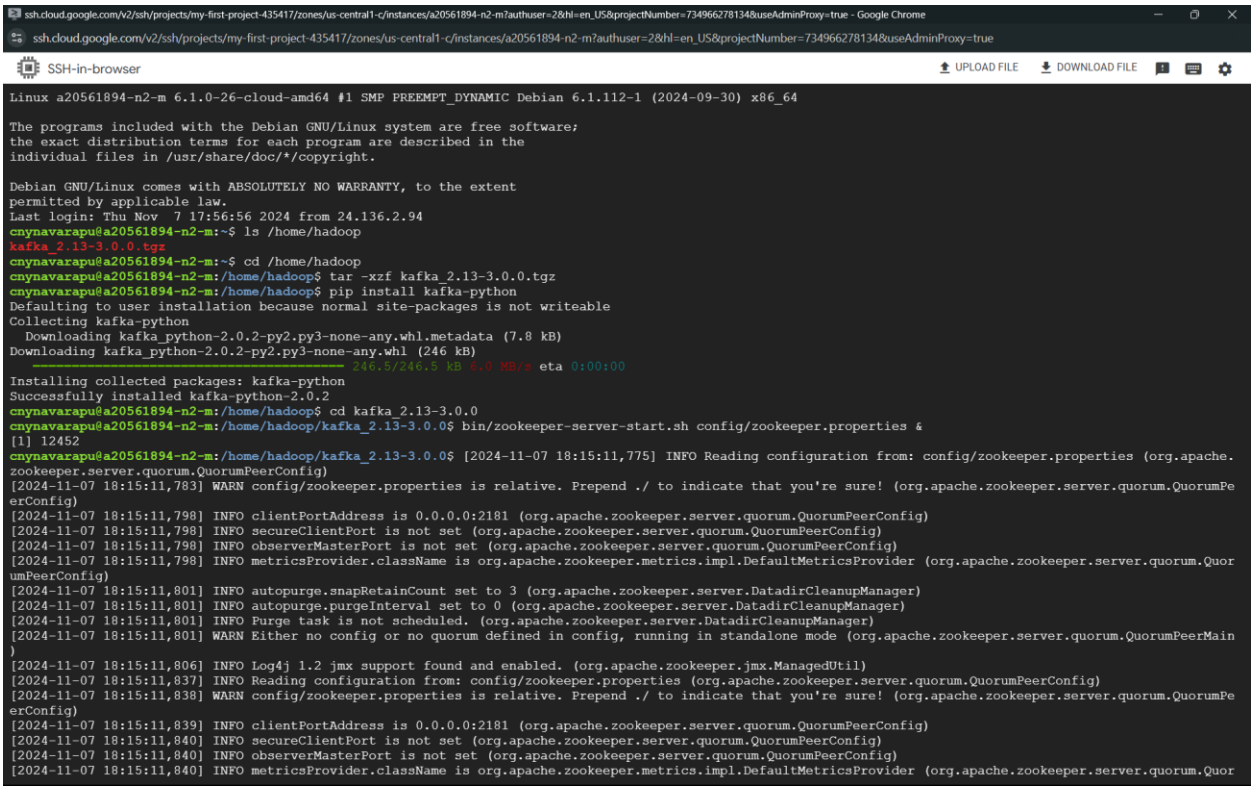
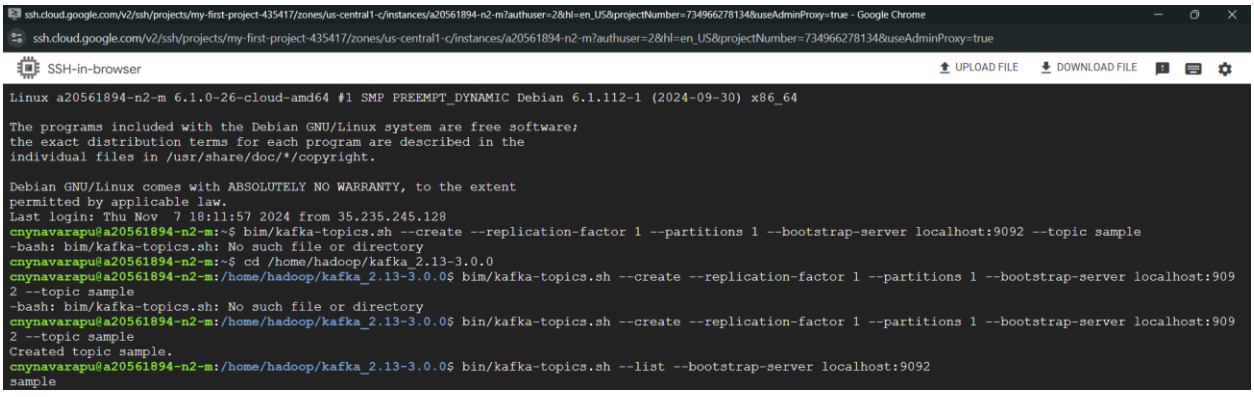
Chaitanya Durgesh Nynavarapu
A20561894

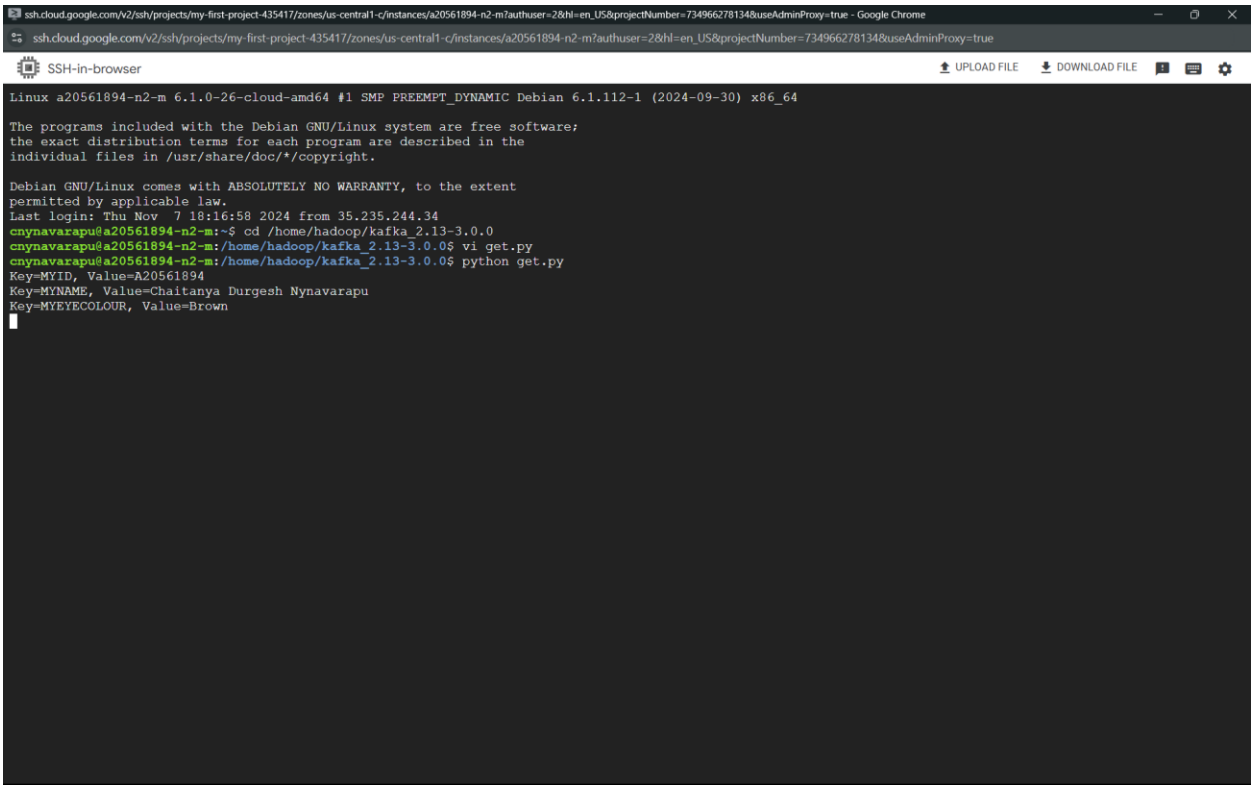
ASSIGNMENT - 9

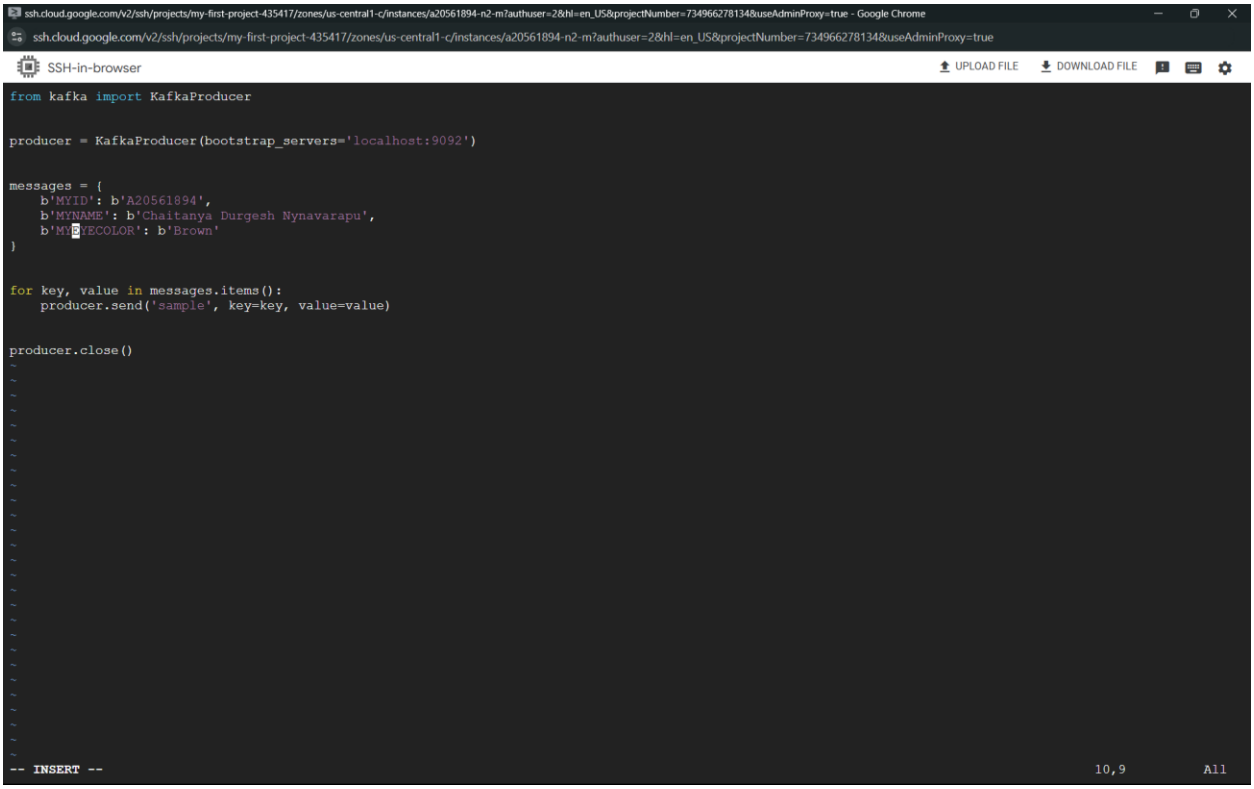
BIG DATA (CSP 554)

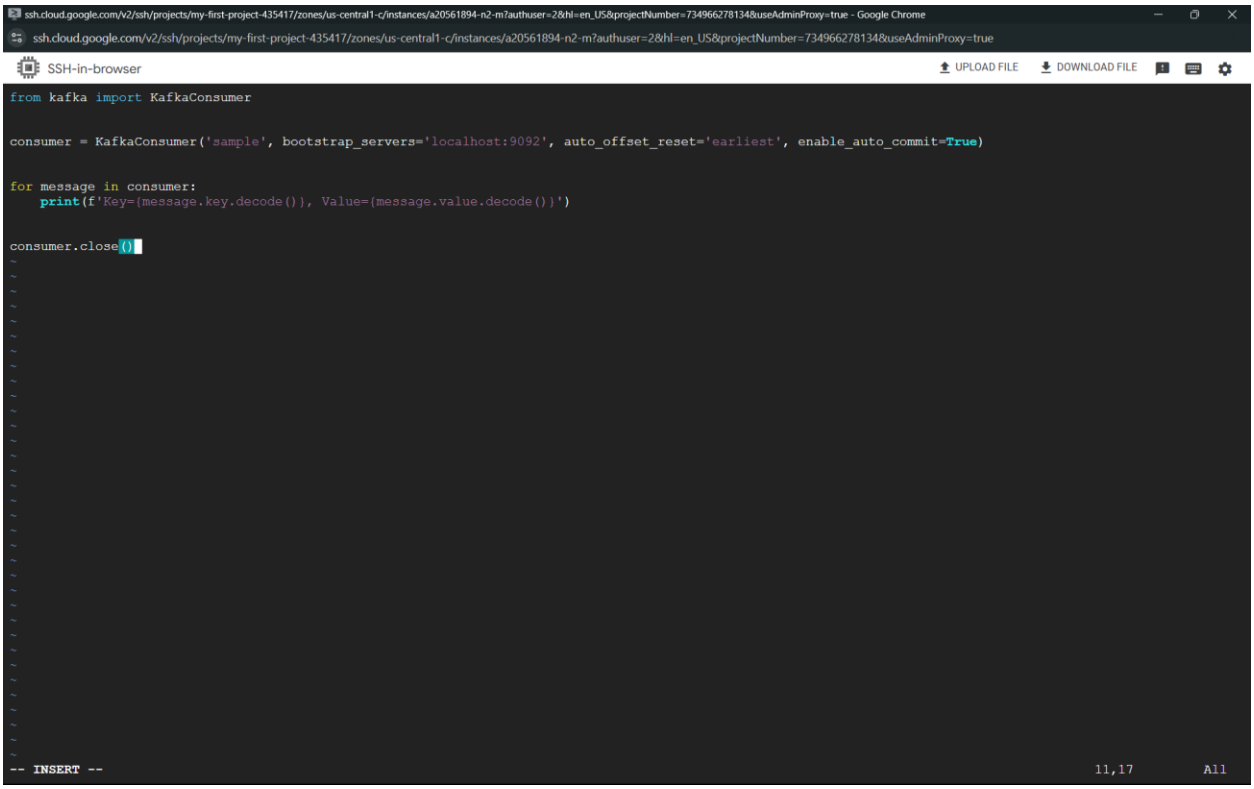
Exercise1:



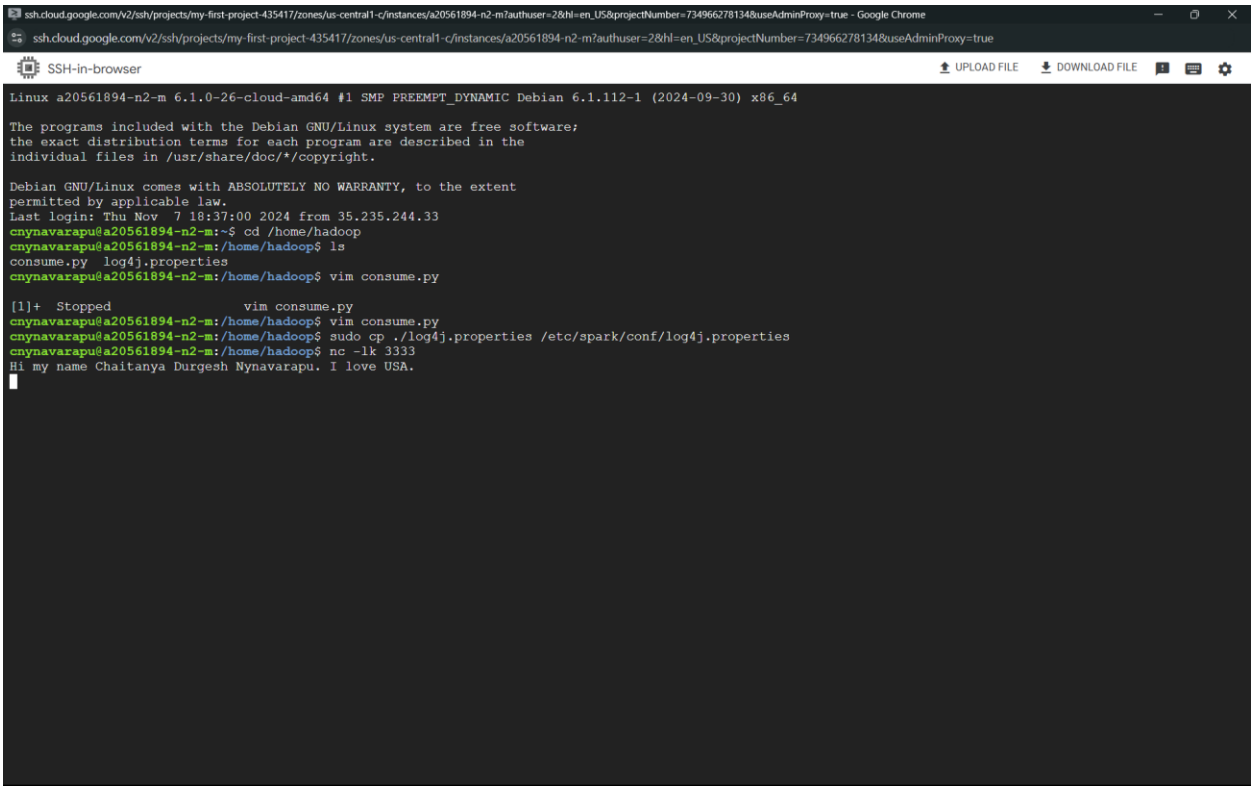


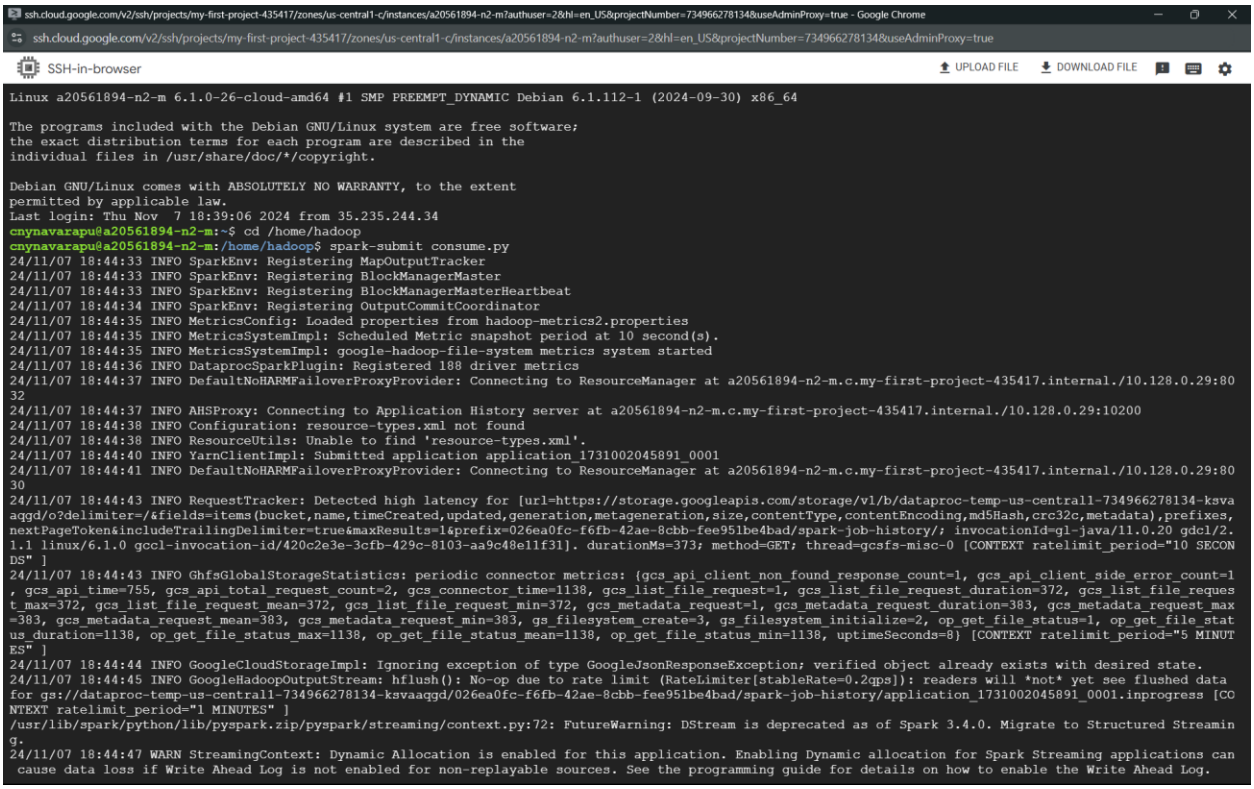

ssh.cloud.google.com/v2/ssh/projects/my-first-project-435417/zones/us-central1-c/instances/a20561894-n2-m?authuser=2&hl=en_US&projectNumber=734966278134&useAdminProxy=true - Google Chrome
ssh.cloud.google.com/v2/ssh/projects/my-first-project-435417/zones/us-central1-c/instances/a20561894-n2-m?authuser=2&hl=en_US&projectNumber=734966278134&useAdminProxy=true
SSH-in-browser
UPLOAD FILE
DOWNLOAD FILE
Linux a20561894-n2-m 6.1.0-26-cloud-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.112-1 (2024-09-30) x86_64
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Thu Nov 7 18:16:58 2024 from 35.235.244.34
cnynavarapu@a20561894-n2-m:~$ cd /home/hadoop/kafka_2.13-3.0.0
cnynavarapu@a20561894-n2-m:/home/hadoop/kafka_2.13-3.0.0$ vi get.py
cnynavarapu@a20561894-n2-m:/home/hadoop/kafka_2.13-3.0.0$ python get.py
Key=MYID, Value=A20561894
Key=MYNAME, Value=Chaitanya Durgesh Nynavarapu
Key=MYEYECOLOUR, Value=Brown

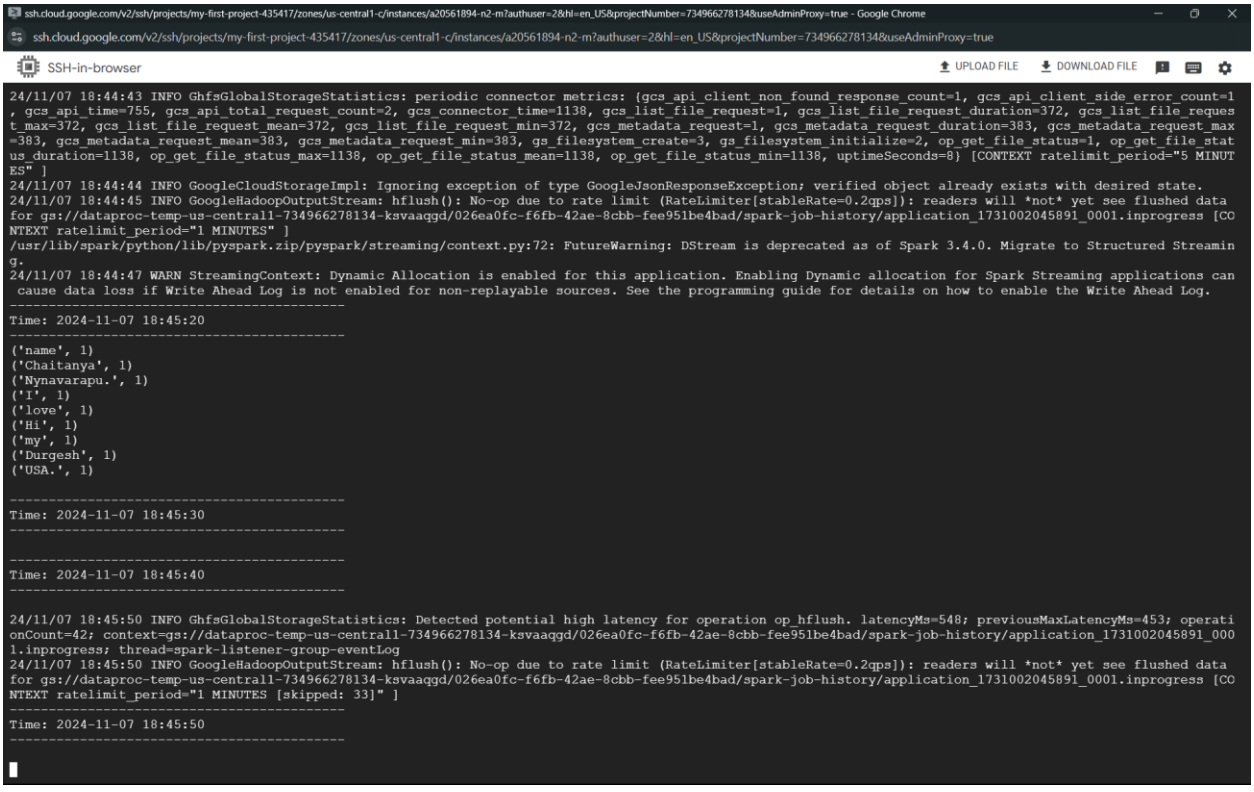

ssh.cloud.google.com/v2/ssh/projects/my-first-project-435417/zones/us-central1-c/instances/a20561894-n2-m?authuser=2&hl=en_US&projectNumber=734966278134&useAdminProxy=true - Google Chrome
ssh.cloud.google.com/v2/ssh/projects/my-first-project-435417/zones/us-central1-c/instances/a20561894-n2-m?authuser=2&hl=en_US&projectNumber=734966278134&useAdminProxy=true
SSH-in-browser
UPLOAD FILE
DOWNLOAD FILE
from kafka import KafkaProducer
producer = KafkaProducer(bootstrap_servers='localhost:9092')
messages = {
b'MYID': b'A20561894',
b'MYNAME': b'Chaitanya Durgesh Nynavarapu',
b'MYEYECOLOR': b'Brown'
}
for key, value in messages.items():
producer.send('sample', key=key, value=value)
producer.close()
-- INSERT --
10,9
All


ssh.cloud.google.com/v2/ssh/projects/my-first-project-435417/zones/us-central1-c/instances/a20561894-n2-m?authuser=2&hl=en_US&projectNumber=734966278134&useAdminProxy=true - Google Chrome
SSH-in-browser
UPLOAD FILE
DOWNLOAD FILE
from kafka import KafkaConsumer
consumer = KafkaConsumer('sample', bootstrap_servers='localhost:9092', auto_offset_reset='earliest', enable_auto_commit=True)
for message in consumer:
print(f'Key={message.key.decode()}, Value={message.value.decode()}')
consumer.close()
-- INSERT --
11,17
All

Exercise2:


ssh.cloud.google.com/v2/ssh/projects/my-first-project-435417/zones/us-central1-c/instances/a20561894-n2-m?authuser=2&hl=en_US&projectNumber=734966278134&useAdminProxy=true - Google Chrome
SSH-in-browser
UPLOAD FILE
DOWNLOAD FILE
Linux a20561894-n2-m 6.1.0-26-cloud-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.112-1 (2024-09-30) x86_64
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Thu Nov 7 18:37:00 2024 from 35.235.244.33
cnynavarapu@a20561894-n2-m:~$ cd /home/hadoop
cnynavarapu@a20561894-n2-m:/home/hadoop$ ls
consume.py log4j.properties
cnynavarapu@a20561894-n2-m:/home/hadoop$ vim consume.py
[1]+ Stopped vim consume.py
cnynavarapu@a20561894-n2-m:/home/hadoop$ vim consume.py
cnynavarapu@a20561894-n2-m:/home/hadoop$ sudo cp ./log4j.properties /etc/spark/conf/log4j.properties
cnynavarapu@a20561894-n2-m:/home/hadoop$ nc -lk 3333
Hi my name Chaitanya Durgesh Nynavarapu. I love USA.


ssh.cloud.google.com/v2/ssh/projects/my-first-project-435417/zones/us-central1-c/instances/a20561894-n2-m?authuser=2&hl=en_US&projectNumber=734966278134&useAdminProxy=true - Google Chrome
SSH-in-browser
UPLOAD FILE
DOWNLOAD FILE
Linux a20561894-n2-m 6.1.0-26-cloud-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.112-1 (2024-09-30) x86_64
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Thu Nov 7 18:39:06 2024 from 35.235.244.34
cnynavarapu@a20561894-n2-m:~$ cd /home/hadoop
cnynavarapu@a20561894-n2-m:/home/hadoop$ spark-submit consume.py
24/11/07 18:44:33 INFO SparkEnv: Registering MapOutputTracker
24/11/07 18:44:33 INFO SparkEnv: Registering BlockManagerMaster
24/11/07 18:44:33 INFO SparkEnv: Registering BlockManagerMasterHeartbeat
24/11/07 18:44:34 INFO SparkEnv: Registering OutputCommitCoordinator
24/11/07 18:44:35 INFO MetricsConfig: Loaded properties from hadoop-metrics2.properties
24/11/07 18:44:35 INFO MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
24/11/07 18:44:35 INFO MetricsSystemImpl: google-hadoop-file-system metrics system started
24/11/07 18:44:36 INFO DataprocSparkPlugin: Registered 188 driver metrics
24/11/07 18:44:37 INFO DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at a20561894-n2-m.c.my-first-project-435417.internal./10.128.0.29:8032
24/11/07 18:44:37 INFO AHSProxy: Connecting to Application History server at a20561894-n2-m.c.my-first-project-435417.internal./10.128.0.29:10200
24/11/07 18:44:38 INFO Configuration: resource-types.xml not found
24/11/07 18:44:38 INFO ResourceUtils: Unable to find 'resource-types.xml'.
24/11/07 18:44:40 INFO YarnClientImpl: Submitted application application_1731002045891_0001
24/11/07 18:44:41 INFO DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at a20561894-n2-m.c.my-first-project-435417.internal./10.128.0.29:8030
24/11/07 18:44:43 INFO RequestTracker: Detected high latency for [url=https://storage.googleapis.com/storage/v1/b/dataproc-temp-us-central1-734966278134-ksvaaqgd/o?delimiter=/&fields=items(bucket,name,timeCreated,updated,generation,metageneration,size,contentType,contentEncoding,md5Hash,crc32c,metadata),prefixes,nextPageToken&includeTrailingDelimiter=true&maxResults=1&prefix=026ea0fc-f6fb-42ae-8cbb-fee951be4bad/spark-job-history/; invocationId=gl-java/11.0.20 gdcl/2.1.1 linux/6.1.0 gccl-invocation-id/420c2e3e-3cfb-429c-8103-aa9c48e11f31]. durationMs=373; method=GET; thread=gcsfs-misc-0 [CONTEXT ratelimit_period="10 SECONDS"]
24/11/07 18:44:43 INFO GhfsGlobalStorageStatistics: periodic connector metrics: {gcs_api_client_non_found_response_count=1, gcs_api_client_side_error_count=1, gcs_api_time=755, gcs_api_total_request_count=2, gcs_connector_time=1138, gcs_list_file_request=1, gcs_list_file_request_duration=372, gcs_list_file_request_max=372, gcs_list_file_request_mean=372, gcs_list_file_request_min=372, gcs_metadata_request=1, gcs_metadata_request_duration=383, gcs_metadata_request_max=383, gcs_metadata_request_mean=383, gcs_metadata_request_min=383, gs_filesystem_create=3, gs_filesystem_initialize=2, op_get_file_status=1, op_get_file_status_duration=1138, op_get_file_status_max=1138, op_get_file_status_mean=1138, op_get_file_status_min=1138, uptimeSeconds=8} [CONTEXT ratelimit_period="5 MINUTES"]
24/11/07 18:44:44 INFO GoogleCloudStorageImpl: Ignoring exception of type GoogleJsonResponseException; verified object already exists with desired state.
24/11/07 18:44:45 INFO GoogleHadoopOutputStream: hflush(): No-op due to rate limit (RateLimiter[stableRate=0.2qps]): readers will *not* yet see flushed data for gs://dataproc-temp-us-central1-734966278134-ksvaaqgd/026ea0fc-f6fb-42ae-8cbb-fee951be4bad/spark-job-history/application_1731002045891_0001.inprogress [CONTEXT ratelimit_period="1 MINUTES"]
/usr/lib/spark/python/lib/pyspark.zip/pyspark/streaming/context.py:72: FutureWarning: DStream is deprecated as of Spark 3.4.0. Migrate to Structured Streaming.
24/11/07 18:44:47 WARN StreamingContext: Dynamic Allocation is enabled for this application. Enabling Dynamic allocation for Spark Streaming applications can cause data loss if Write Ahead Log is not enabled for non-replayable sources. See the programming guide for details on how to enable the Write Ahead Log.


ssh.cloud.google.com/v2/ssh/projects/my-first-project-435417/zones/us-central1-c/instances/a20561894-n2-m?authuser=2&hl=en_US&projectNumber=734966278134&useAdminProxy=true - Google Chrome
SSH-in-browser
UPLOAD FILE
DOWNLOAD FILE
24/11/07 18:44:43 INFO GhfsGlobalStorageStatistics: periodic connector metrics: {gcs_api_client_non_found_response_count=1, gcs_api_client_side_error_count=1, gcs_api_time=755, gcs_api_total_request_count=2, gcs_connector_time=1138, gcs_list_file_request=1, gcs_list_file_request_duration=372, gcs_list_file_request_max=372, gcs_list_file_request_mean=372, gcs_list_file_request_min=372, gcs_metadata_request=1, gcs_metadata_request_duration=383, gcs_metadata_request_max=383, gcs_metadata_request_mean=383, gcs_metadata_request_min=383, gs_filesystem_create=3, gs_filesystem_initialize=2, op_get_file_status=1, op_get_file_status_duration=1138, op_get_file_status_max=1138, op_get_file_status_mean=1138, op_get_file_status_min=1138, uptimeSeconds=8} [CONTEXT ratelimit_period="5 MINUTES"]
24/11/07 18:44:44 INFO GoogleCloudStorageImpl: Ignoring exception of type GoogleJsonResponseException; verified object already exists with desired state.
24/11/07 18:44:45 INFO GoogleHadoopOutputStream: hflush(): No-op due to rate limit (RateLimiter[stableRate=0.2qps]): readers will *not* yet see flushed data for gs://dataproc-temp-us-central1-734966278134-ksvaaqgd/026ea0fc-f6fb-42ae-8cbb-fee951be4bad/spark-job-history/application_1731002045891_0001.inprogress [CONTEXT ratelimit_period="1 MINUTES"]
/usr/lib/spark/python/lib/pyspark.zip/pyspark/streaming/context.py:72: FutureWarning: DStream is deprecated as of Spark 3.4.0. Migrate to Structured Streaming.
24/11/07 18:44:47 WARN StreamingContext: Dynamic Allocation is enabled for this application. Enabling Dynamic allocation for Spark Streaming applications can cause data loss if Write Ahead Log is not enabled for non-replayable sources. See the programming guide for details on how to enable the Write Ahead Log.

Time: 2024-11-07 18:45:20

('name', 1)
('Chaitanya', 1)
('Nynavarapu.', 1)
('I', 1)
('love', 1)
('Hi', 1)
('my', 1)
('Durgesh', 1)
('USA.', 1)

Time: 2024-11-07 18:45:30

Time: 2024-11-07 18:45:40

24/11/07 18:45:50 INFO GhfsGlobalStorageStatistics: Detected potential high latency for operation op_hflush. latencyMs=548; previousMaxLatencyMs=453; operationCount=42; context=gs://dataproc-temp-us-central1-734966278134-ksvaaqgd/026ea0fc-f6fb-42ae-8cbb-fee951be4bad/spark-job-history/application_1731002045891_0001.inprogress; thread=spark-listener-group-eventLog
24/11/07 18:45:50 INFO GoogleHadoopOutputStream: hflush(): No-op due to rate limit (RateLimiter[stableRate=0.2qps]): readers will *not* yet see flushed data for gs://dataproc-temp-us-central1-734966278134-ksvaaqgd/026ea0fc-f6fb-42ae-8cbb-fee951be4bad/spark-job-history/application_1731002045891_0001.inprogress [CONTEXT ratelimit_period="1 MINUTES [skipped: 33]"]

Time: 2024-11-07 18:45:50

Exercise3:

"Feasibility Analysis of AsterixDB and Spark Streaming with Cassandra for Stream-Based Processing"

It explores two prominent technologies—AsterixDB and Spark (with Cassandra as a data store)—to determine their effectiveness in processing streaming data. This type of data processing, crucial for real-time insights, is particularly important for applications like social media analysis. In the study, the authors use a simulated Twitter environment to test how each technology performs with large-scale tweet ingestion, word count, and sentiment analysis tasks.

Research Focus

The authors address two main questions:

1. How do AsterixDB and Spark + Cassandra perform in terms of processing speed and responsiveness for real-time content and sentiment analysis?
2. How well do these systems scale when additional processing nodes are introduced?

These questions are especially relevant as organizations increasingly need data processing solutions that are not only fast but also capable of handling larger loads without sacrificing performance.

Methodology

In a thoughtfully designed experimental setup, the authors use Eucalyptus, a cloud platform, to simulate the process of streaming data into both AsterixDB and Spark with Cassandra. They implement word count and sentiment analysis algorithms, the latter using the SentiWordNet 3.0 lexicon. To replicate a high-throughput social media setting, the authors also test the scalability of each setup by incrementally adding nodes and observing the impact on performance.

Key Findings

- **Performance:** AsterixDB demonstrated higher throughput and lower latency than Spark + Cassandra in processing the simulated tweet data. This advantage is due, in part, to AsterixDB's data feed feature, which streamlined the data ingestion process. In contrast, Spark's integration with Cassandra introduced delays due to the need for data serialization.
- **Scalability:** Both technologies improved with additional nodes, but AsterixDB scaled more effectively, maintaining consistent performance gains. Spark + Cassandra, meanwhile, faced challenges with increasing node counts due to serialization and communication overhead between Spark and Cassandra.
- **Sentiment Analysis:** For sentiment analysis, AsterixDB's use of an inverted index significantly boosted processing speed, making it an efficient choice for tasks that rely on lexicon-based analysis like SentiWordNet. Spark required substantial resources to achieve similar performance, highlighting AsterixDB's efficiency in handling intensive data processing.

Conclusion and Implications

In closing, the authors find that AsterixDB's architecture, particularly its data feeds and flexible data model, makes it a superior choice for real-time, high-throughput applications. Its advantages in both latency and scalability suggest that AsterixDB may better support businesses and applications where low-latency and large-scale data processing are critical. Spark with Cassandra, while effective, faces limitations under high loads and may be more suited to applications with moderate real-time processing demands.

This research provides valuable insights for those looking to implement or improve stream processing capabilities. By highlighting the comparative strengths of AsterixDB and Spark + Cassandra, the authors offer a helpful framework for selecting a solution tailored to specific real-time data processing needs.