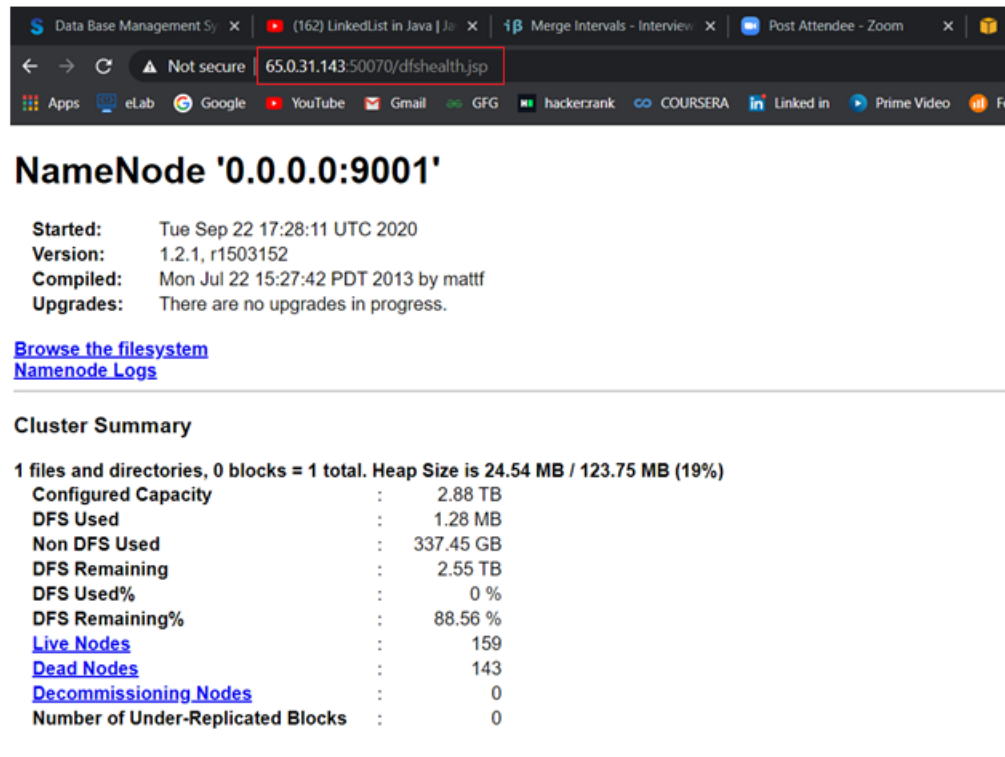


Implementation of the supercomputer:

In previous module, we have created multiple slaves using AWS cloud. Now, it is time that we upload files on the master and see **how it further gets distributed to the slaves**:



NameNode '0.0.0.0:9001'

Started: Tue Sep 22 17:28:11 UTC 2020
Version: 1.2.1, r1503152
Compiled: Mon Jul 22 15:27:42 PDT 2013 by mattf
Upgrades: There are no upgrades in progress.

[Browse the filesystem](#)
[NameNode Logs](#)

Cluster Summary

1 files and directories, 0 blocks = 1 total. Heap Size is 24.54 MB / 123.75 MB (19%)

Configured Capacity	:	2.88 TB
DFS Used	:	1.28 MB
Non DFS Used	:	337.45 GB
DFS Remaining	:	2.55 TB
DFS Used%	:	0 %
DFS Remaining%	:	88.56 %
Live Nodes	:	159
Dead Nodes	:	143
Decommissioning Nodes	:	0
Number of Under-Replicated Blocks	:	0

In this screen screenshot we can see that the configured capacity of the master is 2.88 TB. There are 159 live nodes and 143 dead nodes (we shall further move to permanently activate the configuration so that when the OS is shut down, slave does not die.).

We can see the report of the cluster using our browser as shown on the red box. 65.0.31.143 is the IP of the master. 50070 is used to access the report. By clicking on the **“browse the filesystem”** we can see the directories uploaded by the client to the master.



Currently, the directory is empty. For uploading the files, we need to **configure the client**.

For configuring the client, first of all, we need to install Hadoop and JDK as we have done for the master and the slave nodes. After that, we need to open the **core-site.xml** file and give the following details in it.

```
root@ip-172-31-8-135:/etc/hadoop
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

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

<configuration>

  <property>
    <name>fs.default.name</name>
    <value>hdfs://65.0.31.143:9001</value>
  </property>

</configuration>
~
~
~
~
~
~
~
~
-- INSERT --
```

Here, the **IP provided is that of the master** and the port number is the same as written in the core-site.xml file of master and slaves.

Now, the next step is to create a file and upload it to the master. **In hadoop, by default, size of each block is 64 MB**. However, we can change it. But, for now, we shall create a file of size more than 64 MB to see how it is getting striped and distributed to the slaves.

For creating a file, we shall write the following commands on the client server:

```
cd /etc/hadoop
```

```
vi newfile.txt //write anything that you want.
```

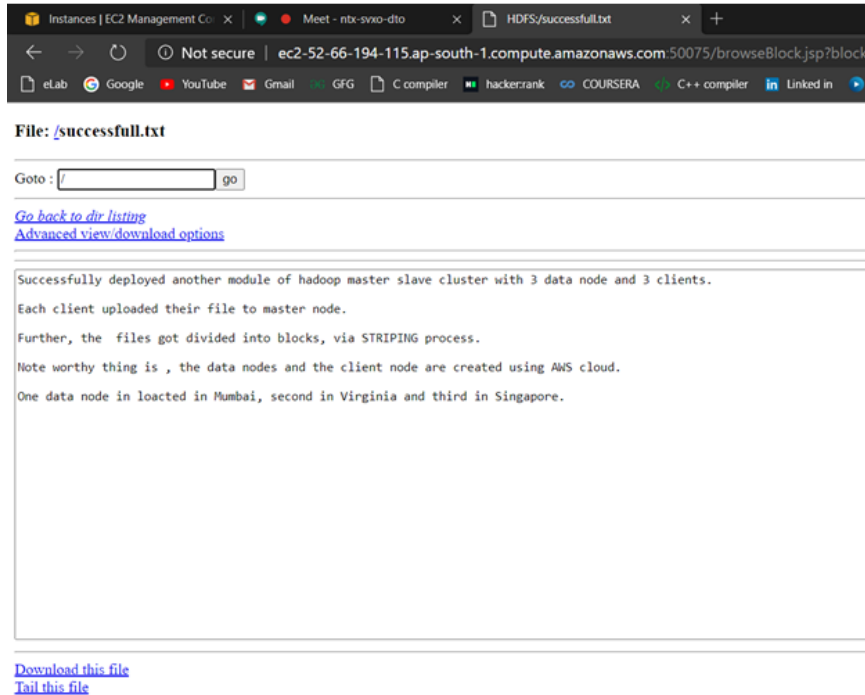
```
hadoop fs -put newfile.txt / // File gets uploaded.
```

Now, when you click on the “Browse the Filesystem” option, you will see the list of all the files, uploaded by several clients. Click on the name of your file to see in how many blocks it has been striped.

The screenshot shows the HDFS web interface. At the top, there is a text area containing a file's content. Below it, there are links for "Download this file" and "Tail this file". A "Chunk size to view (in bytes, up to file's DFS block size):" input field is set to 32768, with a "Refresh" button. The main section displays the "Total number of blocks: 3" in a green box. Below this, there are three columns of block information, each with a header and three data rows. The first column is highlighted with a green box, the second with an orange box, and the third with a pink box.

Block ID	Block Name	Block Size	Block Location
-844519701721016224:	13.233.37.132:50010	3.6.37.110:50010	13.125.188.73:50010
-2353415084506909549:	13.235.99.167:50010	18.232.51.46:50010	15.205.158.29:50010
7419045576885355684:	12.66.23.220:50010	54.198.133.33:50010	52.91.222.242:50010

After you click on your file, you can see its content. At the bottom, you can see total number of blocks. For this file it is 3. As shown on the green box are the **names of the blocks**. The orange box shows the **IPs of the block**. As far as the pink block is considered, for now, take it as the **“backup IP”** of the block.



For now, this is a **public cluster** where anyone can view and download the files of any client. However, in the upcoming modules, I shall provide some **security** so that the data uploaded by the client is safeguarded.

Also, Instagram and Facebook also follow the similar model of master slave cluster. But, when we upload our data, we don't configure our system to connect with their master. Then how do things work? Next module is going to be about these interesting insights.