

Hadoop

### Hadoop Nedir ?

* İlişkisel veri tabanlarından farklı olarak Hadoop verileri bir den fazla bilgisayarda saklar. Bu sayede büyük ve sanal bir disk oluşturur. Bu verileri paralel olarak işleyebilir. Bu sayede işlem yükünü bilgisayarlara dağıtmış olur ve hesaplama zamanı azaltır.

## 

## 

sayfa sonu

### HADOOP Kurulumu

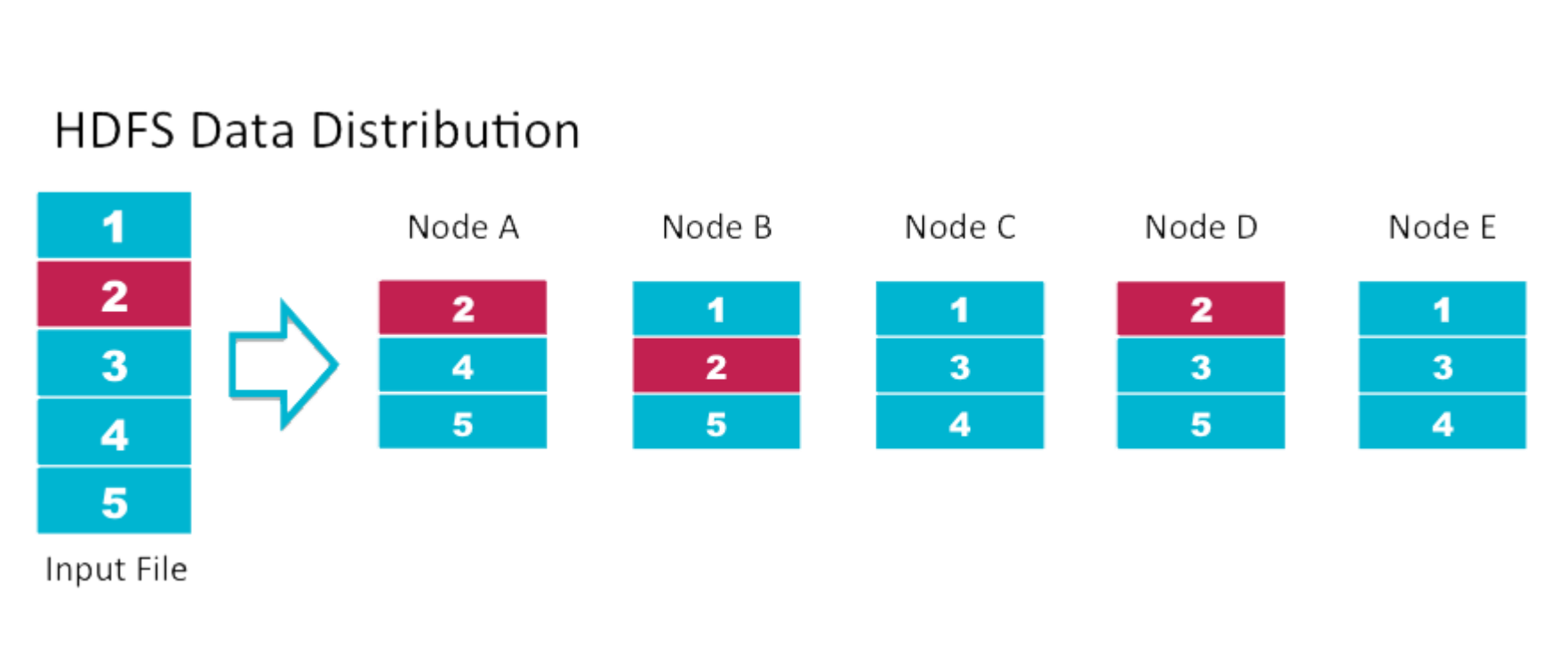
* sudo apt update
* sudo apt install openjdk-8-jdk -y
* sudo adduser hdoop
* su - hdoop
* ssh-keygen -t rsa -P '' -f ~/.ssh/id\_rsa
* cat ~/.ssh/id\_rsa.pub >> ~/.ssh/authorized\_keys
* chmod 0600 ~/.ssh/authorized\_keys
* ssh localhost
* wget <https://downloads.apache.org/hadoop/common/hadoop-3.2.1/hadoop-3.2.1.tar.gz>
* tar xzf hadoop-3.2.1.tar.gz
* sudo nano .bashrc
  + export HADOOP\_HOME=/home/hdoop/hadoop-3.2.1
  + export HADOOP\_INSTALL=$HADOOP\_HOME
  + export HADOOP\_MAPRED\_HOME=$HADOOP\_HOME
  + export HADOOP\_COMMON\_HOME=$HADOOP\_HOME
  + export HADOOP\_HDFS\_HOME=$HADOOP\_HOME
  + export YARN\_HOME=$HADOOP\_HOME
  + export HADOOP\_COMMON\_LIB\_NATIVE\_DIR=$HADOOP\_HOME/lib/native
  + export PATH=$PATH:$HADOOP\_HOME/sbin:$HADOOP\_HOME/bin
  + export HADOOP\_OPTS="-Djava.library.path=$HADOOP\_HOME/lib/nativ"
* source ~/.bashrc
* sudo nano $HADOOP\_HOME/etc/hadoop/hadoop-env.sh
  + export JAVA\_HOME=/usr/lib/jvm/java-8-openjdk-amd64
* readlink -f /usr/bin/javac
* sudo nano $HADOOP\_HOME/etc/hadoop/core-site.xml
  + <configuration>
  + <property>
  + <name>hadoop.tmp.dir</name>
  + <value>/home/hdoop/tmpdata</value>
  + </property>
  + <property>
  + <name>fs.default.name</name>
  + <value>hdfs://127.0.0.1:9000</value>
  + </property>
  + </configuration>
* sudo nano $HADOOP\_HOME/etc/hadoop/hdfs-site.xml
  + <configuration>
  + <property>
  + <name>dfs.data.dir</name>
  + <value>/home/hdoop/dfsdata/namenode</value>
  + </property>
  + <property>
  + <name>dfs.data.dir</name>
  + <value>/home/hdoop/dfsdata/datanode</value>
  + </property>
  + <property>
  + <name>dfs.replication</name>
  + <value>1</value>
  + </property>
  + </configuration>
* sudo nano $HADOOP\_HOME/etc/hadoop/mapred-site.xml
  + <configuration>
  + <property>
  + <name>mapreduce.framework.name</name>
  + <value>yarn</value>
  + </property>
  + </configuration>
* sudo nano $HADOOP\_HOME/etc/hadoop/yarn-site.xml
  + <configuration>
  + <property>
  + <name>yarn.nodemanager.aux-services</name>
  + <value>mapreduce\_shuffle</value>
  + </property>
  + <property>
  + <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
  + <value>org.apache.hadoop.mapred.ShuffleHandler</value>
  + </property>
  + <property>
  + <name>yarn.resourcemanager.hostname</name>
  + <value>127.0.0.1</value>
  + </property>
  + <property>
  + <name>yarn.acl.enable</name>
  + <value>0</value>
  + </property>
  + <property>
  + <name>yarn.nodemanager.env-whitelist</name>
  + <value>JAVA\_HOME,HADOOP\_COMMON\_HOME,HADOOP\_HDFS\_HOME,HADOOP\_CONF\_DIR,CLASSPATH\_PERPEND\_DISTCACHE,HADOOP\_YARN\_HOME,HADOOP\_MAPRED\_HOME</value>
  + </property>
  + </configuration>
* hdfs namenode -format
* ./start-dfs.sh
* ./start-yarn.sh
* jps
* [http://localhost:9870](http://localhost:9870/)
* [http://localhost:9864](http://localhost:9864/)
* http://localhost:8088

## 

Kurulum linkleri:

* https://phoenixnap.com/kb/install-hadoop-ubuntu

### HDFS



* HDFS verileri replication factor (3) değerine göre her verinin kopyalarını diğer node larda da tutar. Bu sayede bir node’un zarar görmesi durumunda veri kaybedilmemiş olur.
* Node’lerde yer alan verileri merkeze toplayıp işlemek yerine SQL diline yakın sorguları Node’lere dağıtarak Node’lerde gerekli işlemi gerçekleştirir. Çünkü her Node’nin kendisine ait işlemcisi ve rami vardır. Bunu yapmasının sebebi ise veriyi merkeze çekip trafik oluşturmamak.

### 

### 

### HDFS Komutları

* hdfs dfs -ls / : Hdfs ana dizinindeki dosyaları listeler.
* hdfs dfs -mkdir /example : Hdfs ana dizinine example adında klasör oluşturur.
* hdfs dfs -count /example : example klasöründeki dosya sayısını gösterir.
* hdfs dfs -copyFromLocal /zaid/example.csv /example/: Localden hdfs e dosya taşımaya yarar.
  + Desktop zaid klasörünün içinde hdfs içinde direk ls çalıştırarak görebiliriz.
* hdfs dfs -cat /example/example.csv : Dosyanın içeriğini gösterir.
* hdfs dfs -cp /example/S01G1AllChannels.csv / : Dosya kopyalamaya yarar.
* hdfs dfs -rm /S01G1AllChannels.csv : Dosya silmeye yarar.
* hdfs dfs -mv /example/S01G1AllChannels.csv / : Dosya taşımaya yarar.
* hdfs dfs -chmod +-izin /example/S01G1AllChannels.csv : İzin alma verme.

### Hadoop-Error

* Hadoop: There are 0 datanode(s) running and no node(s) are excluded in this operation :
  + stop-all.sh
  + rm -rf /usr/local/hadoop\_store/hdfs/datanode/\*
  + start-all.sh

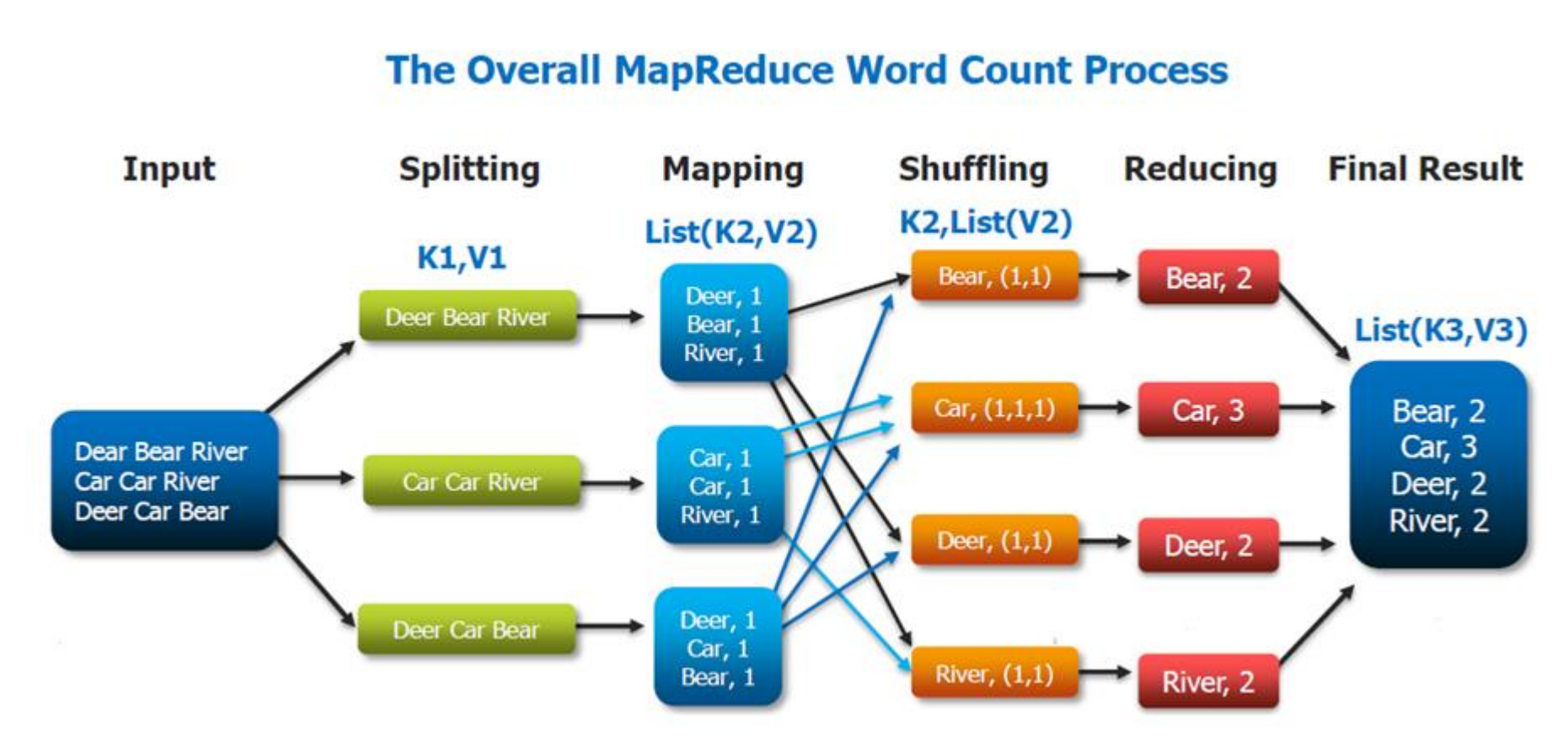
# MapReduce

* Hadoop içerisinde büyük veri kümelerini performanslı bir şekilde işleyen ve analiz eden bileşene MapReduce denir.
* İlk olarak Map(haritalama) aşaması gerçekleşir veriler filtrelenir ve sonrasında sonuç elde etme için Recude(indirgeme) işlemine geçiş yapılır.
* Bir Map işlemi tamamlandığında Recude işlemine geçiş yapılabilmektedir. Tüm Map işlemlerinin beklenmesine gerek duyulmamaktadır.

### 

### 

### MapReduce Adımları



* Input: Verilerin girişlerinin yapıldığı adımdır.
* Splitting: Gelen veriler bu aşamada daha kolay işlenebilmesi için parçalara ayrılır.
* Mapping: Veriler bu aşamada ilgili düğümlere dağıtılır ve kaç tane yedeği olacağı bu adımda belirtilir. Ve daha sonrasında veriler ilgili düğümde işlenir.
* Shuffling: Her düğümde verilerin sayma işlemi yapılır. Örneğin bir text dökümanını input olarak verdiysek ve sonuç olarak hangi kelimenin kaç defa geçtiğini arıyorsak bu aşamada kelime sayıları Node’lerde belirlenir.
* Reducing: Her Node’den gelen sonuç bu aşamada toplanır.
* Final Result: Sonuçlar artık elimizdedir. Bunun raporlamasını yapabiliriz.

## APACHE HİVE

Büyük veriler ile hadoop üzerinden işlem yapmak için kullanılan bir MapReduce geliştirme yöntemidir. Büyük veriler üzerinde sql sorguları ve basit analizler yapmak için kullanılır.

### Apache Hive Kodlama Adımları:

* Veriyi hdfs’ e at.
  + hdfs dfs -copyFromLocal /Kaynak /Hedef
* Veri tabanı oluştur.
  + CREATE database My\_DB;
* Tablo oluştur.
  + CREATE EXTERNAL TABLE IF NOT EXISTS TableName(ColumnName: ColumnType)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ’,’

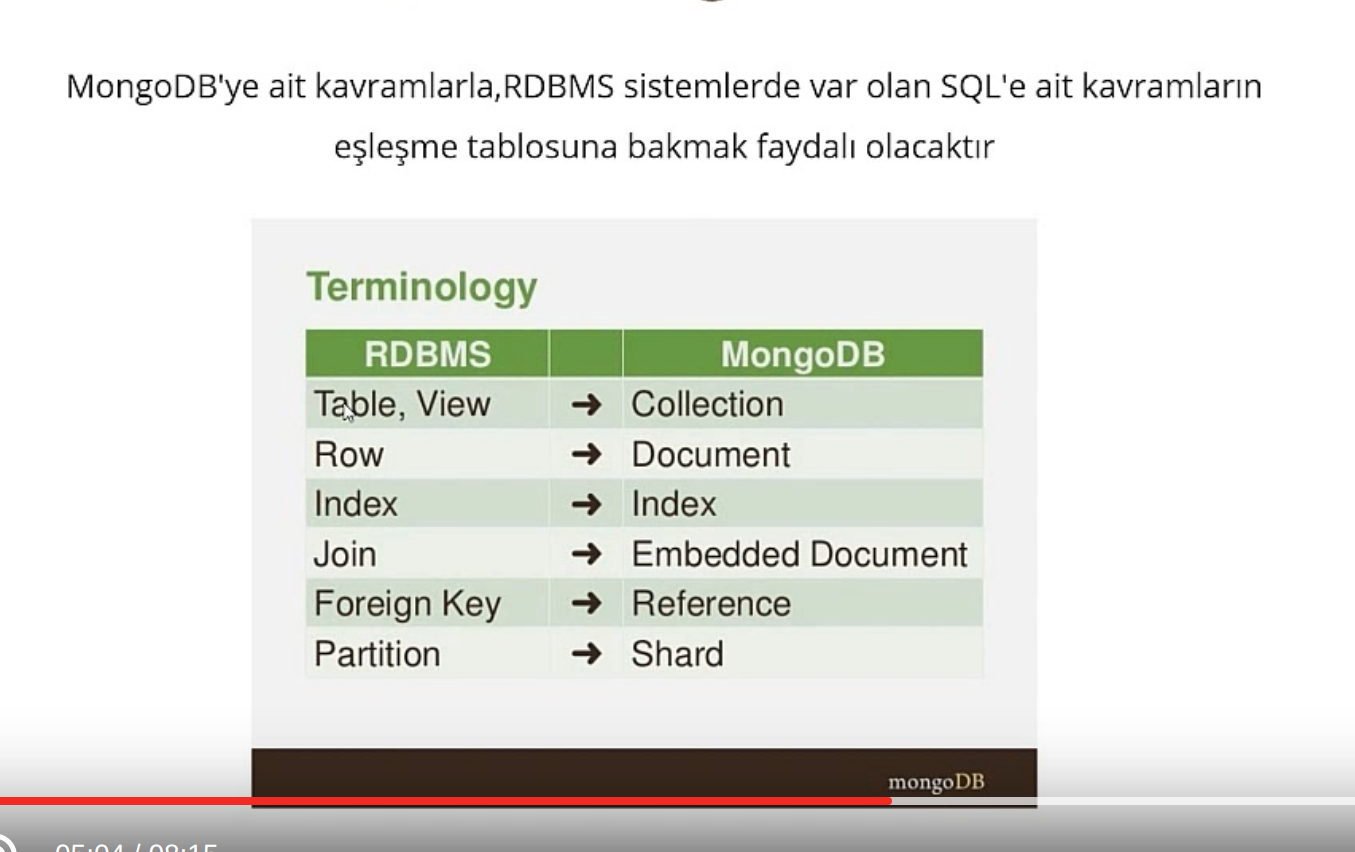
LINES TERMINATED BY ‘\n’

STORED AS TEXTFILE;

* LOAD DATA INPATH ‘/example/MOCK\_DATA.csv’ INTO TABLE My\_DB.TableName;
* LOAD DATA LOCAL ‘/home/MOCK\_DATA.csv’ INTO TABLE My\_DB.TableName;
* Sql sorgusu yaz.

## MongoDB



Belge yönelimli (document-oriented) bir NoSQL veritabanıdır. MongoDB’de her kayıt bir dökümandır. Dökümanlar JSON benzeri Binary JSON(BSN) formatında tutulur.

### Neden MongoDB

* Ölçeklenebilir(Scalable): Veri boyutunun arttığı durumlarda veya performans sorunu yaşadığımız durumlarda makine ekleyebiliriz.
* Veriler document (belge) biçiminde saklanır. Burada JSON verilerini kullanabiliriz.
* Verilerin birden fazla kopyası saklanabilir ve veri kaybı yaşanmaz. (Replication)

### MongoDB Kurulumu

* curl -fsSL https://www.mongodb.org/static/pgp/server-4.4.asc | sudo apt-key add -
* echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu focal/mongodb-org/4.4 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-4.4.list
* sudo apt update
* sudo apt install mongodb-org
  + Hata verirse (libssl1.1):
  + wget http://archive.ubuntu.com/ubuntu/pool/main/o/openssl/libssl1.1\_1.1.0g-2ubuntu4\_amd64.deb
  + sudo dpkg -i ./libssl1.1\_1.1.0g-2ubuntu4\_amd64.deb
* sudo systemctl start mongod.service
* sudo systemctl status mongod
* sudo systemctl enable mongod

### MongoDB Komutları

* show dbs : Databaseleri listeler.
* use database : Var olan database i seçer yok ise oluşturur.
* db.createCollection(‘collection\_name’) : Collection oluşturur.
* db.myCollection.insert({}) : Collection’a document ekler.
* db.myCollection.find() : Docmentleri listeler.
* db.myCollection.find().pretty() : JSON gibi listeler.
* db.myCollection.update({age : 20}, {$set: {age: 23}}) : Güncelleme
* db.myCollection.remove({name: "navindu"}) : Silme

### 

### 

### 

### 

### 

### 

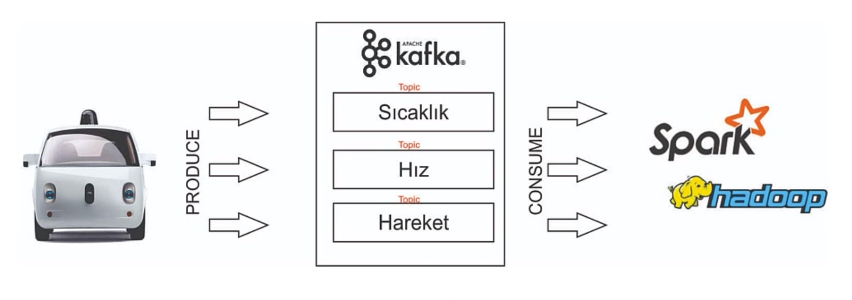
### MongoDB vs SQL

* db.myCollection.find()
  + SELECT \* FROM myCollection;
* db.myCollection.find({‘name’:’Mucahit’})
  + SELECT \* FROM myCollection WHERE name = ‘Mucahit’;
* db.myCollection.find({},{‘\_id’:0, ‘name’:1, ‘job’:1})
  + SELECT name, job FROM myCollection;
* db.myCollection.find({‘name’: {‘$in’:[‘Mucahit’]})
  + SELECT \* FROM myCollection WHERE name in (‘Mucahit’);
* db.myCollection.find({‘$and’:[{‘gender’:’female’}, {‘age’:{‘$gt’:25}}])
  + SELECT \* FROM myCollection WHERE gender = ‘female’ AND age > 25;
* db.myCollection.find({‘$and’:[{‘gender’:’female’}, {‘age’:{‘$lt’:25}}])
  + SELECT \* FROM myCollection WHERE gender = ‘female’ OR age < 25;
* db.myCollection.find({‘name’:{‘$ne’:’Mucahit’}})
  + SELECT \* FROM myCollection WHERE NOT name = ‘Mucahit’;
* db.myCollection.find({‘$and’:[{‘gender’:’female’}, {‘$or’:[‘age’:{‘$lt’:20}, {‘name’:{‘$regex’:’^A’}}]}]})
  + SELECT \* FROM myCollection WHERE gender = ‘female’ AND (age < 20 OR name LIKE ‘A%’);

## Apache Kafka



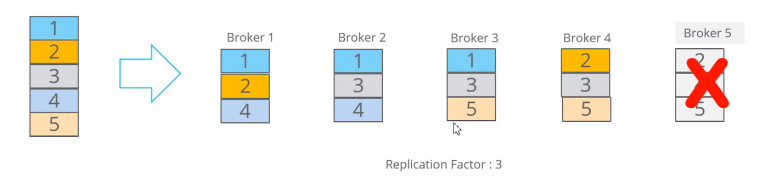
Apache Kafka, verilerin bir sistemden hızlı bir şekilde toplanıp diğer sistemlere hatasız bir şekilde transferini sağlamak için geliştirilen dağıtık bir veri akış mekanizmasıdır.



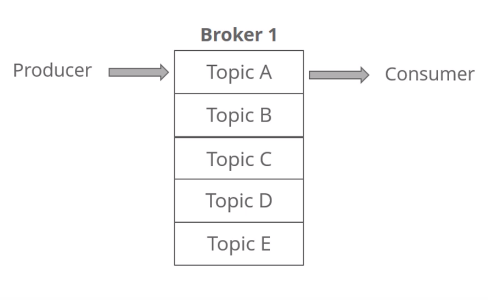
Bu sayede Consumer ‘ın çalışamaz hale geldiği durumlarda kafka veriyi saklar ve Consumer ayağa kalktığında veri akışına devam eder. Kafka FIFO prensibi ile çalışır.

### Kafka Kavramları

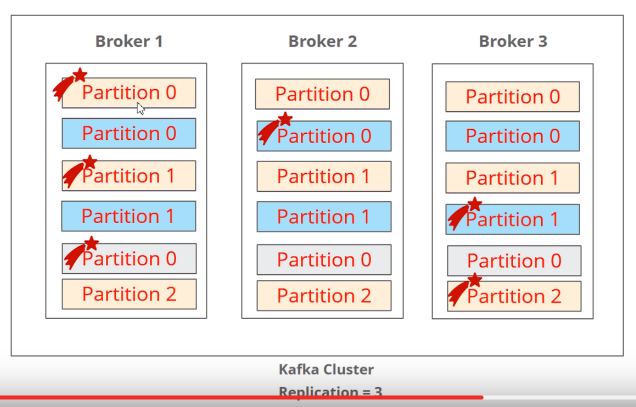
Apache Kafka Replication, replication factor değerine göre yedekleyerek verileri saklar. Bu sayede veri kaybını önler.



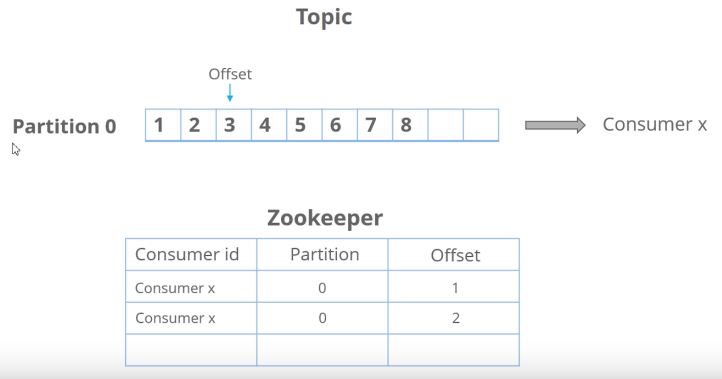
Apache Kafka Topic, Veriler Topiclerde saklanır. Topicler ise Brokerlarda (Kafka Mankinaları) birden fazla brokerın olduğu yapılara kafka clusterı denir. Bir broker içinde birden fazla topic olabilir. Producerler bu topiclere veri gönderir. Consumer ise bu topiclerden verileri alır.



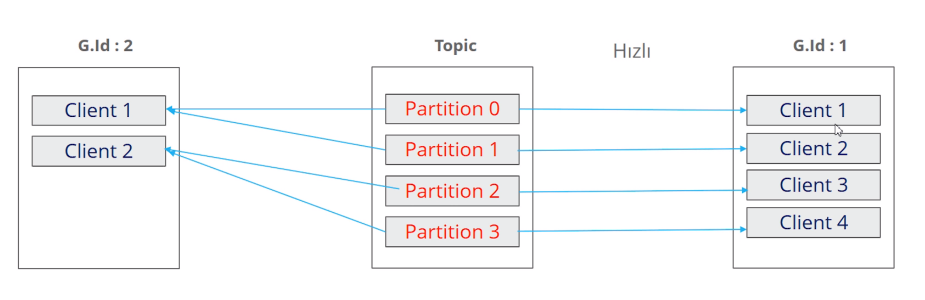
Apache Kafka Partition, Veriler kafka topic lerinde partitionların içinde saklanır. Bu partitionlar verilen partitions sayısına göre yedeklenerek saklanır. Bu sayede veri kaybı önlenir. Bu partitionların leaderleri vardır bu leaderler gelen veriyi diğer replica partitionlara otomatik olarak yazarlar.



Kafka Offset, Consumerin veriyi okurken hangi veride kaldığı bilgisini tutar.



Kafka Group, Partitions lar aynı anda farklı gruplarca okunabilir.



### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### Apache Kafka Kurulumu(Düzenlenecek)

* sudo apt-get update
* sudo apt-get install default-jre
* sudo apt-get install zookeeperd
* netstat -ant | grep :2181
* sudo systemctl status zookeeper
* cd /opt/kafka/ : sudo wget <https://archive.apache.org/dist/kafka/3.1.0/kafka_2.12-3.1.0.tgz>
* sudo tar -xvf kafka 2.12-3.1.0.tgz
* sudo nano /etc/profile :
  + export KAFKA\_HOME=”/opt/kafka/kafka\_2.12-3.1.0”
  + export PATH=”$PATH{$KAFKA\_HOME}/bin”
* sudo nano ~/.bashrc : alias sudo=’sudo env PATH=”$PATH”’
* echo $KAFKA\_HOME & echo $PATH (Kontrol için)
* sudo ln -s $KAFKA\_HOME/config/server.properties /etc/kafka.properties
* ls -ln /etc/kafka.properties
* sudo ufw from ip to any port 2181 (zookeeper)
* sudo ufw from ip to any port 9092 (kafka)
* /opt/kafka/kafka\_2.12-3.1.0/config/server-properties :
  + listeners=PLAINTEXT://0.0.0.0:9092
  + advertised.listeners=PLAINTEXT://ip

<https://tecadmin.net/how-to-install-apache-kafka-on-ubuntu-22-04/>

### 

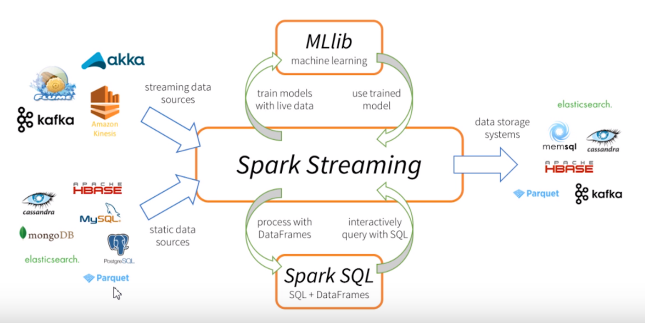
### Apache Kafka Komutları

* sudo systemctl start zookeeper : Zookeeper’ ı ayağa kaldırır.
* sudo systemctl start kafka : Kafka’ yı ayağa kaldırır.
* kafka-topics.sh --create --topic topicName --bootstrap-server localhost:9092 --replication-factor 1 --partitions 1 : Topic oluşturur.
* kafka-topics.sh --list --bootstrap-server localhost:9092 : Topicleri listeler.
* kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic sampleTopic --from-beginning : Kafka Consumer (Listener)
* kafka-console-producer.sh --broker-list localhost:9092 --topic sampleTopic : Kafka Producer

## Apache Spark



Büyük verileri dağıtık işlem ile analiz edebilmek için Scala ile geliştirilmiş Büyük Veri kütüphanesidir. Spark in-memory çalışmaktadır. Bundan dolayı depolama birimi yoktur. Veri analizini ram üzerinde gerçekleştirir.



Spark’ın bir çok modülü vardır. Spark SQL ile sql queryleri yazarak veri analizi yapılabilir. Spark Streaming ile real time analizler yapılabilir. MLlib kütüphanesi ile makine öğrenmesi algoritmaları ile tahminleme yapılabilir.

### Apache Spark Kurulumu

* sudo apt install default-jdk
* cd /opt/spark: sudo wget <https://dlcdn.apache.org/spark/spark-3.2.1/spark-3.2.1-bin-hadoop3.2.tgz>
* sudo tar xvf spark-3.2.1-bin-hadoop3.2.tgz
* sudo nano /etc/profile :
  + export export SPARK\_HOME=/opt/spark/spark-3.2.1-bin-hadoop3.2
  + export PATH=$PATH:$SPARK\_HOME/bin:$SPARK\_HOME/sbin
  + export PYSPARK\_PYTHON=/usr/bin/python3.10
* source ~/.profile or reboot
* Spark başlatma:
  + sudo start-master.sh :
    - [http://127.0.1.1:8080/](http://127.0.0.1:8080/)
    - http://127.0.1.1:8081/
  + sudo start-slave.sh spark://ubuntu-machine:7077
* Spark durdurma:
  + sudo stop-master.sh
  + sudo stop-worker.sh
* Pyspark’ın spark dosyalarını bulunabilmesi için:
  + pip3 install findspark
  + findspark.init(/opt/spark/spark-3.2.1-bin-hadoop3.2)
* Çalışan spark job'unu web'den takip için:
  + <http://192.168.1.108:4040/>
  + http://192.168.1.108:4041/

### Spark-MongoDB Connector Configuration

* /spark/bin/ pyspark:
  + --conf "spark.mongodb.input.uri=mongodb://127.0.0.1/test.myCollection?readPreference=primaryPreferred" \
  + --conf "spark.mongodb.output.uri=mongodb://127.0.0.1/test.myCollection" \
  + --packages org.mongodb.spark:mongo-spark-connector\_2.12:3.0.2
    - Terminalde dosyaların nereye indirildiği gözükür /spark/jar/ içine inmemişse dosyaları bu dizine almalıyız.

### Spark-Kafka Connector Configuration

* /spark/bin/ spark-shell –packages org.apache.spark:spark-sql-kafka-0-10\_2.12:3.2.1
  + Terminalde dosyaların nereye indiği gözükür /spark/jar/ içine inmemişse dosyaları bu dizine almalıyız. (/home/.ivy2 içine iner)

### Spark Streaming

Model Training Aşamaları

* Veriyi train test olarak 2’ ye bölmeliyiz.
  + Cross validation yapmayacaksak 3’ e bölmeliyiz.
* Modeli ve hiper parametreleri k-fold veri setleri ile optimize ediyoruz. Test veri setini kullanmıyoruz.
  + Cross validation yapmayacaksak validation veri seti ile optimize ediyoruz.
* Veri k-fold için uygun büyüklükte ise k-fold yapıyoruz. Veriyi standardize, normalize veya feature selection işlemleri için her bir k-fold için ayrı yapmalı bunun için pipeline kullanılabilir.
  + <https://machinelearningmastery.com/standardscaler-and-minmaxscaler-transforms-in-python/>
* K-fold dan elde ettiğimiz ortalama başarım ile hiç izlemediğimiz veri setinin başarımını karşılaştırılır. Başarımlar birbirine yakın ise modelin dışarıdan gelecek olan veriler içinde genelleştirildiğini varsayabiliriz.
* Cross validation’ a ihtiyacımız olup olmadığını anlamak için veriyi bir k-fold değeri ile eğitip çıkan başarımların birisine yakın olup olmadığı kontrol edilir. Başarımlar çok farklı ise cross validation lazım demektir.