# Lecture 7 exercises

Daniel Skjold Toft

# Start HDFS, Hive and HBase

- Navigate to “BDDST21\Lecture7”
  - “docker-compose up –d”

- Starts HDFS (with some network changes)
- Starts an instance of Hive (4 containers)
- Starts an instance of HBase

# Download the book!

- “docker exec –ti namenode bash”
- “apt update”
- “apt install wget”
- “wget -O alice.txt https://www.gutenberg.org/files/11/11-0.txt”
- “hdfs dfs -mkdir /txt”
- “hdfs dfs -put alice.txt /txt/”

# Simple word count with Hive!

- “docker exec –ti hive-server beeline”
- “!connect jdbc:hive2://localhost:10000”
  - “hive” for both user and password (defined in .env file)
- “SHOW TABLES;” – test successful connection
- “CREATE TABLE lines (line STRING);”
- “LOAD DATA INPATH 'hdfs://namenode:9000/txt/alice.txt' OVERWRITE INTO TABLE lines;”

# Simple word count with Hive!

- “CREATE TABLE word_counts AS
  SELECT word, count(1) AS count FROM
  (SELECT explode(split(line, ' ')) AS word FROM lines) w
  GROUP BY word
  ORDER BY word;”
- “SELECT * FROM word_counts ORDER BY count DESC LIMIT 10;”

# Check out the HDFS!

- On the namenode container:
- “hdfs dfs –ls /txt”
- Is alice.txt there?
  - It is sent to the center of the Hive!
  - What if we don’t want to go to the center of the Hive?

# Hive external tables!

- Motivation?
  - Data maintained by something else than Hive
  - Dropping tables in Hive does not delete the data
- Clean up in Hive
  - “DROP TABLE word_counts”
  - “DROP TABLE lines”
  - “SHOW TABLES;” – verify they are gone
- Put the book back in the folder
  - “hdfs dfs –put alice.txt /txt”

# Hive external tables!

- “CREATE EXTERNAL TABLE lines (line string) LOCATION 'hdfs://namenode:9000/txt’;”
- Repeat the word count!
  - Create the word_counts table again
  - Verify by “SELECT * FROM word_counts ORDER BY count DESC LIMIT 10;”
- Could we put another book into the directory now?
  - Yes! Add the book by “hdfs dfs –put alice.txt /txt/alice2.txt” on namenode
  - “SELECT COUNT(*) FROM lines;”
  - “SELECT INPUT__FILE__NAME FROM lines GROUP BY INPUT__FILE__NAME;”

# Perspective to earlier lectures

- Personally, HiveQL is easier than written Spark jobs
- Hive great for historical analysis
  - Runs on MapReduce, Spark or Tez
  - Not real-time (processing can take time)
- Imagine using something to inject data into the “txt” folder live!
- But let’s store the results then!

# HBase

- “docker exec -ti hbase hbase shell”
- “create 'word_counts', 'cf’”
- “put 'word_counts', 'the', 'cf:count', 1683”
- “put 'word_counts', 'and', 'cf:count', 783”
- “scan 'word_counts’”
- “get ‘word_counts’,’the’”

# Your turn!

- Sentiment exercise with Hive!
- Find the positive and negative word count in Alice.txt using Hive
  - Find the positive and negative words first and put them into HDFS
- Bonus points for putting the result into HBase
- Optional: create a Python program that reads the Hive result and inserts it into Hbase
  - Different solutions: ODBC driver, Thrift or REST API

# Your turn! – Trace my steps

- Find the positive and negative word list from Lecture 4 sentiment exercise
- Get this file into a HDFS folder (just like alice.txt)
- Load this file into Hive in table “positive_words_load”
- Manipulate (replace, split and explode) into one column with a word in each row into table “positive_words”
- Join the “word_counts” and “positive_words” with a count!
- Rinse and repeat with negative words