Mikroservis Mimarisi Nedir?

Mikroservis mimarisi, yazılım geliştirme dünyasında en çok konuşulan konulardan biridir. Bu yazıda mikroservislerin ne olduğunu öğreneceksiniz.

Mikroservisler Nedir?

Mikroservis sistemi, birçok büyük ölçekli yazılım projesinde kullanılan bir mimari stildir. Amaç; **verimliliği, güvenilirliği, performansı ve ölçeklenebilirliği artırmaktır**. Mikroservisleri, geleneksel yaklaşım olan **monolitik mimari** ile karşılaştıralım.

Monolitik Mimari

Monolitik mimari, tüm sistemin tek bir bütün olarak geliştirildiği yapıdır. Bu yaklaşım, küçük projeler için yönetimi kolay ve verimli bir stratejidir.
Monolitik mimarinin avantajları şunlardır:

- **Geliştirmesi kolaydır:** Tüm projeyi bir IDE'de veya basit bir metin düzenleyicide açabilirsiniz.
- **Çalıştırması kolaydır:** IDE'de "çalıştır" tuşuna basarak test etmeye başlayabilirsiniz.
- **Dağıtımı kolaydır:** Sistemi tek bir dosya (örneğin Java uygulamasında bir WAR dosyası) yükleyerek dağıtabilirsiniz.

Ancak bu avantajlar yalnızca **küçük sistemler** için geçerlidir. Sistem büyüdükçe sorunlar artar ve sonunda "**monolitik cehennem**" diye adlandırılan durum ortaya çıkar.

Monolitik Mimaride Ortaya Çıkan Sorunlar

Dağıtım (deployment) daha zor ve riskli hale gelir

Diyelim ki büyük bir hata buldunuz ve hemen düzeltmeniz gerekiyor. Hızla hatayı düzeltirsiniz, ancak şimdi tüm sistemi yeniden dağıtmanız gerekir! O sırada başka özellikler de geliştirilmekte olduğundan, tek bir hatayı düzeltirken birçok yeni hatayı sisteme sokma riski vardır.

"Büyük Patlama" (Big Bang) Alternatifi

Bazı ekipler bu sorunu "büyük patlama sürümleri" yaparak çözmeye çalışır. Yani tüm değişikliklerin ve yeni özelliklerin tek bir tarihte birleştirilip dağıtılması kararlaştırılır.
Bu daha güvenli görünse de büyük bir dezavantajı vardır: **Acil düzeltmeler (emergency fix)** yapılamaz. Proje büyüdükçe bu sürüm aralıkları uzar. Eğer yılda bir kez sürüm çıkıyorsanız, monolitik cehenneme hoş geldiniz!

Sistem çok büyük ve karmaşık hale gelir

Monolit o kadar büyüyebilir ki artık tek bir geliştirici sistemi anlayamaz hale gelir. Modüller arası sınırları korumak zorlaşır, bağımlılıklar birbirine girer ve sonunda "**Çamur Yumağı (Big Ball of Mud)**" dediğimiz yapı ortaya çıkar.

Monolit ölçeklenmesi zor hale gelir

Dağıtım birimi büyüdükçe tek seçenek donanımı büyütmektir (dikey ölçekleme). Ancak bu oldukça **verimsiz ve pahalıdır**. Örneğin belleği iki katına çıkarmak, maliyeti genellikle iki kattan daha fazla artırır. Ayrıca, donanım büyüklüğünün de bir sınırı vardır.

Alternatif: Mikroservis Mimarisi

Mikroservis yaklaşımında, sistem **birbirinden bağımsız küçük servislerin (mikroservislerin)** bir koleksiyonu olarak geliştirilir.
Her mikroservis:

- Ayrı bir ekip tarafından geliştirilir,
- Kendi kod deposunda (örneğin Git'te) barındırılır,
- İşe yarar bir işlevi yerine getirir,
- Bağımsız şekilde dağıtılabilir,
- Tek başına çalıştırılabilir ve test edilebilir,
- Diğer mikroservislerle iş birliği yaparak daha büyük bir amaca hizmet eder.

Mikroservislerin nasıl iletişim kurduğu başlı başına bir konudur. Basit **REST** servisleri olabilir ya da **asenkron mesajlaşma sistemleri** kullanılabilir.

Mikroservisler Nasıl İnşa Edilir?

1. Mikroservisler iş yeteneklerine göre düzenlenmelidir

Örneğin, bir e-ticaret sitesi yalnızca bir "alışveriş sepeti" değildir. Envanter, ürün kataloğu, faturalama, müşteriler, siparişler vb. işlevleri vardır.
Bu işlevlerden her biri bir mikroservis adayı olabilir. Örneğin "envanter yönetimi", diğer kriterleri de sağlıyorsa uygun bir mikroservis adayıdır.

2. Her mikroservis tek bir ekip tarafından geliştirilir, dağıtılır ve işletilir

Bir ekip mikroservisin tamamından sorumludur. Bu, mikroservisin fazla büyümesini engeller.
Amazon'un ünlü bir kuralı vardır: "Bir ekibin toplantısında iki pizza herkese yetebilmeli."

Bu yaklaşık olarak **6–8 kişilik ekipler** anlamına gelir. Eğer 8 kişilik bir ekip bir mikroservisi geliştiremiyorsa, o servis muhtemelen çok karmaşıktır ve parçalanmalıdır.

Ayrıca hiçbir zaman ilk seferde mükemmel bir mikroservis sınırı belirleyemezsiniz. Başarılı mikroservis projeleri **esneklik** üzerine kuruludur: Eğer bir mikroservis üzerinde çalışmak zorlaştıysa, onu iki veya daha fazla servise bölmek normaldir.

3. Mikroservisler yüksek içsel bağlılığa (high cohesion) sahip olmalıdır

Her mikroservis **tek bir işi iyi yapmalıdır**.
Örneğin, "envanter servisi" vergi hesaplamalarını da yapıyorsa, orada bir tasarım hatası vardır. Muhtemelen ayrı bir "vergi servisi" oluşturulmalıdır.
Bağımsız değişen parçalar ayrı mikroservislerde olmalıdır. Vergi oranları sık sık değiştiği için, her seferinde tüm envanter servisini güncellemek acı verici olurdu.

4. Mikroservisler gevşek bağlı (loosely coupled) olmalıdır

Servisler arasındaki bağımlılıklar minimumda tutulmalıdır. Bir servisteki değişiklik diğer servisleri mümkün olduğunca az etkilemelidir.

5. Mikroservisler bağımsız şekilde dağıtılabilmelidir

Örneğin, envanter servisinde bir hata buldunuz. Bu hatayı düzeltip yalnızca o servisi dağıtabilmelisiniz — diğer ekipleri beklemeden.

6. Mikroservislerde otomasyon şarttır

Monolitik bir uygulamayı manuel olarak dağıtmak belki tolere edilebilir. Ancak **yüzlerce dağıtım** ve **binlerce örnek (instance)** söz konusu olduğunda otomasyon zorunludur.
Bu nedenle mikroservis projelerinde **Sürekli Entegrasyon (CI)** ve **Sürekli Teslimat (CD)** süreçleri şarttır.

7. Mikroservisler genellikle dağıtık sistemlerdir

Tüm mikroservisleri tek bir sunucuya dağıtabilirsiniz, hatta her birini farklı bir sunucuya da koyabilirsiniz.
Ancak bu maliyetli olur. En iyi çözüm, her mikroservisi bir **Docker container**'ına dönüştürmek ve **Kubernetes** gibi bir orkestrasyon aracıyla yönetmektir.
Kubernetes, yük dengeleme ve replikasyon özellikleriyle yüksek dayanıklılık sağlar.

Mikroservis Mimarinin Avantajları

- Modülerliği, içsel bağlılığı ve gevşek bağımlılığı teşvik eder.
- Mikroservisler bağımsız olarak dağıtılabilir, bu da hızlı değişiklik ve yeni özellik eklemeyi sağlar.

- Her mikroservis farklı bir programlama diliyle yazılabilir (**polyglot programming**). Örneğin, finans servisi Python ile, yüksek performanslı bir grafik servisi C ile yazılabilir.
- Problemli bir mikroservis kolayca çöpe atılıp yeniden yazılabilir.
- Mikroservisler yatay olarak ölçeklendirilebilir. Bu da dikey ölçeklemeye göre çok daha ucuzdur.

Mikroservis Mimarinin Dezavantajları

- Prensipler basit olsa da, uygulaması zordur.
- Bazı kötü yönetilen projeler mikroservislere geçerek sorunlarını çözmeyi umar, ancak başarısız olunca suçu mikroservislere atarlar.
- Otomasyon, sürekli teslimat, kaynak kontrolü gibi birçok araç gerektirir.
- Mikroservisleri izole şekilde çalıştırmak ve test etmek zor olabilir.

Monolitik mi Mikroservis mi?

Eğer bir uygulama **tek bir ekip** tarafından geliştiriliyor ve işletiliyorsa, monolitik bir mimari makul bir seçimdir.
Ancak ekip büyüdükçe ve sistem karmaşıklaştıkça, **monolitik cehennem** kaçınılmaz hale gelir.
Yine de mikroservislere geçmek her şeyi otomatik olarak çözmez — bu geçiş dikkatli planlanmalıdır.

Bu, mikroservislerin kısa bir genel bakışıydı.
Daha fazlasını öğrenmek ister misiniz? Kariyerinize yatırım yapın, mikroservis kursuma katılın ve **mülakatlarda karşınıza çıkabilecek 25 soruya** hazırlanın.

Mikroservis yolculuğunuzun keyfini çıkarın!