

## Big Data (Volume,Velocity,Variety,Verification,Value) ozelliklere sahip olmalıdır.

Volume(Veri Hacmi) => Verilerimiz her geçen gün hızına hız katarak artıyor, ileride bu veri yığınları ile nasıl başa çıkacağımız iyi düşünmemiz ve bu doğrultuda yapmamız gerekmektedir.

Velocity(Veri Hızı) => Big data üretimi her geçen gün hızına hız katmakta ve bu veriler saniyede inanılmaz boyutlara ulaşmaktadır.

Variety(Veri Çeşitliliği) => Verilerin belirli bir yapısı yoktur, genellikle degışkendir. Resimler , ses dosyaları , text dosyaları örnek verilebilir.

Verification(Dogrulama)=> Veriler içerisinde anlamsız kayıtlar olabilir. Anlamsız kayıtlar analizlerimizin sonuçlarını etkilediği için bu kayıtları temizlemememiz gerekir.

Value(Degerli Veri)=> Büyük verinin üretimi ve işlenmesi katmanlarında elde edilen verilerin şirketimiz için artı deger sağlıyor olması gerekiyor.

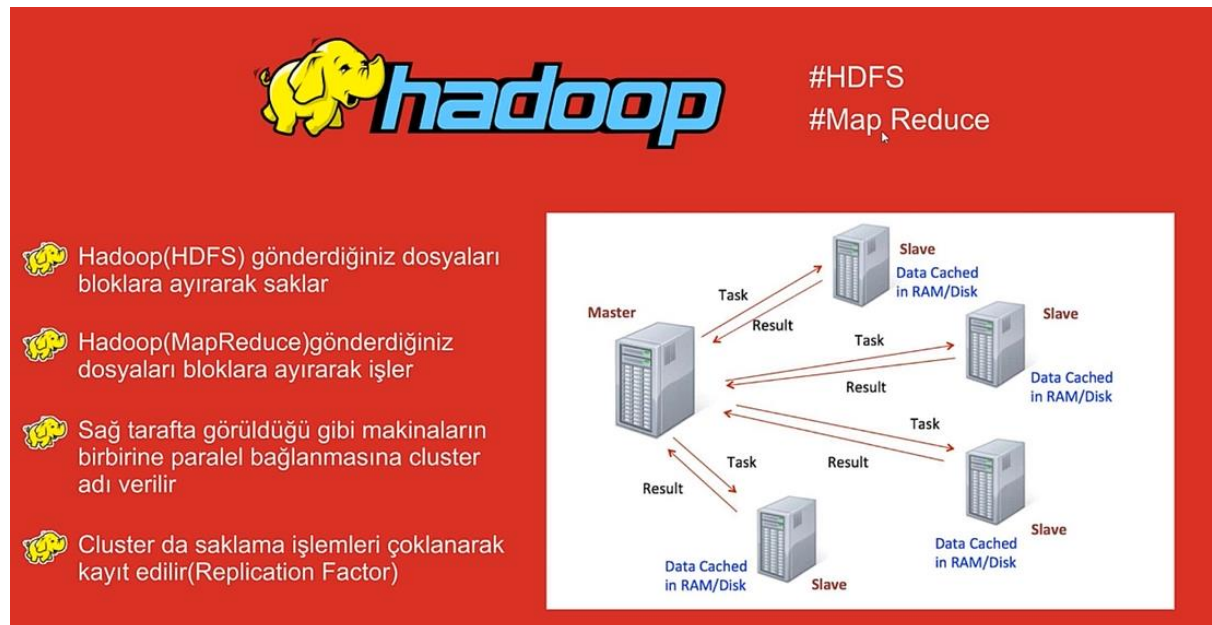
### Big Data Verileri Bu Kadar Hızlı Nasıl İşliyor?

- Dağıtık Sistem

## Hadoop

HDFS => Verileri Dağıtık Şekilde Tutar.

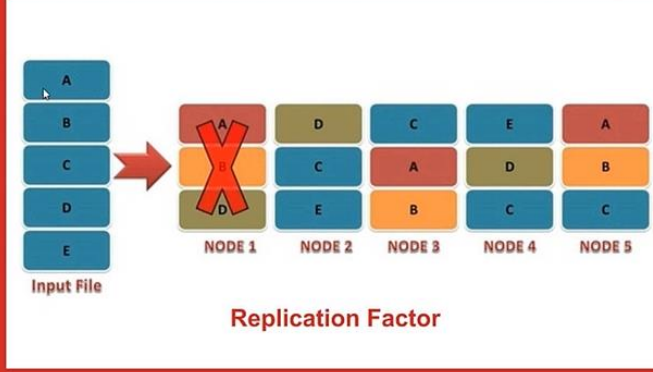
MapReduce => Bu verileri işler. Yani Veriler üzerinde bulma, filtreleme, sayma gibi veriler üzerinde işleme yapar.



Mesela 1-25 arası Slave1 , 26-50 Slave2 diye Node lara ayırır. Aynı Zamanda Yukarıda sistem. Cluster olarak da bilinir.



#HDFS  
#Map Reduce



Hadoop(HDFS) dosyaların kopyasını farklı makinalarda dağıtır

Bu işlemin bize sağladığı avantaj nedir ?

A-B-C-D-E 5 tane input file var ve 5 tane Node oluşturuluyor.

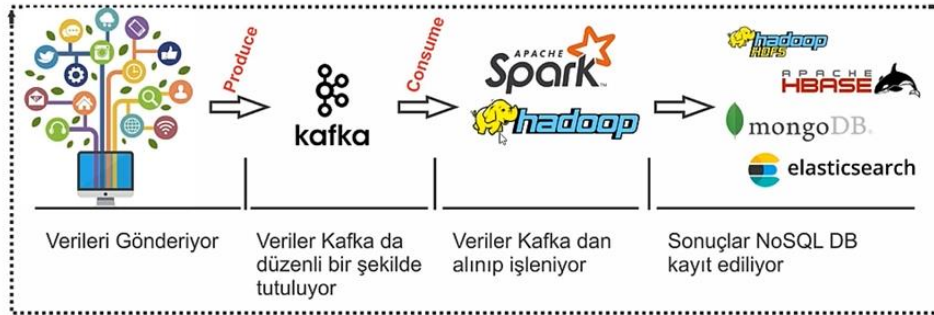
Burada Herhangi bir node sıkıntı yaşadığında diğer node kullanılır.

## Apache Kafka



Büyük verileri etkin bir şekilde kullanabilmek için iki önemli faktör karşımıza çıkar. Bunlar sırasıyla

Büyük verileri toplamak  
Büyük verileri analiz etmek  
Büyük veri bloklarını hatasız ve hızlı bir biçimde toplayıp, diğer sistemlere transfer edebilmek için bir mesajlaşma sistemine(queue) ihtiyacımız vardır.



Bu noktada Apache Kafka , akan verileri bir queue (mesaj kuyruğu) içerisine atarak; Hadoop, Spark, Elasticsearch gibi diğer sistemlere transfer etmemizi sağlar

Kafka; Kuyruk Sistemi ile verileri depolayıp herhangi bir kayıp olmadan tutuyor.

## NOSQL



### 1- NoSQL tanımı?

"NoSQL" terimi, ilişkisel olmayan veritabanı türlerini ifade eder ve bu veritabanları verileri ilişkisel tablolardan farklı bir formatta depolar. Ancak, NoSQL veritabanları deyimisel dil API'leri, tanımlayıcı yapılandırılmış sorgu dilleri ve örneğe göre sorgulama dilleri kullanılarak sorgulanabilir. Bu nedenle bunlara "sadece SQL" veritabanları değil" veritabanları değil" adı da verilir.

### 2- NoSQL veritabanı ne için kullanılır?

NoSQL veritabanlarının temel avantajı yüksek ölçeklenebilirlik ve yüksek erişilebilirlik olduğundan bu veritabanları gerçek zamanlı web uygulamalarında ve büyük veri alanında yaygın olarak kullanılmaktadır.

- NoSQL veritabanları, değişen gereksinimlere hızla uyum sağlayarak çevik bir geliştirme paradigmaya olanak tanıdığından geliştiricilerin de tercihidir.
- NoSQL veritabanları, verilerin daha sezgisel ve anlaşılması kolay veya verilerin uygulamalar tarafından kullanılma tarzına daha yakın bir şekilde depolanmasına olanak sağlar.
- NoSQL tarzı API'ler kullanıldığında verileri depolama veya alma sırasında daha az sayıda dönüşüm gerekir.
- Ayrıca, NoSQL veritabanları buluttan tam olarak yararlanarak sıfır kapalı kalma süresi sağlayabilir.

### 3- SQL ve NoSQL karşılaştırması?

- SQL veritabanları ilişkiselken NoSQL veritabanları ilişkisel değildir.
- İlişkisel veritabanı yönetim sistemi (RDBMS), kullanıcıların yüksek derecede yapılandırılmış tablolardaki verilere erişmesine ve verileri işlemesine olanak tanıyan yapılandırılmış sorgu dilinin (SQL) temelini oluşturur.
- Bu, MS SQL Server, IBM DB2, Oracle ve MySQL gibi veritabanı sistemleri için temel modeldir.

# NO SQL

Not Only


Document

Wide Column

Key Value

Graph DB

**Doküman (Document) tabanlı:** Document store tipinde ise veriler document(belge) şeklinde kaydedilir. (ElasticSearch , MongoDB) Mesela bir json verisi document olarak kabul edilebilir




## mongoDB.

Mongo DB verileri JSON tabanlı kaydeden açık kaynak kodlu ilişkisel olmayan bir veritabanıdır

```
{  "first_names": "Paul",  "surname": "Miller",  "oull": 44755795611,  "city": "London",  "location": [45.123,47.232],  "Profession": ["banking", "finance", "trader"],  "cars": [    { model: "Bentley",      year: 1973,      value: 100000, - },    { model: "Rolls Royce",  year: 1965,      value: 330000, - }  ]}
```

Fields → String → Typed field values  
Fields → Number →  
Fields → Geo-Coordinates →  
Fields → array → Fields can contain arrays  
Fields → array of objects → Fields can contain an array of sub-documents



## elasticsearch

Metin arama işlemlerinde kullanılır

Full-text Search Index

blue  
pants  
red  
sweater

Blue sweater  
Red sweater  
Red pants

## Wide Column Tabanlı NoSQL


# NOSQL

Not Only

Document   **Wide Column**   Key Value   Graph DB


**Wide Column Store tabanlı:**  
(Hadoop, HBase, Cassandra)

Buradaki yapı klasik veritabanına benzer. Tablolar, kolonlar ve satırlar vardır. En büyük farkı ise yapıyı belirlerken süper column olacak alanları belirleriz ve bu alanların altına her kayıt için farklı bir kolon gelebilir



Açık kaynak kodlu Apache hadoop projesinin bir alt projesidir.

Row Key	Customers		Products	
Customer ID	Customer Name	City & Country	Product Name	Price
1	Sam Smith	California, US	Mike	\$500
2	Arijit Singh	Goa, India	Speakers	\$1000
3	Ellie Goulding	London, UK	Headphones	\$800
4	Wiz Khalifa	North Dakota, US	Guitar	\$2500



HDFS sayesinde sıradan sunucuların diskleri bir araya gelerek büyük, tek bir sanal disk oluştururlar. Bu sayede çok büyük boyutta bir çok dosya bu dosya sisteminde saklanabilir.

## Key Value Tabanlı NoSQL

# NOSQL

Not Only

Document   Wide Column   **Key Value**   Graph DB

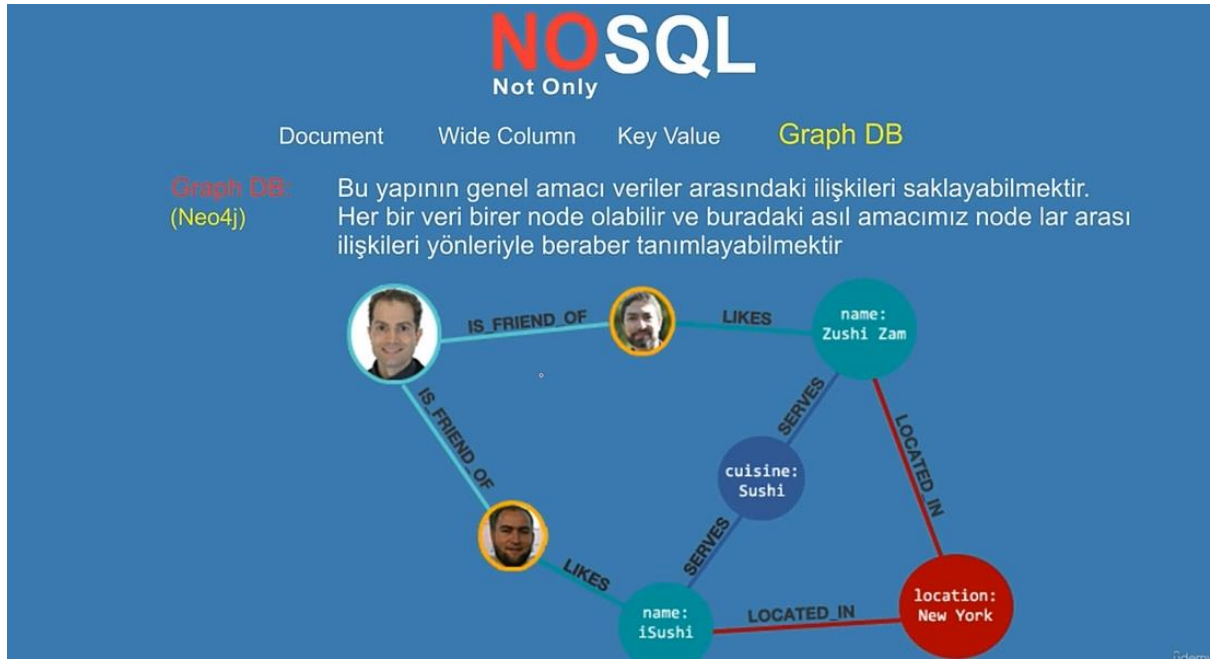
**Key Value Tabanlı:**  
(Redis, DynamoDB)

Bu yapıda her bir key karşılığında bir array yada veri yapısı bulunur. Amacımız key ler üzerinden hızlı bir şekilde verilere ulaşabilmektir

Key	Value
6515551234	Jon Doe, Pre-Paid, 40.00
AAA999	Mazda, Black, 626
321-651A	Random Data



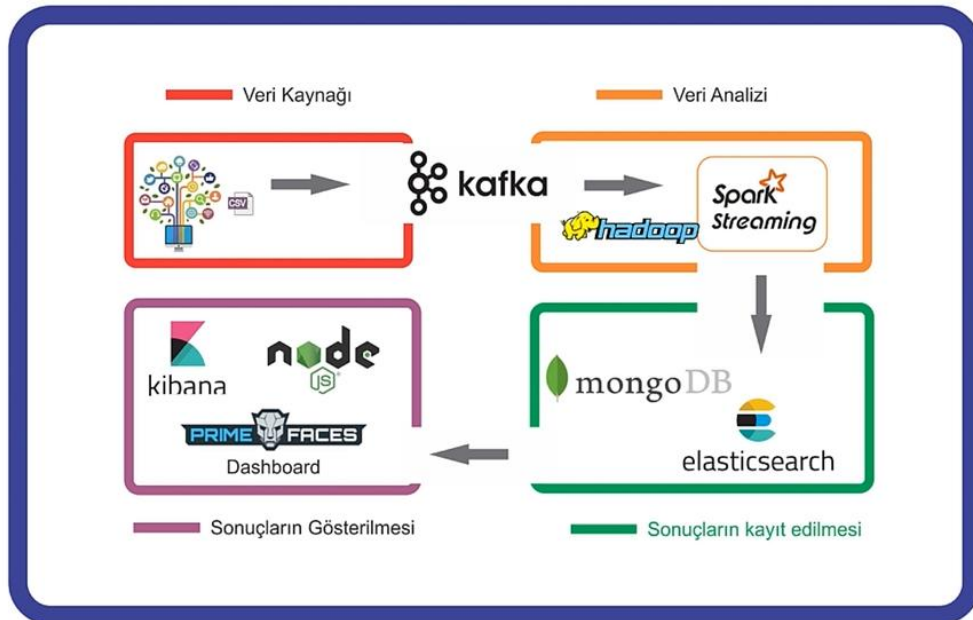
## Graph DB Tabanlı NoSQL



Graph; Verileri Gorsel hale getirmek için kullanılır.

## PROJE YONETİM MİMARİSİ

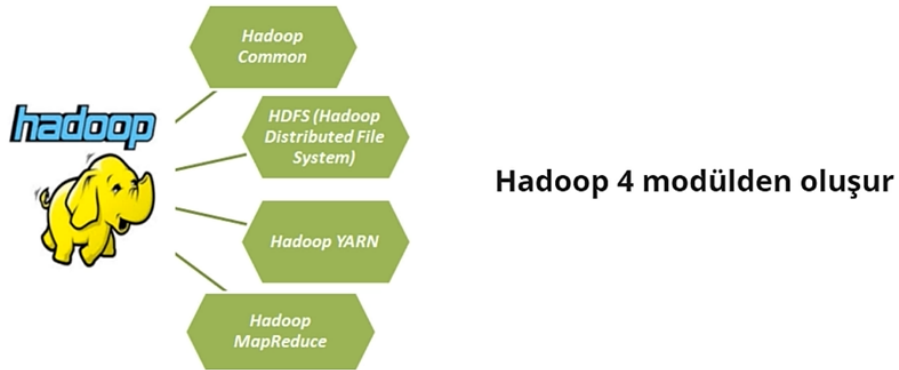
### Proje Yönetim Mimarisi



## HADOOP VE TEMEL BİLEŞENLERİ

### Hadoop ve Temel Bileşenleri

Hadoop, büyük veri kümeleri ile birden fazla makineda paralel olarak işlem yapmamızı sağlayan Java ile geliştirilmiş açık kaynak kodlu kütüphanedir



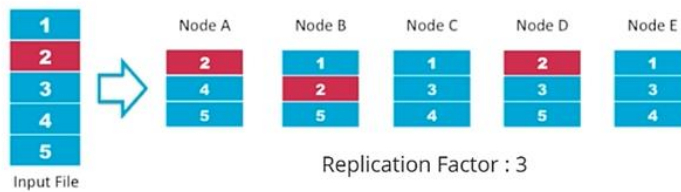
#### HDFS



Hadoop içerisinde büyük verileri sakladığımız bileşene HDFS denir.

**Hdfs verileri saklarken bloklara ayırarak saklar.**

HDFS Data Distribution



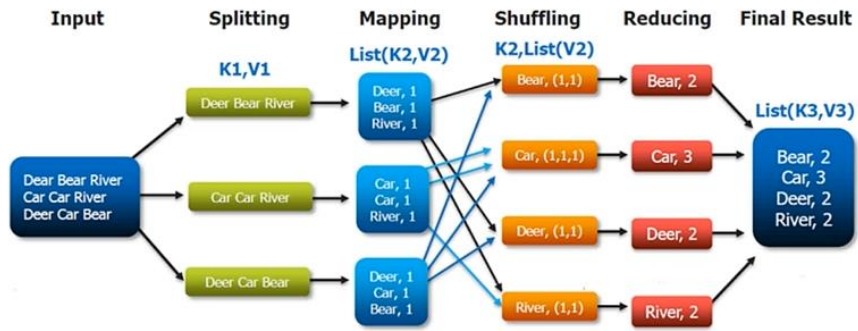
Replication Factor 3 olmasının nedeni. Mesela 2 kısmı; NodeA, NodeB ve NodeD olmasından dolayı

## MapReduce



Hadoop içerisinde büyük verileri paralel olarak işleyebileğimiz bileşene MapReduce denir

The Overall MapReduce Word Count Process



## Yarn



YARN genel olarak MapReduce gibi dağıtık uygulamalarımız için kaynak yönetimini sağlar(ram,cpu).





## MAPREDUCE YAPAN TEKNOLOJİLER

### Apache Pig



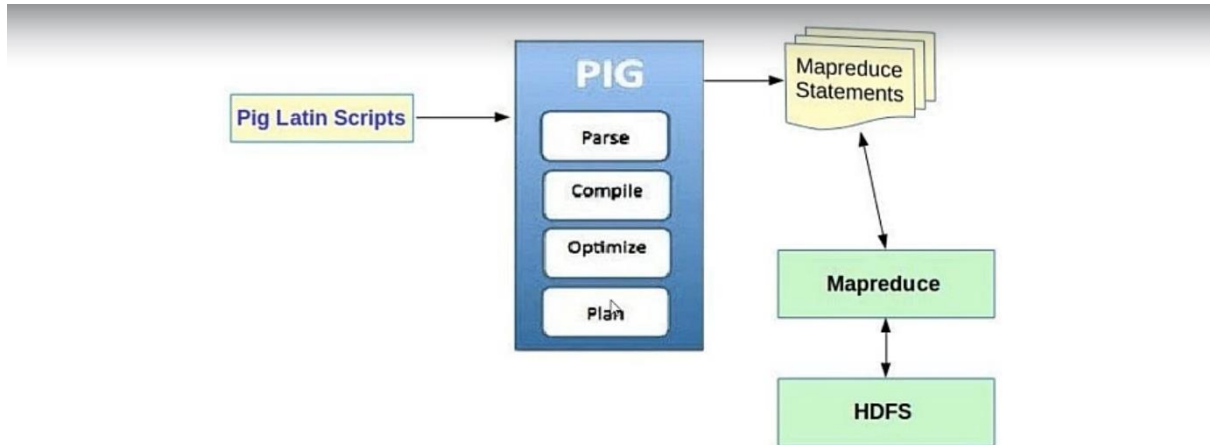
# Apache Pig

Büyük verileri **MapReduce** ile analiz edebiliriz.

MapReduce geliştirme yöntemleri;

- Java,Python,Scala MapReduce
- **Apache Pig**
- Apache Hive

Java , Python ve Scala da MapReduce yazmak zordur. Ondan dolayı Apache Pig kullanılması önerilir.



**Parse :** Syntax kontrolü yapılır.Yazım yanlışı varsa hata verir

**Compile :** Yazmış olduğumuz kodlar MapReduce'a çevrilir

**Optimize-Plan :** Kodların optimizasyonu Pig tarafından yapılır.Kodların daha performanslı çalışması için Pig Optimizasyonlar yapar

## Apache Pig Avantajları ;

- Öğrenmesi ve geliştirmesi basittir
- Büyük veriler üzerinde kolaylıkla analizler yapılabilir
- Yazdığımız kodları optimize eder
- Veri üzerinde analizler yapabileceğimiz metodlar sunar(filter,join vs)
- İhtiyaç halinde javascript,java yada python ile kütüphaneler yazıp apache pig içerisinde kullanabiliriz(udf)

### Apache Hive



## Apache Hive Nedir ?

Büyük Veriler ile Hadoop üzerinde işlemek yapmak için kullanılan bir MapReduce geliştirme yöntemidir

Facebook tarafından geliştirilen açık kaynak bir kütüphanedir

## Apache Hive hangi durumlarda tercih etmeliyim?

Hive ile büyük veriler üzerinde sql sorguları ile basit analizler yapmak için kullanılır.

## Hive MetaStore Nedir ?

Şema bilgileri, tablolar, kolonlar, kolon tipleri gibi bilgilerin saklandığı bölümdür.

Metadata: hive.default.orders

	COLUMN_NAME	DATA_TYPE
1	order_id	BIGINT
2	month	VARCHAR
3	cust_id	BIGINT
4	state	VARCHAR
5	prod_id	BIGINT

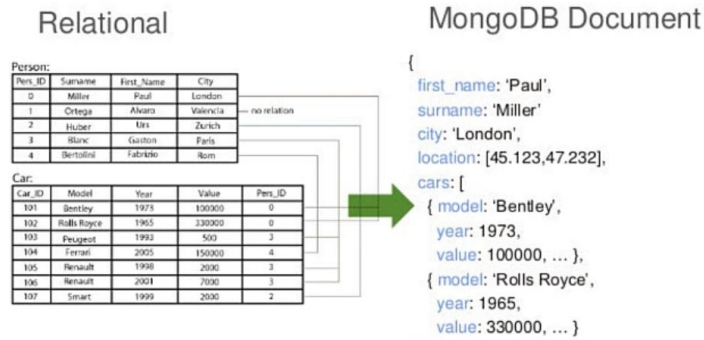
- Hadoop : Büyük verileri Dağıtık (HDFS) şeklinde Tutulur. Hem Depolama yap hemde Veri işleme yapar.
- Spark : Sadece Veri işleme yapar. Hadoop daha hızlıdır. Çünkü veri saklama birimi olmadığı için Memory de (RAM üzerinde) çalışır.
- MapReduce: Bu verilerde işleme yapar. Analiz Yapar. MapReduce En kolay kullanım ise Apache Pig ve Apache Hive dir.

- Metadata : DataNode larda (Veri analizi yapan makinelerde) saklanan veriler hakkında bilgilerdir.

## Mongo Db



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



MongoDB'ye ait kavramlarla,RDBMS sistemlerde var olan SQL'e ait kavramların eşleşme tablosuna bakmak faydalı olacaktır

### Terminology

RDBMS		MongoDB
Table, View	→	Collection
Row	→	Document
Index	→	Index
Join	→	Embedded Document
Foreign Key	→	Reference
Partition	→	Shard

# Neden MongoDB ?

**Ölçeklenebilirdir(Scalable)** : Veri boyutu arttığı durumlarda veya performans sıkıntısı yaşadığımız durumlarda makine ekleyebiliriz

Veriler document (belge) biçiminde saklanır. Burada JSON verilerini kullanabiliriz

Verilerin birden fazla kopyası saklanabilir ve veri kaybı yaşanmaz (Replication)

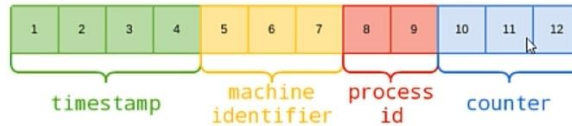
Veri boyutu arttığı durumlarda yada tam tersi veri boyutu azalırsa veri ekleme veya veri çıkarma işlemi makine ekleme veya çıkarma yaparak rahat yapabiliyoruz. Bir sistem kurduk 100mb dan 1TB data ihtiyacımız oldu. Burada sistem çökmeden veri arttıkça yada veri azaldıkça Makine ekleme ve silme yapılmaya izin veriyor. Cluster yani dağıtık işlem yapabiliyoruz. Verilerimiz bu esnada kaybolmuyor.

## Id Bilgisi

MongoDB üzerinde bir kayıt insert edilirken otomatik olarak \_id isimli bir alan eklenir . Bu alan kullanıcı tarafından girilmezse , tekil (unique) bir değer ile kaydedilir

```
{
  "_id" : ObjectId("57b477717edc8005e9ed7fb"),
  "ad" : "kullanıcı",
  "soyad" : "soyadı",
  "no" : 14,
  "sinif" : "altsınıf"
}
```

## Id Bilgisinin Oluşması





Buyuk Veriler arasında metin arama (full text search) işlemlerinde kullanılan NoSQL teknolojisidir.

ElasticSearch altyapısında Apache Lucene ve Solr vardır

Elasticsearch bir kelimenin hangi dökümanda(row) geçtiğini veriler **kaydedilirken** indexler . Sonrasında ise kelime aramak istediğimizde tüm veriler üzerinde arama yapmak yerine, daha önce oluşturulan index listesi üzerinden sonuçlar hızlıca bulunur

## Full-text Search 101: The inverted index

### User queries for "keeper"

1	The old night <b>keeper</b> keeps the keep in the town
2	In the big old house in the big old gown.
3	The house in the town had the big old keep
4	Where the old night <b>keeper</b> never did sleep.
5	The night <b>keeper</b> keeps the keep in the night
6	And keeps in the dark and sleeps in the light.



6 documents to index

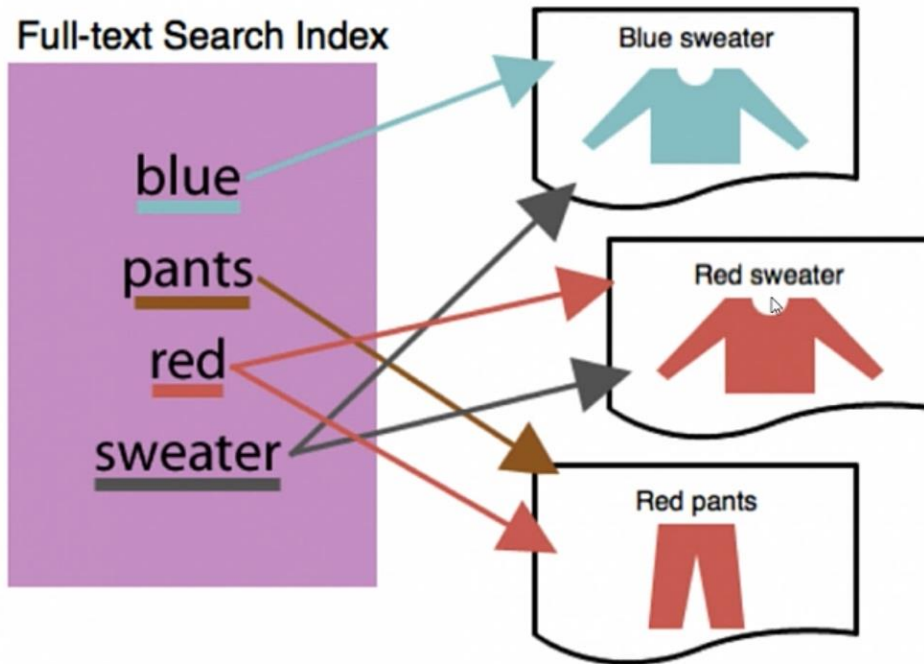
The index:

Dictionary and  
posting lists

Term	Documents
and	<6>
big	<2> <3>
dark	<6>
did	<4>
gown	<2>
had	<3>
house	<2> <3>
in	<1> <2> <3> <5> <6>
keep	<1> <3> <5>
<b>keeper</b>	<1> <4> <5>
keeps	<1> <5> <6>
light	<6>
never	<4>
night	<1> <4> <5>
old	<1> <2> <3> <4>
sleep	<4>
sleeps	<6>
the	<1> <2> <3> <4> <5> <6>
town	<1> <3>
where	<4>

ElasticSearch bunu kelime bazlı ayırma yapıyor. Kayıt ederken indexleme yapıyor.





## Terminology

### Relation Databases

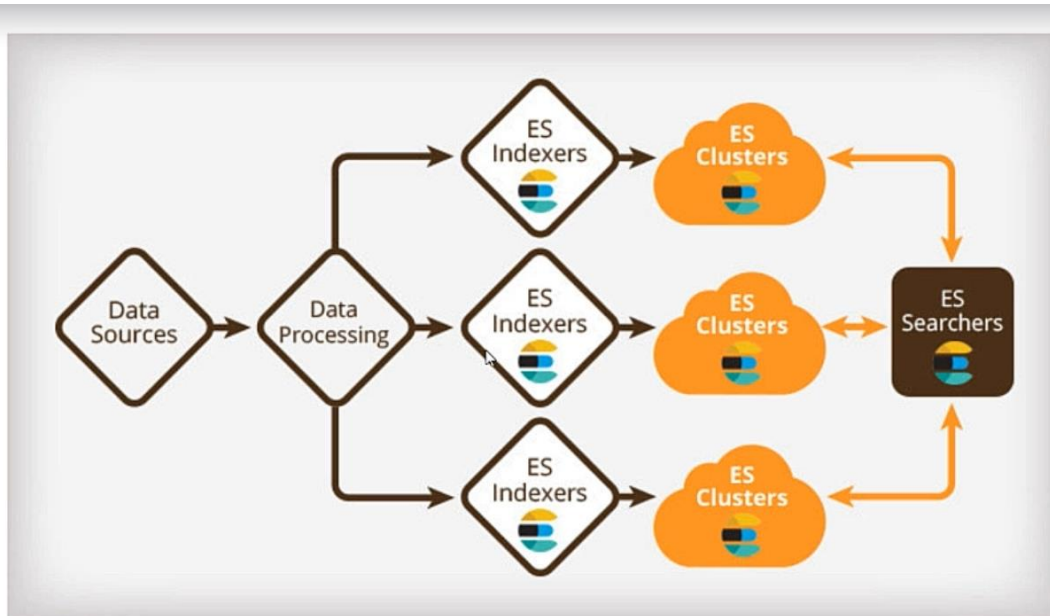
- Database
- Table
- Row
- Column
- Schema



### Elasticsearch

- Index
- Type
- Document
- Fields
- Mapping





ElasticSearch'de diğer Big Data teknolojilerinde olduğu gibi **Cluster** yapısı vardır.

Cluster yapısı olduğundan Replicationda var. Kopyalamada var. Aynı veriler birden fazla makinelerde kopyalanarak saklanıyor ki veri kaybı yaşanmasın. Başka bir makine çöktüğünde diğerinden telafi edebilmesi için. Cluster yapısı olduğundan Ölçeklenebilirlik vardır. Makine eklemek veya çıkarmak kolaydır.

## Documents as JSON

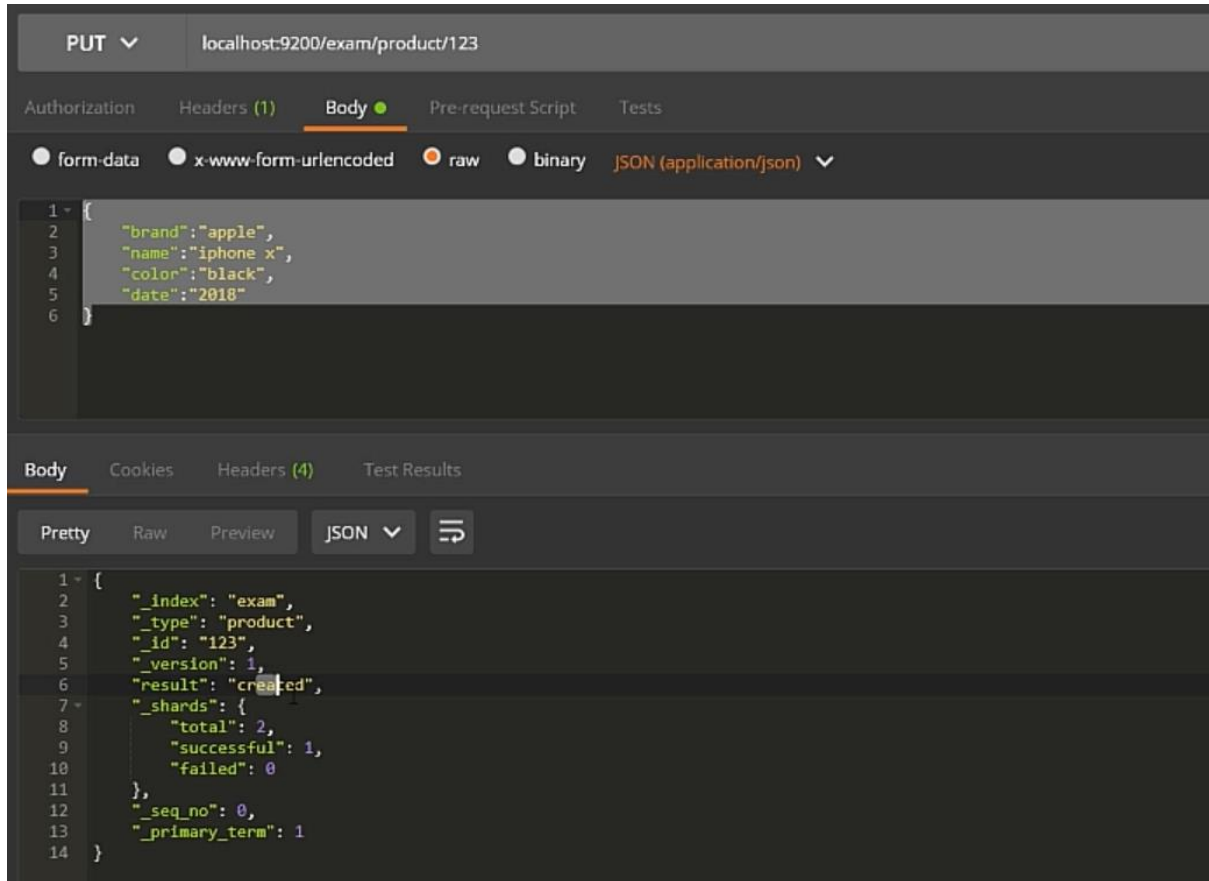
Data structure with basic types, arrays and deep hierarchies

```
{
  "id"       : "abc123",
  "title"    : "A JSON Document",
  "body"     : "A JSON document is a ...",
  "published_on" : "2013/06/27 10:00:00",
  "featured"  : true,
  "tags"     : ["search", "json"],
  "author"   : {
    "first_name" : "Clara",
    "last_name"  : "Rice",
    "email"      : "clara@rice.org"
  }
}
```

Default olarak 9200 portunda hizmet veriyor.

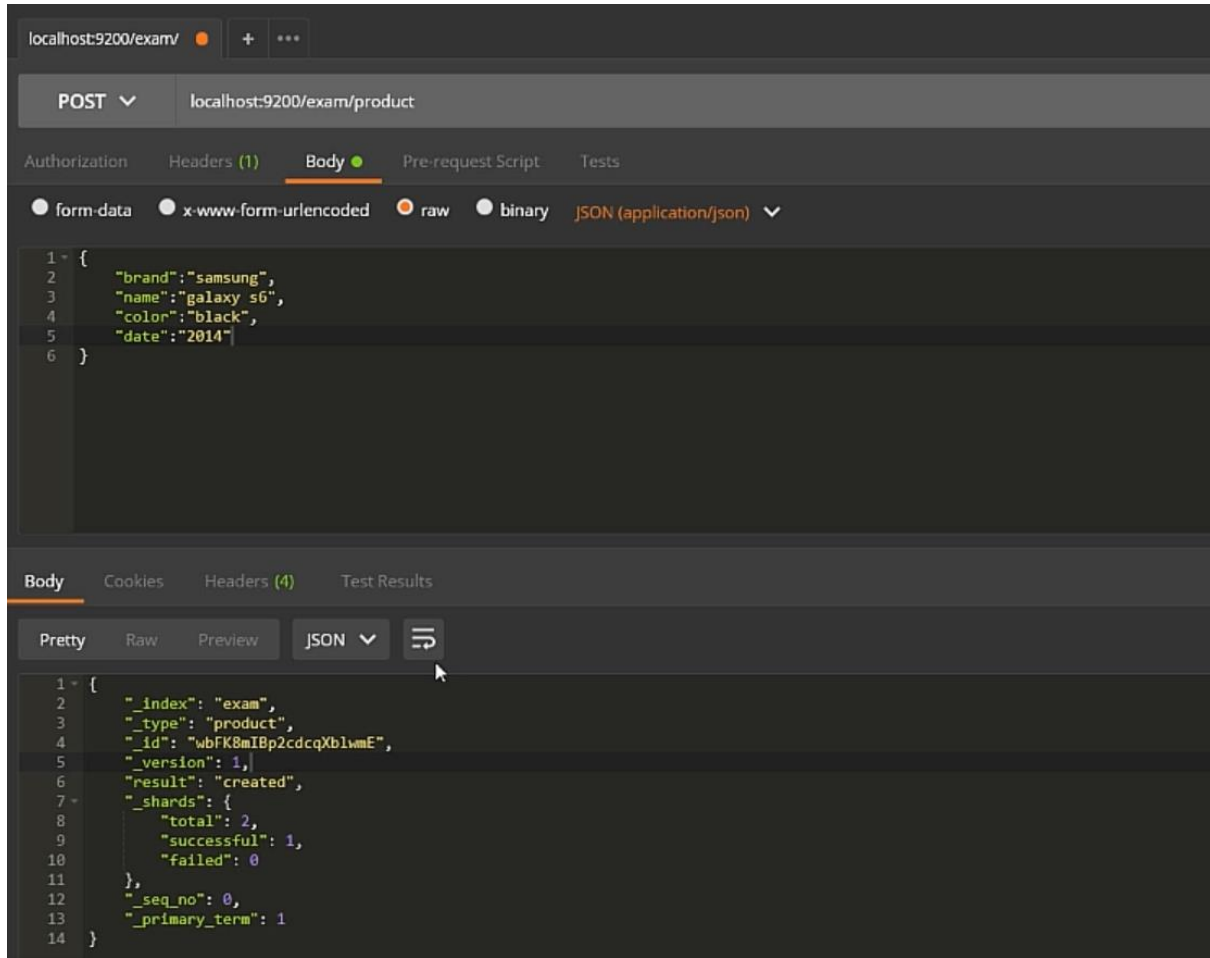
## CRUD işlemleri

### Put ve id ile ekleme yapma



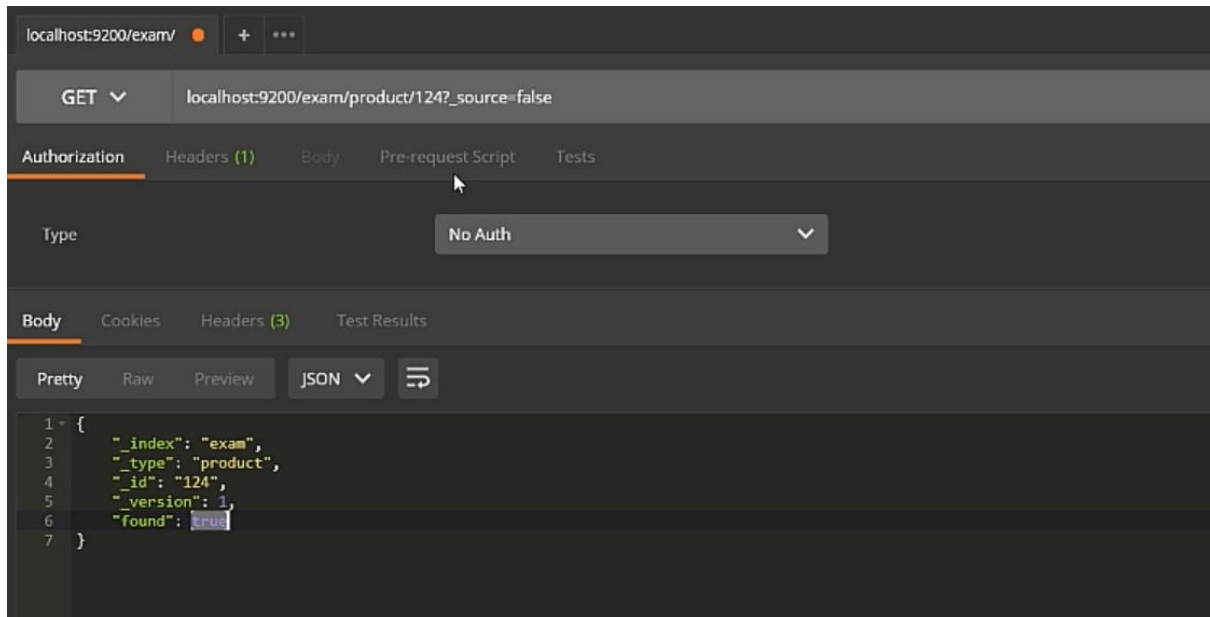
Exam VeriTabanına Product Tablosuna 123 numaralı id ile ilgili bilgi ekleme yaptık.

## Post ile id siz ekleme yapma



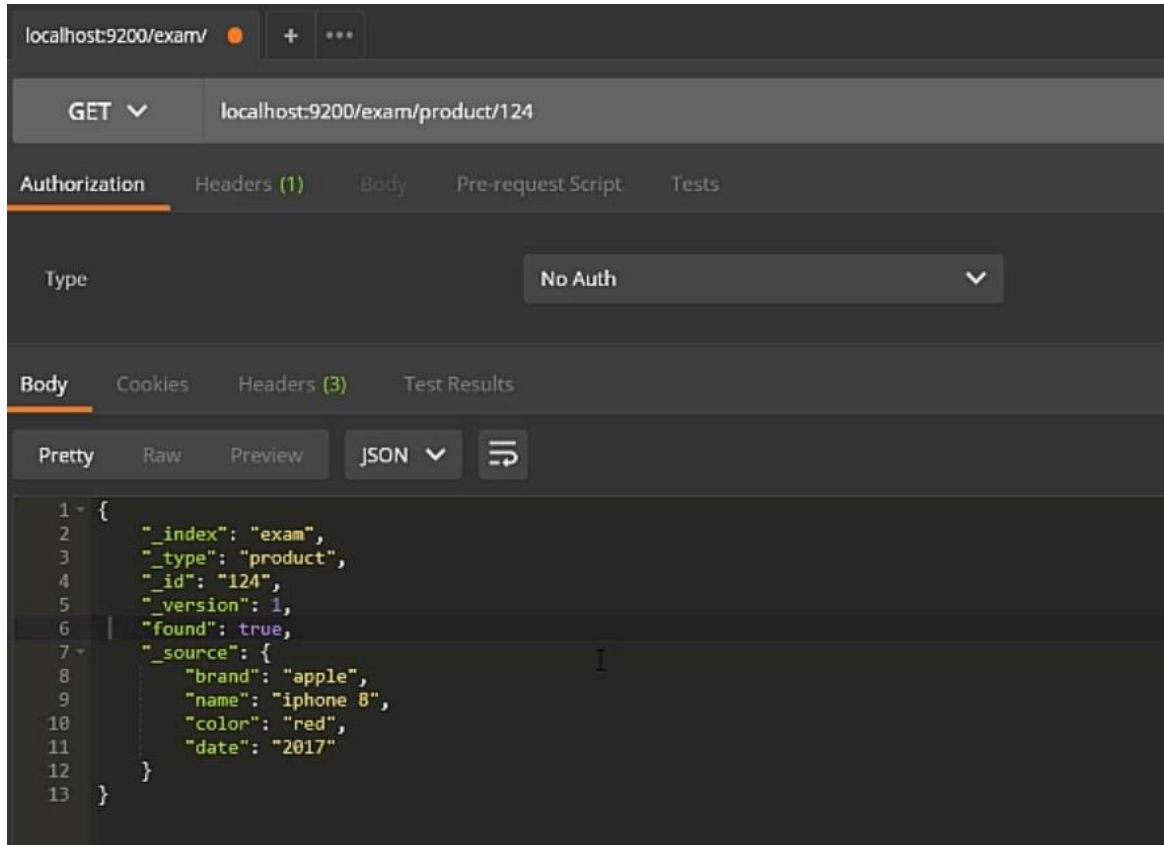
Post ile attığımız zaman id otomatik kendisi belirler.

## Kaynak Getirmeden sorgu var mı yok mu sorma?



Found alanın true olarak var olduğunu dönmüş oldu.

## Get ve id ile Ürün Sorgulama

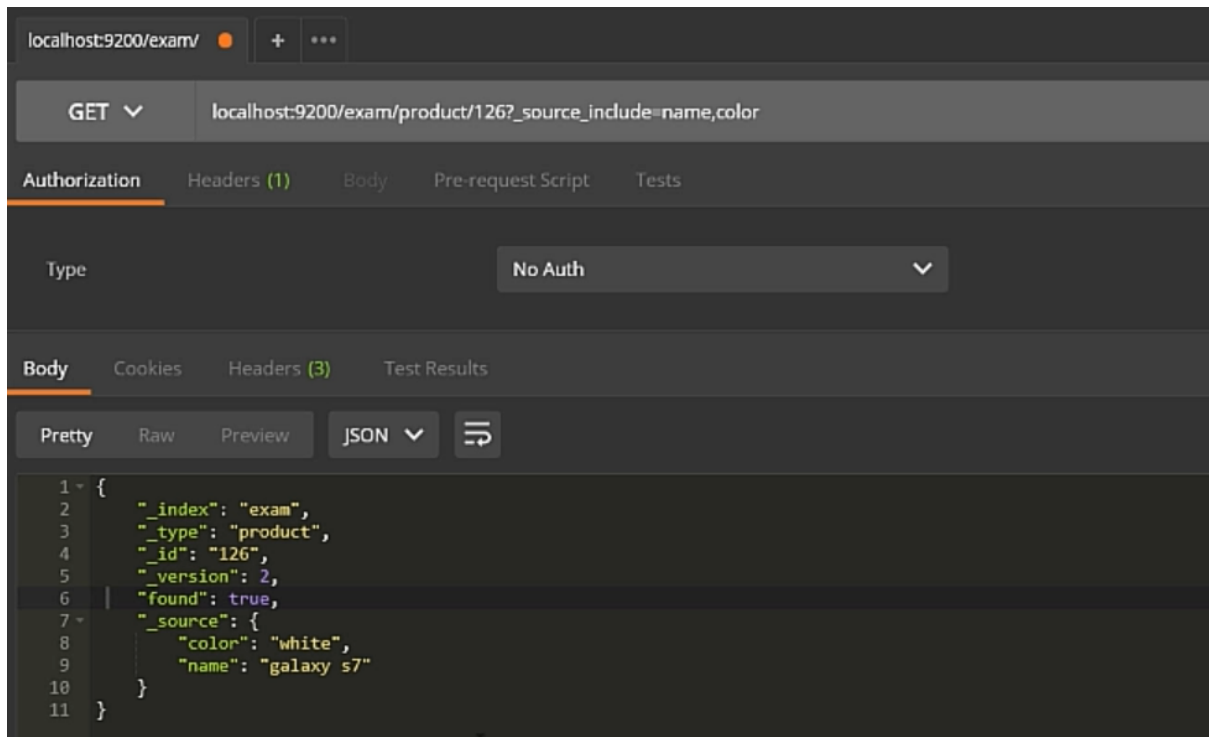


The screenshot shows a REST client interface with the following details:

- URL: `localhost:9200/exam/product/124`
- Method: `GET`
- Authorization: `No Auth`
- Body: `JSON` (Pretty)
- Response (JSON):

```
1 {
2   "_index": "exam",
3   "_type": "product",
4   "_id": "124",
5   "_version": 1,
6   "found": true,
7   "_source": {
8     "brand": "apple",
9     "name": "iphone 8",
10    "color": "red",
11    "date": "2017"
12  }
13 }
```

## Get ile Alan Filtreleme yaparak hepsini degilde belirli alanları getirme



The screenshot shows a REST client interface with the following details:

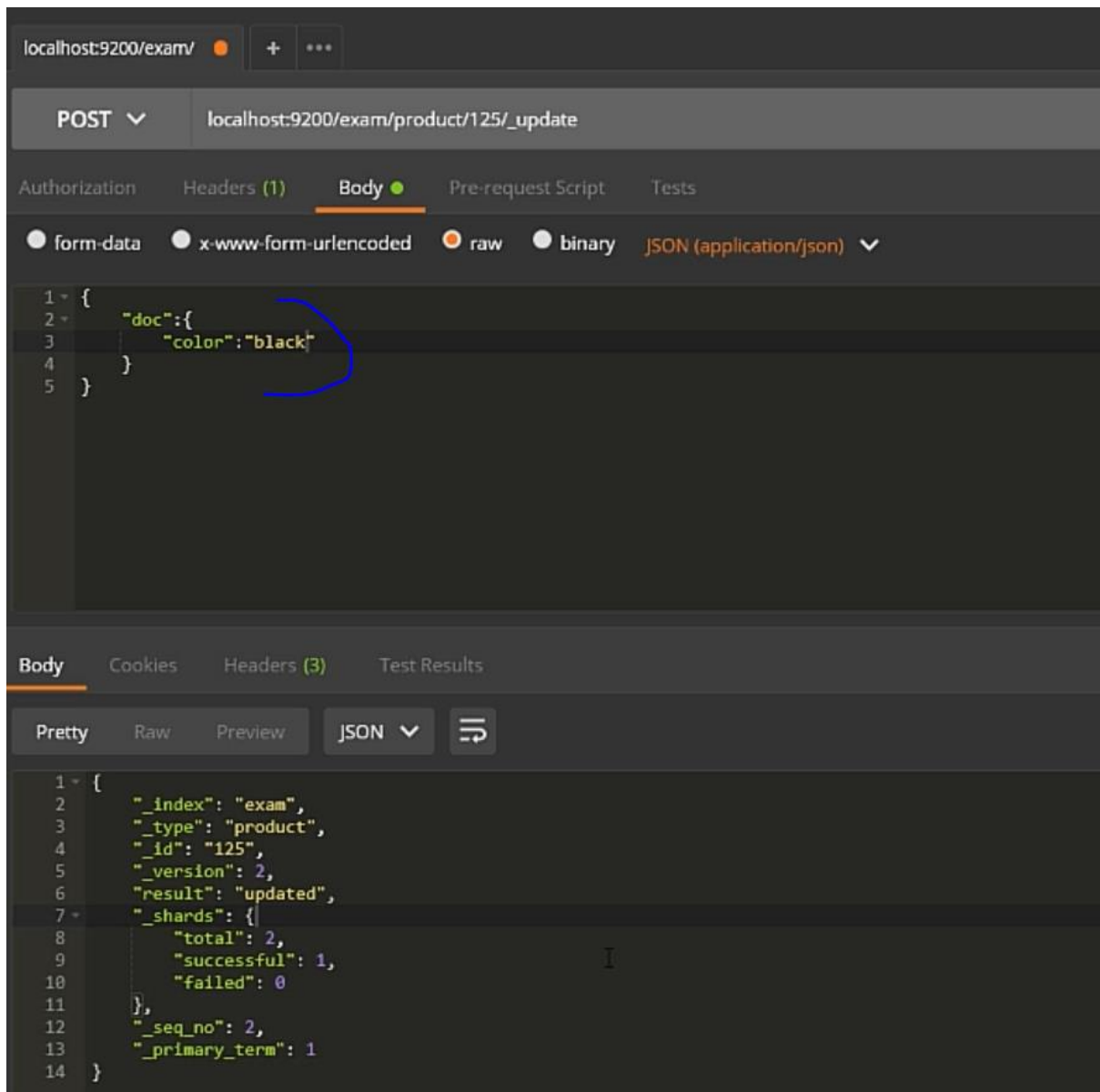
- URL: `localhost:9200/exam/product/126?_source_include=name,color`
- Method: `GET`
- Authorization: `No Auth`
- Body: `JSON` (Pretty)
- Response (JSON):

```
1 {
2   "_index": "exam",
3   "_type": "product",
4   "_id": "126",
5   "_version": 2,
6   "found": true,
7   "_source": {
8     "color": "white",
9     "name": "galaxy s7"
10  }
11 }
```

\_source altında include yazmak gerekiyor.

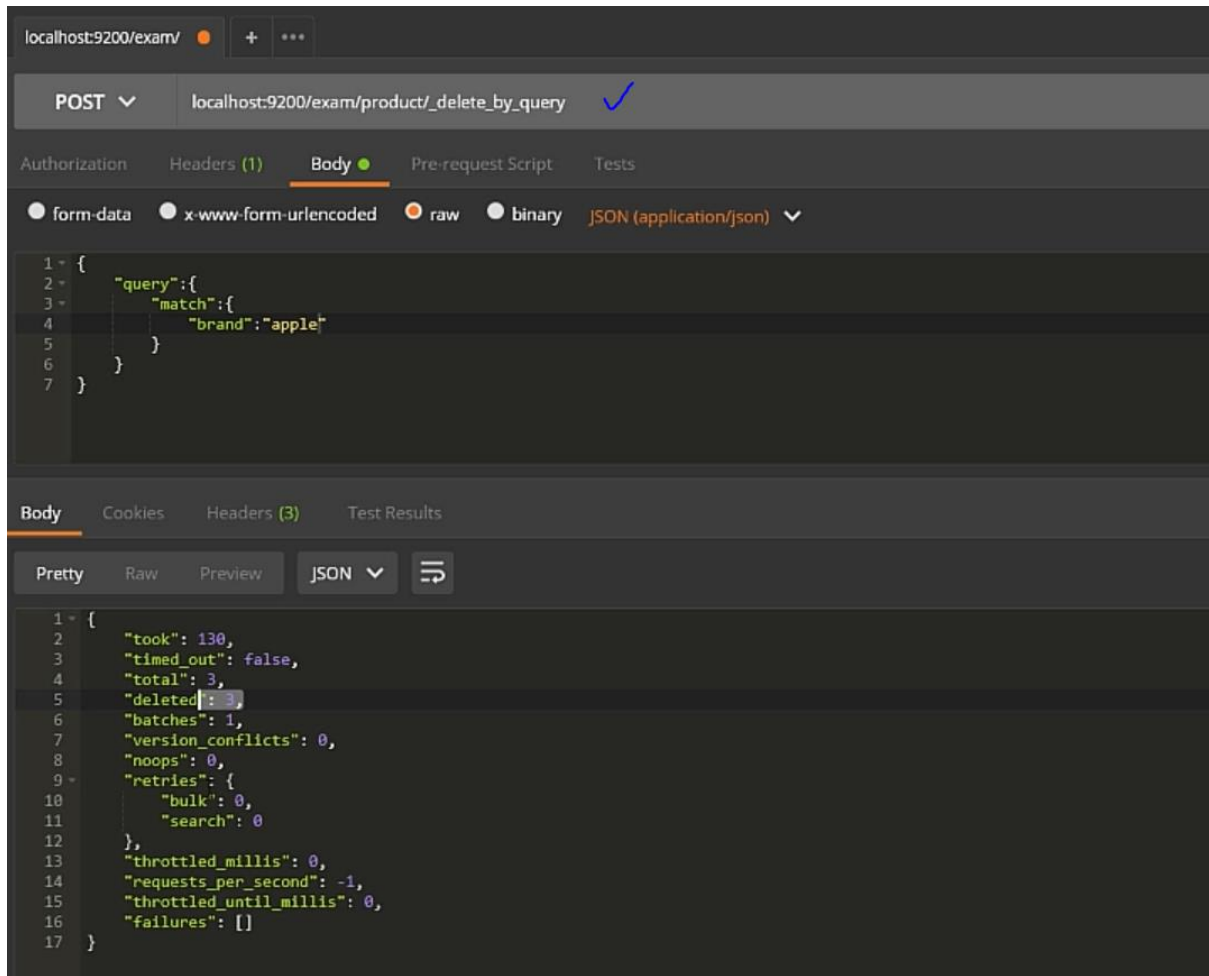


Post ile Güncelleme yapmak istediğimiz alan için



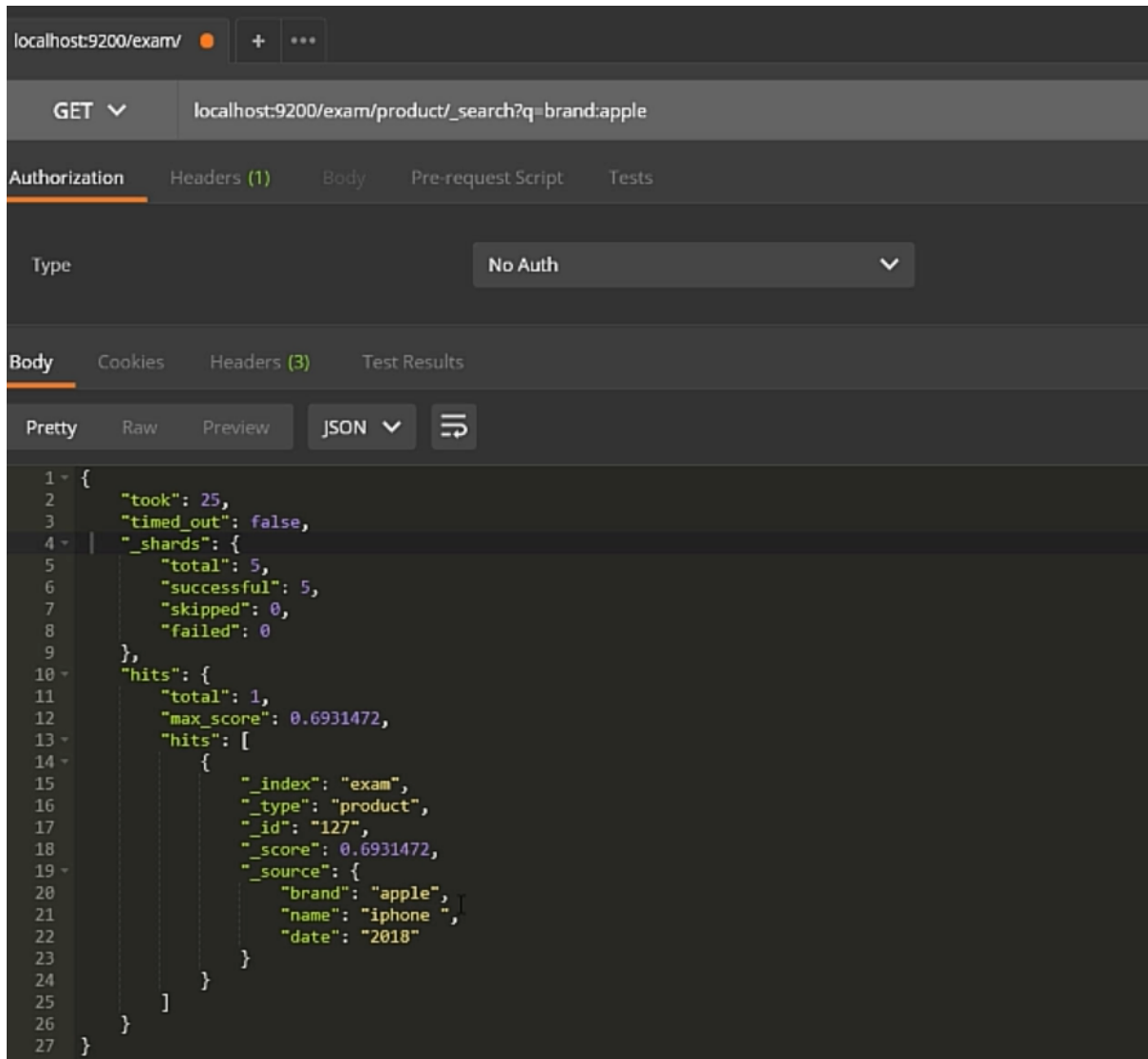
doc syntax altında yapmak gerekiyor.

Post ile Tüm alanlarda marka yani brand alanını silmek istiyorsak.



query-match syntax kullanılıyor. Query olarak `/_delete_by_query` diyerek query bazlı silmek istediğimizi söylüyoruz.

Get ile Query bazlı arama yapmak için. Brand alanında Apple olan marka getir demek istediğimizde.



```
localhost:9200/exam/v  + ...

GET  localhost:9200/exam/product/_search?q=brand:apple

Authorization  Headers (1)  Body  Pre-request Script  Tests

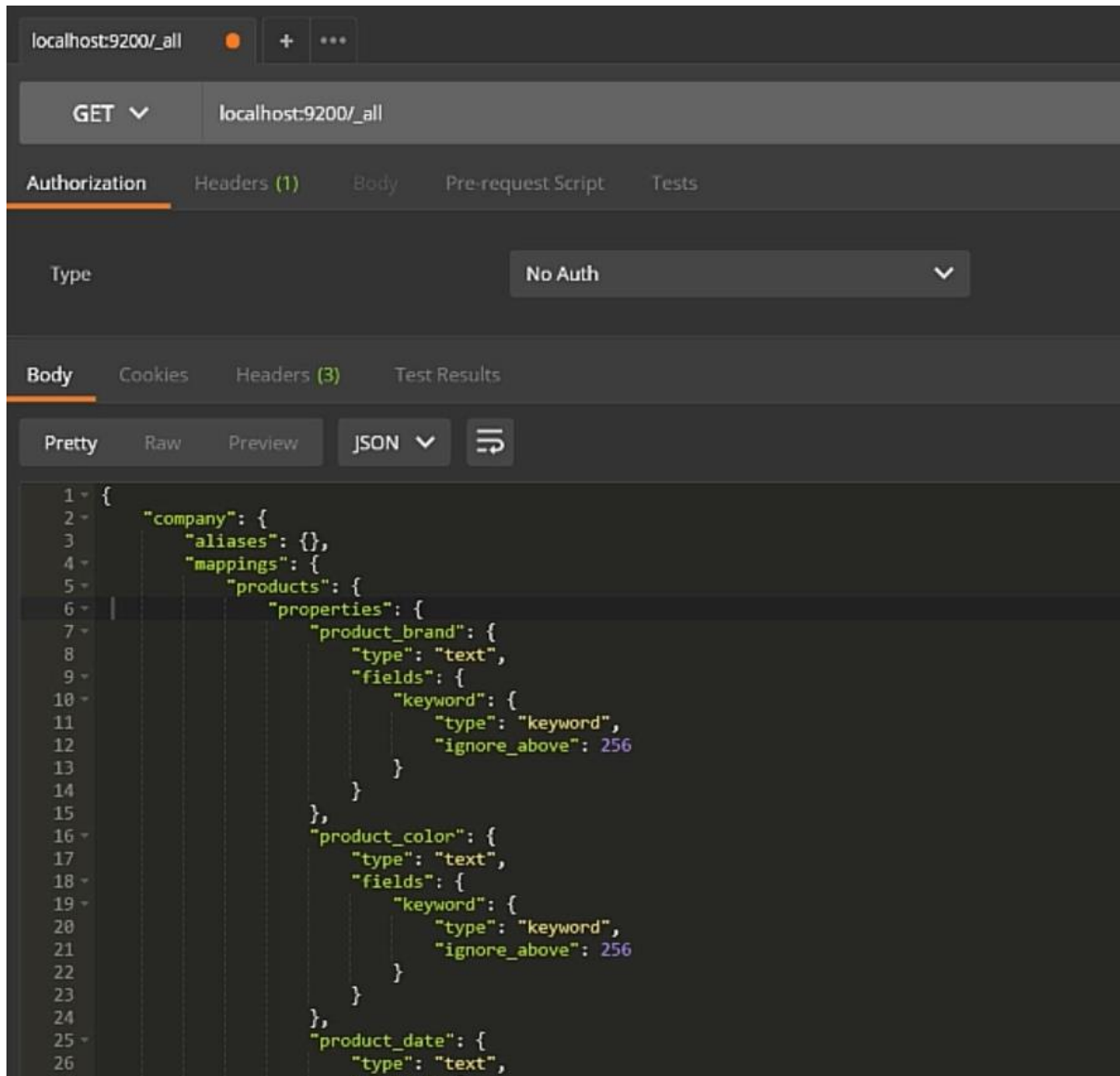
Type  No Auth  v

Body  Cookies  Headers (3)  Test Results

Pretty  Raw  Preview  JSON  v  ≡

1  {
2    "took": 25,
3    "timed_out": false,
4    "_shards": {
5      "total": 5,
6      "successful": 5,
7      "skipped": 0,
8      "failed": 0
9    },
10   "hits": {
11     "total": 1,
12     "max_score": 0.6931472,
13     "hits": [
14       {
15         "_index": "exam",
16         "_type": "product",
17         "_id": "127",
18         "_score": 0.6931472,
19         "_source": {
20           "brand": "apple",
21           "name": "iphone ",
22           "date": "2018"
23         }
24       }
25     ]
26   }
27 }
```

Veri tabanındaki tüm Bilgileri getirmek için.



The screenshot shows a REST client interface with the following details:

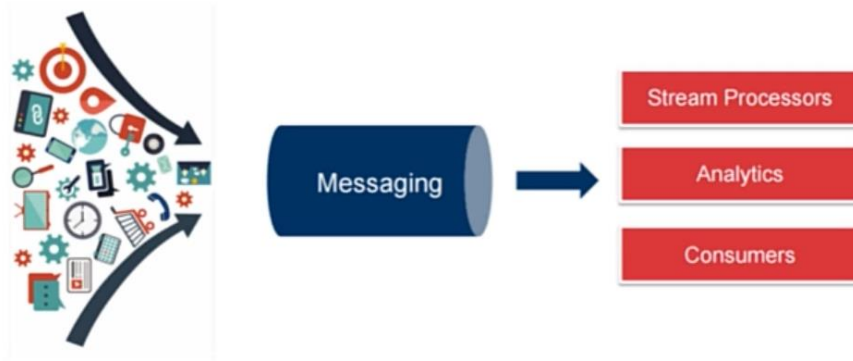
- URL:** localhost:9200/\_all
- Method:** GET
- Authorization:** No Auth
- Body:** Pretty (JSON)
- Response Body (JSON):**

```
1 {
2   "company": {
3     "aliases": {},
4     "mappings": {
5       "products": {
6         "properties": {
7           "product_brand": {
8             "type": "text",
9             "fields": {
10              "keyword": {
11                "type": "keyword",
12                "ignore_above": 256
13              }
14            }
15          },
16          "product_color": {
17            "type": "text",
18            "fields": {
19              "keyword": {
20                "type": "keyword",
21                "ignore_above": 256
22              }
23            }
24          },
25          "product_date": {
26            "type": "text",
```

## APACHE KAFKA



Büyük verileri hızlı ve hatasız bir şekilde toplamak ve analiz etmek çok önemlidir.

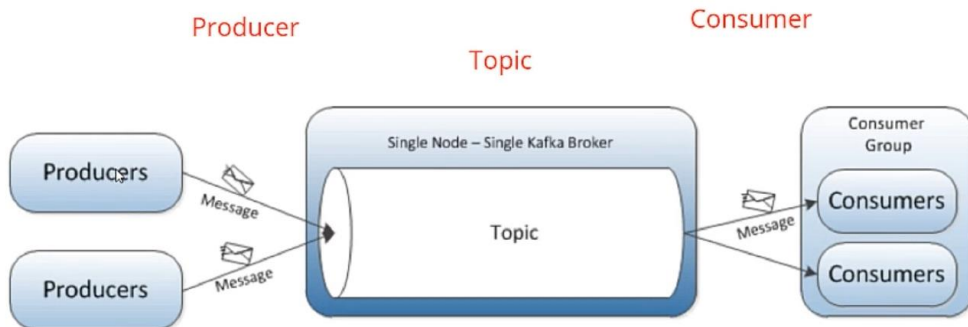


Kafka bir kuyruk mekanizmasıdır. Veriler kaynak tan analiz aşamasına degilde kafka ya yolluyoruz.

Kafka da veriler dağıtık şekilde tutuluyor. Kafka kuyruk mekaniması olduğu için FIFO kuyruktur.

İlk giren ilk çıkar. Veriler direk kaynaktan analiz kısma geçerse saniyelerde milyonlarca veri geldiğinde

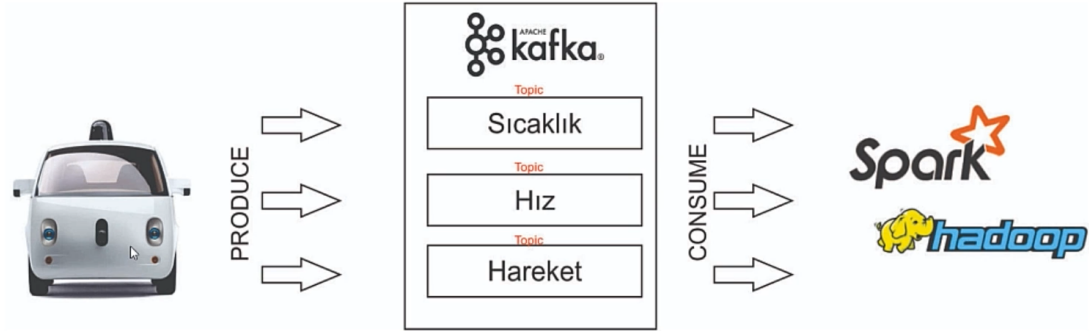
Verilerin kaybolma ihtimali var. Yada analiz makinesi çöktüğü durumlarda veriler geldiğinde boşuna gelmiş olmuş olacak ve yine verilerin uçma ihtimali var. Kafkada böyle bir şey yok. Analiz kısmı çökse bile kafka da veriler yedeklenir ve sonradan tekrardan gönderilme ihtimali vardır.



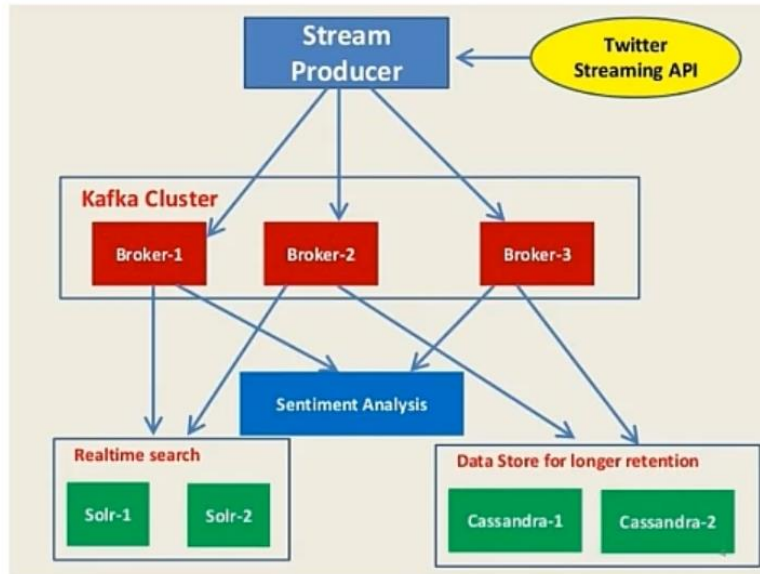


Verileri gönderen kısmı Producers. Topic de verileri belirli başlıklar altında toplama yapabiliyoruz.

Consumer verileri çeken kısımdır. Burada Producer istenilen topic yazma yapıp Consumer ise verileri istediği Consumer çekme yapabilir.



Kafka'da dağıtık veri depolama ve replication mevcuttur.

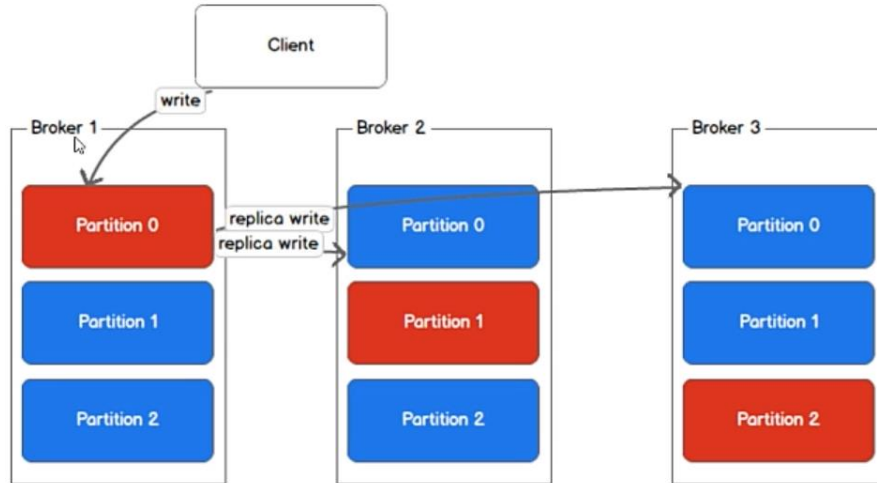


Stream den data geliyor. Her bir kafka server kısmına Broker adını veriyoruz. Yukarıda üç tane Broker dan oluşan Cluster oluşturulmuş. Rahatca ekleme veya eksiltme yapılabilir.

## Broker ile Replication Paylaşması



### Leader (red) and replicas (blue)



Her bir broker Partition-0, Partition-1, Partition-2 olmak üzere veri 3 bölüme ayırmış.

Her bir Brokerin lead farklıdır. Burada kırmızılar lead. Diğerlerinde görüldüğü gibi veriler yedikleme yapılmıştır. Burada Replication Faktör 3 kabul edilmiştir.



Zookeeper, dağıtık sunucu mimarilerinde kaynak yönetimini koordine eder.

Zookeeper, genel olarak konfigürasyon için kullanılır ve konfigürasyon dosyalarını tutar

## 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 analizlerini **Ram** üzerinde gerçekleştirir



VS



Dahili depolama birimi yok.  
Fakat entegrasyon sayesinde  
farklı teknolojilerde  
depolanabilir

Spark Hadoop'a göre 100 kat  
daha hızlı olduğunu savunuyor

Spark Rdd ve Sql sayesinde  
yüksek seviyeli operatörler  
olduğu için programlaması  
kolaydır

**Depolama**

HDFS

**Hız**

Diskten okuma ve yazma yaptığı  
için Spark'a göre yavaştır

**Zorluk**

MapReduce geliştirmek zordur.  
Bundan dolayı Pig ve Hive  
kullanılıyor

Genel olarak Hadoop ta veriler tutulur Apache Spark da ise hızından dolayı veri analizi yapar



VS



Spark kendi yönetimini  
sağlamaktadır

**Yönetim**

YARN'a ihtiyaç duyar

Spark Streaming ile saniyede  
milyonlarca veriyi anlık canlı  
olarak analiz edilebilir

**Gerçek  
Zamanlı  
Analiz**

Gerçek zamanlı analiz aracı  
yoktur