**Cluster Setup Documentation**

--------------------------------------------------------------------------------------------

**1. Technologies Used**

* **Apache Spark**: Distributed processing engine for large-scale data analytics.
* **Hadoop HDFS**: Distributed storage system for handling tweet datasets.
* **Python/PySpark**: For implementing NLP, ML models, and Spark jobs.
* **Jupyter Notebook**: Interactive environment for prototyping and analysis.
* **OpenCage Geocoding API**: For converting text locations to coordinates.
* **Folium/Plotly**: Visualization libraries for spatial and sentiment analysis.

--------------------------------------------------------------------------------------------------------------------------------------

**2. Cluster Configuration**

**Hardware Setup**:

* **Master Node**: 1 PC (4 CPU cores, 4GB RAM, 25GB SSD).
* **Slave Nodes**: 2 PCs (each with 4 CPU cores, 4GB RAM, 25GB SSD).

**Software Stack**:

* **OS**: Ubuntu 20.04 LTS.
* **Java**: OpenJDK 8 (required for Spark/Hadoop).
* **Spark**: v3.2.1 (configured in standalone mode).
* **Hadoop**: v3.3.1 (for HDFS storage).

**Network**:

* Static IPs assigned to all nodes.
* Password-less SSH configured between master and slaves.

--------------------------------------------------------------------------------------------------------------------------------------

**3. Steps to Set Up the Cluster**

**A. Prerequisites**

1. **Install Java**:

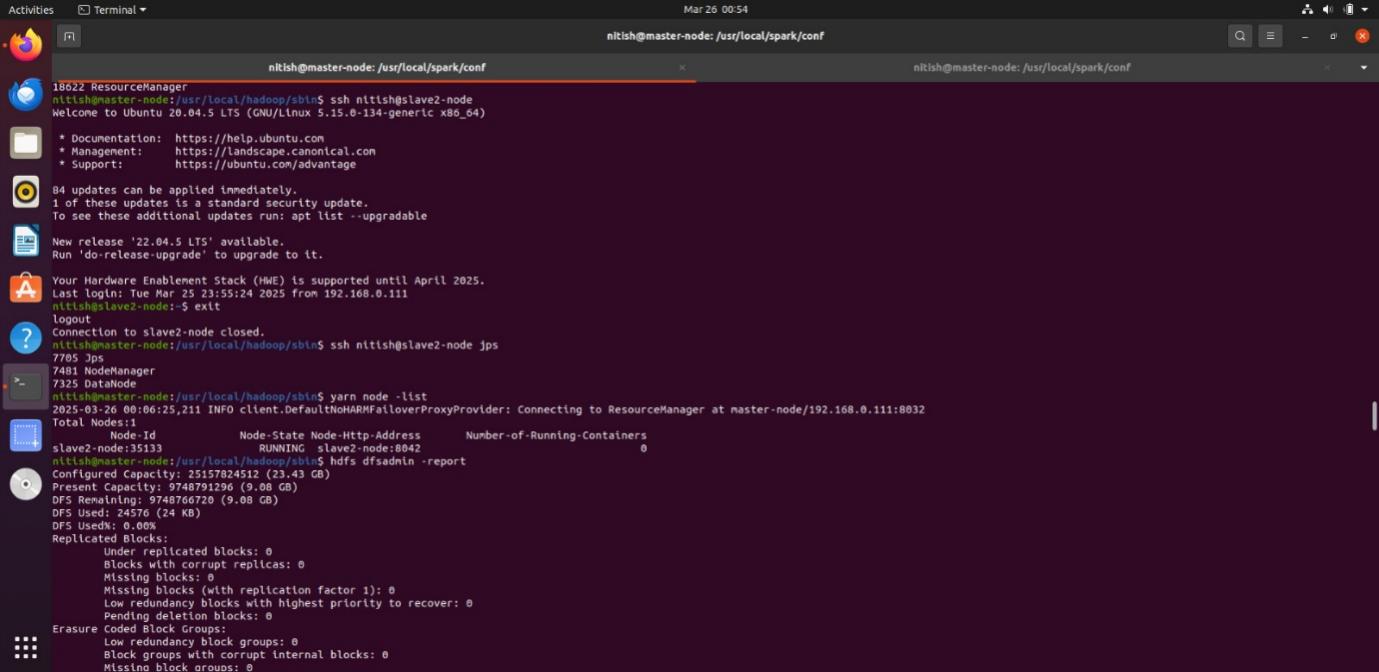
sudo apt update && sudo apt install openjdk-8-jdk

1. **Configure SSH**:

ssh-keygen -t rsa # Generate keys on master

ssh-copy-id slave1 # Copy keys to slaves

ssh-copy-id slave2



**B. Install Hadoop & Spark**

1. **Download and Extract**:

wget https://downloads.apache.org/hadoop/common/hadoop-3.3.1/hadoop-3.3.1.tar.gz

tar -xvzf hadoop-3.3.1.tar.gz

wget https://downloads.apache.org/spark/spark-3.2.1/spark-3.2.1-bin-hadoop3.2.tgz

tar -xvzf spark-3.2.1-bin-hadoop3.2.tgz

1. **Configure Environment Variables**:

Add to ~/.bashrc:

export JAVA\_HOME=/usr/lib/jvm/java-8-openjdk-amd64

export HADOOP\_HOME=/path/to/hadoop-3.3.1

export SPARK\_HOME=/path/to/spark-3.2.1-bin-hadoop3.2

export PATH=$PATH:$HADOOP\_HOME/bin:$SPARK\_HOME/bin

**C. Configure Spark & Hadoop**

1. **Spark Config**:
   * Edit $SPARK\_HOME/conf/spark-env.sh:

export SPARK\_MASTER\_HOST=<master-node-IP>

export SPARK\_WORKER\_CORES=4

export SPARK\_WORKER\_MEMORY=8g

* + Update $SPARK\_HOME/conf/slaves:

slave1

slave2

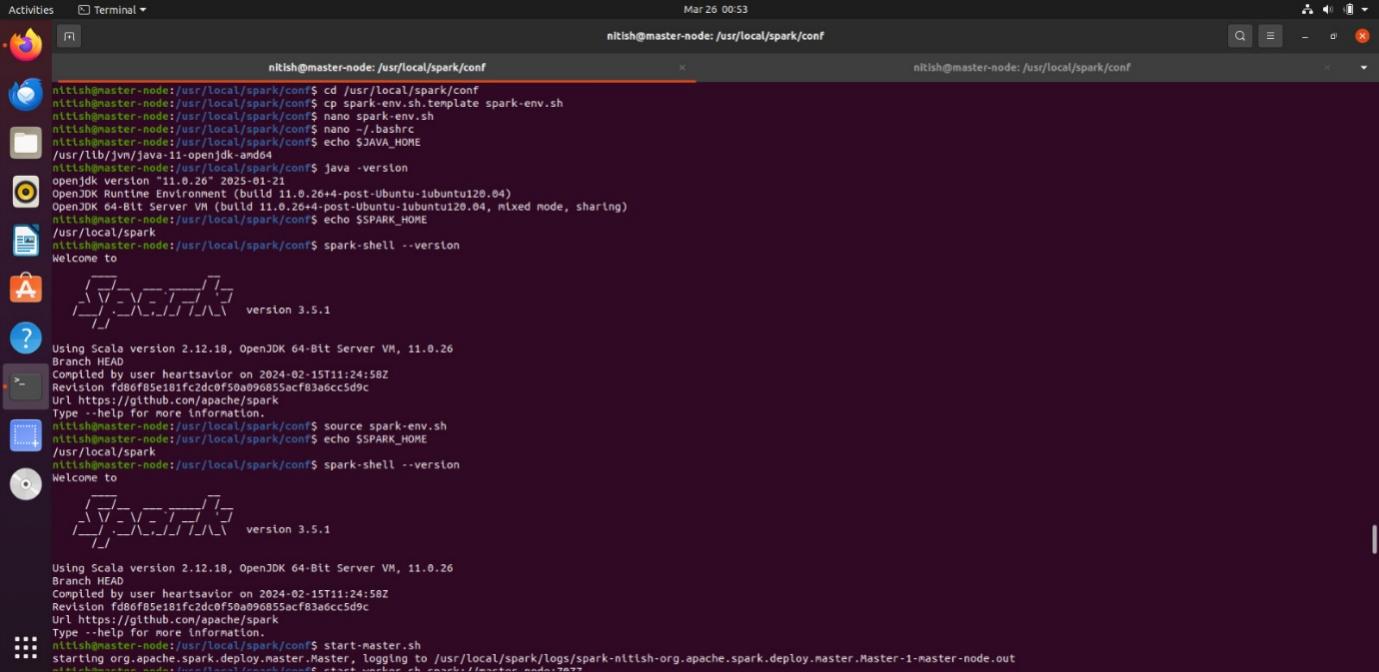
1. **Hadoop Config**:
   * Core-site.xml: Set HDFS URI.
   * Hdfs-site.xml: Configure replication factor (2)

**D. Start the Cluster**

1. **Launch Hadoop HDFS**:

hdfs namenode -format # First-time setup

start-dfs.sh



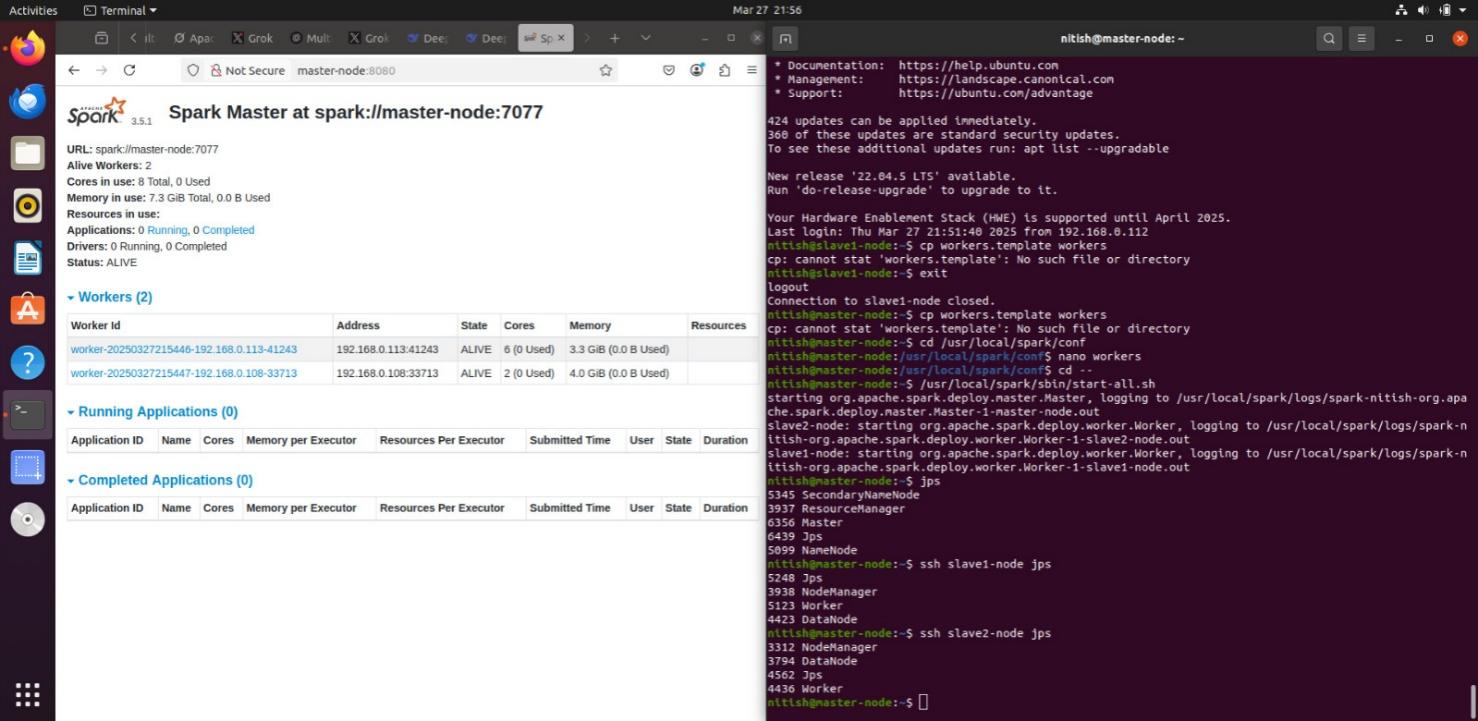
1. **Start Spark Cluster**:

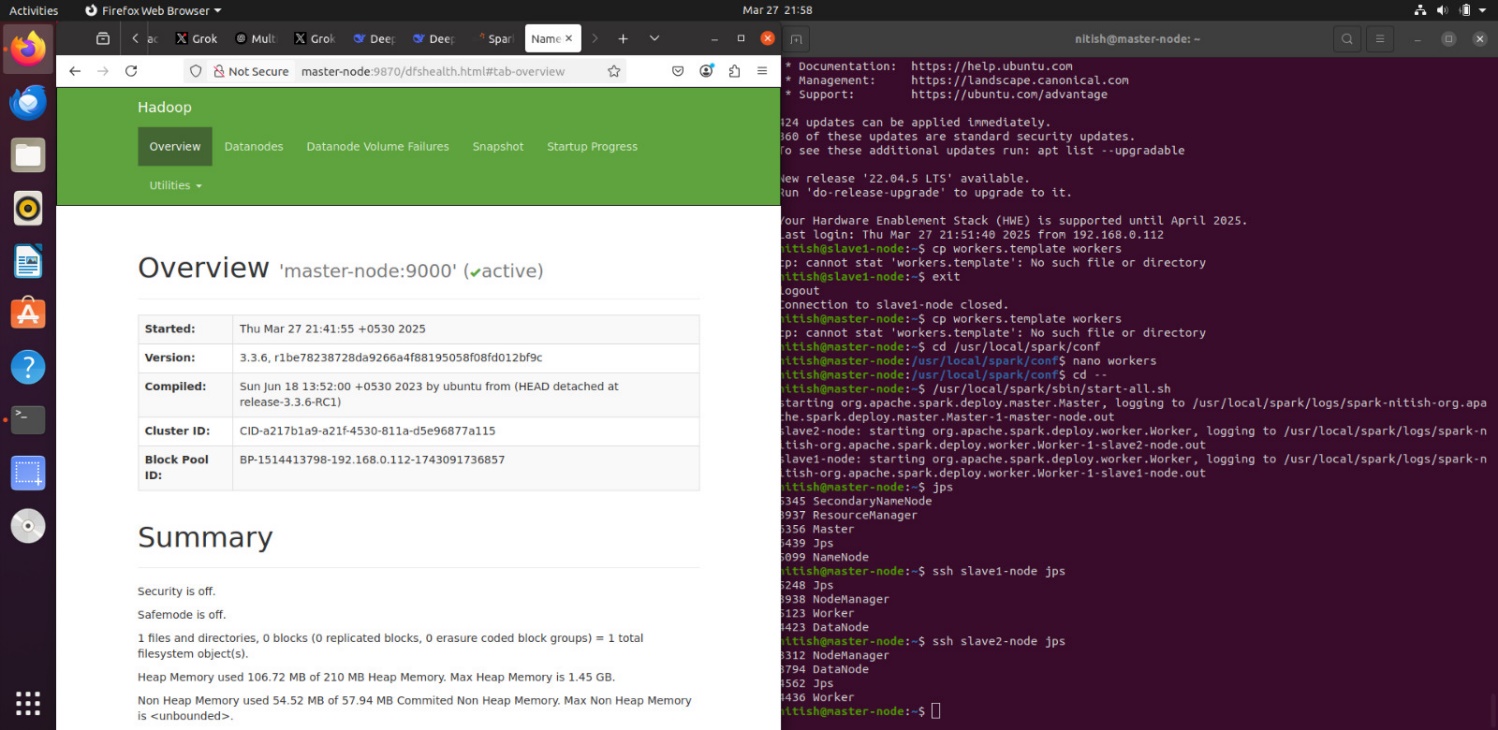
$SPARK\_HOME/sbin/start-master.sh

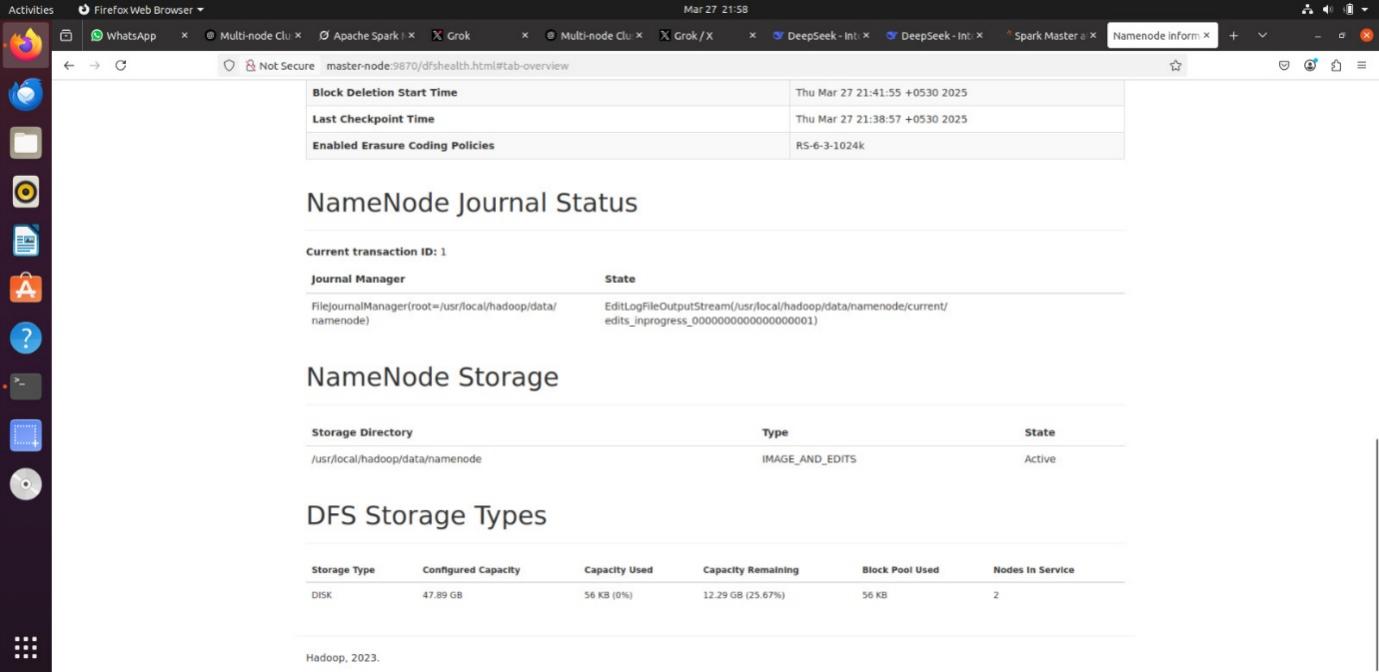
$SPARK\_HOME/sbin/start-workers.sh

**E. Verify Setup**

* **Spark UI**: Access http://<master-IP>:8080 to view active workers.
* **HDFS UI**: Check http://<master-IP>:9870.







-------------------------------------------------------------------------------------------------------------------------------------

**4. Issues Encountered & Resolutions**

| **Issue** | **Resolution** |
| --- | --- |
| **Slave nodes not connecting** | Ensured SSH keys were copied correctly and firewall rules allowed port 8080/7077. |
| **Out-of-memory errors** | Reduced SPARK\_WORKER\_MEMORY to 6g and optimized partition sizes. |
| **HDFS permission errors** | Ran hdfs dfs -chmod -R 777 / for development (not recommended for production). |
| **Geocoding API rate limits** | Implemented caching for repeated location queries. |
| **Jupyter-Spark integration fails** | Used findspark.init() and set PYSPARK\_PYTHON explicitly. |