**Big Data Hadoop H07 assignment**

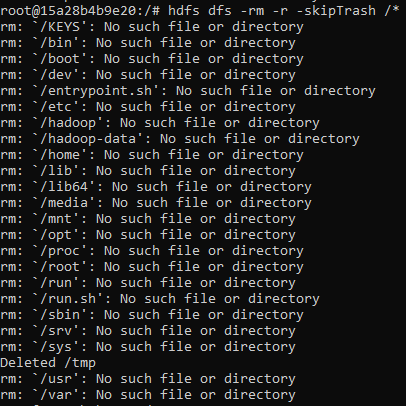
**25156**

**Shahmeer khan**

**Note:**

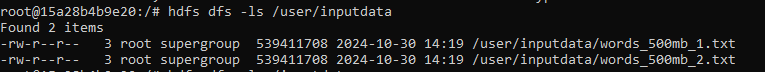
Using files of 500mbs each

**Remove directories visible from root prompt**

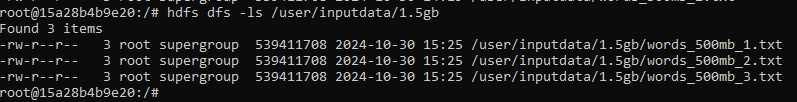


**Copy into Hadoop**

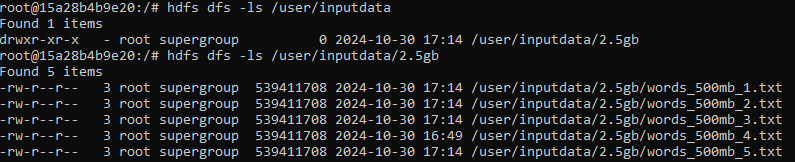
**1gb part**

****

**1.5gb part**

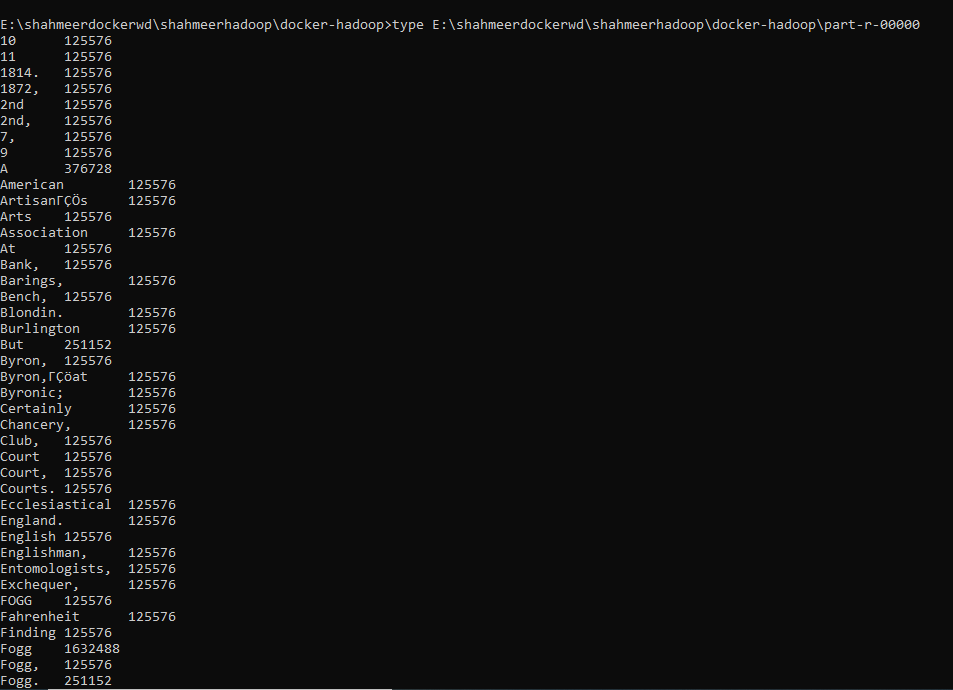
****

**2.5gb part**

****

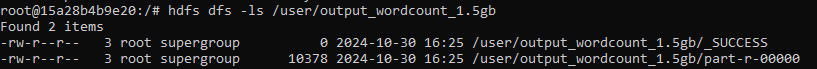
**WordCount snapshots:**

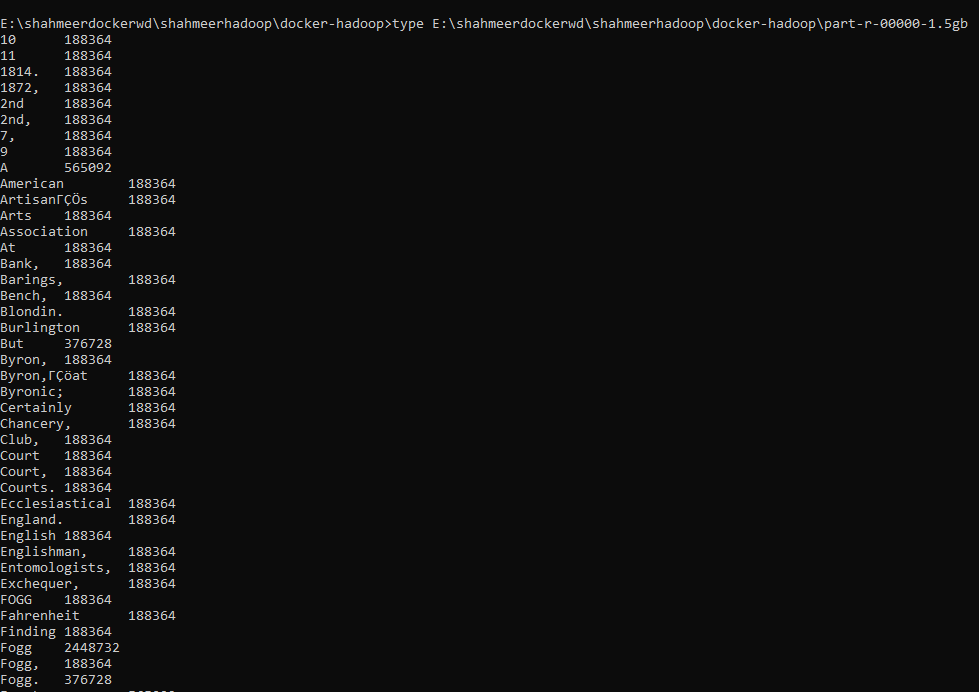
**1 gb**

****

**1.5gb**

Successful run

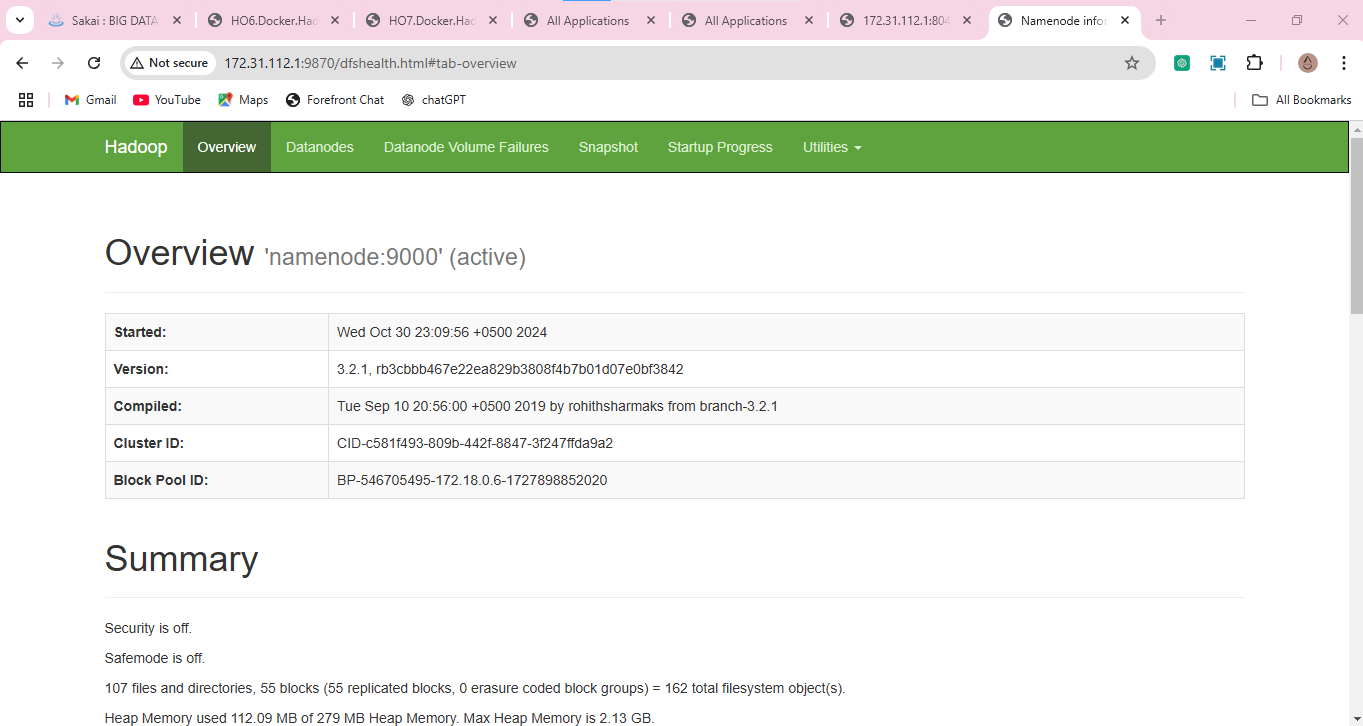
****

****

**2.5gb part**

Faced several issues with docker itself, kept crashing sometimes gave random wsl errors, sometimes containers stopped working, this happened throughout but managed to get 1gb and 1.5gb done, 2.5gb led to these issues happening constantly so had to give up

**Name Node**



**Configured capacity:** 1006.85 GB

****

This is the total storage capacity I can use for Hadoop cluster i.e. amount of data I can store in HDFS, no all the space in HDD isn’t in dockerized Hadoop, only enough to cater for the images.

**DFS usage:** 5.59 MB



This means 5.59 MB of the total allocated capacity of HDFS is being used

**Under replicated blocks:** 55



This means I have 55 data blocks that do not have the minimum required number of copies across the cluster

**Block deletion time**: Oct 30th

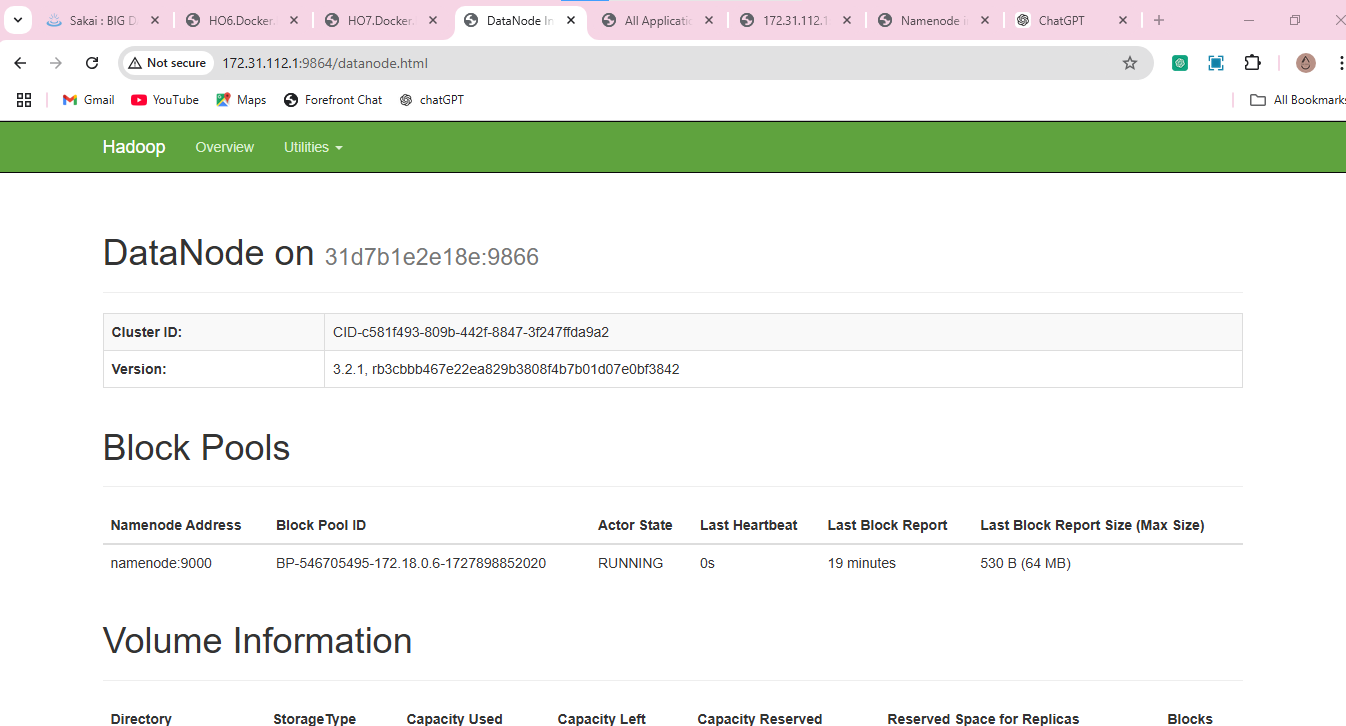


Block deletion doesn’t happen instantaneously, it can take some time i.e. a few minutes to a few hours, to check we can look at logs of data node and name node

**Checkpoint time:** Oct 30th



**Data Node**



**Metrics**

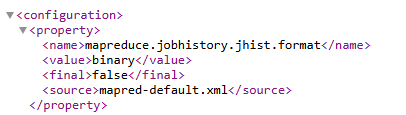


Yarn.home.dir refers to the path to where yarn is saved

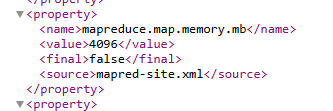
Java.vm.vendor refers to the VM provider

Os.name refers to the OS being used

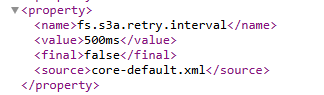
**Configuration**

****

Configuration for maintaining the history of mapreduce jobs performed

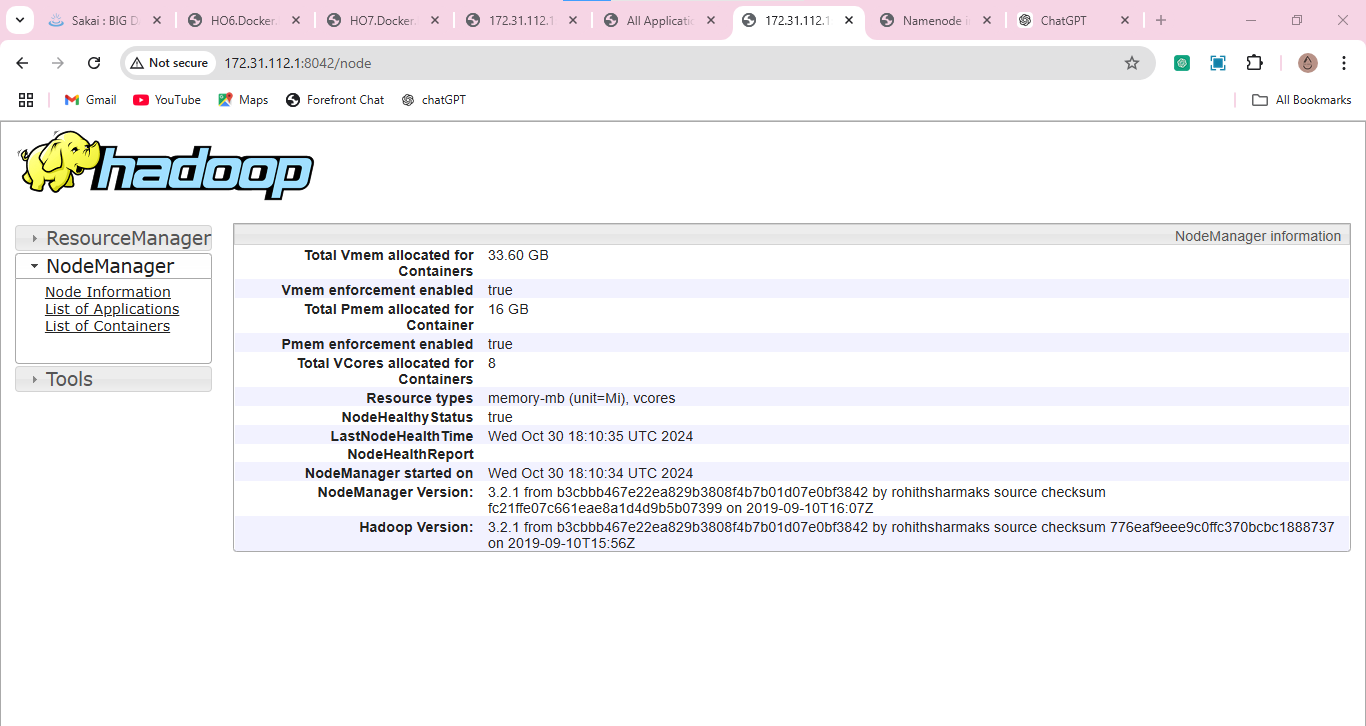


To keep memory of mapping part of map reduce



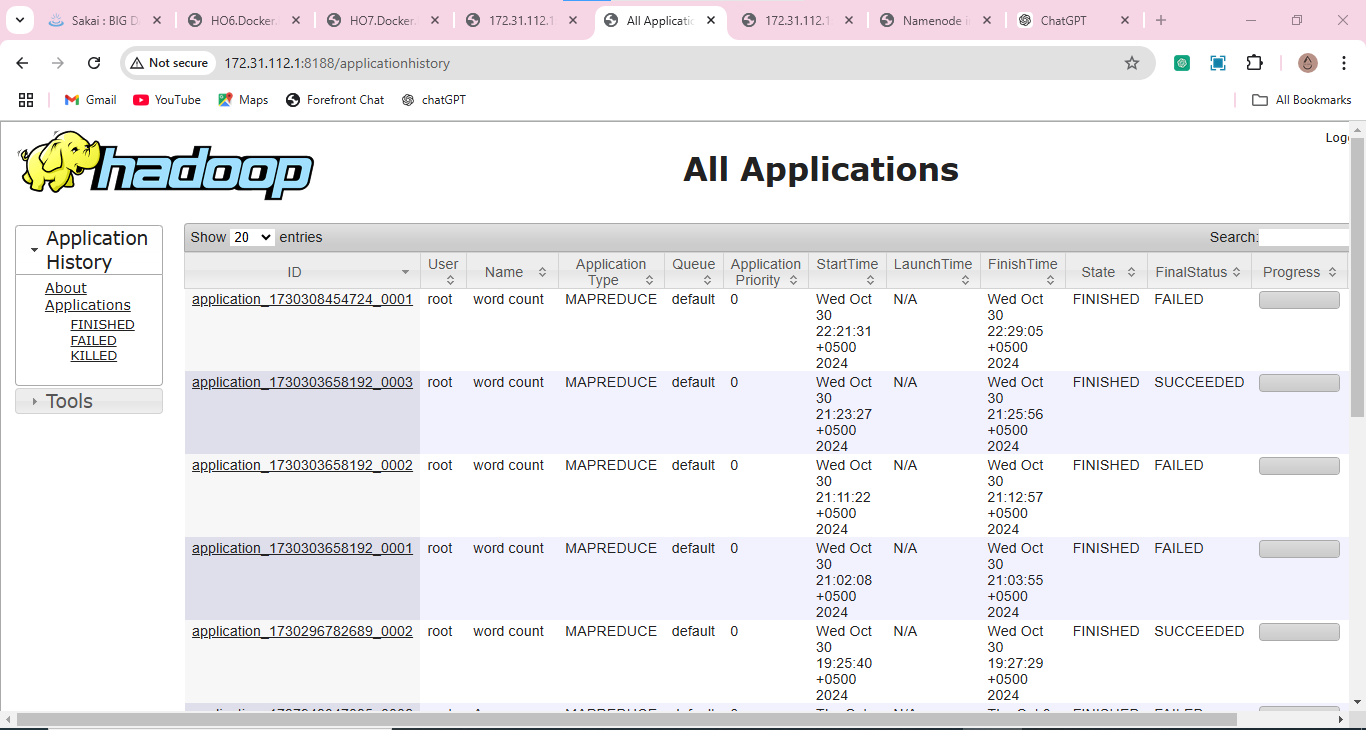
For s3a file system in Hadoop, specifically for interacting with Amazon S3

**Node manager**

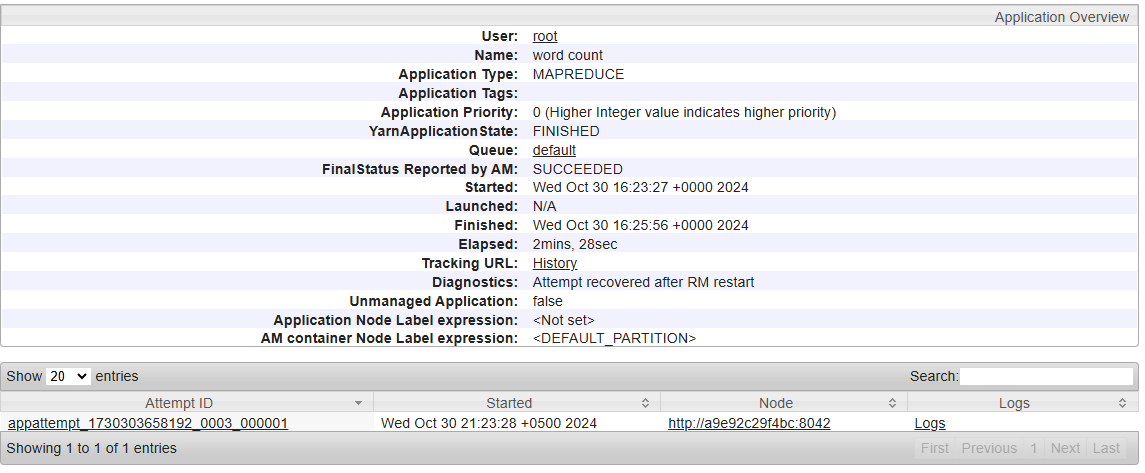
****

These are for yarn containers, not docker containers. Managed by yarn for jobs like mapreduce for resource allocation. Vmem is the total virtual memory allocated, Pmem is the physical memory and Vcores are the virtual cores. I think its just enough space needed.

**History server**

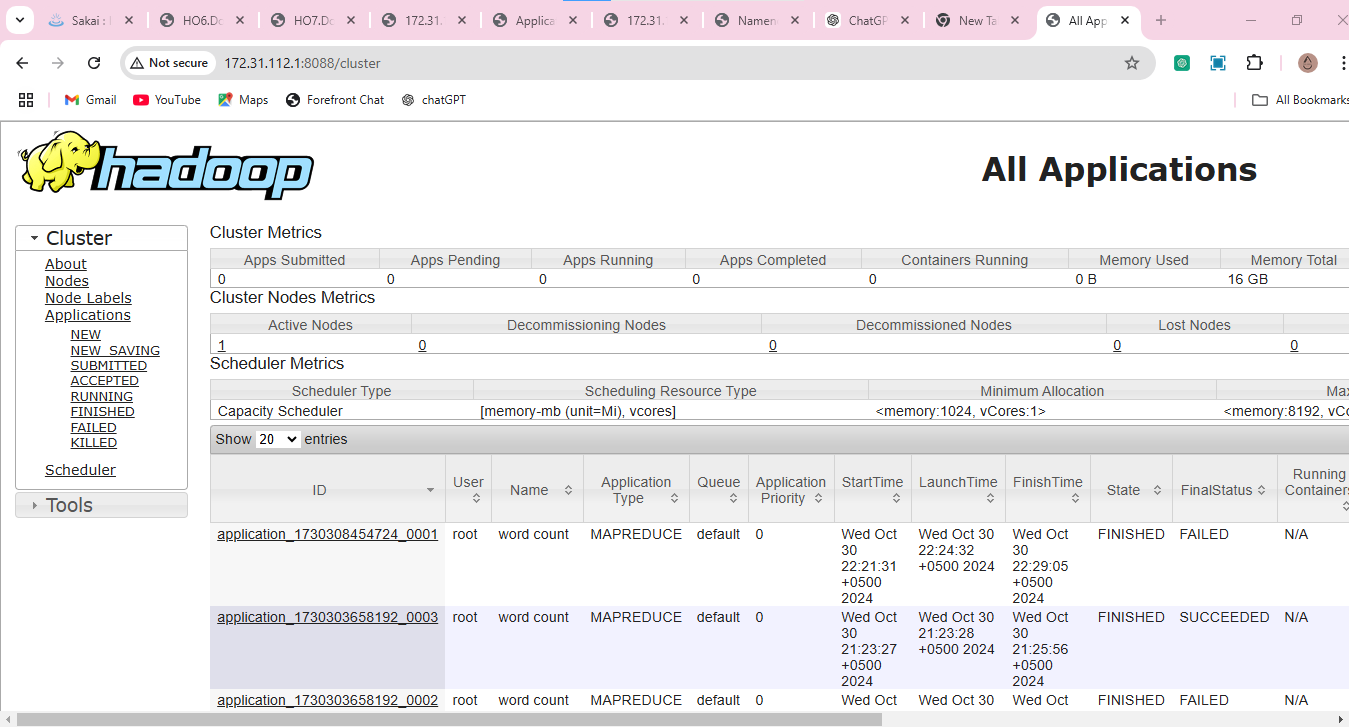
****

Maintains a log of all the jobs ran through Hadoop.

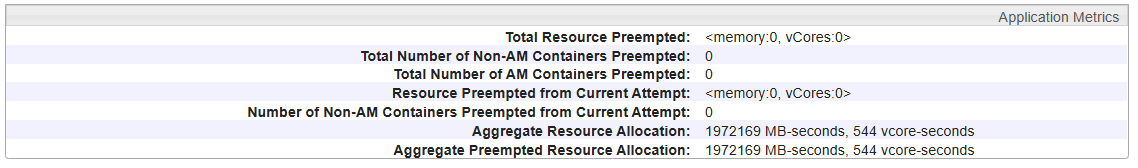


Executed task, took 2 mins and 28 seconds.

**Resource manager**



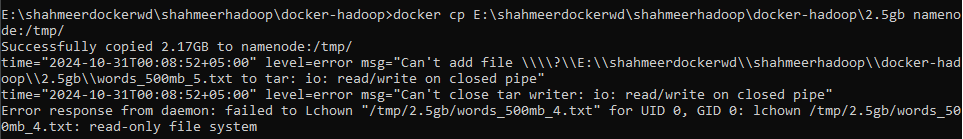
Here we can see the resources for a successful job



544 seconds of Vcores allocated.

**Main task:**

Cannot perform it for all big datasets due to docker failure, cluster issues and constant wsl errors**.**

****

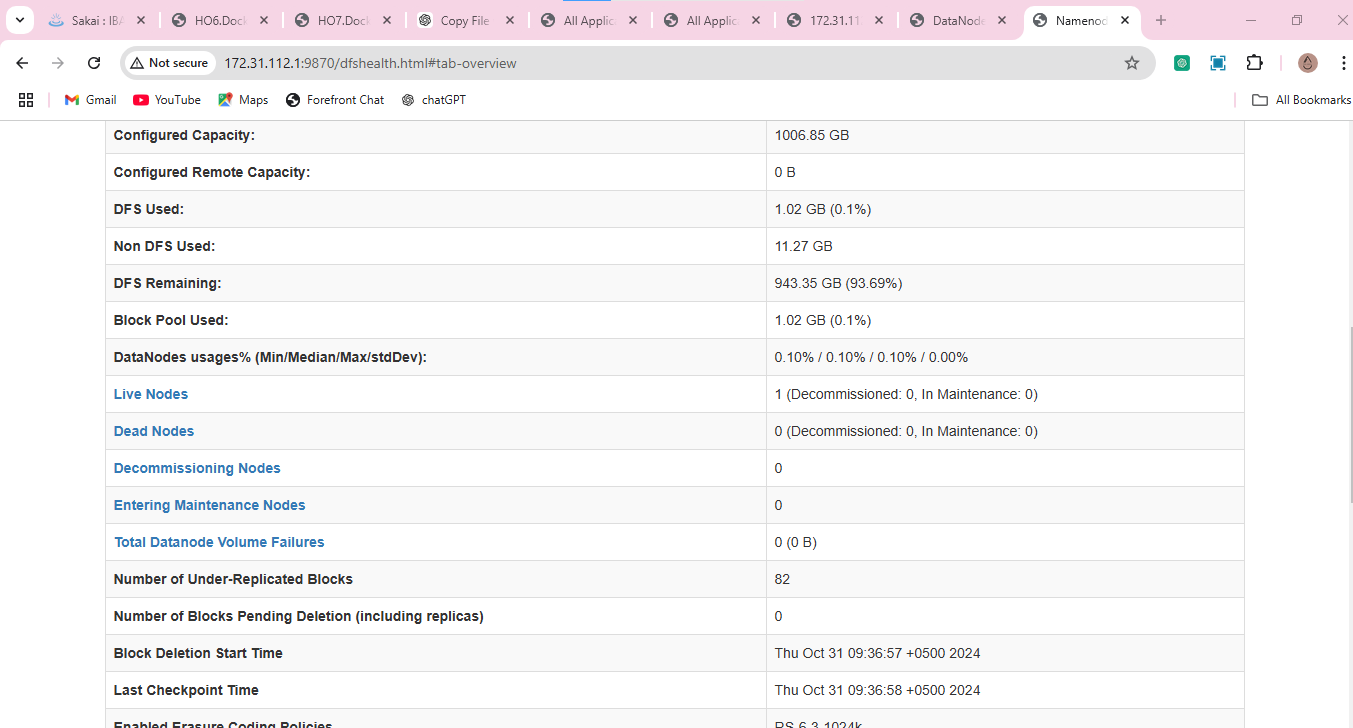
Errors like this keep happening and I have to restart my system(2-3 times at least) to get docker to work, could be an issue with the file size being 2.6 GB.

**For 1gb file**

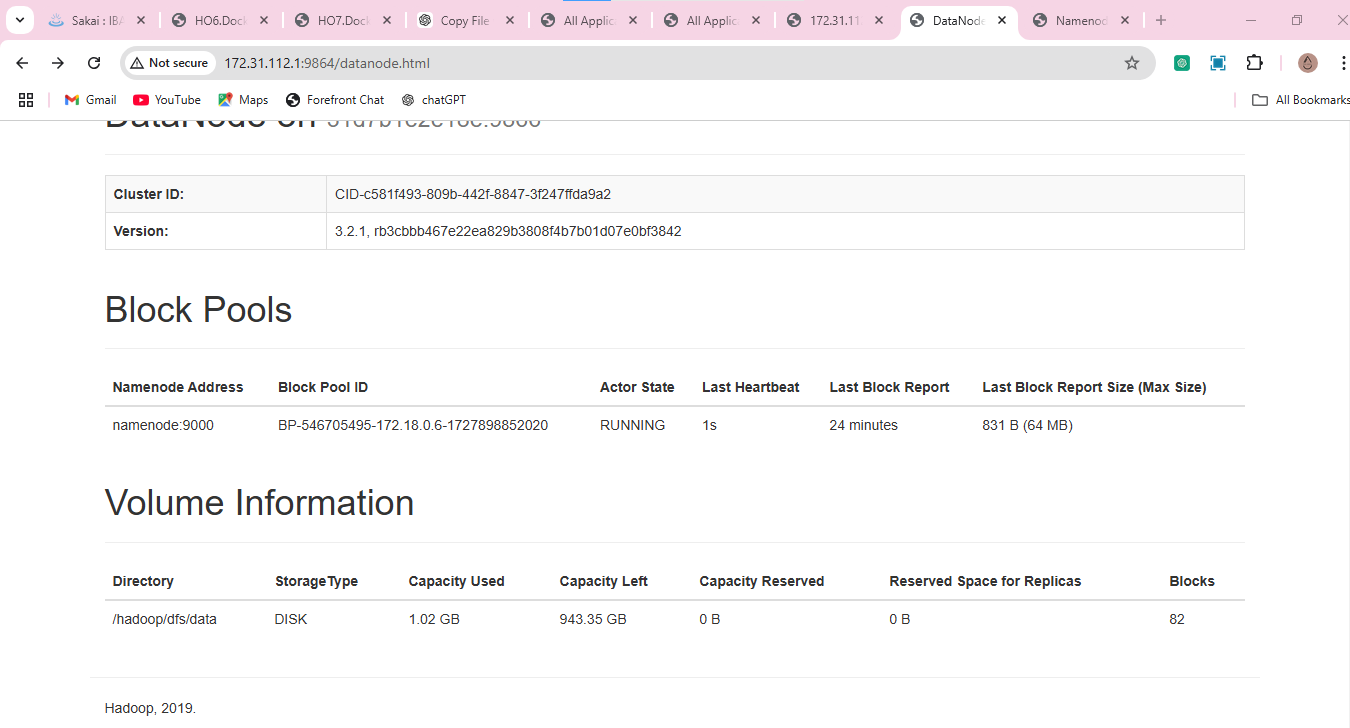
**Namenode**

1 GB of DFS capacity used out of 1006 GB configured capacity for HDFS

1 live node, 82 blocks with less than required number of replicas

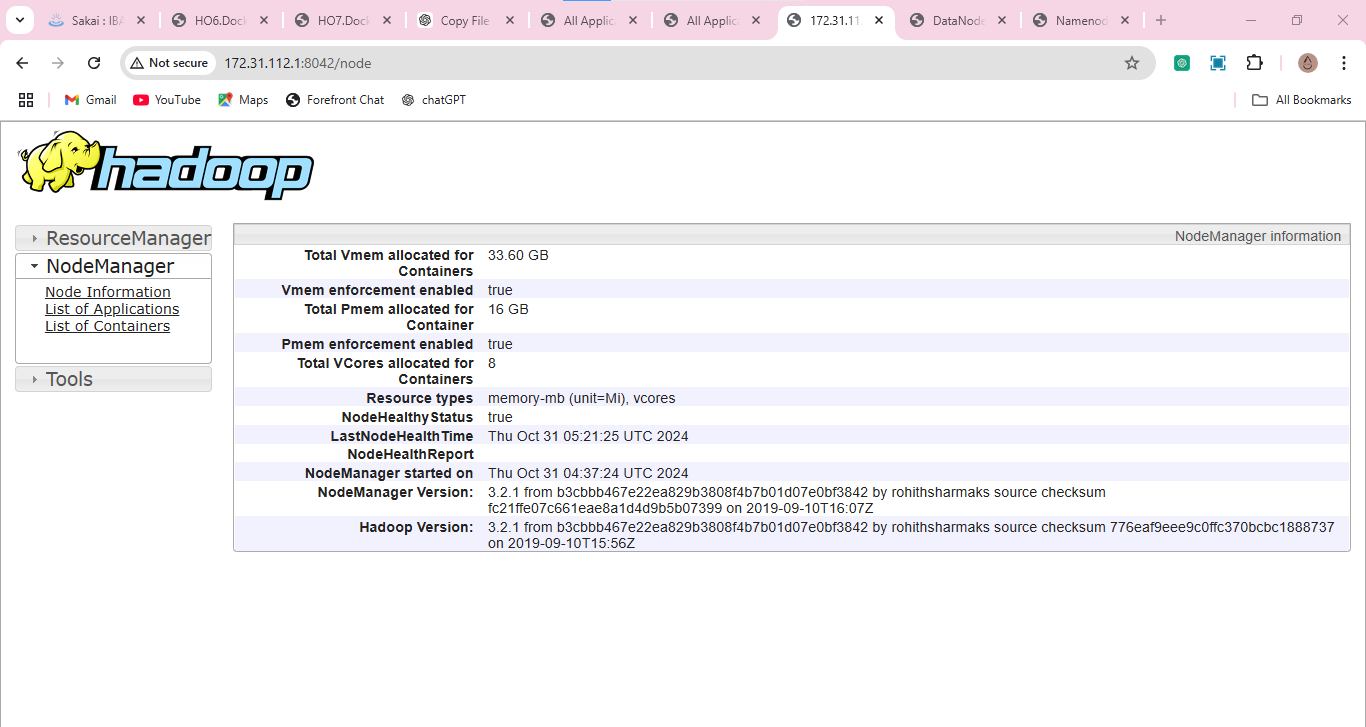
****

**Data node**

****

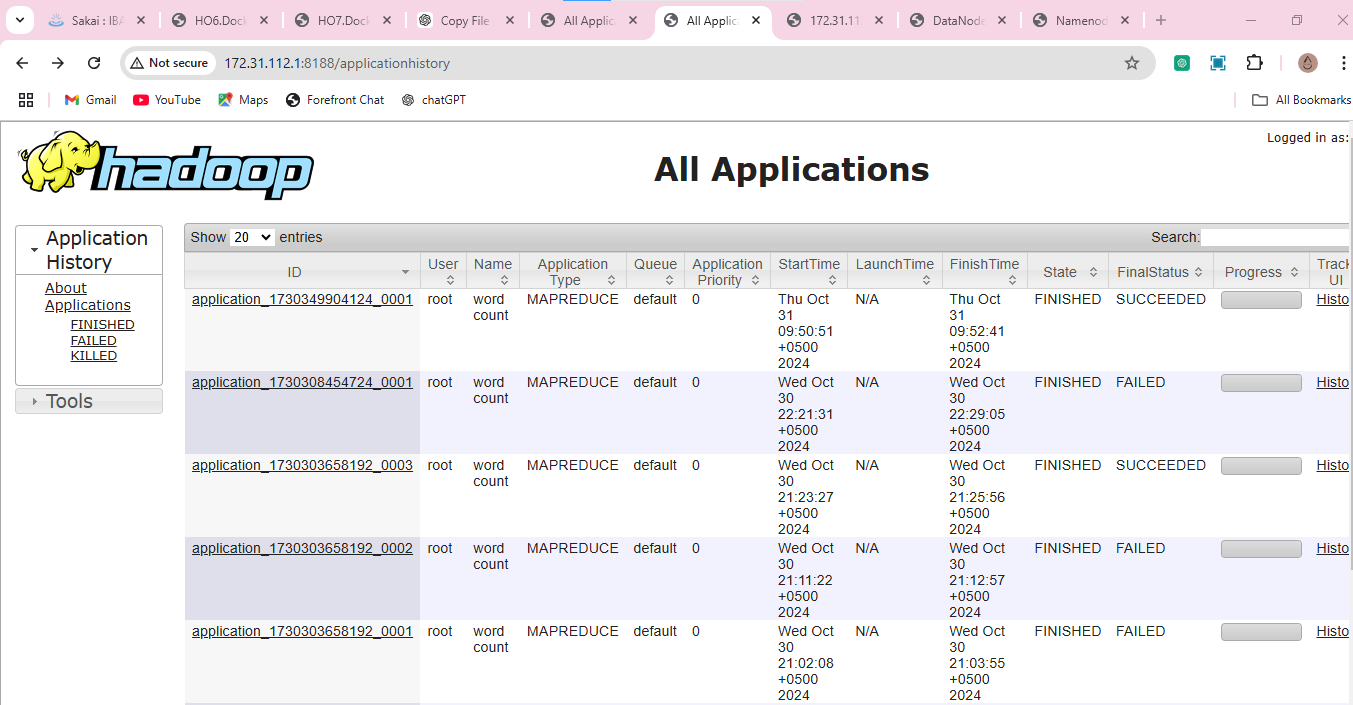
Status of name node and last block as well as amount of volume used out of total volume.

**Node manager**



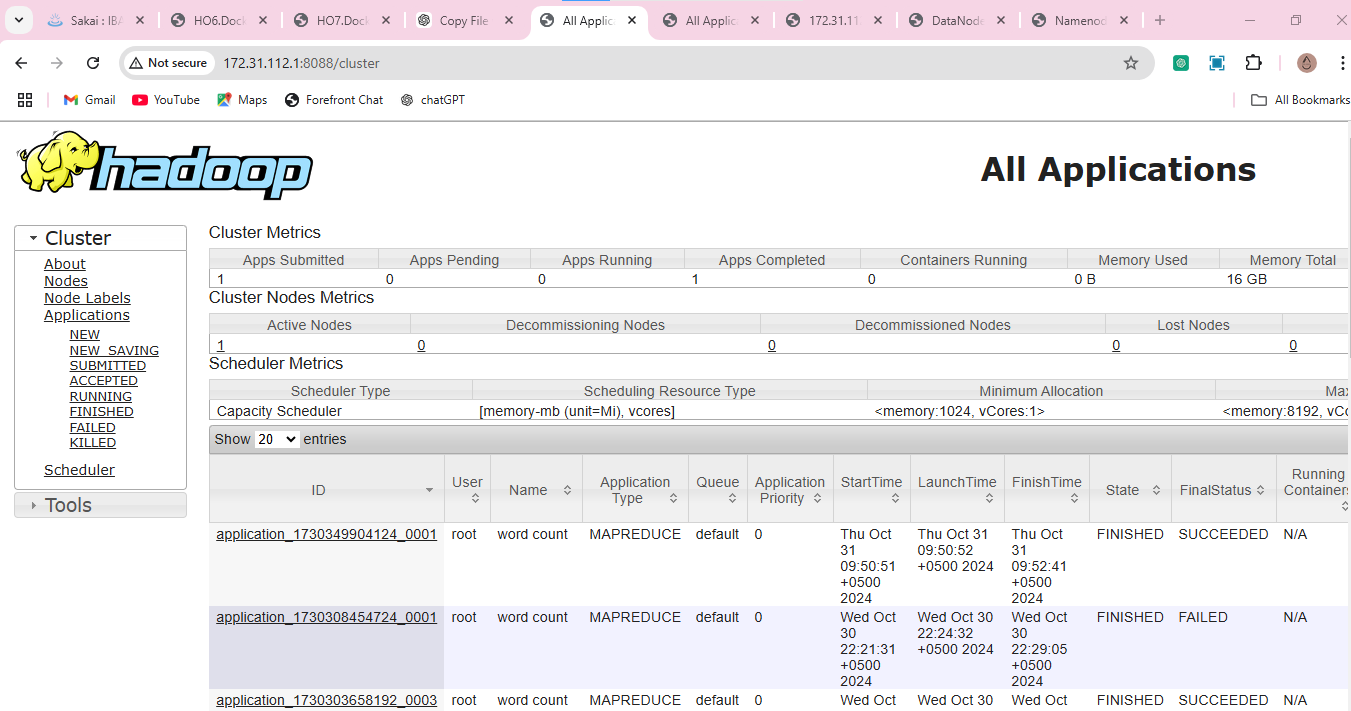
Basic info about virtual memory, physical memory and Vcores allocated for resource management for YARN containers

**History manager**

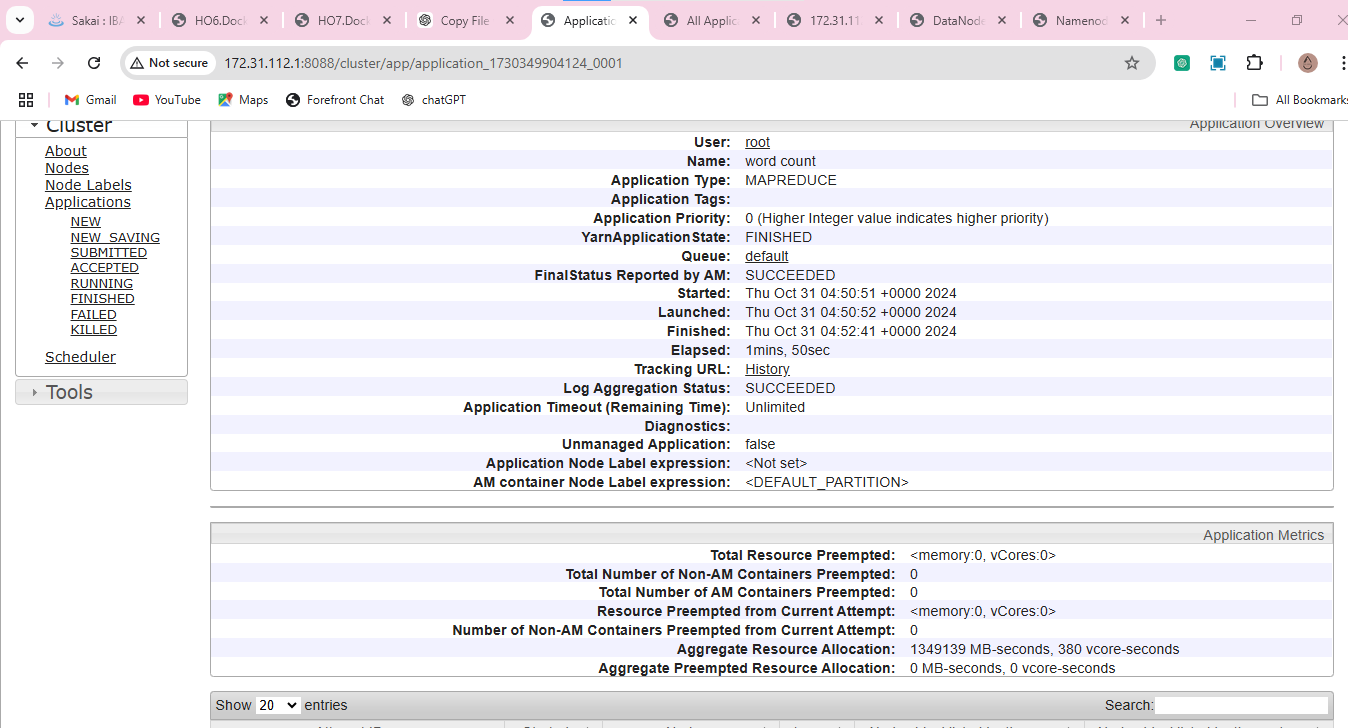


Updated with history of most recent 1gb job

**Resource manager**



Shows metrics of the allocated Vcores, the number of apps submitted



Shows job success or failure, time taken to complete the job, time taken in Vcores.

**Basic hdfs commands:**

**View directory usage**

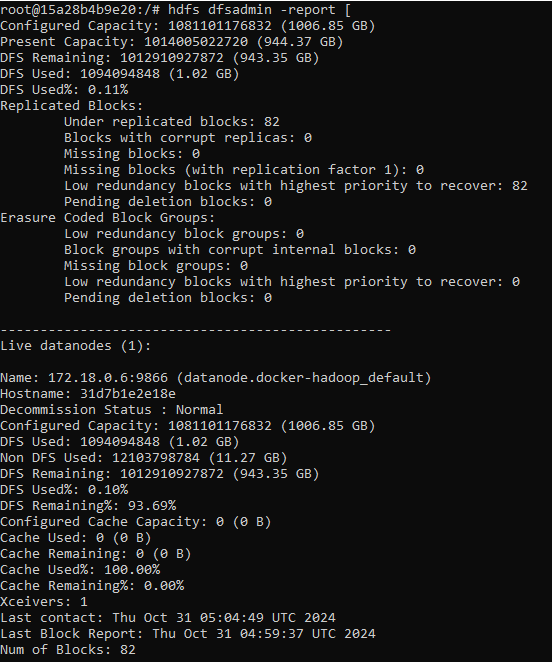
****

**View statistics**

****

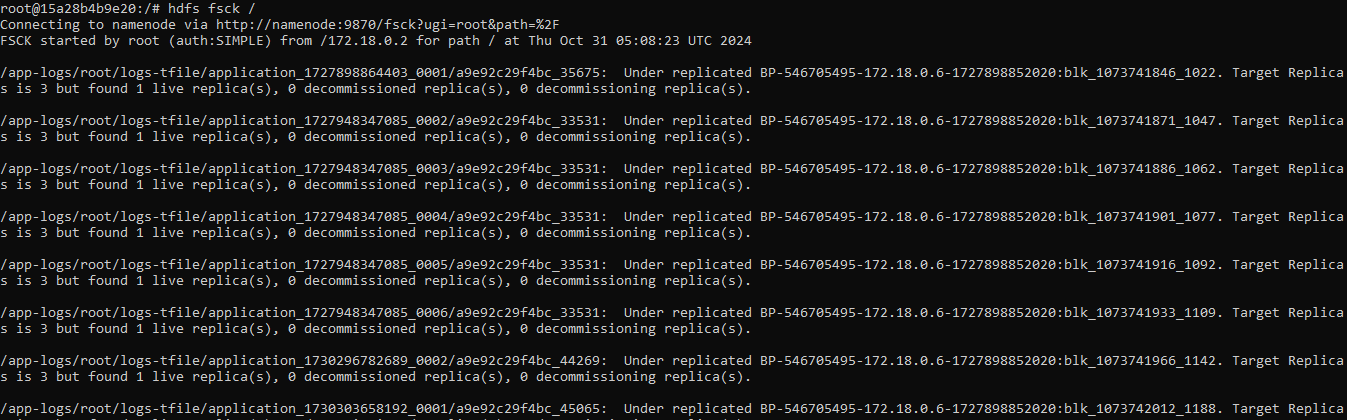
This shows the last performed modification of the inputdata directory

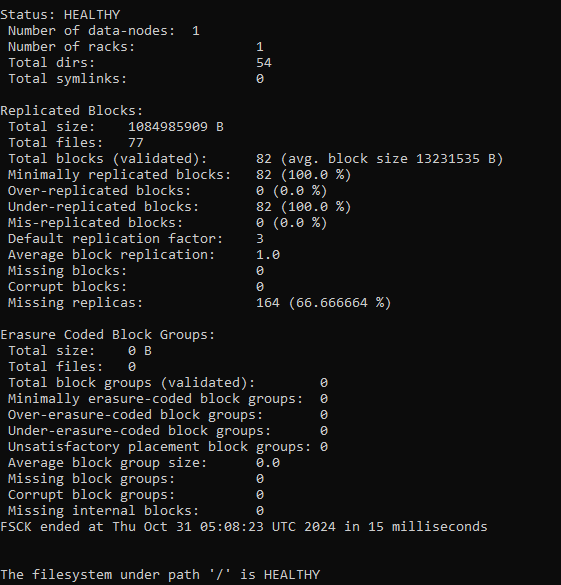
**Check namenode status**



Out of a total of 1006 GB configured capacity for use in HDFS 1.02GB was used in performing the job i.e. 0.1%. 82 under replicated blocks exist, i.e. these don’t have the minimum number of copies across the cluster.

**HDFS file system check:**

****

****

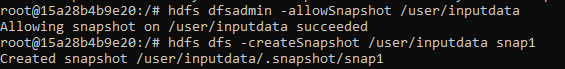
A total of 1 data node with 82 under replicated blocks, or 164 missing replicas, the details above are for each under replicated block

**Listing data nodes**



Shows that 1 data node exists in the cluster along with its ip address, port and hostname

**Creating snapshot**



**Listing snapshot**



**Deleting and checking snapshot**



**Decommissioning and formatting the data node**

Decommissioning will make it so the data node can no longer store or process the data, any data within it will get deleted and this will reduce overall cluster performance.

Formatting will clear its metadata so information about directories, list of data nodes and files etc. Any jobs relying on this info will fail.

**Effect on data upload and query execution**

**Increasing data nodes:**

Data upload time decreases as multiple data nodes enable parallel processing and thus allowing multiple simultaneous data uploads, this also enables better resource usage.

Query execution time similarly also decreases as the parallel processing allows multiple queries to be executed simultaneously making better usage of idle resources.

Increasing the number of data nodes allow for more storage capacity but run the risk of more unreplicated blocks, this can happen in the case of not adjusting the replication factor to cater for the new data nodes.

**Increasing number of mappers:**

Data upload times decrease due to more mappers enabling parallelism, with the increased mappers the system can handle more data at a time thus increasing throughput and resource usage.

More mappers can allow more processing of different parts of the dataset concurrently thus improving resource allocation as well as load distribution thus decreasing query execution time.

Increased mappers can mean reduced time to write into HDFS, however it also means more and more write loads which may lead to potential bottlenecks. In terms of unreplicated blocks, if the replication happens slower than data being written, and a node fails before being replicated, it could lead to data loss.

**Increasing number of reducers:**

In terms of data upload time, it isn’t directly affected.

Query execution time on the other hand greatly benefits due to allowing parallelism and distributing the workload across multiple reducers thus decreasing latency and optimizing resource usage.

Reducers already suffer the most when it comes to unreplicated blocks and cause latency, increasing them would only add to this more if such blocks aren’t decreased.