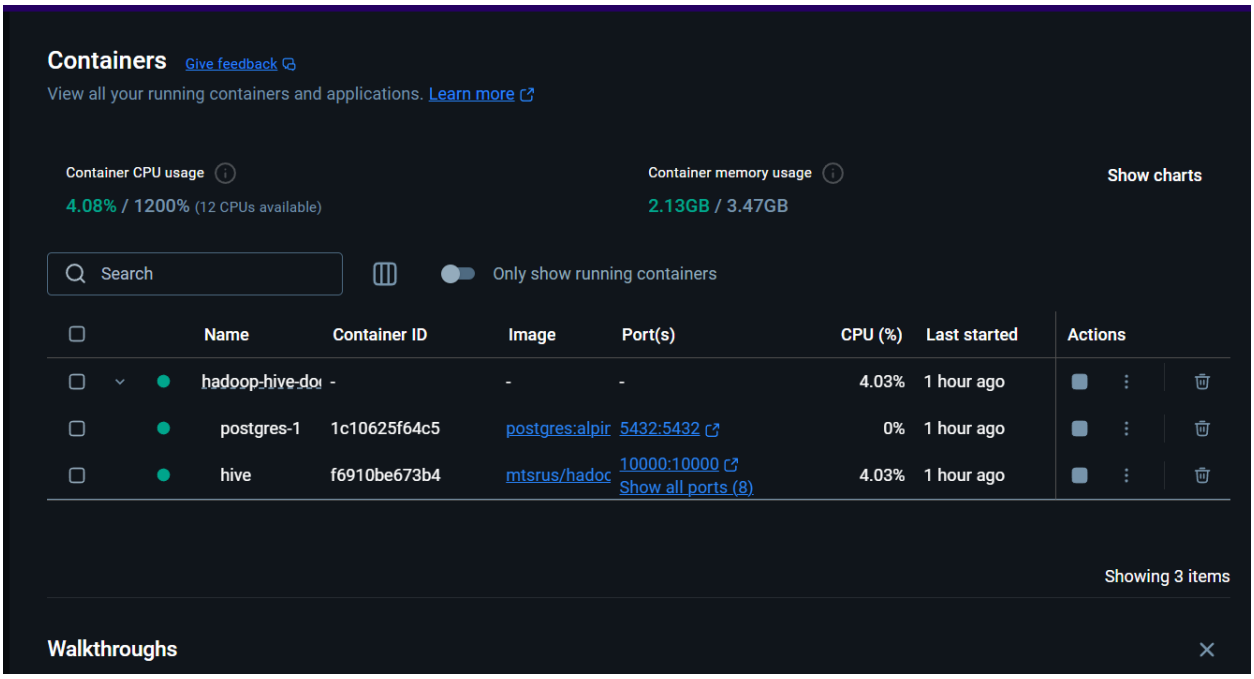
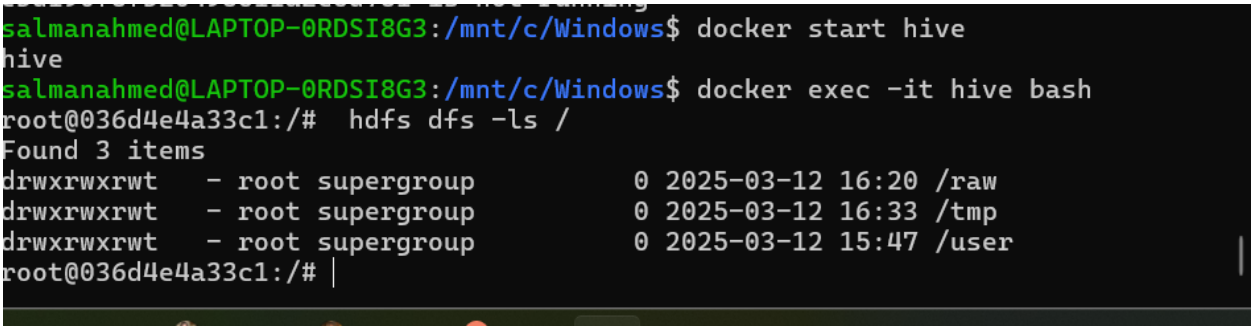
Building a Batch Analytics Pipeline on HDFS & Hive

Docker Container:



HDFS Working:



Hive Loading:

```
root@036d4e4a33c1:/# hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/hive/lib/log4j-slf4j-impl-2.17.2.jar!
/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/opt/hadoop/share/hadoop/common/lib/slf4j-
log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanat
ion.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactor
y]

Logging initialized using configuration in jar:file:/opt/hive/lib/hive-commo
n-2.3.10.jar!/hive-log4j2.properties Async: true
Hive-on-MR is deprecated in Hive 2 and may not be available in the future ve
rsions. Consider using a different execution engine (i.e. spark, tez) or usi
ng Hive 1.X releases.
hive>
```

Data mounting:

```
 hours   0.0.0.0:5432->5432/tcp   hadoop-hive-docker-postgres-1
salmanahmed@LAPTOP-0RDSI8G3:/mnt/c/Windows$ docker run -v "/mnt/c/Users/HP/D
esktop/DESKTOP/LUMS/1. SEMESTER 2/DATA ENGINEERING AI 601/ASSIGNMENTS/3/user
_data (1):/data" --name hive -d mtsrus/hadoop:hadoop2-hive
036d4e4a33c18fe33b975ad56d39e0a878d322ebd190f8fb20498511a2c5d781
salmanahmed@LAPTOP-0RDSI8G3:/mnt/c/Windows$ docker exec -it hive bash
root@036d4e4a33c1:/# ls/data
bash: ls/data: No such file or directory
root@036d4e4a33c1:/# ls /data
2023-09-01.csv  2023-09-06.csv            user_logs_2023-09-03.csv
2023-09-02.csv  2023-09-07.csv            user_logs_2023-09-04.csv
2023-09-03.csv  content_metadata.csv      user_logs_2023-09-05.csv
2023-09-04.csv  user_logs_2023-09-01.csv  user_logs_2023-09-06.csv
2023-09-05.csv  user_logs_2023-09-02.csv  user_logs_2023-09-07.csv
root@036d4e4a33c1:/# salmanahmed@LAPTOP-0RDSI8G3:/mnt/c/Windows$ docker star
```

Creating Raw Logs:

```
root@036d4e4a33c1: /

hive> CREATE EXTERNAL TABLE logs_table (user_id STRING, content_id STRING, a
ction STRING, timestamp STRING, device STRING, region STRING, session_id STR
ING) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE LOCATI
ON '/raw/logs';
NoViableAltException(287@[])
        at org.apache.hadoop.hive.ql.parse.HiveParser.columnNameTypeOrPKOrFK
(HiveParser.java:33341)
        at org.apache.hadoop.hive.ql.parse.HiveParser.columnNameTypeOrPKOrFK
List(HiveParser.java:29513)
        at org.apache.hadoop.hive.ql.parse.HiveParser.createTableStatement(H
iveParser.java:6175)
        at org.apache.hadoop.hive.ql.parse.HiveParser.ddlStatement(HiveParse
r.java:3808)
        at org.apache.hadoop.hive.ql.parse.HiveParser.execStatement(HivePars
er.java:2382)
        at org.apache.hadoop.hive.ql.parse.HiveParser.statement(HiveParser.j
ava:1333)
        at org.apache.hadoop.hive.ql.parse.ParseDriver.parse(ParseDriver.jav
a:208)
        at org.apache.hadoop.hive.ql.parse.ParseUtils.parse(ParseUtils.java:
77)
        at org.apache.hadoop.hive.ql.parse.ParseUtils.parse(ParseUtils.java:
70)
        at org.apache.hadoop.hive.ql.Driver.compile(Driver.java:468)
        at org.apache.hadoop.hive.ql.Driver.compileInternal(Driver.java:1317
)
        at org.apache.hadoop.hive.ql.Driver.runInternal(Driver.java:1457)
        at org.apache.hadoop.hive.ql.Driver.run(Driver.java:1237)
        at org.apache.hadoop.hive.ql.Driver.run(Driver.java:1227)
        at org.apache.hadoop.hive.cli.CliDriver.processLocalCmd(CliDriver.ja
va:233)
        at org.apache.hadoop.hive.cli.CliDriver.processCmd(CliDriver.java:18
4)
        at org.apache.hadoop.hive.cli.CliDriver.processLine(CliDriver.java:4
03)
        at org.apache.hadoop.hive.cli.CliDriver.executeDriver(CliDriver.java
:821)
        at org.apache.hadoop.hive.cli.CliDriver.run(CliDriver.java:759)
        at org.apache.hadoop.hive.cli.CliDriver.main(CliDriver.java:686)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorI
```

ENG US 21:37 12/03/2025

Creating of metadata:

```
root@036d4e4a33c1: /

        at org.apache.hadoop.hive.ql.parse.HiveParser.createTableStatement(H
iveParser.java:6175)
        at org.apache.hadoop.hive.ql.parse.HiveParser.ddlStatement(HiveParse
r.java:3808)
        at org.apache.hadoop.hive.ql.parse.HiveParser.execStatement(HivePars
er.java:2382)
        at org.apache.hadoop.hive.ql.parse.HiveParser.statement(HiveParser.j
ava:1333)
        at org.apache.hadoop.hive.ql.parse.ParseDriver.parse(ParseDriver.jav
a:208)
        at org.apache.hadoop.hive.ql.parse.ParseUtils.parse(ParseUtils.java:
77)
        at org.apache.hadoop.hive.ql.parse.ParseUtils.parse(ParseUtils.java:
70)
        at org.apache.hadoop.hive.ql.Driver.compile(Driver.java:468)
        at org.apache.hadoop.hive.ql.Driver.compileInternal(Driver.java:1317
)
        at org.apache.hadoop.hive.ql.Driver.runInternal(Driver.java:1457)
        at org.apache.hadoop.hive.ql.Driver.run(Driver.java:1237)
        at org.apache.hadoop.hive.ql.Driver.run(Driver.java:1227)
        at org.apache.hadoop.hive.cli.CliDriver.processLocalCmd(CliDriver.ja
va:233)
        at org.apache.hadoop.hive.cli.CliDriver.processCmd(CliDriver.java:18
4)
        at org.apache.hadoop.hive.cli.CliDriver.processLine(CliDriver.java:4
03)
        at org.apache.hadoop.hive.cli.CliDriver.executeDriver(CliDriver.java
:821)
        at org.apache.hadoop.hive.cli.CliDriver.run(CliDriver.java:759)
        at org.apache.hadoop.hive.cli.CliDriver.main(CliDriver.java:686)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorI
mpl.java:62)
        at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodA
ccessorImpl.java:43)
        at java.lang.reflect.Method.invoke(Method.java:498)
        at org.apache.hadoop.util.RunJar.run(RunJar.java:221)
        at org.apache.hadoop.util.RunJar.main(RunJar.java:136)
FAILED: ParseException line 1:84 cannot recognize input near 'timestamp' 'ST
RING' ',' in column name or primary key or foreign key
```

ENG US 21:37 12/03/2025

Ingestion of Scripts:

```
salmanahmed@LAPTOP-0RDSI8G3:/mnt/c/Windows$ docker exec -it hive bash
root@036d4e4a33c1:/# cd/scripts
bash: cd/scripts: No such file or directory
root@036d4e4a33c1:/# mkdir -p /scripts
root@036d4e4a33c1:/# cd/scripts
bash: cd/scripts: No such file or directory
root@036d4e4a33c1:/# cd /scripts
root@036d4e4a33c1:/scripts# nano --version
 GNU nano, version 7.2
 (C) 2023 the Free Software Foundation and various contributors
 Compiled options: --disable-libmagic --enable-utf8
root@036d4e4a33c1:/scripts# nano ingest_logs.sh
root@036d4e4a33c1:/scripts# chmod +x ingest_logs.sh
root@036d4e4a33c1:/scripts# ./ingest_logs.sh 2023-09-01
No log files found for 2023-09-01
No metadata files found for 2023-09-01
Ingestion completed for date: 2023-09-01
root@036d4e4a33c1:/scripts# salmanahmed@LAPTOP-0RDSI8G3:/mnt/c/Windows$
salmanahmed@LAPTOP-0RDSI8G3:/mnt/c/Windows$ docker exec -it hive bash
Error response from daemon: container 036d4e4a33c18fe33b975ad56d39e0a878d322
ebd190f8fb20498511a2c5d781 is not running
salmanahmed@LAPTOP-0RDSI8G3:/mnt/c/Windows$ docker start hive
hive
salmanahmed@LAPTOP-0RDSI8G3:/mnt/c/Windows$ docker exec -it hive bash
root@036d4e4a33c1:/# cd /scripts
root@036d4e4a33c1:/scripts# ./ingest_logs.sh 2023-09-01
```

Write up

Developed a batch analytics pipeline for the media industry, enabling the ingestion of user activity logs and content metadata into HDFS. The process involves transferring daily log files from a local directory to HDFS while maintaining a well-structured folder hierarchy. Additionally, a Docker container was set up to streamline the environment configuration and deployment.

Ingestion Script Section (ingest_logs.sh)

A Shell Script (`ingest_logs.sh`) is used for automation:

- Accepts a **date parameter** (e.g., `2023-09-01`).
- Parses **year, month, and day** from the date.
- Copies **user logs and metadata files** into HDFS using structured directories.
- Data is **generated using LLM**, saved locally, and **mounted to the container** for ingestion.
- The script runs **inside the container**, ensuring logs and metadata are structured correctly.

Data mounting code: “”“

docker run -v "/mnt/c/Users/HP/Desktop/DESKTOP/LUMS/1. SEMESTER 2/DATA ENGINEERING AI 601/ASSIGNMENTS/3/user_data (1):/data" --name hive -d mtsrus/hadoop:hadoop2-hive

036d4e4a33c18fe33b975ad56d39e0a878d322ebd190f8fb20498511a2c5d781

“”“

Raw Logs Table:

```
CREATE EXTERNAL TABLE raw_logs (
   user_id STRING,
   content_id STRING,
   action STRING,
   timestamp STRING,
   device STRING,
   region STRING,
   session_id STRING
)
PARTITIONED BY (year INT, month INT, day INT)
STORED AS ORC  -- More efficient than raw CSV
LOCATION '/raw/logs/'
TBLPROPERTIES (
   "orc.compress"="SNAPPY",  -- Fast compression for ORC
   "skip.header.line.count"="1"
);
```

Fact Table:

```
CREATE TABLE fact_user_actions (
   user_id STRING,
   content_id STRING,
   action STRING,
   timestamp TIMESTAMP,  -- Converted to TIMESTAMP for faster queries
   device STRING,
   region STRING,
   session_id STRING
)
PARTITIONED BY (year INT, month INT, day INT)
CLUSTERED BY (user_id) INTO 16 BUCKETS  -- Bucketing improves JOIN performance
STORED AS PARQUET;
```

Dimension Table (dim_content)

```
CREATE TABLE dim_content
STORED AS PARQUET
AS
SELECT
content_id,
title,
category,
artist,
my_length,
year,
```

```
month,
day
FROM raw_metadata;
```

Data Transformation Process

The transformation process involves **populating fact and dimension tables** using Hive **INSERT OVERWRITE** commands.

- **Timestamps are converted to proper formats** for efficient querying.
- **Dynamic partitioning is enabled** to automatically assign partitions based on `year, month, day`.
- **Column names are refined** for clarity and consistency.

Optimized Hive Transformation Queries

Enable dynamic partitioning for flexible inserts:

```
SET hive.exec.dynamic.partition = true;
SET hive.exec.dynamic.partition.mode = nonstrict;
```

Populating the Dimension Table (`dim_content`)

- Converts metadata into a structured **dimension table**.
- Removes redundant column prefixes (`my_`).
- Uses **INSERT OVERWRITE** to ensure fresh data updates.

```
INSERT OVERWRITE TABLE dim_content PARTITION (year, month, day)
SELECT
  content_id,
  title,  -- Fixed column name issue
  category,
  artist,
  length, -- Removed "my_" prefix
  year,
  month,
```

```
    day
FROM raw_metadata;
```

Populating the Fact Table:

- Ensures `timestamp` is converted to **TIMESTAMP format**.
- Uses **INSERT OVERWRITE** instead of `INSERT INTO` (avoids duplicate data).
- Aligns with **partitioning strategy** for optimized queries.

```
INSERT OVERWRITE TABLE fact_user_actions PARTITION (year, month, day)

SELECT

    user_id,

    content_id,

    action,

    CAST(timestamp AS TIMESTAMP),  -- Converts string to TIMESTAMP

    device,

    region,

    session_id,

    year,

    month,

    day

FROM raw_logs;
```