1) **Conway'in Kanunu' nu (Conway's Law) açıklayınız.**
   1968 yılındaki modüler programlama sempozyumunda Melvin Conway tarafından ortaya atılan bir kural/gözlemdir.
   Sistemleri tasarlayan organizasyonlar ... kendi iletişim yapılarının birer kopyasını üretmekle sınırlıdır"
   Yaklaşım basitçe bir yazılımın geliştirilme sürecinde çok sayıdaki farklı modülün farklı kişiler tarafından geliştirilmesi süresinde yaşanan iletişimin yazılıma da yansımasından bahsetmektedir. Örneğin bir yazılımın ara-yüzünü tasarlayan ve arkadaki kodları yazan iki farklı kişi olduğunu düşünelim, yazılımın tamamlanması için bu iki kişinin birbiri ile iletişim halinde olması gerekecektir. Ara-yüzde görülen her özellik aslında bu iletişimin bir sonucu olacak ve yazılımın başarısı, işlevselliği kullanılabilirliği gibi çok sayıdaki özellik de bu iletişime bağlı olacaktır.

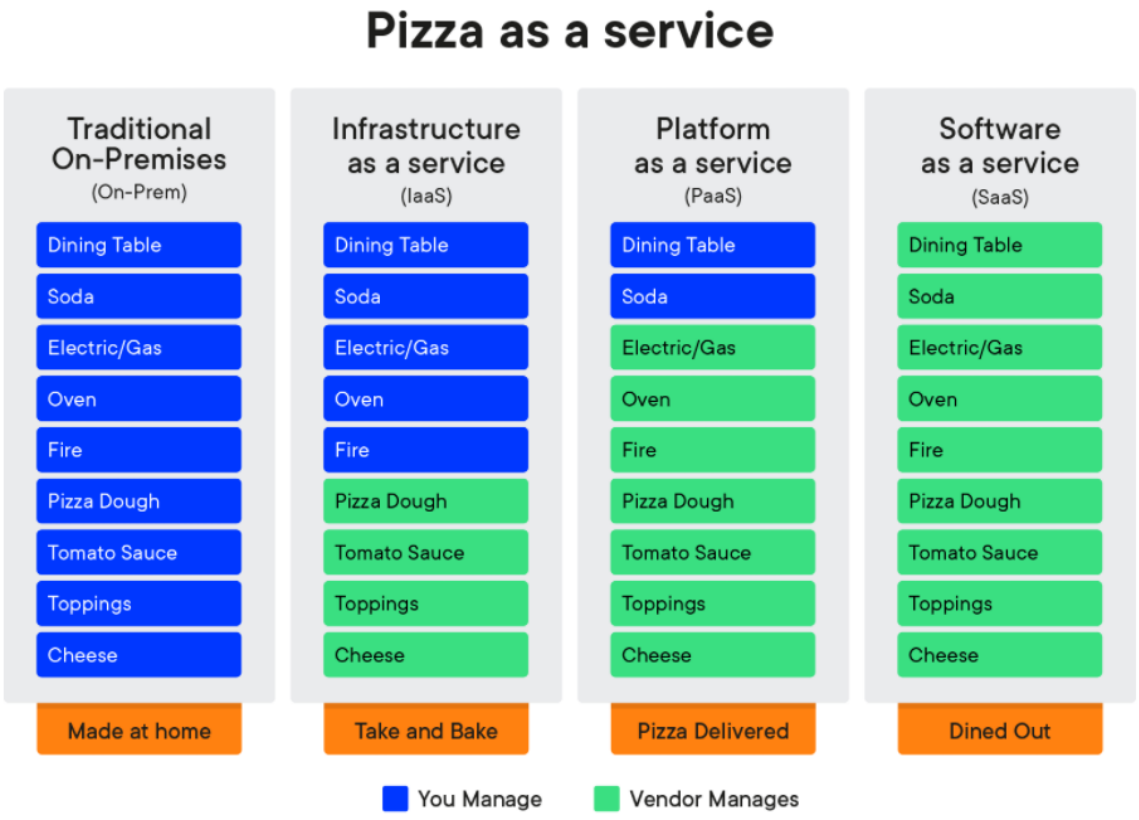
2) **Software as a Service (SaaS), Platform as a Service (PaaS), Infrastructure as a Service (IaaS), On-Premises kavramlarını örneklerle açıklayınız.**

   SaaS: Web dağıtım modeli nedeniyle, SaaS, IT personelinin her bir bilgisayara uygulamaları indirip yüklemesini gerektirmez. SaaS ile, satıcılar veri, ara katman yazılımı, sunucular ve depolama gibi tüm potansiyel teknik sorunları yönetir ve bu da işletme için kolaylaştırılmış bakım ve destek sağlar.

   PaaS: Platform olarak bir Hizmet Olarak Platform (PaaS) olarak da bilinen bulut platformu hizmetleri, temel olarak uygulamalar için kullanılırken, belirli yazılımlara bulut bileşenleri sağlar. PaaS, geliştiriciler için özelleştirilmiş uygulamalar oluşturmak için kullanabilecekleri ve kullanabilecekleri bir çerçeve sunar. Tüm sunucular, depolama ve ağ iletişimi kurum veya üçüncü taraf bir sağlayıcı tarafından yönetilirken, geliştiriciler uygulamaların yönetimini koruyabilir.

   IaaS: Hizmet Olarak Altyapı (IaaS) olarak bilinen bulut altyapısı hizmetleri, ölçeklenebilir ve otomatik bilgi işlem kaynaklarından yapılır. IaaS, bilgisayarlara, ağlara, depolama alanlarına ve diğer servislere erişmek ve bunları izlemek için tamamen self servistir. IaaS işletmelerin donanım satın almak yerine talep üzerine ve ihtiyaç duydukları kaynakları satın almalarını sağlar.

   Pizza as a servis örneğine bakacak olursak IaaS yapısında Ateşi, fırını, elektriği, masayı biz sağlıyoruz pizza hamuru, sosu, malzemeleri, peyniri bize hazır olarak sağlanıyor ve pizzamızı yapıyoruz. PaaS yapısında sadece Masa ve sodayı biz sağlıyoruz pizzanın pişirilmesi de dahil olmak üzere hepsi sağlanıyor. SaaS yapısında herşey bize sağlanıyor ve sadece pizzayı yeme işi bize kalıyor.



3) **Continuous Integration, Continuous Delivery ve Continuous Deployment kavramlarını açıklayınız.**

Continuous Integration: yazılım geliştirme değişikliklerini sürekli olarak entegre etmek için otomatikleştirilmiş bir süreçtir. kaynak kodun oluşturulmasını, test edilmesini ve doğrulanmasını otomatikleştirir. teknik amacı, uygulamaları oluşturmak, paketlemek ve test etmek için tutarlı ve otomatik bir yol oluşturmaktır.

Continuous Delivery: Sürekli teslimin amacı, paketlenmiş bir yapıyı üretim ortamına teslim etmektir. CD, dağıtım süreci de dahil olmak üzere tüm teslimat sürecini otomatikleştirir. Sorumlulukları, altyapı sağlamayı, değişiklikleri yönetmeyi (biletleme), yapıtları dağıtmayı, bu değişiklikleri doğrulamayı ve izlemeyi ve herhangi bir sorun varsa bu değişikliklerin gerçekleşmemesini sağlamayı içerebilir. CD ise sürekli entegrasyonun bittiği yerde başlar. CD, uygulamaların seçilen altyapı ortamlarına teslimini otomatikleştirir.

4) **API Gateway pattern'ı açıklayınız.**

API Gateway, istemcilerle backend sunucuları / mikro servisler arasında duran bir API yönetim aracıdır. API Gateway istek sınırlandırma, istatistik, kimlik doğrulama vs. çeşitli sık kullanılan işlevleri üzerine alarak asıl API sunucularınızın önünde bir üst katman oluşturur. API Gateway ile Authentication, Authorization, Logging, Response Caching, Routing işlemlerini yapabiliriz.

5) **Backend for frontend (BFF) pattern' ı açıklayınız.**
Her bir Client(Frontend) turu için ayrı ayrı API Gateway kullanılmasına Backend for Frontend deniyor. Mesela Web Application tipinde client'lar icin ayrı bir API Gateway, Mobile Application tipinde client'lar icin ayrı bir API Gateway, 3rd Party Application tipinde client'lar icin ayri bir API Gateway olusturulabilir.
Bunun en onemli sebebi, client'larin Backend tarafindan istedikleri data ve bekledikleri teknolojinin farkli olabilmesi.

6) **Circuit-breaker pattern' ı açıklayınız.**
Mesela bir backend servisimiz var diyelim. Bir de bu servisten çağırdığımız external(harici) bir servis olsun. Bazı durumlarda bu external servis fail olabiliyorsa buna karşı bir önlem olarak Circuit Breaker Pattern'i kullanabiliriz.
Circuit Breaker Pattern'e göre 3 state(state machine şeklinde) var. Bunlar: Closed, Open, Half-Open. Eğer ki external servisimiz çalışıyorsa ve bir sıkıntı yoksa Circuit Breaker'imiz Closed state'inde olur. Eğer external servisimiz fail olursa Circuit Breaker'imiz Open state'inde olur, ve belirli aralıklarla Half-Open state'ine geçerek external servisin eski haline gelip gelmediğini kontrol eder.

7) **Microservice chassis pattern' ı kısaca açıklayınız.**
Biz uygulamamız için çeşitli microservice'ler yazarken cross-cutting concern'ler icin her serviste ister istemez benzer kodlar yazıyoruz. Buradaki cross-cutting concern'leri servislerimizden ayırarak ayrı bir serviste yazsaydık hem kod çoklanmasinin önüne geçmiş olurduk, hem de cross-cutting concern'lerle ilgili bir değişiklik yaptığımızda tüm servislerde güncelleme yapmak zorunda kalmazdık.