

## **HADOOP:**

### **1. Hadoop Common**

Hadoop Common, Hadoop ekosisteminin temelini oluşturan kütüphaneleri ve araçları içerir. Bu kütüphaneler, tüm Hadoop modüllerinin üzerinde çalışmasını sağlayan temel kod yapısını sağlar. Örneğin, dosya sistemine erişim, sıkıştırma, ve diğer yardımcı araçları içeren kütüphaneler burada bulunur. Common sayesinde tüm Hadoop bileşenleri birbirleriyle uyum içinde çalışır.

### **2. HDFS (Hadoop Distributed File System)**

HDFS, büyük veri kümelerini saklamak için kullanılan dağıtık bir dosya sistemidir. Verileri bloklara ayırarak birden fazla düğüme (node) dağıtır. HDFS, büyük miktarda veriyi birçok sunucuya dağıtarak hem verinin daha güvenli olmasını sağlar, hem de erişimi hızlandırır. Verinin kopyalanarak saklanması sayesinde, herhangi bir düğümde hata meydana gelse bile veri kaybı yaşanmaz, çünkü diğer düğümlerde verinin yedekleri bulunur.

#### **Öne Çıkan Özellikleri:**

- **Dağıtık Saklama:** Veriler bloklara ayrılır ve birden fazla düğümde saklanır.
- **Kopyalama (Replication):** Veriler farklı düğümlerde kopyalanarak güvenli bir şekilde saklanır.
- **Yüksek Performans:** Büyük veriye hızlı erişim sağlar.

### **3. MapReduce**

MapReduce, Hadoop ekosisteminde veri işleme modeli olarak kullanılır. Büyük veri kümelerini iki ana aşamada işler:

- **Map:** Veriyi küçük parçalara ayırarak işler. Örneğin, her düğümdeki veriyi ayrı ayrı işleyip sonucu bir ara dosyada saklar.
- **Reduce:** Map aşamasında işlenen verileri birleştirip, nihai sonucu ortaya çıkarır.

MapReduce, özellikle paralel işlemeye uygun bir model olduğu için büyük veri işleme görevlerinde yaygın olarak kullanılır.

#### **Öne Çıkan Özellikleri:**

- **Paralel İşleme:** Veriyi eşzamanlı olarak birçok düğümde işler.
- **Basit Kodlama:** Karmaşık işlemleri basit map ve reduce işlemleri ile gerçekleştirir.

### **4. YARN (Yet Another Resource Negotiator)**

YARN, Hadoop ekosisteminde kaynak yönetiminden sorumludur. Hadoop'un ikinci versiyonuyla birlikte gelen bu bileşen, iş yüklerini yönetmek için işlemci ve bellek gibi kaynakların dağıtımını koordine eder. Bu sayede Hadoop üzerinde birden fazla uygulamanın aynı anda çalışması sağlanır.

#### **Öne Çıkan Özellikleri:**

- **Kaynak Yönetimi:** Sistem kaynaklarını iş yüklerine uygun şekilde dağıtır.
- **Çoklu Uygulama Desteği:** Birden fazla veri işleme uygulamasının aynı anda çalışmasını sağlar.

Bu dört bileşen sayesinde Hadoop, büyük veri kümeleri üzerinde dağıtık ve paralel işleme imkânı sunarak yüksek performans sağlar.

## **APACHE NİFİ**

Apache NiFi, veri akışlarını tasarlamak, yönetmek ve otomatikleştirmek için kullanılan açık kaynaklı bir veri entegrasyon aracıdır. Özellikle büyük veri ortamlarında, veri akışlarının verimli bir şekilde yönetilmesi için güçlü bir çözüm sunar. NiFi, veriyi bir noktadan diğerine taşıırken, dönüştürme, filtreleme, doğrulama ve zenginleştirme gibi işlemleri kolayca gerçekleştirebilir. "Flow-based

Programming” prensibine dayanarak, verilerin kaynaktan hedefe kolayca yönlendirilmesine olanak tanır.

### Apache NiFi ile Yapılabilecekler

1. **Veri Toplama:** Farklı veri kaynaklarından (veritabanları, IoT cihazları, dosya sistemleri, bulut depoları, vb.) veri toplayarak birleştirebilir.
2. **Veri Dönüştürme:** Veriyi hedef sistemlerin ihtiyaçlarına göre dönüştürme, yapılandırma ve normalleştirme işlemleri yapılabilir.
3. **Veri Filtreleme ve Doğrulama:** Gereksiz verileri filtreleyerek veya verinin doğru formatta olduğundan emin olarak veri kalitesini artırabilir.
4. **Veri Rotalama:** Veriyi farklı hedeflere (veritabanları, veri gölleri, veri ambarları) yönlendirme veya çoklu hedeflere yönlendirme işlemleri yapılabilir.
5. **Görselleştirme:** NiFi, veri akışlarını grafiksel bir arayüzle gösterir, böylece kullanıcılar süreçleri kolayca takip edebilir ve düzenleyebilir.

### Büyük Veri ve İş Zekası ile Bağlantısı

Apache NiFi, büyük veri platformlarına veri entegrasyonunda etkin bir rol oynar. Örneğin:

- **Büyük Veri Sistemleri ile Entegrasyon:** Apache NiFi, Hadoop, Spark, Kafka gibi büyük veri ekosistemleriyle entegre olarak çalışır. Bu sayede büyük veri platformlarına veri taşıırken veya bu platformlardan veri alırken akışları yönetmek mümkün olur.
- **ETL Süreçleri:** İş zekası uygulamalarında veri hazırlama süreçlerinin büyük bir kısmını otomatikleştirmek için NiFi kullanılabilir. Bu, verilerin iş zekası platformlarına (örneğin Power BI, Tableau) aktarılmasını ve görselleştirilmesini kolaylaştırır.
- **Gerçek Zamanlı Analitik:** NiFi, veriyi gerçek zamanlı olarak işleyebilir ve veri akışlarını analiz için anında uygun platformlara yönlendirebilir. Bu, iş zekası projelerinde anlık raporlamalar ve karar destek sistemleri için kritik öneme sahiptir.

Apache NiFi, büyük veri ve iş zekası ortamlarında veri entegrasyonunu basitleştirir ve veri yönetimini otomatikleştirerek süreçleri hızlandırır.

**ETL (EXTRACT, TRANSFORM, LOAD)**, veri entegrasyonu ve veri yönetimi için yaygın olarak kullanılan bir süreçtir. Verinin çeşitli kaynaklardan alınarak hedef sistemlere taşınması, dönüştürülmesi ve yüklenmesi işlemlerini ifade eder. Özellikle veri ambarı (data warehouse) projelerinde kullanılır ve büyük veri analitiği ile iş zekası uygulamaları için veri hazırlama sürecinin temelini oluşturur.

### ETL Süreçlerinin Aşamaları

1. **Extract (Veri Çekme):**
  - Bu aşamada veri, farklı kaynaklardan (veritabanları, dosyalar, API'ler, IoT cihazları vb.) alınır.
  - Veri kaynakları heterojen (farklı veri yapısında) olabilir; ETL süreci bu veriyi standart bir formata çekerek işlemeye başlar.
  - Çekilen veri ham halde olur ve temizleme, düzenleme gibi işlemlere ihtiyaç duyar.
2. **Transform (Veri Dönüştürme):**
  - Verinin analiz edilebilir, uyumlu ve anlamlı hale getirilmesi için dönüştürme işlemleri yapılır.
  - Bu aşamada veri temizleme, birleştirme, filtreleme, normalleştirme, zenginleştirme, hesaplama gibi işlemler gerçekleştirilir.
  - Örneğin, tarih formatlarının uyumlu hale getirilmesi, eksik veya hatalı verilerin düzeltilmesi gibi dönüşümler yapılabilir.

### 3. Load (Veri Yükleme):

- Dönüştürülen veri, hedef sistemlere (genellikle veri ambarları veya veri gölleri) yüklenir.
- Bu aşamada verinin, iş zekası araçları veya raporlama sistemleri tarafından analiz edilmesi için hazır hale getirilmesi sağlanır.
- Veri yükleme işlemi, tam yükleme veya kısmi güncellemeler şeklinde olabilir; büyük veri projelerinde bu işlem genellikle düzenli olarak otomatikleştirilir.

### ETL Süreçlerinin Önemi

ETL süreçleri, veriyi analiz için kullanılabilir hale getirmek açısından kritik bir rol oynar. ETL olmadan:

- Farklı kaynaklardan gelen veriler birbirine uymaz ve anlamlı raporlar veya analizler çıkarmak zorlaşır.
- Veriyi güvenilir ve hızlı bir şekilde analiz edilebilecek bir ortama taşımak mümkün olmaz.

### Büyük Veri ve İş Zekası ile İlişkisi

- **Veri Ambarı (Data Warehouse):** ETL süreçleri genellikle veri ambarlarına veri taşımak için kullanılır. Veri ambarları, uzun dönemli analiz ve raporlama için kullanılır ve ETL sayesinde bu ortamlar sürekli güncel ve temiz veriyle beslenir.
- **İş Zekası Araçları:** Power BI, Tableau gibi iş zekası araçlarının veri çekebileceği, analiz yapabileceği temiz veri setleri ETL süreçleriyle hazırlanır.
- **Büyük Veri Ekosistemleri:** ETL işlemleri, Hadoop, Spark gibi büyük veri platformlarına veri taşıırken veya bu platformlardan veriyi alırken kullanılır. Apache NiFi gibi araçlar, büyük veri ortamlarında ETL süreçlerinin yönetimini kolaylaştırır.

ETL süreçleri, iş zekası ve veri analitiği uygulamaları için veriyi hazır hale getirerek işletmelerin karar alma süreçlerinde önemli bir rol oynar.

## SENARYO: E-TİCARET SİTESİNDE SATIŞ VE MÜŞTERİ VERİLERİNİN ANALİZİ

Bir e-ticaret sitesinde, farklı veri kaynaklarından alınan satış ve müşteri verilerini analiz etmek istiyoruz. Bu veriler, satış trendlerini anlamak, müşteri segmentasyonu yapmak ve kampanyalar düzenlemek için kullanılacak. Apache NiFi ile bu veri akışını oluşturup otomatikleştirerek bir veri ambarına aktaracağız.

### Adım Adım Senaryo Akışı

#### 1. Veri Çekme (Extract)

- **Veri Kaynakları:** Satış verileri, MySQL veritabanında; müşteri verileri ise bir CRM sisteminde (örneğin, Salesforce) ve sosyal medya platformlarından toplanıyor.
- **Apache NiFi İşlemleri:**
  - **MySQL'den Veri Çekme:** NiFi'nin *ExecuteSQL* işlemcisi kullanılarak, belirli bir SQL sorgusuyla MySQL veritabanındaki satış verileri çekilir.
  - **API Üzerinden Veri Alma:** Müşteri verileri, sosyal medya API'lerinden (örneğin Twitter veya Facebook) *InvokeHTTP* işlemcisi ile JSON formatında çekilir.
  - **CRM Verisi Çekme:** CRM sisteminden (Salesforce, HubSpot gibi) veri almak için API entegrasyonları yapılır. NiFi'nin *GetHTTP* veya *InvokeHTTP* işlemcileri, CRM verilerini JSON veya XML olarak alabilir.

#### 2. Veri Dönüştürme (Transform)

- Bu aşamada, farklı kaynaklardan gelen veriyi uyumlu ve analiz edilebilir bir formata dönüştürmemiz gerekiyor.
- **Apache NiFi İşlemleri:**

- **Veriyi Temizleme:** *ReplaceText*, *UpdateAttribute*, ve *ConvertRecord* gibi işlemcilerle eksik ya da yanlış veri değerleri temizlenir, tarih formatları veya metin düzenlemeleri yapılır.
- **Veriyi Birleştirme:** Müşteri ve satış verileri ortak bir kimlik (örneğin, müşteri ID'si) üzerinden birleştirilir. *MergeContent* işlemcisi, JSON dosyalarını birleştirip tek bir dosya veya kayıt olarak çıkartabilir.
- **Format Dönüştürme:** JSON, CSV ve XML formatındaki veriler analiz için uygun hale getirilir. Örneğin, veriyi *ConvertJSONToSQL* veya *ConvertRecord* gibi işlemcilerle SQL formatına dönüştürebiliriz.

### 3. Veri Yükleme (Load)

- Dönüştürülmüş ve temizlenmiş veriyi analiz için bir veri ambarına ya da veri gölüne yükleyeceğiz.
- **Apache NiFi İşlemleri:**
  - **Veriyi Veri Ambarına Yükleme:** *PutDatabaseRecord* işlemcisi ile veri doğrudan veri ambarındaki bir tabloya (örneğin, PostgreSQL veya Amazon Redshift) yazılabilir.
  - **Hadoop veya S3'e Yükleme:** Büyük veri için Hadoop HDFS veya Amazon S3 gibi bulut depolama sistemleri kullanılabilir. NiFi'nin *PutHDFS* veya *PutS3Object* işlemcileri, veriyi doğrudan bu sistemlere yazabilir.
  - **Gerçek Zamanlı Raporlama için Kafka:** Eğer verinin anlık olarak analiz edilmesi gerekiyorsa, NiFi ile veriyi *PublishKafka* işlemcisi üzerinden bir Kafka konusuna gönderebiliriz. Bu da Spark gibi gerçek zamanlı analiz araçlarının bu veriyi anında işleyebilmesini sağlar.

### Ekstra: Otomasyon ve İzleme

Apache NiFi, akışları otomatik olarak çalıştırabilir ve belirli zaman aralıklarında veriyi güncelleyebilir. Ayrıca, her işlem adımının akış grafiği üzerinde görsel olarak izlenebilmesi, sorun çıktığında hızlı müdahale edebilme olanağı sağlar.

### Senaryo Çıktısı

Bu süreç sayesinde:

- E-ticaret sitesi, her gün düzenli olarak veri ambarına güncellenmiş satış ve müşteri verilerini gönderir.
- Bu veriler iş zekası araçlarıyla (Power BI, Tableau) analiz edilerek satış trendleri, müşteri davranışları ve potansiyel müşteri segmentleri belirlenir.
- Sosyal medya verisi sayesinde, müşterilerin memnuniyeti ve geri bildirimleri izlenebilir, markanın sosyal medya üzerindeki etkisi analiz edilebilir.

### Apache NiFi'nin Bu Süreçteki Avantajları

- **Otomasyon:** Veri akışı tamamen otomatik hale gelir, manuel işlem ihtiyacını ortadan kaldırır.
- **Esneklik:** Farklı veri kaynaklarını kolayca entegre eder ve istenen hedef sistemlere yönlendirir.
- **Gerçek Zamanlı İşleme:** Büyük veri projelerinde veriyi hızlı ve sürekli olarak işlemek için idealdir.

Bu şekilde, Apache NiFi ile ETL süreci kurulup yönetilebilir, büyük veriler analiz için sürekli olarak hazır hale getirilir.

### **FLOW-BASED PROGRAMMING**

(FBP), yani Akış Tabanlı Programlama, bir uygulamayı bağımsız ve yeniden kullanılabilir bileşenler (komponentler) olarak modelleyip, veriyi bu bileşenler arasında bir ağ (network) üzerinden yönlendirme yaklaşımıdır. Bu programlama modelinde, bileşenler birbirine

bağılıdır, ancak doğrudan etkileşime geçmezler. Bunun yerine, veri akışı üzerinde çalışırlar ve veriyi bir kuyruğa veya veri kanalına bırakıp alırlar. FBP, özellikle veri yoğun uygulamalarda karmaşık iş akışlarını yönetmeyi kolaylaştırır.

### Flow-based Programming Prensipleri

1. **Bağımsız Bileşenler:** Her bir bileşen, belirli bir işlevi yerine getiren, kendi girdisini ve çıktısını işleyen bağımsız bir ünedir.
2. **Veri Akışı ile İletişim:** Bileşenler birbirleriyle doğrudan iletişime geçmez; bunun yerine veriyi akış kanallarında paylaşır.
3. **Yeniden Kullanılabilirlik ve Modülerlik:** Bileşenler, farklı uygulamalarda tekrar kullanılabilir.
4. **Paralel Çalışma:** Her bileşen bağımsız çalıştığı için, işlemler paralel olarak gerçekleştirilebilir ve bu da performansı artırır.

### Örnek: Apache NiFi ile Akış Tabanlı Programlama

Apache NiFi, FBP prensibini uygulayarak veri akışlarını yönetir. Diyelim ki sosyal medya üzerinden bir marka ile ilgili yorumları toplayıp analiz eden bir süreç oluşturacağız. İşte örnek bir iş akışı:

#### 1. Veri Kaynağı (Data Source) - Yorum Çekme

- *GetTwitter* adlı bir bileşen kullanarak Twitter API'sinden belirli bir markayla ilgili son yorumları çekiyoruz. Bu bileşen sadece Twitter'dan veri almak için yapılandırılmış bağımsız bir bileşendir.

#### 2. Veri Temizleme - Gereksiz Bilgileri Çıkarma

- *ReplaceText* adlı bir bileşen, yorumlardaki gereksiz etiketleri veya sembolleri temizlemekle sorumlu. Bu bileşen yalnızca veriyi temizler; diğer bileşenler veriyi temizleme işlemi hakkında bilgi sahibi olmaz.

#### 3. Veri Analizi - Duygu Analizi

- *InvokeHTTP* adlı bir bileşen ile duygu analizini gerçekleştiren bir API'ye (örneğin, bir makine öğrenmesi modeline) yorumları gönderiyoruz. Bu bileşen, yorumların duygu durumlarını analiz edip geri döner.

#### 4. Veri Depolama - Sonuçları Kaydetme

- *PutDatabaseRecord* bileşeni, analiz sonuçlarını bir veritabanına (örneğin MySQL) kaydetmekle sorumlu. Bu bileşen yalnızca veri tabanına yazma işlemini yapar, analiz bileşeninin detaylarıyla ilgilenmez.

#### 5. Veri Görselleştirme - Raporlama

- Analiz edilen ve veritabanında saklanan veriler daha sonra iş zekası araçları tarafından alınarak görselleştirilir. Bu da iş akışının son adımı olur.

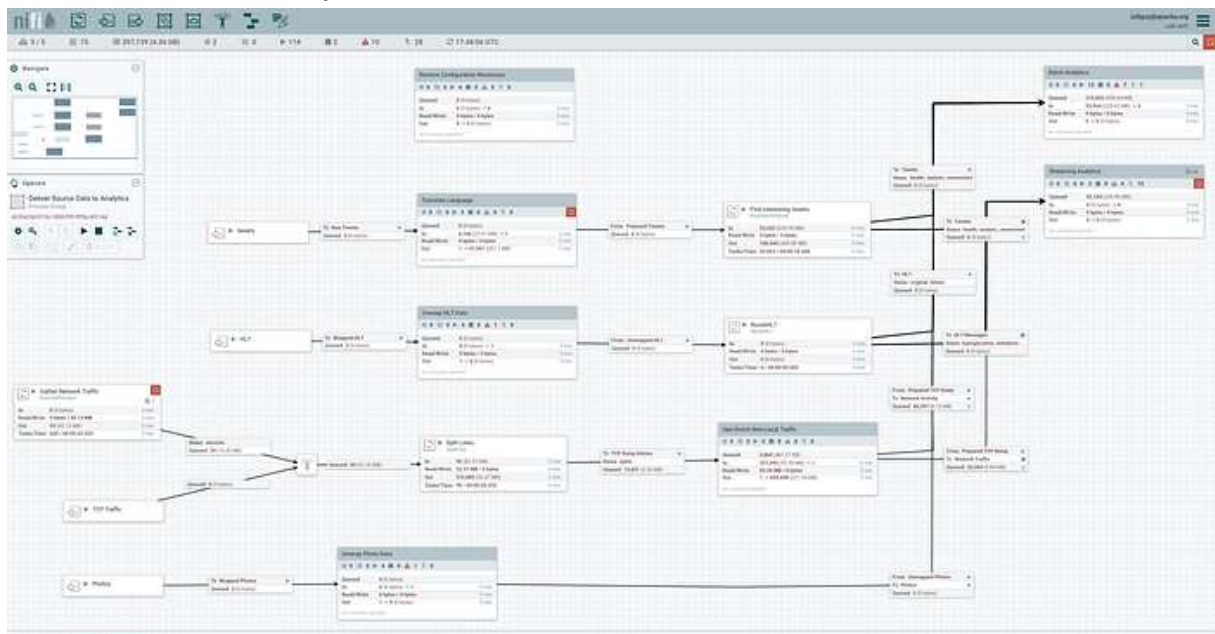
### FBP'nin Avantajları Bu Örnekte Nasıl Görülür?

- **Bağımsız Bileşenler:** Her adım (veri çekme, temizleme, analiz, saklama) birbirinden bağımsız bileşenler tarafından yönetildiği için, her bileşen üzerinde bağımsız değişiklik yapılabilir.
- **Paralel İşlem:** Birden fazla bileşen aynı anda çalışabilir, örneğin yorum çekilirken temizleme işlemi başka bir bileşende devam edebilir.
- **Modülerlik ve Yeniden Kullanım:** *GetTwitter* bileşeni, başka projelerde Twitter'dan veri çekmek için yeniden kullanılabilir.

**Apache Nifi**, açık kaynaklı bir veri entegrasyon ve akış işlem platformudur. NiFi'nin temel amacı, farklı veri kaynakları arasında veri akışlarını kolayca yönetmek, işlemek ve taşımak için bir hizmet sunmaktır.

1. **Veri Toplama ve Akış İşleme:** NiFi, farklı kaynaklardan veri toplamak ve bu verileri işlemek için kullanılabilir. Sensörlerden, veritabanlarından, uygulamalardan veya diğer veri kaynaklarından gelen verileri yakalayabilir, temizleyebilir ve işleyebilirsiniz.
2. **Veri Entegrasyonu:** Farklı veri formatları arasında dönüşüm yapabilir ve veri entegrasyonu işlemleri için kullanabilirsiniz. Örneğin, JSON verilerini XML'e dönüştürebilir veya farklı veri tabanları arasında veri aktarımı yapabilirsiniz.
3. **Veri Akışı Yönetimi:** NiFi, veri akışlarını yönetmek için kullanışlı araçlar sunar. Verileri çeşitli işlem aşamalarından (filtreleme, düzenleme, zenginleştirme, güvenlik) geçirebilir ve hedef sistemlere iletebilirsiniz.
4. **Gerçek Zamanlı İşleme:** NiFi, gerçek zamanlı veri işleme senaryoları için uygundur. Veriler anlık olarak işlenip iletilirken, gecikme sürelerini minimize edilebilir.

## Processor, Processor Group ve Daha Fazlası



**Processor’lar**, görevleri otomatikleştirmenize, veri akışlarınızı işlemeye ve verileri istenen formata getirmenize olanak tanır. Özellikle veri entegrasyonu, veri akışı işleme ve ETL (Extract, Transform, Load) işlemleri için güçlü bir şekilde kullanılırlar. Apache NiFi, birçok yerleşik Processor ile birlikte gelir ve ayrıca özelleştirilmiş Processor’lar oluşturmanıza da olanak tanır.

Her Processor, belirli bir işlevi yerine getirir. Örneğin, veri çıkarma, veri dönüşümü, filtreleme, veri zenginleştirme, güvenlik işlemleri ve daha fazlası gibi farklı işlevleri olan Processor'lar bulunmaktadır. Kullanıcılar bu Processor'ları bağlayarak ve yapılandırarak veri akışları oluşturabilir ve verileri istedikleri biçimde işleyebilirler.


**Processor Group'lar**, aşağıdaki amaçlar için kullanılabilir:

1. **Modülerlik:** Benzer işlemleri bir araya getirerek, bir Processor Group içindeki işlevselliği daha kolay anlayabilir ve yönetebilirsiniz. Ayrıca, bu modüler yapı sayesinde belirli işlemleri yeniden kullanabilirsiniz.

2. İzolasyon: Her Processor Group, kendi ayarları, ilişkilendirmeleri ve izinleriyle birlikte gelir. Bu, farklı işlemlere sahip Processor Group'ları izole etmenizi sağlar, böylece hataların yayılmasını önler.
3. Güvenlik ve İzinler: Processor Group'lar, işlevleri sınırlamak ve izinleri daha iyi yönetmek için kullanılır. Her Processor Group, kendi erişim izinleri ve güvenlik ayarlarına sahip olabilir.
4. İzleme ve Yönetim: Processor Group'lar, NiFi akışınızı daha büyük ve karmaşık hale getirdiğinizde, işlevsellikleri daha kolay izlemenize ve yönetmenize yardımcı olur.
5. Paralel İşleme: Processor Group'lar, işlevleri paralel olarak yürütmek için kullanılabilir. Belirli bir Processor Group içindeki işlemler, aynı anda birden çok işlemciye yayılabilir.

### Sık Kullanılan Processörlara Örnek

#### 1. Invoke Http



**InvokeHTTP**  
InvokeHTTP 1.23.2  
org.apache.nifi - nifi-standard-nar


In	0 (0 bytes)	5 min
Read/Write	0 bytes / 0 bytes	5 min
Out	0 (0 bytes)	5 min
Tasks/Time	0 / 00:00:00.000	5 min

Property	Value
HTTP Method	GET
HTTP URL	https://random-data-api.com/api/v2/users
HTTP/2 Disabled	False
SSL Context Service	No value set
Socket Connect Timeout	5 secs
Socket Read Timeout	15 secs
Socket Write Timeout	15 secs
Socket Idle Timeout	5 mins
Socket Idle Connections	5
Proxy Configuration Service	No value set
Proxy Host	No value set
Request OAuth2 Access Token Provider	No value set

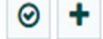
#### Örnek Endpoint İsteği

Bir endpointe istek atmak için kullanılır. Endpointin döndüğü veri türünden cevap döner.

## 2. Split Record

	<b>SplitRecord</b> SplitRecord 1.23.2 org.apache.nifi - nifi-standard-nar	
In	0 (0 bytes)	5 min
Read/Write	0 bytes / 0 bytes	5 min
Out	0 (0 bytes)	5 min
Tasks/Time	0 / 00:00:00.000	5 min


Required field



Property	Value
Record Reader	 JsonTreeReader →
Record Writer	 JsonRecordSetWriter →
Records Per Split	 2

Gelen veriyi böler. Record reader/writer avro, csv, json,xml, parguet olabilir. İhtiyacınıza uygun veri formatını seçmeniz gerekir.

## 3. SplitJson

	<b>subscription</b> SplitJson 1.23.2 org.apache.nifi - nifi-standard-nar	
In	0 (0 bytes)	5 min
Read/Write	0 bytes / 0 bytes	5 min
Out	0 (0 bytes)	5 min
Tasks/Time	0 / 00:00:00.000	5 min

Property	Value
JsonPath Expression	 \$..subscription
Null Value Representation	 empty string


Bu process json veri türünü ayırmamıza ve istediğimiz fieldları çekmemize yardımcı olur.

*“Jsonpath Expression” parametresi ile istediğiniz field’a erişebilirsiniz.*

[Jsonpath yapısı kullanılarak filtreleme yapılır.](#)

[Nifi üzerinde veri filtrelemek için kendine özel dil olan Nifi Expression Language kullanılır.](#)

## 4. PutDatabaseRecord

	<b>PutDatabaseRecord</b> PutDatabaseRecord 1.23.2 org.apache.nifi - nifi-standard-nar	
In	0 (0 bytes)	5 min
Read/Write	0 bytes / 0 bytes	5 min
Out	0 (0 bytes)	5 min
Tasks/Time	0 / 00:00:00.000	5 min



Property	Value
Record Reader	JsonTreeReader →
Database Type	PostgreSQL
Statement Type	INSERT
Data Record Path	No value set
Database Connection Pooling Service	DBCPConnectionPool →
Catalog Name	No value set
Schema Name	No value set
Table Name	subscriptions
Translate Field Names	true
Unmatched Field Behavior	Ignore Unmatched Fields
Unmatched Column Behavior	Fail on Unmatched Columns
Quote Column Identifiers	false

Veritabanlarına veri yazmak için kullanılır. Özellikle ETL işlemleri için kullanılır ve veri tabanlarında veri depolama amaçları için kullanışlıdır.

**Record Reader:** Gelen verinin hangi türde olduğunu belirler

**Statement Type:** Veritabanına hangi CRUD işleminin yapılacağını belirler.

**Table Name:** Verilerin hangi tabloya yazılacağını belirler.

**Database Connection Pooling Service:** veritabanı erişim bilgilerini içerir.

#### Diğer Processorler

1. **SplitText:** Metin verilerini belirli bir ayırıcıya göre bölmek için kullanılır. Özellikle log dosyaları veya CSV dosyaları gibi metin tabanlı verileri işlerken kullanışlıdır.
2. **MergeContent:** Birden çok akışı birleştirmek için kullanılır. Özellikle paralel işlem gerektiren veri işleme görevlerinde kullanılır.
3. **UpdateAttribute:** Bu Processor, akış içindeki özellikleri güncellemek için kullanılır. Özellikle akış içindeki verileri dönüştürme veya zenginleştirme işlemlerinde kullanılır.
4. **RouteOnAttribute:** Akış içindeki özelliklere dayalı olarak belirli koşullara göre yönlendirme yapar. Özellikle filtreleme ve yönlendirme görevleri için kullanılır.
5. **ExecuteScript:** Bu Processor, kullanıcı tarafından yazılan betikleri çalıştırmak için kullanılır. Python, Groovy, Ruby gibi dilleri destekler ve özelleştirilmiş işlemler için kullanılır.
6. **ListenHTTP ve HandleHttpRequest:** Bu Processor'lar, HTTP isteklerini dinlemek ve işlemek için kullanılır. Özellikle dış dünyadan gelen HTTP isteklerini işlemek için kullanışlıdır.
7. **GetKafka ve PublishKafka:** Apache Kafka ile entegrasyon için kullanılır. Kafka akışlarından veri alma ve Kafka'ya veri gönderme işlemlerini gerçekleştirir.

## Apache Airflow,

iş akışlarının (workflows) programlanabilir bir şekilde oluşturulmasına, zamanlanmasına ve izlemesine olanak tanıyan açık kaynaklı bir araçtır. Büyük veri işleme, veri mühendisliği, veri analitiği gibi alanlarda iş akışlarını otomatikleştirmek için yaygın olarak kullanılır. Airflow, bir iş akışını bir dizi adıma

bölüp bu adımların bağımlılıklarına göre sıraya koyarak, karmaşık veri işlemlerini düzenli bir şekilde yönetmeyi sağlar.

### Apache Airflow Nerelerde Kullanılır?

Airflow, özellikle veri mühendisliği ve analitiğinde kullanılır. Bazı yaygın kullanım alanları:

1. **Veri Tüketimi:** Verilerin bir veri ambarına veya veri gölüne aktarımı.
2. **ETL İşlemleri:** Extract (Çekme), Transform (Dönüştürme) ve Load (Yükleme) süreçlerinde verilerin işlenmesi.
3. **Veri Bilimi Modellerinin Eğitimi:** Belirli aralıklarla veri bilimcilerin modellerini güncellemek veya yeniden eğitmek.
4. **Bildirimler ve Raporlama:** Belirli bir zamanda çalışan raporlama ve bildirim süreçleri.
5. **Dış Veri Kaynakları ile Entegrasyon:** API veya veri tabanı gibi dış kaynaklardan veri alınıp işlenmesi ve bu verilerin raporlanması.

### Airflow'un Faydaları

1. **Otomasyon:** Süreçlerin otomatikleştirilmesi sayesinde manuel hatalar azalır.
2. **Zamanlama ve İzleme:** Zaman planına göre süreçlerin çalıştırılması ve izlenmesi kolaylaşır.
3. **Bağımlılık Yönetimi:** İş adımları arasındaki bağımlılıklar kolayca yönetilir.
4. **Ölçeklenebilirlik:** İş akışlarının farklı makinelerde paralel çalıştırılması mümkündür.

### Kısa Bir Senaryo ile Anlatım

Diyelim ki bir e-ticaret şirketinde çalışıyorsunuz ve her gün müşteri davranışlarına yönelik analizler yapıyorsunuz. Airflow ile bu süreci otomatik hale getirmek için şöyle bir iş akışı (DAG) tasarlayabilirsiniz:

1. **Veri Çekme:** Her sabah saat 6'da, veritabanından yeni müşteri davranış verileri çekilir.
2. **Veri Dönüştürme:** Çekilen veriler ETL işlemine alınarak temizlenir, eksik veriler tamamlanır ve analiz için uygun hale getirilir.
3. **Analiz ve Model Çalıştırma:** İşlenmiş veriler üzerine makine öğrenmesi modeli çalıştırılarak müşteri segmentasyonu yapılır.
4. **Raporlama:** Modelin sonuçları günlük olarak bir rapor haline getirilir ve ilgili yöneticilere e-posta ile gönderilir.

Bu iş akışında, her adım bir sonraki adıma bağımlı olduğu için Airflow, adımların sırasıyla çalışmasını ve başarısızlık durumunda tekrar denemesini sağlar.

## KAFKA

Apache Software Foundation tarafından geliştirilen açık kaynak kodlu, dağıtık bir veri akış platformudur. Temel olarak, yüksek hacimli verilerin güvenli ve hızlı bir şekilde iletilmesi, depolanması ve işlenmesi amacıyla kullanılır. Kafka, büyük veri mimarisinin kritik bir bileşeni olarak kabul edilir çünkü gerçek zamanlı veri akışı, veri entegrasyonu, veri depolama ve büyük verinin anlık analizine imkan tanır.

### Kafka'nın Kullanım Alanları

Kafka'nın yaygın kullanım alanları şunlardır:

1. **Gerçek Zamanlı Veri Akışı:** Özellikle büyük veri ve yapay zeka projelerinde verilerin gerçek zamanlı analiz edilmesi gerekebilir. Kafka, milyonlarca veriyi saniyeler içinde aktarabilir.

2. **Veri Tabanı Güncellemeleri ve Senkronizasyon:** Farklı veri tabanları arasında veri akışını sağlayarak veri güncellemelerini eş zamanlı yapar.
3. **Olay İzleme:** Bir uygulamanın veya sistemin olaylarını (log) izlemek için kullanılır. Özellikle güvenlik veya müşteri davranış analizleri gibi alanlarda kullanılır.
4. **Mesajlaşma Sistemi:** Kafka, çok yüksek hacimdeki mesajları güvenli ve verimli bir şekilde bir yerden başka bir yere aktarabilir.
5. **Ölçeklenebilirlik Gereken Sistemler:** Yüksek hacimli veriyi işleyebildiği için büyük veri sistemlerinde ölçeklenebilir bir yapı sağlar.

#### **Kafka'nın Kullanma Amacı**

Kafka'nın temel amacı, veri akışlarını işlemek ve yüksek verimlilikle bu veriyi taşıyarak sistemler arasında iletişim kurmaktır. Böylece:

- **Veri entegrasyonu sağlar:** Farklı sistemlerdeki veriler merkezi bir yerden beslenir ve analiz edilebilir.
- **Yüksek verimlilik ve güvenilirlik sağlar:** Kafka, büyük hacimli verileri kayıpsız bir şekilde işleyebilir.
- **Gerçek zamanlı analiz yapılır:** Veri akışları anlık olarak analiz edilebilir ve hızlı karar alma süreçlerine katkıda bulunur.

#### **Büyük Veri ile Kafka'nın Bağlantısı**

Kafka, büyük veri platformlarının en önemli bileşenlerinden biridir. Büyük veri sistemlerinde veri toplama, depolama, işleme ve analiz işlemleri yapılır. Kafka, bu sürecin **veri toplama ve veri akışı** kısmında yer alır. Veriler çok sayıda kaynaktan (örn. uygulamalar, IoT cihazları, sosyal medya akışları vb.) toplandıktan sonra Kafka aracılığıyla Hadoop, Spark, veya NoSQL gibi büyük veri işleme ve depolama sistemlerine iletilir. Böylece verilerin anlık olarak işlenmesi ve analiz edilmesi sağlanır.

#### **Kafka Kullanım Senaryosu: E-Ticaret Sitesinde Gerçek Zamanlı Analiz**

Bir e-ticaret sitesinin ziyaretçileri, her saniye binlerce veri noktası oluşturur (sayfa ziyaretleri, arama terimleri, sepete ürün ekleme, satın alma vb.). Bu eylemleri analiz etmek, müşterilerin ilgi alanlarını ve alışveriş alışkanlıklarını anlamak için çok değerlidir. Kafka, bu süreci kolaylaştırır:

1. **Olay Akışı Yaratma:** Kullanıcılar web sitesinde bir eylem yaptığında (örneğin, bir ürün sayfasını ziyaret ettiğinde) bu olay bir veri olarak Kafka'ya gönderilir.
2. **Veriyi Toplama:** Kafka, bu olayları gerçek zamanlı olarak toplar ve bir mesaj kuyruğuna koyar.
3. **Veriyi Depolama ve Analiz:** Kafka'daki veriler, anlık analiz için Spark gibi bir veri işleme platformuna aktarılır. Spark, verileri analiz ederek kullanıcıların hangi ürünlere daha fazla ilgi gösterdiğini anlar.
4. **Öneri Sistemi:** Analiz sonuçlarına göre, kullanıcıya anında ilgi duyabileceği ürün önerileri yapılır. Örneğin, "Bu ürünü satın alanlar, şunları da satın aldı" gibi öneriler gösterilir.
5. **Raporlama ve Karar Destek:** Kafka üzerinden toplanan tüm veriler, günlük/haftalık raporlar olarak yöneticilere sunulabilir ve stratejik kararlar alınabilir.

Bu örnekte, Kafka sayesinde yüksek hacimli veri, anlık olarak analiz edilip işlenmiş olur ve e-ticaret sitesinin müşteri deneyimi ve pazarlama stratejileri geliştirilir. Kafka, büyük veri sistemlerinin etkin şekilde çalışmasına katkıda bulunarak, işletmelere veri odaklı karar verme süreçlerinde büyük fayda sağlar.

## RabbitMQ

açık kaynak kodlu bir mesajlaşma aracıdır ve temel olarak farklı uygulamalar veya bileşenler arasında iletişimi sağlamak için kullanılır. Mesajların bir kuyruğa alınıp belirli bir sırayla veya belirli bir şekilde işlenmesi gereken yerlerde kullanılır. RabbitMQ, mesajların güvenilir, hızlı ve verimli bir şekilde gönderilmesini ve alınmasını sağlar, bu sayede uygulamalar arasındaki entegrasyon süreçleri sorunsuz gerçekleşir.

### RabbitMQ'nun Kullanım Alanları

RabbitMQ'nun yaygın kullanım alanları şunlardır:

1. **Dağıtık Sistemlerde İletişim:** Mikro hizmet mimarilerinde, farklı hizmetler arasında veri akışını sağlamak için kullanılır.
2. **Mesaj Sıralama:** RabbitMQ, mesajları belirli bir sırayla işleme alır. Özellikle sipariş işleme gibi sıralı veri akışlarının önemli olduğu sistemlerde tercih edilir.
3. **Görev Dağıtımı:** Büyük görevlerin küçük parçalara bölünüp farklı çalışanlara (worker) dağıtılmasını sağlar. Bu, iş yükünün dengeli bir şekilde dağıtılmasına yardımcı olur.
4. **Gerçek Zamanlı İşlem Gerektiren Sistemler:** Canlı veri akışları, bildirim sistemleri veya anlık analiz gerektiren uygulamalarda kullanılır.
5. **E-posta Bildirimleri ve İşlem Sonuçları:** Özellikle müşteri işlemlerinde ya da olay odaklı sistemlerde e-posta bildirimleri veya diğer işlemlerin sonuçlarını bildirmek için tercih edilir.

### RabbitMQ'nun Kullanma Amacı

RabbitMQ'nun temel amacı, bir uygulamadan diğerine güvenilir bir şekilde mesaj iletimini sağlamaktır. Bu, bazı avantajları beraberinde getirir:

- **Sistemlerin Bağılantısız Çalışabilmesi:** Uygulamalar birbirine sıkı sıkıya bağlı olmadan iletişim kurabilir. Mesajlar kuyrukta bekleyebilir ve işlenmeye hazır olduğunda alınabilir.
- **Ölçeklenebilirlik ve İş Yüğü Yönetimi:** RabbitMQ ile sistemin iş yükü, çalışanlara dağıtılabilir. Bu sayede ani yük artışları dengelenebilir.
- **Esneklik ve Modülerlik Sağlar:** RabbitMQ, farklı bileşenlerin bir arada çalışmasını kolaylaştırır ve modüler bir yapı kurmayı sağlar.

### Büyük Veri ile RabbitMQ'nun Bağlantısı

RabbitMQ, doğrudan büyük veri işleme platformu değildir, ancak büyük veri sistemlerinde mesajların ön işlenmesi veya veri akışı sırasında kullanılır. Örneğin, IoT cihazlarından gelen veriler önce RabbitMQ gibi bir mesajlaşma aracında toplanabilir, ardından analiz için büyük veri platformlarına (örn. Hadoop, Spark) gönderilebilir. RabbitMQ, bu süreçte **veri entegrasyonu** ve **mesaj akışı yönetimi** sağlar. Büyük veri altyapısında, RabbitMQ veri hacmini yönetmeye yardımcı olur ve verilerin güvenilir bir şekilde toplanıp işlenmesini destekler.

### RabbitMQ Kullanım Senaryosu: E-Ticaret Sitesinde Sipariş İşleme Sistemi

Bir e-ticaret sitesinin sipariş işlemlerini düşünelim. Müşteriler alışveriş yaptıktan sonra siparişlerin hızla işlenmesi, faturaların kesilmesi ve envanterin güncellenmesi gerekir. Bu süreçte RabbitMQ, işlemleri daha düzenli ve sorunsuz hale getirebilir:

1. **Sipariş Talebi Gönderme:** Müşteri siparişi verdikten sonra sipariş bilgileri RabbitMQ'ya bir mesaj olarak gönderilir.
2. **Sipariş İşleme:** RabbitMQ'da bir kuyruk oluşur. Çalışan servisler (örneğin, faturalama, stok güncelleme ve nakliye hizmetleri) bu kuyruğa erişir ve sırayla siparişleri işler.

3. **Görev Dağıtımı:** Sipariş bilgisi, farklı görevler için ayrılarak ilgili servislerin kuyruğına gönderilir. Örneğin, bir siparişin ödemesi tamamlanmışsa bu bilgi faturalama servisine iletilir, ardından stok güncelleme işlemi başlar ve nakliye servisine bilgi verilir.
4. **Bildirim Gönderme:** Sipariş tamamlandıktan sonra, RabbitMQ bir bildirim mesajı gönderir ve bu mesaj bir bildirim servisine alınarak müşteriye siparişinin başarıyla tamamlandığı bilgisi gönderilir.
5. **Raporlama:** RabbitMQ üzerinden geçen mesajlar günlük olarak analiz edilebilir ve e-ticaret sitesi yönetimi tarafından sipariş sayıları, teslimat süreleri gibi raporlar oluşturulabilir.

Bu senaryoda RabbitMQ, e-ticaret sitesindeki sipariş işleme sürecinin sorunsuz bir şekilde yürütülmesini sağlar. Veriler kuyruğına alınır, sırayla işlenir, böylece siparişlerin anlık olarak işlenmesi mümkün olur. Bu yapı, büyük veri sistemlerinde veri akışını düzenli ve güvenilir hale getirmek için kullanılabilir ve yüksek iş yükü altında sistemin kararlılığını sağlar.

**Trino (eski adıyla Presto),** dağıtık bir SQL sorgu motoru (distributed SQL query engine) olup, büyük veri ve iş zekası (business intelligence) uygulamalarında yaygın olarak kullanılır.

Trino'nun Özellikleri ve Kullanım Alanları:

1. **Dağıtık Mimari:** Trino, verilerin birden fazla kaynaktan (veri ambarı, NoSQL veritabanları, dosya sistemleri vb.) okunmasını ve sorgulanmasını sağlayan dağıtık bir mimari sunar. Bu sayede, farklı veri kaynaklarından gelen verileri birleştirerek sorgulamak mümkündür.
2. **Yüksek Performans:** Trino, paralel işleme ve görüntüleme özellikleriyle yüksek performans sağlar. Büyük veri kümelerini hızlı bir şekilde sorgulamak ve analiz etmek için idealdir.
3. **Ölçeklenebilirlik:** Trino, ihtiyaç duyulan kaynak miktarına göre ölçeklenebilir. Kullanıcı sayısı veya veri hacmi arttıkça, Trino ek düğümler eklenerek genişletilebilir.
4. **Veri Kaynağı Bağımsızlığı:** Trino, çeşitli veri kaynaklarıyla (Hive, Hadoop, Amazon S3, MySQL, PostgreSQL vb.) çalışabilir. Bu sayede, farklı veri depolarından gelen verileri tek bir sorgu ile birleştirmek mümkündür.

Büyük Veri ve İş Zekası ile Bağlantısı:

Trino, büyük veri ve iş zekası uygulamalarında sıklıkla kullanılır:

**Örnek Senaryo:** Bir e-ticaret şirketi, müşteri alışveriş davranışlarını, ürün satış verilerini, stok durumlarını ve lojistik bilgilerini analiz etmek istiyor. Bu veriler, farklı veri kaynaklarında (veri ambarı, NoSQL veritabanı, CSV dosyaları vb.) dağınık bir şekilde bulunuyor.

Trino kullanarak, şirket aşağıdaki adımları gerçekleştirebilir:

1. Trino, farklı veri kaynaklarındaki verileri (müşteri işlemleri, ürün satışları, stok durumu, teslimat bilgileri vb.) birleştirebilir.
2. Müşteri alışveriş davranışları, en çok satan ürünler, stok seviyeleri, teslimat performansı gibi birleşik raporlar oluşturabilir.
3. Analitikçiler, Trino üzerinden gerçekleştirdikleri sorgularla, satış trendlerini, müşteri segmentlerini, stok optimizasyonunu ve lojistik iyileştirmelerini keşfedebilir.
4. Trino'nun paralel işleme özelliği sayesinde, bu tür kapsamlı analizler hızlı bir şekilde gerçekleştirilebilir.

Sonuç olarak, Trino, büyük veri kaynaklarındaki verileri birleştirme, sorgulama ve analiz etme konusunda güçlü bir araçtır. E-ticaret şirketinin örnek senaryosunda olduğu gibi, iş zekası

uygulamalarında Trino'nun kullanımı, şirketin operasyonel ve stratejik kararlarını destekleyecek önemli içgörüler sağlar.

## ELK ve Solr Nedir?

**ELK Stack:** ELK, Elasticsearch, Logstash ve Kibana ürünlerinin birleşimidir. Bu üç araç bir araya gelerek güçlü bir arama, log analizi ve görselleştirme platformu sunar:

1. **Elasticsearch:** Tam metin arama ve analiz için kullanılan, dağıtık ve hızlı bir arama motorudur. Büyük veri kümelerinde veriyi hızlıca indeksleme, arama ve analiz etme imkanı sağlar.
2. **Logstash:** Veriyi toplamak, işlemek ve analiz etmek için kullanılan bir veri işleme aracıdır. Logstash, birçok kaynaktan veri toplayabilir, bu veriyi işleyip Elasticsearch gibi sistemlere aktarabilir.
3. **Kibana:** Elasticsearch'ten alınan veriyi görselleştirmek için kullanılan bir araçtır. Farklı görselleştirme seçenekleriyle kullanıcıların veriyi daha kolay analiz etmelerini sağlar.

**Solr:** Solr da bir arama platformudur, özellikle Apache Lucene üzerine kuruludur. Büyük veri kümelerinde metin tabanlı aramalar yapmak, veriyi indekslemek ve analiz etmek için kullanılır. Elasticsearch gibi Solr da dağıtık bir yapıya sahip olup çok hızlı sonuçlar sunar. Ancak Solr, daha fazla yapılandırma ve özelleştirme imkanı sunarken Elasticsearch biraz daha kullanıcı dostudur.

### Nerelerde Kullanılırlar?

- **ELK Stack** genellikle log yönetimi ve analizinde, sistem izleme, güvenlik olaylarının analizi ve iş zekası projelerinde tercih edilir.
- **Solr** ise büyük veri kümelerinde hızlı arama ihtiyaçlarında (örneğin e-ticaret ürün arama özellikleri) yaygın olarak kullanılır.

### Büyük Veri ve İş Zekası ile İlgisi:

- **Büyük Veri:** ELK ve Solr, büyük miktardaki yapılandırılmış ve yapılandırılmamış veriyi hızla işleyip analiz ederek büyük veri işleme süreçlerine katkıda bulunur. Örneğin, sunucu log'larının işlenmesi veya sosyal medya verilerinin analiz edilmesi gibi.
- **İş Zekası:** ELK ve Solr, iş zekası süreçlerinde kullanıcıya daha anlamlı veri sunmak için kullanılır. ELK Stack, veriyi gerçek zamanlı izleyerek anomali tespiti yapabilir, iş süreçlerini optimize edebilir. Kibana gibi araçlarla görselleştirilen veriler, iş kararları için kullanıcılara kolay anlaşılır grafikler sağlar.

### Örnek Senaryo: E-ticaret Sitesinde Kullanım

Bir e-ticaret sitesinin trafiğini ve müşteri davranışlarını analiz etmek istediğinizi düşünelim. İşlemler şöyle ilerleyebilir:

1. **Veri Toplama (Logstash):** Kullanıcıların site üzerindeki tüm hareketleri (ürün aramaları, tıklamalar, satın almalar) log olarak kaydedilir. Logstash bu log verilerini toplayarak işlemeye başlar.
2. **Veri İndeksleme ve Arama (Elasticsearch veya Solr):** Bu log verileri Elasticsearch veya Solr kullanılarak indekslenir. Kullanıcıların siteyi nasıl kullandıkları hızlıca analiz edilebilir hale gelir. Örneğin, en çok aranan ürünler, kullanıcıların hangi sayfalarda zaman geçirdiği gibi bilgiler burada depolanır.
3. **Veri Görselleştirme ve Analiz (Kibana):** Kibana üzerinden çeşitli grafikler ve görseller oluşturulur. Satış trendleri, müşteri hareketleri gibi bilgiler anlık olarak raporlanır. Bu analiz,

kullanıcıların site üzerindeki yolculuklarını anlamak ve siteyi optimize etmek için iş zekası ekibine yol gösterir.

Bu senaryoda ELK Stack, büyük veri analitiği ve iş zekası için güçlü bir araç olarak görev yaparken, Solr ise kullanıcıların arama deneyimlerini iyileştirir.

## ELK ve Solr Arasındaki Benzerlikler ve Farklılıklar

### Benzerlikler

#### 1. Temel Arama ve İndeksleme Fonksiyonları:

- Hem **Elasticsearch** (ELK'in bir parçası) hem de **Solr** dağıtık arama ve indeksleme işlemleri için kullanılır. Her iki sistem de **Apache Lucene** altyapısını kullanır, bu da onları büyük veri kümelerinde hızlı ve etkili arama yapabilen güçlü araçlar haline getirir.
- **Örnek:** Bir e-ticaret sitesinde, kullanıcıların hızlı ürün aramaları yapmasını sağlamak için her iki sistem de kullanılabilir. Milyonlarca ürün arasından belirli özelliklere sahip ürünleri hızlıca filtreleyip arama sonuçlarını döndürebilir.

#### 2. Yapılandırılmamış Veriyi İşleyebilme:

- İki sistem de metin bazlı arama ihtiyaçlarını karşılamak üzere tasarlanmıştır ve yapılandırılmamış büyük veri ile çalışabilirler.
- **Örnek:** Bir haber sitesinde, makaleler içinde yapılan kelime aramaları için hem Solr hem de Elasticsearch kullanılabilir. Örneğin, "yapay zeka" terimi arandığında her iki sistem de o terimin geçtiği makaleleri hızlıca bulabilir.

#### 3. Dağıtık Yapı:

- Hem Elasticsearch hem de Solr, dağıtık bir yapıya sahip olup veri replikasyonu ve ölçeklenebilirlik imkanı sunar. Böylece büyük veri kümeleri üzerinde yüksek performansla çalışabilirler.
- **Örnek:** Yoğun trafiğe sahip bir sosyal medya platformunda, kullanıcının önceki gönderileri veya yorumlarını hızlıca aramak için her iki sistem de kullanılabilir ve performansları büyük veri altında korunabilir.

### Farklılıklar

#### 1. Kullanım Kolaylığı ve Yapılandırma:

- **Elasticsearch** daha kullanıcı dostu ve kurulum, yapılandırma açısından daha basit bir yapıya sahiptir. JSON API'leri ile kolayca entegre edilebilir. Ayrıca varsayılan ayarları kullanıcıları yönlendirdiğinden, çok fazla özelleştirme yapmadan dahi verimli çalışabilir.
- **Solr**, daha fazla yapılandırma ve özelleştirme seçeneği sunar ancak bununla birlikte biraz daha karmaşıktır. XML tabanlı bir konfigürasyon yapısına sahip olduğundan detaylı konfigürasyonlar ve daha yüksek bir öğrenme eğrisi gerektirebilir.
- **Örnek:** Basit bir ürün arama motoru kurmak istiyorsanız, Elasticsearch daha hızlı bir çözüm sunarken, özelleştirilmiş filtreleme ve karmaşık sorgulamalar için Solr tercih edilebilir.

## 2. Log ve Analitik İhtiyaçlar:

- **Elasticsearch**; log yönetimi ve analitik işlemler için ELK Stack'in bir parçası olarak oldukça yaygındır. Logstash ile veri toplama, Kibana ile veri görselleştirme kolayca entegre çalışır. Bu yüzden ELK, özellikle sistem log yönetimi ve izleme işlerinde tercih edilir.
- **Solr**, bu tür log ve analitik ihtiyaçlara doğrudan odaklanmaz, daha çok metin aramaları ve arama motoru işlevselliği için optimize edilmiştir.
- **Örnek:** Bir IT şirketi, sistem loglarını analiz etmek ve görselleştirmek için ELK Stack'i tercih edebilirken, yalnızca doküman tabanlı bir arama platformu kurmak isteyen bir firma Solr'u kullanabilir.

## 3. Arama İşlevsellikleri ve Özelleştirme:

- **Solr**, facet (kategori veya filtreleme) özellikleri ve belge bazında gelişmiş özelleştirmeler için daha esnektir. Özellikle kompleks arama gereksinimleri ve kullanıcı filtreleme ihtiyaçları olan uygulamalarda daha iyi performans sunabilir.
- **Elasticsearch** ise daha hızlıdır ve filtreleme ya da analiz işlemlerinde yüksek performans sergiler. Fakat bu özellikler Solr'daki kadar gelişmiş olmayabilir.
- **Örnek:** Bir e-ticaret sitesinde, müşterilerin ürün aramalarını kategori, marka ve fiyat aralığı gibi detaylara göre filtreleyebilmesi için Solr'un facet özelliği daha güçlü bir çözüm sunar.

## 4. Topluluk Desteği ve Ekosistem:

- **Elasticsearch** son yıllarda daha geniş bir topluluğa ve daha büyük bir ekosisteme sahiptir. Özellikle ELK Stack'in popüleritesi sayesinde daha fazla belge, eğitim kaynağı ve topluluk desteği mevcuttur.
- **Solr**, biraz daha sınırlı bir kullanıcı kitlesine hitap eder, ancak uzun süredir kullanımda olduğu için daha deneyimli bir geliştirici kitlesine sahiptir.

### Özet Bir Senaryo: ELK ve Solr Karşılaştırması

Bir medya şirketi, kullanıcıların içerik arayabileceği bir sistem ve aynı zamanda kullanıcı davranışlarını analiz edebileceği bir izleme sistemi kurmak istiyor.

- **Elasticsearch (ELK Stack):** Kullanıcı davranışlarının (hangi makaleleri okudukları, ne kadar süre kaldıkları) loglarını analiz etmek ve görselleştirmek için ELK Stack kullanılabilir. Böylece içerik tercihleri hakkında görsel raporlar alınabilir.
- **Solr:** Aynı içeriklerin hızlı bir arama motoru oluşturulması için Solr kullanılabilir. Kullanıcılar Solr üzerinden belirli yazar, kategori ya da konu başlıklarına göre içerik arayabilir.

Bu senaryoda ELK, analitik ve raporlama işlerinde etkin rol oynarken, Solr içerik aramaları için daha verimli olacaktır.

---

**Cassandra, MongoDB, HBase ve Couchbase gibi popüler NoSQL veritabanları arasında önemli farklılıklar** bulunmaktadır. Aşağıda bu NoSQL sistemleri arasındaki temel farkları açıklıyorum:



### 1. Veri Modeli:

- Cassandra: Sütun aile (column family) tabanlı bir veri modeli kullanır. Verileri sütunlar ve satırlar şeklinde depolar.
- MongoDB: Doküman (document) tabanlı bir veri modeli kullanır. Veriler JSON-benzeri dokümanlar halinde saklanır.
- HBase: Hadoop altyapısını kullanan bir sütun aile veri tabanıdır.
- Couchbase: Doküman tabanlı bir veri modeli kullanır, ancak performans için bir bellek içi (in-memory) bileşen sunar.

### 2. Veri Dağıtımı ve Çoğaltma:

- Cassandra: Simetrik, eşit dağılımlı düğümlere sahiptir. Veri çoğaltması, yüksek kullanılabilirlik ve dayanıklılık sağlar.
- MongoDB: Öncelikle tek bir ana düğüme (primary) sahiptir ve çoğaltma için ikincil düğümler kullanır.
- HBase: Hadoop Dağıtık Dosya Sistemi (HDFS) üzerinde çalışır ve veri çoğaltması HDFS tarafından sağlanır.
- Couchbase: Eşit dağılımlı düğümlere sahiptir ve veri çoğaltmasını kendi içinde sağlar.

### 3. Veri Tutarlılığı:

- Cassandra: Genellikle "Kullanılabilirlik ve Parçalanma Toleransı" (AP) prensibini benimser.
- MongoDB: Varsayılan olarak "Tutarlılık, Kullanılabilirlik ve Parçalanma Toleransı" (CP) prensibini benimser, ancak gevşek tutarlılık (eventual consistency) da sağlar.
- HBase: ACID özelliklerine sahiptir ve "Tutarlılık ve Kullanılabilirlik" (CP) prensibini benimser.
- Couchbase: Gevşek tutarlılık (eventual consistency) sağlar ve "Kullanılabilirlik ve Parçalanma Toleransı" (AP) prensibini benimser.

### 4. Sorgu Dilleri:

- Cassandra: CQL (Cassandra Query Language) kullanır, SQL benzeridir.
- MongoDB: Kendi sorgulama dilini (MQL) kullanır.
- HBase: Temelde Hadoop ekosisteminde kullanılan Apache Phoenix ve Apache Spark gibi araçları kullanır.
- Couchbase: N1QL sorgu dilini kullanır, SQL benzeridir.

### 5. Kullanım Alanları:

- Cassandra: Gerçek zamanlı uygulamalar, log analizi, IoT, finansal sistemler
- MongoDB: Genel amaçlı uygulamalar, içerik yönetim sistemleri, mobil uygulamalar
- HBase: Büyük veri uygulamaları, gerçek zamanlı analizler, IoT
- Couchbase: Eticaret, mobil uygulamalar, oyunlar, IoT

Bu karşılaştırma, NoSQL veritabanları arasındaki temel farklılıkları özetlemektedir. Uygulamanın gereksinimleri doğrultusunda, bu özellikler göz önünde bulundurularak en uygun NoSQL çözümü seçilebilir.