The Godfather of Talent | Futurense

Hadoop Cheat Sheet

HDFS NameNode Commands

- Format the NameNode:
  `hdfs namenode -format`
- Start the NameNode:
  `hdfs --daemon start namenode`
- Stop the NameNode:
  `hdfs --daemon stop namenode`
- Check the status of the NameNode:
  `hdfs --daemon status namenode`
- Safemode enter:
  `hdfs dfsadmin -safemode enter`
- Safemode leave:
  `hdfs dfsadmin -safemode leave`
- Check filesystem consistency:
  `hdfs fsck <path>`
- Show the status of all DataNodes:
  `hdfs dfsadmin -report`
- Refresh the list of DataNodes:
  `hdfs dfsadmin -refreshNodes`
- Set the quota for a directory:
  `hdfs dfsadmin -setSpaceQuota <quota> <path>`

HDFS DataNode Commands

- Start the DataNode:
  `hdfs --daemon start datanode`
- Stop the DataNode:
  `hdfs --daemon stop datanode`
- Check the status of the DataNode:
  `hdfs --daemon status datanode`
- Refresh the DataNode's policy:
  `hdfs dfsadmin -refreshDataNodeAdminConfiguration`
- List all available DataNode storage policies:
  `hdfs storagepolicies -listPolicies`
- Check if DataNodes are alive and responding:
  `hdfs dfsadmin -ping`
- List all DataNode storage report:
  `hdfs dfsadmin -listStorageReports`
- Check the status of DataNode storage volumes:
  `hdfs dfsadmin -getVolumeFailureSummary`
- Display the current DataNode's maximum number of concurrent xcievers:
  `hdfs dfsadmin -getXceiverCount`

HDFS Commands

LIST FILES

- List the contents of a directory:
  `hdfs dfs -ls <path>`
- List the contents of a directory recursively:
  `hdfs dfs -ls -R <path>`
- List all files in a directory:
  `hdfs dfs -ls <path> | grep -E '^-'`
- List all directories in a directory:
  `hdfs dfs -ls <path> | grep -E '^d'`

- List files sorted by file size:
  `hdfs dfs -ls <path> | grep -E '^-' | sort -k5,5n`
- List files sorted by modification time (newest first):
  `hdfs dfs -ls -t <path>`
- List files sorted by modification time (oldest first):
  `hdfs dfs -ls -t -r <path>`
- List files matching a specific pattern (\.txt):
  `hdfs dfs -ls <path> | grep PATTERN`
- List files in a directory excluding a specific pattern:
  `hdfs dfs -ls <path> | grep -v PATTERN`
- List files with human-readable file sizes:
  `hdfs dfs -ls -h <path>`

READ / WRITE FILES

- Create an empty file:
  `hdfs dfs -touchz /user/hadoop/file.txt`
- Read the content of a file:
  `hdfs dfs -cat /user/hadoop/data.txt`
- Read the content of a file and display line numbers:
  `hdfs dfs -cat /user/hadoop/data.txt | nl`
- Read the first few lines of a file:
  `hdfs dfs -head /user/hadoop/data.txt`
- Read the last few lines of a file:
  `hdfs dfs -tail /user/hadoop/data.txt`
- Read a file in HDFS and save it to a local file:
  `hdfs dfs -cat <hdfs_path> > <local_path>`
- Append the content of a local file to an existing HDFS file:
  `hdfs dfs -appendToFile /home/user/local_data.txt /user/hadoop/hdfs_data.txt`

UPLOAD / DOWNLOAD FILES

- Upload a local file to HDFS:
  `hdfs dfs -put <local_path> <hdfs_path>`
- Upload a local file to HDFS (overwrite if exists):
  `hdfs dfs -put -f <local_path> <hdfs_path>`
- Upload multiple local files to HDFS:
  `hdfs dfs -put <local_path1> <local_path2> ... <hdfs_path>`
- Upload a local directory to HDFS:
  `hdfs dfs -put <local_directory> <hdfs_path>`
- Upload a local file to HDFS and create any missing directories in the destination path:
  `hdfs dfs -put -p <local_path> <hdfs_path>`
- Upload a local file to HDFS using the copyFromLocal command:
  `hdfs dfs -copyFromLocal <local_path> <hdfs_path>`
- Download an HDFS file to the local filesystem using the copyToLocal command:
  `hdfs dfs -copyToLocal <hdfs_path> <local_path>`
- Download an HDFS file to the local filesystem:
  `hdfs dfs -get <hdfs_path> <local_path>`
- Download an HDFS directory to the local filesystem:
  `hdfs dfs -get <hdfs_directory> <local_directory>`
- Download only specific files from an HDFS directory to the local filesystem:
  `hdfs dfs -get <hdfs_path>/*<file_pattern>* <local_path>`
- Download an HDFS file as a compressed file on the local filesystem:
  `hdfs dfs -get -text <hdfs_path> | gzip > <local_path>.gz`

FILE MANAGEMENT

- Create an HDFS directory:
  `hdfs dfs -mkdir <hdfs_path>`
- Create multiple nested HDFS directories in a single command:
  `hdfs dfs -mkdir -p <hdfs_path>`
- Delete a directory:
  `hdfs dfs -rmdir <hdfs_path>`
- Delete an HDFS file:
  `hdfs dfs -rm <hdfs_path>`
- Delete an HDFS directory and its contents:
  `hdfs dfs -rm -r <hdfs_path>`
- Move an HDFS file or directory:
  `hdfs dfs -mv <source_hdfs_path> <destination_hdfs_path>`
- Copy an HDFS file or directory:
  `hdfs dfs -cp <source_hdfs_path> <destination_hdfs_path>`
- Display the disk usage of an HDFS file or directory:
  `hdfs dfs -du -h <hdfs_path>`
- Display the summary of disk usage by each user:
  `hdfs dfs -count -q <hdfs_path>`
- Display the information about an HDFS file or directory, including block size, replication factor, etc.:
  `hdfs dfs -stat <format> <hdfs_path>`
- Change the replication factor of an HDFS file:
  `hdfs dfs -setrep <replication_factor> <hdfs_path>`
- Concatenate HDFS files and display the output to the console:
  `hdfs dfs -cat <hdfs_path1> <hdfs_path2>`
- Create an empty HDFS file:
  `hdfs dfs -touchz <hdfs_path>`
- Copy an HDFS file or directory with a different replication factor:
  `hdfs dfs -cp -R -w <new_replication_factor> <source_hdfs_path> <destination_hdfs_path>`
- Copy an HDFS file or directory while preserving the original timestamps and ownership:
  `hdfs dfs -cp -p <source_hdfs_path> <destination_hdfs_path>`
- Move an HDFS file or directory and overwrite the destination if it exists:
  `hdfs dfs -mv -f <source_hdfs_path> <destination_hdfs_path>`

- **Remove all files in an HDFS directory while preserving the directory itself:**
  `hdfs dfs -rm -r -skipTrash <hdfs_directory>/*`
- **Remove all files matching a specific pattern in an HDFS directory** (eg:- .txt)**:**
  `hdfs dfs -rm -r -skipTrash <hdfs_directory>/*<pattern>`
- **Move all files from one HDFS directory to another:**
  `hdfs dfs -mv <source_hdfs_directory>/* <destination_hdfs_directory>/`
- **Move all files matching a specific pattern from one HDFS directory to another:**
  `hdfs dfs -mv <source_hdfs_directory>/*<pattern> <destination_hdfs_directory>/`
- **Move an HDFS file or directory across file systems:**
  `hdfs dfs -cp -f <source_fs_path> <destination_fs_path>`

OWNERSHIP AND VALIDATION

- **Change the owner of an HDFS file or directory:**
  `hdfs dfs -chown <owner> <hdfs_path>` (eg:- hdfs dfs -chown user1 /user/hadoop/data.txt)
- **Change the owner of an HDFS file or directory recursively:**
  `hdfs dfs -chown -R <owner> <hdfs_path>`
- **Change the group of an HDFS file or directory:**
  `hdfs dfs -chgrp <group> <hdfs_path>`
- **Change the group of an HDFS file or directory recursively:**
  `hdfs dfs -chgrp -R <group> <hdfs_path>`
- **Change both the owner and the group of an HDFS file or directory:**
  `hdfs dfs -chown <owner>:<group> <hdfs_path>`
- **Change both the owner and the group of an HDFS file or directory recursively:**
  `hdfs dfs -chown -R <owner>:<group> <hdfs_path>`
- **Change the permissions of an HDFS file or directory:**
  `hdfs dfs -chmod <permissions> <hdfs_path>` (eg:- hdfs dfs -chmod 644 /user/hadoop/data.txt)
- **Change the permissions of an HDFS file or directory recursively:**
  `hdfs dfs -chmod -R <permissions> <hdfs_path>`
- **Check the HDFS file system for inconsistencies and report any issues:**
  `hdfs fsck <hdfs_path> [-files] [-blocks] [-locations] [-racks]`
- **Check the HDFS file system for missing blocks and files with a zero replication factor:**
  `hdfs fsck <hdfs_path> -list-corruptfileblocks`
- **Check the HDFS file system for under-replicated blocks:**
  `hdfs fsck <hdfs_path> -list-underreplicated`

- Test if an HDFS file or directory exists:
  `hdfs dfs -test -e <hdfs_path>`
- Test if an HDFS file or directory exists:
  `hdfs dfs -test -d <hdfs_path>`
- Test if an HDFS path is a directory:
  `hdfs dfs -test -d <hdfs_path>`
- Compute the checksum of an HDFS file:
  `hdfs dfs -checksum <hdfs_path>`

FILESYSTEM

- Shows the capacity, free and used space of the filesystem:
  `hdfs dfs -df /hadoop`
- Format the sizes of files in a human-readable fashion:
  `hdfs dfs -df -h /hadoop`
- Show the amount of space, in bytes, used by the files that match the specified file pattern:
  `hdfs dfs -du /hadoop/file`
- Show the total (summary) size:
  `hdfs dfs -du -s /hadoop/file`
- Formats the sizes of files in a human-readable fashion:
  `hdfs dfs -du -h /hadoop/file`

MapReduce

COMMANDS TO INTERACT WITH MAPREDUCE

- Command to submit the Jobs created:
  `hadoop job -submit <job-file>`
- Command to show the map and reduce completion status and all job counters:
  `hadoop job -status <job-id>`
- Print the counter value:
  `hadoop job -counter <job-id> <group-name> <countername>`
- Command to kill the job:
  `hadoop job -kill <job-id>`
- Details of event received by the job tracker for the given range:
  `hadoop job -events <job-id> <from_event-#> <#-of-events>`

- Print the job details, killed and failed tip details:
  `hadoop job -history [all] <jobOutputDir>`
- Command used to display all the jobs:
  `hadoop job -list[all]`
- Command used to kill the tasks:
  `hadoop job -kill-task <task-id>`
- Command used to fail the task:
  `hadoop job -fail-task <task-id>`
- Command to change and set the priority of the job:
  `hadoop job -set-priority <job-id> <priority>`
- Command to kill the job created:
  `HADOOP_HOME/bin/hadoop job -kill <JOB-ID>`
- Command to kill the job created:
  `HADOOP_HOME/bin/hadoop job -history <DIR-NAME>`
- Display counters for a specific MapReduce job:
  `mapred job -counter <job_id>`
- Set the priority of a running MapReduce job:
  `mapred job -set-priority <job_id> <priority_level>`
- Execute a MapReduce job using a JAR file:
  `hadoop jar <jar_file> <driver_class> <input_path> <output_path>`

YARN Commands

- Display YARN application types:
  `yarn application -appTypes`
- Display YARN application tags:
  `yarn application -appTags`
- Check the status of a specific YARN node:
  `yarn node -status <node_id>`
- Check the status of a specific application attempt:
  `yarn appattempt -status <app_attempt_id>`
- List all running applications:
  `yarn application -list`

- Check the status of a specific application:
  `yarn application -status <application_id>`
- Kill a running application:
  `yarn application -kill <application_id>`
- Check the status of a specific container:
  `yarn container -status <container_id>`
- List all containers for a specific application attempt:
  `yarn container -list <app_attempt_id>`
- List YARN node information:
  `yarn node -list [-all] [-states <states_list>]`
- View YARN logs for a specific application:
  `yarn logs -applicationId <application_id> [-appOwner <app_owner>]`
- List all application attempts for a specific application:
  `yarn appattempt -list <application_id>`
- Move an application to a different queue:
  `yarn application -movetoqueue <application_id> -queue <queue_name>`
- Update priority of an application:
  `yarn application -updatePriority <application_id> -priority <priority_level>`

Hadoop Administration

- Check the vesrion of Hadoop
  `hadoop version`
- Display the HDFS report:
  `hdfs dfsadmin -report`
- Refresh the DataNode list:
  `hdfs dfsadmin -refreshNodes`
- Set the balancer bandwidth:
  `hdfs dfsadmin -setBalancerBandwidth <bandwidth_in_bytes_per_second>`
- Execute the HDFS balancer:
  `hdfs balancer -threshold [<threshold_value>]`
- Check the HDFS file system:
  `hdfs fsck <path> [-delete] [-files] [-blocks] [-locations] [-racks]`

SECURITY COMMANDS

Kerberos:

- `kinit <principal_name>`:Obtain a Kerberos ticket-granting ticket (TGT) for the specified principal
- `klist`:List the current user's Kerberos tickets
- `kdestroy`:Destroy the current user's Kerberos tickets
- `kadmin`:Access the Kerberos administration tool
- `kadmin.local`:Access the Kerberos administration tool locally on the KDC (Key Distribution Center) server

Ranger:

- `ranger-admin start`:Start the Ranger Admin server
- `ranger-admin stop`:Stop the Ranger Admin server
- `ranger-usersync start`:Start the Ranger User Sync process
- `ranger-usersync stop`:Stop the Ranger User Sync process
- `ranger-tagsync start`:Start the Ranger Tag Sync process
- `ranger-tagsync stop`:Stop the Ranger Tag Sync process

Knox:

- `knoxcli.sh create-master`:Create a master secret for Knox
- `knoxcli.sh create-cert`:Create a public and private key pair for Knox
- `knoxcli.sh export-cert`:Export Knox's public key
- `knoxcli.sh list-topologies`:List the available Knox topologies
- `gateway.sh start`:Start the Knox Gateway server
- `gateway.sh stop`:Stop the Knox Gateway server

HDFS:

- `hdfs dfsadmin -safemode enter`:Enter HDFS safe mode
- `hdfs dfsadmin -safemode leave`:Leave HDFS safe mode
- `hdfs dfsadmin -refreshSuperUserGroupsConfiguration`:Refresh the superuser groups configuration

YARN:

- `yarn rmadmin -refreshQueues`:Refresh YARN queues
- `yarn rmadmin -refreshNodes`:Refresh YARN nodes
- `yarn rmadmin -refreshUserToGroupsMappings`:Refresh user-to-groups mappings

Hive:

- `beeline -u "jdbc:hive2://<host>:<port>/<db>;principal=<principal>"`: Connect to a secure HiveServer2 instance

The Godfather of Talent | Futurense

How to use ChatGPT Effectively for Bigdata Workflow

Speed-Up Bigdata Workflow

HIVE

- Ask ChatGPT for help in writing complex HiveQL queries, optimizing query performance, or partitioning strategies.
- Query for suggestions on UDFs (User-Defined Functions) or UDAFs (User-Defined Aggregate Functions) for custom data processing.

HADOOP

- Get guidance from ChatGPT on optimizing Hadoop job configurations or tuning parameters for better performance.
- Seek help on data partitioning and data locality strategies to improve job execution times.

HDFS

- Request help from ChatGPT for efficient HDFS file operations, such as copying, moving, or deleting data.
- Get suggestions for HDFS data storage and replication strategies, as well as best practices for working with HDFS.

YARN

- Consult ChatGPT for advice on configuring and managing YARN applications or optimizing resource allocation.
- Get help with YARN troubleshooting and monitoring, as well as understanding YARN logs

SQOOP

- Seek help from ChatGPT on writing Sqoop commands for importing and exporting data between HDFS and relational databases
- Ask for guidance on optimizing Sqoop job performance, using incremental imports, or managing database connections.

MAPREDUCE

- Request help from ChatGPT for writing efficient MapReduce jobs in Java or Python, including custom partitioners and combiners.
- Get advice on optimizing MapReduce performance, such as configuring memory settings or tuning I/O operations.

To use ChatGPT effectively across these technologies, follow these steps:

- Write down a clear and concise query describing your problem or the code snippet you need for a specific technology.
- Use ChatGPT to generate a response or code snippet based on your query.
- Review and analyze the response or code snippet to ensure it's suitable for your use case.
- If the response is not suitable or needs adjustments, refine your query and repeat steps 2-3
- Incorporate the code snippets or suggestions into your Big Data workflows and pipelines