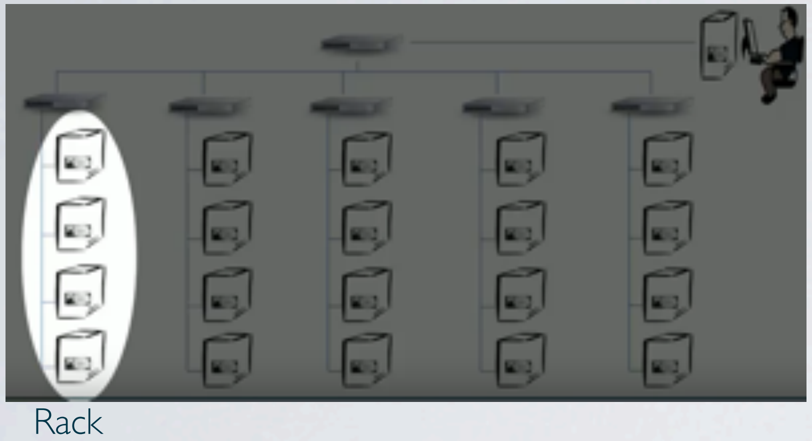
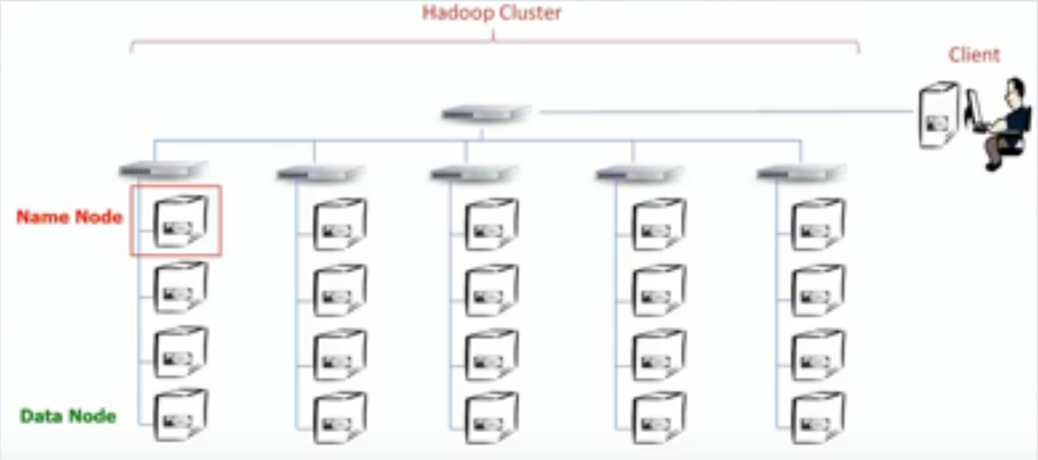
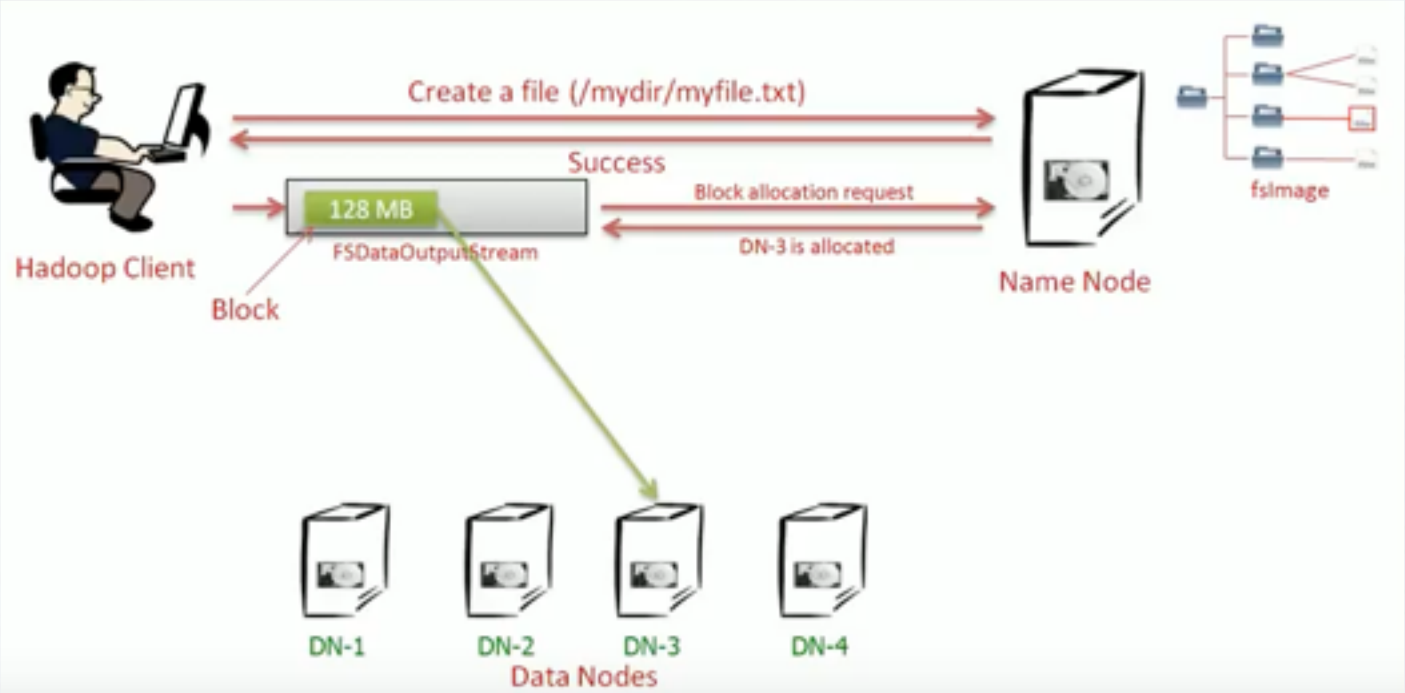
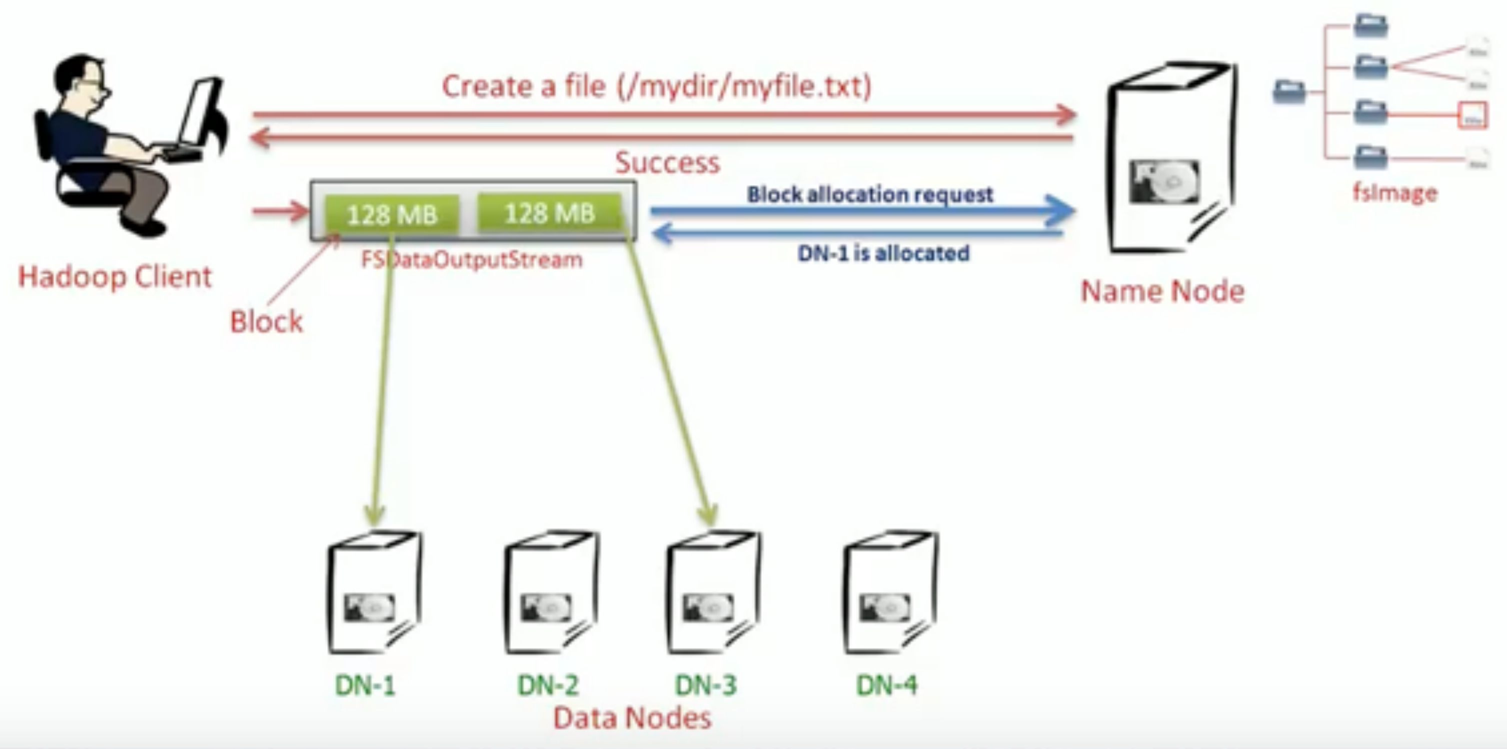
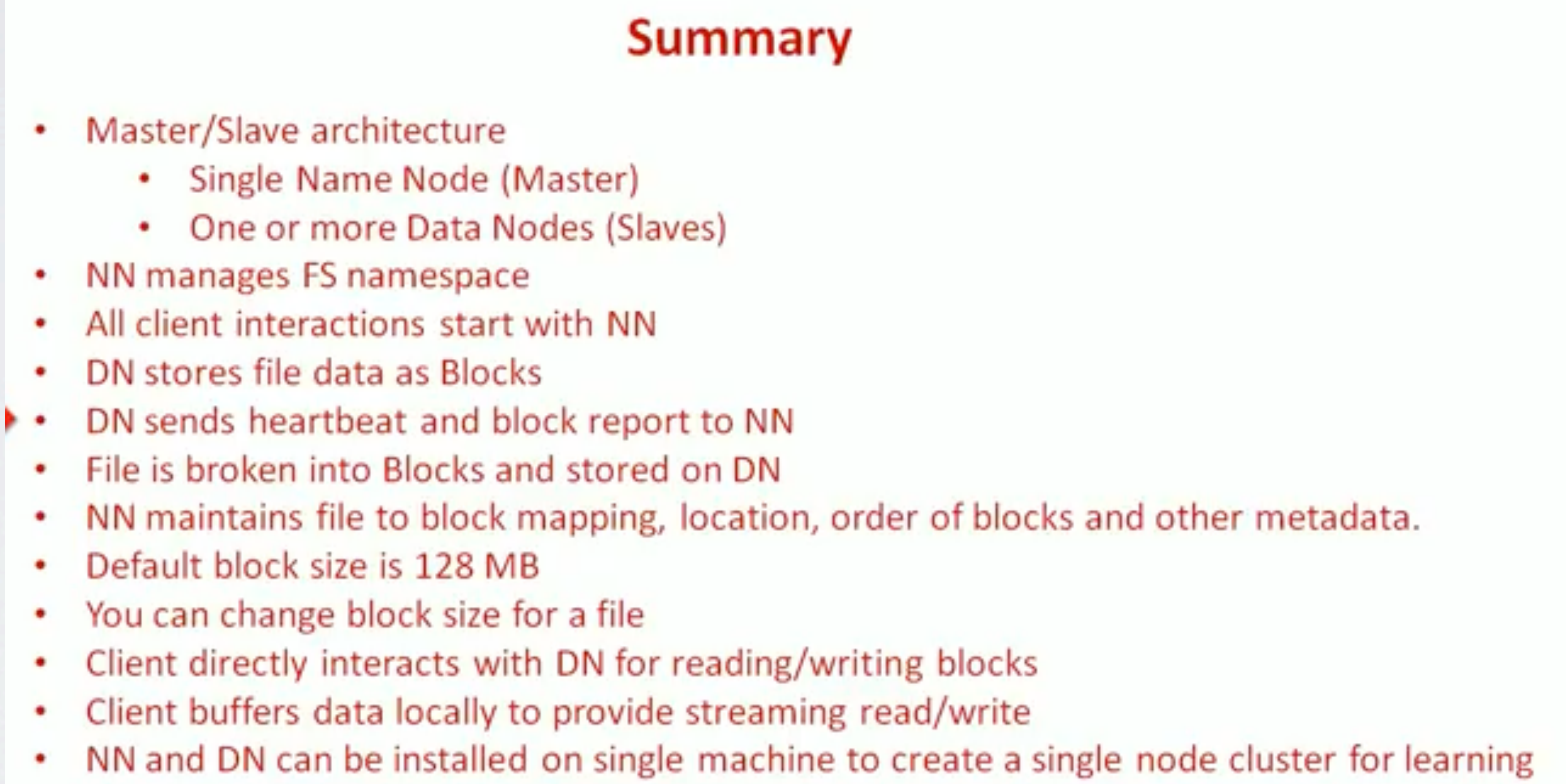
Hadoop


Rack


Hadoop Cluster
Client
Name Node
Data Node


Create a file (/mydir/myfile.txt)
Success
128 MB
FSDataOutputStream
Block allocation request
DN-3 is allocated
Hadoop Client
Block
Name Node
fsImage
DN-1
DN-2
DN-3
DN-4
Data Nodes

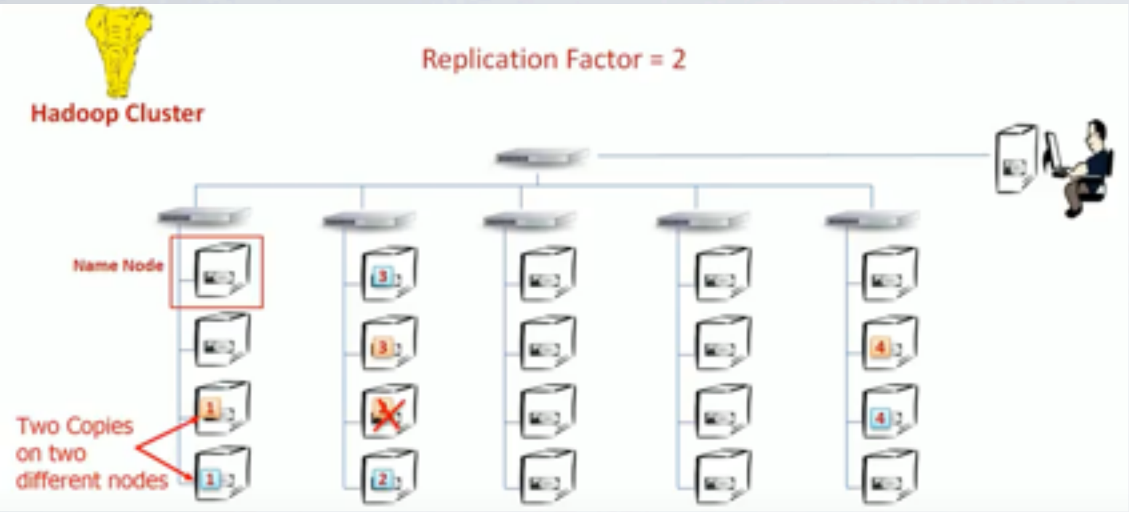




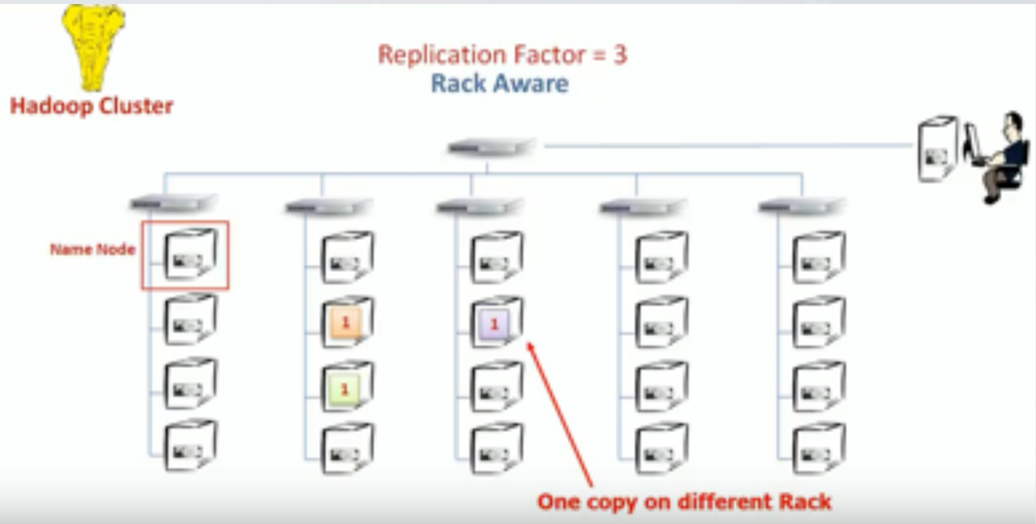
Summary

- Master/Slave architecture
  - Single Name Node (Master)
  - One or more Data Nodes (Slaves)
- NN manages FS namespace
- All client interactions start with NN
- DN stores file data as Blocks
- DN sends heartbeat and block report to NN
- File is broken into Blocks and stored on DN
- NN maintains file to block mapping, location, order of blocks and other metadata.
- Default block size is 128 MB
- You can change block size for a file
- Client directly interacts with DN for reading/writing blocks
- Client buffers data locally to provide streaming read/write
- NN and DN can be installed on single machine to create a single node cluster for learning

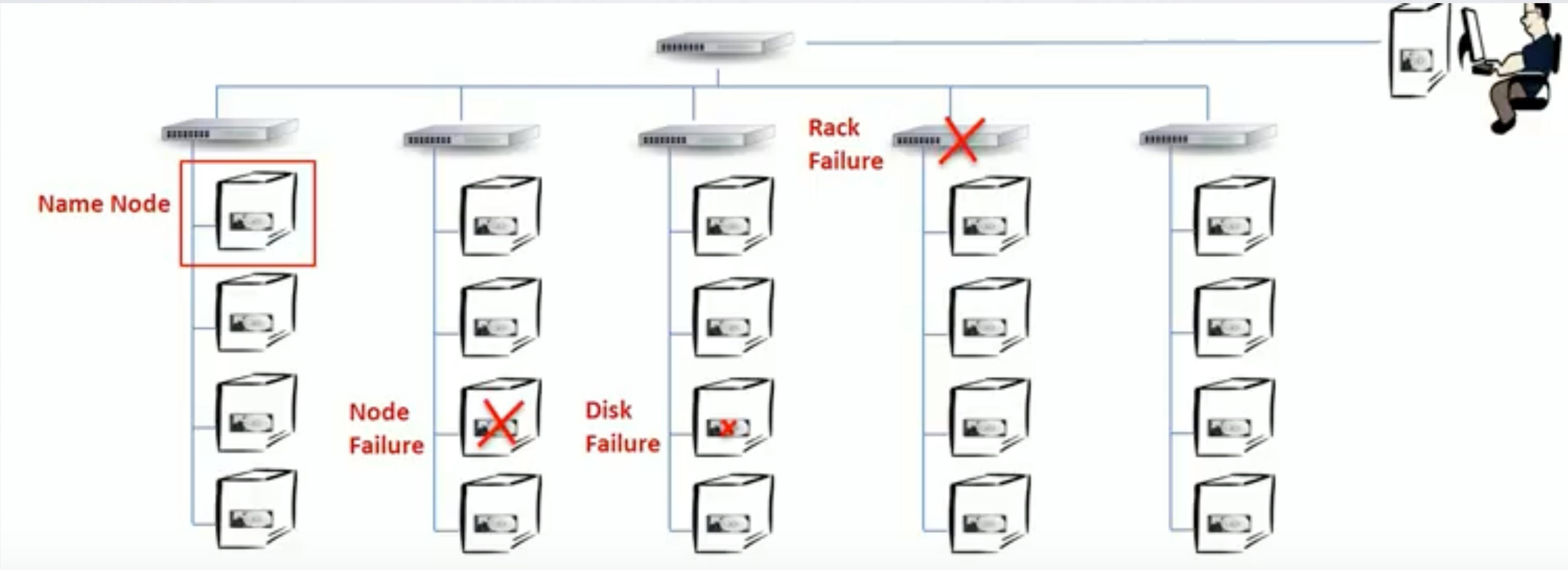
To keep multiple copies of data block or files is for fault tolerant purpose. So here we use replication factor = 2 copies



If you make copies on same racks machines then whole rack may fail. so make copies on different rack just in case whole rack fails..

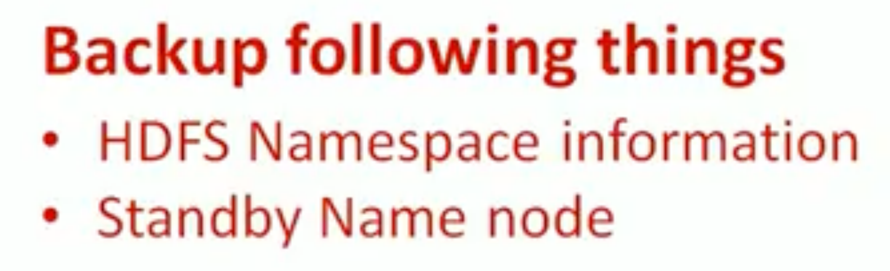


So basically failure of data node can be different types



High availability is about protecting name node.. if NN fails ,
whole cluster down. Fault tolerant is about data node failure.

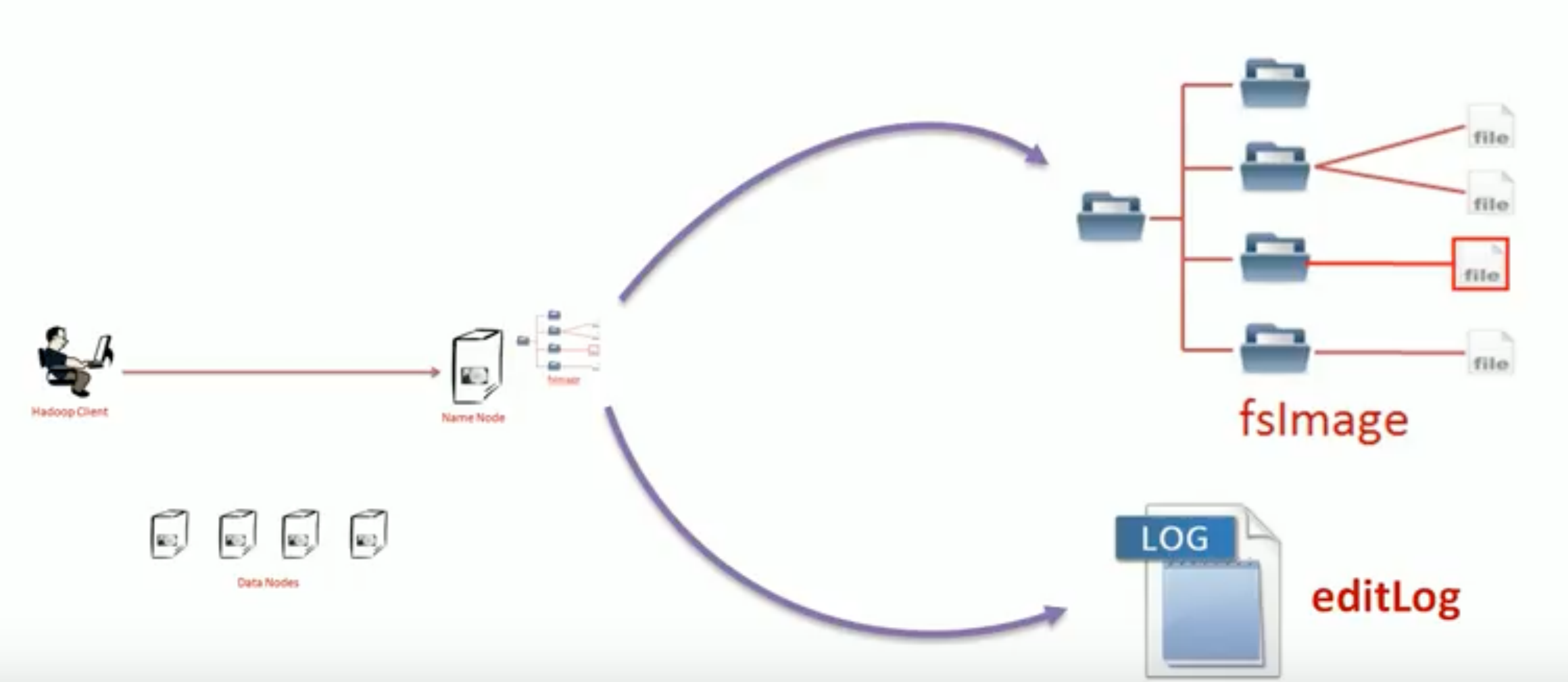
High availability is about protecting name node.. if NN fails ,
whole cluster down so back up is a solution



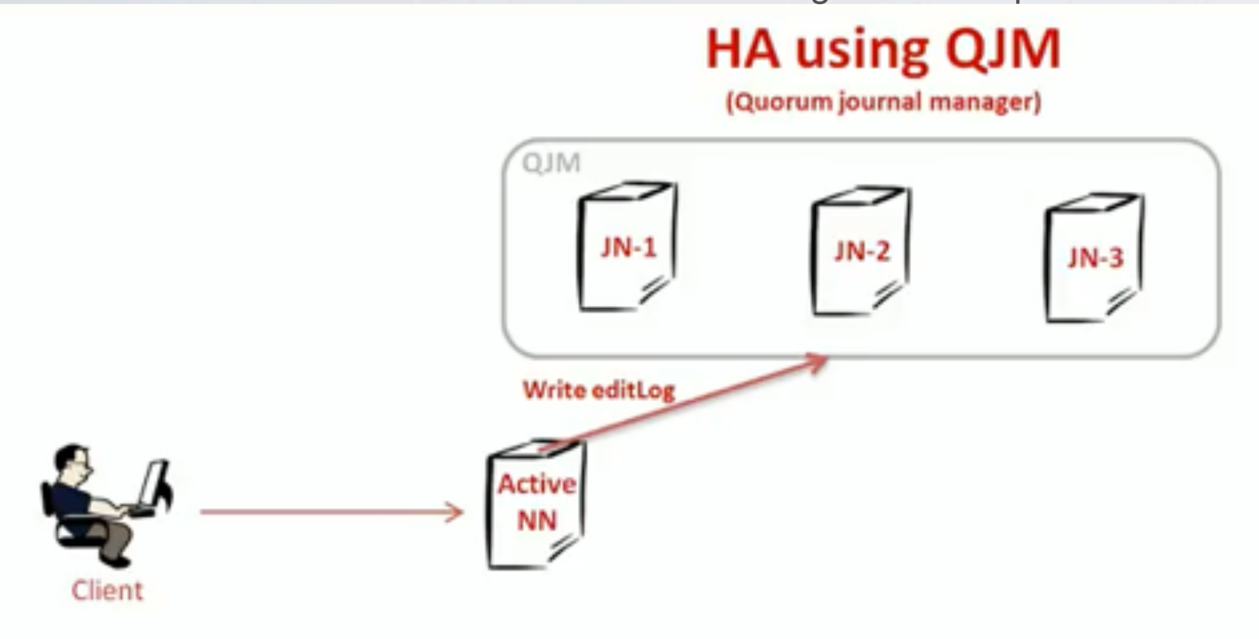
Backup following things

- HDFS Namespace information
- Standby Name node

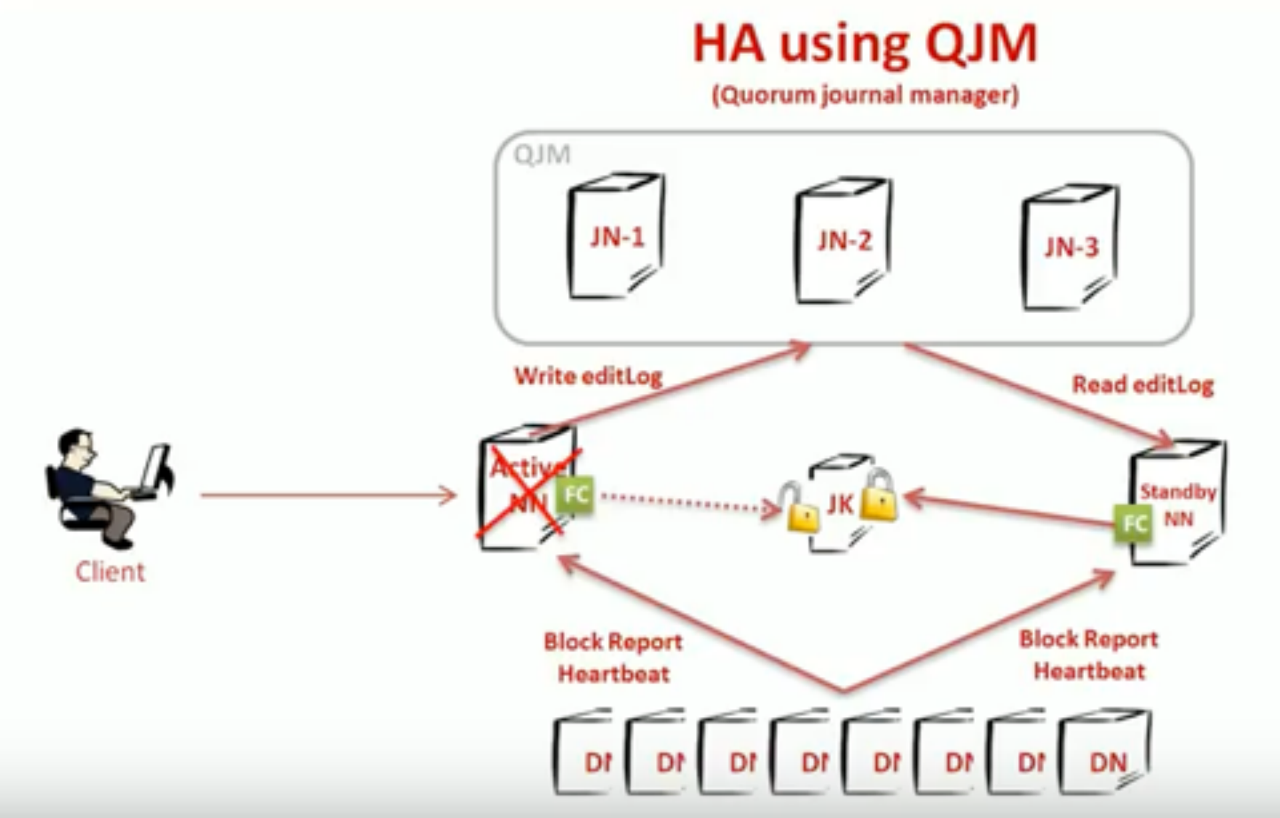
Editlog is like change log or journal log of what happens to NN
and it has fsimage so both these 2 can be backed up



QJM is the solution to take backup of NN into at least 3 machines as shown below. There is small journal daemon software runs on active NN which takes regular back up

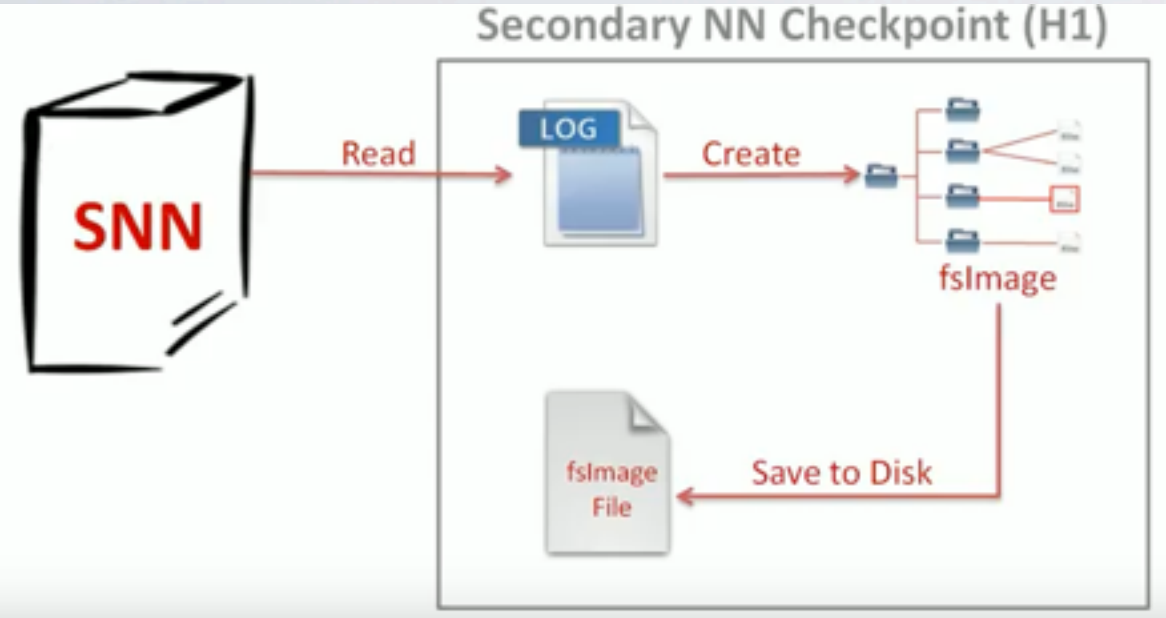


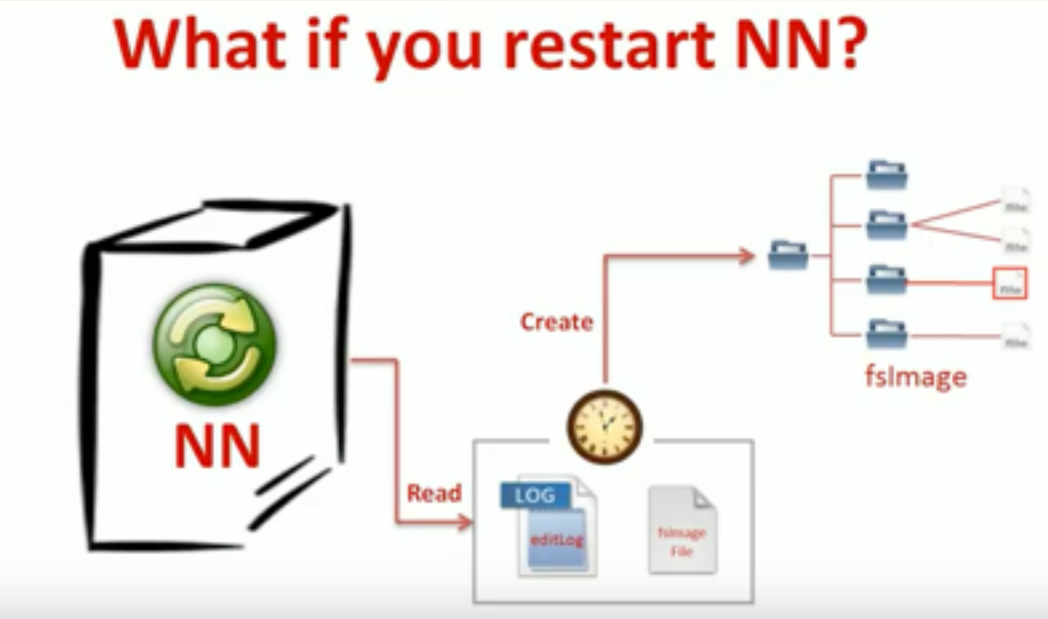
Second approach is using standby NN which reads log of failure from Zookeeper (which is named JK box in middle) . This standby NN also gets heartbeat from data node.



What is secondary name node ?

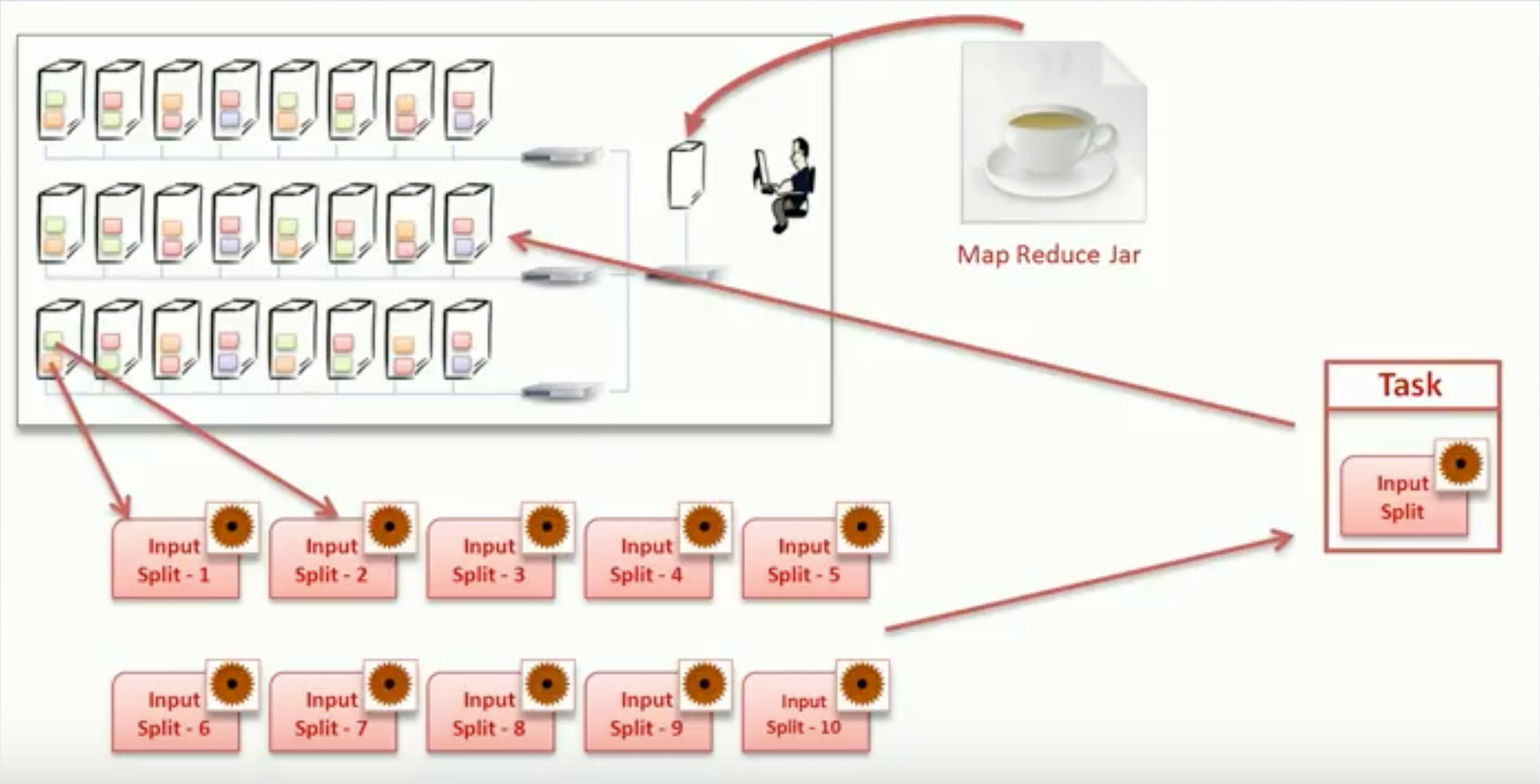
NN has 2 component, fsimage and editlog. All changes are in editlog and all file namespace are in fsimage. Let’s say fsimage fails , in that case , NN will pull info from editlog which can take several hours so there will be downtime. To avoid that we use secondary NN to keep reading editlog and building fsimage every hour so that if main NN fails then it can use that fsimage built by secondary NN. So as shown below, secondary NN will have fsimage file ready into disk every hour which can be used if main NN fails.



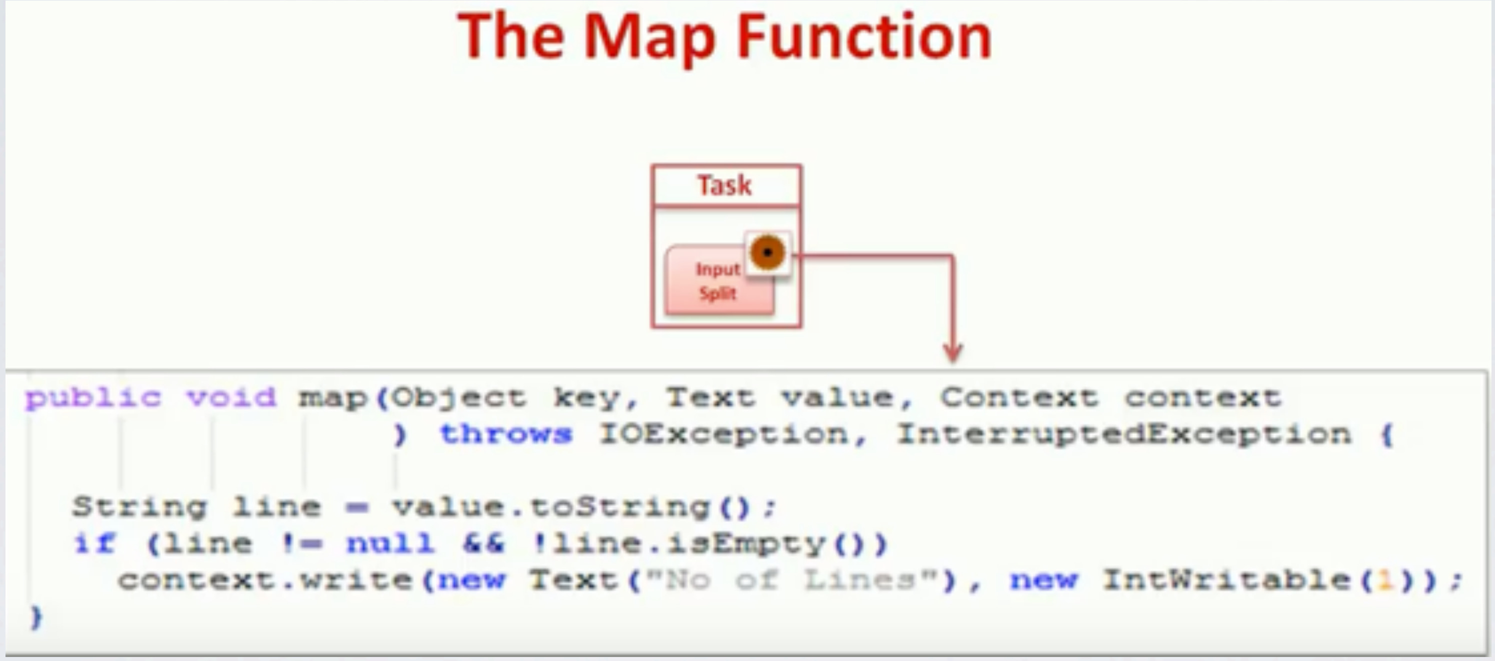


Secondary NN is not required for high availability concept because standby NN perform this checkpoint and creating fsimage activity so secondary NN is not required

As shown below, we have mapreduce code /jar is executed on mapreduce framework using yarn. Here lets say we divided 20 TB file into data nodes and key replication factor -2 . Now, we also split file into small blocks or input split and then ran map function on each input split.



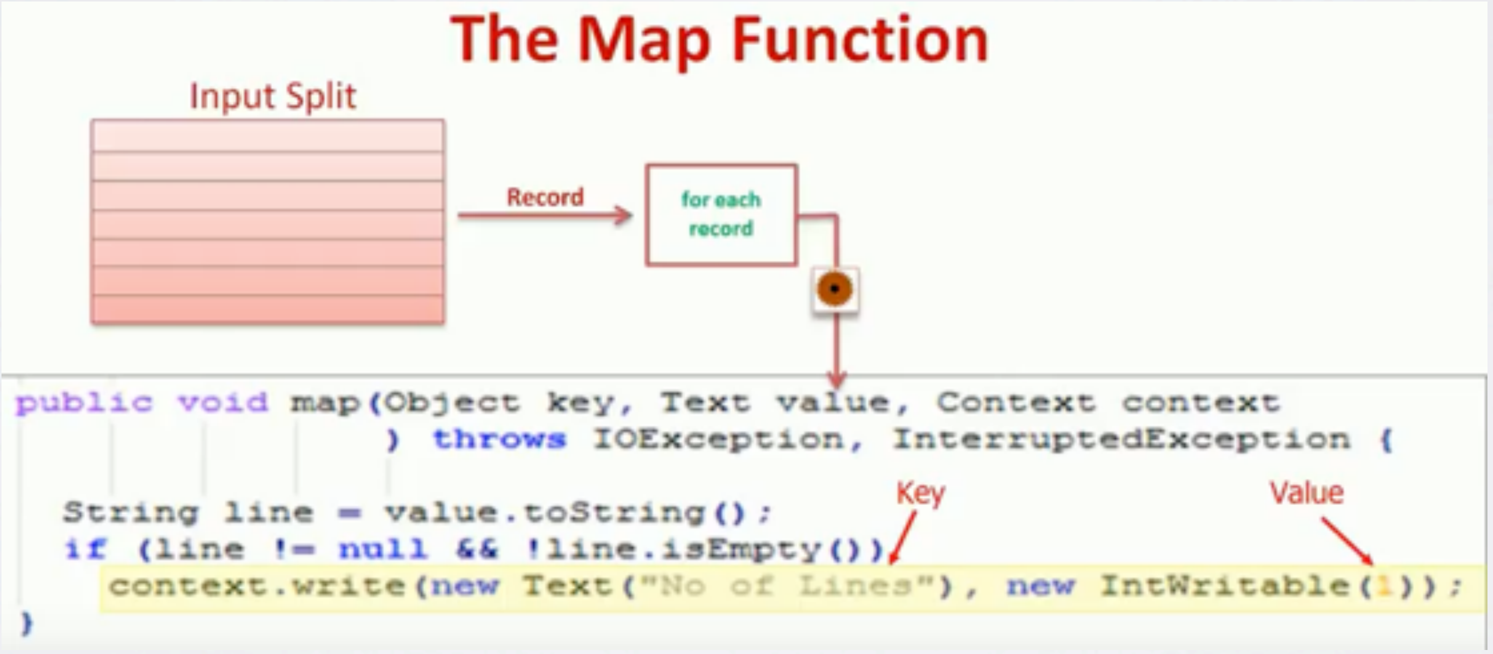
here is map function looks like.. which process one record from file at a time



The Map Function

```
public void map(Object key, Text value, Context context
                ) throws IOException, InterruptedException {

  String line = value.toString();
  if (line != null && !line.isEmpty())
    context.write(new Text("No of Lines"), new IntWritable(1));
}
```

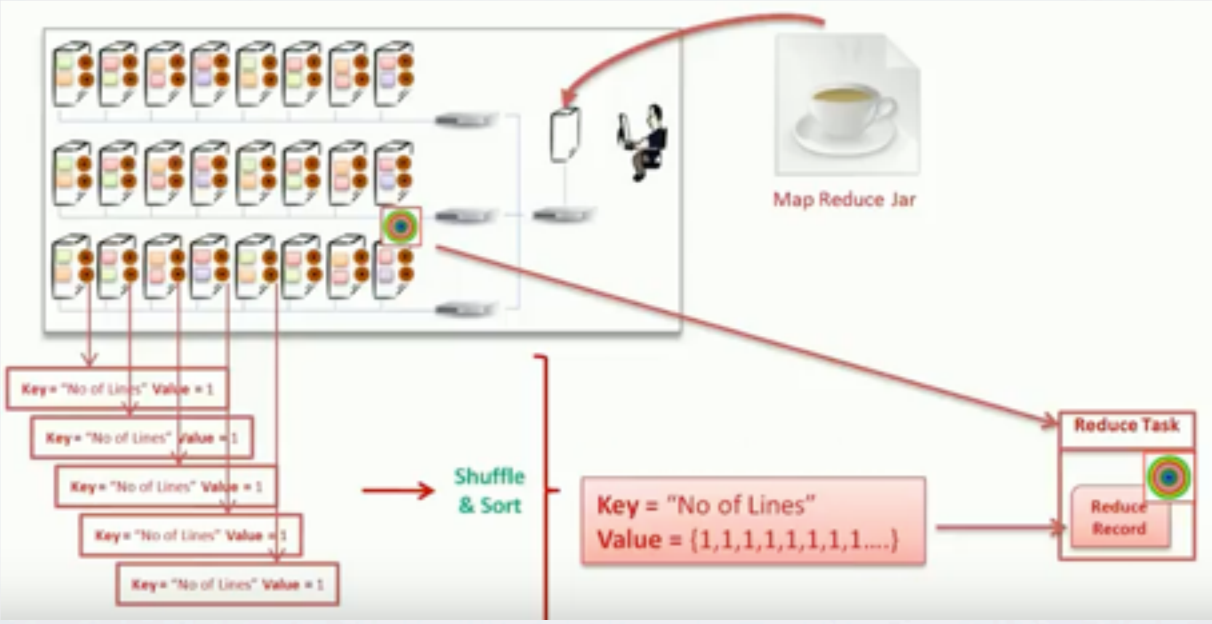


The Map Function

```
public void map(Object key, Text value, Context context
                ) throws IOException, InterruptedException {

  String line = value.toString();
  if (line != null && !line.isEmpty())
    context.write(new Text("No of Lines"), new IntWritable(1));
}
```

Now reducer will reduce it to count


Map Reduce Jar
Key = "No of Lines" Value = 1
Key = "No of Lines" Value = 1
Key = "No of Lines" Value = 1
Key = "No of Lines" Value = 1
Key = "No of Lines" Value = 1
Shuffle & Sort
Key = "No of Lines"
Value = {1,1,1,1,1,1,1,1....}
Reduce Task
Reduce Record