**Redis Kullanım Alanlar ve Örnekler**

1. **Web Uygulamalarında Önbellek (Cache) Olarak Kullanımı Redis, web uygulamalarında hız ve verimlilik sorunlarıyla mücadele etmek için sıklıkla kullanılır. Örneğin:**

Büyük bir e-ticaret sitesi düşünelim. Bu site, Black Friday döneminde yoğun bir trafikle karşı karşıya kalıyor. Redis'i önbellek olarak kullanarak:

* Ürün detayları, fiyatlar ve stok bilgileri gibi sık erişilen verileri Redis'te saklayabilirler. Böylece her istek için veritabanına gitmek yerine, bu bilgileri hızlıca Redis'ten çekebilirler
* Online olan kullanıcı sepetlerini Redis'te tutarak, kullanıcıların alışveriş deneyimini hızlandırabilirler. Örneğin, bir kullanıcı ürün eklediğinde veya çıkardığında, bu değişiklikler anında Redis'e yazılır ve okunur.
* Sayfa önbellekleme yaparak, özellikle ana sayfa, kategori sayfaları gibi sık ziyaret edilen sayfaların yüklenme sürelerini önemli ölçüde azaltabilirler.

**2. Gerçek Zamanlı Analitik ve Loglama**

Örnek: Büyük bir sosyal medya platformu düşünelim. Bu platform, kullanıcı etkileşimlerini gerçek zamanlı olarak analiz etmek istiyor.

* Her bir gönderi için beğeni, yorum ve paylaşım sayılarını Redis'te tutabilirler. Böylece, bir gönderinin popülerliğini anlık olarak takip edebilir ve trend olan içerikleri hızlıca belirleyebilirler.
* Kullanıcıların platform üzerindeki hareketlerini (tıklamalar, sayfa görüntülemeleri, vb.) Redis'e kaydederek, kullanıcı davranışlarını gerçek zamanlı olarak analiz edebilirler.
* Sistem hatalarını ve performans metriklerini Redis'e loglayarak, anormal durumları hızlıca tespit edebilir ve müdahale edebilirler.

**3. Mesajlaşma ve Pub/Sub Sistemleri:**Örnek: Çok oyunculu bir online oyun düşünelim. Bu oyun, oyuncular arasında gerçek zamanlı iletişim ve etkileşim gerektiriyor.

* Oyuncular arasındaki chat mesajlarını Redis'in pub/sub özelliğini kullanarak iletebilirler. Böylece, bir oyuncu mesaj gönderdiğinde, bu mesaj anında diğer ilgili oyunculara iletilebilir.
* Oyun içi etkinlikleri (örneğin, bir oyuncunun önemli bir başarı elde etmesi) Redis üzerinden yayınlayarak, tüm ilgili oyunculara anında bildirim gönderebilirler.
* Oyun sunucuları arasında hızlı veri senkronizasyonu sağlayabilirler. Örneğin, bir oyuncunun konumu veya envanteri değiştiğinde, bu bilgi Redis üzerinden diğer sunuculara hızlıca iletilebilir.

**RabbitMQ Kullanım Alanlar ve Örnekler**

**1. E-ticaret Sistemlerinde Sipariş İşleme**

Örnek: Büyük bir online market zinciri düşünelim. Bu market, özellikle yoğun dönemlerde çok sayıda sipariş alıyor.

* Siparişler alındığında, RabbitMQ kuyruğuna eklenir. Böylece, sistem yoğun olsa bile hiçbir sipariş kaybolmaz.
* Farklı işlem adımları (ödeme onayı, stok kontrolü, kargo hazırlığı) için ayrı kuyruklar oluşturulur. Her adım tamamlandıkça, sipariş bir sonraki kuyruğa aktarılır.
* Örneğin, bir müşteri sipariş verdiğinde, bu sipariş önce "ödeme onayı" kuyruğuna gider. Ödeme onaylandıktan sonra "stok kontrolü" kuyruğuna, oradan da "kargo hazırlığı" kuyruğuna geçer. Bu sayede, her adım bağımsız ve ölçeklenebilir şekilde işlenir.

**2. IoT (Nesnelerin İnterneti) Veri Yönetimi**

Örnek: Büyük bir tarım işletmesi düşünelim. Bu işletme, tarlalarındaki sensörlerden gelen verileri analiz ederek sulama ve gübreleme kararları alıyor.

* Tarladaki yüzlerce sensör, toprak nemi, sıcaklık, pH değeri gibi verileri sürekli olarak RabbitMQ'ya gönderir.
* Bu veriler, RabbitMQ üzerinden farklı analiz sistemlerine dağıtılır. Örneğin, sulama sistemi kendi kuyruğundan nem verilerini alırken, gübreleme sistemi pH değerlerini takip eder.
* Merkezi kontrol sistemi, analiz sonuçlarına göre sulama veya gübreleme komutlarını yine RabbitMQ üzerinden ilgili sistemlere iletir.

**3. Mikroservis Mimarisinde Servisler Arası İletişim**

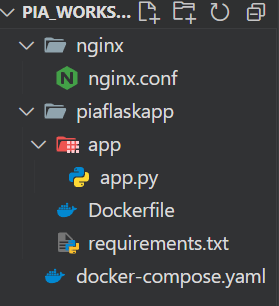
Örnek: Büyük bir haber sitesi düşünelim. Bu site, mikroservis mimarisi kullanarak farklı işlevleri bağımsız servislere ayırmış.

* Yeni bir haber girildiğinde, "İçerik Yönetim Servisi" bu haberi RabbitMQ'ya gönderir.
* "Bildirim Servisi" bu mesajı alarak, ilgili kullanıcılara push notification gönderir.
* "Önbellek Yönetim Servisi" aynı mesajı alarak, ilgili sayfaların önbelleğini günceller.
* "SEO Optimizasyon Servisi" haberi alarak, gerekli meta etiketlerini oluşturur ve site haritasını günceller.

**Docker Workshop**

**1. Dosya Yapısı:**

Dosyalar ve uygulama kodlar ıaşağıdaki gibi kurgulanmıştır. Python Flask app ve nginx configuration



1. **Dockerfile ve NGINX Config dosyası:**

Python flask uygulamasındaki Dockerfile aşağıdaki gibidir. Uygulama basitlik için debug modda 5000 porttan expose olmuştur  
  
FROM python:**3.9-slim-buster**

WORKDIR /app

COPY requirements.txt .

RUN pip install --no-cache-dir -r requirements.txt

COPY . .

ENV FLASK\_APP=app.app

ENV FLASK\_DEBUG=1

ENV FLASK\_RUN\_HOST=0.0.0.0

CMD ["python", "-m", "flask", "run", "--host", "0.0.0.0"]

NGINX config:  
  
events {

    worker\_connections 1024*;*

}

http {

    upstream flask\_app {

        server flask\_app:5000*;*

    }

    server {

        listen 80*;*

        server\_name localhost*;*

        location / {

            proxy\_pass http://flask\_app*;*

            proxy\_set\_header Host $host*;*

            proxy\_set\_header X-Real-IP $remote\_addr*;*

        }

    }

}

Son olarak PostgreSQL dahil edilmiş Docker Compose yaml dosyası:  
services:

  postgres:

    image: postgres:13-alpine

    container\_name: postgres\_db

    environment:

      POSTGRES\_USER: ecommerceuser

      POSTGRES\_PASSWORD: ecommercepasswd

      POSTGRES\_DB: ecommercedb

    volumes:

      - postgres\_data:/var/lib/postgresql/data

    networks:

      - app\_network

  flask\_app:

    build:

      context: ./piaflaskapp

      dockerfile: Dockerfile

    container\_name: flask\_app

    depends\_on:

      - postgres

    environment:

      - FLASK\_APP=app.app

      - FLASK\_RUN\_HOST=0.0.0.0

    ports:

      - "5000:5000"

    networks:

      - app\_network

  nginx:

    image: nginx:alpine

    container\_name: nginx\_server

    ports:

      - "80:80"

    volumes:

      - ./nginx/nginx.conf:/etc/nginx/nginx.conf:ro

    depends\_on:

      - flask\_app

    networks:

      - app\_network

volumes:

  postgres\_data:

networks:

  app\_network:

    driver: bridge