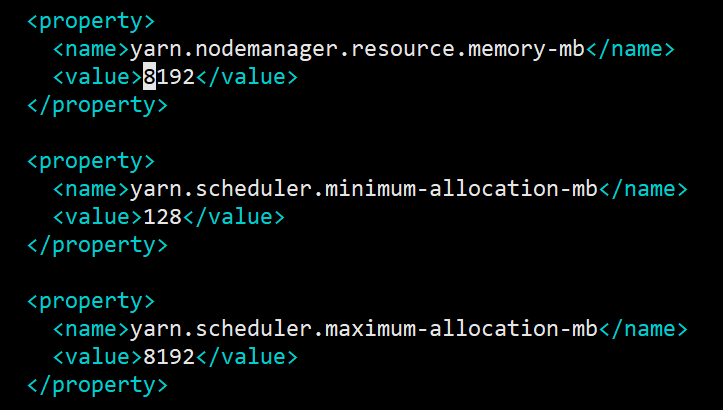
* start-dfs.sh - Starts the Hadoop DFS daemons, the namenode and datanodes. Use this before start-mapred.sh
* stop-dfs.sh - Stops the Hadoop DFS daemons.
* start-mapred.sh - Starts the Hadoop Map/Reduce daemons, the jobtracker and tasktrackers.
* stop-mapred.sh - Stops the Hadoop Map/Reduce daemons.
* start-all.sh - Starts all Hadoop daemons, the namenode, datanodes, the jobtracker and tasktrackers. Deprecated; use start-dfs.sh then start-mapred.sh
* stop-all.sh - Stops all Hadoop daemons. Deprecated; use stop-mapred.sh then stop-dfs.sh
* Alternatively, use hadoop-daemon.sh and yarn-daemon.sh scripts to start the specific HDFS and YARN services respectively on each node.

For configuration of the memory allocation, refer to the below.

* How much memory can be allocated for YARN containers on a single node -> This value is configured in yarn-site.xml with yarn.nodemanager.resource.memory-mb.
* How much memory a single container can consume and the minimum memory allocation allowed. -> Those values are configured in yarn-site.xml with *yarn.scheduler.maximum-allocation-mb* and *yarn.scheduler.minimum-allocation-mb*.
* How much memory will be allocated to the ApplicationMaster. This is a constant value that should fit in the container maximum size. -> This is configured in mapred-site.xml with yarn.app.mapreduce.am.resource.mb.
* How much memory will be allocated to each map or reduce operation. This should be less than the maximum size. ->This is configured in mapred-site.xml with properties mapreduce.map.memory.mb and mapreduce.reduce.memory.mb.



Current setting for 12GB: (could increase the nodemanager & maximum -> 10GB for 12GB core, or 14GB for 16GB core depending on the requirements of jobs)



Open ports: 9866 (datanodes to masternode - The datanode server address and port for data transfer)

Format the disk:

On the name node:

hdfs namenode -format

**Start and Stop HDFS:**

On the NameNode:

start-dfs.sh

start NameNode and SecondaryNameNode on namenode, and DataNode on datanodes in the workers config file.

On namenode:

21922 Jps

21603 NameNode

21787 SecondaryNameNode

And on datanode:

19728 DataNode

19819 Jps

To stop HDFS on master and worker nodes, run the following command from node-master:

stop-dfs.sh

**Monitor your HDFS Cluster**

You can get useful information about running your HDFS cluster with the hdfs dfsadmin command. Try for example:

hdfs dfsadmin -report

This will print information (e.g., capacity and usage) for all running DataNodes. To get the description of all available commands, type:

hdfs dfsadmin -help

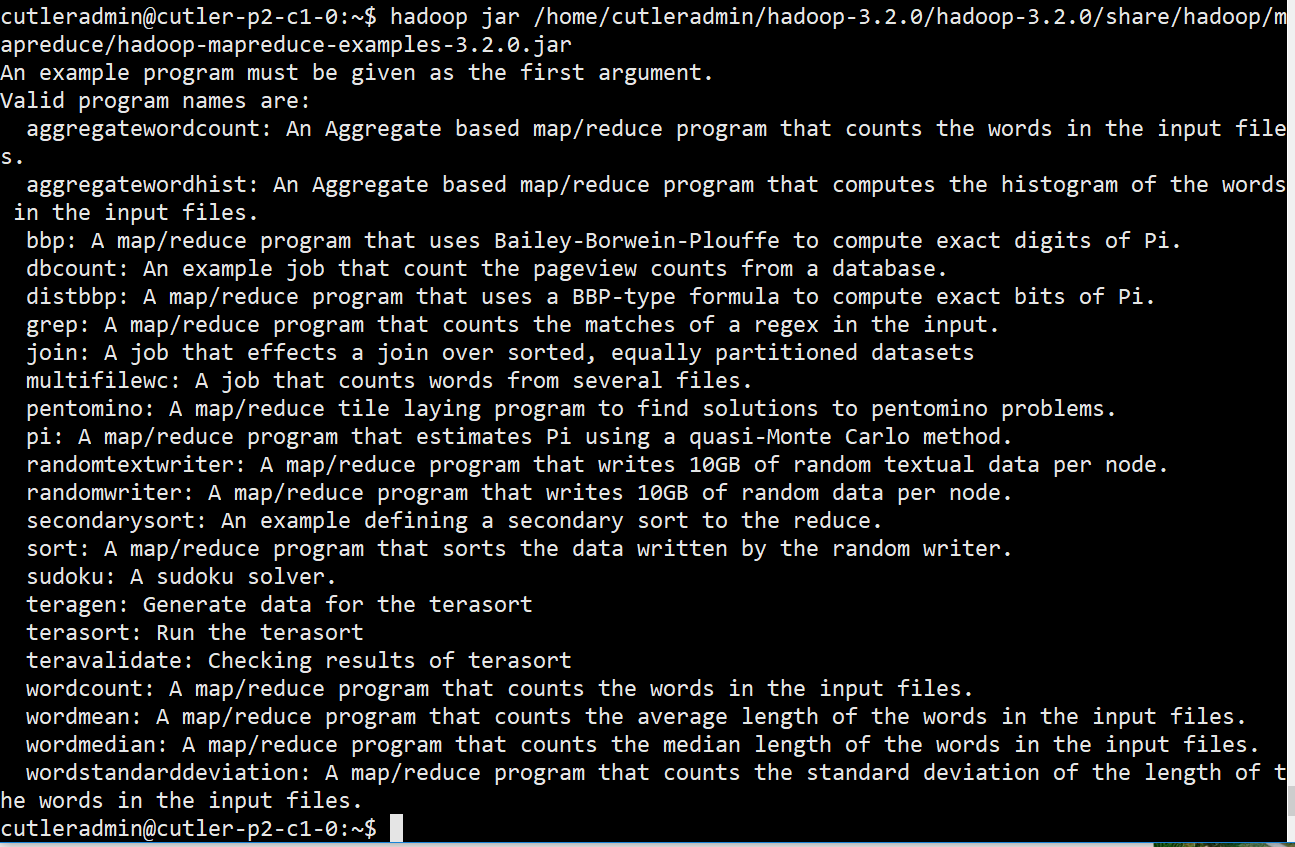
You can also automatically use the friendlier web user interface. Point your browser to http://node-master-IP:9870, where node-master-IP is the IP address of your node-master, and you’ll get a user-friendly monitoring console.

1. Submit a job:
2. List all \*hadoop\*.jar

$ sudo find / -name "\*hadoop\*.jar" | grep example

1. Checking the job example:

$ hadoop jar /home/cutleradmin/hadoop-3.2.0/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.2.0.jar

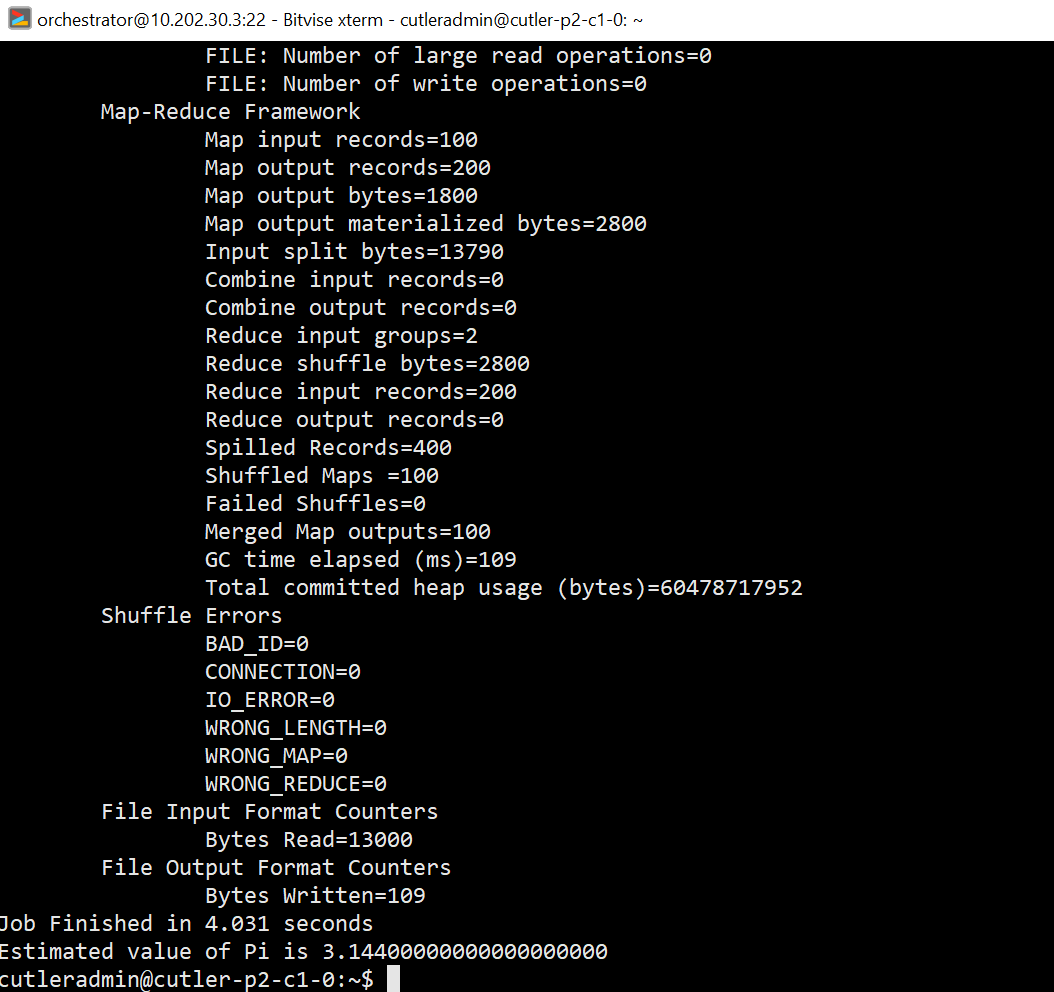


hadoop dfsadmin -safemode leave

1. Submit a job

Estimate Pi, 100 mappers, 20 samples/mapper

$ hadoop jar /home/cutleradmin/hadoop-3.2.0/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.2.0.jar pi 100 20



* Hadoop is working fine.

1. Monitoring

**Testing Yarn RM:**

Make sure you've started Yarn. Use this command to start it:

start-yarn.sh

Then use this command to verify that the Resource Manager is running:

jps

The output should look something like this:

17542 NameNode

17920 SecondaryNameNode

22064 Jps

17703 DataNode

18226 ResourceManager

18363 NodeManager

**Issues:**

Issue# Missing JAVA\_HOME,

sudo nano /etc/profile.d/hadoop-env.sh -> updated

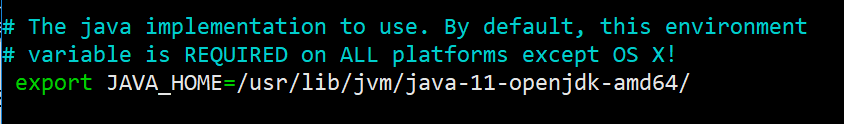
Export in /etc/hadoop/hadoop-env.sh

export JAVA\_HOME=/usr/lib/jvm/<jdk folder>

echo $JAVA\_HOME to see the path.

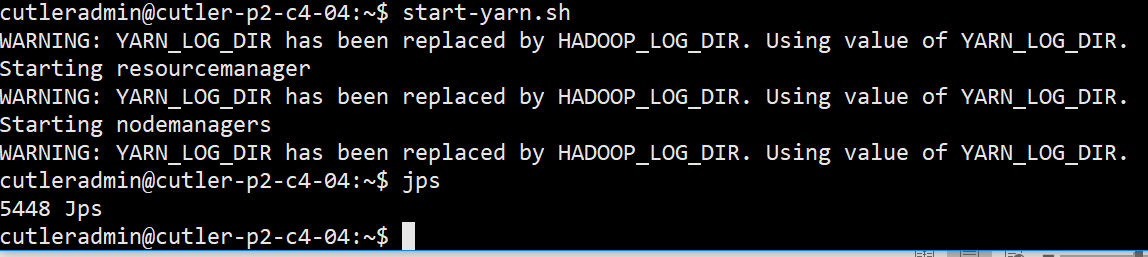
File: /home/cutleradmin/hadoop-3.2.0/hadoop-3.2.0/etc/hadoop/hadoop-env.sh

export JAVA\_HOME=/usr/lib/jvm/java-11-openjdk-amd64/

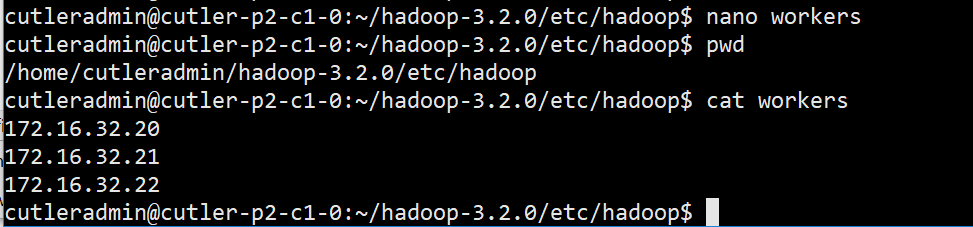


JPS (Java Virtual Machine Process Status Tool) is a command is used to check all the Hadoop daemons

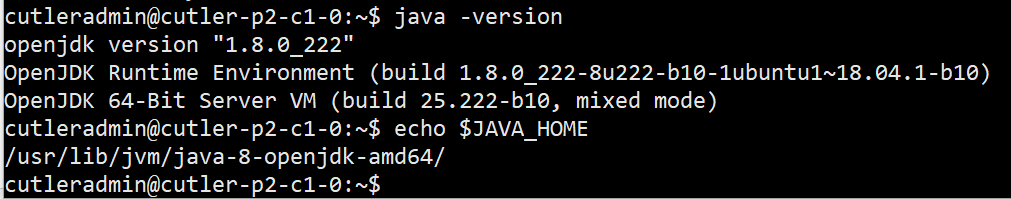
After fixing the java home, issue:



Workers are wrong-defined.



Downgrade to jvm 8:



Added all hosts and IPs in the cluster

sudo nano /etc/hosts

172.16.32.10 cutler-p2-c1-0

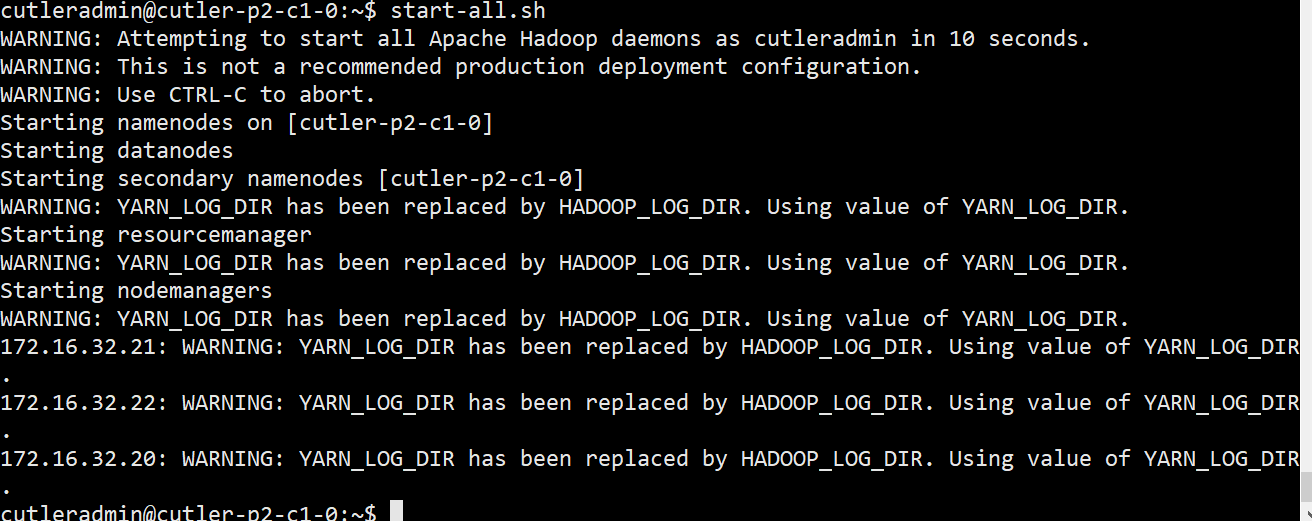
172.16.32.20 cutler-p2-c2-0

172.16.32.21 cutler-p2-c2-1

172.16.32.22 cutler-p2-c2-2

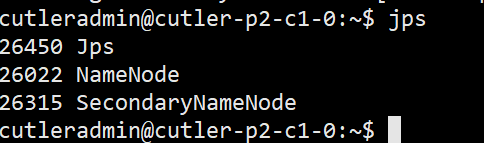
172.16.32.44 cutler-p2-c4-4

After all fixes above, run start-all.sh on the namenode, then check status on each node, all correct:

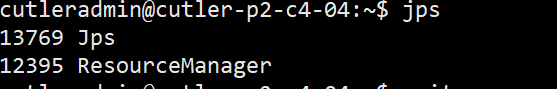


**Status: looks good**

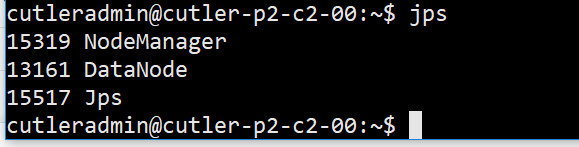
NameNode:



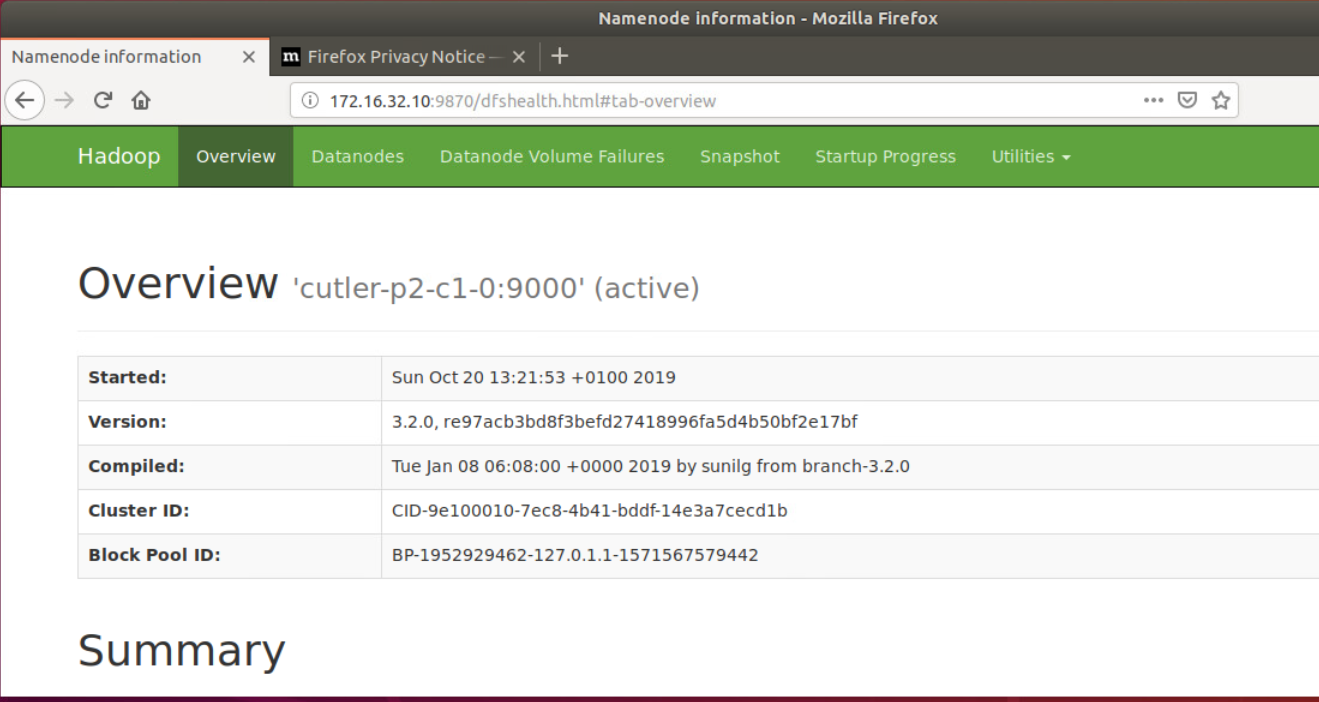
ResourceNode:

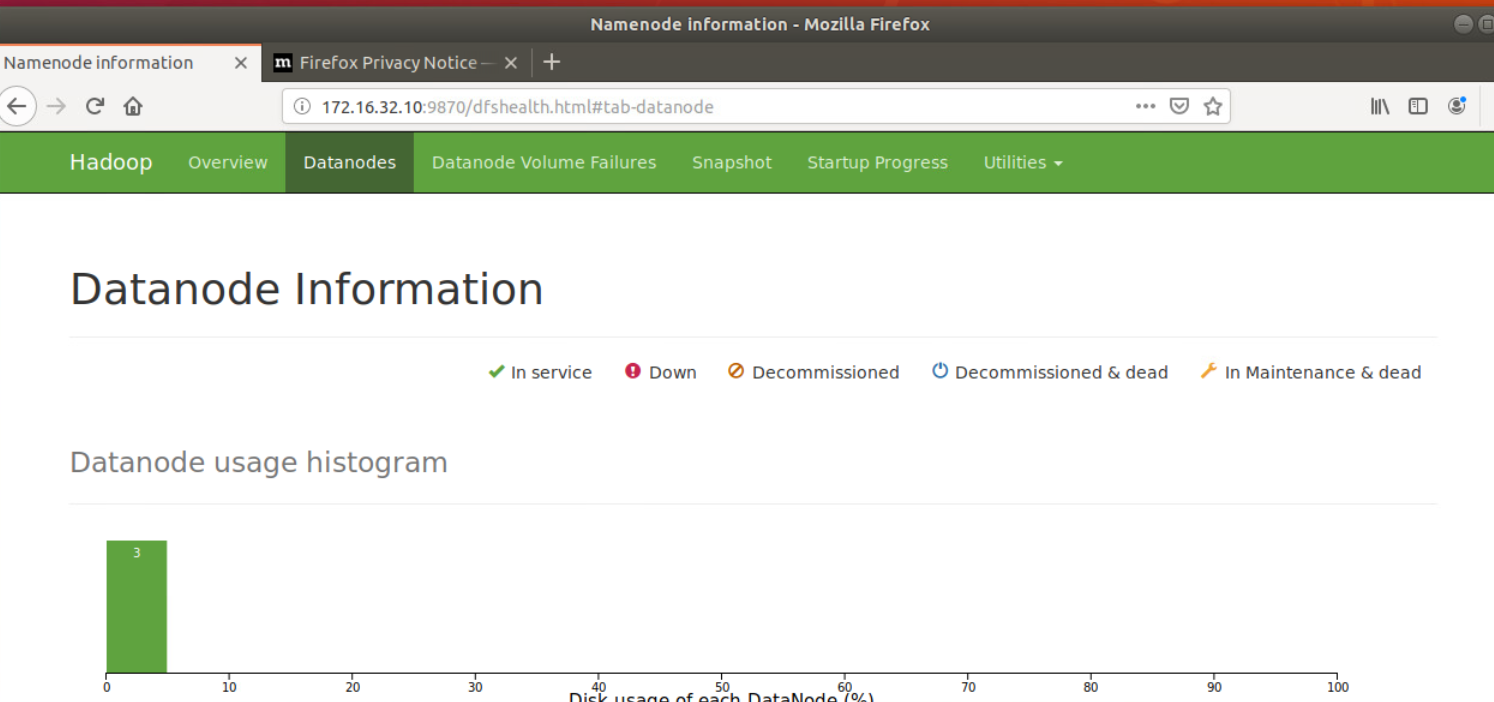


DataNode:

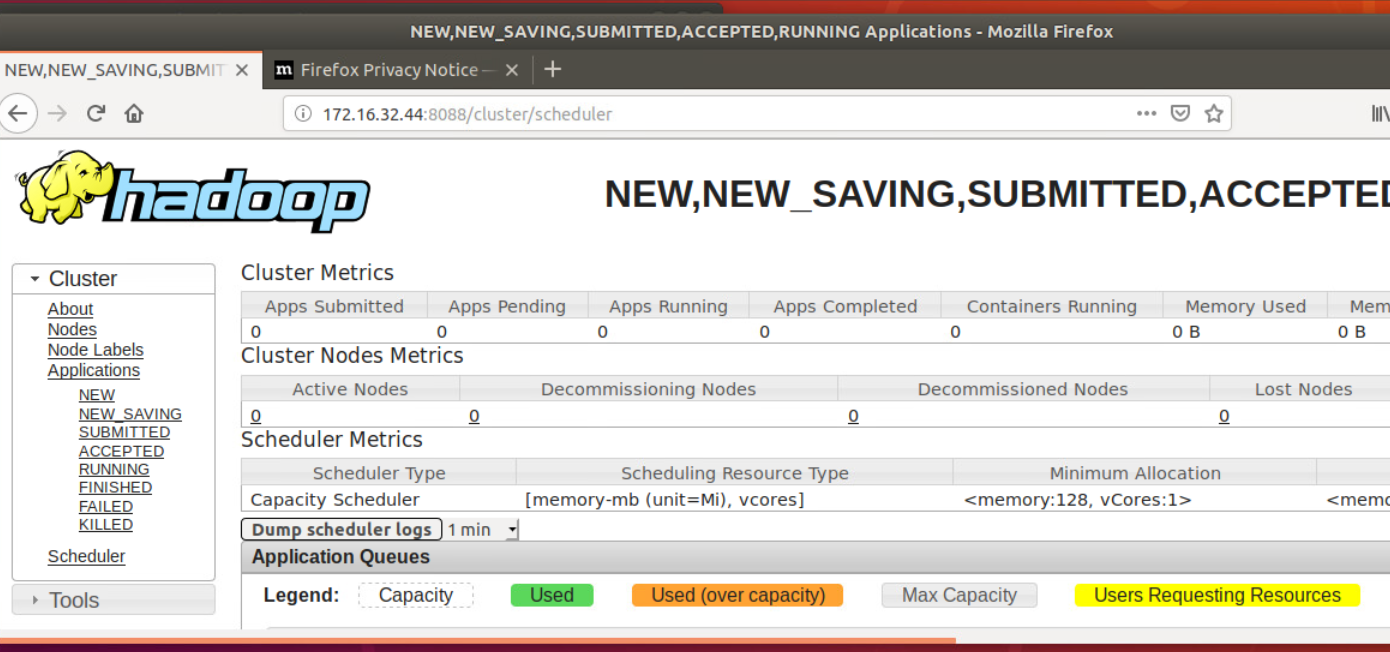


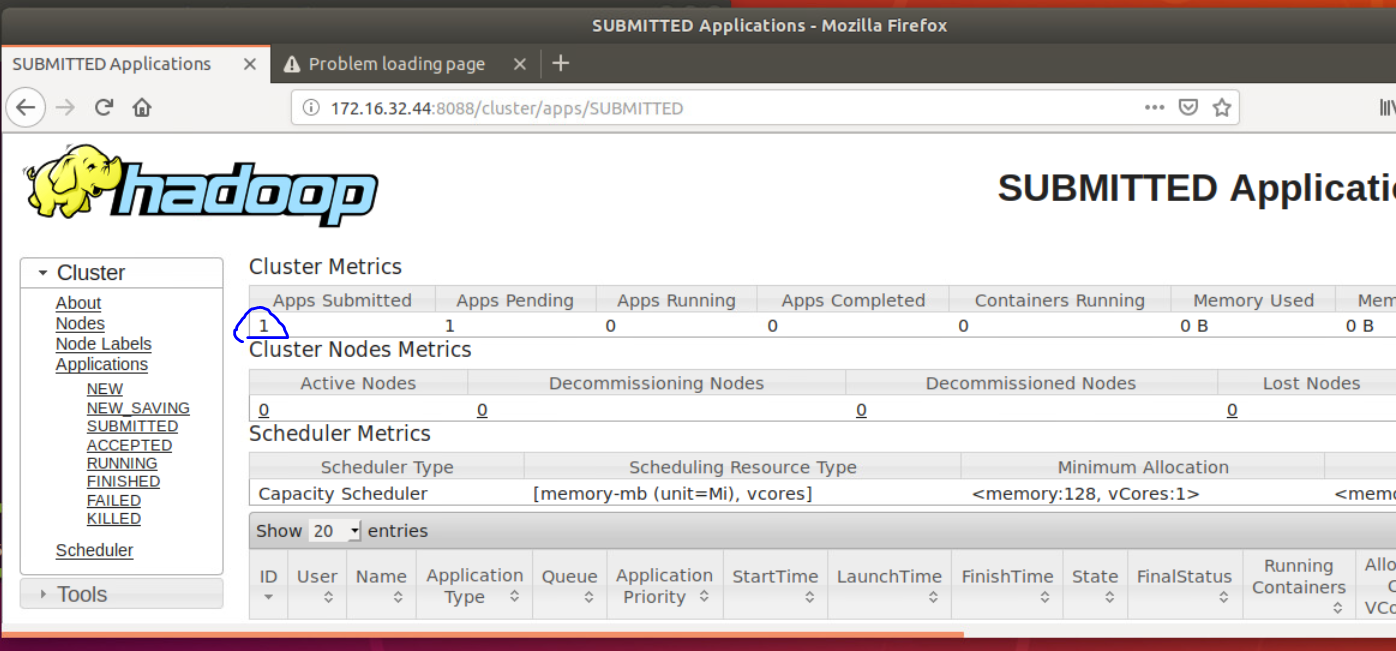
**Hadoop NameNode Web UI:**

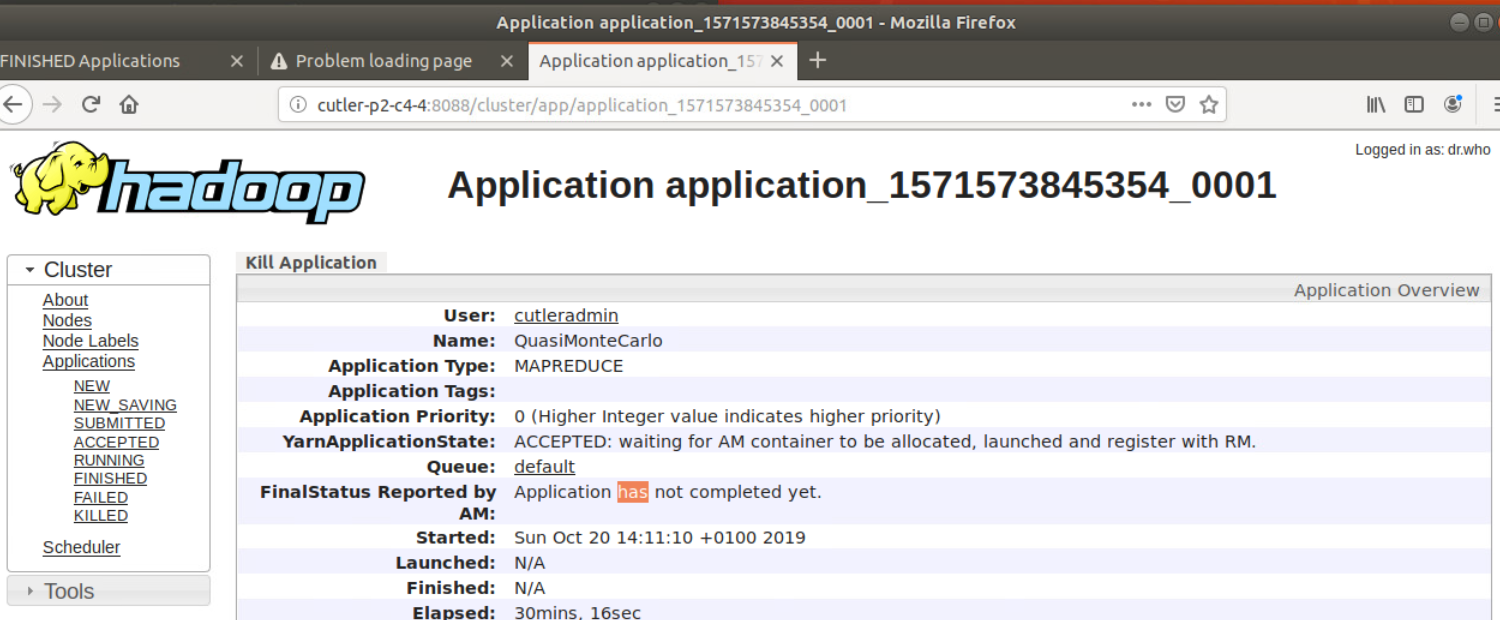




Resource Manager web UI:







**Test Spark:**

sudo nano /etc/profile.d/spark.sh

export PATH=$PATH:/opt/spark/bin

export SPARK\_HOME=/opt/spark

export HADOOP\_CONF\_DIR=$HADOOP\_HOME/etc/hadoop

export LD\_LIBRARY\_PATH=$LD\_LIBRARY\_PATH:$HADOOP\_HOME/hadoop/lib/native

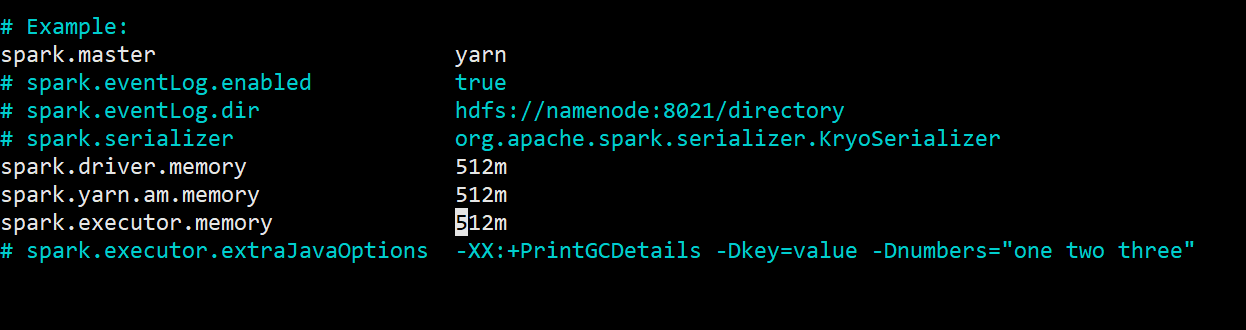
export SPARK\_DIST\_CLASSPATH=$(hadoop classpath)

*# Passing a Hadoop configuration directory*

#export SPARK\_DIST\_CLASSPATH=**$(**hadoop --config /path/to/configs classpath**)**

sudo mv /opt/spark/conf/spark-defaults.conf.template /opt/spark/conf/spark-defaults.conf

sudo nano /opt/spark/conf/spark-defaults.conf

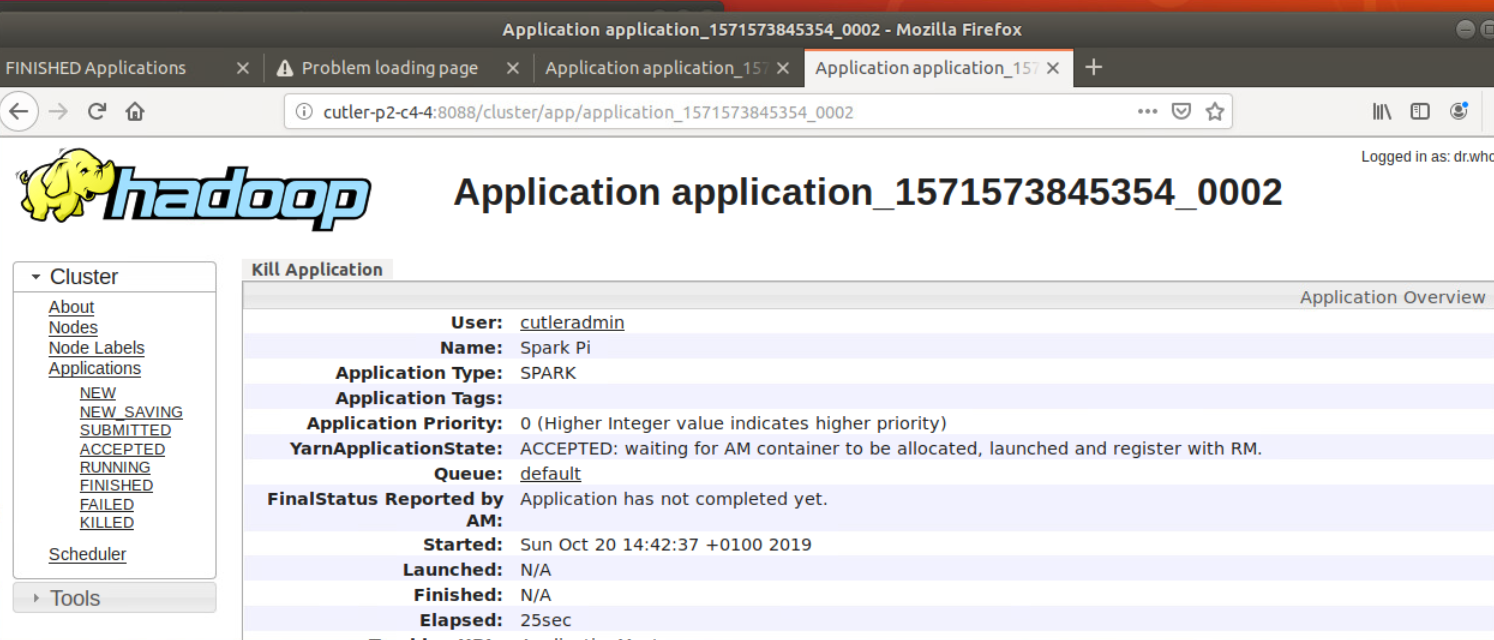


**Submitting a Spark job example:**

Run the sample Pi calculation, use the following command (at cutleradmin home directory):

$ spark-submit --deploy-mode client --class org.apache.spark.examples.SparkPi /opt/spark/examples/jars/spark-examples\_2.11-2.4.3.jar

*--deploy-mode*, specifies which mode to use, *client* or *cluster*.



**Testing Front-End server:**

# Automatically generated for Debian scripts. DO NOT TOUCH!

[client]

host = localhost

user = debian-sys-maint

password = s0cu4gnic8ZXhlXM

socket = /var/run/mysqld/mysqld.sock

[mysql\_upgrade]

host = localhost

user = debian-sys-maint

password = s0cu4gnic8ZXhlXM

socket = /var/run/mysqld/mysqld.sock

