📌 Apache Kafka Nedir?

Apache Kafka, gerçek zamanlı olarak büyük veri akışlarını yönetmek için geliştirilmiş dağıtık bir mesajlaşma sistemidir. Yüksek performanslı, esnek ve dayanıklıdır.

Kafka Ne İşe Yarar?

- Gerçek zamanlı veri akışlarını almak, saklamak ve dağıtmak
- Uygulamalar arasında asenkron mesajlaşma sağlamak
- Mikroservis mimarilerinde event-based haberleşme kurmak

Kafka'nın Temel Bileşenleri

- **Producer:** Kafka'ya veri (mesaj) gönderen uygulama
- **Consumer:** Kafka'dan mesaj okuyan uygulama
- **Topic:** Mesajların mantıksal kategorilere ayrıldığı yapı
- **Partition:** Topic'lerin bölünerek paralel işlenmesini sağlar
- **Broker:** Kafka'nın çalıştığı sunucu (her biri bir node)

Kafka Nasıl Çalışır?

1. Producer bir topic'e mesaj gönderir.
2. Kafka bu mesajı partition'lara yazar.
3. Consumer'lar bu mesajı okur.
4. Kafka mesajları diske yazar, istenirse tekrar oynatılabilir.

Kafka'nın Avantajları

- **Ölçeklenebilir:** Veriyi partition'lar ile dağıtarak paralel çalıştırır.
- **Hızlı:** Yüksek throughput ile milyonlarca mesaj işleyebilir.
- **Dayanıklı:** Disk tabanlı yapı, mesaj kaybına karşı koruma sağlar.
- **Esnek:** Kafka Streams, ksqlDB gibi araçlarla veri işleme yapılabilir.

Kafka API'leri

API	Açıklama
Producer API	Veri yayınlamak için kullanılır
Consumer API	Veri tüketmek için kullanılır
Streams API	Gerçek zamanlı veri işlemek için
Connect API	Kafka ile başka sistemleri bağlamak için

Kafka Kullanım Senaryoları

- Sipariş yönetimi (e-ticaret)
- Log toplama (monitoring)
- Finansal işlem izleme
- Gerçek zamanlı analiz (stream analytics)
- Mikroservis haberleşmesi
-

📌 RabbitMQ Nedir?

RabbitMQ, uygulamalar arasında veri iletimi sağlamak için kullanılan, **açık kaynaklı, mesaj tabanlı (message broker)** bir sistemdir. Özellikle **görev sıralama (task queue)**, **arka plan işlemleri**, ve **asenkron işlem yönetimi** gibi durumlar için tercih edilir.

Temel Kavramlar

Kavram	Açıklama
Producer	Mesaj üreten ve RabbitMQ'ya gönderen uygulamadır.
Consumer	Mesajları kuyruktan okuyan uygulamadır.
Queue	Gelen mesajların sıralandığı bekleme alanıdır.
Exchange	Mesajların hangi kuyruklara yönlendirileceğini belirler.
Routing Key	Exchange ve Queue arasındaki bağlantıyı tanımlar.

RabbitMQ Nasıl Çalışır?

1. Producer, RabbitMQ'ya bir mesaj gönderir.
2. Bu mesaj **Exchange** üzerinden belirli kurallara göre **Queue**'lara yönlendirilir.
3. Consumer, kuyruktaki mesajları sırasıyla alır ve işler.
4. Mesaj tüketildikten sonra (acknowledge edilir) kuyruktan silinir.

RabbitMQ Kullanım Senaryoları

- Kullanıcıya e-posta / SMS gönderiminde arka planda kuyruk oluşturma
- Uzun süren işlemleri servis dışına alma (örneğin video işleme)
- Bir işlemi birden çok consumer'a bölerek paralel işleme
- Asenkron iş akışı tasarımı (örneğin fatura oluşturma)

Kafka ile RabbitMQ Arasındaki Farklar

Özellik	RabbitMQ	Kafka
Mesaj Saklama	Tüketildikten sonra silinir	Diskte uzun süre saklanabilir
Tüketim Modeli	Push (mesajlar consumer'a gönderilir)	Pull (consumer mesajları kendisi çeker)
Protokol	AMQP (Advanced Message Queuing Protocol)	Kendi protokolü (TCP tabanlı)
Yaygın Kullanım Alanı	Queue tabanlı task management	Stream processing, real-time analytics
Geriye Dönük Okuma	❌	✅ (offset ile yeniden okunabilir)
Hız/Throughput	Orta seviye	Çok yüksek (milyonlarca msg/sn)
Öğrenme Eğrisi	Daha basit	Daha karmaşık, yüksek esneklik

Özet

RabbitMQ, özellikle **mesaj sıralama**, **görev dağıtımı**, ve **arka plan işlemleri** için uygun, **hafif ve esnek** bir mesajlaşma sistemidir. Kafka ile kıyaslandığında daha basit ve klasik iş yükleri için idealdir.

log4net Nedir?

log4net, .NET uygulamalarında log (kayıt) yazmak için kullanılan bir kütüphanedir. Uygulama içi olayların izlenmesini sağlar.

log4net Ne İçin Kullanılır?

- Hataları tespit etmek ve loglamak
- Performans ölçümü yapmak
- Kullanıcı davranışlarını kaydetmek
- Uygulama içi süreçleri belgelemek

Örnek log4net Yapılandırması

```
<log4net>
  <appender name="FileAppender" type="log4net.Appender.FileAppender">
    <file value="Logs/transactions.log" />
    <appendToFile value="true" />
    <layout type="log4net.Layout.PatternLayout">
      <conversionPattern value="%date %-5level %message%newline" />
    </layout>
  </appender>
  <root>
    <level value="INFO" />
    <appender-ref ref="FileAppender" />
  </root>
</log4net>
```

C# Örnek Kullanım

```
private static readonly ILog log = LogManager.GetLogger(typeof(YourClass));

try {
    log.Info("İşlem başladı");
    // işlem...
    log.Info("İşlem tamamlandı");
}
catch (Exception ex) {
    log.Error("Hata oluştu", ex);
}
```

Log4net ile Takip Edilebilecekler

- İş akışı takibi
- Performans süreleri
- Güvenlik olayları (audit log)
- API çağrıları ve dış servis cevapları
- Kullanıcı etkileşimleri

Özet

Kafka, büyük verinin gerçek zamanlı akışını sağlarken
log4net, bu sistemlerin her adımını kayıt altına alır.
Birlikte kullanıldığında, sisteminiz hem **ölçeklenebilir**, hem de **izlenebilir** hale gelir.

📌 Docker Nedir?

Docker, uygulamaları ve bağımlılıklarını birlikte içeren **hafif, taşınabilir yazılım paketleri (container)** oluşturmanı sağlayan bir konteyner teknolojisidir.

Docker'ın Özellikleri:

- Her şey **container** içinde çalışır (uygulama, kütüphane, ayar)
- Her ortamda aynı şekilde çalışır (developer → test → prod)
- Sanal makineye göre **daha hızlı ve hafiftir**
- Tek bir makinede birden çok container çalıştırabilirsin

📌 Kubernetes Nedir?

Kubernetes (K8s), birden fazla Docker container'ını **otomatik olarak dağıtan, yöneten ve ölçeklendiren** bir **container orkestrasyon sistemidir**. Google tarafından geliştirilmiştir.

Kubernetes'in Özellikleri:

- Çok sayıda container'ı **birlikte yönetir**
- **Otomatik yeniden başlatma, yük dengeleme, self-healing** sağlar
- **Servis keşfi**, **gizli veri yönetimi**, **CI/CD entegrasyonu** içerir
- Docker container’larını **küme (cluster)** olarak çalıştırır

Docker vs Kubernetes: Karşılaştırma Tablosu

Özellik	Docker	Kubernetes
Amaç	Container oluşturmak ve çalıştırmak	Container'ları yönetmek ve ölçeklemek
Yapı	Tekil container veya docker-compose	Dağıtılmış container kümesi (cluster)
Kullanım Alanı	Uygulama paketleme ve çalıştırma	Production ortamda dağıtım, yedeklilik, scaling
Ölçeklenebilirlik	Elle yapılır	Otomatik ve yatay/dengelemeli
Gözlemleme/ Loglama	Kısıtlı	Built-in monitoring ve health-check destekli
Durum Yönetimi	Yok	Var (örn. yeniden başlatma, rollout yönetimi)
Network Yönetimi	Temel	Hizmet yönlendirme, DNS, load balancer
Öğrenme Eğrisi	Kolay	Orta/Zor (daha karmaşık yapıya sahip)
Kullanım Birlikteliği	Kubernetes altında çalıştırılır	Docker container'ları olmadan çalışmaz

Peki Hangisi Ne İçin Kullanılır?

- **Docker**:
  → Geliştirme sürecinde uygulamayı izole edip her yerde çalışmasını sağlamak için idealdir.

- **Kubernetes**:
  → Gerçek dünya **production** ortamlarında onlarca/yüzlerce container'ı **yönetmek, dağıtmak, ölçeklendirmek** için kullanılır.

Özet:

Docker, uygulamanın nasıl paketleneceğini çözer.
Kubernetes, bu paketlerin nasıl dağıtılacağını ve yönetileceğini çözer.

Birlikte kullanıldığında:

- Docker → container'ı oluşturur
- Kubernetes → bu container'ları akıllıca yönetir.