Why HDFS?

Liberation

essence of HDFS

1. Distributes data across multiple nodes / machine
2. Replicates data to ensure fault tolerance
3. Provides efficient access for Big Data Processing

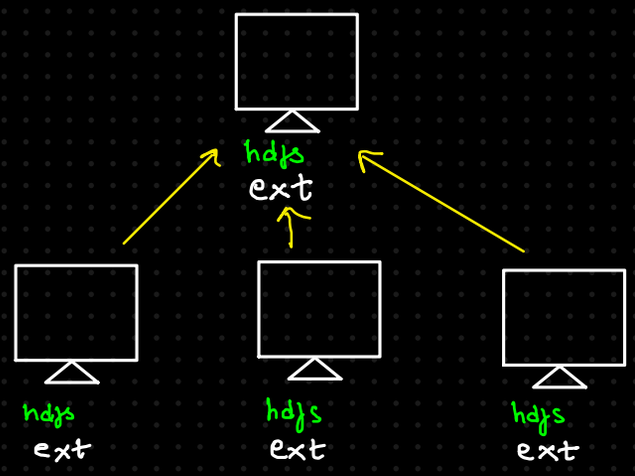
HDFS architecture

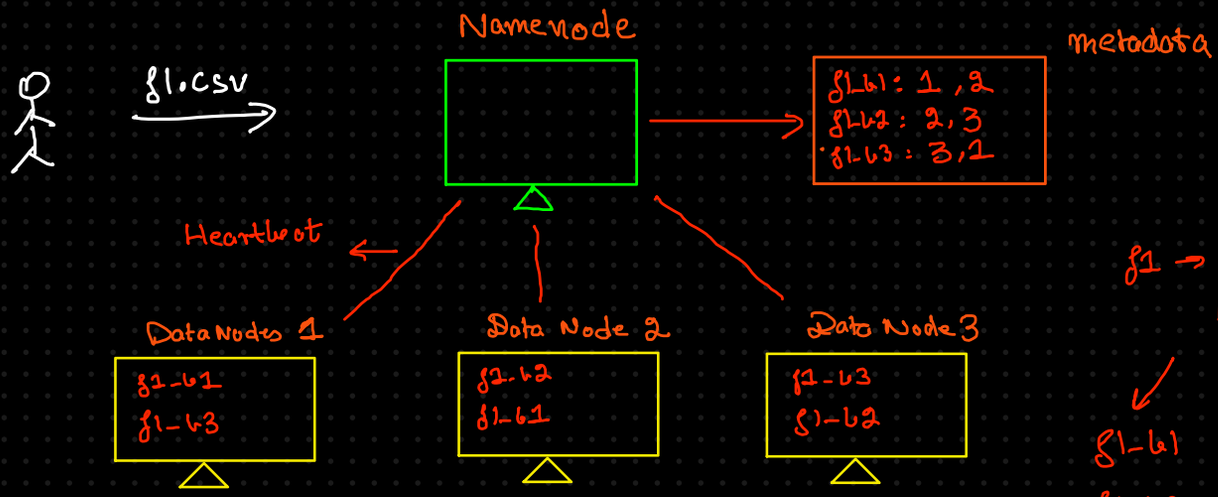
machine cluster

hadoop

dataproc

Master - Worker Architecture





metadata

f1 → 300mb

128 mb

f1-b1

f1-b2

f1-b3

Replication factor = 2

Name node → librarian

Data nodes → shelves

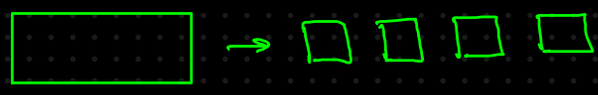
Data blocks → books

Aspect	NameNode	DataNode
Definition	The master node in the HDFS architecture. It manages metadata and file system namespace.	The worker nodes in HDFS. They store the actual file data in blocks.
Purpose	Keeps track of file locations, file metadata, and directory structure (but does not store data).	Responsible for storing, retrieving, and replicating data blocks.
Role	Coordinator – directs how files are stored and accessed across DataNodes.	Executor – performs read and write operations as instructed by the NameNode.
Data Stored	Metadata (e.g., file names, permissions, and locations of data blocks).	Actual data split into blocks.
Dependency	HDFS cannot function without a NameNode. It is the single point of failure in non-HA setups.	HDFS can function with one or more DataNodes, and their failure does not stop the system.
Communication	Communicates with clients for file operations and with DataNodes to track block statuses.	Communicates with the NameNode for block operations and periodically sends heartbeat signals.
Fault Tolerance	Does not store redundant copies (replication is for data blocks only).	Handles fault tolerance through block replication across multiple DataNodes.
Processes	Maintains and updates the file system namespace dynamically.	Regularly reports to the NameNode with block reports and health checks.
Example Role	Acts like a **manager in a library**, knowing the exact shelf (DataNode) where a book (block) is stored.	Acts like **the shelves in the library**, holding and managing the physical books (blocks).

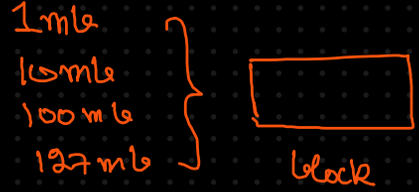
Blocks in HDFS

- a block is the smallest unit of storage in HDFS
- Each file stored in HDFS is divided into blocks, which are then distributed across multiple data nodes.

default size - 128 mb



Name node has less metadata to manage



1. efficient for large files
2. Distributed storage and computation

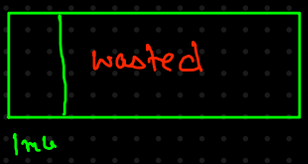
Q. Can we change the block size in our Hadoop cluster?

Yes.

1. Decreasing the block size

128 mb → 1mb

500mb → □ * 500



help in increasing parallelism but also metadata will increase

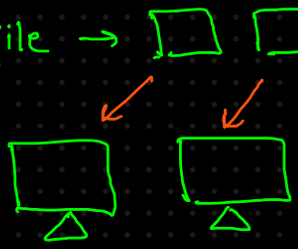
2. increase the block size:

128mb → 512 mb

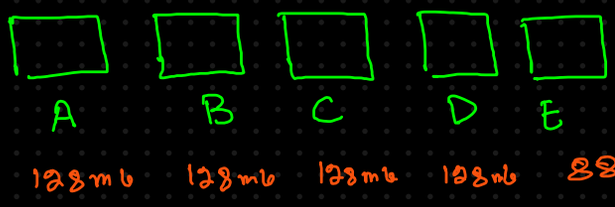
less parallelism

⇒ less metadata

1gb file →



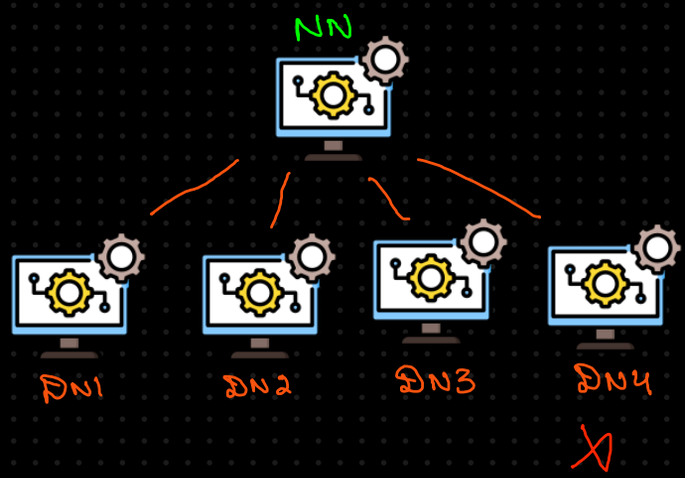
600 mb file →



900mb file →

129 mb file last block size of 64 mb →

Data Node (Worker) failure in HDFS



DN are susceptible for failure

Replication in hdfs

default RF = 3 — 1 original, 2 copies

1gb → 3gb

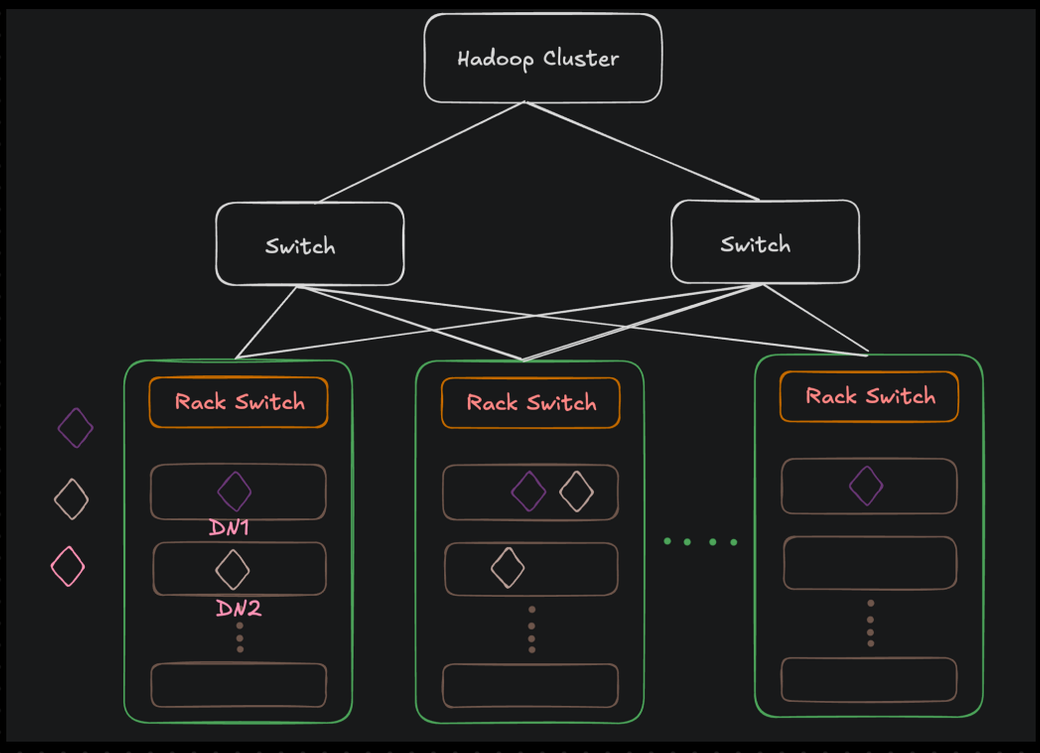
300mb → A B C

4 DataNodes

Rack awareness in HDFS

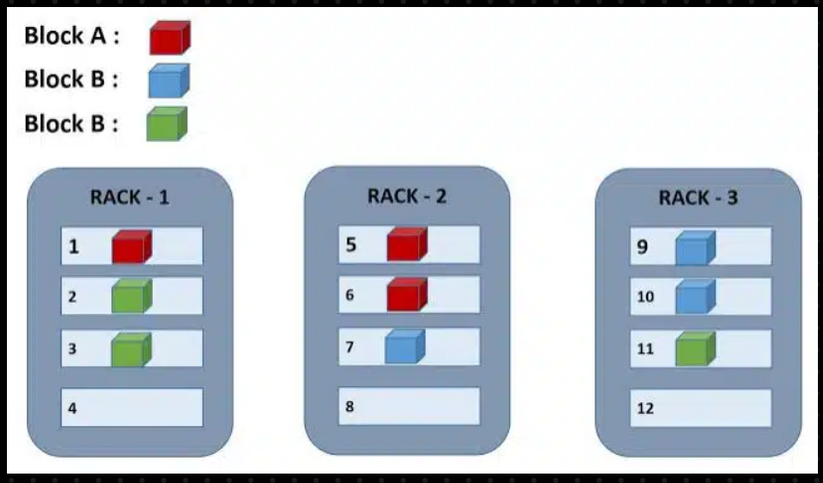
A rack is a collection of Datanodes that are physically close together. often in the same network switch.

rack awareness algorithm ensures that replicas are distributed across different racks and not just different DataNodes



Fault tolerance

Optimized Network Topjic

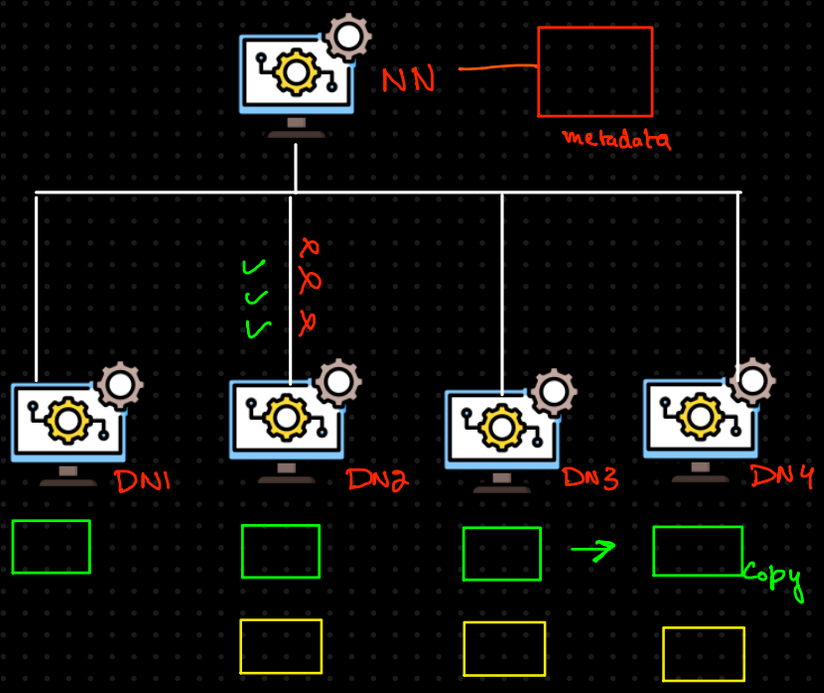


Data Node Failures in Hadoop Cluster

Temporary failure

DN is unavailable due to some transient issue.

1. Network outage
2. Software crashes
3. Node reboot maintenance



→ heartbeat
→ block report

1. Detection 10.5 minutes
2. Replication
   - 1 green block
   - 1 yellow
   - under-replicated
3. when DN recovers

   metadata is updated to ensure consistency after the excess replicas are deleted

Library

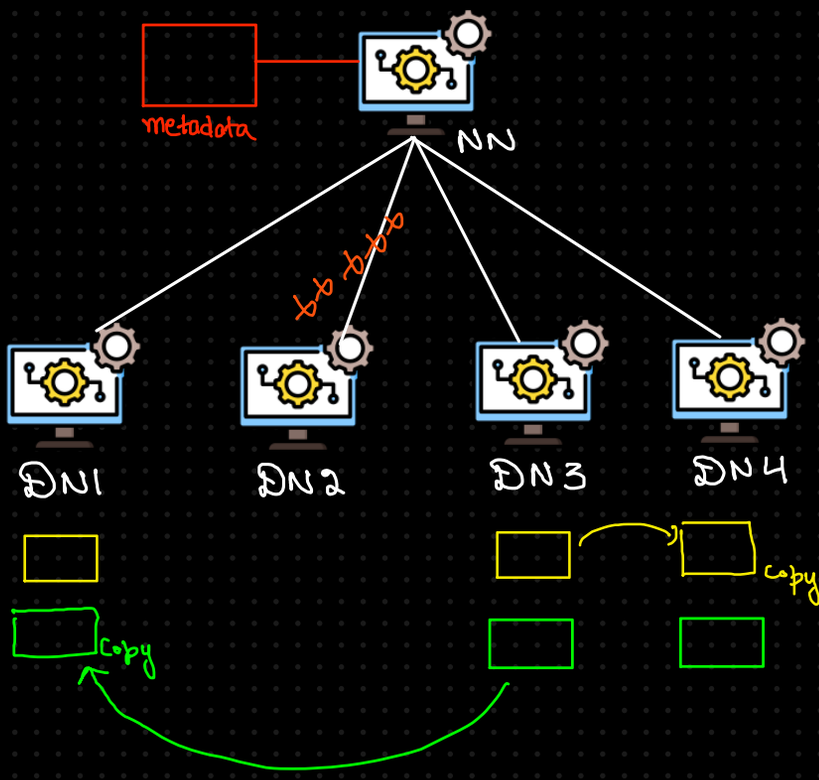
book → block
DN → rock

Permanent Failure

When a data node is permanently lost.

1. Hardware failure
2. Disk corruption
3. Decommissioning of a node

> 10.5 minutes



1. Detection

2. Replication

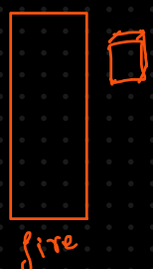
   blocks on DN2 are marked lost

3. Machine comes back

   The m/c is not trusted

   data node can get reintegrated in our system as a fresh node ie. blocks are ignored and cleared up to avoid data inconsistency.

Library



Aspect	Temporary Failure	Permanent Failure
Failure Cause	Network issues, software crashes, or reboots.	Hardware failures, disk corruption, or decommissioning.
Detection	DataNode stops sending heartbeat signals; marked as temporarily unavailable.	DataNode stops sending heartbeat signals for an extended period; marked as dead.
Replication Trigger	Blocks on the failed DataNode are marked as "under-replicated," and new replicas are created.	Blocks on the failed DataNode are treated as lost, and new replicas are created.
Returning DataNode Behavior	Sends a block report to the NameNode. Excess replicas are deleted to restore the replication factor.	Returning DataNode is not trusted. Its blocks are treated as stale and eventually cleaned up.
Metadata Updates	Metadata dynamically updates to include new replicas and remove excess copies.	Metadata permanently removes blocks from the failed DataNode and updates mappings for new replicas.
Library Analogy	A locked bookshelf temporarily restricts access. Extra copies are reconciled when it becomes available.	A destroyed bookshelf is permanently removed from the catalog, and its books are replaced elsewhere.

1. HDFS handles failures gracefully
2. Metadata is the backbone of HDFS
3. Replication factor is strictly maintained.

Name Node failure

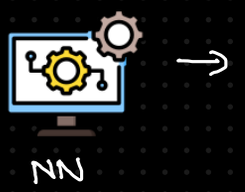
The namenode is the most critical component in the HDFS as it manages the metadata and namespace of the filesystem.

Single point of failure (SPOF)

Standby NN

Secondary Name node

it helps in checkpointing & size management of our primary datanode.



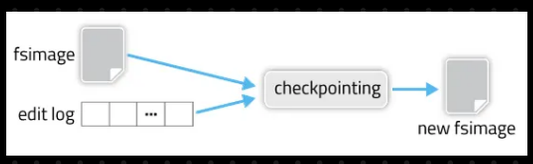
register

new block addition
replication
deletion etc.

Edit logs

Secondary Name Node

1. checkpointing
2. size mgmt



Combining → fsimage'

Secondary NN is not a failover mechanism

Works fine if we can afford downtimes.

Standby Name node

Hadoop 2.x

it is a hot backup for your NN

High availability

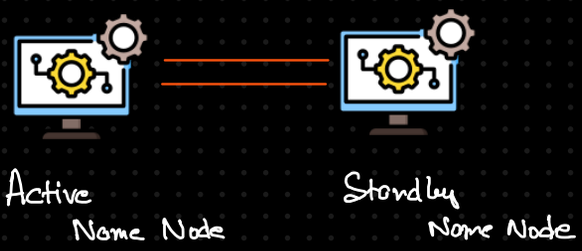
→ Constant connection with master

→ makes the copy of namespace image in memory

Secondary NN

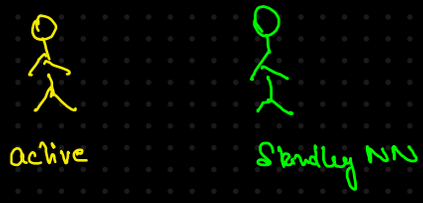
1. Checkpointing
2. Size management

⊖ Still our NN / master is the SPOF



1. High availability : it can take the role of active NN when required.
2. Failover mechanism : Seamless takeover ensuring no data loss and minimum downtime.
3. Synchronization : all edit logs are applied & Standby NN always up-to date

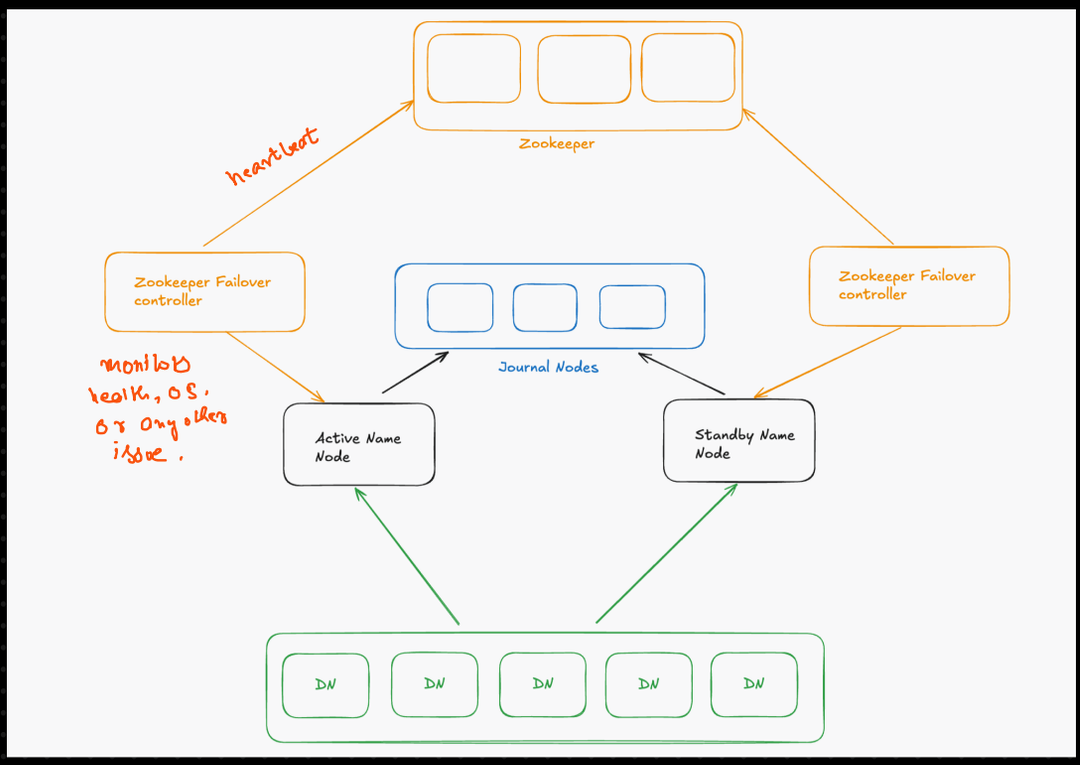
library



Aspect	Secondary NameNode	Standby NameNode
Role	Helper for checkpointing and edits log management.	Hot backup for Active NameNode, enabling seamless failover.
High Availability	Does not provide HA; cannot replace the NameNode. NN is SPOF	Ensures HA by taking over the Active NameNode's role on failure.
Synchronization	No continuous synchronization; works periodically.	Continuously synchronized with the Active NameNode.
Failover Capability	Not capable of failover; only provides updated fsimage.	Fully capable of taking over in case of Active NameNode failure.
Introduced In	Hadoop 1.x	Hadoop 2.x

Hadoop High Availability (HA) Architecture

High availability architecture ensures the continous operation of the hadoop cluster even if the active Name node fails.



→ high availability
→ minimal downtime

Critical for a production grade cluster

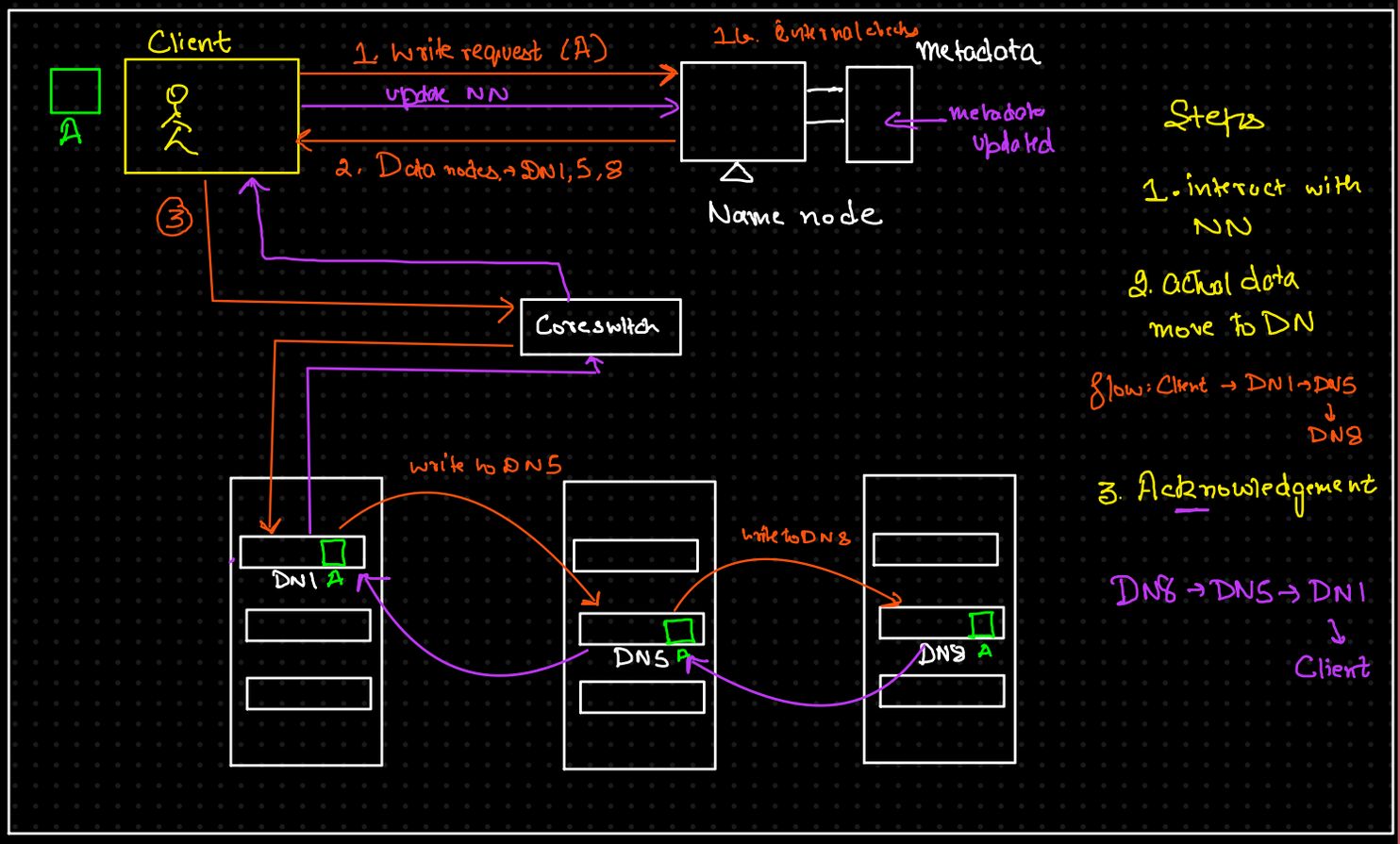
Component	Role	Explanation
Active NameNode	Main controller managing HDFS namespace and block information.	Handles all client operations like file creation, deletion, and metadata management.
Standby NameNode	Hot backup of the Active NameNode, ready to take over during failure.	Synchronizes its state with the Active NameNode using edit logs stored in JournalNodes.
DataNodes (DN)	Store actual data blocks and handle data read/write operations.	Send regular block reports to both Active and Standby NameNodes.
JournalNodes (JN)	Maintain a shared edit log that keeps both NameNodes synchronized.	Facilitate consistent updates to the Standby NameNode to ensure it has the latest changes in the HDFS namespace.
Zookeeper (ZK)	Coordinates failover and prevents split-brain scenarios.	Monitors NameNode states and triggers leader election during failover.
FailoverController	Monitors the health of NameNodes and initiates failover.	Works with Zookeeper to perform automatic failover when the Active NameNode is unresponsive.
Heartbeat Mechanism	Periodic signals exchanged between components to check health.	Ensures timely detection of failures in NameNodes or DataNodes.
Block Reports	Periodic reports from DataNodes to NameNodes about stored blocks.	Ensures both Active and Standby NameNodes have accurate information about the location of data blocks in the cluster.
Clients	Users or applications interacting with the Hadoop cluster.	Access HDFS files through the Active NameNode. Failover is transparent to clients, ensuring uninterrupted service.

we can also use NFS (Network File Storage)

Split-brain scenario → ensuring only one Active NN thus preventing conflict

Write operation in HDFS

using hdfs as a distributed data storage



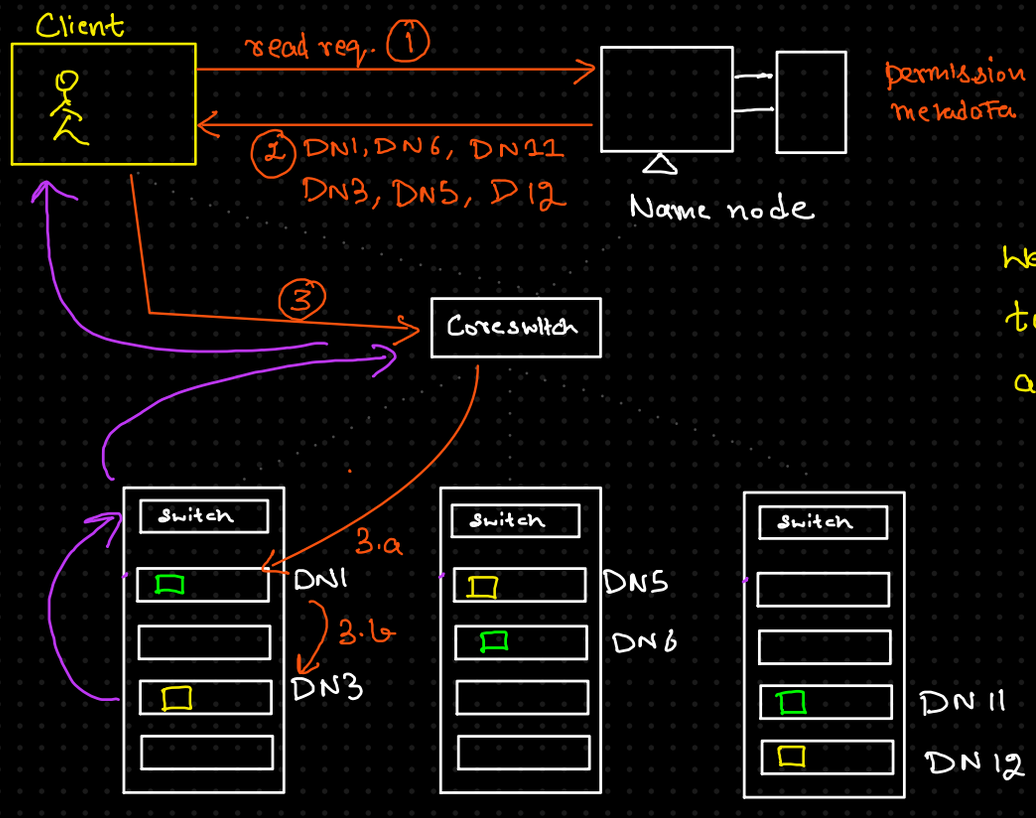
Fault tolerance & failure handling

1. DN fails during the write.
2. Client fail
3. Network / Rack failure
4. Name node failure
5. DN failure

Read operation in HDFS

The process of reading the data kept on the HDFS cluster

Read request does not require an acknowledgement



We don't need to read data from all the copies.

Google cloud platform

cloud.google.com

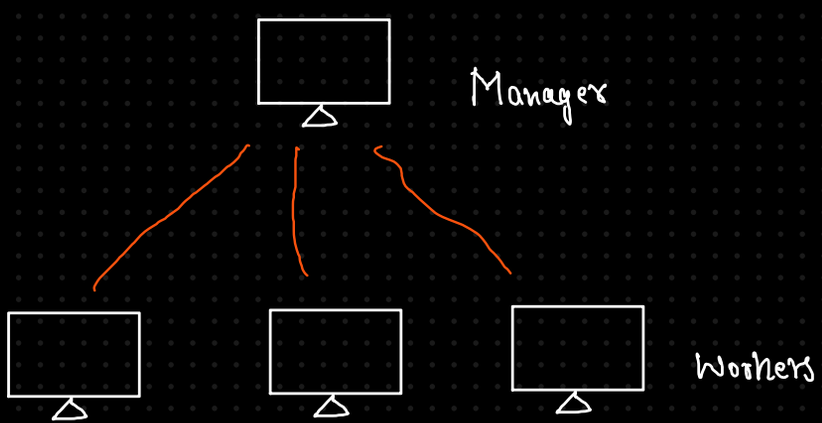
Please make sure to first focus on getting the service we want work perfectly.

Cluster type

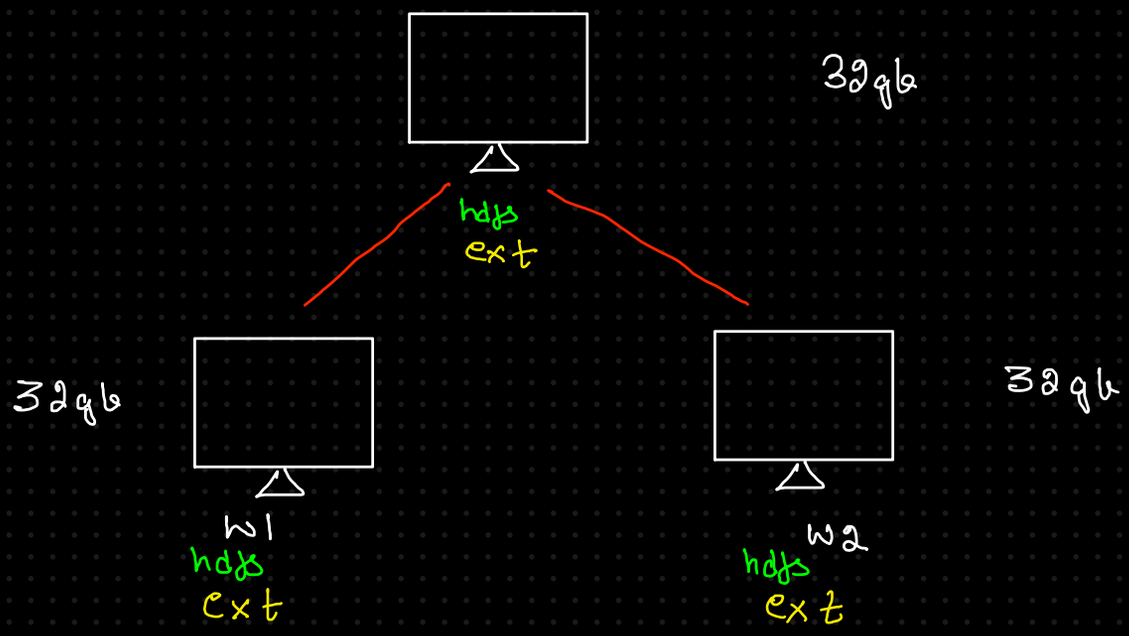
- (•) Standard (1 master, N workers)
- ( ) Single Node (1 master, 0 workers)
  Provides one node that acts as both master and worker. Good for proof-of-concept or small-scale processing
- ( ) High Availability (3 masters, N workers)
  Hadoop High Availability mode provides uninterrupted YARN and HDFS operations despite single-node failures or reboots

⇒ Pseudo distributed cluster

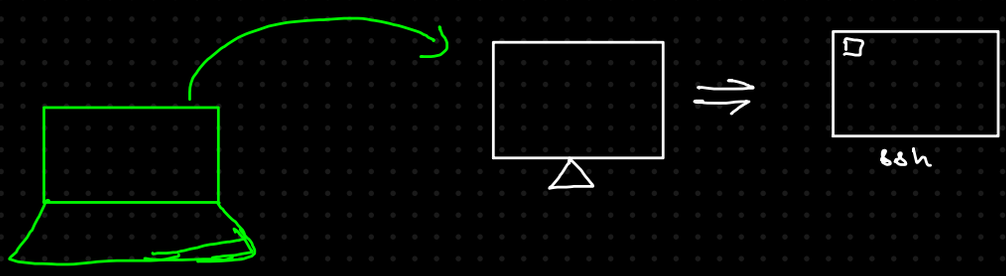
Autoscaling



More on Google dataproc Hadoop cluster



Some internal space will be used by machine for OS, drivers.



linux
mac
] -> terminal

windows -> cmd / powershell

Linux Commands

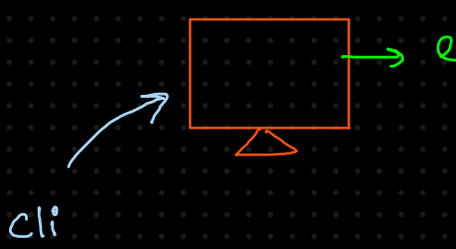
As a data engineer, proficiency in linux command is crucial for

- managing data pipelines
- handling files
- working with distributed systems

Windows Linux Macos

gcp → 3 machines debian os

1M 2W



→ very lightweight
→ easy to install & maintain

terminal → linux & macos

Windows terminal / Ubuntu subsystem / github cli

clear

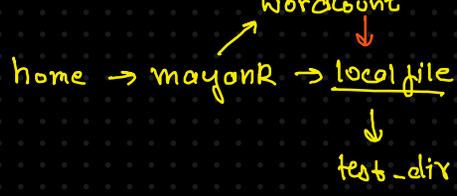
1. Navigation Command

(a) pwd → print working directory

(b) cd → change directory

absolute vs relative path

tab helps to autocomplete



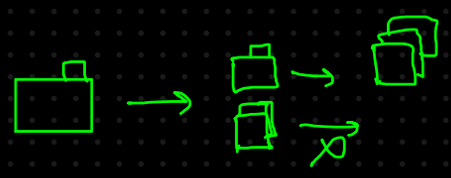
(c) directory listing

ls

blue → folder/dir

green → executable files

black → normal files



```
drwxr-xr-x 2 mayank0953 mayank0953 4.0K Jan 11 05:12 wordcount
-rw-r--r-- 1 mayank0953 mayank0953 231M Jan 11 15:27 logfile.txt
drwxr-xr-x 2 mayank0953 mayank0953 4.0K Jan 12 03:35 wordcount_two_reducers
drwxr-xr-x 2 mayank0953 mayank0953 4.0K Jan 12 04:09 wordcount_no_reducer
drwxr-xr-x 3 mayank0953 mayank0953 4.0K Jan 13 09:57 localFile
-rw-r--r-- 1 mayank0953 mayank0953    0 Jan 13 10:04 test.bat
-rwxr-xr-x 1 mayank0953 mayank0953    0 Jan 13 10:07 test.exe
```

d / -

rwx r-x r-x

owner group others

r → read (4)

w → write (2)

x → execute (1)

chmod

More linux Commands

File operations

Move

mv filename final location

Remove

rm filename

rm -r dir

Copy

cp source destination

File → create and view content

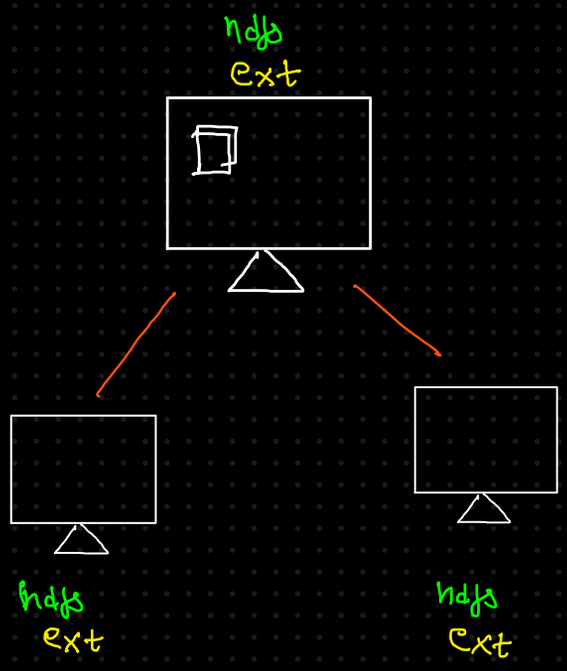
→ please refer here for complete details on command discussed

https://allthingdata.substack.com/p/essential-linux-commands-for-data

HDFS commands

hadoop fs

hdfs dfs



Copy data from local to hdfs

hadoop fs -put

-copy from local

file name

destination

https://allthingdata.substack.com/p/essential-hdfs-commands-for-data