API Gateway'lerin Yatayda Olceklendirilmesi

Ö ncelikle ölçeklendirme yontemlerinden bahsetmek istioyrum. Dikeyde ölçeklendirme, bir programın çalıştığı makinede donanım aygıtlarını yükselterek sağlanır. Örnek vermek gerekirse, sunucunun çalıştığı makinede bulunan 8 GB RAM'e ek olarak 8 GB RAM daha eklemek dikey ölçeklendirme yöntemine girer.

Yatayda ölçeklendirme ise çalışan programın sayısını artırarak performans artışı sağlamaktır. Örneğin, bir mail servisimiz olduğunu düşünelim. Bu servis yetersiz kaldığında, aynı servisten bir tane daha dağıtarak (deploy) ve yönlendirmeyi bir mesaj kuyruğu (message queue) veya başka bir yöntemle ayarlayarak yükü iki servise bölmüş oluruz. Böylece performans artışı sağlanır. İşte bu yönteme yatay ölçeklendirme denir.

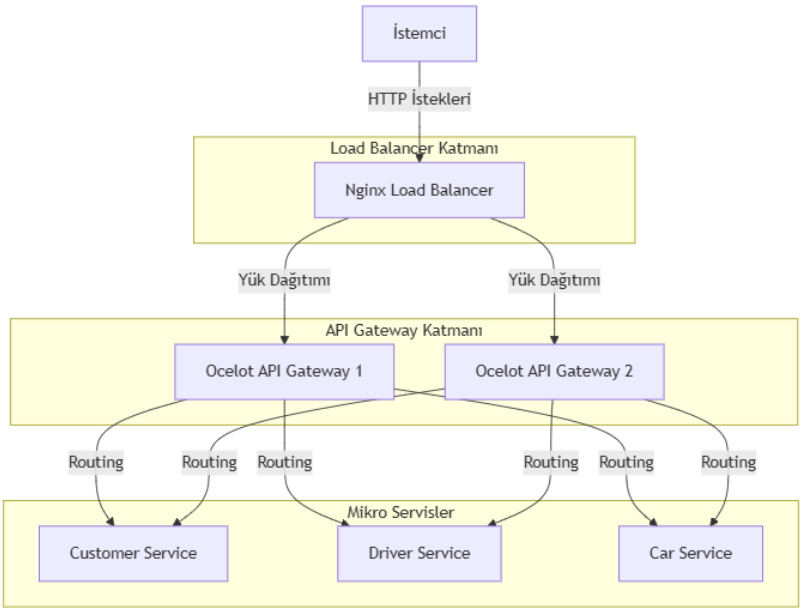
API Gatewayler mikro servis mimarilerinde birçok işlevleri olmasının yanında load balancer özelliği sayesinde servislerimizi ölçeklendirme konusunda güçlü araçlardır, peki API Gatewayler nasıl yatayda ölçeklendirilebilir? Problemin doğası yüzünden servisleri ölçeklendirmekten daha karmaşık bir problem ama yine de birkaç çözüm var. Birincisi, API Gatewaylerin sorumluluklarının bölünmesi. Mesela belirli servisler belirli bir gateway altındayken, belirli servisler başka bir gateway altında tutulabilir. Ama bu yapıldığında gatewayler üzerinde farklı yapılandırmalar yapılması gerekecek ve yatayda her ölçeklendirmek istediğimizde var olan gatewaylerin yapılandırmasını değiştirmemiz gerekecek. Daha uygun çözümler ise load balancer kullanmak veya DNS sağlayıcıları üzerinden ayarlamalar yaparak bölgesel yönlendirme gibi yöntemler kullanmak.

Ben projemde lokalimde kurduğum Nginx (lokalde kurulması ve ayarlanması kolay olduğu için) load balancer kullanmaya karar verdim. Bu Nginx, gelen istekleri API Gatewaylerin aksine yönlendirmeyi Application Layer'da değil, Transport Layer'da yapıyor ve dolayısıyla isteklerin içeriğini analiz etmeden doğrudan ayarlanan load balancing algoritmasını kullanarak yönlendirme yapıyor.

Burada biraz da Load Balance algoritmalarından bahsetmek istiyorum. Bazı popüler olanlar:

- Round Robin (İsteği sürekli bir sonraki gateway'e yönlendiriyor. Birinci istek birinci gateway'e, ikinci ikinciye, üçüncü yeniden birinciye gibi.)
- IP Hash (İstemci IP'sini hashleyerek bulduğu değere göre yönlendirme yapıyor. Aynı IP adresinden gelen her istek aynı yere yönlendiriliyor.)
- Least Connections (En az bağlantı olan sunucuya yönlendiriyor.)

Ben projemde Round Robin algoritmasını kullanmaya karar verdim.



Nginx Load Balanceri ayarlamak icin nginx.conf dosyasi duzenlenmeli, benim projemdeki config icerigi su sekilde:

events {}

stream {

upstream backend_servers {

server localhost:6001; # Ocelot Gateway 1

server localhost:6002; # Ocelot Gateway 2

}

server {

listen 8080; # NGINX, 8080 portunda TCP trafiğini dinler

```
        proxy_pass backend_servers;  # Gelen trafiği backend_servers'a yönlendirir
        }
}
```

Örnek bir istek senaryosu şu şekilde gerçekleşiyor; client, araba servisinden bilgileri almak için şu adrese istek atıyor: ***http://localhost:8080/cars***. Daha sonrasında Nginx Load Balancer, duruma göre isteği ***http://localhost:6001/cars*** (API Gateway 1) veya ***http://localhost:6002/cars*** (API Gateway 2)'e yönlendiriyor. Daha sonrasında ise API Gateway, bu isteği ***http://localhost:5001/api/cars*** (backend servisi) adresine yönlendiriyor ve yanıtı bu servisten alıyor.

Sistemi calistirmak icin github reposuna ekledigim solutionin icindeki tum projeleri run etmeniz gerekmektedir zaten startup olarak ayarladim hepsini dogrudan run butonuna tiklamaniz yeterli. Sonrasinda Nginx'i run etmeniz gerekmektedir bunun icin bir konsol acip yine git hub reposuna ekledigim nginx-1.26.3 kalsorune gidin ve su baslatma komutunu yazin, ***start nginx*** . *Bu komutu yazinca nginx arka planda calismaya baslayacak, durdurmak istediginizde ayni terminale* ***nginx -s stop*** *komutunu yazabilirsiniz*.

Sonrasinda test etmek icin su browser uzerinden su url'ler ile istek atabilirsiniz:

- *http://localhost:8080/cars*
- *http://localhost:8080/customers*
- *http://localhost:8080/drivers*

Burak Kaya