strategy | consulting | digital | technology | operations

Apache Kafka

Concepts and Hands-on

Agenda

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

At the end of the session you will be able to understand

- **What is Data Streaming ?**
- **What is Apache Kafka?**
- **Kafka as a Messaging System**
- **Working with Kafka**
- **Simple Producer-Consumer Example using Single Node, Single Broker Kafka Cluster**
- **Simple Producer-Consumer Example using Single Node, Multi Broker Kafka Cluster**

Topics in detail

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

- **Introduction to Data Streaming**
  - What is Streaming Data?
  - Streaming Data Examples
  - Use Cases of Streaming Data
  - Comparison between Batch Processing and Stream Processing
  - Challenges in Working with Streaming Data
  - What is “streaming platform” ?
- **Introduction to Apache Kafka**
  - What is “Apache Kafka” ?
  - Use case for Apache Kafka

Topics in detail contd.

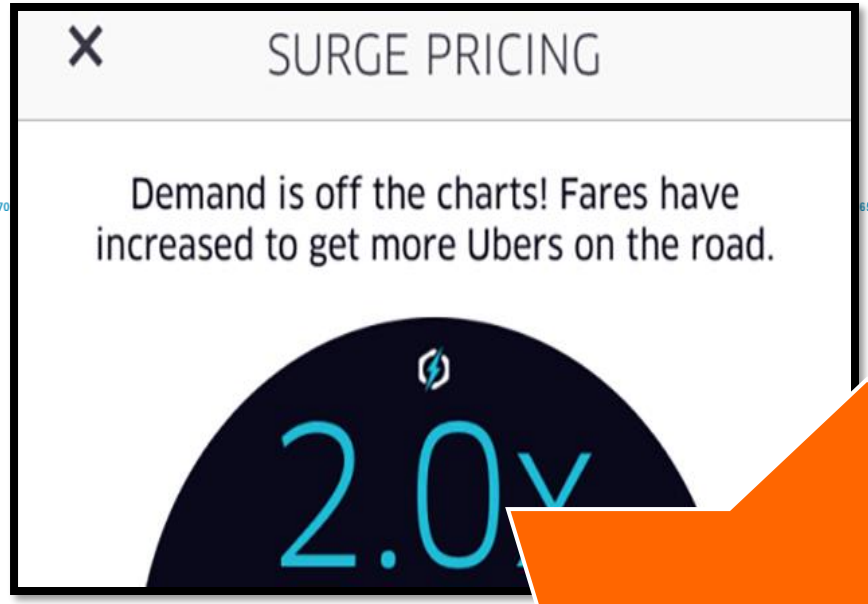
2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

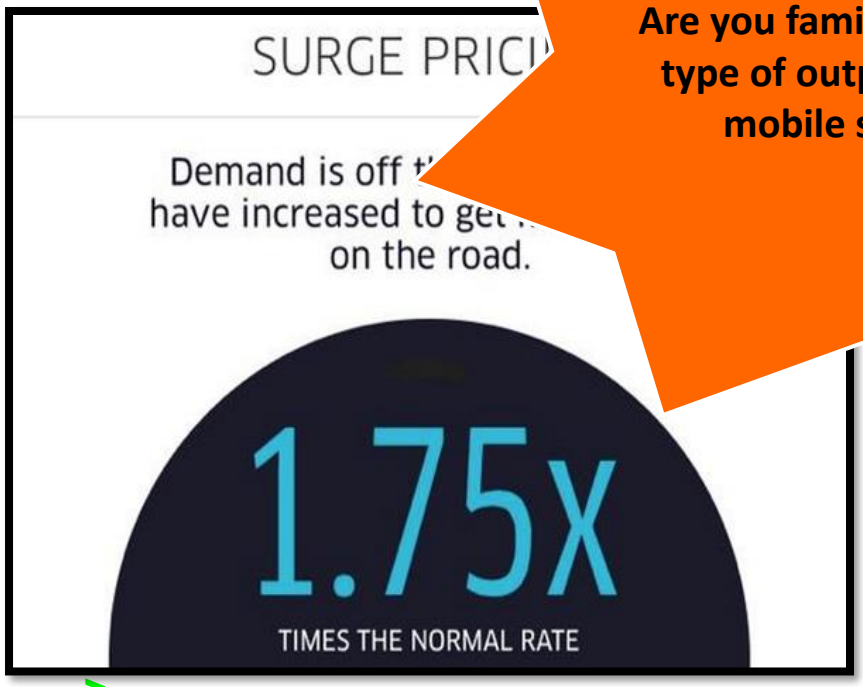
- **Working with Kafka**
  - Terminologies (Message, Producer, Consumer, Broker, Topic)
  - Core API of Kafka
  - Brief Architecture of Kafka Cluster
  - Generic messaging model
  - Kafka messaging model
  - Message Processing in Kafka

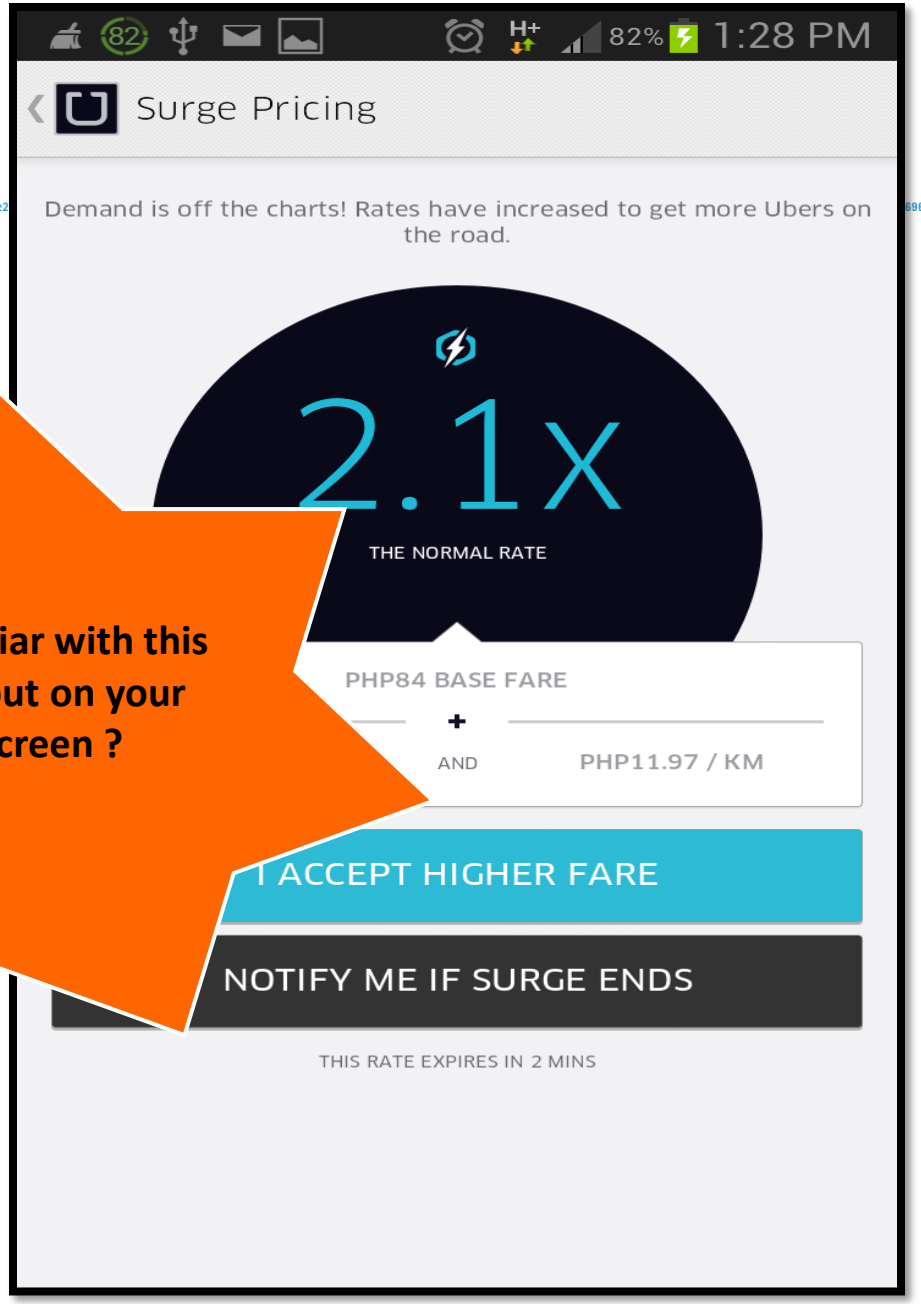
- **Simple Producer-Consumer Example using Single Node, Single Broker Kafka Cluster**

- **Simple Producer-Consumer Example using Single Node, Multi Broker Kafka Cluster**

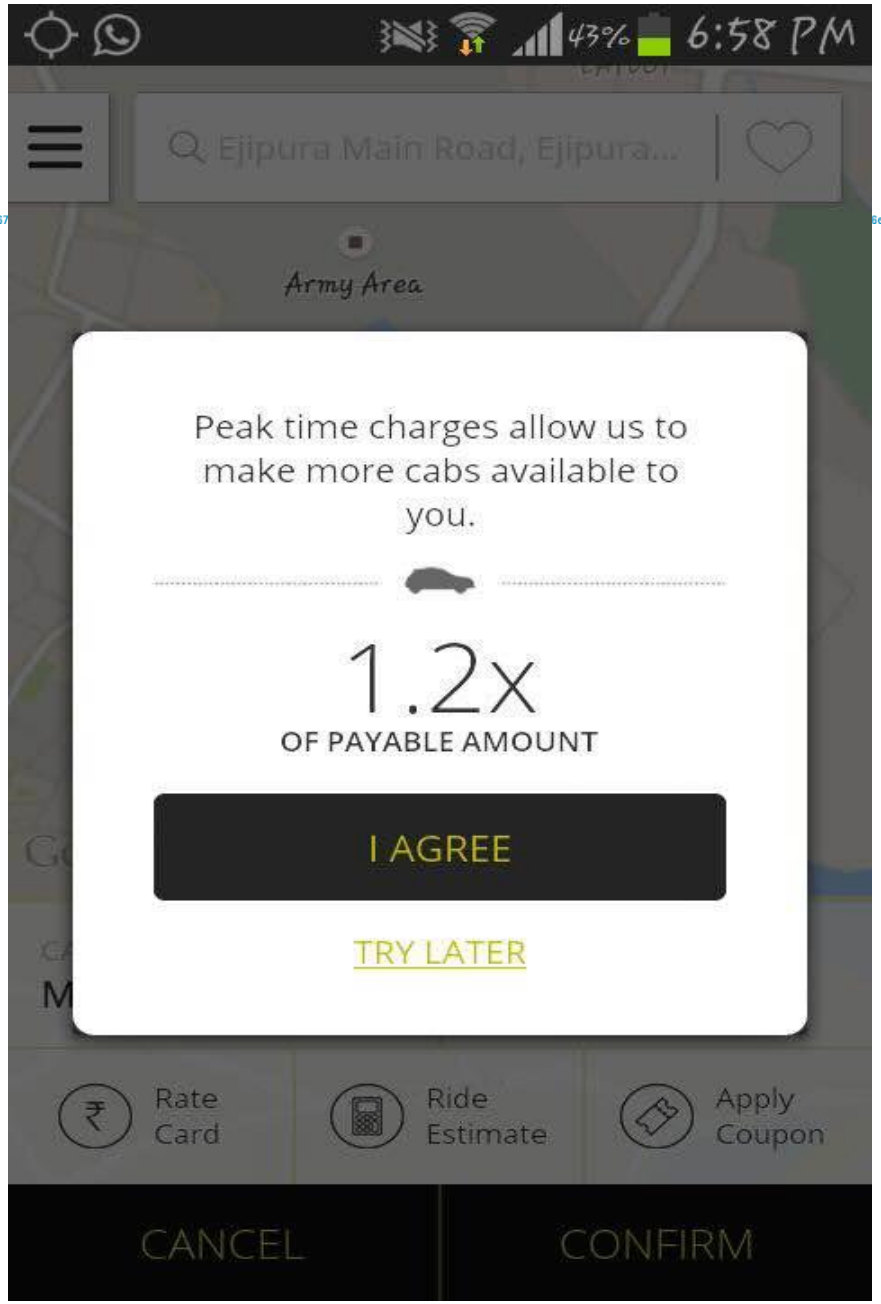
Introduction to Data Streaming



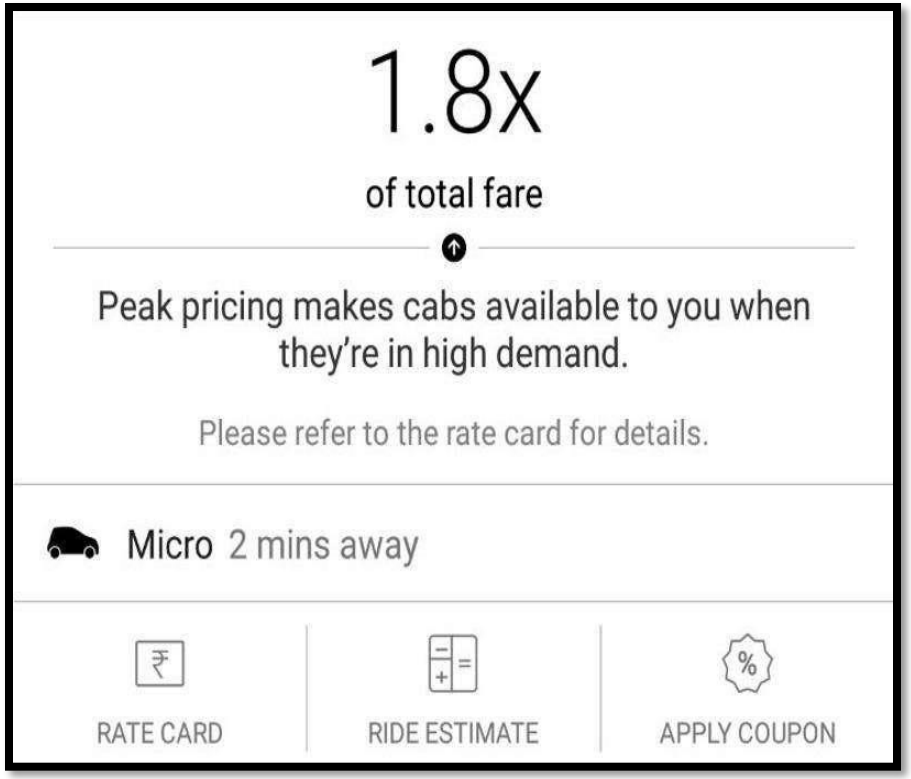




Are you familiar with this type of output on your mobile screen ?



Are you familiar with this type of display on your mobile screen ?



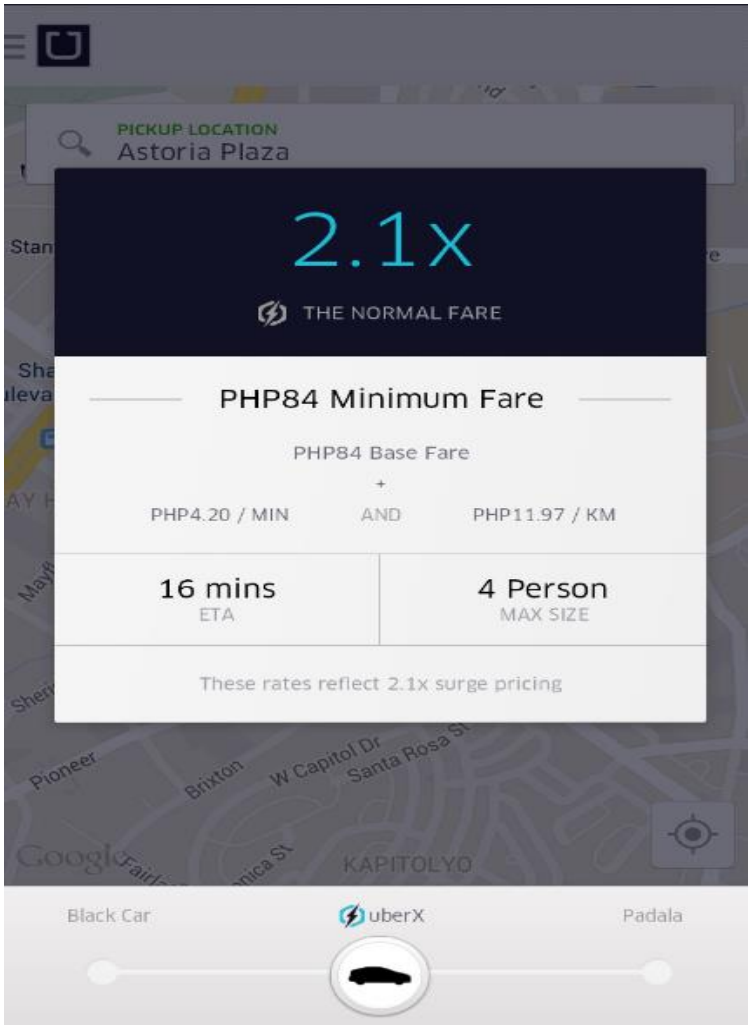
Set the Context

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

Surcharge information form Uber

When did we use to get this information ?

How does Uber apply surcharge ?

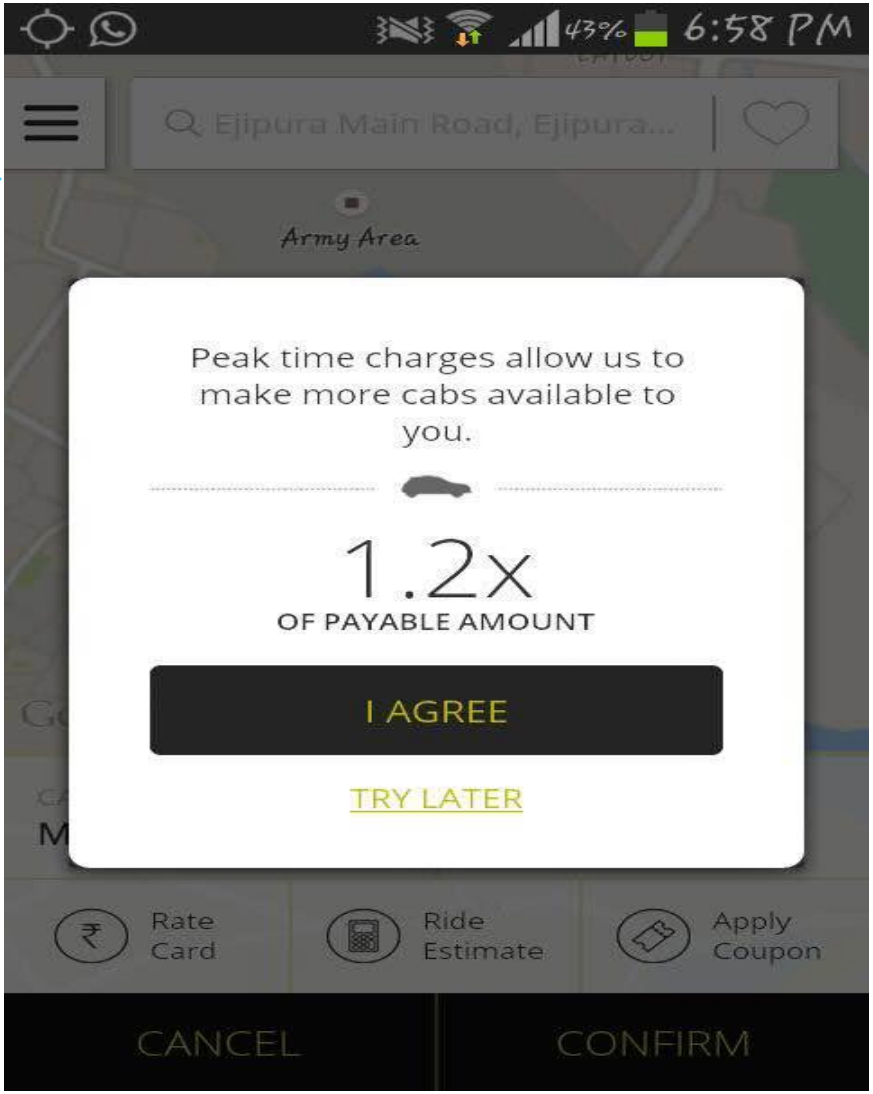


Set the Context

Surcharge information form OLA

When did we use to get this information ?

How does OLA apply surcharge ?



What is Streaming Data?

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

- **Streaming Data**
  - generated **<u>continuously by thousands of data sources</u>**, send in the data records simultaneously,
  - small sizes (order of Kilobytes).

Streaming Data Examples

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

- **Streaming data** includes a wide variety of data such as
  - **log files** generated by customers using your **mobile** or **web applications**,
  - ecommerce purchases,
  - in-game player activity,
  - information from social networks,
  - financial trading floors,
  - geospatial services,
  - telemetry from connected devices
  - instrumentation in data centers.

Use of Streaming Data

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

- Processed **sequentially** and **incrementally** on a record-by-record basis or over sliding time windows

- Used for generating wide variety of analytical reports which gives companies visibility into many aspects of their business and customer activity such as
  - **service usage (for metering/billing),**
  - **server activity,**
  - **website clicks, and**
  - **geo-location of devices, people, and physical goods** –and enables them to respond promptly to emerging situations.

For example,

Organization can track changes in **public sentiment** on their **brands** and **products** by continuously analysing **social media streams**, and respond in a timely fashion as the necessity arises.

Use of Streaming Data

2f486967682070657266 6f726d616c63652c2044656c6976657265642c2f486967682070657266 6f726d616c63652c2044656c6976657265642c2f486967682070657266 6f726d616c63652c2044656c6976657265642c2f486967682070657266 6f726d616c63652c2044656c6976657265642c2f486967682070657266 6f726d616c63652c2044656c6976657265642c2f486967682070657266 6f726d616c63652c2044656c6976657265642c2f486967682070

- Companies generally begin with simple applications such as collecting system logs and rudimentary processing like rolling min-max computations.
  - Then, these applications evolve to more sophisticated **near-real-time processing**.

Initially, applications may process data streams to produce simple reports, and perform simple actions in response, such as emitting alarms when key measures exceed certain thresholds.

Eventually, those applications perform more sophisticated forms of data analysis, like applying machine learning algorithms, and extract deeper insights from the data.

Over time, complex, stream and event processing algorithms are applied, further enriching the insights.

Batch Processing vs. Stream Processing

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

- *Batch processing* can be used to compute arbitrary queries over different sets of data.
  - Usually computes results that are derived from all the data it encompasses, and enables deep analysis of **big data sets**.
  - MapReduce-based systems - like Amazon EMR, are examples of platforms that support batch jobs.

- In contrast, *stream processing* requires ingesting a sequence of data, and incrementally updating metrics, reports, and summary statistics in response to **each arriving data record**.
  - It is better suited for **real-time monitoring and response** functions.

Batch Processing vs Stream Processing

2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070

	Batch processing	Stream processing
Data scope	Queries or processing over all or most of the data in the dataset.	Queries or processing over data within a **rolling time window**, or on just the **most recent data record**.
Data size	Large batches of data.	Individual records or micro batches consisting of a few records.
Performance	Latencies in minutes to hours.	Requires latency in the order of seconds or milliseconds.
Analyses	Complex analytics.	Simple response functions, aggregates, and rolling metrics.

Stream Processing + Batch Processing

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

- Many organizations are building a **hybrid model** by combining the <u>two approaches</u>, and maintain **a real-time layer** and **a batch layer**.
  - Data is **first** processed by a **streaming data platform** to extract **real-time insights**, and then
  - Persisted(stored) into a store like HDFS, where it can be transformed and loaded for a variety of **batch processing use cases**.

What is “ streaming platform ” ?

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

- Streaming data processing requires two layers:
  - **a storage layer** and
  - **a processing layer**

Storage layer

- Support record ordering and strong consistency
- Enable fast, inexpensive, and replayable reads and writes of large streams of data.

Processing layer

- **C**onsumes data from the **storage layer**,
- **Runs** computations on that data, and then
- Notify the storage layer to delete data that is no longer needed.

What is “ streaming platform ” ?

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

A streaming platform can be thought of as having three key capabilities:

- It lets us **publish and subscribe** to <u>streams of records</u>.
- It lets us **store streams of records** in a fault-tolerant way.
- It lets you **process streams of records** as they occur.
- It should also support **scalability**, **data durability**, and **fault tolerance** in both the **<u>storage</u>** and **<u>processing</u>** <u>layers</u>.

Example of “ streaming platform ” ?

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

- Many platforms have emerged that provide the **infrastructure** needed to build streaming data applications , like
  - Amazon Kinesis Streams,
  - **Apache Kafka**,
  - Apache Flume,
  - Apache Spark Streaming, and
  - Apache Storm.

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

Introduction to Apache Kafka

What is “Apache Kafka” ?

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

- **Apache Kafka** is an open-source ***distributed* stream processing platform** developed by the Apache Software Foundation written in **Scala** and **Java**.
  - Provide a unified, high-throughput, low-latency platform for handling real-time data feeds.
- Its **storage layer** is essentially a *"massively scalable pub/sub message queue architected as a distributed transaction log*," to process streaming data.
- Kafka connects to external systems (for data import/export) via Kafka Connect
- It also provides Kafka Streams, a Java stream processing library.
- The design is heavily influenced by **transaction logs**.

Use cases for Kafka

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

It gets used for two broad classes of application:

- Building **real-time streaming data pipelines**
- Building **real-time streaming appl**ications

Kafka - A Brief History

2f486967682070657266f726d616c63652c2044656c6976657265642c2f486967682070657266f726d616c63652c2044656c6976657265642c2f486967682070657266f726d616c63652c2044656c6976657265642c2f486967682070657266f726d616c63652c2044656c6976657265642c2f486967682070657266f726d616c63652c2044656c6976657265642c2f486967682070

- Apache Kafka was originally developed by LinkedIn, and was subsequently open sourced in early 2011.
- Graduation from the Apache Incubator occurred on 23 October 2012.
- In November 2014, several engineers who worked on Kafka at **LinkedIn** created a new company named **Confluent** with a focus on Kafka.

Apache Kafka[1]

Developer(s)	Apache Software Foundation
Initial release	January 2011; 6 years ago[2]
Stable release	0.10.20 / February 22, 2017; 2 months ago
Repository	git-wip-us.apache.org/repos/asf/kafka.git
Development status	Active
Written in	Scala, Java
Operating system	Cross-platform
Type	Stream processing, Message broker
License	Apache License 2.0
Website	kafka.apache.org

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

Kafka as a Messaging System

Terminologies

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

- Kafka Broker
  - Kafka is run as a **cluster on one or more servers , called <u>Kafka Broker</u>**
  - Responsible for storing and processing of messages

- Producer
  - Produce messages/stream of records
  - **Publish** the messages/stream of records to **topics**

- Topic
  - The Kafka cluster **stores streams of *records*** in <u>categories</u> called ***topics***.
  - <u>Each record</u> consists of **a key**, **a value**, and **a timestamp**.

- Consumer
  - **Subscribe topics**
  - Consume the messages/stream of records from **topics**

Core API of Kafka

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

Kafka has four core APIs:

- **Producer API**
  - allows an application to **publish a stream of records** to one or more **Kafka topics**.
- **Consumer API**
  - allows an application to **subscribe to one or more topics** and **process the stream of records** produced to them.

Core API of Kafka

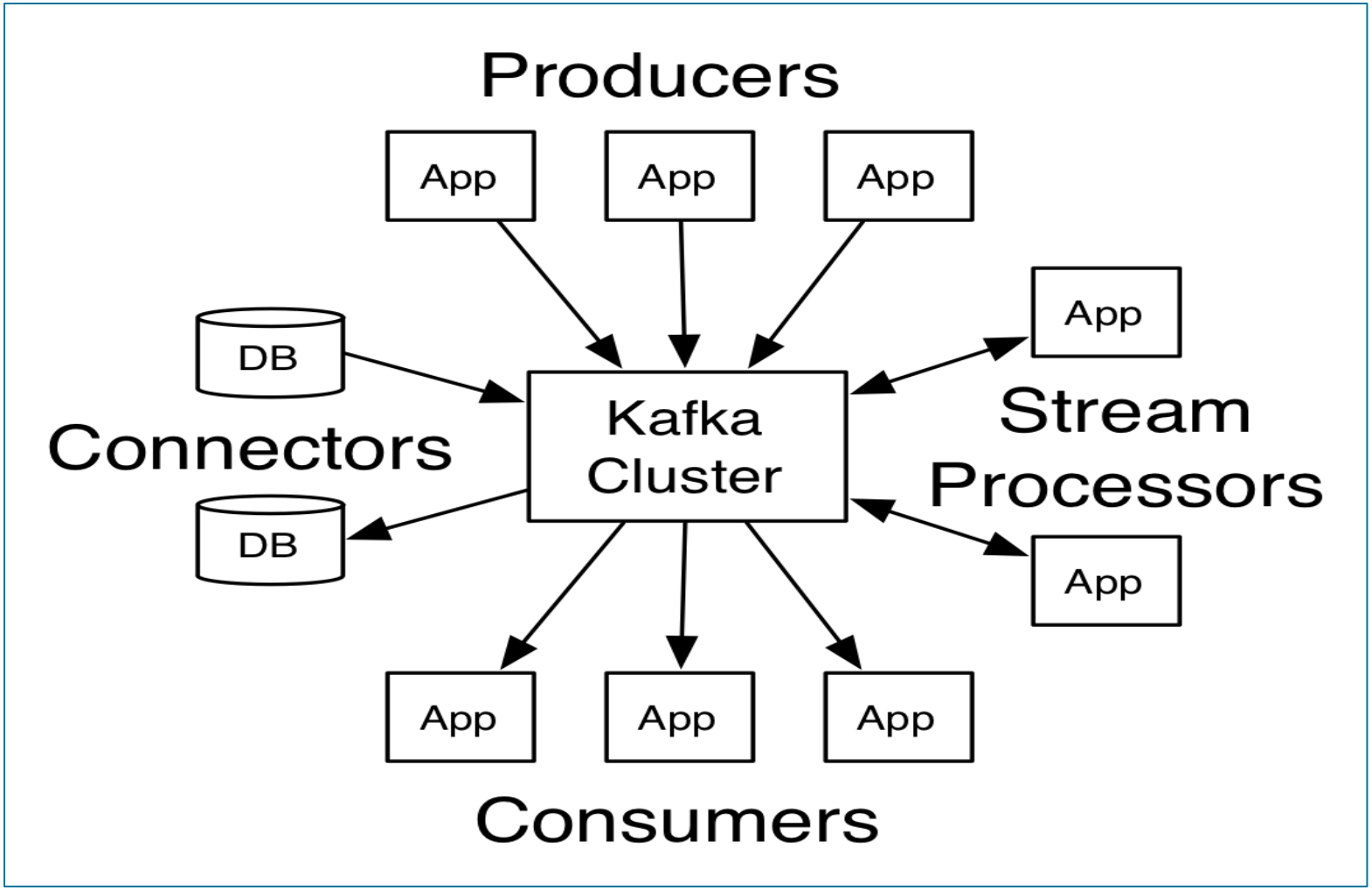
2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

- **Streams API**
  - allows an application to **act as a *stream processor***,
  - consuming an input stream from one or more topics and
  - producing an output stream to one or more output topics,
  - *effectively transforming the input streams to output streams*.

- **Connector API**
  - allows **building and running** reusable producers or consumers that connect **Kafka topics** to existing applications or data systems.
  - For example, *a connector to a relational database might capture every change to a table*.

Core API of Kafka

2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070



2f486967682070657266f726d616e63652c2044656c69766657265642c2f486967682070657266f726d616e63652c2044656c69766657265642c2f486967682070657266f726d616e63652c2044656c69766657265642c2f486967682070657266f726d616e63652c2044656c69766657265642c2f486967682070657266f726d616e63652c2044656c69766657265642c2f486967682070

Working with Kafka

Single Node – Single Broker Kafka Cluster

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

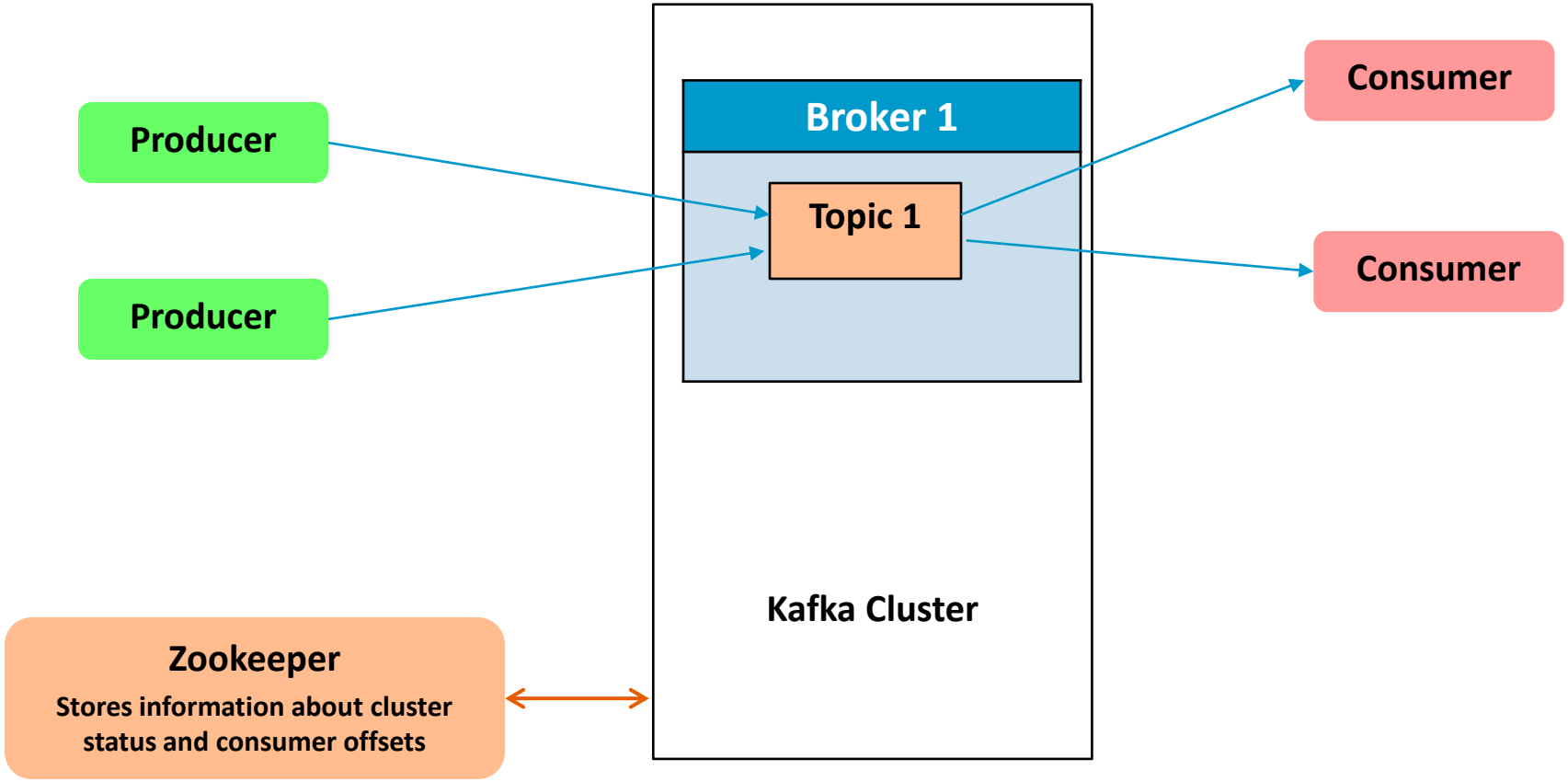
To start working with Kafka within a Single Node (<u>Single Node Kafka Cluster</u>) , the following services/components are required –

- **Zookeeper**
  - **Stores information about cluster status and consumer offsets**
- At least one **Kafka Server,** also called **Kafka Broker**
  - **Stores Topics**
  - **Maintain partitions of Topic**
  - **Maintain replication of Topic**

- **Topics**
  - **Messages are stored within Topic**
- **Producer**
  - **Publish messages to Topic**
- **Consumer**
  - **Consumes messages from Topic**

Single Node – Single Broker Kafka Cluster

2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070

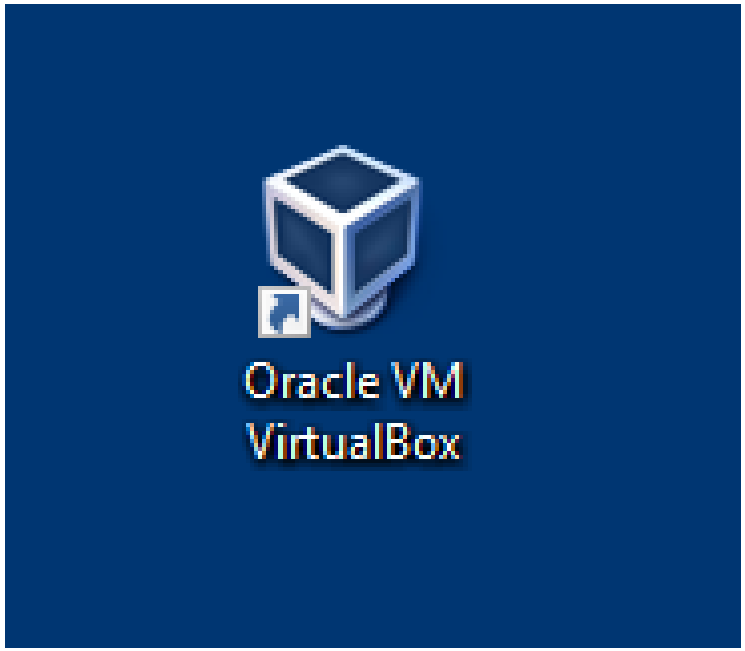


2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

If your Single Node Hadoop Cluster on Oracle Virtual Box is not ready , click here and follow the steps to install it.

Start your environment

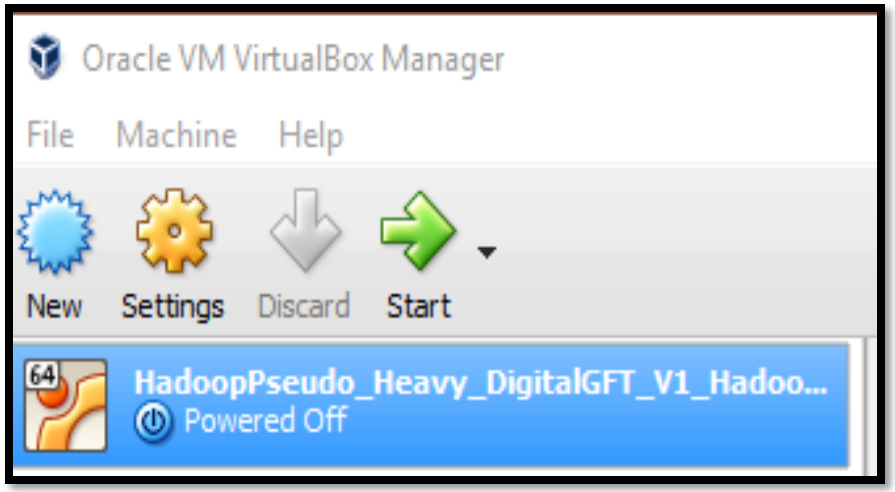
Open the Oracle VM Virtual Box



Start your environment (From powered off mode)

2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070

- Open Oracle Virtual Box
- If your Linux box is **powered off** in the Virtual Box, please select the Linux Box → click on Start



User Name :- vagrant

Password :- vagrant

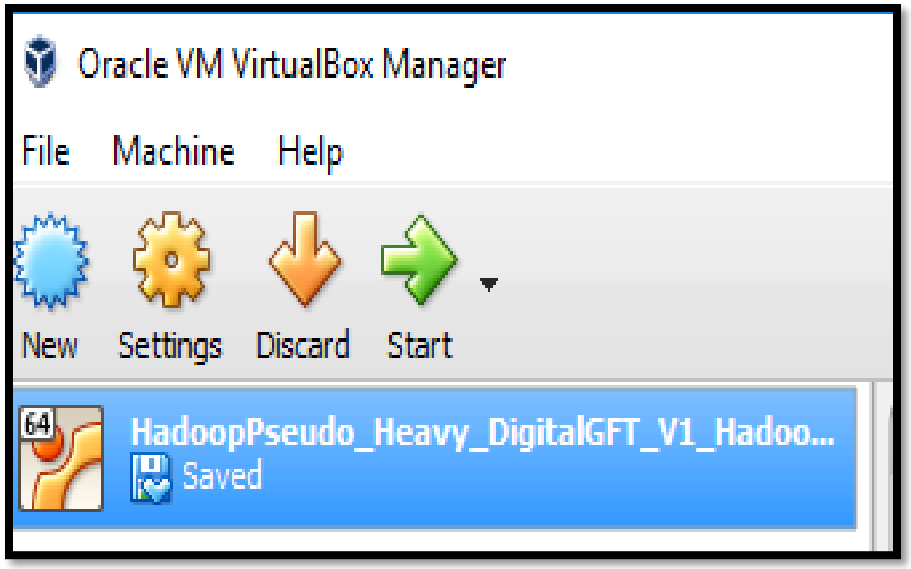
IP Address :- **192.168.56.70**

Connect the Linux Node through SmarTTY client

Start your environment (From saved mode)

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

- Open Oracle Virtual Box
- If your Linux box is in "**Saved Mode**" in the Virtual Box, please select the Linux Box → click on Start



User Name :- vagrant

Password :- vagrant

IP Address :- 192.168.56.70

Connect the Linux Node through SmarTTY client

Check the services in the Hadoop Node

2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070

Check the Hadoop services

$ jps

```
1491 NameNode
1751 SecondaryNameNode
2087 NodeManager
1976 ResourceManager
1595 DataNode
2123 Jps
```

If the Services are running , Stop all services

$ stop-dfs.sh

$ stop-yarn.sh

Stop all the Hadoop services

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

If the Services are running , Stop all services

$ stop-yarn.sh

```
stopping yarn daemons
stopping resourcemanager
localhost: stopping nodemanager
localhost: nodemanager did not stop gracefully after 5 seconds: killing with kill -9
no proxyserver to stop
```

$ stop-dfs.sh

```
Stopping namenodes on [localhost]
localhost: stopping namenode
localhost: stopping datanode
Stopping secondary namenodes [0.0.0.0]
0.0.0.0: stopping secondarynamenode
```

Check the Hadoop services

$ jps

```
vagrant@master:~$ jps
4207 Jps
vagrant@master:~$ _
```

Checking the Kafka Environment

Checking the Kafka Installation

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

Check "bigdata" directory under your Linux Home Directory (/home/vagrant)

```
vagrant@master:~$ ls bigdata/
```

```
cassandra   hadoop_tmp   hive       java   pig   scala
hadoop      hbase        hive_tmp   kafka  sbt   spark
```

Kafka installation directory

Note :- If Kafka is already installed , skip the next two slides

Installing Kafka if NOT installed

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

If Kafka is **NOT installed** in your Linux Node , please follow the following steps to install it

- **Download Kafka**

vagrant@master:~$ wget http://apache.mirror.gtcomm.net/kafka/0.10.2.1/kafka_2.12-0.10.2.1.tgz

```
Saving to: `kafka_2.12-0.10.2.1.tgz'

100%[=================================================================>]
```

- **Check the Downloaded file**

vagrant@master:~$ ls

```
kafka_2.12-0.10.2.1.tgz
```

Installing Kafka if NOT installed

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

- Extract the kafka_2.12-0.10.2.1.tgz file

```
vagrant@master:~$ tar xf kafka_2.12-0.10.2.1.tgz
vagrant@master:~$ ls
```

```
kafka_2.12-0.10.2.1
kafka_2.12-0.10.2.1.tgz
```

- Move the `kafka_2.12-0.10.2.1` directory to "bigdata" directory and rename it as "kafka"

```
vagrant@master:~$ mv kafka_2.12-0.10.2.1  bigdata/kafka
```

Checking Kafka Installation

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

```
vagrant@master:~$ cd bigdata/kafka/

vagrant@master:~/bigdata/kafka$ ls
```

```
bin  config  libs  LICENSE  NOTICE  site-docs
```

Checking the Kafka Installation contd.

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

```
vagrant@master:~/bigdata/kafka$ ls bin
```

```
connect-distributed.sh
connect-standalone.sh
kafka-acls.sh
kafka-broker-api-versions.sh
kafka-configs.sh
kafka-console-consumer.sh
kafka-console-producer.sh
kafka-consumer-groups.sh
kafka-consumer-offset-checker.sh
kafka-consumer-perf-test.sh
kafka-mirror-maker.sh
kafka-preferred-replica-election.sh
kafka-producer-perf-test.sh
kafka-reassign-partitions.sh
kafka-replay-log-producer.sh
kafka-replica-verification.sh
kafka-run-class.sh
kafka-server-start.sh
kafka-server-stop.sh
kafka-simple-consumer-shell.sh
kafka-streams-application-reset.sh
kafka-topics.sh
kafka-verifiable-consumer.sh
kafka-verifiable-producer.sh
windows
zookeeper-security-migration.sh
zookeeper-server-start.sh
zookeeper-server-stop.sh
zookeeper-shell.sh
```

"bin" directory contains all the service related commands of Kafka

Checking the Kafka Installation contd.

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

```
vagrant@master:~/bigdata/kafka$ ls config
```

"config" directory contains the property files related to different Kafka Services

Property file for Zookeeper Service

Property file for Kafka Broker Service

```
connect-console-sink.properties     consumer.properties
connect-console-source.properties   log4j.properties
connect-distributed.properties      producer.properties
connect-file-sink.properties        server.properties
connect-file-source.properties      tools-log4j.properties
connect-log4j.properties            zookeeper.properties
connect-standalone.properties
```

Checking the Kafka Installation contd.

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

Check the Property file for **Zookeeper Service** →

```
vagrant@master:~/bigdata/kafka$ vi config/zookeeper.properties
```

```
# the port at which the clients will connect
clientPort=2181
```

Check the **Port Number** for the Zookeeper Service

Press <**ESC**> key followed by **:q** → to quit from **vi** editor

Check the following properties of Kafka Server

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

Check the Property file for **Kafka Broker Service** →

```
vagrant@master:~/bigdata/kafka$ vi config/server.properties
```

Check the following properties for Kafka Broker

```
broker.id=0

#listeners=PLAINTEXT://:9092

log.dirs=/tmp/kafka-logs

zookeeper.connect=localhost:2181
```

Port Number for the **Kafka Server**

Location where streaming messages are stored in Kafka Server

Check the following properties of Kafka Server

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

```
# The id of the broker. This must be set to a unique integer
 for each broker.
broker.id=0
```

```
#listeners=PLAINTEXT://:9092
```

```
# A comma seperated list of directories under which to store
 log files
log.dirs=/tmp/kafka-logs
```

```
zookeeper.connect=localhost:2181
```

Press <ESC> key followed by :q → to quit from vi editor

Message Handling in Kafka

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

- Kafka provides an **abstraction** for a stream of records/messages— **the topic**.
- **Records are published to a topic**
- **Topics** in Kafka are ***always multi-subscriber***;
  - **a topic can have zero, one, or many consumers that subscribe to the data written to it.**

Message Handling in Kafka

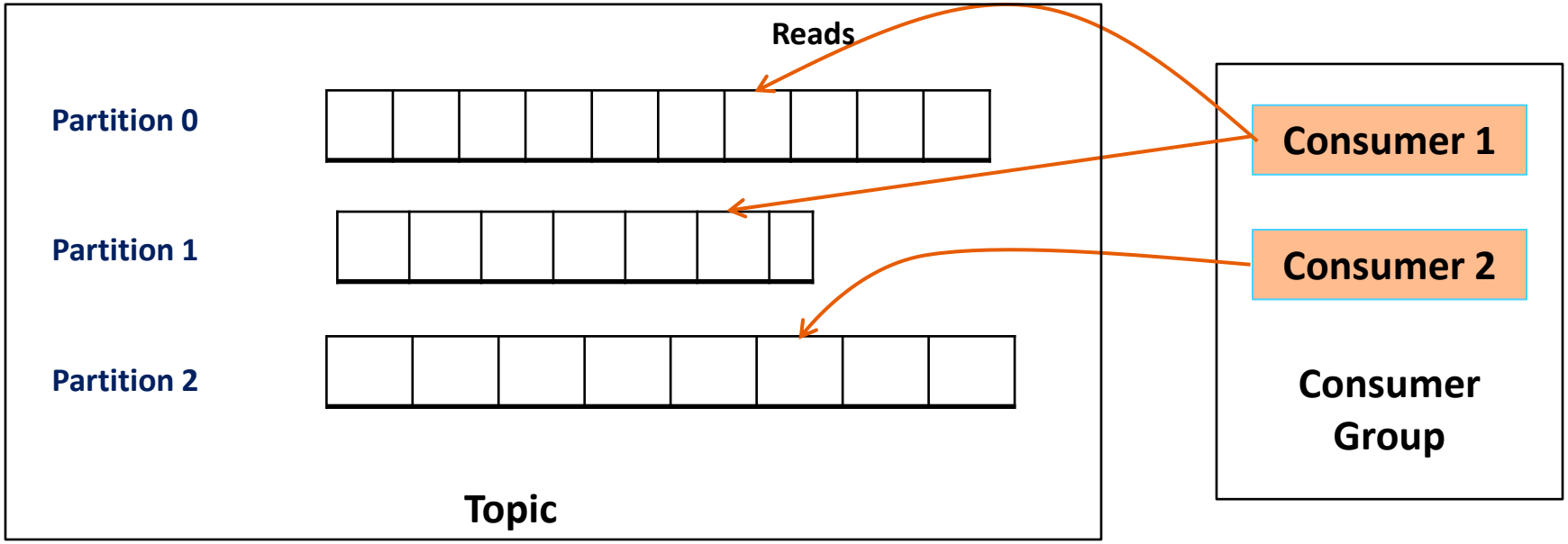
2f486967682070657266f726d616c63652c2044656c6976657265642c2f486967682070657266f726d616c63652c2044656c6976657265642c2f486967682070657266f726d616c63652c2044656c6976657265642c2f486967682070657266f726d616c63652c2044656c6976657265642c2f486967682070657266f726d616c63652c2044656c6976657265642c2f486967682070

- In Kafka, ***each topic is divided*** into a **set of logs** known as ***partitions***.
- **Producers** write to the tail of these **logs** and **Consumers** read the **logs** at their own pace.
- Kafka **scales topic consumption** by distributing partitions among a *consumer group*, which is a **set of consumers** sharing a common group identifier.

Message Handling in Kafka

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

- The diagram below shows a **single topic** with **three partitions** and **a consumer group** with **two members**.
- Each **partition** in the **topic** is ***assigned to exactly one member in the group***.



Implementing Simple Producer – Consumer example in Single Node Kafka Cluster

Implementing Simple Producer – Consumer example

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

Tasks to do

1. Start **Zookeeper service**

2. Start **Kafka Server**

3. Create one **Topic**

4. Start a **Producer** to publish messages to the Topic

5. Start a **Consumer** to consume messages from the Topic

Implementing Simple Producer – Consumer example

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

Step 1:- Start **Zookeeper service**

In the First TAB of your SmarTTY console

- **Change you current directory to KAFKA_HOME i.e. "/home/vagrant/bigdata/kafka" directory**

```
vagrant@master:~$ cd /home/vagrant/bigdata/kafka/

vagrant@master:~/bigdata/kafka$ pwd

    /home/vagrant/bigdata/kafka
```

Implementing Simple Producer – Consumer example

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

Step 1:- Start **Zookeeper service**

- **Issue the following command to start the Zookeeper server and observe the console output**

```
vagrant@master:~/bigdata/kafka$ bin/zookeeper-server-start.sh
config/zookeeper.properties
```

```
[2017-11-25 08:40:00,308] INFO Server environment:user.name=vagrant (org.apache.zookeeper.server.ZooKeeperSer
ver)
[2017-11-25 08:40:00,308] INFO Server environment:user.home=/home/vagrant (org.apache.zookeeper.server.ZooKee
perServer)
[2017-11-25 08:40:00,309] INFO Server environment:user.dir=/home/vagrant/bigdata/kafka (org.apache.zookeeper.
server.ZooKeeperServer)
[2017-11-25 08:40:00,385] INFO tickTime set to 3000 (org.apache.zookeeper.server.ZooKeeperServer)
[2017-11-25 08:40:00,385] INFO minSessionTimeout set to -1 (org.apache.zookeeper.server.ZooKeeperServer)
[2017-11-25 08:40:00,385] INFO maxSessionTimeout set to -1 (org.apache.zookeeper.server.ZooKeeperServer)
[2017-11-25 08:40:00,496] INFO binding to port 0.0.0.0/0.0.0.0:2181 (org.apache.zookeeper.server.NIOServerCnx
nFactory)
```

Implementing Simple Producer – Consumer example

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

Step 2:- Start **Kafka server**

- **Open Another TAB (Second TAB) in the SmarTTY console**
- **Change you current directory to KAFKA_HOME i.e. "/home/vagrant/bigdata/kafka" directory**

```
vagrant@master:~$ cd /home/vagrant/bigdata/kafka/

vagrant@master:~/bigdata/kafka$ pwd

    /home/vagrant/bigdata/kafka
```

Implementing Simple Producer – Consumer example

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

Step 2:- Start **Kafka Server**

- **Issue the following command to start the Kafka server and observe the console output**

```
vagrant@master:~/bigdata/kafka$ bin/kafka-server-start.sh
config/server.properties
```

```
[2017-11-25 08:48:19,416] INFO New leader is 0 (kafka.server.ZookeeperLeaderElector$LeaderChangeListener)
[2017-11-25 08:48:19,468] INFO Kafka version : 0.10.2.1 (org.apache.kafka.common.utils.AppInfoParser)
[2017-11-25 08:48:19,478] INFO Kafka commitId : e89bffd6b2eff799 (org.apache.kafka.common.utils.AppInfoParser
)
[2017-11-25 08:48:19,485] INFO [Kafka Server 0], started (kafka.server.KafkaServer)
```

Implementing Simple Producer – Consumer example

2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070

Step 3:- Create **TOPIC**

- **Open Another TAB (Third TAB) in the SmarTTY console**
- **Change you current directory to KAFKA_HOME i.e. "/home/vagrant/bigdata/kafka" directory**
- **Check the Kafka services that are running in the Node**

```
vagrant@master:~$ cd /home/vagrant/bigdata/kafka/

vagrant@master:~/bigdata/kafka$ jps
```

```
1586 QuorumPeerMain
1939 Kafka
2325 Jps
vagrant@master:~/bigdata/kafka$
```

Zookeeper service

Implementing Simple Producer – Consumer example

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

Step 3:- Create **TOPIC**

- Create a **topic** named "**gft-topic**" with a single partition and only one replica:

```
vagrant@master:~/bigdata/kafka$ bin/kafka-topics.sh --create --
zookeeper localhost:2181 --replication-factor 1 --partitions 1 --
topic gft-topic

Created topic "gft-topic".
```

Implementing Simple Producer – Consumer example

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

Step 3:- Create **TOPIC**

- **Display all the topics created in the cluster**

```
vagrant@master:~/bigdata/kafka$ bin/kafka-topics.sh --list
--zookeeper localhost:2181

gft-topic
```

Implementing Simple Producer – Consumer example

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

Step 4:- <u>Start a Producer - Send some messages</u>

- **Open Another TAB (Fourth TAB) in the SmarTTY console**
- **Change you current directory to KAFKA_HOME i.e. "/home/vagrant/bigdata/kafka" directory**
- **Kafka** comes with a command line client that will take **input** from a file or from **standard input** and send it out as **messages** to the Kafka cluster.
- By default, each line will be sent as a separate message.

Implementing Simple Producer – Consumer example

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

Step 4:- Start a Producer - Send some messages

- Run the **producer** using the following command and then type a few messages into the console to send to the server.

```
vagrant@master:~$ cd bigdata/kafka/

vagrant@master:~/bigdata/kafka$ bin/kafka-console-producer.sh --broker-list localhost:9092 --topic gft-topic
```

```
vagrant@master:~/bigdata/kafka$ bin/kafka-console-producer.sh --broker-list localhost:9092 --topic gft-topic
_
```

Cursor will wait for user input

type a few messages into the console to send to the server

Implementing Simple Producer – Consumer example

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

Step 4:- <u>Start a Producer - Send some messages</u>

- Type a few messages into the **Producer** console to send to the server.

This is first line

This is second Line

This is third line

```
vagrant@master:~/bigdata/kafka$ bin/kafka-console-producer.sh --broker-list localhost:9092 --topic gft-topic
This is first line
This is second Line
This is third line
```

Implementing Simple Producer – Consumer example

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

Step 5:- Start a Consumer to print the message

- **Kafka** also has a **command line consumer** that will dump out messages to standard output.
- **Open Another TAB (Fifth TAB) in the SmarTTY console**
- **Change you current directory to KAFKA_HOME i.e. "/home/vagrant/bigdata/kafka" directory**

Implementing Simple Producer – Consumer example

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

Step 5:- Start a Consumer to print the message

- **Issue the following command to Start the Kafka Consumer which will print the messages from the topic "gft-topic"** to standard output.

```
vagrant@master:~/bigdata/kafka$ bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic gft-topic --from-beginning
```

This is first line
This is second Line
This is third line

Implementing Simple Producer – Consumer example

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

Step 5:- Start a Consumer to print the message

- **Issue the following command to Start the Kafka Consumer which will print the messages from the topic "gft-topic"** to standard output.

```
vagrant@master:~/bigdata/kafka$ bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic gft-t
opic --from-beginning
This is first line
This is second Line
This is third line
```

Implementing Simple Producer – Consumer example

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

Working with Producer and Consumer parallelly

- In the Producer TAB(Fourth Tab), type another message , press <enter>

```
vagrant@master:~/bigdata/kafka$ bin/kafka-console-producer.sh --broker-list localhost:9092 --topic gft-topic
This is first line
This is second Line
This is third line
This is Fourth Line
```

- **In the Consumer TAB(Fifth Tab), check the message will be displayed**

```
vagrant@master:~/bigdata/kafka$ bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic gft-t
opic --from-beginning
This is first line
This is second Line
This is third line
This is Fourth Line
```

Check the services running in the Kafka Cluster

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

```
vagrant@master:~$ jps
```

```
3856 Jps
1377 QuorumPeerMain
3269 ConsoleProducer
3514 ConsoleConsumer
1727 Kafka
```

Producer Service

Consumer Service

Getting Detailed Description of Topic

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

- Go to the Third Tab of the Console where we have created the Topic "gft-topic"

vagrant@master:~/bigdata/kafka$ bin/kafka-topics.sh --describe --zookeeper localhost:2181 --topic gft-topic

Topic:gft-topic PartitionCount:1 ReplicationFactor:1 Configs:
Topic: gft-topic Partition: 0 Leader: 0 Replicas: 0 Isr: 0

```
Topic:gft-topic PartitionCount:1        ReplicationFactor:1     Configs:
        Topic: gft-topic        Partition: 0    Leader: 0       Replicas: 0     Isr: 0
```

Getting Detailed Description of Topic

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

Topic:gft-topic PartitionCount:1 ReplicationFactor:1 Configs:
Topic: gft-topic Partition: 0 Leader: 0 Replicas: 0 Isr: 0

- The **first line** gives a **summary** of all the partitions,
- **Each additional line** gives information about **one partition**.
  - Since we have **only one partition for this topic there is only one line**.
- "**leader**" is the node responsible for all reads and writes for the given partition.
  - **Each node will be the leader** for a randomly selected portion of the partitions.
- "**replicas**" is the **list of nodes** that replicate the log for this partition regardless of whether they are the leader or even if they are currently alive.
- "**isr**" is the set of "**in-sync**" replicas. This is the subset of the replicas list that is currently alive and caught-up to the leader.

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

Deleting a Topic in Kafka Cluster

Deleting a Topic

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

- Set the properties

  delete.topic.enable = true

  in **config/server.properties** of Kafka brokers

Note :- we have add this line in **config/server.properties file**

vagrant@master:~/bigdata/kafka$ bin/kafka-topics.sh --delete --zookeeper localhost:2181 --topic gft-topic

Topic gft-topic is marked for deletion.
Note: This will have no impact if delete.topic.enable is not set to true.

```
Topic gft-topic is marked for deletion.
Note: This will have no impact if delete.topic.enable is not set to true.
```

Deleting a Topic

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

- List all the Topics in the Kafka Cluster using the following command

vagrant@master:~/bigdata/kafka$ bin/kafka-topics.sh --list --zookeeper localhost:2181

- **You will be still observing the "gft-topic" listed in the output**
  - Because the properties

    `delete.topic.enable = true`

    `is NOT set` in **config/server.properties** of Kafka brokers

Stop the Services in Kafka Cluster

2f486967682070657266 6f726d616c63652c2044656c6976657265642c2f486967682070657266 6f726d616c63652c2044656c6976657265642c2f486967682070657266 6f726d616c63652c2044656c6976657265642c2f486967682070657266 6f726d616c63652c2044656c6976657265642c2f486967682070657266 6f726d616c63652c2044656c6976657265642c2f486967682070

Stop Consumer

- In the 5th tab of the console , press **CTRL+C** to stop **Consumer**

```
This is first line
this is second line
this is third line
^CProcessed a total of 3 messages
vagrant@master:~/bigdata/kafka$ _
```

Stop Producer

- In the 4th tab of the console , press **CTRL+C** to stop **Producer**

```
This is first line
this is second line
this is third line
vagrant@master:~/bigdata/kafka$
```

Stop the Services in Kafka Cluster

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

Stop Kafka Server

- In the 2nd tab of the console , press **CTRL+C** to stop **Kafka Server**

```
[2017-11-26 02:54:26,265] INFO [Kafka Server 0], shut down completed (kafka.server.KafkaServer)
```

Do NOT stop ZooKeeper Service

Check the Kafka Services

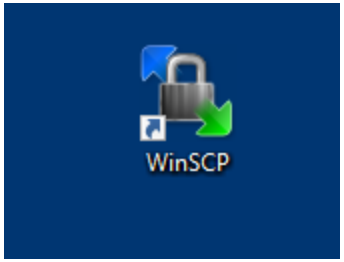
```
1377 QuorumPeerMain
5196 Jps
```

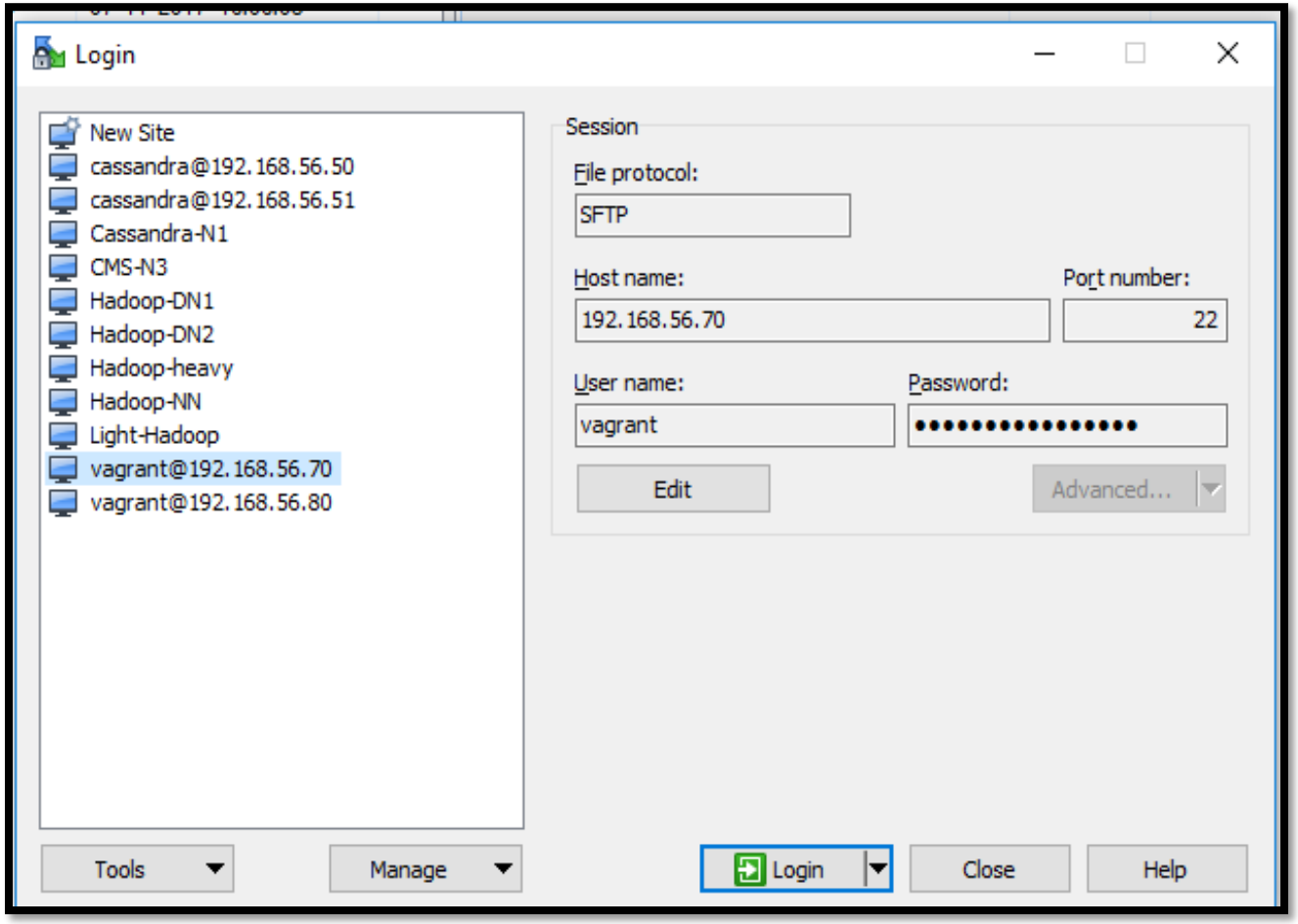
Only ZooKeeper service is running

Modify server.properties file of Kafka Server

2f486967682070657266f726d616c63652c2044656c6976657265642c2f486967682070657266f726d616c63652c2044656c6976657265642c2f486967682070657266f726d616c63652c2044656c6976657265642c2f486967682070657266f726d616c63652c2044656c6976657265642c2f486967682070657266f726d616c63652c2044656c6976657265642c2f486967682070

Connect WinSCP with the Linux Node

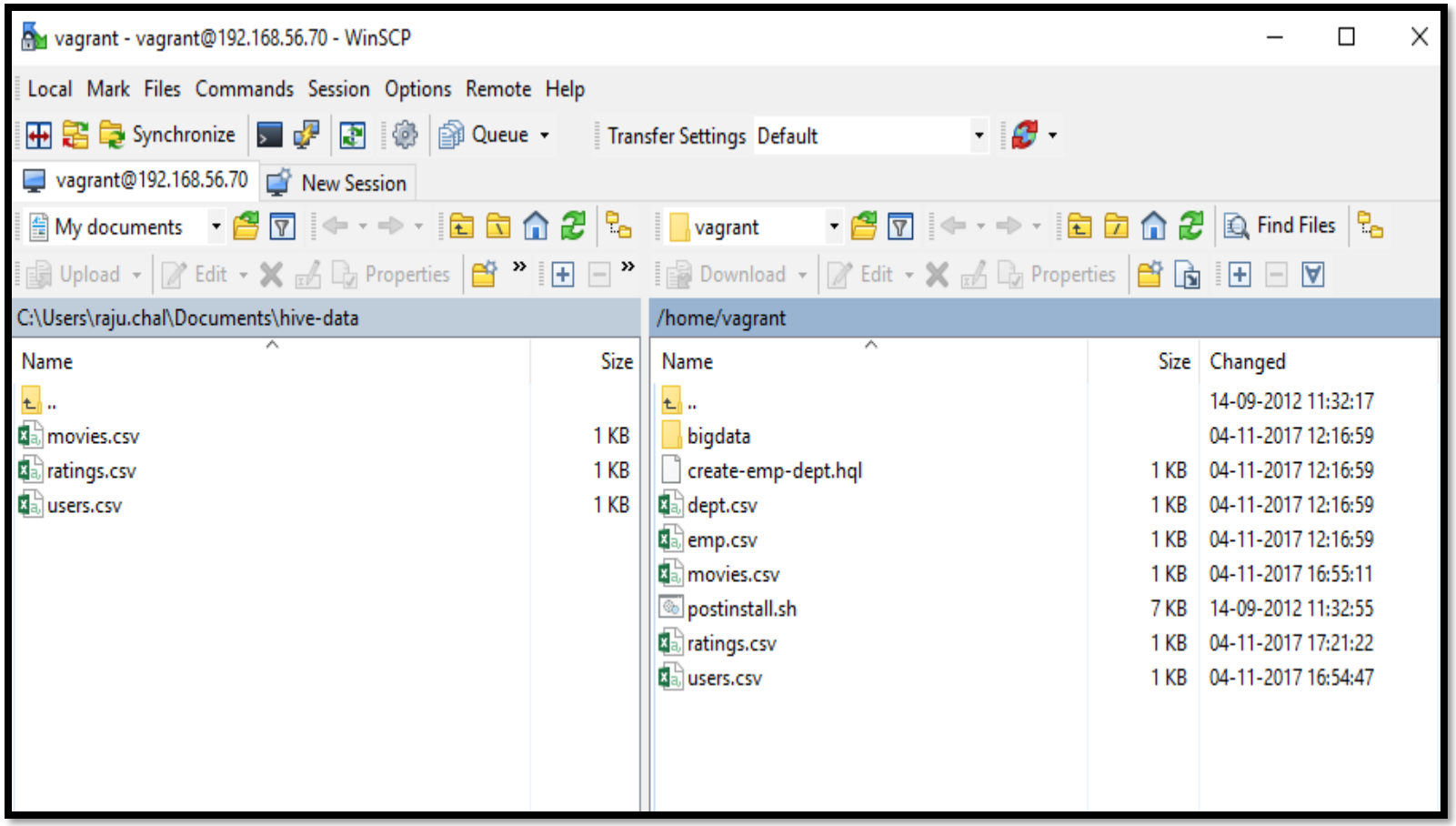




Modify server.properties file

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

WinSCP explorer after connection with Linux Node

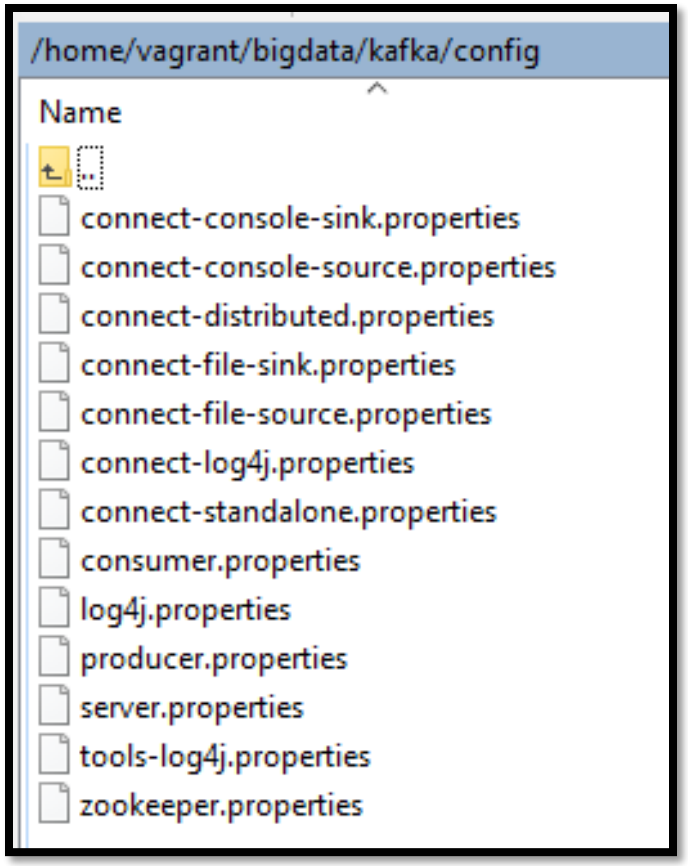


accenture

Modify server.properties file

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

- **In** the "**Right Pane**" of **WinSCP** , Open **config** directory in the following path , Open **bigdata →kafka→config**



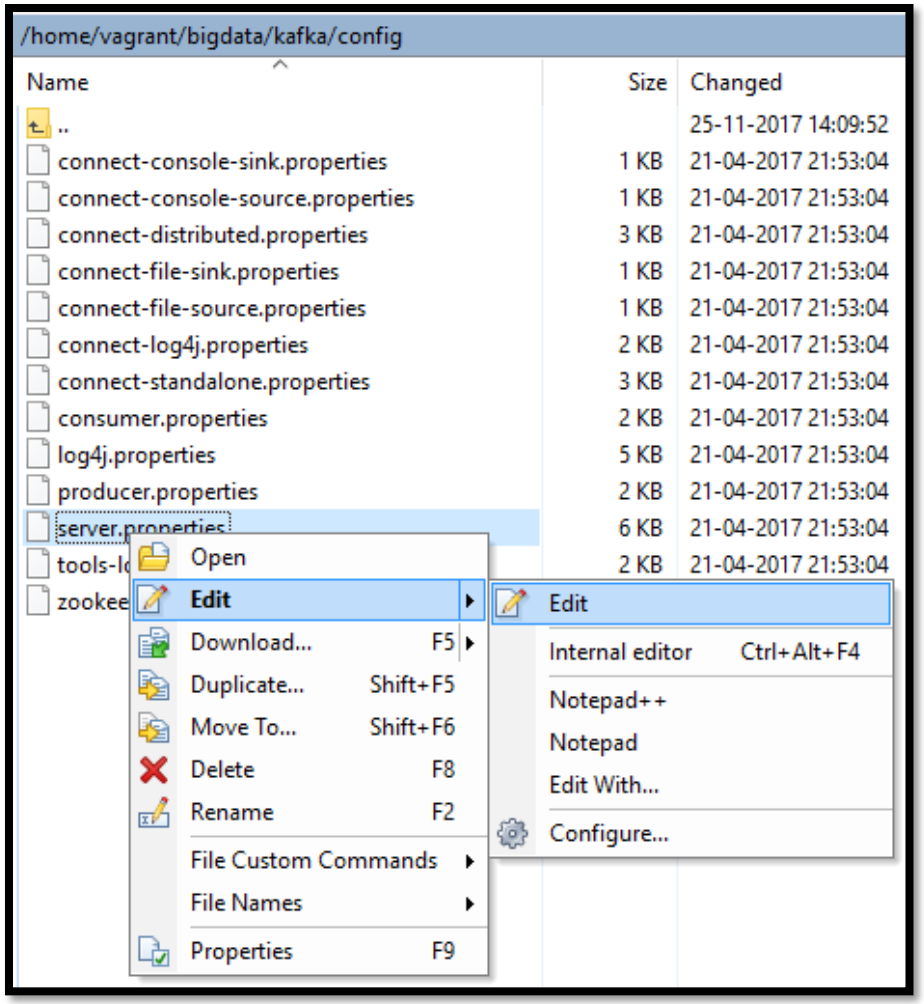
Modify server.properties file

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682

- **In** the "**Right Pane**" of **WinSCP** ,

  To open the **server.properties** file,

  Right click on it →Edit→Edit



accenture

Modify server.properties file

- Find the following properties in the file

```
#delete.topic.enable=true
```

```
# Switch to enable topic deletion or not, default value is false
#delete.topic.enable=true
```

- Remove the # (comment entry) from the beginning of the Line
- Save the file
- Close the file

```
# Switch to enable topic deletion or not, default value is false
delete.topic.enable=true
```

Start the Kafka Server again

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

- Go to the Second TAB of the Console
- Issue the following command to start the Kafka server and observe the console output

```
vagrant@master:~/bigdata/kafka$ bin/kafka-server-start.sh
config/server.properties
```

```
[2017-11-25 08:48:19,416] INFO New leader is 0 (kafka.server.ZookeeperLeaderElector$LeaderChangeListener)
[2017-11-25 08:48:19,468] INFO Kafka version : 0.10.2.1 (org.apache.kafka.common.utils.AppInfoParser)
[2017-11-25 08:48:19,478] INFO Kafka commitId : e89bffd6b2eff799 (org.apache.kafka.common.utils.AppInfoParser
)
[2017-11-25 08:48:19,485] INFO [Kafka Server 0], started (kafka.server.KafkaServer)
```

Now again Try to Delete the Topic “gft-topic”

2f486967682070657266 6f726d616c63652c2044656c6976657265642c2f486967682070657266 6f726d616c63652c2044656c6976657265642c2f486967682070657266 6f726d616c63652c2044656c6976657265642c2f486967682070657266 6f726d616c63652c2044656c6976657265642c2f486967682070657266 6f726d616c63652c2044656c6976657265642c2f486967682070657266 6f726d616c63652c2044656c6976657265642c2f486967682070

- Go to the **Third TAB** of the console
- Issue the following command to delete the topic “gft-topic”

vagrant@master:~/bigdata/kafka$ bin/kafka-topics.sh --delete --zookeeper localhost:2181 --topic gft-topic

```
Topic gft-topic is marked for deletion.
Note: This will have no impact if delete.topic.enable is not set to true.
```

- Now List all the Topics in the Kafka Cluster using the following command

vagrant@master:~/bigdata/kafka$ bin/kafka-topics.sh --list --zookeeper localhost:2181

“gft-topic” will not be listed in the output

Stop the Kafka Server

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

Stop Kafka Server

- In the 2nd tab of the console , press **CTRL+C** to stop **Kafka Server**

```
[2017-11-26 02:54:26,265] INFO [Kafka Server 0], shut down completed (kafka.server.KafkaServer)
```

Do NOT stop ZooKeeper Service

Check the Kafka Services

```
1377 QuorumPeerMain
5196 Jps
```

Only ZooKeeper service is running

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

END PART -1

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

PART -2

Understanding Partition within a Topic

Partition within a Topic

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

- Let us understand the command used for creating a **topic** named "**gft-topic**" **in Kafka Cluster**

```
vagrant@master:~/bigdata/kafka$ bin/kafka-topics.sh --
create --zookeeper localhost:2181 --replication-factor 1
--partitions 1 --topic gft-topic
```

Number of Partitions Within the topic

Name of the topic

Number of Replications For each Partition Within the topic

What is Partition within a Topic ?

2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070

- **Within a Topic, Messages can be divided into Partitions**

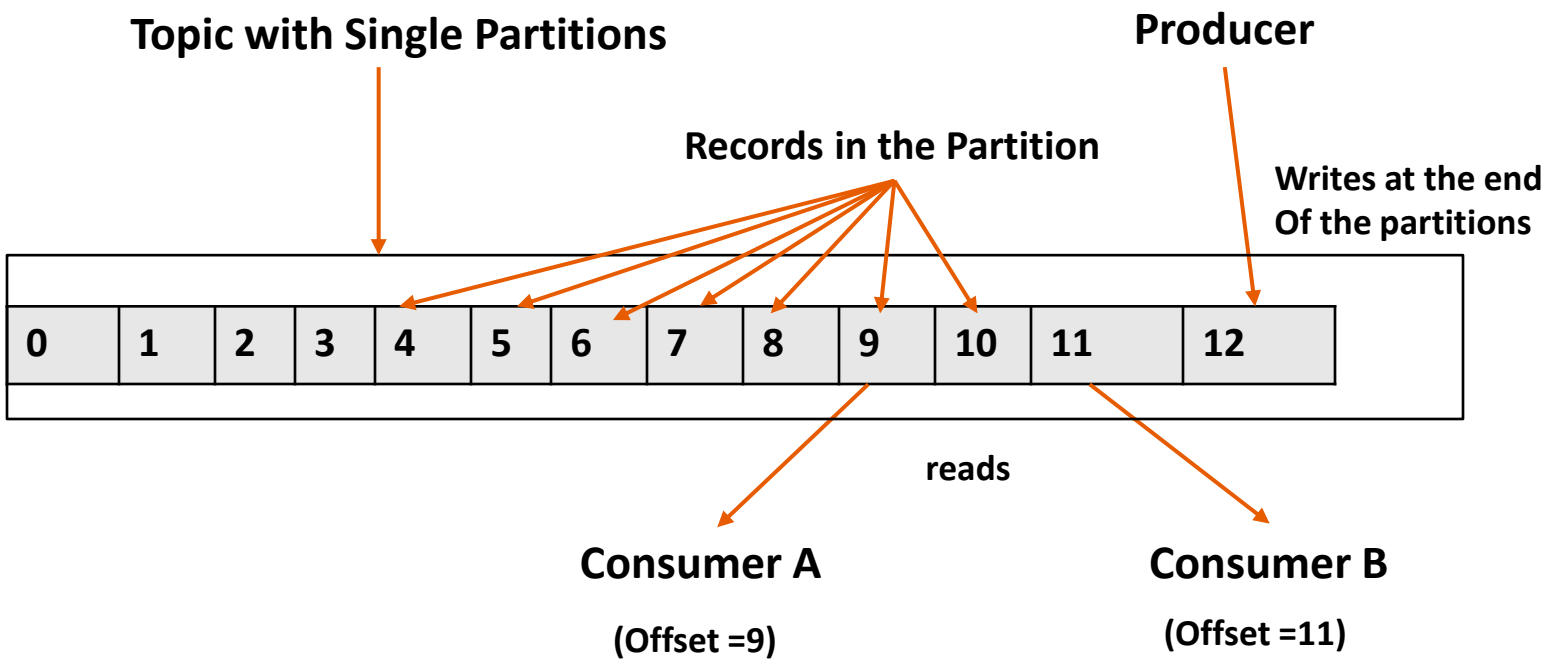
- **Each partition** is an ordered, immutable sequence of records that is continually appended ***to—a structured commit log***.

- The **records** in the partitions are ***each assigned a sequential id number called the offset*** that uniquely identifies each record within the partition.

- This **offset** is controlled by the **consumer**: *normally a **consumer** will advance its offset linearly as it **reads records***, but, *it can consume records in any order it likes*

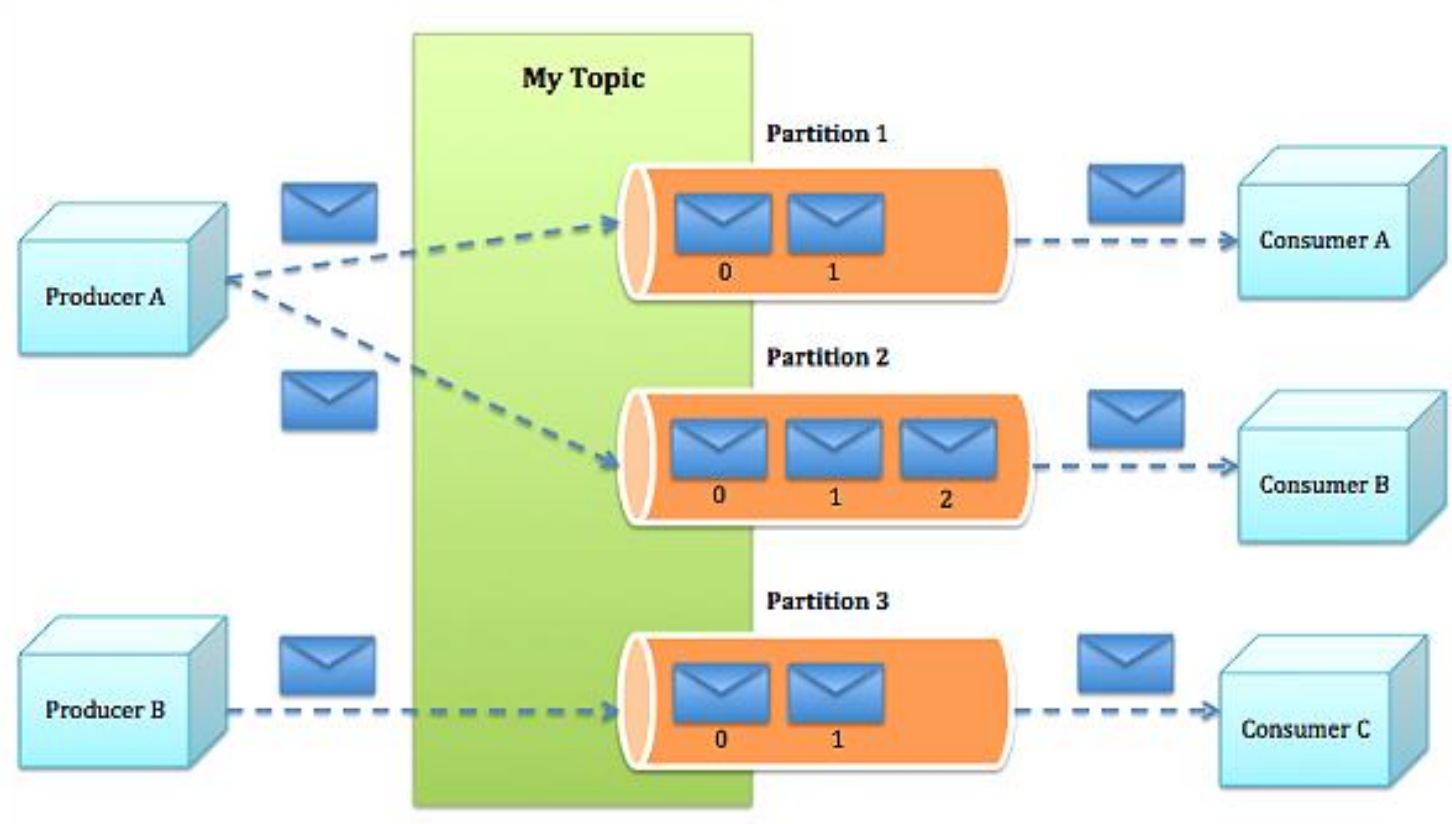
Partition within a Topic

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070



Message Processing in Kafka

2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070



Partition within a Topic

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

- **The partitions in the log serve several purposes**.
  - **First**, Messages are divided into Partitions so that **one Partition** can fit on a single server.
    - **A topic may have many partitions so it can handle an arbitrary amount of data.**
  - **Second** they act as the unit of parallelism, i.e. each partition can be read and processed parallelly by different Kafka Server.
- **Producers** publish data to the **topics** of their choice.
  - The **producer** is responsible for choosing which record to assign to which partition within the topic.

Replication for Partition Fault Tolerance

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

- During the creation of the Topic , we can define the **number of replications** that each partitions of the Topic should have.

- Provides the **Fault Tolerance** capability
  - If suddenly one Kafka server is down , the same partition will be available from another Kafka Server to be read or processed.

Partition Fault Tolerance.

2f486967682070657266f726d616c63652c2044656c6976657265642c2f486967682070657266f726d616c63652c2044656c6976657265642c2f486967682070657266f726d616c63652c2044656c6976657265642c2f486967682070657266f726d616c63652c2044656c6976657265642c2f486967682070657266f726d616c63652c2044656c6976657265642c2f486967682070

- **Each partition** has
  - one server which acts as the "**leader**" and
  - **zero** or **more servers** which act as "**followers**".
- The **leader** handles all read and write requests for the partition
- The **followers** passively replicate the leader.
- **If the leader fails, one of the followers will automatically become the new leader.**
- ***Each server acts as a leader for some of its partitions and a follower for others so load is well balanced within the cluster.***

Configuring Multiple Kafka Broker in Single Node

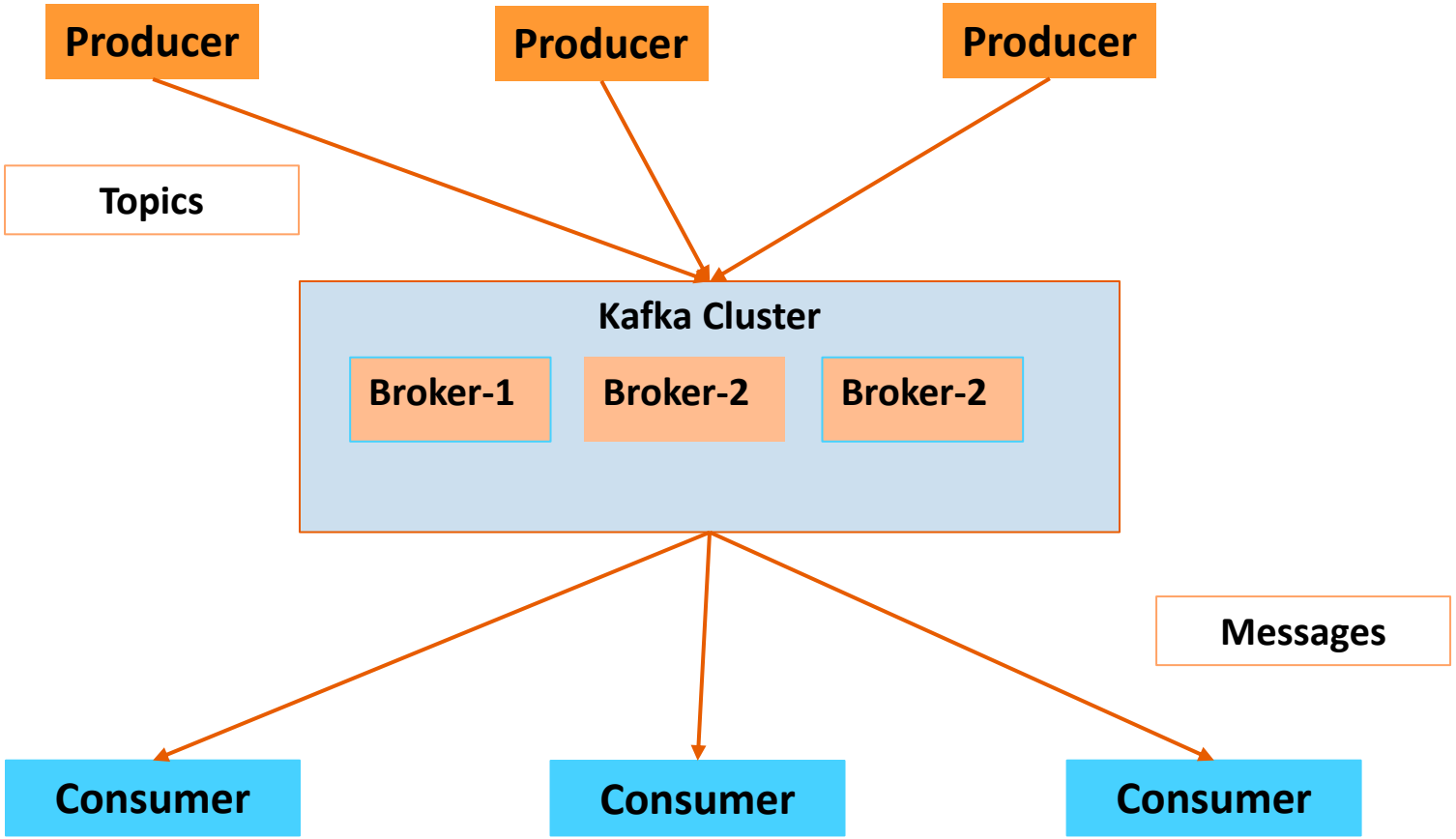
Single Node – Multiple Kafka Broker Cluster

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

- In the following example we will configure **Multiple Kafka Broker** in our Linux Node
  - Will work as **Multi Broker Kafka Cluster**
  - Having **Single Zookeeper Server**

Architecture of Kafka Cluster

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070



Configure Multiple Kafka Broker

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

- Go to "/home/vagrant/bigdata/kafka/config" directory
- **Create three copies of server.properties file as follows**

```
vagrant@master:~$ cd /home/vagrant/bigdata/kafka/config/

vagrant@master:~/bigdata/kafka/config$ cp server.properties server1.properties

vagrant@master:~/bigdata/kafka/config$ cp server.properties server2.properties

vagrant@master:~/bigdata/kafka/config$ cp server.properties server3.properties

vagrant@master:~/bigdata/kafka/config$ ls
```

```
server1.properties  server2.properties  server3.properties  server.properties
```

Modify server1.properties file

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

Using **WinSCP** modify server1.properties file with the following contents

```
broker.id=1

listeners=PLAINTEXT://:9093

log.dirs=/tmp/kafka-logs1

zookeeper.connect=localhost:2181
```

broker.id changed

Comment removed and Port no changed

Log file location changed

Save and Close the file

Modify server1.properties file

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

Using **WinSCP** modify server1.properties file with the following contents

```
# The id of the broker. This must be set to a unique integer for each broker.
broker.id=1

# Switch to enable topic deletion or not, default value is false
delete.topic.enable=true
```

```
# The address the socket server listens on. It will get the value returned from
# java.net.InetAddress.getCanonicalHostName() if not configured.
#   FORMAT:
#     listeners = listener_name://host_name:port
#   EXAMPLE:
#     listeners = PLAINTEXT://your.host.name:9092
listeners=PLAINTEXT://:9093
```

```
############################# Log Basics #############################

# A comma seperated list of directories under which to store log files
log.dirs=/tmp/kafka-logs1
```

Modify server2.properties file

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

Using **WinSCP** modify server2.properties file with the following contents

```
broker.id=2

listeners=PLAINTEXT://:9094

log.dirs=/tmp/kafka-logs2

zookeeper.connect=localhost:2181
```

broker.id changed

Comment removed and Port no changed

Log file location changed

Save and Close the file

Modify server2.properties file

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

Using **WinSCP** modify server2.properties file with the following contents

```
############################# Server Basics #############################

# The id of the broker. This must be set to a unique integer for each broker.
broker.id=2

# Switch to enable topic deletion or not, default value is false
delete.topic.enable=true
```

```
############################# Socket Server Settings #############################

# The address the socket server listens on. It will get the value returned from
# java.net.InetAddress.getCanonicalHostName() if not configured.
#   FORMAT:
#     listeners = listener_name://host_name:port
#   EXAMPLE:
#     listeners = PLAINTEXT://your.host.name:9092
listeners=PLAINTEXT://:9094
```

```
############################# Log Basics #############################

# A comma seperated list of directories under which to store log files
log.dirs=/tmp/kafka-logs2
```

Modify server3.properties file

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

Using **WinSCP** modify server3.properties file with the following contents

```
broker.id=3

listeners=PLAINTEXT://:9095

log.dirs=/tmp/kafka-logs3

zookeeper.connect=localhost:2181
```

broker.id changed

Comment removed and Port no changed

Log file location changed

Save and Close the file

Modify server3.properties file

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

Using **WinSCP** modify server3.properties file with the following contents

```
############################# Server Basics #############################

# The id of the broker. This must be set to a unique integer for each broker.
broker.id=3

# Switch to enable topic deletion or not, default value is false
delete.topic.enable=true
```

```
############################# Socket Server Settings #############################

# The address the socket server listens on. It will get the value returned from
# java.net.InetAddress.getCanonicalHostName() if not configured.
#   FORMAT:
#     listeners = listener_name://host_name:port
#   EXAMPLE:
#     listeners = PLAINTEXT://your.host.name:9092
listeners=PLAINTEXT://:9095
```

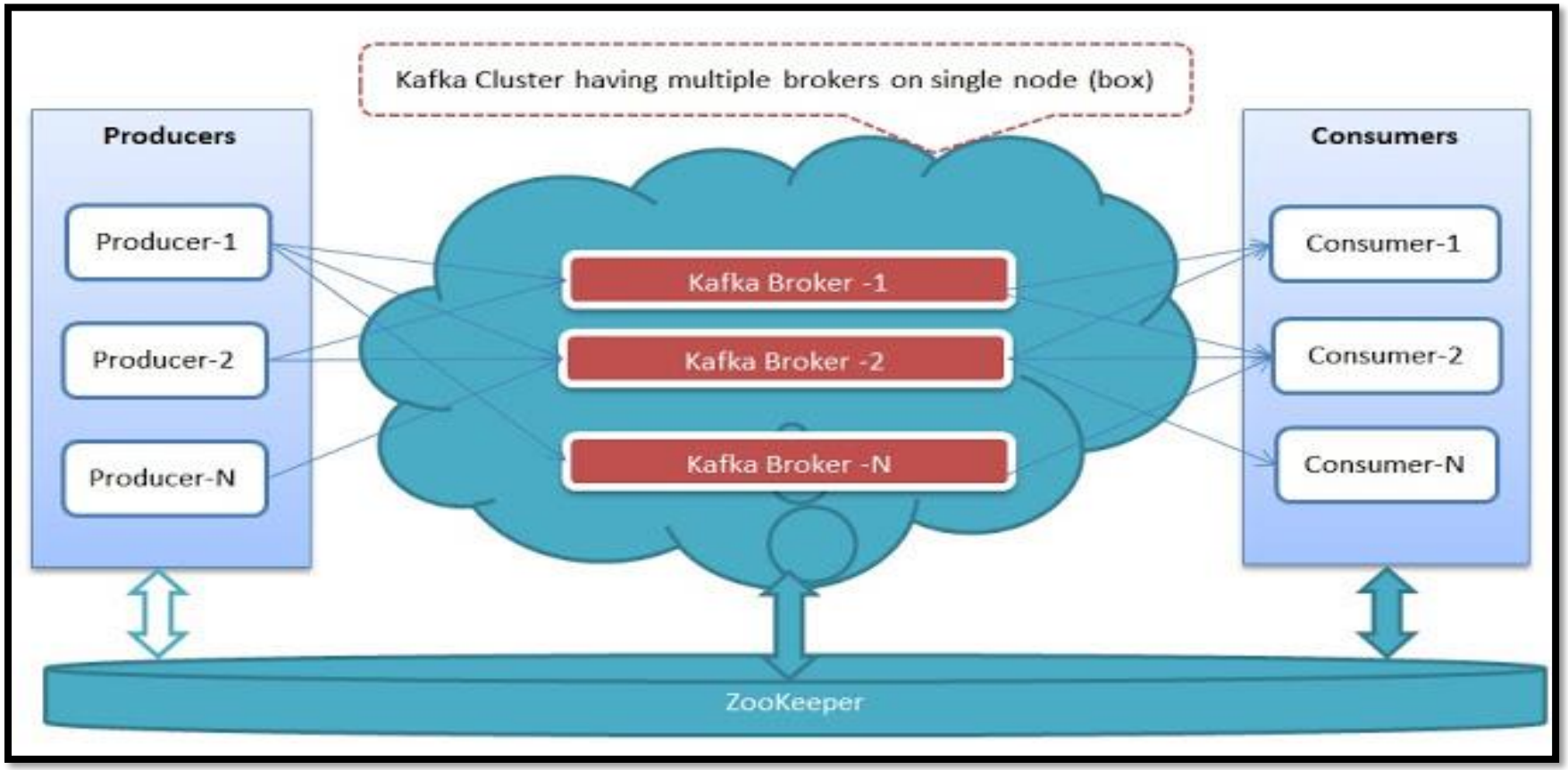
```
############################# Log Basics #############################

# A comma seperated list of directories under which to store log files
log.dirs=/tmp/kafka-logs3
```

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

Implementing Simple Producer – Consumer example-2 in Single Node – Multiple Kafka Broker Cluster

Kafka Cluster- Single Node, Multi Broker

2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070



Implementing Simple Producer – Consumer example-2

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

Tasks to do

1.Start the **Zookeeper service**

2.Start All the **Kafka Servers**

3.Create one **Topic with Three Replication**

4.Start a **Producer** to publish messages to the Topic

5.Start a **Consumer** to consume messages from the Topic

Implementing Simple Producer – Consumer example-2

2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070

Step 1:- Start **Zookeeper service**

- Check in the **First TAB** of your **SmarTTY** console, **Zookeeper** service is already running .
- If the **ZooKeeper** server is stopped , start the **ZooKeeper** server using the following command

- **Change you current directory to KAFKA_HOME i.e. "/home/vagrant/bigdata/kafka" directory**

```
vagrant@master:~$ cd /home/vagrant/bigdata/kafka/

vagrant@master:~/bigdata/kafka$ pwd

    /home/vagrant/bigdata/kafka
```

Implementing Simple Producer – Consumer example-2

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

Step 1:- Start **Zookeeper service**

- **Issue the following command to start the Zookeeper server and observe the console output**

```
vagrant@master:~/bigdata/kafka$ bin/zookeeper-server-start.sh
config/zookeeper.properties
```

```
[2017-11-25 08:40:00,308] INFO Server environment:user.name=vagrant (org.apache.zookeeper.server.ZooKeeperSer
ver)
[2017-11-25 08:40:00,308] INFO Server environment:user.home=/home/vagrant (org.apache.zookeeper.server.ZooKee
perServer)
[2017-11-25 08:40:00,309] INFO Server environment:user.dir=/home/vagrant/bigdata/kafka (org.apache.zookeeper.
server.ZooKeeperServer)
[2017-11-25 08:40:00,385] INFO tickTime set to 3000 (org.apache.zookeeper.server.ZooKeeperServer)
[2017-11-25 08:40:00,385] INFO minSessionTimeout set to -1 (org.apache.zookeeper.server.ZooKeeperServer)
[2017-11-25 08:40:00,385] INFO maxSessionTimeout set to -1 (org.apache.zookeeper.server.ZooKeeperServer)
[2017-11-25 08:40:00,496] INFO binding to port 0.0.0.0/0.0.0.0:2181 (org.apache.zookeeper.server.NIOServerCnx
nFactory)
```

Implementing Simple Producer – Consumer example-2

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

Step 2:- Start the **Kafka server-1**

- **Open Another TAB (Second TAB) in the SmarTTY console**
- **Change you current directory to KAFKA_HOME i.e. "/home/vagrant/bigdata/kafka" directory**

```
vagrant@master:~$ cd /home/vagrant/bigdata/kafka/

vagrant@master:~/bigdata/kafka$ pwd

    /home/vagrant/bigdata/kafka
```

Implementing Simple Producer–Consumer example-2

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

Step 2:- Start **Kafka Server-1**

- **Issue the following command to start the Kafka server and observe the console output**

```
vagrant@master:~/bigdata/kafka$ bin/kafka-server-start.sh
config/server1.properties
```

```
)
[2017-11-26 10:50:56,026] INFO [Kafka Server 1], started (kafka.server.KafkaServer)
```

Implementing Simple Producer – Consumer example-2

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

Step 2:- Start the **Kafka server-2**

- **Open Another TAB (Third TAB) in the SmarTTY console**
- **Change you current directory to KAFKA_HOME i.e. "/home/vagrant/bigdata/kafka" directory**

```
vagrant@master:~$ cd /home/vagrant/bigdata/kafka/

vagrant@master:~/bigdata/kafka$ pwd

    /home/vagrant/bigdata/kafka
```

Implementing Simple Producer – Consumer example-2

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

Step 2:- Start **Kafka Server-2**

- **Issue the following command to start the Kafka server and observe the console output**

```
vagrant@master:~/bigdata/kafka$ bin/kafka-server-start.sh
config/server2.properties
```

```
)
[2017-11-26 10:53:51,046] INFO [Kafka Server 2], started (kafka.server.KafkaServer)
```

Implementing Simple Producer – Consumer example-2

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

Step 2:- Start the **Kafka server-3**

- **Open Another TAB (Fourth TAB) in the SmarTTY console**
- **Change you current directory to KAFKA_HOME i.e. "/home/vagrant/bigdata/kafka" directory**

```
vagrant@master:~$ cd /home/vagrant/bigdata/kafka/

vagrant@master:~/bigdata/kafka$ pwd

    /home/vagrant/bigdata/kafka
```

Implementing Simple Producer – Consumer example-2

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

Step 2:- Start **Kafka Server-3**

- **Issue the following command to start the Kafka server and observe the console output**

```
vagrant@master:~/bigdata/kafka$ bin/kafka-server-start.sh
config/server3.properties
```

```
)
[2017-11-26 10:56:01,577] INFO [Kafka Server 3], started (kafka.server.KafkaServer)
```

Implementing Simple Producer – Consumer example-2

2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070

Step 3:- Create **TOPIC**

- **Open Another TAB (Fifth TAB) in the SmarTTY console**
- **Change you current directory to KAFKA_HOME i.e. "/home/vagrant/bigdata/kafka" directory**
- **Check the Kafka services that are running in the Node**

```
vagrant@master:~$ cd /home/vagrant/bigdata/kafka/

vagrant@master:~/bigdata/kafka$ jps
```

```
6144 Kafka
6977 Jps
1377 QuorumPeerMain
6423 Kafka
6700 Kafka
```

Zookeeper service

Kafka Servers

Implementing Simple Producer – Consumer example-2

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

Step 3:- Create **TOPIC**

- Create a **topic** named "**gft-topic**" with a single partition and only one replica:

```
vagrant@master:~/bigdata/kafka$ bin/kafka-topics.sh --
create --zookeeper localhost:2181 --replication-factor 3 --
partitions 1 --topic gft-replicated-topic


Created topic "gft-replicated-topic".
```

Implementing Simple Producer – Consumer example-2

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

Step 3:- Create **TOPIC**

- **Display all the topics created in the cluster**

```
vagrant@master:~/bigdata/kafka$ bin/kafka-topics.sh --list
--zookeeper localhost:2181
```

```
__consumer_offsets
gft-replicated-topic
```

Implementing Simple Producer – Consumer example-2

2f486967682070657266f726d616c63652c2044656c6976657265642c2f486967682070657266f726d616c63652c2044656c6976657265642c2f486967682070657266f726d616c63652c2044656c6976657265642c2f486967682070657266f726d616c63652c2044656c6976657265642c2f486967682070657266f726d616c63652c2044656c6976657265642c2f486967682070

Get the Description of the **TOPIC "gft-replicated-topic"**

```
vagrant@master:~/bigdata/kafka$ bin/kafka-topics.sh --describe
--zookeeper localhost:2181 --topic gft-replicated-topic
```

Topic:gft-replicated-topic PartitionCount:1 ReplicationFactor:3 Configs:
Topic: gft-replicated-topic Partition: 0 Leader: 3 Replicas: 3,1,2 Isr: 3,1,2

```
Topic:gft-replicated-topic      PartitionCount:1        ReplicationFactor:3     Configs:
        Topic: gft-replicated-topic     Partition: 0    Leader: 3       Replicas: 3,1,2 Isr: 3,1,2
```

Only one Partition for this Topic

Three Replications for each Partition

Kafka Server 3 is the Leader for this Partition 0

Getting Detailed Description of Topic

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

Topic:gft-replicated-topic PartitionCount:1 ReplicationFactor:3 Configs:
Topic: gft-replicated-topic Partition: 0 Leader: 3 Replicas: 3,1,2 Isr: 3,1,2

- The **first line** gives a **summary** of all the partitions,
- **Each additional line** gives information about **one partition**.
  - Since we have **only one partition for this topic there is only one line**.
- "**leader**" is the node responsible for all reads and writes for the given partition.
  - **Each node will be the leader** for a randomly selected portion of the partitions.
- "**replicas**" is the **list of nodes** that replicate the log for this partition regardless of whether they are the leader or even if they are currently alive.
- "**isr**" is the set of "**in-sync**" replicas. This is the subset of the replicas list that is currently alive and caught-up to the leader.

Implementing Simple Producer – Consumer example

2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070

Step 4:- <u>Start a Producer - Send some messages</u>

- **Open Another TAB (Sixth TAB) in the SmarTTY console**
- **Change you current directory to KAFKA_HOME i.e. "/home/vagrant/bigdata/kafka" directory**
- **Kafka** comes with a command line client that will take **input** from a file or from **standard input** and send it out as **messages** to the Kafka cluster.
- By default, each line will be sent as a separate message.

Implementing Simple Producer – Consumer example

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

Step 4:- Start a Producer - Send some messages

- Run the **producer** using the following command and then type a few messages into the console to send to the server.

```
vagrant@master:~$ cd /home/vagrant/bigdata/kafka/

vagrant@master:~/bigdata/kafka$ bin/kafka-console-producer.sh --broker-list localhost:9093,localhost:9094,localhost:9095 --sync  --topic gft-replicated-topic
```

Implementing Simple Producer – Consumer example

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

Step 4:- <u>Start a Producer - Send some messages</u>

- Type a few messages into the **Producer** console to send to the server.

This is first line

This is second Line

This is third line

```
vagrant@master:~/bigdata/kafka$ bin/kafka-console-producer.sh --broker-list localhost:9093,localhost:9094,loc
alhost:9095 --sync  --topic gft-replicated-topic
This is first line
this is second line
this is third line
```

Implementing Simple Producer – Consumer example

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

Step 5:- Start a Consumer to print the message

- **Kafka** also has a **command line consumer** that will dump out messages to standard output.
- **Open Another TAB (Seventh TAB) in the SmarTTY console**
- **Change you current directory to KAFKA_HOME i.e. "/home/vagrant/bigdata/kafka" directory**

Implementing Simple Producer – Consumer example

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

Step 5:- Start a Consumer to consume the messages

- **Issue the following command to Start the Kafka Consumer which will print the messages from the topic "gft-replicated-topic"** to standard output.

```
vagrant@master:~$ cd /home/vagrant/bigdata/kafka/

vagrant@master:~/bigdata/kafka$ bin/kafka-console-consumer.sh --zookeeper localhost:2181 --topic gft-replicated-topic --from-beginning
```

```
Using the ConsoleConsumer with old consumer is deprecated and will be removed in a future major release. Cons
ider using the new consumer by passing [bootstrap-server] instead of [zookeeper].
This is first line
this is second line
this is third line
```

Implementing Simple Producer – Consumer example

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

Step 5:- Start a Consumer to print the message

- **Issue the following command to Start the Kafka Consumer which will print the messages from the topic "gft-topic"** to standard output.

```
vagrant@master:~/bigdata/kafka$ bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic gft-t
opic --from-beginning
This is first line
This is second Line
This is third line
```

Implementing Simple Producer – Consumer example

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

Working with Producer and Consumer parallelly

- In the Producer TAB(Sixth Tab), type another message , press <enter>

```
vagrant@master:~/bigdata/kafka$ bin/kafka-console-producer.sh --broker-list localhost:9093,localhost:9094,loc
alhost:9095 --sync  --topic gft-replicated-topic
This is first line
this is second line
this is third line
this is Fourth Line
```

- **In the Consumer TAB(Seventh Tab), check the message will be displayed**

```
vagrant@master:~/bigdata/kafka$ bin/kafka-console-consumer.sh --zookeeper localhost:2181 --topic gft-replicat
ed-topic --from-beginning
Using the ConsoleConsumer with old consumer is deprecated and will be removed in a future major release. Cons
ider using the new consumer by passing [bootstrap-server] instead of [zookeeper].
This is first line
this is second line
this is third line
this is Fourth Line
```

Check the services running in the Kafka Cluster

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

Open Another TAB

```
vagrant@master:~$ jps
```

```
6144 Kafka
7729 ConsoleProducer
1377 QuorumPeerMain
10147 Jps
6423 Kafka
6700 Kafka
9788 ConsoleConsumer
```

Producer Service

Consumer Service

Checking Fault Tolerance of the Topic in Kafka Cluster

Check Fault Tolerance of the Topic

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

- Stop the **Leader Kafka Server**, in this example Leader is "**Kafka Server 3**"
- **Go to the TAB where the Leader Kafka Server is running , press CTRL+C to stop the server**

```
[2017-11-26 11:56:01,227] INFO Session: 0x15ff3d138460010 closed (org.apache.zookeeper.ZooKeeper)
[2017-11-26 11:56:01,266] INFO [Kafka Server 3], shut down completed (kafka.server.KafkaServer)
```

Check Fault Tolerance of the Topic contd.

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

- Go to the **Producer (Sixth) TAB of the console**
- **Type Some More messages**

```
vagrant@master:~/bigdata/kafka$ bin/kafka-console-producer.sh --broker-list localhost:9093,localhost:9094,loc
alhost:9095 --sync  --topic gft-replicated-topic
This is first line
this is second line
this is third line
this is Fourth Line
This is fifth line
This is sixth line
```

Check Fault Tolerance of the Topic contd.

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

- Go to the **Consumer (Seventh) TAB of the console**
- **Check the new messages from the Producer consumed by the Consumer**

```
vagrant@master:~/bigdata/kafka$ bin/kafka-console-consumer.sh --zookeeper localhost:2181 --topic gft-replicat
ed-topic --from-beginning
Using the ConsoleConsumer with old consumer is deprecated and will be removed in a future major release. Cons
ider using the new consumer by passing [bootstrap-server] instead of [zookeeper].
This is first line
this is second line
this is third line
this is Fourth Line
[2017-11-26 11:55:58,886] ERROR [ConsumerFetcherThread-console-consumer-54803_master-1511696684348-6e9dd8db-0
-3], Error for partition [gft-replicated-topic,0] to broker 3:kafka.common.NotLeaderForPartitionException (ka
fka.consumer.ConsumerFetcherThread)
This is fifth line
This is sixth line
```

Check Fault Tolerance of the Topic contd.

2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070

- Now , Check the Description of the Topic "**gft-replicated-topic**" in the **Fifth TAB**

```
vagrant@master:~/bigdata/kafka$ bin/kafka-topics.sh --describe --zookeeper localhost:2181 --topic gft-replicated-topic
```

Topic:gft-replicated-topic PartitionCount:1 ReplicationFactor:3 Configs:
Topic: gft-replicated-topic Partition: 0 Leader: 1 Replicas: 3,1,2 Isr: 1,2

```
Topic:gft-replicated-topic      PartitionCount:1        ReplicationFactor:3     Configs:
        Topic: gft-replicated-topic     Partition: 0    Leader: 1       Replicas: 3,1,2 Isr: 1,2
```

Observe the Leadership has been changed from 3 to 1 implicitly to continue Writing & Reading of messages in uninterrupted manner even after the Leader is DOWN

Stop all the services in Kafka Cluster

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

- Go to each TAB of the console
- Press **CTRL+C** to stop the service

Use Cases

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

Here is a description of a few of the popular use cases for Apache Kafka

Log Aggregation

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

- Many people use **Kafka** as a <u>replacement for a **log aggregation solution**</u>.
  - **Log aggregation** typically collects **physical log files** off servers and puts them in a central place (a file server or HDFS perhaps) for processing.

- ***Kafka*** *abstracts away the details of files and gives a cleaner abstraction of log or event data as <u>a stream of messages</u>*.

Stream Processing

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

- Many users of Kafka process data in processing pipelines consisting of multiple stages, where
  - raw input data is consumed from **Kafka topics** and then **aggregated**, **enriched**, or otherwise transformed into new topics for further consumption or follow-up processing.

Questions

Conclusion

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

In this session we have discussed about

- **What is Data Streaming ?**
- **What is Apache Kafka?**
- **Kafka as a Messaging System**
- **Working with Kafka**
- **Simple Producer-Consumer Example using Single Node, Single Broker Kafka Cluster**
- **Simple Producer-Consumer Example using Single Node, Multi Broker Kafka Cluster**

References

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

- https://kafka.apache.org/

2f486967682070657266f726d616e63652c2044656c69766657265642c2f486967682070657266f726d616e63652c2044656c69766657265642c2f486967682070657266f726d616e63652c2044656c69766657265642c2f486967682070657266f726d616e63652c2044656c69766657265642c2f486967682070657266f726d616e63652c2044656c69766657265642c2f486967682070657266f726d616e63652c2044656c69766657265642c2f486967682070

Appendix

2f486967682070657266 6f726d616c63652c2044656c6976657265642c2f486967682070657266 6f726d616c63652c2044656c6976657265642c2f486967682070657266 6f726d616c63652c2044656c6976657265642c2f486967682070657266 6f726d616c63652c2044656c6976657265642c2f486967682070657266 6f726d616c63652c2044656c6976657265642c2f486967682070657266 6f726d616c63652c2044656c6976657265642c2f486967682070

Automated setup of Single Node Hadoop Cluster (Pseudo Distributed Mode) with Apache Kafka

Pre-requisite Software download & installation

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

Please download the following software and install it on your Windows Desktop/Laptop .

1) **Download Oracle Virtual Box and install it**

http://download.virtualbox.org/virtualbox/5.2.0/VirtualBox-5.2.0-118431-Win.exe

2) **Download Virtual Box extension pack and install it**

http://download.virtualbox.org/virtualbox/5.2.0/Oracle_VM_VirtualBox_Extension_Pack-5.2.0-118431.vbox-extpack

3) **Download Vagrant Tool and install it**

https://releases.hashicorp.com/vagrant/2.0.1/vagrant_2.0.1_x86_64.msi

4) **Download Putty**

https://the.earth.li/~sgtatham/putty/latest/w64/putty.exe

Pre-requisite Software download & installation contd.

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

5) **Download WinSCP and install it**

https://cdn.winscp.net/files/WinSCP-5.11.2-Setup.exe?secure=a9zndySJto_YagLtOO8c3Q==,1509768826

6) **Download Multi-Tab Putty Client and install it**

http://sysprogs.com/files/SmarTTY/SmarTTY-2.2.msi

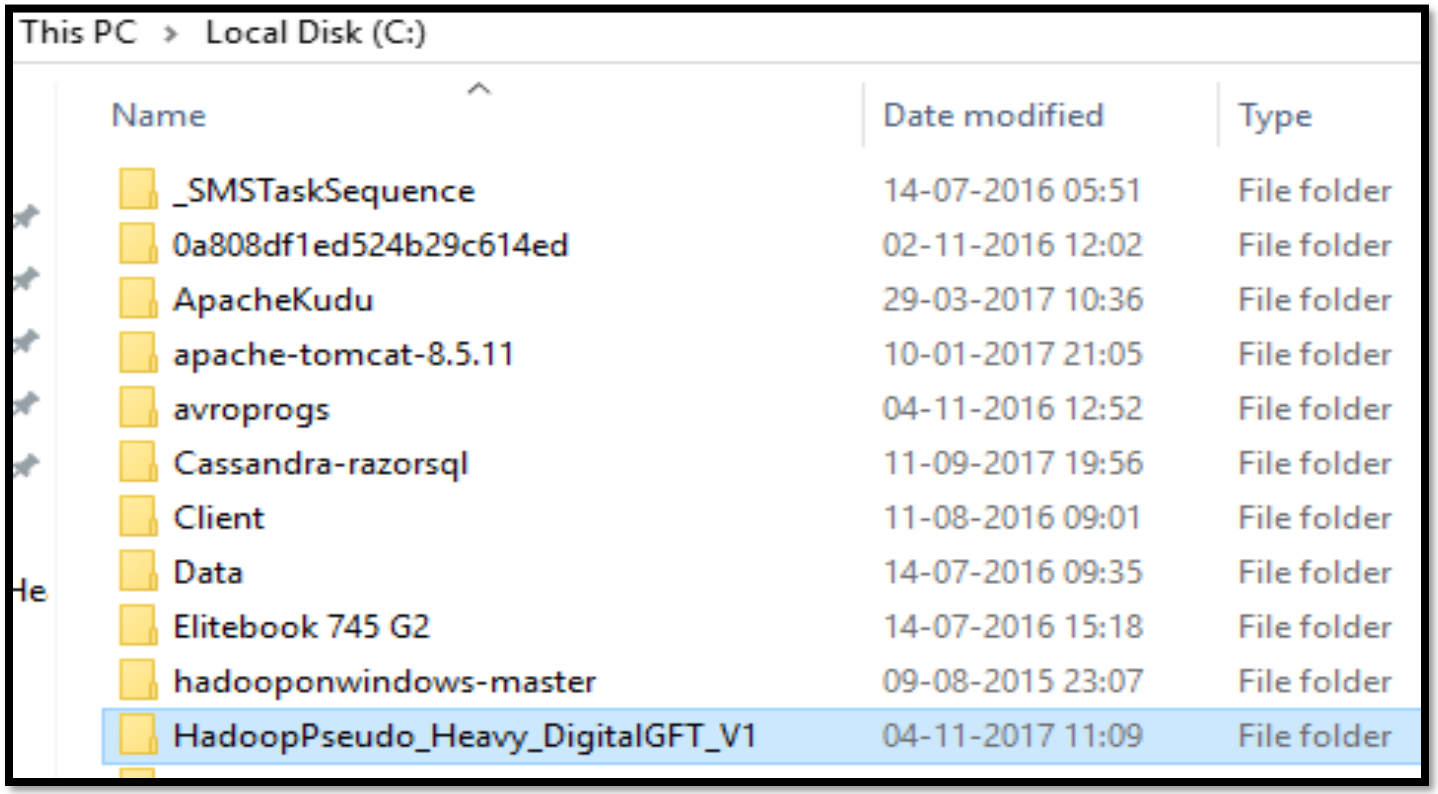
Start installation of Hadoop Cluster

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

1. Check the location in your desktop where you have copied the following " **HadoopPseudo_Heavy_DigitalGFT_V1.zip**" , which has been shared with you.
   - The "Zip" file will be shared by your respective Faculty Member /Co-ordinator
   - Preferably keep the file directly in "C" drive

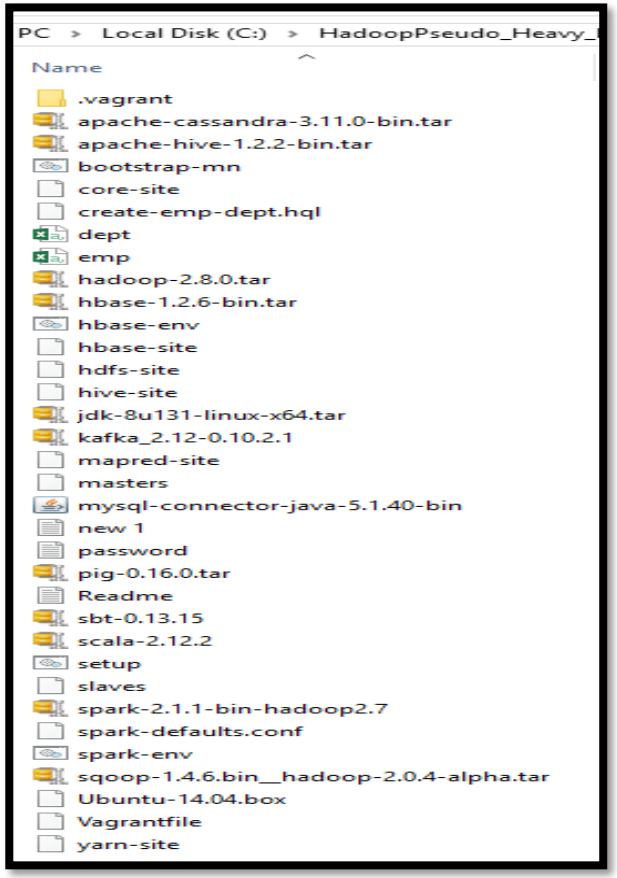
Installation contd.

2. Unzip it directly in C drive , it will create a folder by the name of "HadoopPseudo_Heavy_DigitalGFT_V1"



Installation contd.

3. After unzipping, open the folder , the contents of the folder will be shown as following –



Installation contd.

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

4. Open **command prompt** of your Windows, change your current location to **HadoopPseudo_Heavy_DigitalGFT_V1 directory**

Command Prompt

```
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\raju.chal>cd ..

C:\Users>cd ..

C:\>cd HadoopPseudo_Heavy_DigitalGFT_V1

C:\HadoopPseudo_Heavy_DigitalGFT_V1>
```

Installation contd.

2f486967682070657266f726d616c63652c2044656c6976657265642c2f486967682070657266f726d616c63652c2044656c6976657265642c2f486967682070657266f726d616c63652c2044656c6976657265642c2f486967682070657266f726d616c63652c2044656c6976657265642c2f486967682070657266f726d616c63652c2044656c6976657265642c2f486967682070

5. Now in the **<u>command prompt</u>** type the command "**vagrant up** ", press Enter

C:\HadoopPseudo_Heavy_DigitalGFT_V1> vagrant up

```
C:\HadoopPseudo_Heavy_DigitalGFT_V1>vagrant up
Bringing machine 'Hadoop' up with 'virtualbox' provider...
==> Hadoop: Clearing any previously set forwarded ports...
```

Installation contd.

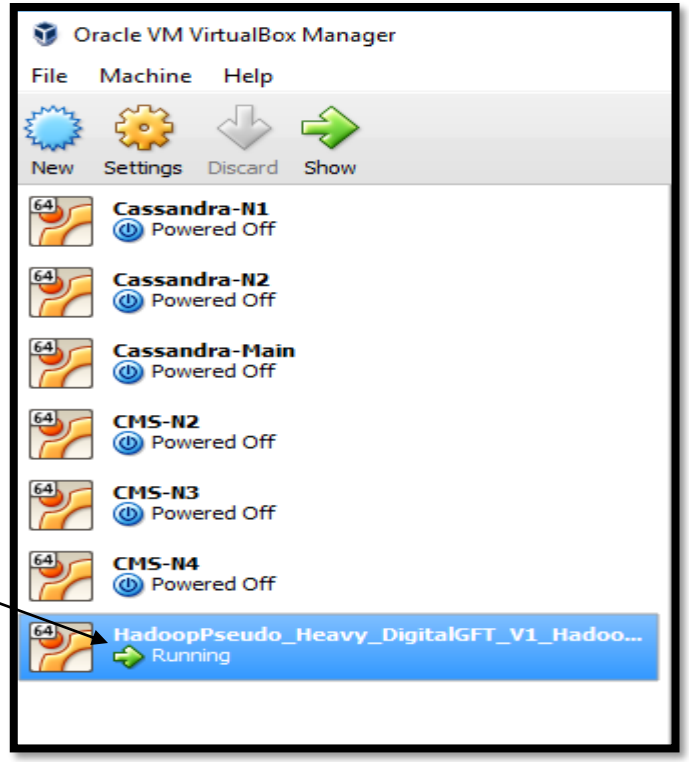
2f486967682070657266f726d616c63652c2044656c6976657265642c2f486967682070657266f726d616c63652c2044656c6976657265642c2f486967682070657266f726d616c63652c2044656c6976657265642c2f486967682070657266f726d616c63652c2044656c6976657265642c2f486967682070657266f726d616c63652c2044656c6976657265642c2f486967682070

- The setup will start. It will take 35 minutes to 50 minutes to complete all the installations and configurations depending on the band-width of internet connection.
- Please wait till you get back the command prompt on your screen.

C:\HadoopPseudo_Heavy_DigitalGFT_V1>

6. Open the Oracle VirtualBox that you have already installed, you will observe one

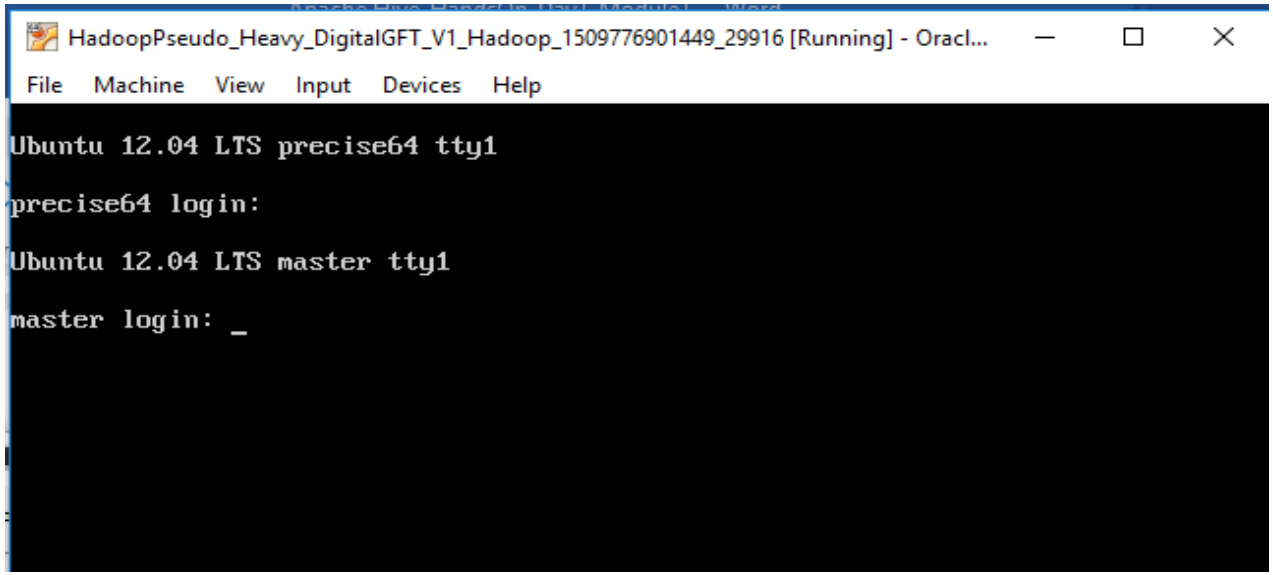
Linux machine is running as shown in the figure



Installation contd.

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

7. Select the Linux box and click on the <u>Show</u> button in the toolbar, you will be getting the following screen

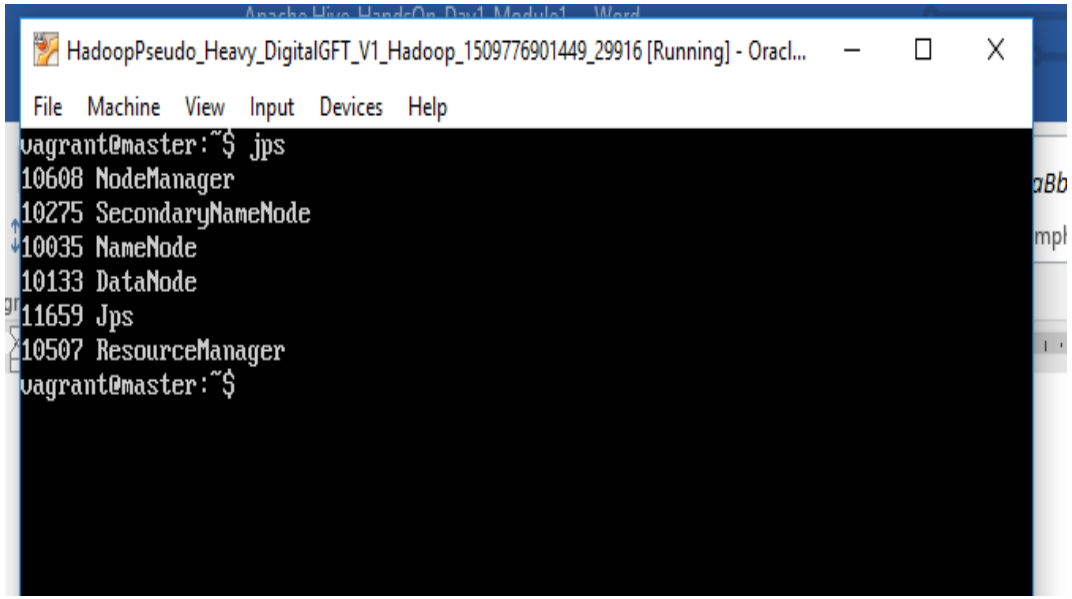


Login name :- vagrant
Password :- vagrant
IP Address :- 192.168.56.70

Check the Hadoop Services

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

Type jps command in $ prompt



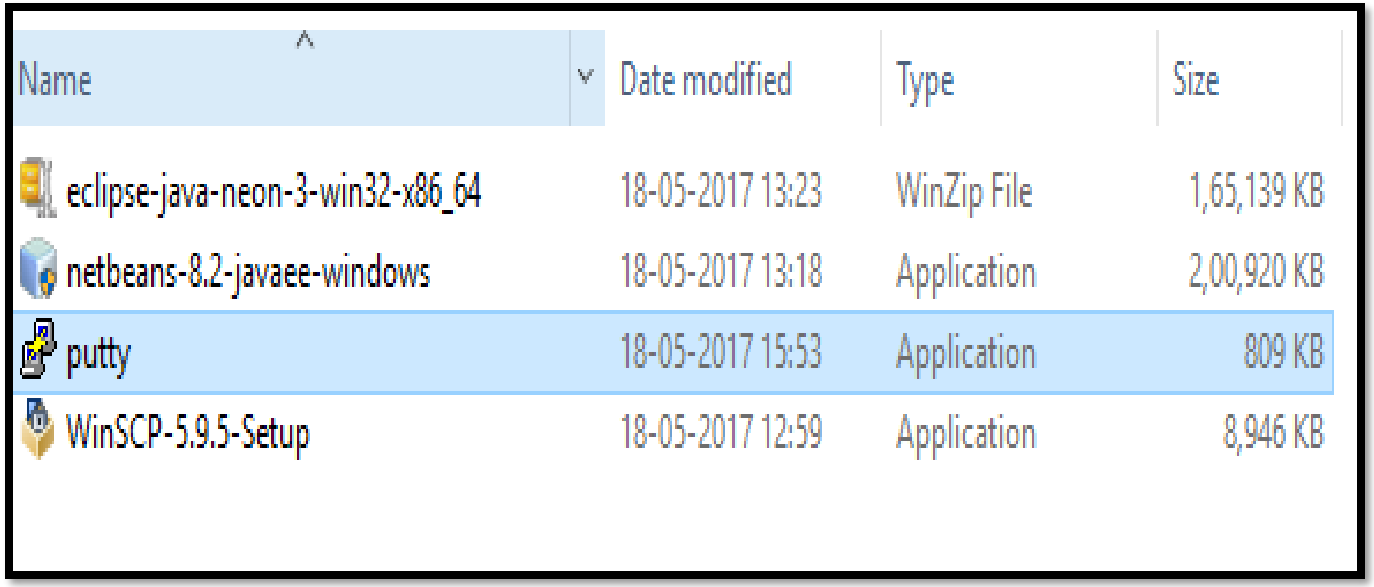
The following services are running in the Hadoop Node

1. Primary Name Node /Name Node
2. Secondary Name Node
3. Data Node
4. Resource Manager
5. Node Manager

Connecting putty with the Hadoop Node

2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070

8. Open the "putty" that you have already downloaded just by double clicking on it .

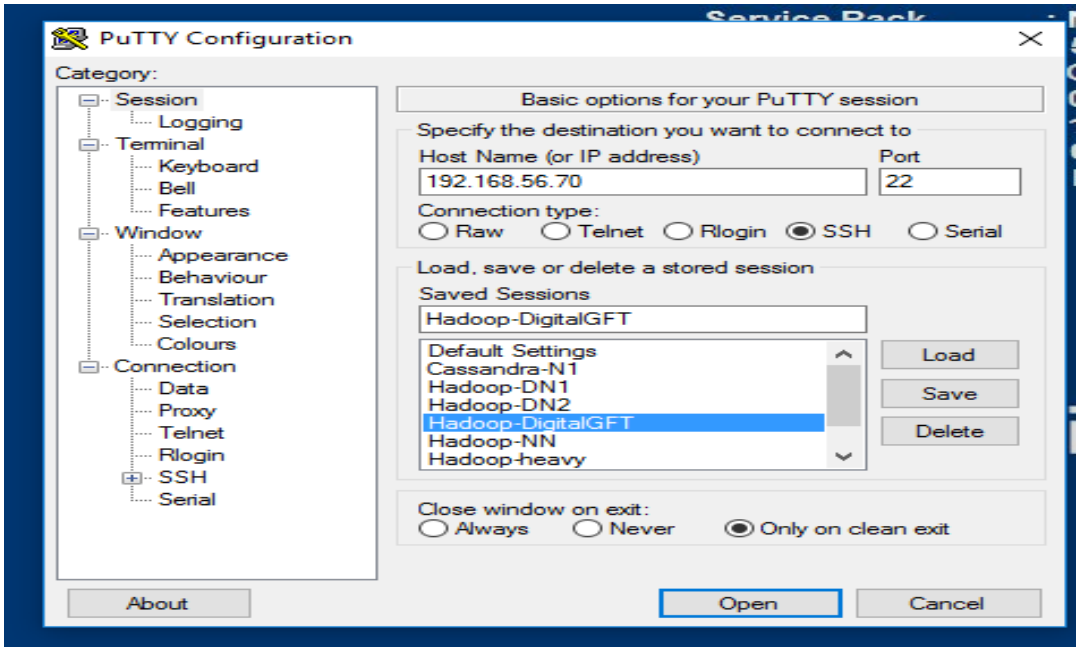


Name	Date modified	Type	Size
eclipse-java-neon-3-win32-x86_64	18-05-2017 13:23	WinZip File	1,65,139 KB
netbeans-8.2-javaee-windows	18-05-2017 13:18	Application	2,00,920 KB
putty	18-05-2017 15:53	Application	809 KB
WinSCP-5.9.5-Setup	18-05-2017 12:59	Application	8,946 KB

Connecting putty with the Hadoop Node

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

9. You will be getting the following screen, Fill the text boxes with the given data and click on OPEN.



- **Put 192.168.56.70 in Host Name Text Box**
- **Save the session as Hadoop-DigitalGFT**

Connecting putty with the Hadoop Node

2f486967682070657266 6f726d616c63652c2044656c6976657265642c2f486967682070657266 6f726d616c63652c2044656c6976657265642c2f486967682070657266 6f726d616c63652c2044656c6976657265642c2f486967682070657266 6f726d616c63652c2044656c6976657265642c2f486967682070657266 6f726d616c63652c2044656c6976657265642c2f486967682070

10. You will be getting another dialog box, click on OK.

You will be connected with you Linux box from Putty and will be getting the $ prompt on the screen .

vagrant@master: ~

```
vagrant@master:~$ jps
2580 DataNode
3669 Jps
2421 NameNode
3207 Master
3575 Worker
2809 SecondaryNameNode
2987 ResourceManager
3149 NodeManager
vagrant@master:~$
```

Type the command jps in the $ prompt and confirm the above services are running.

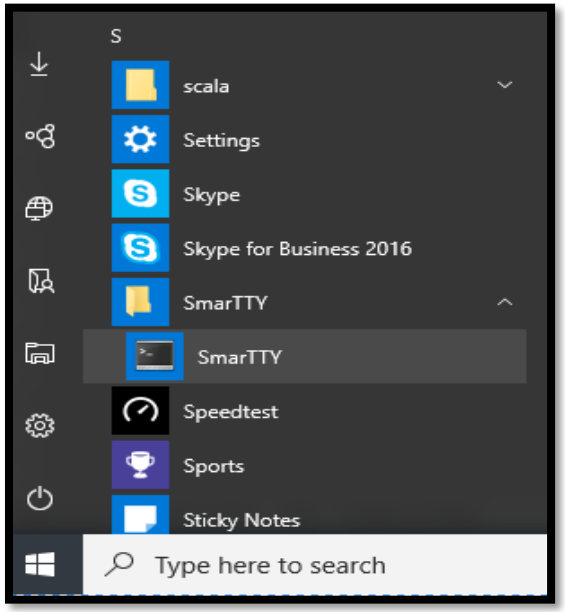
Connecting SmarTTY with the Hadoop Node

2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070657266 6f726d616e63652c2044656c6976657265642c2f486967682070

11. SmarTTY is a free multi-tabbed SSH client that supports copying files and directories with SCP on-the-fly and editing files in-place.

Note :- To work in $ prompt and hive CLI simultaneously we need multi-tabbed SSH client . If you work with SmartTTY, putty is not required.

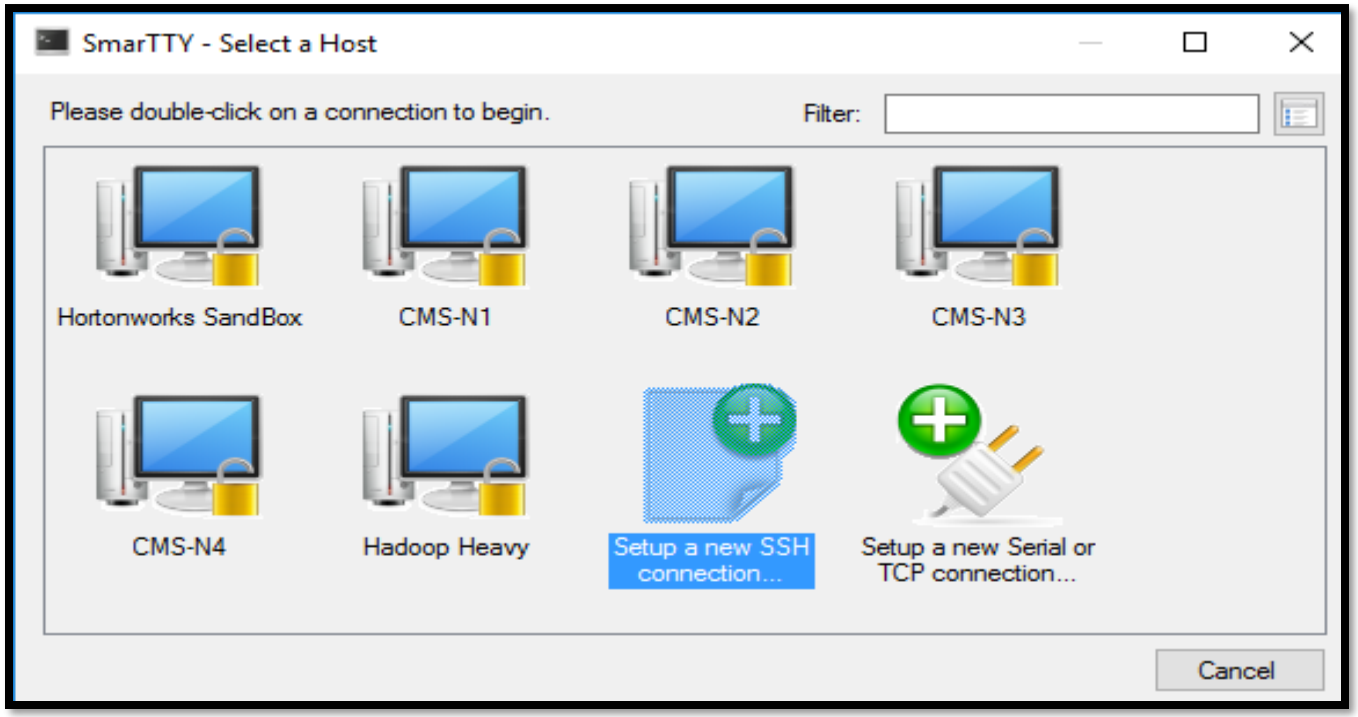
To Connect SMartTTY with Hadoop Node , click on SmartTTY menu ,



Connecting SmarTTY with the Hadoop Node

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

Click on "Setup a new SSH Connection "



Connecting SmarTTY with the Hadoop Node

2f486967682070657266 6f726d616c63652c2044656c6976657265642c2f486967682070657266 6f726d616c63652c2044656c6976657265642c2f486967682070657266 6f726d616c63652c2044656c6976657265642c2f486967682070657266 6f726d616c63652c2044656c6976657265642c2f486967682070657266 6f726d616c63652c2044656c6976657265642c2f486967682070

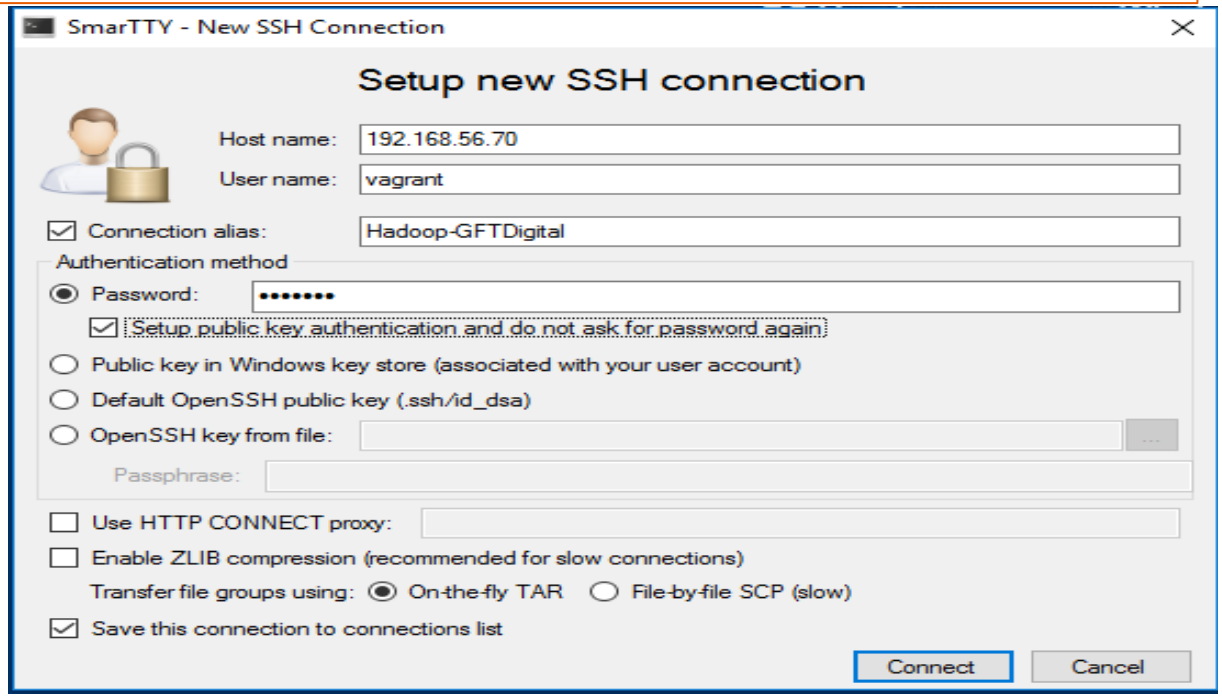
Fill the "New SSH Connection" dialog box with the following information as shown below

Host Name :-
192.168.56.70

User Name :- vagrant

Password :- vagrant

Connection Alias :-
Hadoop-GFTDigital



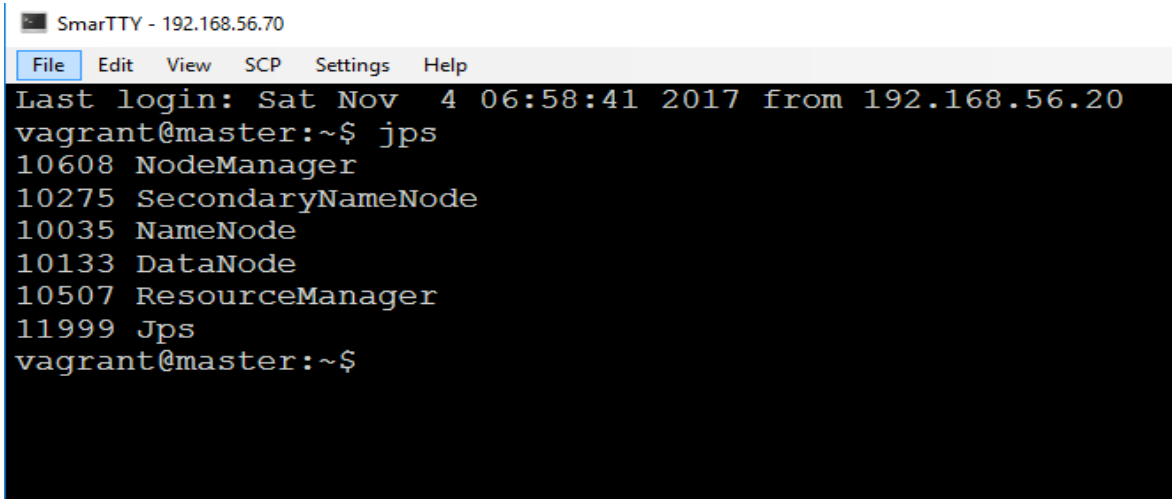
Click on "Connect"

Connecting SmarTTY with the Hadoop Node

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

You will be getting Linux console with $ prompt

Type jps in $ prompt



Now your environment is ready .

2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070657266f726d616e63652c2044656c6976657265642c2f486967682070

END of MODULE-1

THANK YOU