Dağıtık Sistemler

Proje Hakkında

Bu proje, modern yazılım geliştirme ve dağıtık sistemler mimarileri hakkında derinlemesine bilgi edinmek amacıyla tasarlanmıştır. Docker Compose kullanılarak, Nginx, Spring Boot uygulamaları, PostgreSQL ve Redis container ortamlarında yapılandırılmıştır. Ayrıca, Nginx yapılandırması ile yüksek erişilebilirlik sağlanmış ve bir sunucunun arızalanması durumunda diğer sunucunun devreye girmesi sağlanmıştır.

Projemi çalıştırmak için kullandığım komutlar;

1. `mvn clean package`

- **Açıklama:** Bu komut, Spring Boot projenizin derlenmesi ve paketlenmesi için kullanılır. Maven, projeyi temizler ve ardından projeyi `.jar` ya da `.war` formatında paketler.
- **Projede Kullanımı:** Bu komut, uygulamanızın en güncel versiyonunun çalıştırılabilir hale getirilmesini sağlar. Docker container'larını başlatmadan önce uygulamanızın doğru şekilde paketlendiğinden emin olmak için bu komutu çalıştırmanız önemlidir.

2. `docker-compose up --build`

- **Açıklama:** Docker Compose, çoklu container'ları aynı anda yönetmek için kullanılan bir araçtır. Bu komut, tüm container'ları başlatır ve `--build` parametresi sayesinde container'lar önceden var ise bile, yeni yapılandırmalara göre yeniden inşa edilir.
- **Projede Kullanımı:** Docker Compose, projenizdeki tüm servisleri (Nginx, Spring Boot uygulamaları, PostgreSQL, Redis) bir arada çalıştırmanızı sağlar. Bu komut, her şeyin en güncel haliyle başlatılmasını garantiler ve container'lar arasındaki bağlantıları oluşturur.

3. `docker-compose stop app1`

- **Açıklama:** Bu komut, `docker-compose.yml` dosyasındaki `app1` adlı servisi durdurur.
- **Projede Kullanımı:** Projenizdeki `app1` adlı Spring Boot uygulama sunucusunu geçici olarak durdurmak için kullanılır. Eğer sunucu üzerinde bir değişiklik yaptıysanız ya da bir güncelleme yapmanız gerekirse, geçici olarak bu servisi durdurabilirsiniz.

4. `docker-compose start app1`

- **Açıklama:** Bu komut, `docker-compose.yml` dosyasındaki `app1` adlı servisi yeniden başlatır.
- **Projede Kullanımı:** `app1` servisi durdurulduktan sonra tekrar çalıştırılmasını sağlar. Bu, herhangi bir değişiklik yapıldıktan sonra servisi yeniden başlatmak için kullanılabilir.

5. `docker-compose stop app2`

- **Açıklama:** Bu komut, `app2` adlı servisi durdurur.
- **Projede Kullanımı:** Projeye dahil olan ikinci Spring Boot uygulama sunucusunun (örneğin, yük dengeleme yapılacak ikinci sunucu) durdurulması için kullanılır. Bu, uygulama üzerinde geçici bir bakım veya güncelleme yaparken faydalıdır.

6. `docker-compose start app2`

- **Açıklama:** Bu komut, `docker-compose.yml` dosyasındaki `app2` adlı servisi tekrar başlatır.
- **Projede Kullanımı:** `app2` servisi durdurulduğunda, yeniden başlatmak için bu komut kullanılır. Bu, bir servisin geçici olarak durdurulup tekrar başlatılmasını sağlar, özellikle failover mekanizmalarında kullanılabilir.

7. `docker-compose down`

- **Açıklama:** Bu komut docker container'larını durdurmak için kullanılır.

`8.docker ps`

- **Açıklama:** Bu komut çalışan container'ları listeler.

Proje Bağlamında:

- **Docker Compose** kullanarak tüm servislerinizi (Nginx, Spring Boot uygulamaları, PostgreSQL ve Redis) container'lar içerisinde izole ederek çalıştırıyorsunuz. Bu komutlar, bu container'ları yönetmek ve her bir servisi durdurmak ya da başlatmak için kullanılır.
- **Failover** senaryosunda, Nginx yapılandırması sayesinde bir sunucu arızalandığında, diğer Spring Boot sunucusuna yönlendirme yapılır. Bu senaryolarda, `docker-compose stop` ve `docker-compose start` komutları, sunucuları yeniden başlatmak ya da durdurmak için kullanışlıdır.

Bu şekilde Docker Compose komutları, projenizin dağıtık yapısını yönetmenize ve servislere müdahale etmenize olanak tanır.

Ön Koşullar

- Docker Compose yüklü olmalı.

Uygulamaya Erişim İçin

- Uygulamaya http://localhost:8090/api/test bağlantısıyla erişim sağlanır.