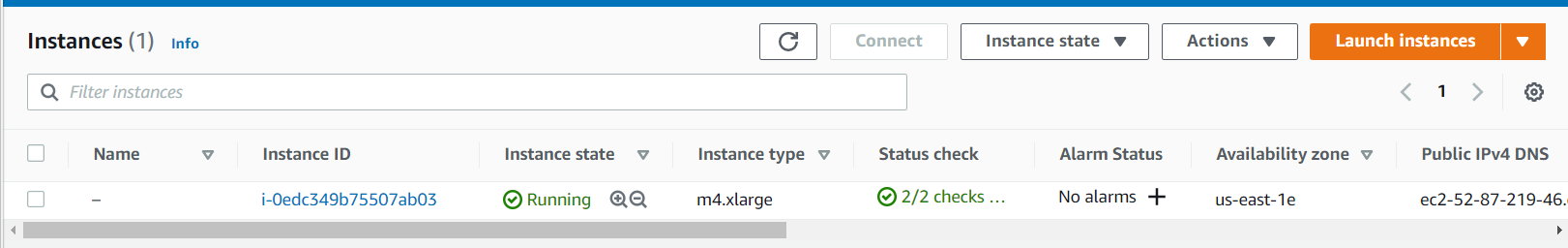
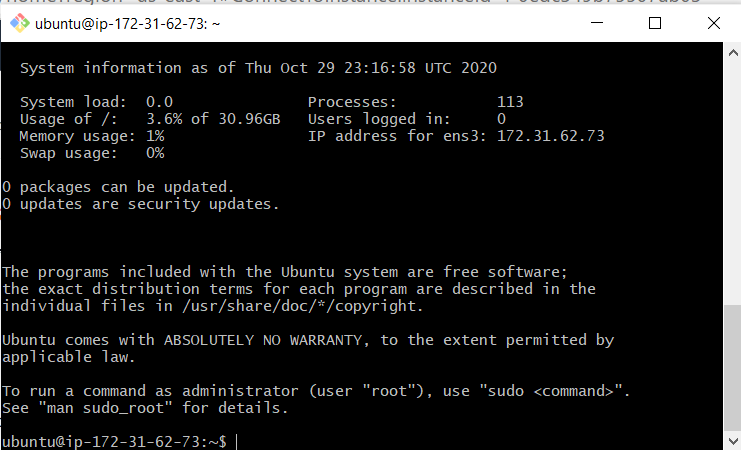
**Exercise 2)**

**EC2 Creation**

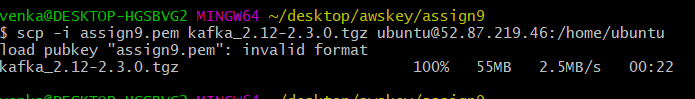


**Connecting to UButntu EC2 Server:**



**3)**

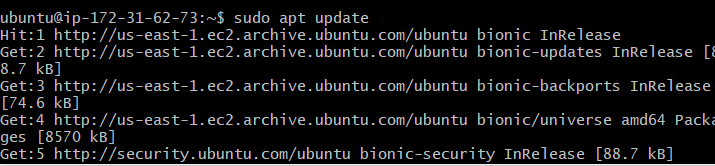
***Command Used:*** *scp -i assign9.pem kafka\_2.12-2.3.0.tgz ubuntu@52.87.219.46:/home/ubuntu*

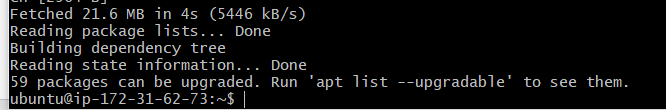




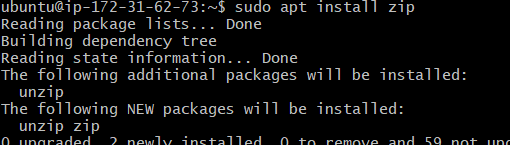
***4)***

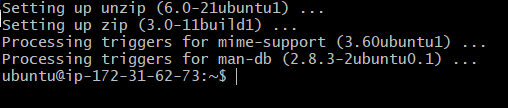
***I)******Command Used:*** *sudo apt update*



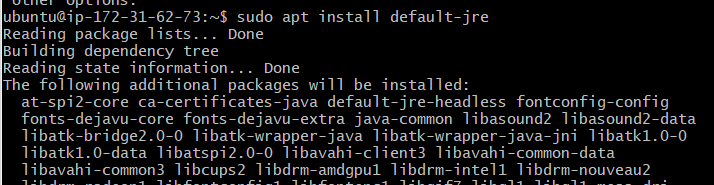


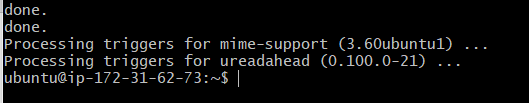
***II) Command Used:*** *sudo apt install zip*



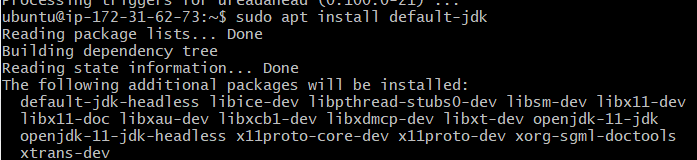


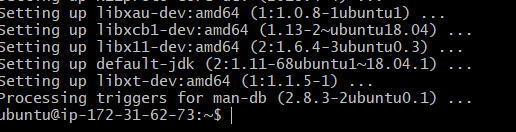
***III) Command Used :*** *sudo apt install default-jre*



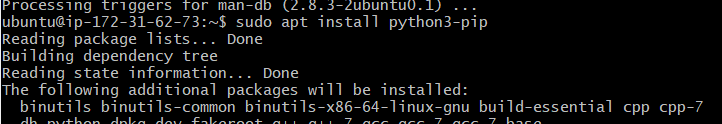


***IV) Command Used:*** *sudo apt install default-jdk*

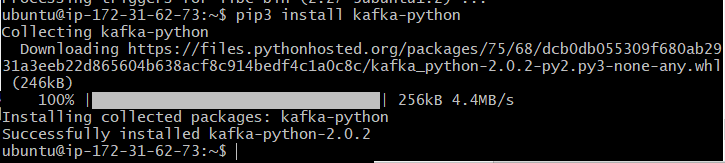




***V) Command Used:*** *sudo apt install python3-pip*

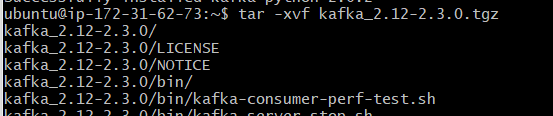


***VI) Command Used:****pip3 install kafka-python*



5) ***Entering a Command in KT2***

***Command Used:*** *tar -xvf kafka\_2.12-2.3.0.tgz*



6)

***Install location in KT2***

***Command Used:***

*export PATH=/home/ubuntu/kafka\_2.12-2.3.0/bin:$PATH*

*Then change to the /home/ubuntu/kafka\_2.12-2.3.0 directory:*

*cd /home/ubuntu/kafka\_2.12-2.3.0*





**SECTION 3**

1)

***Install location in KT3***

***Command Used:***

*export PATH=/home/ubuntu/kafka\_2.12-2.3.0/bin:$PATH*

*Then change to the /home/ubuntu/kafka\_2.12-2.3.0 directory:*

*cd /home/ubuntu/kafka\_2.12-2.3.0*



2) ***Install location in KT4***

***Command Used:***

*export PATH=/home/ubuntu/kafka\_2.12-2.3.0/bin:$PATH*

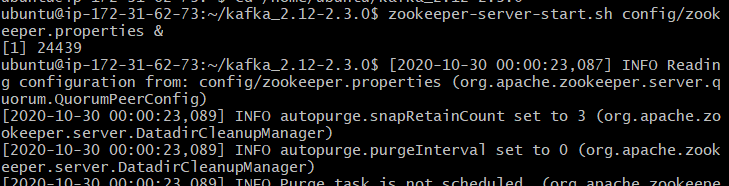
*Then change to the /home/ubuntu/kafka\_2.12-2.3.0 directory:*

*cd /home/ubuntu/kafka\_2.12-2.3.0*



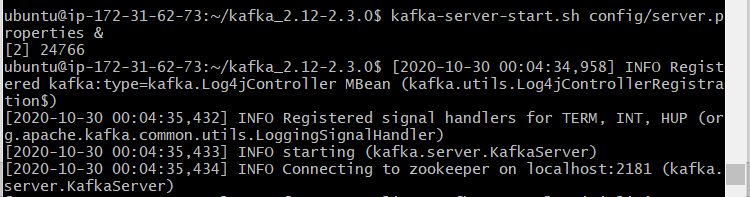
3)

***Command Used:*** *zookeeper-server-start.sh config/zookeeper.properties &*



4)

***Command Used :*** *kafka-server-start.sh config/server.properties &*



**Section 4**

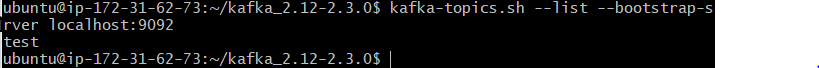
1)

***Command used:*** *kafka-topics.sh --create --bootstrap-server localhost:9092 --replication-factor 1 --partitions 1 --topic test*

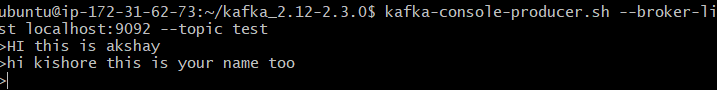


2)

***Command Used****: kafka-topics.sh --list --bootstrap-server localhost:9092*

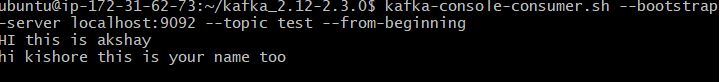


3)

***Command Used:***  *kafka-console-producer.sh --broker-list localhost:9092 --topic test*

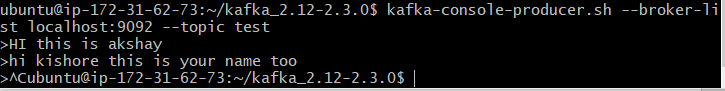
4)

***Command Used:*** *kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic test --from-beginning*



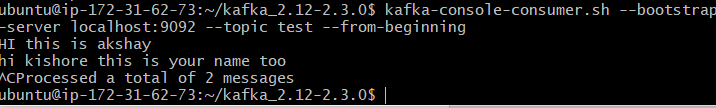
5) ***Ending Demo in KT3***

***Command Used: control +C***



***Ending Demo in KT4***

***Command Used: control +C***



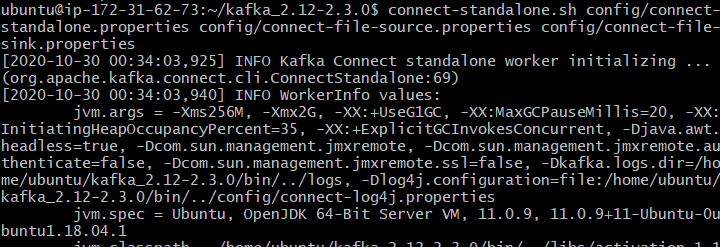
7)

***Command Used:***  *echo -e "foo\nbar" > test.txt*

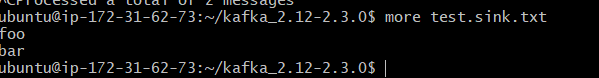


8)

***Command used:*** *connect-standalone.sh config/connect-standalone.properties config/connect-file-source.properties config/connect-file-sink.properties*

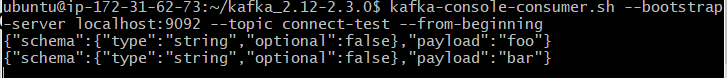


***Command Used:*** *more test.sink.txt*



9)

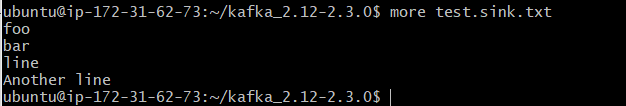
***Command Used:*** *kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic connect-test --from-beginning*



10)

***Command Used:*** *echo Another line>> test.txt*

***Command Used:*** *more test.sink.txt*



***Exercise 1)***

1. Kappa architecture is a stream processing system and to achieve this it employs a powerful stream processor capable of coping with data at a far greater rate than it is incoming. In lambda architecture all data is periodically recomputed in batch layer and in kappa architecture the only perform computation when the business logic changes by replaying historical data.

**Advantages:**

1. Purely stream-oriented systems provide very low latency and relatively high per-item cost. streaming provides scalable streaming system for data retention.
2. It restricts batch size to reduce latency.

**Drawbacks:**

1. Batch-oriented systems achieve unparalleled resource-efficiency at the expense of latency that is prohibitively high for real-time applications.
2. Trident processes one -at -time which is in favor of throughput than latency.
3. The pipeline in storm is called topology. The nodes that ingest data and initiate the data flow is called spouts. Spouts emit tuples to the nodes at downstream which are called bolts and do the processing. Storm distributes spouts and bolts across the nodes in the cluster in a round robin fashion. Storm does not provide any guarantee on the order in which tuples are processed, it provides the option of at-least-once processing through acknowledgement feature that tracks the processing.
4. Spark Streaming shifts Spark’s batch-processing approach towards real-time requirements by chunking the stream of incoming data items into small batches, transforming them into RDDs and processing them as usual. It further takes care of data flow and distribution automatically. Data is ingested and transformed into a sequence of RDDs which is called DStream before processing through workers.