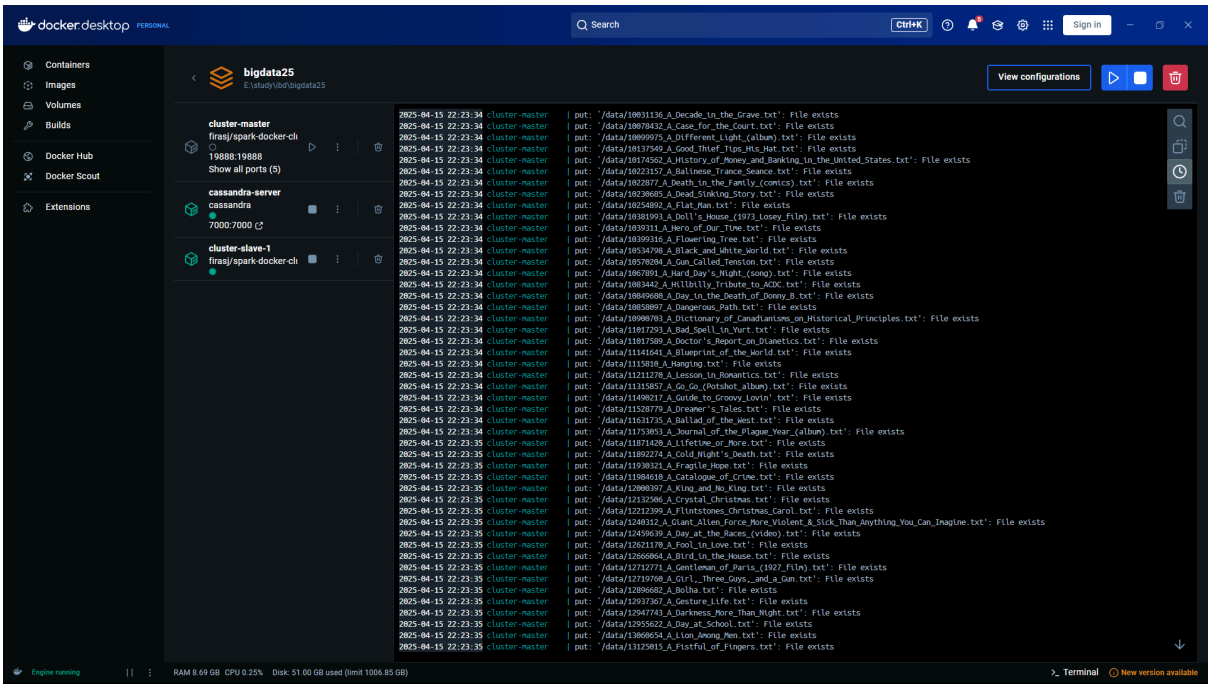
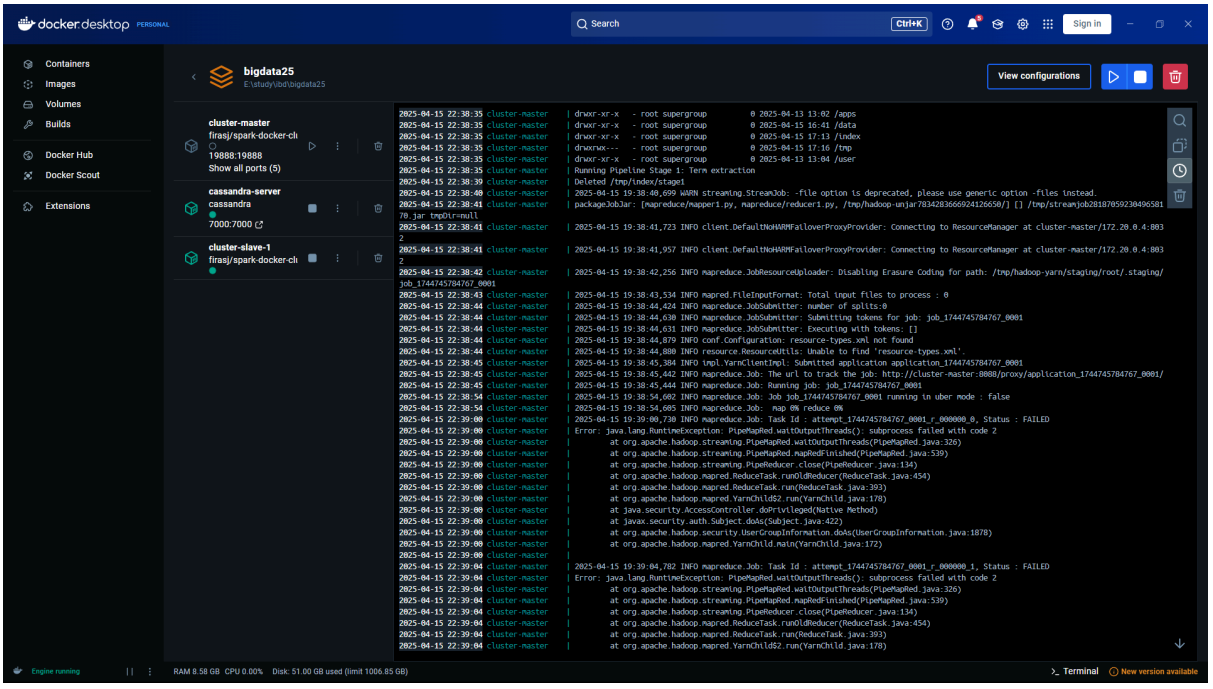
Methodology

Data preparation runs successfuly:



(It says file already exist because I did not delete the files in hdfs after the previous composition)

However, I get some errors on the map-reducing(indexing) stage. It would be great if I get the feedback about what I did wrong, because I’m trying to run the container for a few days and cannot understand what goes wrong. Thanks!



Files that I modified to run containers:

Folder mapreduce:
__init__.py

```
#!/usr/bin/env python3
from cassandra.cluster import Cluster
import logging

# Configure logging
logging.basicConfig(
    level=logging.INFO,
    format='%(asctime)s [%(levelname)s] %(message)s'
)
logger = logging.getLogger(__name__)

def initialize_cassandra():
    try:
        # Connect to Cassandra
        cluster = Cluster(
            ['cassandra-server'],
            port=9042
        )
        session = cluster.connect()

        logger.info("Successfully connected to Cassandra")

        # Create keyspace
        session.execute("""
            CREATE KEYSPACE IF NOT EXISTS search_index
            WITH replication = {
                'class': 'SimpleStrategy',
                'replication_factor': 1
            }
        """)
        logger.info("Created keyspace 'search_index'")

        # Switch to the keyspace
        session.set_keyspace('search_index')

        # Create tables
        tables = {
            'terms': """
                CREATE TABLE IF NOT EXISTS terms (
                    term text PRIMARY KEY,
                    document_frequency int
                )
            """,
```

```
            'document_index': """
                CREATE TABLE IF NOT EXISTS document_index (
                    term text,
                    doc_id text,
                    term_frequency int,
                    positions list<int>,
                    PRIMARY KEY (term, doc_id)
                )
            """,
            'document_stats': """
                CREATE TABLE IF NOT EXISTS document_stats (
                    doc_id text PRIMARY KEY,
                    doc_length int,
                    avg_term_length float
                )
            """
        }

        for table_name, query in tables.items():
            session.execute(query)
            logger.info(f"Created table '{table_name}'")

        logger.info("Cassandra schema initialization completed successfully")
        return True

    except Exception as e:
        logger.error(f"Error initializing Cassandra: {str(e)}")
        return False
    finally:
        if 'cluster' in locals():
            cluster.shutdown()

if __name__ == "__main__":
    logger.info("Starting Cassandra initialization...")
    if initialize_cassandra():
        logger.info("Initialization completed successfully")
    else:
        logger.error("Initialization failed")
        sys.exit(1)
```

Initialization is supposed to connect to Cassandra and create a few keyspaces and tables.

mapper1.py

```
#!/usr/bin/env python3
import sys
import re
from collections import defaultdict
```

```
print("this is mapper 1")

def tokenize(text):
    # Tokenizing text to find words
    words = re.findall(r'\b\w[\w-]*\b', text.lower())
    return words

for line in sys.stdin:
    try:
        # Parsing tab-separated input
        doc_id, doc_title, text = line.strip().split('\t', 2)
        words = tokenize(text)
        term_positions = defaultdict(list)

        # Calculating word positions
        for position, word in enumerate(words):
            term_positions[word].append(position)

        for term, positions in term_positions.items():
            print(f"{term}\t{doc_id}\t{len(positions)}\t{','.join(map(str,
positions))}")
    except Exception as e:
        # Skip malformed lines but log the error
        sys.stderr.write(f"ERROR processing line: {line}\n")
        continue
```

What it supposed to do:

1. Read input line by line from stdin (Hadoop streams data)
2. Split each line into doc_id, title, and text
3. Tokenize the text into lowercase terms
4. Record each term's positions in the document
5. Output: term, doc_id, term_frequency, and comma-separated positions

reducer1.py

```
#!/usr/bin/env python3
import sys

print("this is reducer 1")

current_term = None
term_data = []

def output():
    for doc_id, tf, positions in term_data:
        print(f"{current_term}\t{doc_id}\t{tf}\t{positions}")
```

```
for line in sys.stdin:
    term, doc_id, tf, positions = line.strip().split('\t')

    if term != current_term:
        if current_term is not None:
            output()
        current_term = term
        term_data = []

    term_data.append((doc_id, int(tf), positions))

if current_term is not None:
    output()
```

What it supposed to do:

1. Group mapper output by term
2. For each term, collect all document occurrences
3. Output the same structure but with grouped data

mapper2.py

```
#!/usr/bin/env python3
import sys

print("this is mapper 2")

# Identity mapper - just pass through the data
for line in sys.stdin:
    print(line.strip())
```

It does nothing - just copies the input - just needed because Hadoop need both mapper and reducer

reducer2.py

```
#!/usr/bin/env python3
import sys
from cassandra.cluster import Cluster

print("this is reducer 2")

# Connect to Cassandra
cluster = Cluster(['cassandra-server'])
session = cluster.connect('search_index')

current_term = None
documents = []
df = 0
```

```
def output():
    # Inserting into terms table
    session.execute(
        "INSERT INTO terms (term, document_frequency) VALUES (%s, %s)",
        (current_term, df)
    )

    # Inserting into document_index for each document
    for doc_id, tf, positions in documents:
        positions_list = list(map(int, positions.split(',')))
        session.execute(
            """INSERT INTO document_index
               (term, doc_id, term_frequency, positions)
               VALUES (%s, %s, %s, %s)""",
            (current_term, doc_id, tf, positions_list)
        )

        # Updating document stats (total terms in document)
        # We'll use the sum of all term frequencies as doc length
        session.execute(
            """UPDATE document_stats
               SET doc_length = doc_length + %s
               WHERE doc_id = %s""",
            (tf, doc_id)
        )

for line in sys.stdin:
    term, doc_id, tf, positions = line.strip().split('\t')

    if term != current_term:
        if current_term is not None:
            output()
        current_term = term
        documents = []
        df = 0

    documents.append((doc_id, tf, positions))
    df += 1

if current_term is not None:
    output()

cluster.shutdown()
```

What it supposed to do:

1. Group data by term (like Reducer1)

2. For each term:
    - Calculate document frequency - DF
    - Store in Cassandra terms table
    - Store each term-document relationship in document_index
    - Update document length statistics in document_stats

I also updated .sh files:

app.sh

```
#!/bin/bash
# Start ssh server
service ssh restart

# Starting the services
bash start-services.sh

# Creating a virtual environment
python3 -m venv .venv
source .venv/bin/activate

# I had problems with Spark, so I need wheel package
pip install wheel

# Install any packages
pip install -r requirements.txt

# Package the virtual env.
venv-pack -o .venv.tar.gz

# Collect data
bash prepare_data.sh

# Initialize Cassandra
python3 mapreduce/__init__.py

# Run the indexer
bash index.sh data/sample.txt

# Run the ranker
bash search.sh "this is a query!"
```

index.sh

```
#!/bin/bash
echo "This script include commands to run mapreduce jobs using hadoop
streaming to index documents"
```

```
echo "Input file is :"
echo $1


hdfs dfs -ls /

# Pipeline Stage 1: Term extraction
echo "Running Pipeline Stage 1: Term extraction"
hdfs dfs -test -d /tmp/index/stage1 && hdfs dfs -rm -r -f /tmp/index/stage1
hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-*.jar \
    -input /index/data \
    -output /tmp/index/stage1 \
    -mapper "/usr/bin/python3 mapreduce/mapper1.py" \
    -reducer "/usr/bin/python3 mapreduce/reducer1.py" \
    -file mapreduce/mapper1.py \
    -file mapreduce/reducer1.py

if [ $? -ne 0 ]; then
    echo "Pipeline Stage 1 failed"
    exit 1
fi

# Pipeline Stage 2: Cassandra storage
echo "Running Pipeline Stage 2: Cassandra storage"
hdfs dfs -test -d /tmp/index/stage2 && hdfs dfs -rm -r -f /tmp/index/stage2
hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-*.jar \
    -input /tmp/index/stage1 \
    -output /tmp/index/stage2 \
    -mapper "/usr/bin/python3 mapreduce/mapper2.py" \
    -reducer "/usr/bin/python3 mapreduce/reducer2.py" \
    -file mapreduce/mapper2.py \
    -file mapreduce/reducer2.py

if [ $? -ne 0 ]; then
    echo "Pipeline Stage 2 failed"
    exit 1
fi

echo "MapReduce pipeline completed successfully"
```

So, my index.sh is supposed to execute all the mappers and reducers in 2 stages, but the container exits mid-execution signaling about an error.

In conclusion, I tried to execute many different codelines, but my cluster-master reported an error every time, so here I’m just trying to elaborate about my work.