**Sanjana Asrani**

**D17B/01/Batch B**

**DC-09 Zookeeper**

**Aim:** Demonstrate Election algorithm in Zookeeper

**Theory:**

Hadoop is a free, Java-based programming framework that supports the processing of large data sets in a distributed computing environment. It is part of the Apache project sponsored by the Apache Software Foundation.

Hadoop makes it possible to run applications on systems with thousands of nodes involving thousands of terabytes. Its distributed file system facilitates rapid data transfer rates among nodes and allows the system to continue operating uninterrupted in case of a node failure. This approach lowers the risk of catastrophic system failure, even if a significant number of nodes become inoperative.

Hadoop was inspired by Google's MapReduce, a software framework in which an application is broken down into numerous small parts. Any of these parts (also called fragments or blocks) can be run on any node in the cluster. Doug Cutting, Hadoop's creator, named the framework after his child's stuffed toy elephant. The current Apache Hadoop ecosystem consists of the Hadoop kernel, MapReduce, the Hadoop distributed file system (HDFS) and a number of related projects such as Apache Hive, HBase and Zookeeper.

To understand election (Bully) algorithm for coordinator selection through Zookeeper

**Basic election algorithm**:

**Description :**  The coordinator election problem is to choose a process from among a group of processes on different processors in a distributed system to act as the central coordinator.

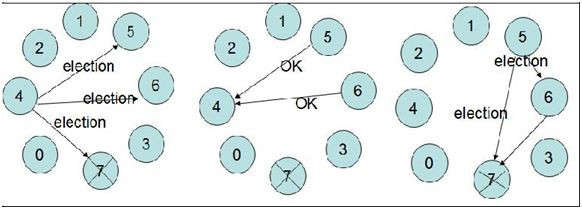
Distributed algorithm requires the need of the coordinator to be elected amongst the ones which are currently active.

**Bully Algorithm:** In this algorithm. It is assumed that every process knows the priority no. of every other process in the system.

**Working:**

● Process p calls an election when it notices that the coordinator is no longer responding. High numbered processes “bully” low numbered processes out of the election, until only one process remains.

● When a crashed process reboots, it holds an election. If it is now the highest numbered live process, it will win.



● Process p sends an election message to all higher numbered processes in the system.

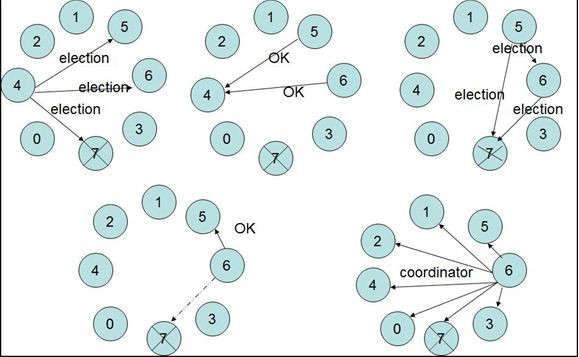
● If no process response, then p becomes the coordinator.

● If a higher level process (q) responds, it sends p a message that terminates p’s role in the algorithm.

● The process q now calls an election (if it has not already done so).

● Repeat until no higher level process responds. The last process to call an election “wins” the election.

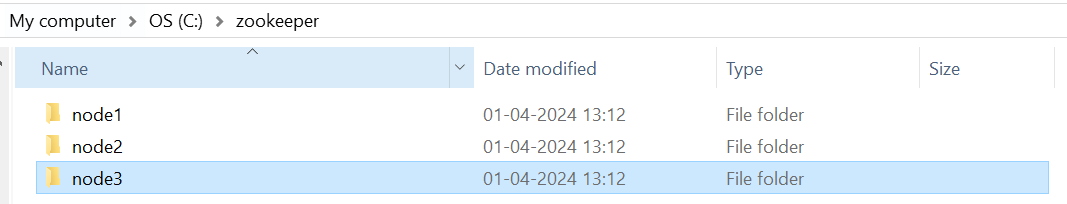
● The winner sends a message to other processes announcing itself as the new coordinator.



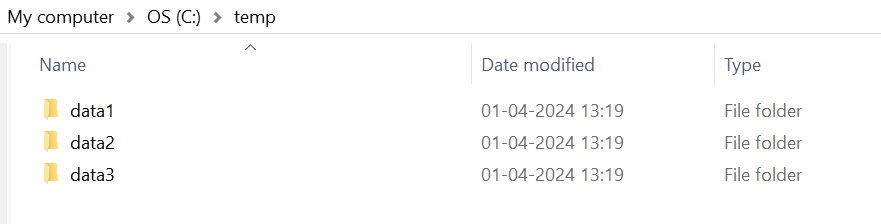
**Practical:**

**Create 2 folders:** zookeeper **&** temp **in C drive.**

**Create 3** node **Folders inside the** zookeeper **folder.**

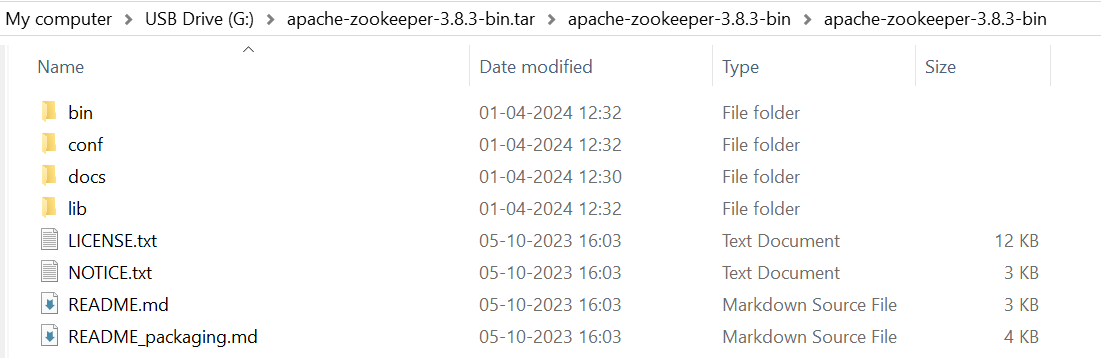


**Create 3** data **folders inside the** temp **folder.**

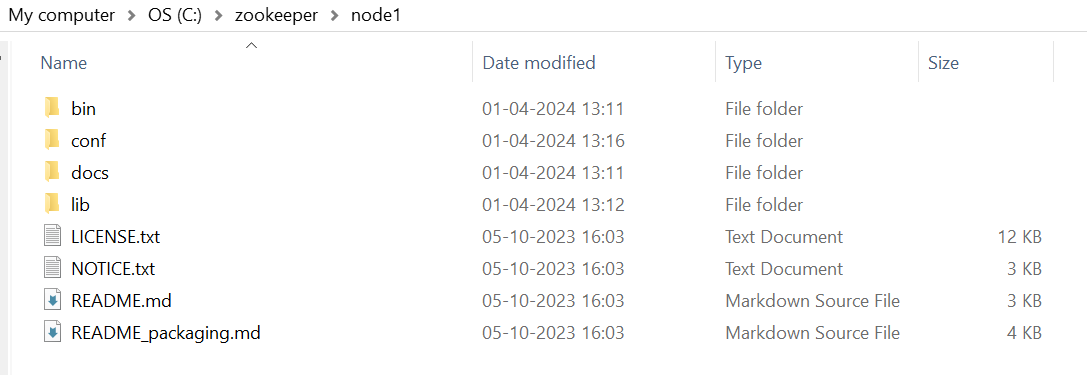


Download this **zookeeper** file from here: <https://archive.apache.org/dist/zookeeper/zookeeper-3.8.3/apache-zookeeper-3.8.3-bin.tar.gz>

Extract it somewhere:



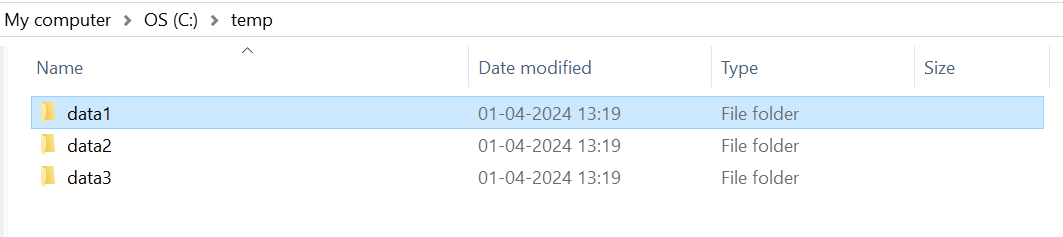
Copy all these extracted files in all 3 node folders, like this:

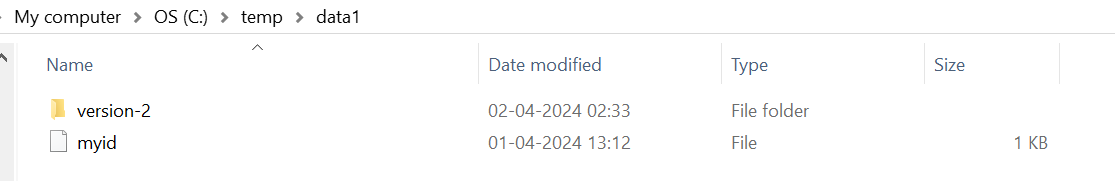


**Open zoo.cfg for node1, node2, node3:**

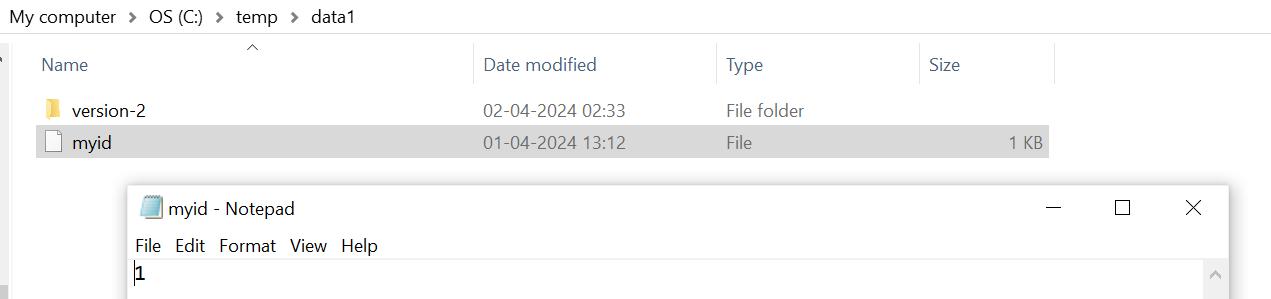
**(edit the dataDir as per the directory names and the client port respectively)**

|  |  |  |
| --- | --- | --- |
|  |  |  |





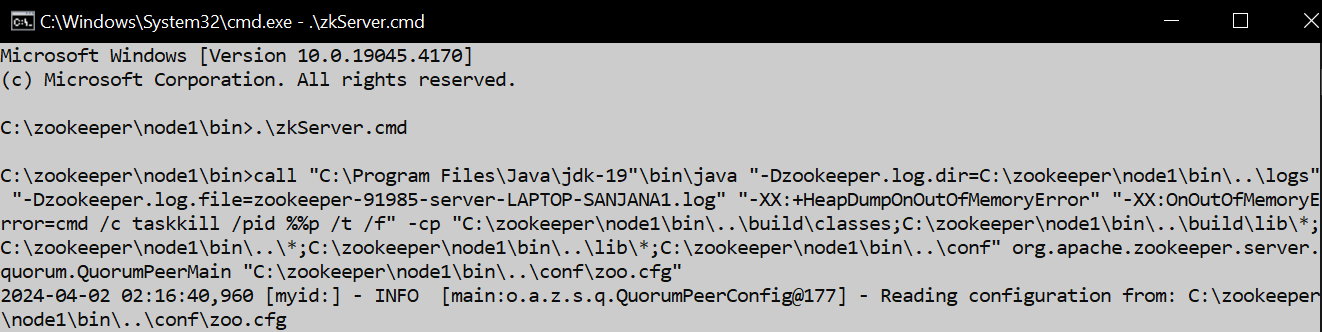
Edit the myid file for every data in this way:



Similarly do for data2 and data3.

**Open Cmd from inside bin folders of all nodes, run the cmd:** *.\zkServer.cmd*

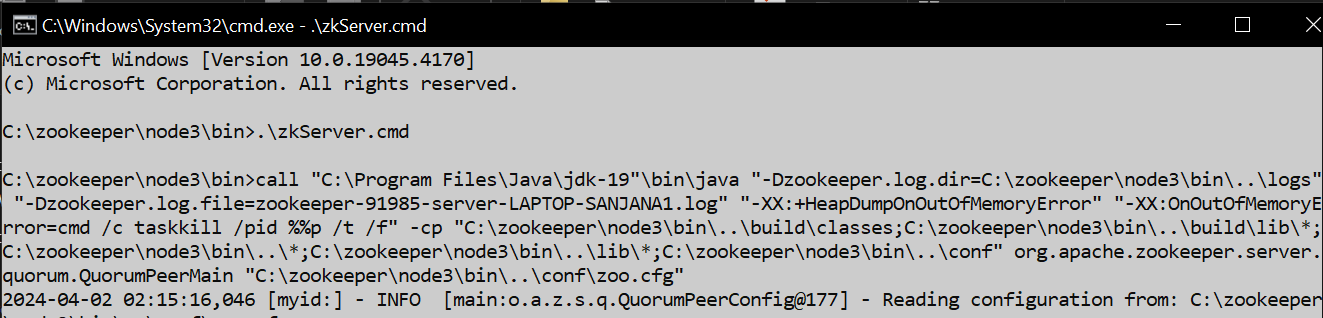
For node1:



For node2:



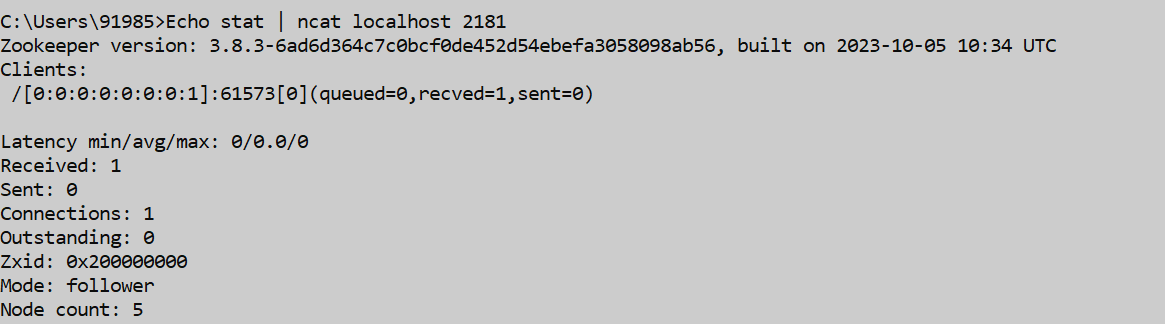
For node3:



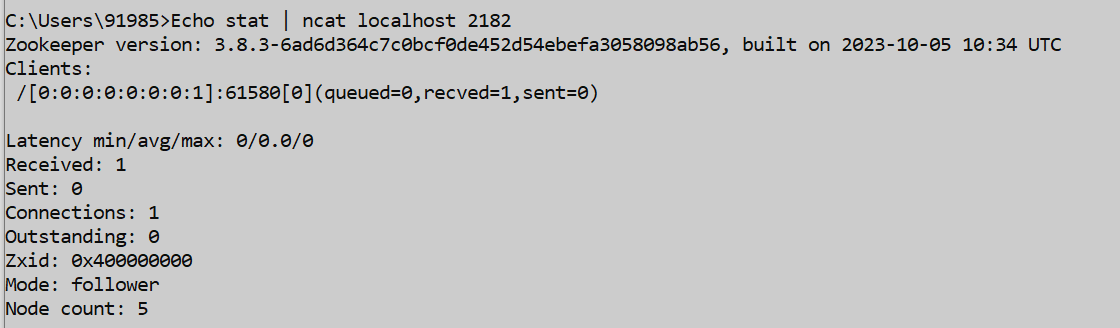
**This is similar to starting all the servers before running the client.**

**Open cmd prompt and run:**

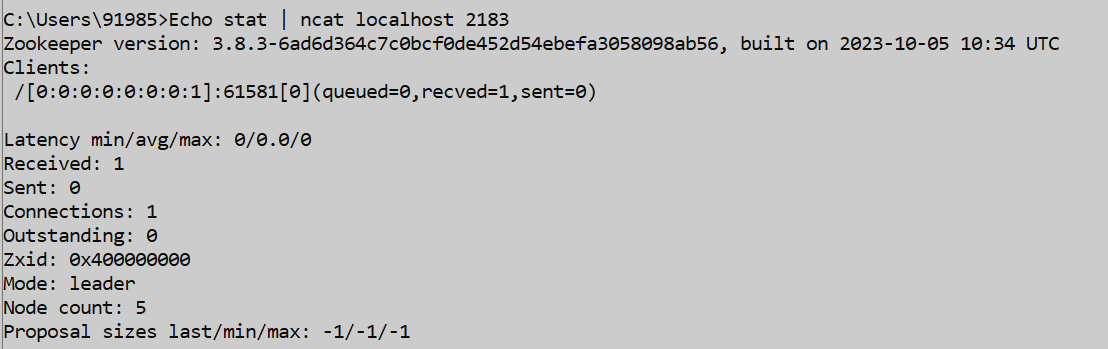
*Echo stat | ncat localhost 2181* **to start node1**



*Echo stat | ncat localhost 2182* **to start node2**



*Echo stat | ncat localhost 2183* **to start node3**



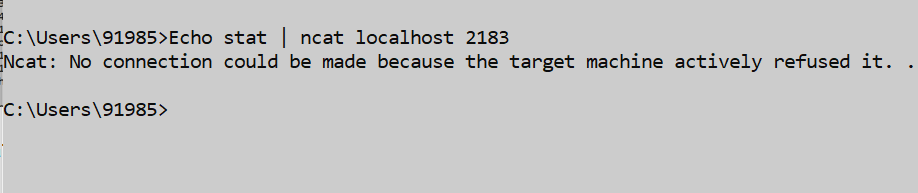
**One of the nodes is selected as a leader and rest are followers. Here, node3 is selected as leader.**

**If the leader is crashed, a new leader from the rest of followers is chosen:**

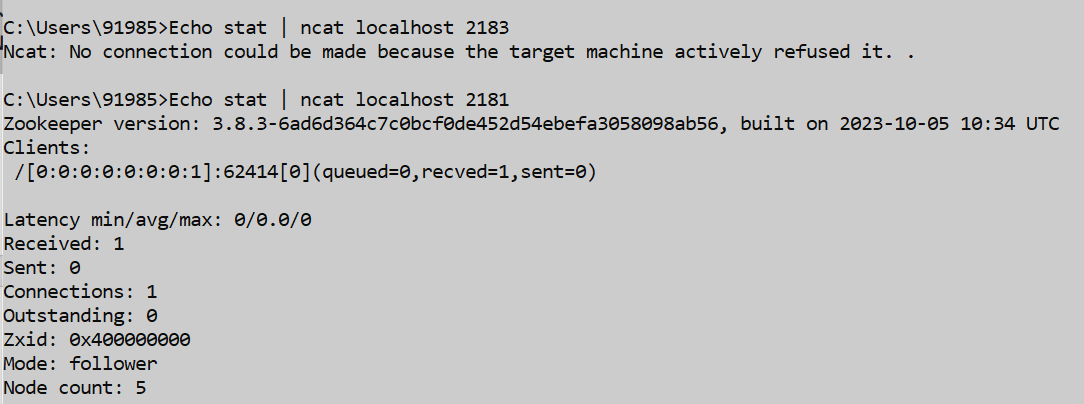


(To locate node3, find the myid=3 in the cmd output)

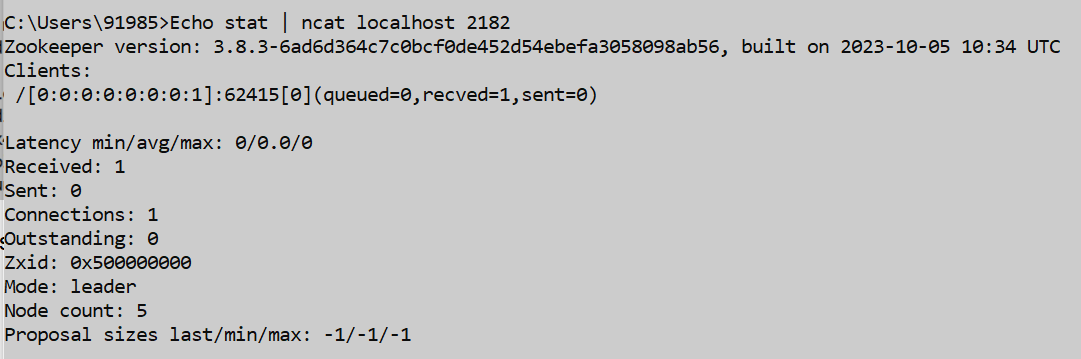
Since node3 isnt active anymore:



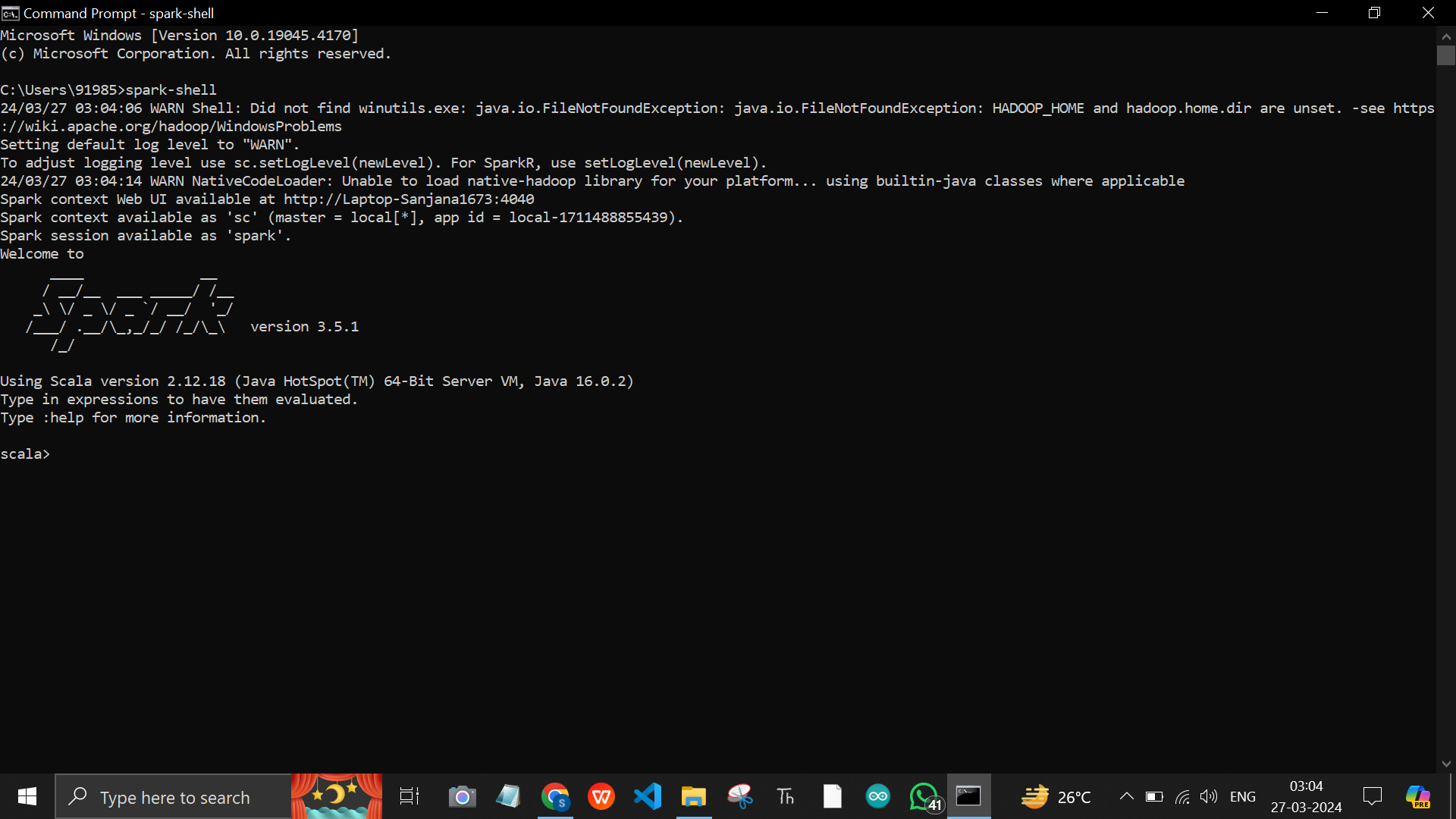
Now check the state of the other two nodes:



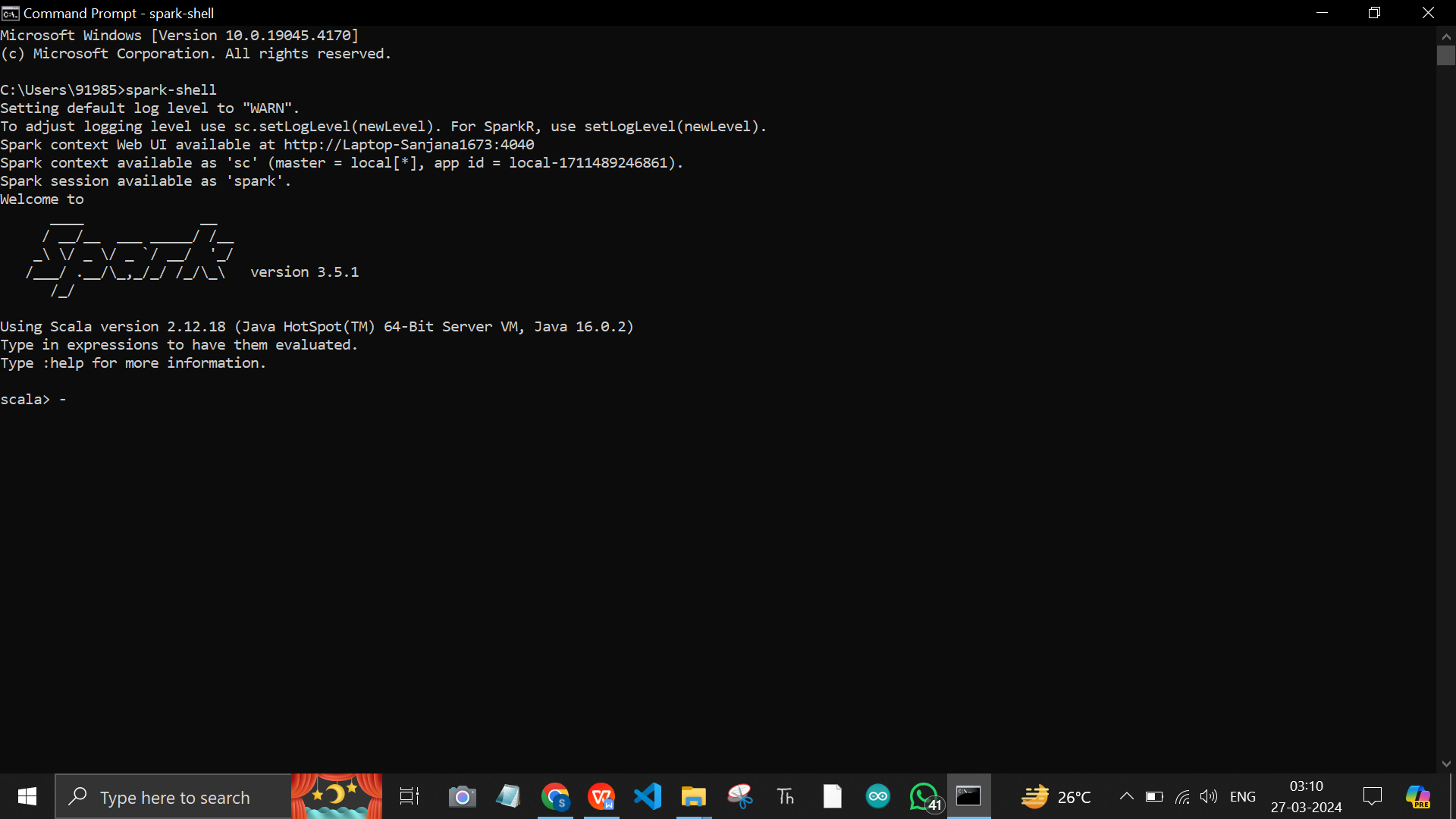
Node2 is chosen as the leader:



Setting up hadoop for demonstrating Hadoop Distributed File System (HDFS) using basic commands:



Now download winuitls for hadoop:



**Conclusion:**

ZooKeeper nodes for implementing an election algorithm yielded insightful results. This experiment demonstrated the effectiveness of ZooKeeper in managing distributed systems, ensuring consistent and reliable coordination even in the face of node failures or network partitions. Additionally, the election algorithm implemented on top of ZooKeeper showcased its suitability for critical tasks requiring consensus and leader selection in distributed environments. Overall, this experiment highlights the significance of ZooKeeper as a fundamental tool for building resilient distributed systems and underscores its importance in modern computing architectures.