

# Optimization of Systems-Development Life Cycle through Automation using Ansible

Kartik Vadhera<sup>1</sup>, Abhishek Deshwal<sup>2</sup> and Amrendra Tripathi<sup>3</sup>

University of Petroleum and Energy Studies, Dehradun, Uttarakhand, India

<sup>1</sup>kartikvadhera007@gmail.com

**Abstract.** In order to sustain oneself in this era of cutting-edge technology the companies need to adopt different technologies for the efficient execution of tasks. In pursuit of accomplishing the targets the companies install number of software or add lines of codes. DevOps is a large number of programming improvement repetitions that consolidate software development (Dev) and information-technology operations (Ops) to shorten the lifecycle of system development while delivering on patches, updates and features from time to time in close collaboration with business objectives. Time is what everybody wants to save and this where Automation comes into practice. This paper throws light on how companies can adopt the simple automation by creating HA proxy server, webservers, and Hadoop-cluster with the help of automation tool called Ansible. Ansible is an open-source mechanization platform, used for IT jobs, for example framework management, application instalment, intra-service coordination and accoutering. Here, YAML and Jinja languages are used to create and automate both the HA proxy server and Hadoop cluster.

**Keywords:** Automation, Ansible, Hadoop, HA proxy server, YAML

## 1 Introduction

There are some programming elements and innovations available to facilitate big data analytics. Hadoop is a key innovation used to process huge information and handle big data, their review and analysis, and stream computing. It is an open source programming project that manages and enables us the distributed processing of large data sets across clusters of commodity servers. Hadoop includes different modules i.e. HDFS (hadoop distributed file system), Map Reduce, Pig, HBase, etc. Hadoop is designed to scale up from single servers to thousands of machines, each of these offering local computation and storage [1]. The Hadoop Distributed File System (HDFS) provides global access to the files in the cluster. For extreme transportability, HDFS is updated as a client-level file system in Java that exploits the local file system on each hub or node, for example, ext3 or NTFS, to store information. The files in HDFS are isolated in large blocks usually 64MB, and each square is stored as a different file in the neighboring file system. HDFS is put into effect by two services: The Name Node and Data Node. The Name Node supervises the maintenance of the HDFS directory tree and is a centralized service in the cluster operating on a single node. Customers contact the Name Node to perform a basic activity of the file system, such

as opening, closing, renaming, and erasing. The Name Node does not store the HDFS information itself, but instead maintains a schedule between the HDFS file name, a preview of the blocks in the file, and the Data node(s) of which those blocks are being kept[2].

MapReduce is a programming model and associated execution for the management and production of huge collections of information that can be managed in a wide range of tasks in real-world. Customers specify the computation in terms of a map and a reduce function, and the rudimentary runtime system automatically parallelizes the calculation through huge clusters of machines, manages the failures of the machine and the schedules communication between the machines to efficiently use the network and disks. Software engineers discover that the system is simple to use: more than ten thousand particular MapReduce programs have been updated at Google in recent years, and a normal one hundred thousand MapReduce professions are run on Google groups every day, processing a sum of more than twenty petabytes of information per day[3].

HA Proxy, which represents High availability Proxy, is a well-known open source programming HTTP load balancer and proxy arrangement that can be run on Solaris, FreeBSD, and Linux. Its most normal use is to improve the reliability quality and performance of a server environment by circulating the remaining task at hand between different servers (e.g. Web, database, application). It is used in some important conditions, including Instagram, Twitter, Imgur and GitHub. There are four fundamental segments in the HA proxy configuration document[4].

They are global, backend, frontend, and default. These four sections specify how the server as an unabbreviated performs, what your default settings are, and how client requests are received and routed to your back-end servers.

With the advancement in technology time is becoming a major issue. Several things are done manually which takes time. In today's era, almost all the companies are facing the problem to store and process their large amount of data. Setting up a Hadoop cluster with a new configuration in a company is a very time taking task. Also, the users generally face problems like server not found or page not found due to excess load on the web server due to which sometimes one is not able to access the website. Henceforth, one often complaint that the particular website got crashed or the server is down. Now, suppose the new system comes into the industry, then one has to deploy all the codes according to the needs which already exists in the industry and this takes a lot of time to do all the changes. Howsoever, these things can be now done through automation.

This paper provides automated software which will setup and install software's on different platforms. Further in this paper, implementation of an automated Hadoop cluster so as to solve the big data problem through Ansible (Devops) is done. Also, the implementation of an automated HA Proxy Server is conducted so as to solve the problem of website crashing or server down problem.

## **2 Architecture of Hadoop and Ansible**

The Fig. 1 displays the architecture of Hadoop distributed file system commonly known as hdfs wherein a four-terabyte block is comprised of four blocks each of one terabyte from different systems. This is how several storage blocks can be added in order to

provide the required size by providing high availability to the users/companies.

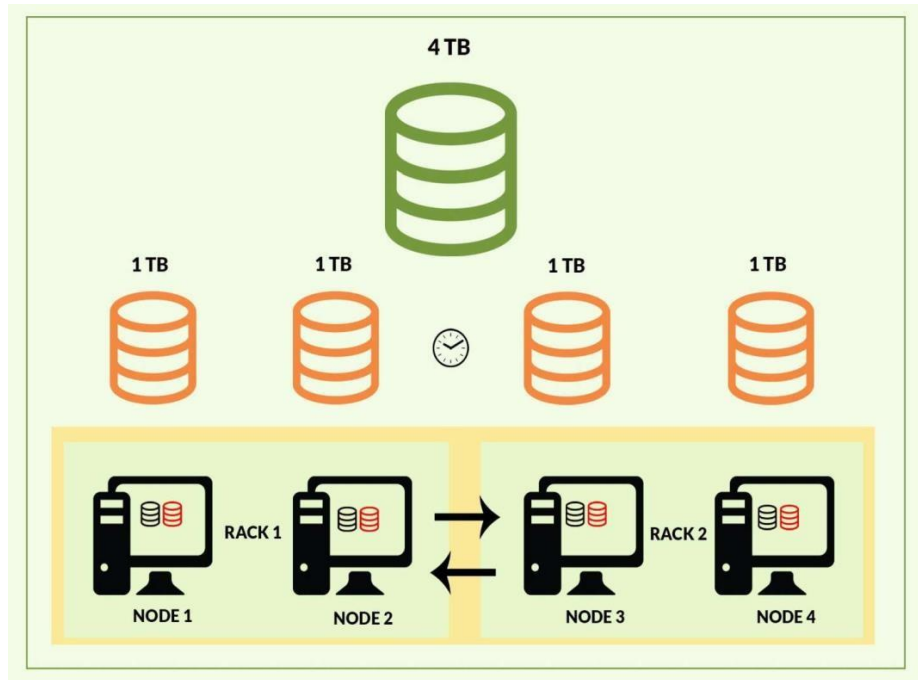


Fig.1. Architecture of Hadoop distributed file system

The Ansible environment is constituted by following items [5]:

- 1 Modules:** The modules look like small programs that Ansible sends from a control or master machine to all remote hosts or hubs. The modules are run using playbooks (see below), and they control things, for example, packages, services, and documents. Ansible runs all modules to introduce updates or whatever company is needed, and then evacuates them once completed. Ansible gives more than 450 modules to regular tasks.
- 2 APIs:** Various APIs (application programming interfaces) are accessible with the goal that one can broaden connection types (which means something more than just SSH for transport), callbacks and more.
- 3 Inventories:** All machines used with Ansible (the master or authority machine in addition to nodes) are stored in a simple and solitary document, next to their servers, databases, IP addresses, etc. When the inventory or log is enrolled then one can allot variables to any of the hosts utilizing a simple text document. Indeed, even we can extract the inventory from sources like EC2 (Amazon Elastic Compute Cloud).
- 4 Plugins:** They are extra bits of code that increase utility. Ansible accompanies some of its plugins, but you can also compose on your own. Cache, callback and action plugins are three models.
- 5 Playbooks:** Ansible playbooks resemble handbook or guide for tasks. These are basic

files written in YAML, which implies YAML Ain't Markup Language, a data serialization language comprehensible by Human. Playbooks are truly at the core of what makes Ansible so well known, in light of the fact that they portray the tasks to be performed rapidly and without the client having to know or recall a specific syntax. Not exclusively would they be able to announce configurations, however they can likewise organize the means of any physically controlled undertaking and perform assignments simultaneously or at various occasions. Every playbook comprises of at least one pieces, and the reason for a play is to map a gathering of hosts to all around characterized jobs, represented by tasks.

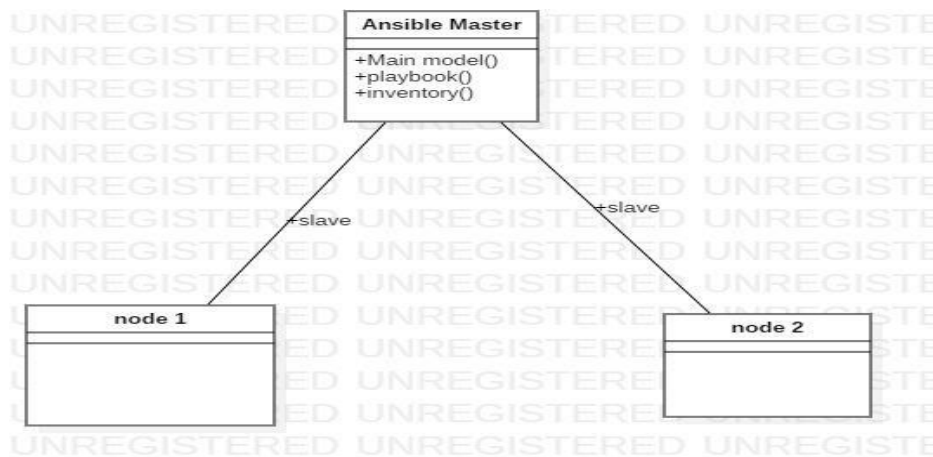


Fig.2. Ansible architectural diagram

### 3 METHODOLOGY:

#### 3.1 Hadoop

The primary infrastructure software services aimed to automate are as follows: -

**Step 1:** Install Ansible package in Linux using yum. In this installation of Ansible package using “yum install Ansible” command is initiated. Before this, yum is to be configured.

**Step 2:** Then make Ansible galaxy of Hadoop Cluster in some different folders like playbooks. In this Hadoop cluster is implemented to solve the big data problem using command “Ansible-galaxy in it Hadoop cluster”.

**Step 3:** Accordingly put the client IP in hosts file so that it can read the IP from there and so that playbook can be automatically run in that system. The location of host file would be “/etc/ansible/hosts”.

**Step 4:** Configure Ansible file according to the need in ansible.cfg file.

**Step 5:** Write a Hadoop cluster role to setup Master Node, Slave Node.

**Step 6:** Then create a site.yml file in which write a code to import the role Hadoop.

**Step 7:** Execute the file using the command “Ansible-playbook site.yml”.

**Step 8:** In the Client node role: we copy the java and Hadoop setup files to the respective nodes.

**Step 9:** In the Master node role: we copy the master node configuration i.e. core-site.xml and hdfs-site.xml on the master node machine.

**Step 10:** In the Slave node role: we copy the slave-node configuration i.e. core-site.xml and hdfs-site.xml on the master node machine.

**Step 11:** Now we have to run the following command:

On Name Node - "hadoop-daemon.sh start namenode" On Data Node - "hadoop-daemon.sh start datanode"

**Step 12:** On the client machine check the hadoop setup by running the following command: "hadoopfsadmin -report"

**Step 13:** To upload a file use the command: "hadoopfs -put filename /"

**Step 14:** To upload a file use the command: "hadoopfs cat /filename"

### 3.2 HA Proxy

**Step 1:** Install ansible on the system.

**Step 2:** Then make ansible galaxy of HA Proxy Server in a folder. In this we are making HA Proxy which will work as a load balancer. Also, Hadoop cluster is implemented to solve the bigdata problem using command "ansible-galaxy in it haproxyserver".

**Step 3:** Accordingly put the client IP in hosts file so that it can read the IP from there and so that

playbook can be automatically run in that system. The location of host file would be "/etc/ansible/hosts".

**Step 4:** Configure ansible file according to the need in ansible.cfg file.

**Step 5:** Write a suitable code inside roles for deploying HA proxy and hadoop cluster inside tasks folder.

**Step 6:** Then create a site.yml file in which write a code to import the role for HA proxy and hadoop.

### 3.3 Web server and ftp server

**Step 1:** Create a file with yml as an extension i.e. vim webserver.yml

**Step 2:** Write the code inside the file as written below.

```
- hosts: 192.168.1.1 //IP address of the system where you want to configure
the webserver.
tasks:
  - package:
    name: httpd
    state: present

  - service:
    name: httpd
```

```
state: started
enabled: true
```

Note: Indentation has to be kept in mind.

**Step 3:** Now save the file and run the playbook using the command:-  
ansible-playbook webserver.yml

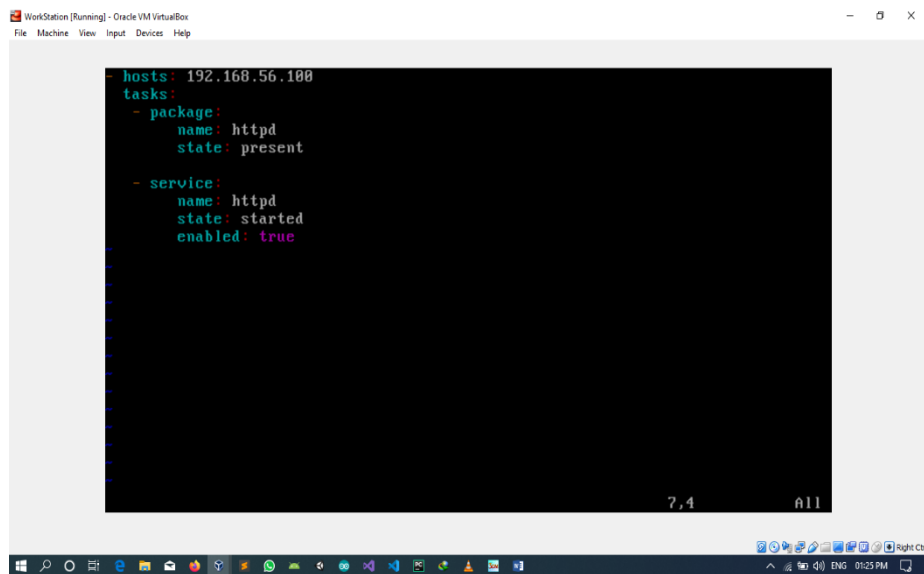


Fig.3. Web server setup

Setting up FTP server using Ansible:

**Step 1:** Create a file with yml as an extension i.e. vim ftpserver.yml

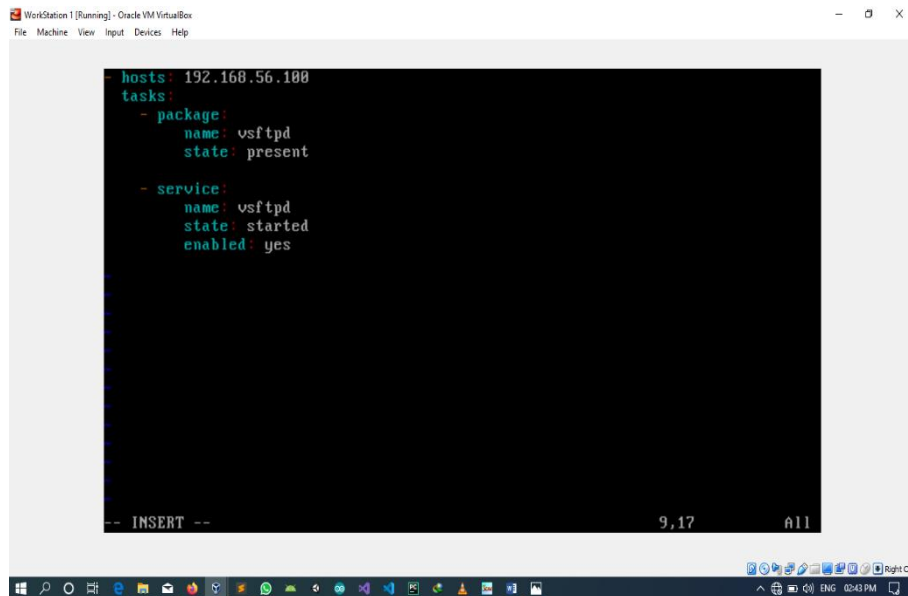
**Step 2:** Write the code inside the file as written below.

```
- hosts: 192.168.1.1 //Ip address where you want to configure the ftpserver.
tasks:
  - package:
      name: vsftpd
      state: present

  - service:
      name: vsftpd
      state: started
      enabled: true
```

**Step 3:** Now save the file and run the playbook using the command:

ansible-playbook ftpserver.yml



```
hosts: 192.168.56.100
tasks:
  - package:
      name: vsftpd
      state: present

  - service:
      name: vsftpd
      state: started
      enabled: yes

-- INSERT --
```

Fig.4. ftp server setup

### Files in YAML Syntax

The indentations are to be used strictly as described below.

```
-name: deploy slave node import_playbook: sn.yml
-name: deploy client node import_playbook: cn.yml Sn.yml
-hosts: dn roles:
- role: slavenode Inventory
[dn] DATA NODE
192.168.56.115 ansible_user=root ansible_password=redhat #slave1 192.168.56.116
ansible_user=root ansible_password=redhat #slave2 192.168.56.116
ansible_user=root ansible_password=redhat #slave No
[nn] MASTER NODE
192.168.56.114 ansible_user=root ansible_password=redhat #master
```

Client role: main.yml

```
-command: "rpm -ivh hadoop-1.2.1-1.x86_64.rpm --force"
```

```
-command: "rpm -ivh jdk-8u171-linux-x64.rpm --force"
```

-template:

```
src: ".bashrc"
```

```
dest: "/root/.bashrc"
```

-template:

```
src: "core-site.xml.j2"
```

```
dest: "/etc/hadoop/core-site.xml"
```

Master Node: main.yml

```

-command: "rpm -ivh hadoop-1.2.1-1.x86_64.rpm --force"
-command: "rpm -ivh jdk-8u171-linux-x64.rpm --force"
-template:
src: ".bashrc"
dest: "/root/.bashrc"
-file:
path: /master state:
directory 21
-template:
src: "hdfs-site.xml"
dest: "/etc/hadoop/hdfs-site.xml"
-template:
src: "core-site.xml.j2"
dest: "/etc/hadoop/core-site.xml"
-command: "hadoopnamenode -format -force"
#- command: "hadoop-daemon.sh start namenode"

```

```

Core-Site.xml
<configuration>
<property>
<name>fs.default.name</name>
{ % for i in groups["nn"] % }
<value>hdfs://{ i }:9001</value>
{ % endfor % }
</property>
</configuration>

```

## 4 RESULTS:

Fig.5. reflects the web-based hdfs portal and gives information about the Hadoop cluster. It tells the live numbers of data nodes with their capacity, current use and the last contract. Basically, it is an overview of the complete cluster configuration.



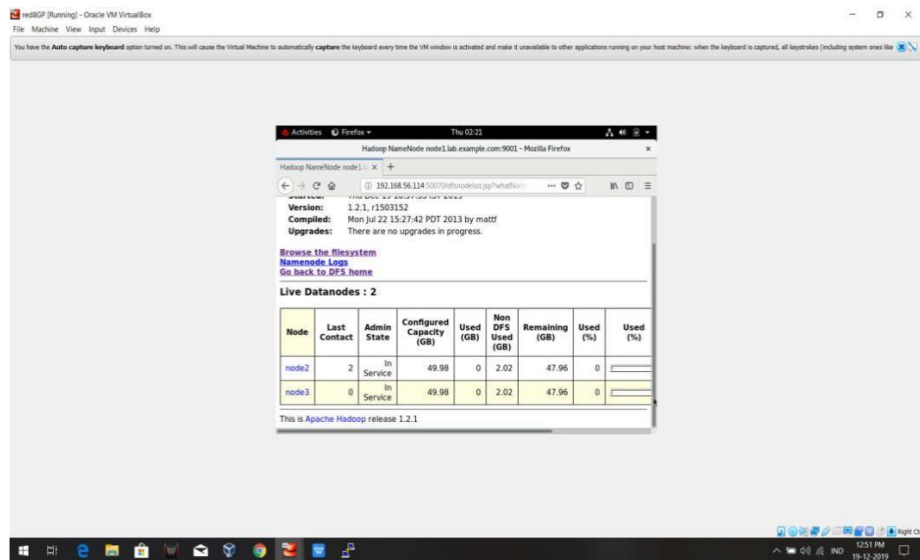


Fig.5. Hadoop Cluster

The following Fig.6. describes the summary of the particular cluster in detail including the no of replicated blocks and the heap size. It also holds information about the no of dead nodes, the configured capacity, dfs used, dfs remaining and the log file of the configuration.

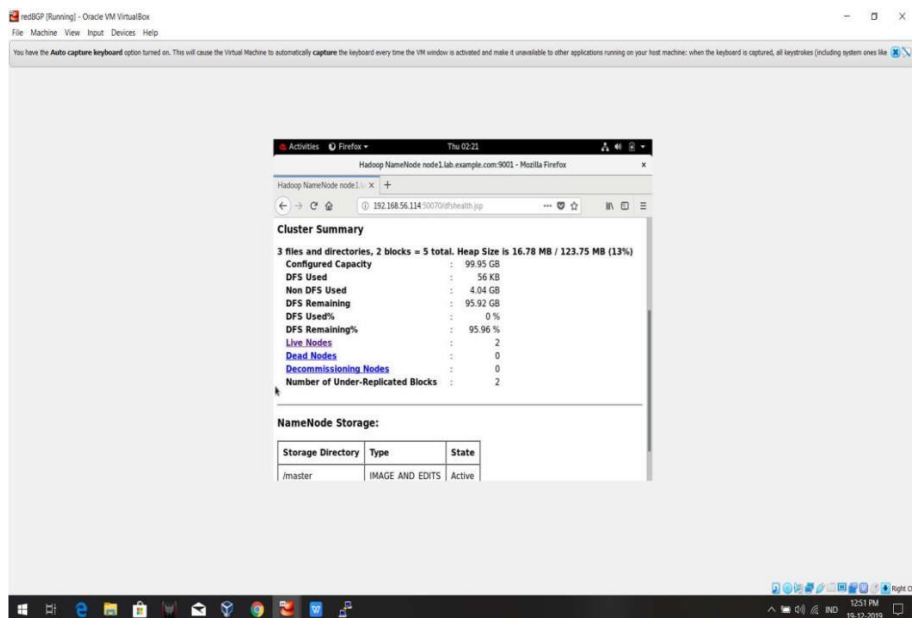


Fig.6. Cluster summary

The Fig.7. shows the information about the files that have been stored on the particular cluster. In addition to that detailed information about the file is available that includes the type of the file, the size, the permission, the owner and the group. The data of the files can also be accessed from the portal along with the local logs for the file entries present on the bottom of the screen.

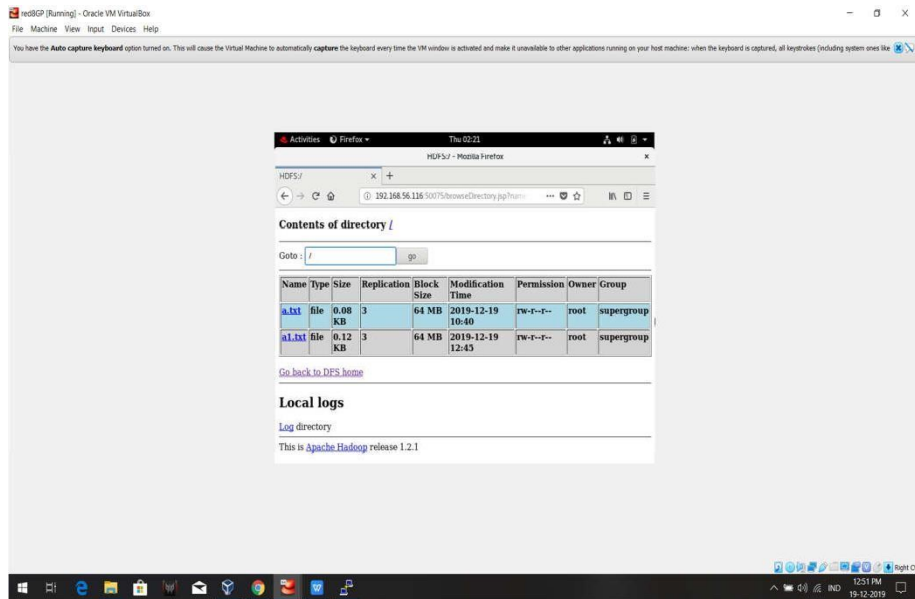


Fig.7. File stored in cluster

In this paper we have obtained the following result: -

- Successfully created a multi-node cluster which gives us the required amount of hdfs storage. In our setup fifty terabytes of storage from each of the slave nodes add up and give as hundred terabytes of storage as a whole.
- Successfully uploaded and retrieved files from the cluster which can be seen in the file upload section below.
- Managed to retrieve the file even when one of the nodes was disconnected while retrieving improvising high availability of the current prototype.
- Successfully setup the HA Proxy server i.e. the load balancer mode, webserver and ftp-servers.

## 5 CONCLUSION:

In this paper HA proxy server, webserver, and Hadoop-cluster with the help of automation tool called Ansible had been created. Ansible is an open-source mechanization platform, used for IT jobs, for example framework management, application instalment, intra-service coordination and accoutering. Further in the paper

the YAML and jinja languages are used to automate both the HA Proxy server and Hadoop cluster. Henceforth , the paper highlights the implementation of automated Hadoop cluster so as to solve the big data problem through Ansible (Devops) also it make use of an automated HA Proxy Server in order to solve the problem of website crashing or server down problem.

## References

1. Anuradha, J. (2015). A brief introduction on Big Data 5Vs characteristics and Hadoop technology. *Procedia Computer Science*, 48, 319-324.
2. Anuradha, J. (2015). A brief introduction on Big Data 5Vs characteristics and Hadoop technology. *Procedia Computer Science*, 48, 319-324.
3. Shafer, Jeffrey, Scott Rixner, and Alan L. Cox. (2010) "The Hadoop distributed file system: Balancing portability and performance."2010 IEEE International Symposium on Performance Analysis of Systems & Software (ISPASS). IEEE, 2010.
4. Dean, J., & Ghemawat, S. (2008). MapReduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1), 107-113.
5. <https://www.howtoforge.com/tutorial/ubuntu-load-balancer-haproxy/>
6. <https://www.redhat.com/files/summit/session-assets/2017/LT122010-Observability-and-automation.pdf>