BUILDING HADOOP CLUSTER USING ANSIBLE

About Ansible:

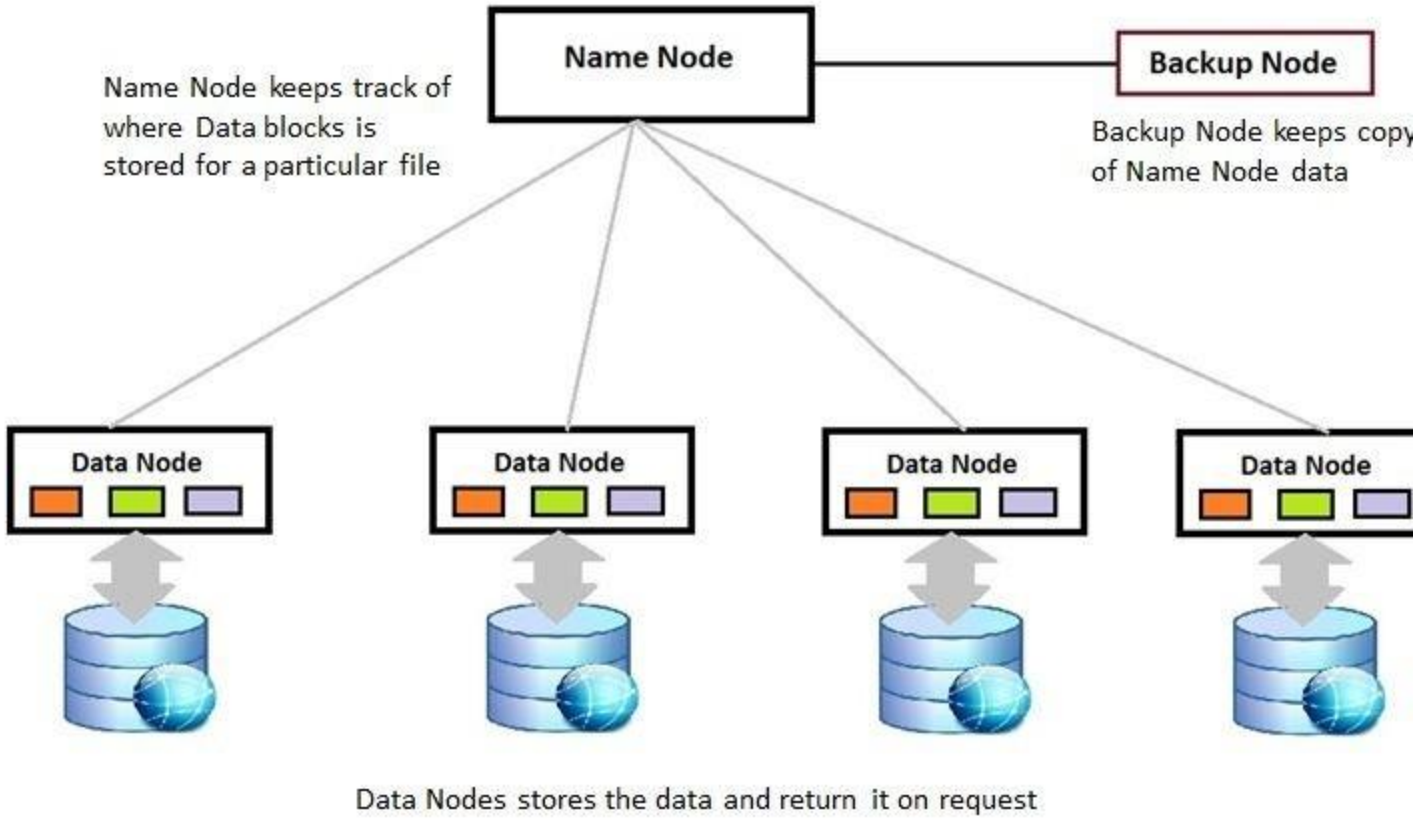
Ansible is an open-source software provisioning, configuration management, and application-deployment tool enabling infrastructure as code. It runs on many Unix-like systems, and can configure both Unix-like systems as well as Microsoft Windows. It includes its own declarative language to describe system configuration. Ansible was written by Michael DeHaan and acquired by Red Hat in 2015. Ansible is agentless, temporarily connecting remotely via SSH or Windows Remote Management (allowing remote PowerShell execution) to do its tasks.Ansible playbook:

Apache Hadoop:

The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Rather than rely on hardware to deliver high-availability, the library itself is designed to detect and handle failures at the application layer, so delivering a highly-available service on top of a cluster of computers, each of which may be prone to failures..Hadoop is an open-source software framework for storing data and running applications on clusters of commodity hardware. It provides massive storage for any kind of data, enormous processing power and the ability to handle virtually limitless concurrent tasks or jobs.

Hadoop clusters:

In talking about Hadoop clusters, first we need to define two terms: cluster and node. A cluster is a collection of nodes. A node is a process running on a virtual or physical machine or in a container. We say process because a code would be running other programs beside Hadoop.When Hadoop is not running in cluster mode, it is said to be running in local mode. That would be suitable for, say, installing Hadoop on one machine just to learn it. When you run Hadoop in local node it writes data to the local file system instead of HDFS (Hadoop Distributed File System).Hadoop is a master-slave model, with one master (albeit with an optional High Availability hot standby) coordinating the role of many slaves. Yarn is the resource manager that coordinates what task runs where, keeping in mind available CPU, memory, network bandwidth, and storage.One can scale out a Hadoop cluster, which means add more nodes. Hadoop is said to be linearly scalable. That means for every node you add you get a corresponding boost in throughput. More generally if you have n nodes then adding 1 mode give you (1/n) additional computing power. That type of distributed computing is a major shift from the days of using a single server where when you add memory and CPUs it produces only a marginal increase in throughout.


Name Node
Name Node keeps track of where Data blocks is stored for a particular file
Backup Node
Backup Node keeps copy of Name Node data
Data Node
Data Node
Data Node
Data Node
Data Nodes stores the data and return it on request

Ansible playbook:

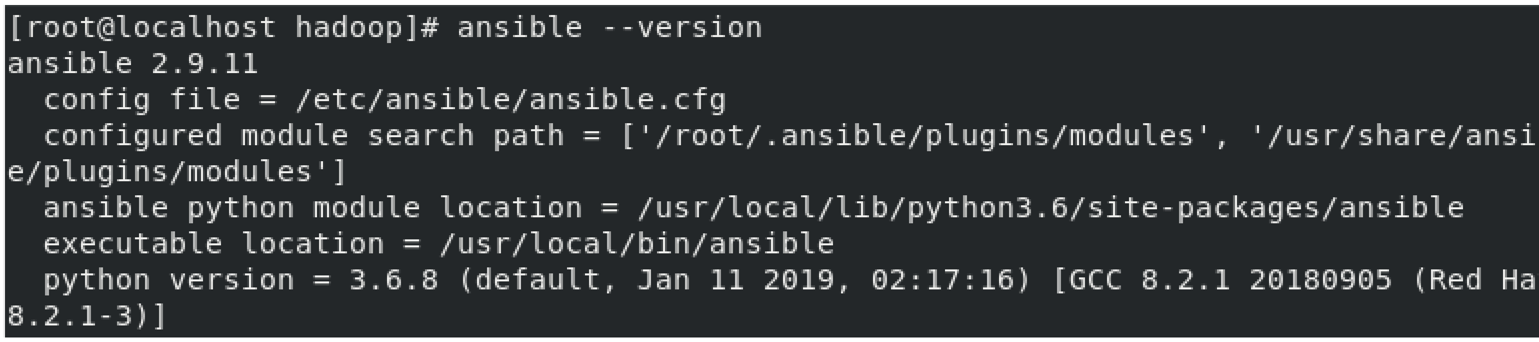
An Ansible playbook is an organized unit of scripts that defines work for a server configuration managed by the automation tool Ansible. Ansible is a configuration management tool that automates the configuration of multiple servers by the use of Ansible playbooks.

Pre-requisites:

~ Installation of Ansible:

As Ansible is made on top of python language ,so we have to install ansible using following command: # pip3 install ansible

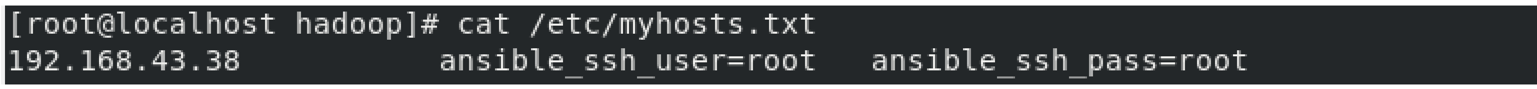
for checking version of installed ansible tool we can use following command.



```
[root@localhost hadoop]# ansible --version
ansible 2.9.11
  config file = /etc/ansible/ansible.cfg
  configured module search path = ['/root/.ansible/plugins/modules', '/usr/share/ansi
e/plugins/modules']
  ansible python module location = /usr/local/lib/python3.6/site-packages/ansible
  executable location = /usr/local/bin/ansible
  python version = 3.6.8 (default, Jan 11 2019, 02:17:16) [GCC 8.2.1 20180905 (Red Ha
8.2.1-3)]
```

~ Configuration of Ansible:

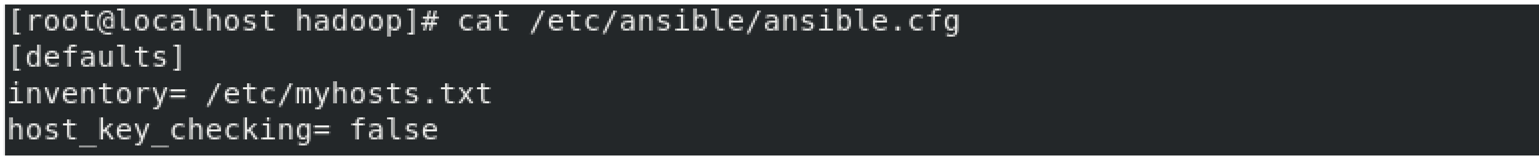
Next, we have to create one text file at /etc/myhost.txt which store IP of managed node along with username and password.



```
[root@localhost hadoop]# cat /etc/myhosts.txt
192.168.43.38           ansible_ssh_user=root   ansible_ssh_pass=root
```

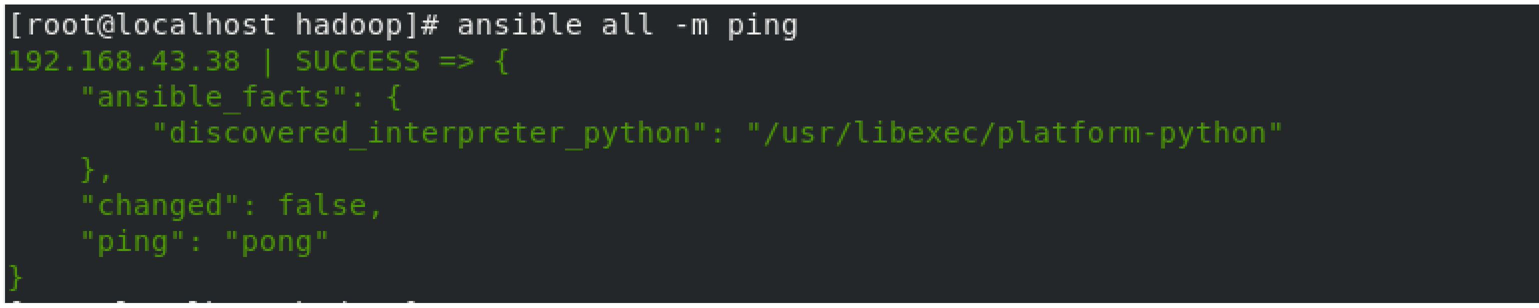
Ansible works against multiple systems in your infrastructure at the same time. It does this by selecting portions of the systems listed in Ansible's inventory file. You can specify a different inventory file using the -i <path> option on the command line. Not only is this inventory configurable, but you can also use multiple inventory files at the same time.

To check list of all host use following command:



```
[root@localhost hadoop]# cat /etc/ansible/ansible.cfg
[defaults]
inventory= /etc/myhosts.txt
host_key_checking= false
```

We need to check, that all managed node is properly connected to the controller node or not. For that use the following command.



```
[root@localhost hadoop]# ansible all -m ping
192.168.43.38 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false,
    "ping": "pong"
}
```

....

Creating Hadoop Cluster:

Typically one machine in the cluster is designated as the NameNode and another machine the as *Data Node*, exclusively. These are the *masters*. The rest of the machines in the cluster act as both DataNode *and* TaskTracker. These are the *slaves*.

Here we creating ansible playbook's for configure **Data Nodes.**

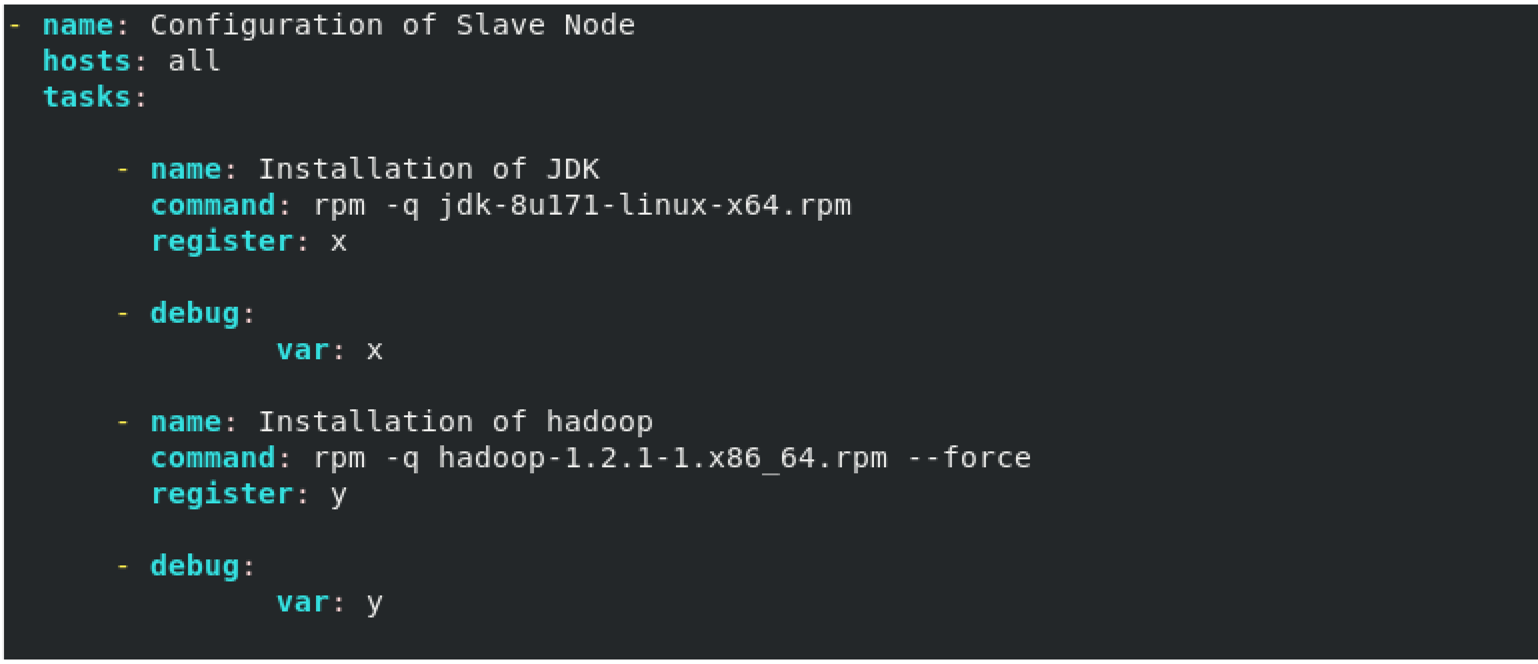
....

1. Installation of Hadoop package:

It's now time to install Hadoop. Most important part is to install hadoop in OS we requires JDK . Hence first we install JDK and then we install hadoop. Typically one machine in the cluster is designated as the NameNode and another machine the as JobTracker, exclusively. These are the *masters*. The rest of the machines in the cluster act as both DataNode *and* TaskTracker. These are the *slaves*.

For that create one yml playbook named as *hadoop.yml* . Here we use **Ansible command module** to install jdk using rpm commnad and then we install hadoop package using rpm command.

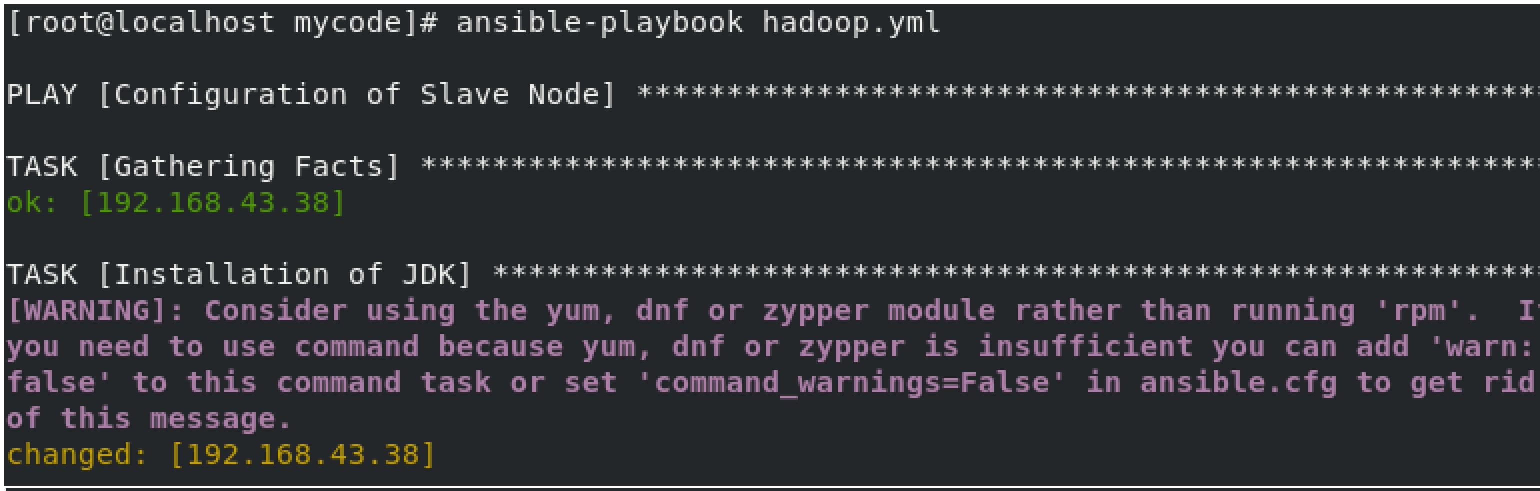
# vim hadoop.yml



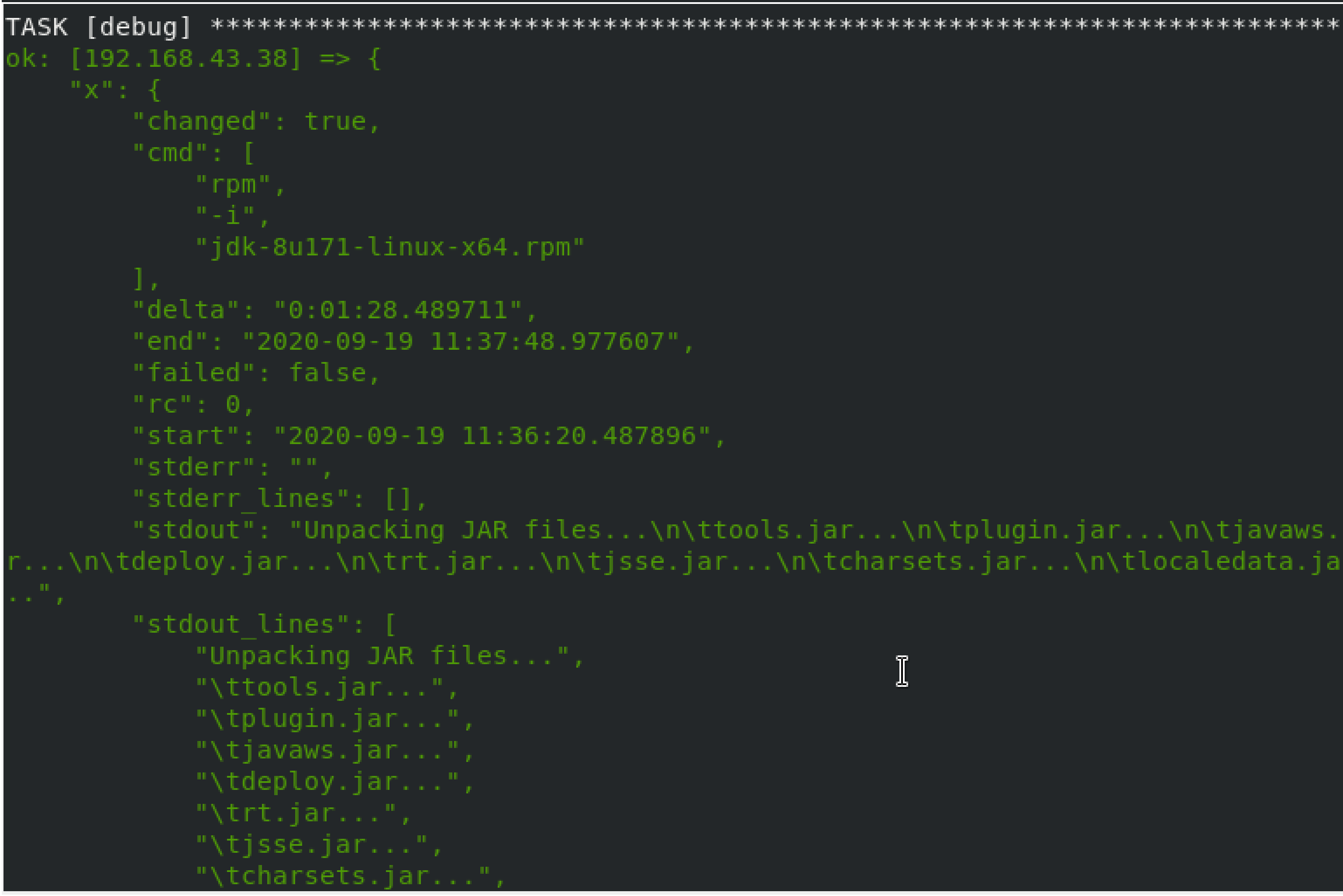
```
- name: Configuration of Slave Node
  hosts: all
  tasks:

      - name: Installation of JDK
        command: rpm -q jdk-8u171-linux-x64.rpm
        register: x

      - debug:
               var: x

      - name: Installation of hadoop
        command: rpm -q hadoop-1.2.1-1.x86_64.rpm --force
        register: y

      - debug:
               var: y
```

Then run the ansible playbook command for hadoop.yml

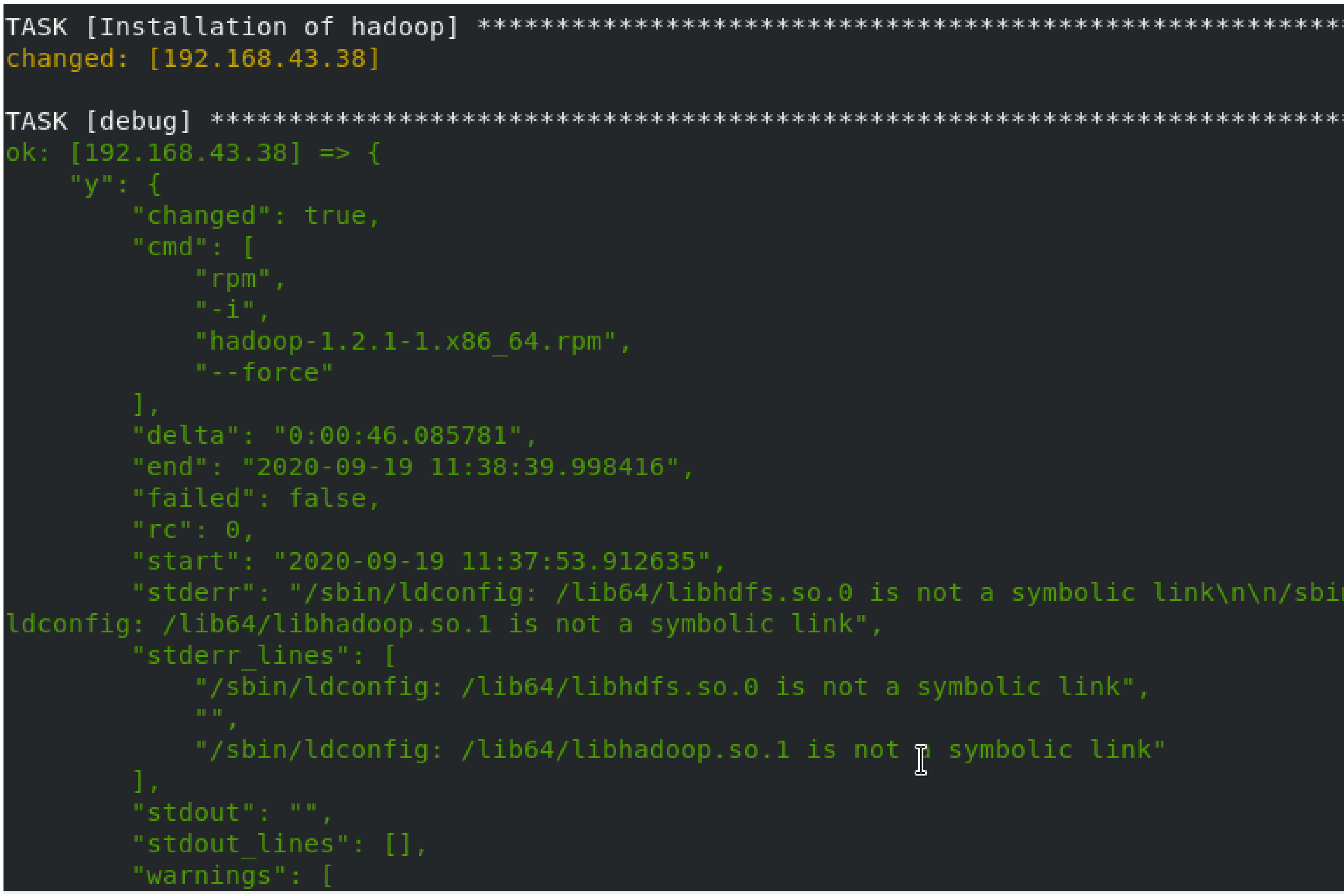
# ansible-playbook hadoop.yml



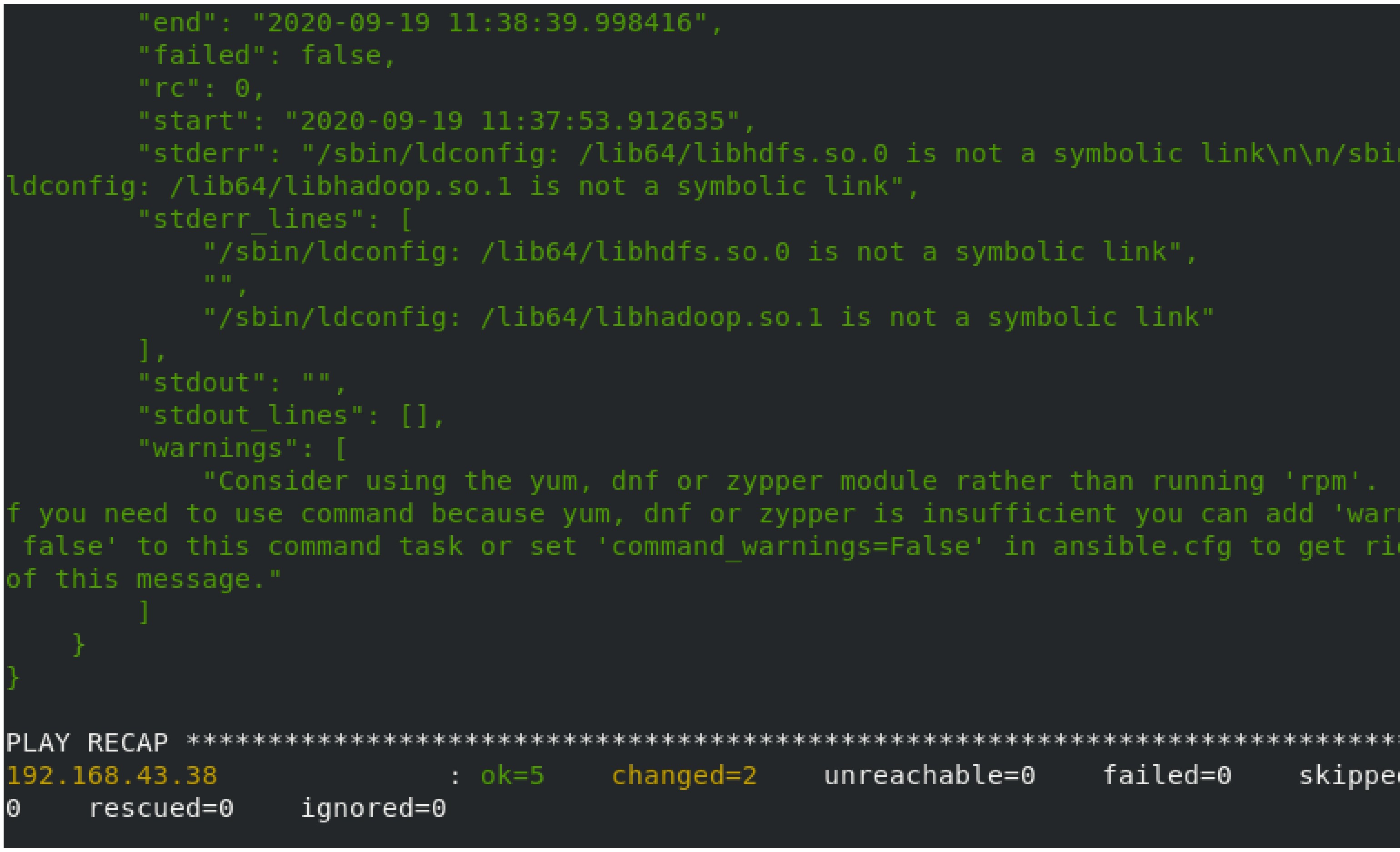
```
[root@localhost mycode]# ansible-playbook hadoop.yml

PLAY [Configuration of Slave Node] *****************************************************

TASK [Gathering Facts] *****************************************************************
ok: [192.168.43.38]

TASK [Installation of JDK] *************************************************************
[WARNING]: Consider using the yum, dnf or zypper module rather than running 'rpm'.  I
you need to use command because yum, dnf or zypper is insufficient you can add 'warn:
false' to this command task or set 'command_warnings=False' in ansible.cfg to get rid
of this message.
changed: [192.168.43.38]
```



```
TASK [debug] ***************************************************************************
ok: [192.168.43.38] => {
    "x": {
        "changed": true,
        "cmd": [
            "rpm",
            "-i",
            "jdk-8u171-linux-x64.rpm"
        ],
        "delta": "0:01:28.489711",
        "end": "2020-09-19 11:37:48.977607",
        "failed": false,
        "rc": 0,
        "start": "2020-09-19 11:36:20.487896",
        "stderr": "",
        "stderr_lines": [],
        "stdout": "Unpacking JAR files...\n\ttools.jar...\n\tplugin.jar...\n\tjavaws.
r...\n\tdeploy.jar...\n\trt.jar...\n\tjsse.jar...\n\tcharsets.jar...\n\tlocaledata.ja
..",
        "stdout_lines": [
            "Unpacking JAR files...",
            "\ttools.jar...",
            "\tplugin.jar...",
            "\tjavaws.jar...",
            "\tdeploy.jar...",
            "\trt.jar...",
            "\tjsse.jar...",
            "\tcharsets.jar...",
```



```
TASK [Installation of hadoop] ****************************************************
changed: [192.168.43.38]

TASK [debug] *********************************************************************
ok: [192.168.43.38] => {
    "y": {
        "changed": true,
        "cmd": [
            "rpm",
            "-i",
            "hadoop-1.2.1-1.x86_64.rpm",
            "--force"
        ],
        "delta": "0:00:46.085781",
        "end": "2020-09-19 11:38:39.998416",
        "failed": false,
        "rc": 0,
        "start": "2020-09-19 11:37:53.912635",
        "stderr": "/sbin/ldconfig: /lib64/libhdfs.so.0 is not a symbolic link\n\n/sbi
ldconfig: /lib64/libhadoop.so.1 is not a symbolic link",
        "stderr_lines": [
            "/sbin/ldconfig: /lib64/libhdfs.so.0 is not a symbolic link",
            "",
            "/sbin/ldconfig: /lib64/libhadoop.so.1 is not a symbolic link"
        ],
        "stdout": "",
        "stdout_lines": [],
        "warnings": [
```



```
        "end": "2020-09-19 11:38:39.998416",
        "failed": false,
        "rc": 0,
        "start": "2020-09-19 11:37:53.912635",
        "stderr": "/sbin/ldconfig: /lib64/libhdfs.so.0 is not a symbolic link\n\n/sbi
ldconfig: /lib64/libhadoop.so.1 is not a symbolic link",
        "stderr_lines": [
            "/sbin/ldconfig: /lib64/libhdfs.so.0 is not a symbolic link",
            "",
            "/sbin/ldconfig: /lib64/libhadoop.so.1 is not a symbolic link"
        ],
        "stdout": "",
        "stdout_lines": [],
        "warnings": [
            "Consider using the yum, dnf or zypper module rather than running 'rpm'.
f you need to use command because yum, dnf or zypper is insufficient you can add 'war
 false' to this command task or set 'command_warnings=False' in ansible.cfg to get ri
of this message."
        ]
    }
}

PLAY RECAP ***********************************************************************
192.168.43.38              : ok=5    changed=2    unreachable=0    failed=0    skippe
0    rescued=0    ignored=0
```

Hence JDK and Hadoop installed successfully.

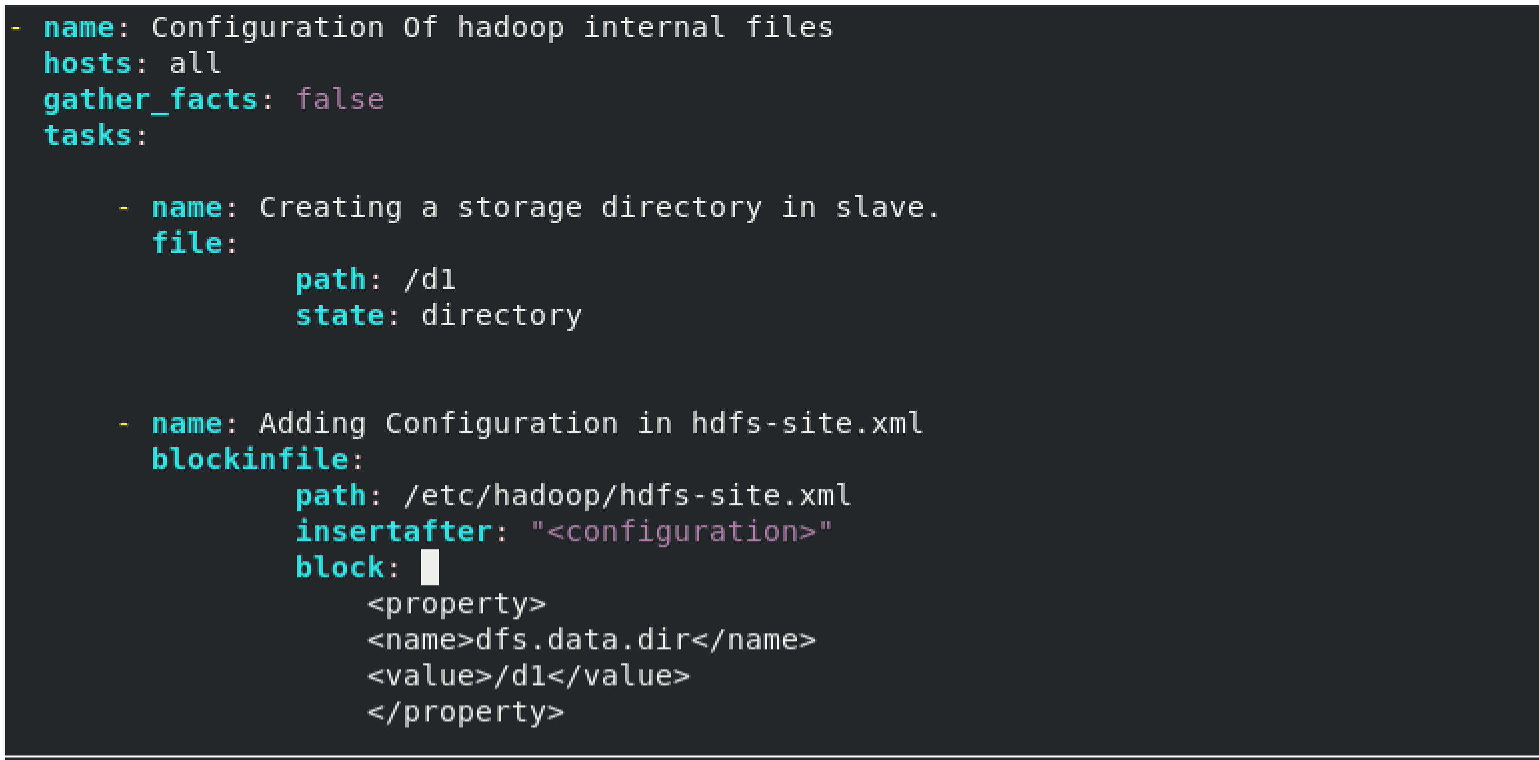
....

2. Configuration of Hadoop files:

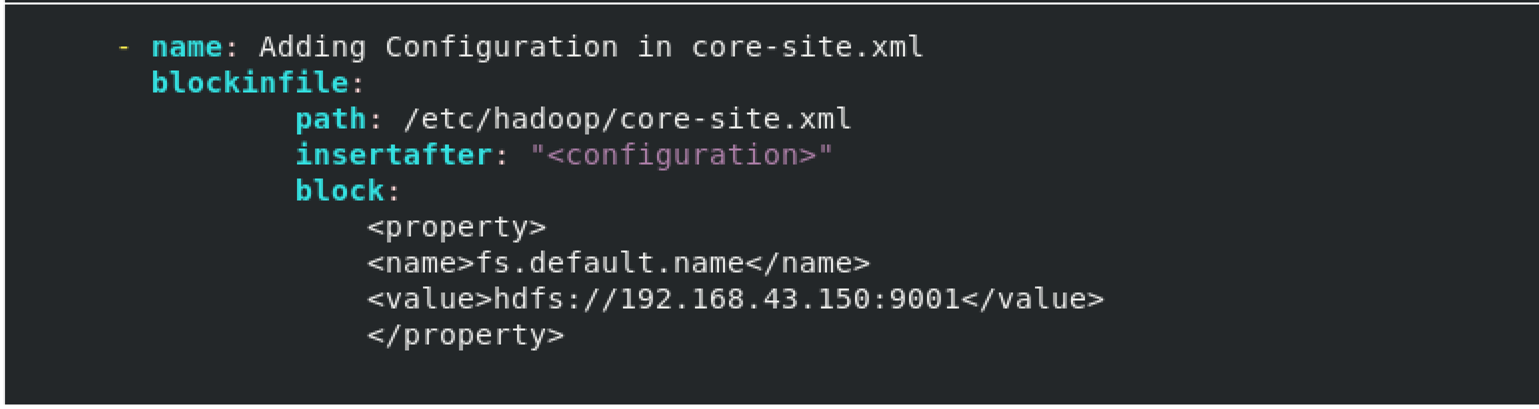
First we Hadoop configuration is driven by two types of important configuration files:Site-specific configuration - *conf/core-site.xml*, *conf/hdfs-site.xml* To configure the Hadoop cluster you will need to configure the *environment* in which the Hadoop daemons execute as well as the *configuration parameters* for the Hadoop daemons. Hence to share storage from data node to master node we have to create directory of storage.

For that we create had1.yml playbook to add block in */etc/hadoop/* internal files. For that we use **Ansible Blockinfile Module**.Also to share storage from data node to master node we have to create directory of storage(we use **file module**).

# vim had1.yml



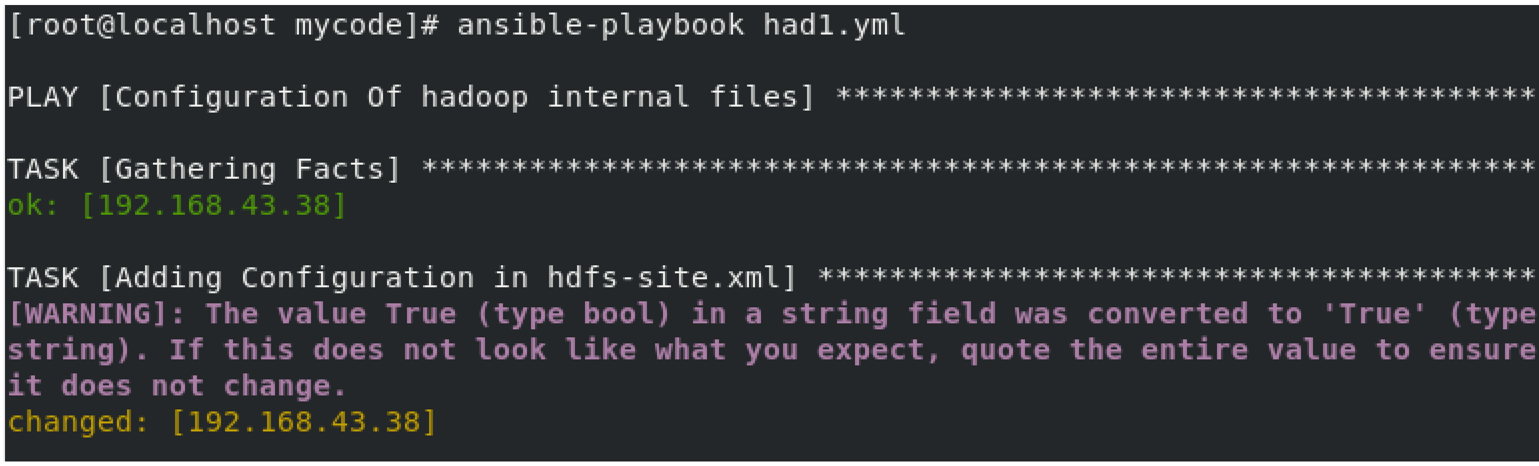
```
- name: Configuration Of hadoop internal files
  hosts: all
  gather_facts: false
  tasks:

      - name: Creating a storage directory in slave.
        file:
                path: /d1
                state: directory


      - name: Adding Configuration in hdfs-site.xml
        blockinfile:
                path: /etc/hadoop/hdfs-site.xml
                insertafter: "<configuration>"
                block: 
                    <property>
                    <name>dfs.data.dir</name>
                    <value>/d1</value>
                    </property>
```



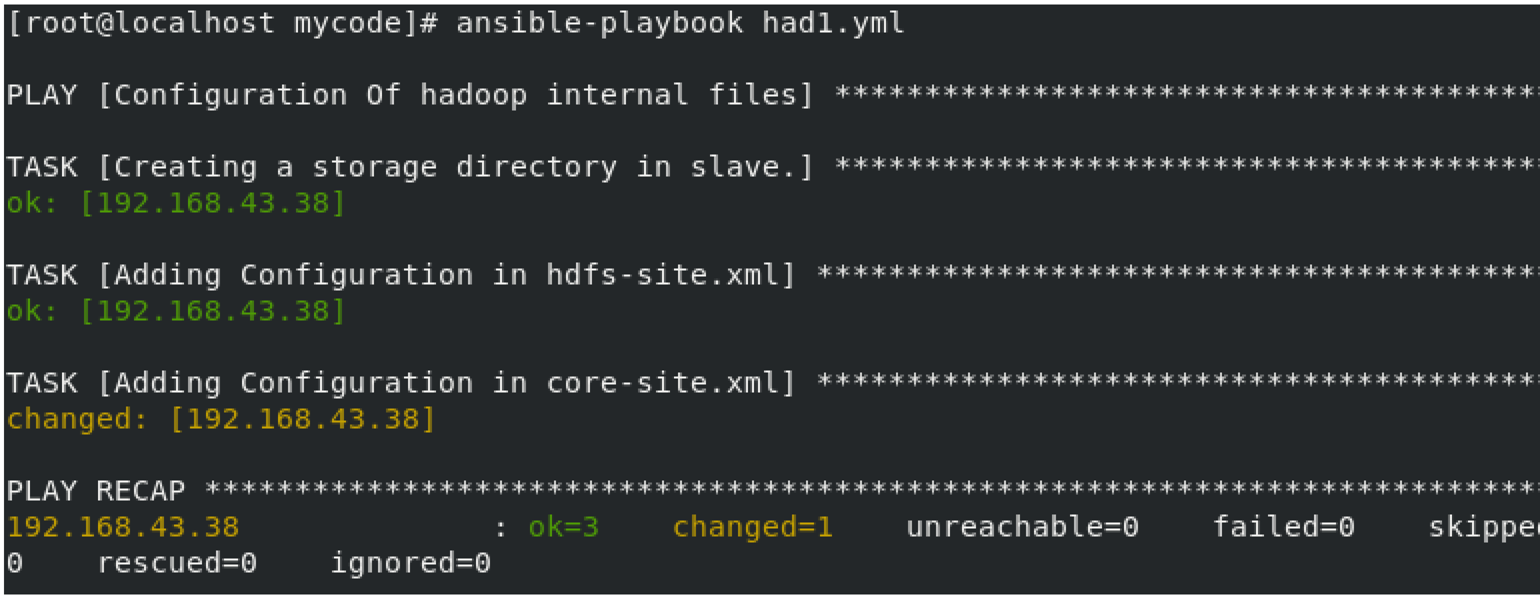
```
      - name: Adding Configuration in core-site.xml
        blockinfile:
                path: /etc/hadoop/core-site.xml
                insertafter: "<configuration>"
                block:
                    <property>
                    <name>fs.default.name</name>
                    <value>hdfs://192.168.43.150:9001</value>
                    </property>
```

............run the playbook............

```
# ansible-playbook had1.yml
```



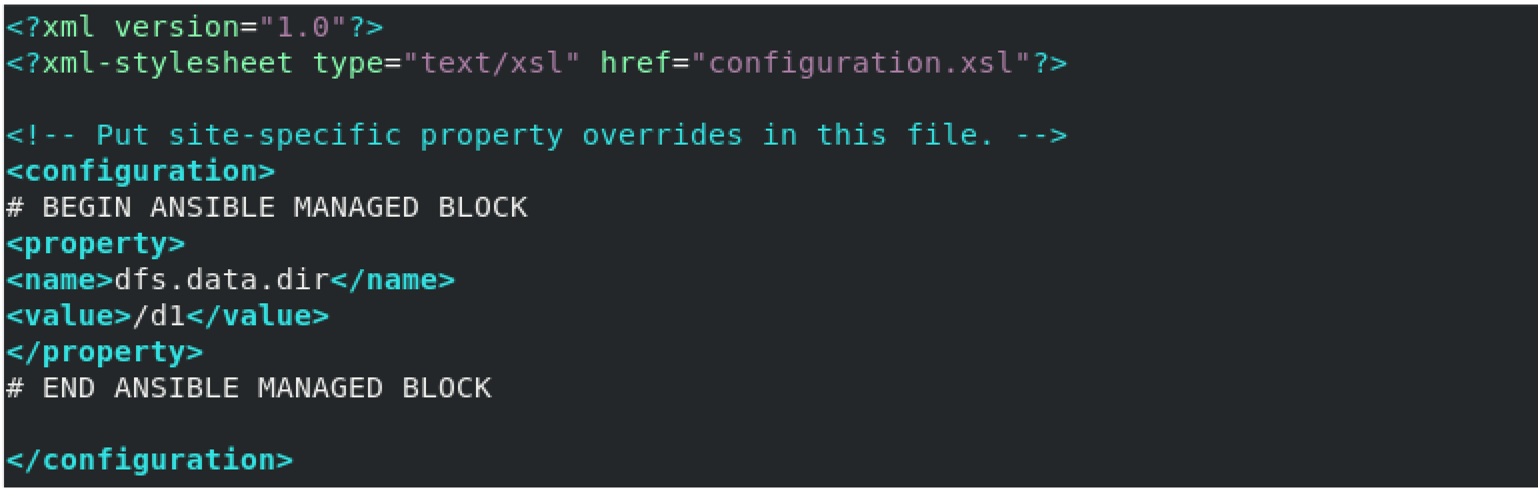
```
[root@localhost mycode]# ansible-playbook had1.yml

PLAY [Configuration Of hadoop internal files] ***************************************

TASK [Gathering Facts] **************************************************************
ok: [192.168.43.38]

TASK [Adding Configuration in hdfs-site.xml] ****************************************
[WARNING]: The value True (type bool) in a string field was converted to 'True' (type
string). If this does not look like what you expect, quote the entire value to ensure
it does not change.
changed: [192.168.43.38]
```



```
[root@localhost mycode]# ansible-playbook had1.yml

PLAY [Configuration Of hadoop internal files] ****************************************

TASK [Creating a storage directory in slave.] ****************************************
ok: [192.168.43.38]

TASK [Adding Configuration in hdfs-site.xml] *****************************************
ok: [192.168.43.38]

TASK [Adding Configuration in core-site.xml] *****************************************
changed: [192.168.43.38]

PLAY RECAP ***************************************************************************
192.168.43.38              : ok=3    changed=1    unreachable=0    failed=0    skippe
0    rescued=0    ignored=0
```

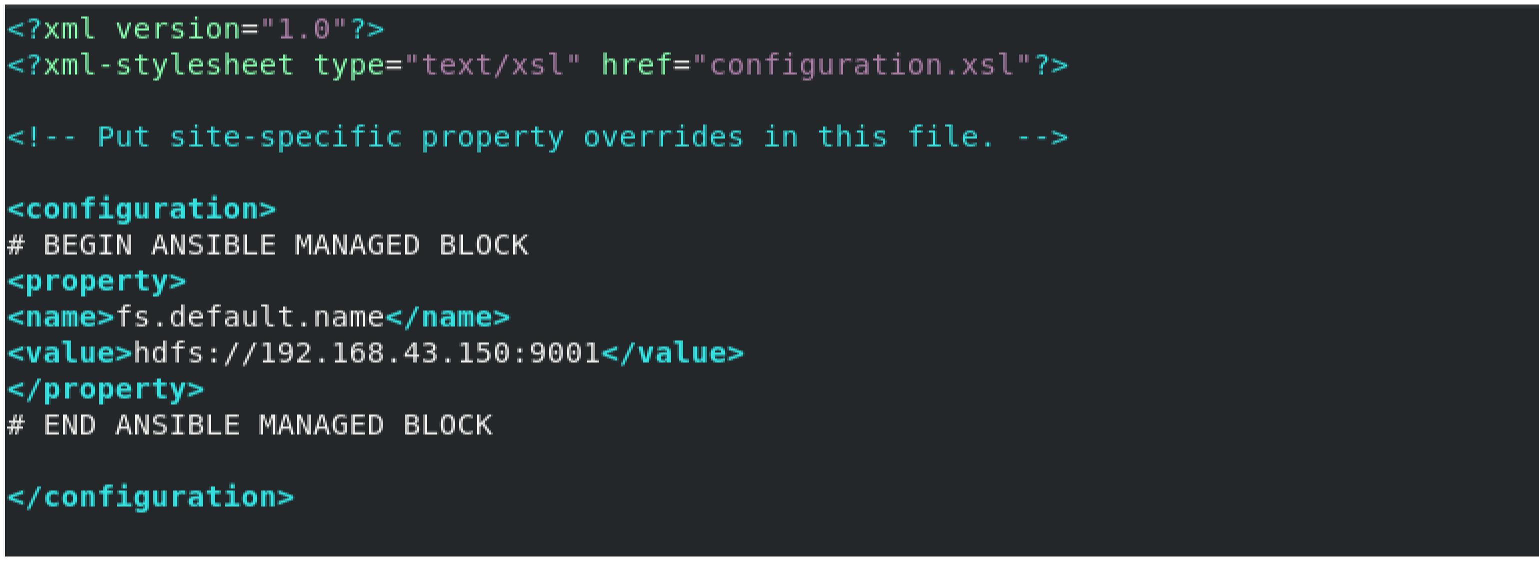
**Hence configuration in hadoop files done . For checking we go to *slave node* and check files.

# vim /etc/hadoop/hdfs-site.xml



```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<!-- Put site-specific property overrides in this file. -->
<configuration>
# BEGIN ANSIBLE MANAGED BLOCK
<property>
<name>dfs.data.dir</name>
<value>/d1</value>
</property>
# END ANSIBLE MANAGED BLOCK

</configuration>
```

# vim /etc/hadoop/core-site.xml


```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<!-- Put site-specific property overrides in this file. -->

<configuration>
# BEGIN ANSIBLE MANAGED BLOCK
<property>
<name>fs.default.name</name>
<value>hdfs://192.168.43.150:9001</value>
</property>
# END ANSIBLE MANAGED BLOCK

</configuration>
```

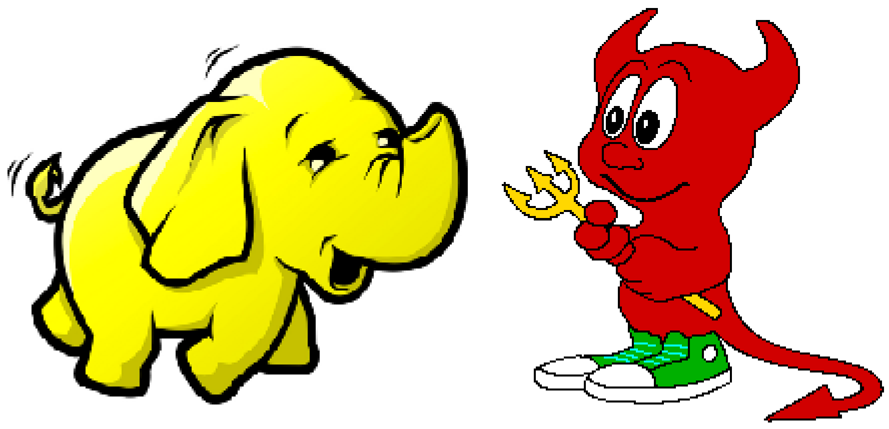
Hence adding block in files is done.

....

3. Hadoop Daemon:

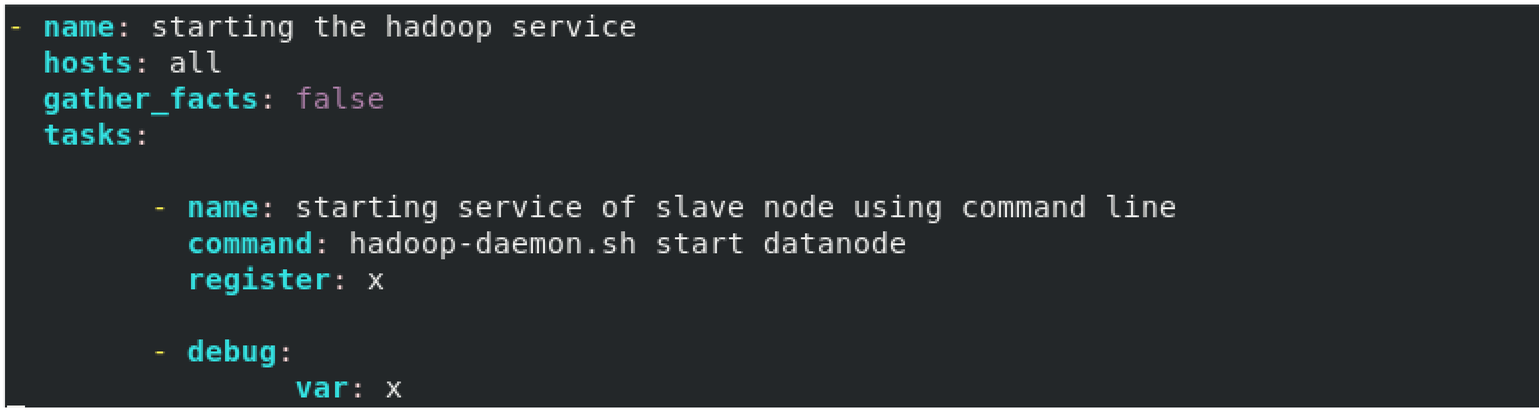
We can come to the conclusion that the Hadoop cluster is running by looking at the Hadoop daemon itself. A Daemon is nothing but a process. So, Hadoop daemons are nothing but Hadoop processes. As Hadoop is built using Java, all the Hadoop daemons are Java processes.

We can find these daemons in the *sbin* directory of Hadoop. After moving into the *sbin* directory, we can start all the Hadoop daemons by using the command *start-all.sh.*After executing the command, all the daemons start one by one. After all the daemons have started, we can check their presence by typing jps, which gives the list of all Java processes that are running.



For that we create playbook ans we use *Ansible Command Module.*

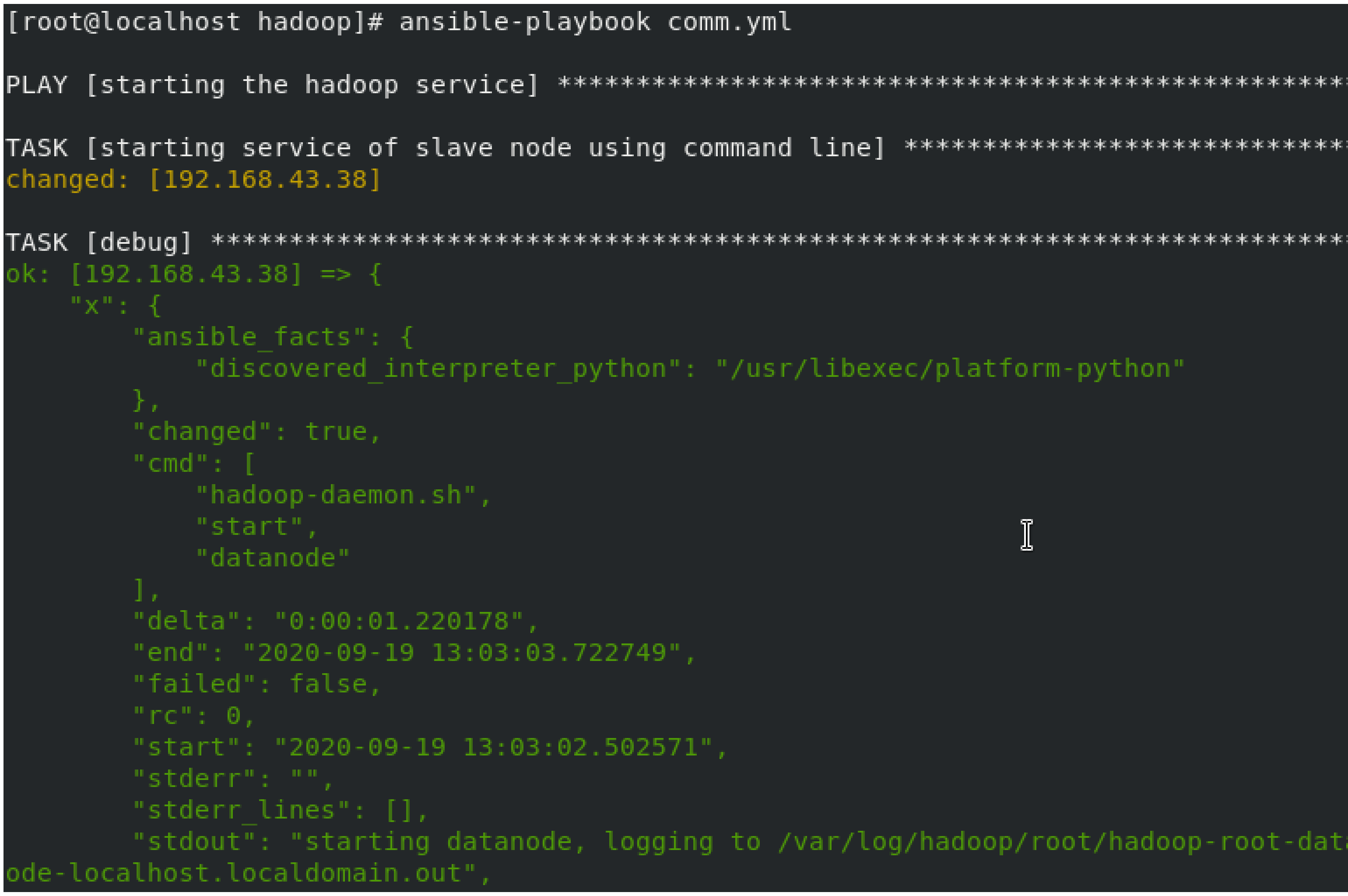
# vim comm.yml



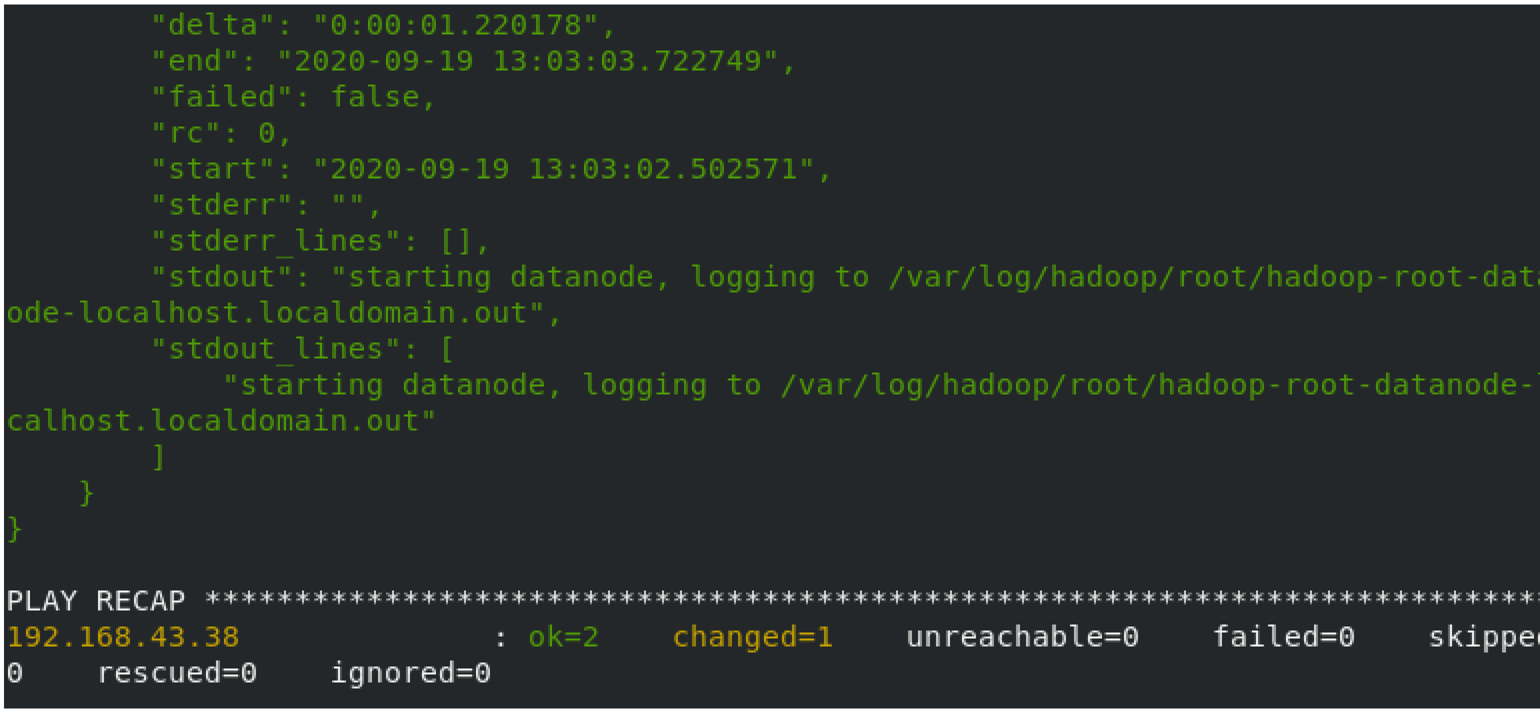
```
- name: starting the hadoop service
  hosts: all
  gather_facts: false
  tasks:

        - name: starting service of slave node using command line
          command: hadoop-daemon.sh start datanode
          register: x

        - debug:
                var: x
```

...........run the playbook............

ansible-playbook comm.yml



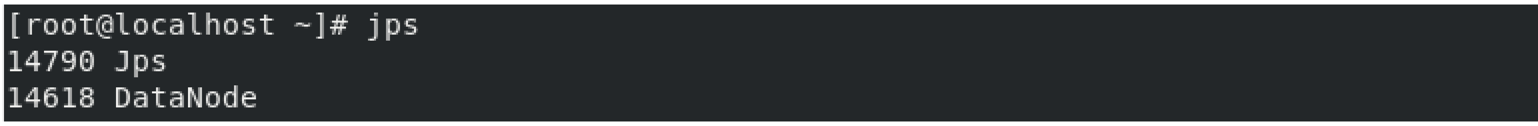
```
[root@localhost hadoop]# ansible-playbook comm.yml

PLAY [starting the hadoop service] *****************************************************

TASK [starting service of slave node using command line] *****************************
changed: [192.168.43.38]

TASK [debug] **************************************************************************
ok: [192.168.43.38] => {
    "x": {
        "ansible_facts": {
            "discovered_interpreter_python": "/usr/libexec/platform-python"
        },
        "changed": true,
        "cmd": [
            "hadoop-daemon.sh",
            "start",
            "datanode"
        ],
        "delta": "0:00:01.220178",
        "end": "2020-09-19 13:03:03.722749",
        "failed": false,
        "rc": 0,
        "start": "2020-09-19 13:03:02.502571",
        "stderr": "",
        "stderr_lines": [],
        "stdout": "starting datanode, logging to /var/log/hadoop/root/hadoop-root-dat
ode-localhost.localdomain.out",
```



```
        "delta": "0:00:01.220178",
        "end": "2020-09-19 13:03:03.722749",
        "failed": false,
        "rc": 0,
        "start": "2020-09-19 13:03:02.502571",
        "stderr": "",
        "stderr_lines": [],
        "stdout": "starting datanode, logging to /var/log/hadoop/root/hadoop-root-data
ode-localhost.localdomain.out",
        "stdout_lines": [
            "starting datanode, logging to /var/log/hadoop/root/hadoop-root-datanode-
calhost.localdomain.out"
        ]
    }
}

PLAY RECAP *********************************************************************
192.168.43.38              : ok=2    changed=1    unreachable=0    failed=0    skippe
0    rescued=0    ignored=0
```

If you are able to see the Hadoop daemons running after executing the *jps* command, we can safely assume *that the H*adoop cluster is running.

Just go to slave and run the command jps.



```
[root@localhost ~]# jps
14790 Jps
14618 DataNode
```

Hence finally we configure our slave node.

Thank You for Reading !!