1. Senkron iletişimde bir işlemin çalışması kendinden önceki işlemlere bağlıdır. Önceki işlemler bitmeden bu işleme geçilemez. Bu yüzden blocking yapısı vardır. Asenkron işlemde ise bağlılık yoktur. İşlemler aynı anda çalışabilir.

2. RabbitMQ mesaj kuyruğu sistemidir. Yapılacak asenkron işlemler bu kuyruğa sokulur. Apache Kafka verilerin bir sistemden alınıp diğer sistemlere hatasız ve hızlı bir şekilde transfer edilmesini sağlayan bir sistemdir.
   RabbitMQ'da mesaj önceliği vardır. Kafka mesaj önceliğini desteklemez.
   RabbitMQ'da tüketiciler mesajları gönderdikleri sırayla alırlar. Kafka ise mesajları sıraya almak için konuları ve bölümleri kullanır. Üreticinin gönderdiği mesaj belirli bir konuya ve bölüme girer. Kafka doğrudan üretici-tüketici alışverişlerini desteklemez. Tüketici bölümden farklı bir sırayla mesaj alır.
   RabbitMQ mesajları bir süreliğine bellekte saklar. Tüketici mesajı alırsa mesaj bellekten silinir. Tüketici mesajı alamazsa RabbitMQ mesajı tekrar gönderir. Kafka tüm olayları disk üzerinde bir günlük olarak saklar.

3. Virtual machine bir makinede birden fazla işletim sisteminin özelliklerini kullanmamızı sağlayan bir programdır. Örneğin Windows kullanılan bir bilgisayarda virtual machine ile Linux'un özelliklerini kullanabiliriz.
   Docker da bir sanallaştırma platformudur. Bir işletim sistemi üzerinde bağımsız containerlar kurarak sanallaştırma sağlar. Docker sayesinde her bir uygulama için yeni bir VM ve işletim sistemi kurma zorunluluğu yoktur.

4. Bazı docker komutları şunlardır:
   Docker: Docker ile ilgili komutları listeler.
   Docker version
   Docker images: Çalıştırılabilir docker imajları listelenir.
   Docker run: Container oluşturmak ve çalıştırmak için kullanılır. Docker create ve docker start komutlarının birleşimidir.
   Docker ps: Çalışan containerları listeler.
   Docker container stop containerı durdurur.
   Docker rm ile container silinir.

5. Monolith mimaride bir uygulama tek bir yapıda toplanmıştır. Böyle bir durumda uygulamanın bileşenleri birbirine sıkıca bağlıdır. Microservice mimarisinde ise uygulama küçük ve birbirinden bağımsız servislere bölünür. Örneğin user ve product servislerimiz olsun. Monolith yapıda bunlar aynı yerdedir. Microservice yapısında ise ayrılırlar. Veri tabanında ilgili kısımlar da birbirinden ayrılmalıdır. İhtiyaç halinde microservislerin birbiriyle iletişim kurması gerekebilir.

Monolith yapıda uygulamayı oluşturmak daha kolaydır ve maaliyeti daha düşüktür. Microservice yapısında ise uygulama büyüdükçe uygulamayı yönetmek veya değişiklik yapmak daha kolay olur.
Monolith uygulamada sistemdeki bir hata bütün sistemi etkileyebilir ama microservislerde yapılar birbirinden bağımsız olduğu için sistem daha güvenlidir. Test etmek daha kolaydır. Bu nedenlerden dolayı büyük uygulamalarda microservis yapısı tercih edilir.

6. JPA Java Persistance API anlamına gelir. Database işlemlerini kolaylaştıran bir spefikasyondur. ORM (Object Relational Mapping) ile obje ve veri tabanındaki entity arasında bir ilişki kurulur. Ancak JPA'yı uygulamak için bir programa ihtiyacımız vardır. Hibernate frameworku bunlardan biridir. Bir class ve entity arasındaki bağı annotation ile ya da xml dosyası ile kurabiliriz. Annotation kullanarak bir örnek:

```
@Entity
@Table(name= "users")
public class User {

@Id
private int id;
private String firstName,lastName;

//getters, setters
```

@Entity bu classı bir entity classı yapar. @Id ile primary keyi belirtiriz. @Column ifadesi ile hangi propertynin hangi sütuna denk geleceğini gösterebiliriz.
Spring Data frameworkü Hibernate ile kullanılabilir. Database'e bağlanacağımız ve query oluşturacağımız zaman kolaylık sağlar. Database'e bağlanma işlemleri otomatik olarak yapılır ve hazır queryler kullanan methodlar kullanılır. findByID(), deleteAll(), findAll(), save() bu methodlardan bazılarıdır.

7. Api gateway apiler arasındaki trafiği yönetir. Microservice yapısında yaygın olarak kullanılır. Client ve servisler arasında bulunur. Gelen istekleri servislere yönlendirmede ve servislerden gelen yanıtları göndermede görevlidir.
Service discovery ise servislerin birbiriyle iletişim kurabilmesini sağlar. Health check ile çalışmayan servislerin sistem dışında kalmasını sağlar. Discovery kavramı servisler arası iletişimle ilgilidir. Load balancing gelen trafiği bir sunucu havuzunda aynı göreve sahip olan sunucular arasında paylaştırma işlemidir.