Mini Project Report

Student Name: Karan **UID:24MCC20023**
Branch: MCA(CCD) **Section/Group:24MCD-1A**
Semester: 2nd **Date of Performance: 01-04-2025**
Subject: Big Data Lab **Subject Code: 24CAP-684**

Project Report: Automated File Backup System Using Hadoop HDFS

Aim of the project:

Develop a system to automatically back up essential Linux files.That stores backups in HDFS to enhance fault tolerance and scalability. By implementing an automated scheduling mechanism to perform backups periodically and maintain detailed logs for backup monitoring and auditing.

Introduction:

Data security is a crucial aspect of modern computing. Traditional backup methods can be inefficient and prone to errors. This project leverages Hadoop Distributed File System (HDFS) to automate and enhance backup efficiency, ensuring data safety and high availability.

Components required

- **Source Files** – Linux system files and user data requiring periodic backup.
- **Backup Script** – A Bash script to compress and upload files to HDFS.
- **HDFS Storage** – A distributed storage framework for secure data retention.
- **Logging System** – Tracks backup operations and stores execution details.

- **Scheduler** – Automates backup execution at predefined intervals.

System requirements

- **Hadoop Installation**: HDFS must be set up and configured.
- **User Permissions**: The system user must have access to read files and write to HDFS.

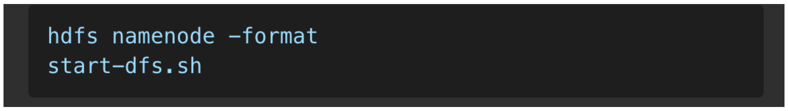
Backup Process

- The system archives designated directories into a compressed `.tar.gz` file.
- The backup file is transferred to HDFS.
- A log entry is generated to document the backup operation.
- The process repeats based on a schedule.

Implementation of the project:

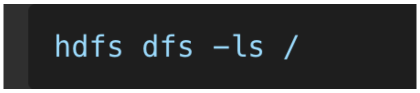
Configuration of Hadoop

Step 1: Initialize Hadoop services



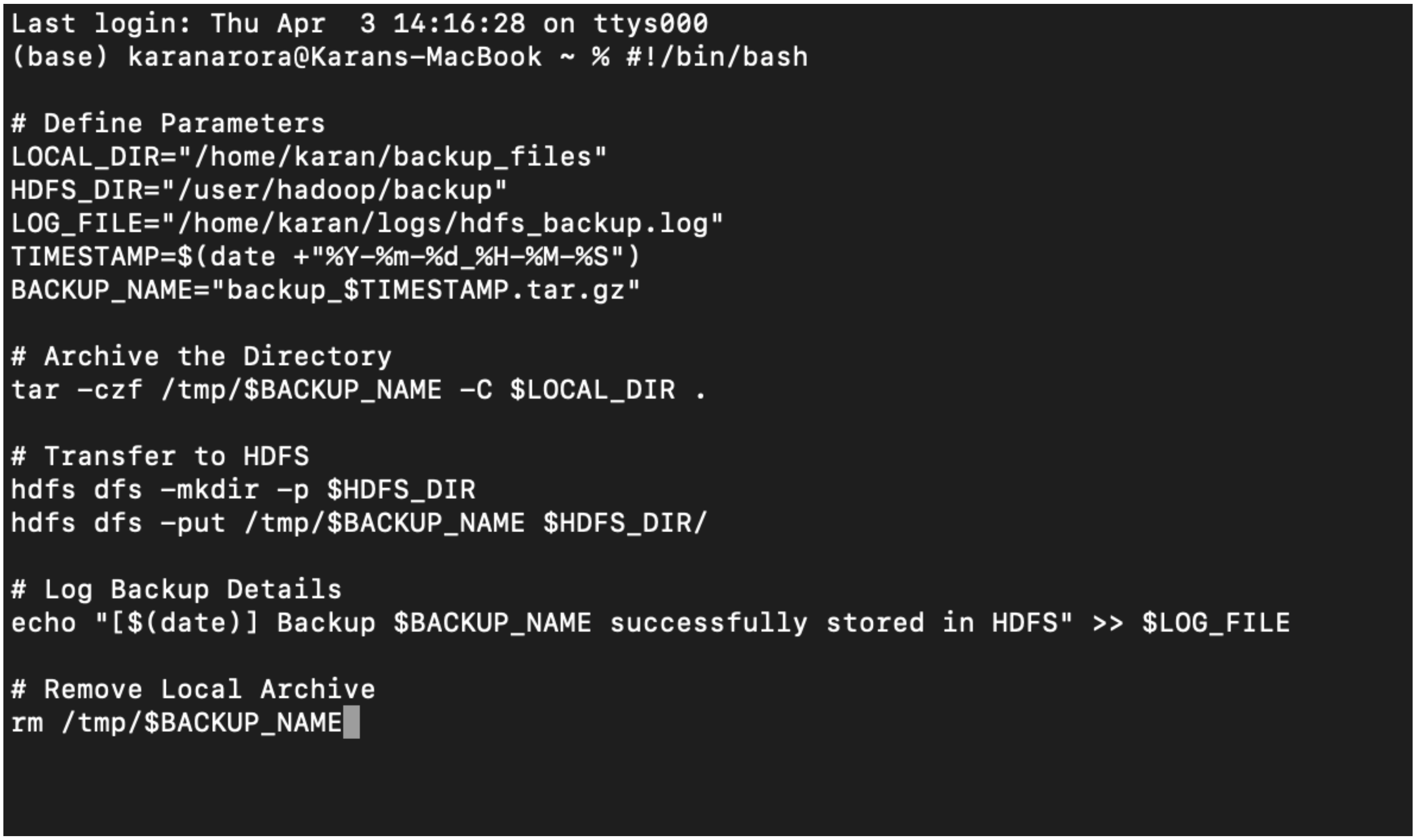
```
hdfs namenode -format
start-dfs.sh
```

Step 2: Verify the cluster status



```
hdfs dfs -ls /
```

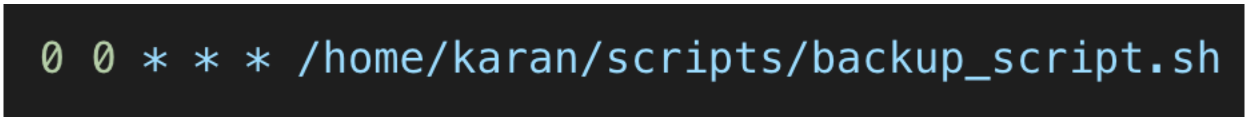
Step 3: Create a backup automation script



```
Last login: Thu Apr  3 14:16:28 on ttys000
(base) karanarora@Karans-MacBook ~ % #!/bin/bash

# Define Parameters
LOCAL_DIR="/home/karan/backup_files"
HDFS_DIR="/user/hadoop/backup"
LOG_FILE="/home/karan/logs/hdfs_backup.log"
TIMESTAMP=$(date +"%Y-%m-%d_%H-%M-%S")
BACKUP_NAME="backup_$TIMESTAMP.tar.gz"

# Archive the Directory
tar -czf /tmp/$BACKUP_NAME -C $LOCAL_DIR .

# Transfer to HDFS
hdfs dfs -mkdir -p $HDFS_DIR
hdfs dfs -put /tmp/$BACKUP_NAME $HDFS_DIR/

# Log Backup Details
echo "[$(date)] Backup $BACKUP_NAME successfully stored in HDFS" >> $LOG_FILE

# Remove Local Archive
rm /tmp/$BACKUP_NAME
```

Step 4: Automating Backups Using Cron Jobs

To execute backups automatically at midnight every day, add this line to the system's crontab:

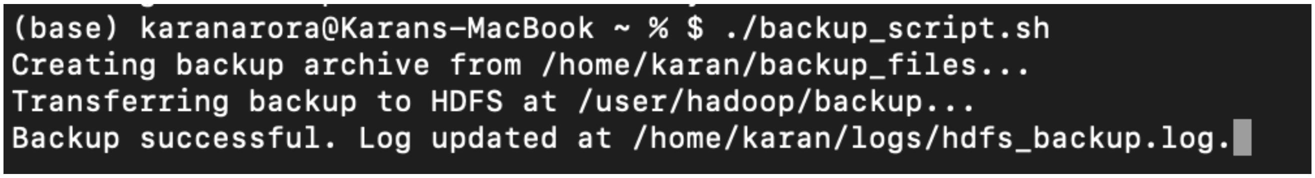


```
0 0 * * * /home/karan/scripts/backup_script.sh
```

Step 5:Testing and Validation

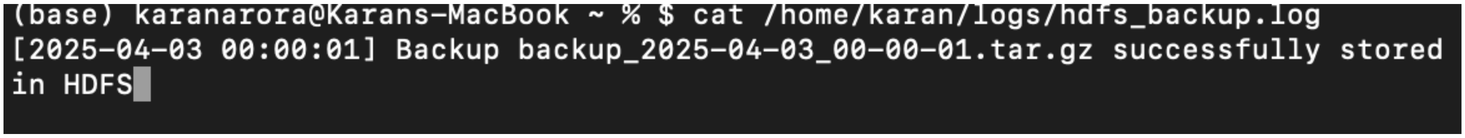
- **Initial Backup Execution**: The backup script should successfully archive and transfer files to HDFS.
- **Scheduled Backup Execution**: The cron job should execute at the predefined time and create a backup.
- **Log File Verification**: Log entries should be correctly stored and updated after each backup.
- **Error Handling (e.g., No HDFS Connection)**: The script should fail gracefully and log an appropriate error message.

Terminal Output



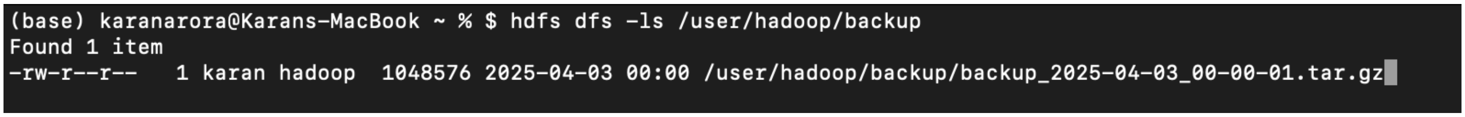
```
(base) karanarora@Karans-MacBook ~ % $ ./backup_script.sh
Creating backup archive from /home/karan/backup_files...
Transferring backup to HDFS at /user/hadoop/backup...
Backup successful. Log updated at /home/karan/logs/hdfs_backup.log.
```

Log File Entry Output



```
(base) karanarora@Karans-MacBook ~ % $ cat /home/karan/logs/hdfs_backup.log
[2025-04-03 00:00:01] Backup backup_2025-04-03_00-00-01.tar.gz successfully stored
in HDFS
```

Backup Confirmation output



```
(base) karanarora@Karans-MacBook ~ % $ hdfs dfs -ls /user/hadoop/backup
Found 1 item
-rw-r--r--   1 karan hadoop  1048576 2025-04-03 00:00 /user/hadoop/backup/backup_2025-04-03_00-00-01.tar.gz
```

Conclusion

The project successfully implemented an automated file backup system using HDFS. The solution provides a reliable and fault-tolerant method for storing Linux system files, ensuring data safety and availability.