**CLOUD SYSTEMS**

**☁️ 1. Infrastructure as a Service (IaaS) — Altyapı Hizmeti**

* **Nedir?** Sanal sunucular, ağlar, depolama gibi fiziksel altyapı kaynaklarının internet üzerinden sağlanmasıdır.
* **Kullanıcı neyi yönetir?** İşletim sistemi, uygulamalar, veriler.
* **Sağlayıcı neyi yönetir?** Fiziksel donanım, ağ, sanallaştırma.
* **Örnek:** Amazon EC2, Microsoft Azure VM, Google Compute Engine.

**🛠️ 2. Platform as a Service (PaaS) — Platform Hizmeti**

* **Nedir?** Uygulama geliştirmek için gerekli olan altyapıyı ve araçları sunar.
* **Kullanıcı neyi yapar?** Kod yazar, uygulama geliştirir.
* **Sağlayıcı neyi sunar?** Altyapı + çalışma zamanı ortamı + veritabanı + araçlar.
* **Örnek:** Google App Engine, Heroku, Microsoft Azure App Services.

**🖥️ 3. Software as a Service (SaaS) — Yazılım Hizmeti**

* **Nedir?** İnternet üzerinden erişilen, kullanıma hazır yazılım hizmetidir.
* **Kullanıcı ne yapar?** Giriş yapar ve uygulamayı kullanır.
* **Yönetim:** Her şey sağlayıcı tarafından yönetilir.
* **Örnek:** Gmail, Microsoft 365, Dropbox, Zoom.

**⚙️ 4. Function as a Service (FaaS) — Fonksiyon Hizmeti (Sunucusuz)**

* **Nedir?** Küçük kod parçalarının (fonksiyonların) sadece çağrıldıklarında çalışması.
* **Avantaj:** Sunucu yönetimi gerekmez, olay bazlı çalışır.
* **Örnek:** AWS Lambda, Azure Functions, Google Cloud Functions.

**🗄️ 5. Database as a Service (DBaaS) — Veritabanı Hizmeti**

* **Nedir?** Veritabanlarının kurulumu, yönetimi ve ölçeklendirilmesi sağlayıcı tarafından yapılır.
* **Avantaj:** Yedekleme, replikasyon, bakım işleri otomatik yapılır.
* **Örnek:** Amazon RDS, Azure SQL Database, Firebase Realtime Database.

**🛡️ 6. Security as a Service (SECaaS) — Güvenlik Hizmeti**

* **Nedir?** Antivirüs, güvenlik duvarı, kimlik doğrulama gibi hizmetlerin bulut üzerinden sağlanmasıdır.
* **Amaç:** Kurumları siber tehditlere karşı korumak.
* **Örnek:** Cloudflare, Okta, Symantec Cloud Security.

**💾 7. Storage as a Service (STaaS) — Depolama Hizmeti**

* **Nedir?** Dosya, nesne veya blok düzeyinde veri depolamanın bulut üzerinden sunulması.
* **Avantaj:** Kolay yedekleme, erişim ve ölçeklenebilirlik.
* **Örnek:** Amazon S3, Google Cloud Storage, Dropbox.

Eğer bu modelleri bir **katman** gibi düşünürsen:

* **IaaS**: Temel altyapı.
* **PaaS**: Uygulama geliştirme ortamı.
* **SaaS**: Tamamlanmış yazılım uygulaması.

Diğerleri (FaaS, DBaaS, vb.) bu temel modellerin özel uygulamalarıdır.

1. **Infrastructureas a Service (IaaS) :** AWS, MS Azure, GoogleCloud
2. **Platform as a Service (PaaS)**: Google AppEngine, Heroku, RedhatOpen Shift
3. **Software as a Service (SaaS)**: Trello, Zoom
4. **Functionas a Service (FaaS)**: AWS Lambda
5. **Database as a Service (DBaaS)**: MongoAtlas, FirebaseFirestone
6. **Security as a Service (SECaaS)**: Cloudflare
7. **Storage as a Service (STaaS)**: Google Drive, OneDrive, Box

**🎮 1. Gaming as a Service (GaaS) — Oyun Hizmeti**

* **Nedir?** Oyunların internet üzerinden stream edilerek oynanmasıdır; kullanıcı cihazına oyun yüklemeden oynar.
* **Avantaj:** Donanım gerekmez, sürekli güncellenen içerik sunulur.
* **Örnek:** NVIDIA GeForce NOW, Xbox Cloud Gaming, Google Stadia (kapatıldı), PlayStation Now.

**📺 2. Streaming as a Service (STaaS) — Yayın Hizmeti**

Bu bazen **Streaming** ya da **Storage as a Service** olarak da kısaltılır. Buradaki anlamı “yayın” ise açıklama şu şekilde:

* **Nedir?** Video, müzik ve canlı yayın içeriklerinin internet üzerinden akış (stream) olarak sunulması.
* **Avantaj:** İndirme gerekmez, cihazdan bağımsız erişim sağlanır.
* **Örnek:** Netflix, Spotify, YouTube, Twitch.

**🎨 3. Content Creation as a Service (CCaaS) — İçerik Üretim Hizmeti**

* **Nedir?** Görsel, yazılı ya da video içeriklerin oluşturulması için bulut tabanlı araçlar veya AI destekli sistemlerin sunulmasıdır.
* **Amaç:** İçerik üretimini kolaylaştırmak.
* **Örnek:** Canva, Adobe Creative Cloud (bulut tabanlı kısmı), ChatGPT ile içerik oluşturma, Lumen5 (video).

**💬 4. Communication as a Service (CaaS) — İletişim Hizmeti**

* **Nedir?** Sesli arama, görüntülü konuşma, mesajlaşma gibi iletişim servislerinin internet üzerinden sağlanması.
* **Kullanım Alanı:** Şirket içi iletişim, müşteri destek sistemleri, çağrı merkezleri.
* **Örnek:** Zoom, Microsoft Teams, Twilio, Slack, Google Meet.

**🎓 5. Education as a Service (EaaS) — Eğitim Hizmeti**

* **Nedir?** Eğitim materyallerinin, sınavların, sanal sınıfların ve öğretim hizmetlerinin internet üzerinden sunulması.
* **Avantaj:** Her yerden erişim, özelleştirilebilir öğrenme yolları.
* **Örnek:** Coursera, Khan Academy, Udemy, Google Classroom, edX.

Bu hizmetler genellikle SaaS’ın birer alt kategorisidir ama **alanlarına özel uzmanlaşmış çözümler** sundukları için isimlendirilmişlerdir.

1. **Gaming as a Service (GaaS)**: Xbox CloudGaming, NVIDIA GeForceNow
2. **Streamingas a Service (STaaS)**: Spotify, Netflix, Youtube
3. **Content Creationas a Service (CCaaS)**: Canva, Figma
4. **Communicationas a Service (CaaS)**: Discord, Slack
5. **Educationas a Service (EaaS)**: Duolingo, KhanAcademy

**🧱 Monolithic Architecture (Tek Parça Mimarisi)**

**✔️ Özellikleri:**

* **Tüm uygulama tek bir kod tabanı** içinde yer alır.
* Bileşenler sıkı şekilde birbirine bağlıdır.
* Yaygın olarak geleneksel uygulamalarda kullanılır.

**✅ Avantajları:**

* **Basit geliştirme ve dağıtım**: Küçük ekipler için uygundur.
* Tek sunucuya kolayca kurulabilir.
* Hızlı prototipleme sağlar.

**❌ Dezavantajları:**

* **Bakım ve ölçekleme zordur**: Küçük bir değişiklik tüm sistemi etkiler.
* Büyük kod tabanı karmaşıklaşır.
* Tek bir noktadan hata alma riski yüksek (**single point of failure**).

**🧩 Microservices Architecture (Mikroservis Mimarisi)**

**✔️ Özellikleri:**

* Uygulama, bağımsız çalışan **küçük servislerden oluşur**.
* Her servis kendi veritabanına sahip olabilir.
* Servisler genelde **REST API** veya mesajlaşma ile iletişim kurar.

**✅ Avantajları:**

* **Bağımsız geliştirme ve dağıtım** yapılabilir.
* Her servis farklı teknolojilerle yazılabilir.
* Bir servisteki hata tüm sistemi çökertmez.

**❌ Dezavantajları:**

* **Yönetimi ve testi zordur.**
* Servisler arası iletişim karmaşık olabilir.
* Dağıtık sistemlerin izlenmesi ve güvenliği daha zordur.

**🔁 Monolith vs Microservice Karşılaştırması**

| **Kriter** | **Monolithic** | **Microservice** |
| --- | --- | --- |
| **Kod yapısı** | Tek blok | Küçük bağımsız modüller |
| **Dağıtım** | Tek seferde | Bağımsız servisler |
| **Teknoloji çeşitliliği** | Kısıtlı | Her serviste farklı teknoloji |
| **Geliştirme** | Hızlı başlangıç, sonra yavaşlar | Karmaşık ama ölçeklenebilir |
| **Test** | Daha kolay | Zor (çok servis var) |
| **Arıza etkisi** | Tüm sistemi etkiler | Sadece ilgili servisi etkiler |

BULUT

Bulut, bu sunucuların sahibi olan şirketler tarafından kiralanmak üzere sunulan çok sayıda uzak sunucudan oluşur.

Bulut karakteristikleri:

* Ölçeklenebilirlik : Yük arttıkça performansı koruyun
* Esneklik :Sunucu yapılandırmasını değişen taleplere göre uyarlayın
* Dayanıklılık :Sunucu arızası durumunda hizmeti sürdürün

Ölçeklenebilirlik, yazılımınızın artan sayıda kullanıcıyla başa çıkma yeteneğini yansıtır. Yazılımınız üzerindeki yük arttıkça, yazılımınız otomatik olarak adapte olur ve böylece sistem performansı ve yanıt süresi korunur.

Esneklik, ölçeklenebilirlikle ancak aynı zamanda hem aşağı hem de yukarı ölçeklemeye olanak tanır. uygulamanızdaki talebi izleyebilir ve kullanıcı sayısı değiştikçe sunucuları dinamik olarak ekleyebilir veya kaldırabilirsiniz.

Esneklik, yazılım mimarinizi sunucu arızalarını tolere edecek şekilde tasarlayabileceğiniz anlamına gelir. Yazılımınızın birkaç kopyasını eşzamanlı olarak kullanıma sunabilirsiniz. Bunlardan biri arızalanırsa, diğerleri hizmet vermeye devam eder.

Bulut kullanımının faydaları :

Maliyet

Donanım tedarikinin ilk sermaye maliyetlerinden kaçınırsınız

Başlangıç süresi

Başlamadan önce donanımın teslim edilmesini beklemek zorunda değilsiniz iş. Bulutu kullanarak sunucularınızı birkaç dakika içinde hazır ve çalışır hale getirebilirsiniz.

Sunucu seçimi

Kiraladığınız sunucuların yeterince güçlü olmadığını fark ederseniz, daha güçlü sistemlere yükseltebilirsiniz. Yük testi gibi kısa vadeli gereksinimler için sunucu ekleyebilirsiniz.

Dağıtık geliştirme

Farklı konumlardan çalışan dağıtık bir geliştirme ekibiniz varsa, tüm ekip üyeleri aynı geliştirme ortamına sahip olur ve tüm bilgileri sorunsuz bir şekilde paylaşabilir.

**Sanal Bulut Sunucularının Ayrıntılı Özeti:**

Sanal bulut sunucuları, fiziksel bir bilgisayar (sunucu) üzerinde çalışan, yazılım temelli sistemlerdir. Her sanal sunucu, bir işletim sistemi ve sunucu işlevselliği sağlayan yazılım paketleri içerir. Temel özellikleri şunlardır:

1. **Bağımsız Çalışma:**  
   Sanal sunucular, fiziksel donanımdan bağımsız olarak çalışabilir. Yani belirli bir donanıma bağlı kalmadan, buluttaki herhangi bir fiziksel sistem üzerinde çalıştırılabilirler. Bu durum, dış bağımlılıkların olmamasından kaynaklanır.
2. **Taşınabilirlik ve Esneklik:**  
   “Her yerde çalıştırma” özelliği, sanal sunucuların farklı veri merkezlerinde, farklı donanımlar üzerinde kolayca taşınıp çalıştırılabilmesini sağlar.
3. **Sanal Makineler (VM’ler):**  
   Sanal sunucular genellikle sanal makineler aracılığıyla uygulanır. VM'ler, fiziksel sunucu donanımını taklit eden yapay ortamlardır ve sanal sunucuların çalışması için uygun altyapıyı sağlarlar.
4. **Hipervizör Kullanımı:**  
   Sanal makineler, hipervizör adlı bir yazılım sayesinde fiziksel donanım üzerinde çalıştırılır. Hipervizör, donanımı sanal makineler için emüle eder ve kaynakların paylaşımını yönetir.
5. **Donanım Taklidi:**  
   Hipervizör destekli sanal makineler sayesinde, sanal sunucular fiziksel bir sunucuyla aynı donanım ortamına sahipmiş gibi çalışabilir.

**Sonuç:**  
Sanal bulut sunucuları, fiziksel donanımdan soyutlanmış, taşınabilir, esnek ve yönetimi kolay sistemlerdir. Sanal makineler ve hipervizör teknolojileri sayesinde, fiziksel sunucularla aynı düzeyde performans ve işlevsellik sunabilirler. Bu yapı, bulut bilişimde kaynakların verimli kullanılmasını sağlar.

metin, ekran görüntüsü, diyagram, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

**Görselin Anlattıkları:**

**En Alt Katman: Server Hardware (Fiziksel Sunucu Donanımı)**

Gerçek fiziksel donanımı temsil eder (CPU, RAM, disk vb.).

**Host OS (Ana İşletim Sistemi)**

Fiziksel sunucunun üzerinde çalışan temel işletim sistemidir. Örnek: Linux ya da Windows Server olabilir.

**Hypervisor (Hipervizör)**

Sanal makineleri yöneten yazılımdır. Fiziksel kaynakları sanal makineler arasında paylaştırır. İki türü vardır:

* Type 1: Doğrudan donanım üzerinde çalışır (bare metal).
* Type 2: Ana işletim sistemi üzerinde çalışan (hosted).

**Sanal Makineler (Virtual Machines)**

Bu katmanda, hipervizör tarafından yönetilen iki adet sanal makine gösterilmiştir:

* **Virtual Web Server:**
  + **Guest OS:** Örneğin Linux.
  + **Server Software:** Örneğin Apache Web Server.
* **Virtual Mail Server:**
  + **Guest OS:** Örneğin Windows Server.
  + **Server Software:** Örneğin Outlook Mail Server.

**Özetle:**

Bu şema, tek bir fiziksel sunucu üzerinde **farklı işletim sistemlerine ve görev tanımlarına sahip** iki ayrı sanal sunucunun nasıl kurulabileceğini göstermektedir. Sanal web sunucusu Linux + Apache çalıştırırken, sanal mail sunucusu Windows Server + Outlook çalıştırmaktadır. Her ikisi de hipervizör aracılığıyla aynı fiziksel donanımı paylaşır ama birbirinden izole şekilde çalışır.

Çok sayıda uygulama veya hizmet örneğine sahip bulut tabanlı bir sistem çalıştırıyorsanız ve bunların hepsi aynı işletim sistemini kullanıyorsa, 'konteynerler' adı verilen daha basit bir sanallaştırma teknolojisi kullanabilirsiniz.

Kapsayıcıların kullanılması, sanal sunucuların bulut üzerinde konuşlandırılması sürecini hızlandırır.

Konteynerler genellikle megabayt boyutundadır VM'ler ise gigabayttır.

Konteynerler, bir sanal makine için gereken birkaç dakika yerine birkaç saniye içinde başlatılabilir ve kapatılabilir.

Konteynerler, bağımsız sunucuların tek bir işletim sistemini paylaşmasını sağlayan bir işletim sistemi sanallaştırma teknolojisidir.

Özellikle her kullanıcının bir uygulamanın kendi sürümünü gördüğü izole uygulama hizmetleri sağlamak için kullanışlıdırlar.

metin, ekran görüntüsü, yazı tipi, diyagram içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

**Sol Taraftaki Yapı: Sanallaştırma (Virtualization)**

* **Hypervisor** kullanılarak, her bir sanal makine (örneğin: Virtual web server, Virtual mail server) kendi **Guest OS** (misafir işletim sistemi) ile birlikte çalışıyor.
* Her bir sanal makine, uygulama ve sunucu yazılımını çalıştırmak için ayrı bir işletim sistemi (örneğin Linux, Windows Server) gerektiriyor.
* Bu yapı **daha fazla kaynak tüketimi** anlamına gelir, çünkü her sanal makine tam bir işletim sistemi taşır.

**Sağ Taraftaki Yapı: Konteynerleştirme (Containerization)**

* **Container manager** (örneğin Docker) yardımıyla, her bir kullanıcı için (User 1 ve User 2) ayrı konteynerler oluşturulmuş.
* Her bir konteyner, sadece gerekli olan uygulama ve sunucu yazılımını içerir.
* **Ayrı bir işletim sistemi** barındırmazlar; bunun yerine ortak bir **Host OS** üzerinde çalışırlar.
* Bu da **daha hafif, daha hızlı ve kaynak açısından verimli** bir yapı sağlar.

**Sonuç:**

* Konteynerler, hizmetleri izole bir şekilde sağlamanın **daha az kaynak tüketen** ve **daha esnek** bir yoludur.
* Sol taraftaki yapı geleneksel sanallaştırma iken, sağ taraftaki yapı modern konteyner mimarisi ile hizmet dağıtımını göstermektedir.

**🔹 Konteynerler ile Yalıtılmış Hizmet Sağlama (Genel Kavramlar)**

**Konteyner nedir?**

* Uygulamaları, gerekli tüm bileşenleriyle birlikte tek bir paket içinde çalıştıran, **izole ortamlar**dır.
* Sanal makinelerden farkı: **tam işletim sistemi içermez**, yalnızca çalışması için gereken dosyaları taşır.
* Kaynak tüketimi **az**, başlatma süresi **hızlı**dır.

**Tarihçe:**

* İlk olarak 2007’de **Google** tarafından geliştirildi.
* 2015 yılında **Docker**’ın yaygınlaşmasıyla birlikte ana akım haline geldi.

**🐳 Docker Nedir ve Ne İşe Yarar?**

Docker, konteynerleri oluşturmak, dağıtmak ve çalıştırmak için kullanılan bir **konteyner yönetim sistemidir**.

**Docker Mimarisi:**

🧩 Aşağıdaki şemayı düşün:

java

KopyalaDüzenle

Kullanıcı

↓

Docker Client (istemci) → Docker Daemon (arka plan servisi)

↓ ↓

Dockerfile → Image → Container

**🔑 Docker Bileşenleri (Tablo 5.2’den Özet):**

| **Bileşen** | **Görevleri** |
| --- | --- |
| **Docker Daemon** | Konteynerleri oluşturur, çalıştırır, durdurur, izler. |
| **Docker Client** | Kullanıcıların Docker ile iletişim kurduğu arayüzdür. |
| **Dockerfile** | Bir konteynerin nasıl oluşturulacağını tarif eder. |
| **Image (Görüntü)** | Dockerfile'dan oluşan çalıştırılabilir yazılım paketidir. |
| **Container** | Image'ların çalıştırılmış halidir. İzole bir uygulama ortamıdır. |
| **Docker Hub** | Image’ların bulunduğu çevrim içi arşiv ve paylaşım alanıdır. |

**🧱 Docker Görüntüleri (Images):**

* Uygulamanın çalışması için gereken **her şey**i içerir: ikili dosyalar, kütüphaneler, sistem araçları.
* **Katmanlı yapı** kullanılır:
  + Temel imaj (örneğin: Ubuntu)
  + Üstüne eklenen yazılım bileşenleri (örneğin: Node.js, kendi uygulaman)
  + Güncellemeler yalnızca değişen katmanlar üzerinden yapılır → hızlı ve verimli.

**Görselle Anlatım:**

yaml

KopyalaDüzenle

Image:

├── Katman 1: Taban işletim sistemi

├── Katman 2: Kütüphaneler

├── Katman 3: Uygulama dosyaları

└── Katman 4: Yapılan değişiklikler (güncellemeler)

**🎯 Sınavda Dikkat Edilmesi Gereken Noktalar:**

1. **Docker ile sanallaştırma arasındaki farkı bil:**
   * Sanallaştırma: Her sanal makine kendi işletim sistemini taşır (ağır).
   * Docker: Tüm konteynerler aynı işletim sistemini paylaşır (hafif).
2. **Docker bileşenleri ve işlevleri**: Daemon, client, Dockerfile, image, container.
3. **Dockerfile ne işe yarar?**
   * Konteyneri oluşturacak **talimatları** içerir. (örn: FROM, RUN, COPY, CMD gibi komutlar kullanılır.)
4. **Image vs Container farkı:**
   * Image: Statik, çalıştırılabilir yazılım paketi.
   * Container: Image’ın aktif, çalışan hali.
5. **Kapsayıcıların avantajları:**
   * Hızlı başlatma
   * Düşük kaynak kullanımı
   * Taşınabilirlik (her ortamda aynı şekilde çalışır)

metin, ekran görüntüsü, diyagram, tasarım içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

metin, ekran görüntüsü, meyve içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

**✅ Konteynerlerin Faydaları – Özet**

1. **Bağımlılık Sorununu Ortadan Kaldırır**  
   → Uygulama geliştirme ve dağıtım ortamındaki yazılım/kütüphane farklılıkları sorun olmaz.  
   → Konteyner, gerekli tüm bağımlılıklarla birlikte çalışır.
2. **Taşınabilirlik Sağlar**  
   → Docker konteynerleri, Docker destekli tüm sistem ve bulut sağlayıcılarında **sorunsuz çalışır**.  
   → "Bir kez geliştir, her yerde çalıştır" yaklaşımını mümkün kılar.
3. **Tüm Destek Yazılımlarıyla Paketleme**  
   → Uygulamanız sadece kendi kodunu değil, ihtiyaç duyduğu tüm destek yazılımlarını da içerir.  
   → Böylece uygulama, dış bağımlılıklara ihtiyaç duymadan çalışır.
4. **Hizmet Odaklı Mimarileri Destekler**  
   → Her hizmeti ayrı bir konteynerde çalıştırarak **mikro hizmet mimarilerini** uygular.  
   → Daha **modüler, ölçeklenebilir ve yönetilebilir** sistemler kurulmasını sağlar.
5. **DevOps Süreçlerini Kolaylaştırır**  
   → Geliştirici ve operasyon ekiplerinin iş birliğini artırır.  
   → Otomasyon, sürekli entegrasyon ve sürekli teslimat süreçlerini destekler.

**🎯 Sınav İçin İpucu:**

Konteynerlerin en çok vurgulanan üç avantajı:

* **Taşınabilirlik**
* **Bağımlılık yönetimi**
* **DevOps uyumu**

Her şey bir hizmet olarak

* Sahip olunmak yerine kiralanan bir hizmet fikri, bulut bilişimin temelini oluşturmaktadır.

**Hizmet olarak altyapı (IaaS)**

Bulut sağlayıcıları, sanal sunucuları uygulamak için kullanabileceğiniz işlem hizmeti, ağ hizmeti ve depolama hizmeti gibi farklı türde altyapı hizmetleri sunar.

**Hizmet olarak platform (PaaS)**

Bu, yazılımınızı uygulamak için bulut sağlayıcısı tarafından sağlanan kütüphaneleri ve çerçeveleri kullandığınız orta düzey bir seviyedir. Bunlar SQL ve NoSQL veritabanları da dahil olmak üzere bir dizi işleve erişim sağlar.

**Hizmet olarak yazılım (SaaS)**

Yazılım ürününüz bulut çalışır ve kullanıcılar tarafından bir web tarayıcısı veya mobil uygulama aracılığıyla erişilir.

**📦 Bulut Bilişim Katmanları ve Modelleri – Açıklamalı Şema**

**🔹 1. SaaS – Software as a Service (Hizmet olarak Yazılım)**

* **Tanım:** Yazılımın internet üzerinden son kullanıcıya sunulması.
* **Kullanıcı bakış açısı:** Yalnızca kullanır, altyapı veya kurulumla ilgilenmez.
* **Örnekler:**
  + Fotoğraf düzenleme
  + Lojistik yönetimi
  + E-posta (Gmail)
  + CRM (Müşteri İlişkileri Yönetimi)
  + ERP sistemleri
  + Google Docs, Microsoft 365

🟦 Şemadaki not: **“CONSUME IT”** → SaaS hizmetleri doğrudan tüketilir.

**🔹 2. PaaS – Platform as a Service (Hizmet olarak Platform)**

* **Tanım:** Uygulama geliştirme ve dağıtımı için gerekli altyapının sağlandığı hizmet.
* **Kullanıcı bakış açısı:** Yazılım geliştirir, sunucu kurulumuyla uğraşmaz.
* **Örnekler:**
  + Veritabanı yönetimi
  + Yazılım geliştirme ortamları
  + Web barındırma
  + Karar destek sistemleri
  + Canlı yayın/streaming platformları

🟦 Şemadaki not: **“BUILD ON IT”** → Üzerine yazılım geliştirilen katmandır.

**🔹 3. IaaS – Infrastructure as a Service (Hizmet olarak Altyapı)**

* **Tanım:** Temel bilgi işlem kaynaklarının (sunucular, depolama, ağ) sanal olarak sunulması.
* **Kullanıcı bakış açısı:** Altyapıyı yönetir, üzerine yazılım kurabilir.
* **Örnekler:**
  + Sanallaştırma (VM)
  + Ağ yönetimi
  + Güvenlik hizmetleri
  + Sistem yönetimi
  + Depolama (Amazon S3 gibi)

🟦 Şemadaki not: **“MIGRATE TO IT”** → Geleneksel altyapıdan buluta geçişi ifade eder.

**☁️ Ekstra: Cloud Data Center**

* Bu katman tüm SaaS, PaaS, IaaS hizmetlerinin fiziksel olarak barındırıldığı veri merkezidir.

**☁️ BPaaS (Business Process as a Service) – İş Süreci Hizmeti Olarak**

* **Yukarıdaki tüm katmanları kullanarak iş süreçlerini otomatikleştirir.**
* Örneğin: İnsan kaynakları yönetimi, faturalama süreçleri gibi iş uygulamaları tamamen bulut üzerinde çalışabilir.

**🎯 Sınav için Özet İpucu:**

| **Model** | **Açılımı** | **Kullanım amacı** | **Anahtar fiil** |
| --- | --- | --- | --- |
| SaaS | Software as a Service | Yazılımı doğrudan kullan | Consume |
| PaaS | Platform as a Service | Yazılım geliştir | Build |
| IaaS | Infrastructure as a Service | Sunucu ve altyapı kullan | Migrate |
| BPaaS | Business Process as a Service | İş süreçlerini buluta taşı | Optimize |

**📊 Şekil 5.6 – IaaS ve PaaS’te Yönetim Sorumlulukları**

**1. Software as a Service (SaaS)**

* ✅ **Tüm bileşenler yazılım sağlayıcısı tarafından yönetilir.**
* Kullanıcı yalnızca yazılımı kullanır, hiçbir altyapı veya platform bileşenini yönetmez.

**2. Platform as a Service (PaaS)**

* ✅ **Temel altyapı (sunucular, ağ, depolama)** → **Bulut sağlayıcısı** tarafından yönetilir.
* ✅ **Uygulama servisleri (veritabanı gibi)** → **Yazılım sağlayıcısı** tarafından yönetilir.
* ❌ **Kullanıcı, kendi uygulamasını geliştirir ve dağıtır**, ama altyapı detaylarıyla ilgilenmez.

**3. Infrastructure as a Service (IaaS)**

* ✅ **Temel altyapı (sunucu, ağ, sanallaştırma)** → **Bulut sağlayıcısı** tarafından yönetilir.
* ❌ **Uygulama servisleri ve yazılım** → **Kullanıcının** sorumluluğundadır.
* Kullanıcı, veritabanı kurmak, uygulama yüklemek gibi işlemlerden sorumludur.

**🎯 Sınav İçin Kritik Bilgi:**

| **Katman/Bileşen** | **SaaS** | **PaaS** | **IaaS** |
| --- | --- | --- | --- |
| Uygulama (App) | Sağlayıcı yönetir ✅ | Kullanıcı yönetir ❌ | Kullanıcı yönetir ❌ |
| Uygulama servisleri (DB vb) | Sağlayıcı yönetir ✅ | Sağlayıcı yönetir ✅ | Kullanıcı yönetir ❌ |
| Yönetim hizmetleri | Sağlayıcı yönetir ✅ | Sağlayıcı yönetir ✅ | Sağlayıcı yönetir ✅ |
| Temel bilişim altyapısı | Sağlayıcı yönetir ✅ | Sağlayıcı yönetir ✅ | Sağlayıcı yönetir ✅ |

**🧠 Ezber İçin İpucu:**

* **SaaS:** Her şey hazır → “Sadece kullan”
* **PaaS:** Altyapı hazır → “Sen geliştir”
* **IaaS:** Altyapı sağlanır → “Sen kur, sen yönet”

**✅ Hizmet Olarak Yazılım (SaaS) – Özet**

1. **Yazılım artık bir hizmet olarak sunuluyor**  
   → Fiziksel kurulum yerine internet üzerinden erişim sağlanıyor.
2. **Bulut altyapısı kullanılır**  
   → Yazılım, bulut sağlayıcısında çalışır. Kullanıcının cihazına yüklenmesine gerek yoktur.
3. **Erişim kolaylığı sağlar**  
   → Web tarayıcısı veya mobil uygulama üzerinden her yerden erişilebilir.
4. **Abonelik temelli ödeme modeli**  
   → Yazılım satın alınmaz; **aylık/yıllık ücret** karşılığında kullanılır.

**🎯 Sınav için kilit cümle:**

“SaaS, kullanıcıların yazılımı yüklemeden internet üzerinden, abonelikle kullanmasını sağlayan bulut tabanlı bir hizmet modelidir.”

**SaaS Modelinin Yazılım Sağlayıcıları İçin Faydaları – Özet**

1. **💳 Ödeme Esnekliği**
   * Abonelik veya kullanım başına ödeme seçenekleri sunarak daha fazla müşteri çekilir.
   * Yüksek ön ödeme zorunluluğu ortadan kalkar.
2. **🧪 Satın Almadan Önce Deneme İmkânı**
   * Ücretsiz veya düşük maliyetli sürümler sunularak erken geri bildirim alınabilir.
3. **📊 Veri Toplama ve Pazarlama**
   * Kullanıcı davranışlarına dair veri toplanır → yazılım iyileştirilebilir.
   * Ek ürünler için hedefli pazarlama yapılabilir.
4. **💵 Düzenli Nakit Akışı**
   * Sürekli ödeme modeli sayesinde gelirler yıl boyunca dağılır, finansal istikrar sağlanır.
5. **🔧 Kolay Güncelleme Yönetimi**
   * Tüm kullanıcılar aynı anda en güncel sürümü kullanır.
   * Sürüm uyumsuzlukları ve bakım maliyetleri azalır.
6. **🚀 Sürekli Dağıtım (Continuous Delivery)**
   * Hatalar hızlıca düzeltilir, yazılım sürekli güncel kalır.
   * Yazılımın güvenilirliği ve kalitesi artar.

**✅ SaaS Müşteri Avantajları – Açıklama**

1. **📱 Cihaz Bağımsız Erişim**  
   → Yazılıma **mobil, dizüstü ve masaüstü cihazlardan** kolayca erişilebilir.  
   → Her yerden çalışma imkânı sağlar.
2. **💰 Ön Maliyet Yok**  
   → Sunucu ya da lisans yatırımı gerekmez.  
   → **Abonelikle kullanılır**, özellikle küçük işletmeler için erişimi kolaylaştırır.
3. **⚙️ Anında Güncellemeler**  
   → Kullanıcılar her zaman en güncel sürümü kullanır.  
   → Bakım ve güncelleme işlemleriyle uğraşmazsınız.
4. **📉 Düşük Yazılım Yönetim Maliyeti**  
   → IT ekiplerinin yükü azalır.  
   → Yazılımın kurulumu, bakımı ve desteği SaaS sağlayıcısına aittir.

**⚠️ SaaS Müşteri Dezavantajları – Açıklama**

1. **🔐 Gizlilik Yönetmeliklerine Uyum Zorluğu**  
   → Veriler üçüncü taraf sunucularda tutulur → **KVKK, GDPR gibi yasal düzenlemelere** uyum zorlaşabilir.
2. **🌐 Ağ Bağımlılığı (Network Constraints)**  
   → Yazılım internet bağlantısına bağımlıdır.  
   → Zayıf ağ altyapısı performans sorunlarına yol açabilir.
3. **🛡️ Güvenlik Endişeleri**  
   → Veriler harici sunucularda tutulduğu için siber saldırı riski olabilir.  
   → Verinin kimde ve nasıl korunduğu önemli bir konudur.
4. **🔄 Güncellemeler Üzerinde Kontrol Kaybı**  
   → Tüm kullanıcılar aynı anda otomatik güncellemeye zorlanabilir.  
   → Bazı kullanıcılar eski sürüm tercih edebilir ama bu mümkün olmayabilir.
5. **🔐 Hizmet Sağlayıcıya Bağımlılık (Service Lock-in)**  
   → Veri ve süreçler sağlayıcının sistemine bağımlı hale gelebilir.  
   → Başka bir SaaS sistemine geçmek **zor ve maliyetli** olabilir.
6. **🔄 Veri Değişim Güçlüğü (Data Exchange)**  
   → Farklı sistemler veya SaaS uygulamaları arasında veri alışverişi her zaman kolay değildir.  
   → Entegrasyon sorunları yaşanabilir.

**⚠️ SaaS'ta Veri Depolama ve Yönetim Sorunları – Özet**

**1. 📜 Yönetmelik Uyumu (Regulasyon)**

* Bazı ülkelerde (örneğin AB ülkeleri) **kişisel verilerin nerede saklandığına dair sıkı yasalar** vardır.
* Eğer SaaS sağlayıcısı bu yasalara uyumlu değilse, firmalar hizmeti kullanmakta **çekimser kalabilir**.

**2. 🌐 Veri Aktarımı**

* SaaS kullanımı sırasında çok veri gönderilip alınması gerekiyorsa, **internet hızı** önemli hale gelir.
* Düşük bant genişliği olan kullanıcılar için **performans düşüklüğü** yaşanabilir.
* Özellikle bireyler ve küçük şirketler için yüksek hızlı bağlantılar **maliyetli** olabilir.

**3. 🔐 Veri Güvenliği**

* Hassas veri kullanan işletmeler, verilerini dış bir sağlayıcıya teslim etmek istemeyebilir.
* Büyük sağlayıcılar bile **güvenlik ihlalleri** yaşayabilir; güvenlik garantisi her zaman verilemez.
* **Veri kontrolü kaybı** önemli bir endişe kaynağıdır.

**4. 🔄 Veri Alışverişi (Entegrasyon Sorunları)**

* Bulut yazılımı ile başka sistemler arasında veri paylaşımı gerekebilir.
* Ancak eğer SaaS sağlayıcısı **harici kullanım için açık bir API sunmuyorsa**, bu veri alışverişi **zorlaşır veya imkânsız hale gelir**.

**☁️ SaaS Tasarım Sorunları – Açıklama ve Özet**

**1. 🔁 Yerel ve Uzaktan İşleme (Local vs Remote Execution)**

* SaaS uygulamaları, bazı işlemleri **kullanıcının cihazında (yerel)** bazılarını ise **sunucuda (uzaktan)** yapabilir.
* **Yerel işlem** → Daha hızlı yanıt verir, **ağ trafiğini azaltır**, düşük internet hızlarında avantaj sağlar.
* Ancak yerel çalıştırma, cihazın daha fazla **elektrik ve işlem gücü tüketmesine** neden olur.

**2. 🔐 Kimlik Doğrulama Sorunları**

* Her uygulama kendi giriş sistemiyle gelirse, kullanıcıların **farklı şifreleri hatırlaması** gerekir → zorluk yaratır.
* Çözüm: Google, Facebook gibi üçüncü taraf kimlik doğrulama sistemleri kullanılabilir (**tek oturum açma**).
* Kurumsal uygulamalarda ise **federe kimlik doğrulama** gerekir (şirketin kendi kullanıcı doğrulama sistemini entegre etme).

**3. 🛡️ Bilgi Sızıntısı Riski**

* Aynı SaaS uygulamasını **birden fazla şirket** kullanabilir.
* Eğer sistem iyi tasarlanmazsa, **bir şirketin verileri başka bir şirkete sızabilir**.
* Bu nedenle, veri izolasyonu çok dikkatli bir şekilde sağlanmalıdır.

**4. 🏢 Çok Kiracılı (Multi-Tenant) ve Çok Örnekli (Multi-Instance) Sistemler**

**🔹 Çok Kiracılı Sistem (Multi-Tenant)**

* Tüm müşteriler **aynı yazılım ve veritabanını paylaşır**.
* Veriler tek bir veritabanında tutulur, ancak her müşterinin verileri **mantıksal olarak ayrılır** (kiracı tanımlayıcısı ile).
* Avantaj: **Kaynak kullanımı düşük**, maliyet etkin.
* Risk: Veri izolasyonu doğru yapılmazsa **gizlilik sorunları olabilir**.

**🔹 Çok Örnekli Sistem (Multi-Instance)**

* Her müşteri için **ayrı bir yazılım ve veritabanı kopyası** oluşturulur.
* Avantaj: Daha yüksek güvenlik ve özelleştirme imkânı.
* Dezavantaj: **Daha fazla kaynak** ve bakım gerektirir.

**🎯 Sınav İçin Özet Bilgi:**

| **Tasarım Sorunu** | **Açıklama** |
| --- | --- |
| Yerel/Uzaktan İşleme | Yerel işlem hızlıdır ama cihazı yorar; uzaktan işlem ağ bağımlıdır. |
| Kimlik Doğrulama | Kullanıcı dostu sistemler için tek oturum açma (SSO) önerilir. |
| Bilgi Sızıntısı Riski | Birden fazla şirketin verisi aynı sistemdeyse izolasyon çok önemlidir. |
| Çok Kiracılı Sistem | Tek yazılım/veritabanı, düşük maliyet, iyi izolasyon şart. |
| Çok Örnekli Sistem | Her müşteriye özel yapı, daha güvenli ama maliyetli. |

Çok kiracılı veritabanlarının ;

**Avantajlar:**

1. **Kaynak Kullanımı:** SaaS sağlayıcısı, yazılımı optimize edebilir ve kaynakları etkili bir şekilde yönetebilir, çünkü tüm kaynaklar üzerinde tam kontrol sağlar.
2. **Güvenlik:** Çok kiracılı veritabanları, tek bir veritabanında tüm müşterilerin verilerini tutarak, genellikle daha az güvenlik açığına sahiptir. Ayrıca, güvenlik açıkları keşfedildiğinde yalnızca tek bir veritabanı yazılımı güncellenir, bu da güvenlik yönetimini kolaylaştırır.
3. **Güncelleme Yönetimi:** Tek bir yazılım örneği güncellenebilir, böylece tüm müşterilere aynı anda güncelleme sunulabilir ve herkes en son sürümü kullanır.

**Dezavantajlar:**

1. **Esneklik:** Tüm müşteriler aynı veritabanı şemasını kullanmalıdır. Bu durum, şemayı bireysel ihtiyaçlara göre özelleştirme konusunda sınırlamalar getirir.
2. **Güvenlik:** Veriler aynı veritabanında tutulduğundan, teorik olarak bir müşterinin verilerinin diğerine sızması riski vardır. Ayrıca, bir güvenlik ihlali tüm müşterileri etkileyebilir.
3. **Karmaşıklık:** Çok kullanıcılı sistemler, kullanıcı yönetimi açısından daha karmaşık olabilir, bu da hata olasılığını artırabilir.

SaaS (Software as a Service) uygulamaları için olası özelleştirme seçenekleri açıklanmış. İşletmelerin, SaaS sağlayıcısının sunduğu yazılımı kendi ihtiyaçlarına göre nasıl uyarlayabileceği üzerine bilgiler verilmiş. Şu başlıklar altında olası özelleştirmeler ele alınmış:

1. **Kimlik Doğrulama:** İşletmeler, SaaS sağlayıcısının belirlediği hesap kimlik bilgileri yerine kendi şirket içi kimlik bilgilerini kullanarak kullanıcı kimlik doğrulama sürecini gerçekleştirebilirler. Bu sayede, şirketin güvenlik standartlarına ve yönetim sistemlerine uygun bir doğrulama süreci uygulanabilir.
2. **Markalaşma:** İşletmeler, yazılımın kullanıcı arayüzünü kendi markalarına uygun hale getirebilirler. Örneğin, logo, renk şeması ve şirketin kurumsal kimliğini yansıtan tasarımlar kullanılabilir.
3. **İş Kuralları:** İşletmeler, yazılımın içinde kendi iş kurallarını ve iş akışlarını tanımlayabilir. Bu, işletmenin özgün operasyonel ihtiyaçlarına göre yazılımın işleyişini şekillendirme imkanı sağlar.
4. **Veri Şemaları:** İşletmeler, yazılımda kullanılan standart veri modelini kendi iş ihtiyaçlarına göre genişletebilirler. Bu, veritabanı yapısının özelleştirilmesini ve işletmenin özel veri gereksinimlerine göre uyarlanmasını sağlar.
5. **Erişim Kontrolü:** İşletmeler, belirli kullanıcıların veya kullanıcı gruplarının erişebileceği veriler üzerinde kısıtlamalar koyabilir ve hangi işlemlere izin verildiğini belirleyebilirler. Bu, güvenlik ve veri yönetimi açısından büyük bir esneklik sağlar.

Bu özelleştirmeler, SaaS uygulamalarının daha esnek, güvenli ve işlevsel hale

İşletmelerin, çok kiracılı (multi-tenant) bir veritabanı modelinde esneklik sağlaması gereken durumlarla ilgili potansiyel zorluklar ele alınmıştır. Şimdi, bu konuları daha ayrıntılı şekilde inceleyelim:

**1. Veritabanına Ekstra Sütunlar Ekleme:**

* Bu yaklaşımda, veritabanına bazı ekstra sütunlar eklenerek her müşteriye (kiracıya) özel profiller oluşturulur. Yani, her müşteri için veri modelini genişletmek adına, veritabanı tablosuna yeni sütunlar eklenir ve her sütunun adı, müşteri gereksinimlerine göre özelleştirilir.

**Zorluklar:**

* **Ne kadar sütun eklemeniz gerektiğini bilmek zor:** Ekstra sütunları eklemek, müşterinin ihtiyaçlarına göre değişir. Eğer çok az sütun eklenirse, müşteriler daha fazla veri tutacak alan bulamayabilir. Tam tersi, çok fazla sütun eklerse, çoğu müşteri bunları kullanmadığı için veritabanında gereksiz yere alan harcanmış olur.
* **Farklı türde sütun ihtiyaçları:** Müşteriler farklı veri türlerine ihtiyaç duyabilir. Örneğin, bir müşteri dize türünde (string) sütunlar isterken, başka bir müşteri tam sayı (integer) türünde sütunlar isteyebilir. Bu durumda, veritabanını esnek tutmak zordur çünkü her müşteri için farklı veri türleri gereklidir.
* **Dizelerle genel çözüm:** Tüm verileri dize (string) türünde tutarak bu problemi aşmak mümkün olabilir. Ancak bu durumda, verilerin doğru türdeki öğelere dönüştürülmesi için yazılım tarafında ek dönüştürme işlemleri yapılması gerekir. Bu, ek iş yükü ve karmaşıklık yaratır.

**2. Tablolar Kullanarak Veritabanını Genişletme:**

* Alternatif bir yaklaşım, her müşterinin ihtiyacına göre ek alanlar eklemek yerine, bu alanları bir **ekstra tablo** aracılığıyla yönetmektir. Müşteriler istedikleri kadar ek alan ekleyebilirler ve bu alanlar için adlar, türler ve değerler başka bir tabloda saklanır.

**Zorluklar:**

* **Karmaşıklık artışı:** Bu yaklaşım veritabanı yönetimini karmaşıklaştırır. Ekstra tabloların yönetilmesi ve bu tablolardan gelen bilgilerin doğru şekilde ana veritabanına entegre edilmesi gerekir. Yani, veritabanının yönetimi ve sorgulama süreçleri daha karmaşık hale gelir.
* **Yönetim zorlukları:** Her müşteri için birden fazla ek alan oluşturulabileceğinden, bu verilerin yönetilmesi ve izlenmesi zorlaşır. Ekstra tabloların doğru şekilde güncellenmesi ve kullanılması, veritabanı yöneticilerine ek sorumluluk yükler.

**Özet:**

* **Ekstra sütunlar eklemek**, her müşteri için veri modelini doğrudan genişletmek anlamına gelir, ancak bu, fazla alan harcama ve tür uyumsuzlukları gibi sorunlara yol açabilir.
* **Tablolar kullanarak veritabanını genişletmek** ise daha esnek bir çözüm sunar çünkü her müşteri ihtiyacına göre ayrı bir tabloyla yönetilebilir, ancak bu yaklaşım veritabanı yönetimini daha karmaşık hale getirir.

ok kiracılı (multi-tenant) bir sistemdeki **güvenlik** risklerini ve bu riskleri yönetmek için alınması gereken önlemleri ele almaktadır. Şimdi daha ayrıntılı bir şekilde açıklayalım:

**Çok Kiracılı Sistemlerde Güvenlik Riskleri:**

* **Veri Paylaşımı ve İfşa Olma Riski:** Çok kiracılı bir sistemde, tüm müşterilerin (kiracıların) verileri aynı veritabanında saklanır. Bu nedenle, yazılımda bir hata veya kötü niyetli bir saldırı gerçekleşirse, bu durum bir ya da tüm müşterilerin verilerinin başkalarına ifşa olmasına yol açabilir. Yani, bir kiracının verileri yanlışlıkla veya kötü niyetli bir şekilde diğer kiracılara sızabilir.

**Temel Güvenlik Önlemleri:**

1. **Çok Seviyeli Erişim Kontrolü (Multi-level Access Control):**
   * **Kurumsal Düzeyde ve Bireysel Düzeyde Erişim Kontrolü:** Çok düzeyli erişim kontrolü, verilere erişimi hem kurumsal (şirket) düzeyinde hem de bireysel kullanıcı düzeyinde yönetmeyi ifade eder. Bu, yalnızca belirli kullanıcıların ve grupların verilere erişebilmesini sağlar.
     + **Kurumsal Düzeyde:** Bir organizasyonun veritabanına genel erişim kısıtlamaları getirilir. Örneğin, sadece belirli departmanlar veya yönetim kademesindeki kişiler veritabanını görebilir.
     + **Bireysel Düzeyde:** Kullanıcıların kendi erişim izinlerine göre, yalnızca belirli verilere erişebilmeleri sağlanır. Örneğin, bir çalışan sadece kendi departmanına ait verilere ulaşabilir, diğer departmanların verilerine ulaşamaz.
2. **Veritabanı İşlemleri ve Erişim İzinleri:**
   * Veritabanı işlemleri, yalnızca ilgili kuruluşun (müşterinin) verileri üzerinde yapılmalıdır. Yani, her müşteri yalnızca kendi verilerine erişebilmeli ve üzerinde işlem yapabilmelidir. Ayrıca, her kullanıcı sadece kendisine tanımlı erişim izinlerine göre veri görebilmeli ve işlemler yapabilmelidir. Bu, verilerin güvenliğini sağlamak için temel bir ilkedir.
3. **Veri Şifreleme:**
   * Çok kiracılı bir veritabanındaki verilerin **şifrelenmesi**, özellikle kurumsal kullanıcılar için önemlidir. Şifreleme, bir tür sistem arızası veya güvenlik ihlali durumunda, bir şirketin verilerinin diğer şirketlerin kullanıcıları tarafından görüntülenmesini engeller. Şifreleme sayesinde, veritabanı üzerinde yetkisiz erişim sağlanmış olsa bile, verilerin okunması mümkün olmaz. Bu, verilerin güvenliğini sağlayarak, gizliliği korur.

**Özet:**

* Çok kiracılı sistemlerde güvenlik büyük bir öneme sahiptir çünkü tüm müşterilerin verileri aynı veritabanında tutulur. Bu da potansiyel bir yazılım hatası veya saldırı durumunda verilerin ifşa olma riskini artırır.
* **Çok seviyeli erişim kontrolü** ve **şifreleme**, verilerin güvenliğini sağlamanın temel yollarıdır. Erişim kontrolü, veriye sadece yetkili kişilerin erişmesini sağlar, şifreleme ise verilerin dışarıdan erişimlere karşı korunmasını sağlar.
* Bu güvenlik önlemleri, veritabanı yönetiminin doğru yapılmasını ve her bir müşteri için verilerin gizliliğinin korunmasını mümkün kılar.

**Çok Örnekli Veritabanları (Multi-instance Databases):**  
Bu bölümde, çok örnekli (multi-instance) sistemlerin ne olduğunu ve bunların **SaaS (Software as a Service)** çözümlerindeki rolünü açıklamaktadır. Çok örnekli sistemler, her müşterinin (kiracının) tamamen kendi özel veritabanına ve güvenlik önlemlerine sahip olduğu yazılım sistemleridir. Bu, çok kiracılı (multi-tenant) sistemlerden farklıdır, çünkü her müşteri kendi bağımsız sistemini kullanır. Şimdi, bu kavramları daha ayrıntılı olarak inceleyelim:

**Çok Örnekli Sistemler:**

* **Bağımsız Veritabanı ve Güvenlik Kontrolleri:**  
  Her müşterinin kendi veritabanı ve güvenlik yapılandırması vardır. Bu, her müşterinin verilerinin tamamen izole olduğu anlamına gelir. Yani, müşteriler birbirlerinin verilerine erişemezler ve kendi veritabanlarına yalnızca kendi kullanıcıları erişebilir. Bu durum, güvenlik açısından büyük bir avantaj sağlar çünkü bir müşterinin verileri, diğerlerinden izole edilir.
* **Güvenlik Endişeleri Daha Azdır:**  
  Çok örnekli sistemlerde, bir kurumdan diğerine veri sızıntısı gibi güvenlik endişelerinden kaçınılır. Çünkü her müşteri, tamamen kendi veritabanına ve sistemine sahip olduğundan, bir müşterinin verileri, diğer müşterilere sızmaz. Bu, çok kiracılı sistemlerdeki potansiyel veri ifşası risklerinden kaçınmak anlamına gelir.

**Çok Örnekli Sistemlerin İki Türü:**

1. **VM (Virtual Machine) Tabanlı Çok Örnekli Sistemler:**
   * **Açıklama:**  
     Bu sistemlerde, her müşteri için yazılım ve veritabanı kendi **sanal makinesinde** çalışır. Her sanal makine, bir müşteriye özel olup, diğer müşterilerden tamamen izole edilmiştir.
   * **Paylaşılan Veritabanı Erişimi:**  
     Aynı müşterinin tüm kullanıcıları, kendi sanal makinelerindeki veritabanına erişebilirler. Bu, aynı müşteri için birden fazla kullanıcı olsa bile, veri paylaşımı sadece o müşteri ile sınırlıdır.
   * **Avantajlar:**  
     Müşteriler birbirinin verilerine erişemez, çünkü her biri kendi sanal makinesinde çalışır. Bu, güvenliği artırır ve veri izolasyonunu sağlar.
2. **Konteyner Tabanlı Çok Örnekli Sistemler:**
   * **Açıklama:**  
     Bu sistemlerde, her kullanıcıya bir dizi **izole edilmiş yazılım ve veritabanı sürümü** verilir. Yani her kullanıcı (veya her müşteri) bağımsız bir ortamda çalışır, ancak bu ortamlar konteynerler içinde birbirlerinden ayrılır.
   * **Konteynerler:**  
     Konteynerler, sanal makinelerden daha hafif ve daha hızlıdır, çünkü aynı işletim sistemi çekirdeği üzerinde çalışırlar, ancak her biri izole bir ortamda kendi uygulamalarını çalıştırır.
   * **Uygulama Alanı:**  
     Bu yaklaşım, çoğunlukla bağımsız çalışan kullanıcılar ve az veri paylaşımı olan uygulamalar için uygundur. Örneğin, bireylere hizmet veren yazılımlar veya veri yoğun olmayan ticari ürünler için kullanılır. Müşteriler arasında minimal veri paylaşımı olduğunda konteyner tabanlı çok örnekli sistemler daha verimli olabilir.

, **çok örnekli veritabanlarının avantajları ve dezavantajları** üzerinde duruyor. Şimdi, her bir maddeyi açıklayalım ve özetleyelim:

**Avantajlar:**

1. **Esneklik:**
   * **Uyarlanabilirlik ve Özelleştirilebilirlik:** Çok örnekli sistemlerde her bir yazılım örneği, **müşterinin ihtiyaçlarına göre uyarlanabilir ve adapte edilebilir**. Yani, her müşteri, kendi özel gereksinimlerine uygun veritabanı şemalarını kullanabilir. Bu, **özelleştirme** anlamında büyük bir avantaj sağlar.
   * Ayrıca, bir müşteri veritabanından başka bir veritabanına veri aktarmak oldukça kolaydır. Bu esneklik, farklı iş süreçlerine ve gereksinimlere sahip müşteriler için oldukça faydalıdır.
2. **Güvenlik:**
   * Her müşteri için **bağımsız bir veritabanı** olduğundan, **bir müşteriden diğerine veri sızıntısı olasılığı yoktur**. Bu, güvenliği büyük ölçüde artırır çünkü her müşteri verisi izole edilmiştir ve birbirlerinin verilerine erişim mümkün değildir.
3. **Ölçeklenebilirlik:**
   * Çok örnekli sistemler, **bireysel müşterilerin ihtiyaçlarına göre ölçeklendirilebilir**. Yani, bir müşterinin daha güçlü bir sunucuya ihtiyacı varsa, sistem bu ihtiyaca göre ölçeklendirilebilir. Örneğin, büyük bir müşterinin daha fazla işlem gücüne ihtiyacı olduğunda, ona daha güçlü bir sanal makine (VM) atanabilir.
4. **Dayanıklılık:**
   * Eğer bir yazılım arızası meydana gelirse, bu **muhtemelen yalnızca tek bir müşteriyi etkileyecektir**. Diğer müşteriler normal şekilde çalışmaya devam edebilir. Yani, bir müşterinin yaşadığı sorun, diğerlerinin hizmet almasını engellemez. Bu, **hizmet sürekliliği** açısından büyük bir avantajdır.

**Dezavantajlar:**

1. **Maliyet:**
   * **Sanal makinelerin maliyeti:** Bulut üzerinde birden fazla sanal makine kiralamanın maliyeti yüksektir. Ayrıca, çok örnekli sistemde birden fazla sistemi yönetmek, ek iş gücü ve kaynak gerektirir. Her müşteri için ayrı sanal makineler kullanılması gerektiği için, bu durum maliyetleri artırır.
   * **Yavaş başlatma süresi:** Yeni bir sanal makine başlatmak zaman alabilir. Özellikle çok az talep olduğunda, sanal makinelerin kiralanması ve sürekli çalışır durumda tutulması gerekebilir, bu da kaynak israfına ve **işlem süresinin uzamasına** yol açar.
2. **Güncelleme Yönetimi:**
   * Yazılım güncellemeleri **daha karmaşık hale gelir** çünkü her bir örnek güncellenmek zorundadır. Özellikle, her örnek belirli müşterilerin ihtiyaçlarına göre özelleştirildiyse, her müşteri için ayrı güncellemeler yapılması gerekebilir. Bu da zaman alıcı ve yönetilmesi zor bir süreçtir.

**Veritabanı Kuruluşu Seçerken Sorulacak Sorular (Tablo 5.8):**

1. **Hedef Müşteriler:**
   * **Soru:** Müşteriler, farklı veritabanı şemaları ve kişiselleştirme talepleriyle mi geliyor?  
     **Açıklama:** Eğer müşterilerin veritabanı üzerinde kişiselleştirilmiş şemalara ihtiyacı varsa veya veritabanı paylaşımında güvenlik endişeleri varsa, **çok örnekli bir veritabanı** kullanılabilir. Çünkü çok örnekli veritabanları, her müşteri için izole bir veritabanı sağlar ve güvenlik endişelerini azaltır.
2. **İşlem Gereksinimleri:**
   * **Soru:** Ürününüzün, verilerin her zaman tutarlı olması gereken **ACID işlemlerini** desteklemesi kritik mi?  
     **Açıklama:** ACID (Atomicity, Consistency, Isolation, Durability) işlemleri, veritabanı işlemlerinin güvenilirliğini ve tutarlılığını sağlar. Bu durumda, **çok kiracılı** veya **VM tabanlı çok örnekli** bir veritabanı seçmek daha uygun olabilir.
3. **Veritabanı Boyutu ve Bağlanabilirlik:**
   * **Soru:** Müşteriler tarafından kullanılan veritabanları ne kadar büyük? Veritabanı öğeleri arasında kaç ilişki var?  
     **Açıklama:** **Çok kiracılı model**, büyük veritabanları için daha uygun olabilir çünkü bu model, performansı optimize etmek ve büyük veri kümelerini verimli bir şekilde yönetmek için daha etkili olabilir.
4. **Veritabanı Birlikte Çalışabilirliği:**
   * **Soru:** Müşteriler, mevcut veritabanlarından veri aktarmayı mı bekliyor? Bu veritabanları ile çok kiracılı bir veritabanı arasındaki şema farkları nelerdir?  
     **Açıklama:** Müşterilerin **veri aktarımına** ihtiyaç duyduğu durumlarda, veritabanı şemaları arasında uyumsuzluklar olabilir. Eğer çok sayıda farklı veritabanı şeması varsa, **çok örnekli bir veritabanı** daha iyi bir seçenek olabilir.
5. **Sistem Yapısı:**
   * **Soru:** Sisteminiz hizmet odaklı bir mimari kullanıyor mu?  
     **Açıklama:** Eğer bir sistemde **hizmet odaklı mimari** kullanılıyorsa ve müşteri veritabanları bir dizi bireysel hizmet veritabanına bölünebiliyorsa, **kapsayıcı, çok örnekli veritabanları** tercih edilebilir.

**Ölçeklenebilirlik ve Esneklik:**

1. **Ölçeklenebilirlik:**
   * **Açıklama:**  
     Ölçeklenebilirlik, bir sistemin **yükteki değişikliklere** (örneğin daha fazla kullanıcı veya veri ile karşılaşıldığında) **otomatik olarak uyum sağlama yeteneğidir**. Yani, sistemin verimli çalışabilmesi için gereken kaynaklar (sunucular, bellek vb.) kolayca artırılabilir.
   * **Ölçekleme Yöntemleri:**
     + **Dikey ölçekleme (Vertical scaling):** Sistemin mevcut sunucusunun gücünü artırarak (daha fazla CPU veya bellek ekleyerek) kapasite artırılabilir.
     + **Yatay ölçekleme (Horizontal scaling):** Yeni sunucular ekleyerek (veya sanal makineler ekleyerek) kapasite artırılabilir. Bu, bulut tabanlı sistemlerde yaygın bir yaklaşımdır ve yazılımın paralel olarak çalışabilmesi için uygun hale getirilmesi gerekir.
2. **Esneklik:**
   * **Açıklama:**  
     Esneklik, bir **donanım veya yazılım arızası** durumunda, sistemin **hızla yeniden başlatılabilmesi** ve kritik hizmetlerin kesintisiz olarak sunulmaya devam etmesidir. Bu, özellikle hizmetlerin sürekli erişilebilir olmasını sağlayan önemli bir özelliktir.
   * **Esneklik Sağlama:**  
     Esneklik sağlamak için, sistemin **yedeğe alınması** ve hızlı bir şekilde yeniden yapılandırılması gerekir. Böylece herhangi bir arıza veya kesinti durumunda hizmetler hızlıca eski haline dönebilir.

**Esneklik ve Yedeklilik:**

* **Yedeklilik: Esneklik, yedekliliğe dayanır. Bu, sistemin kesintisiz çalışabilmesi için yazılım ve verilerin farklı yerlerde kopyalarının bulundurulması anlamına gelir. Yedekleme, sistem arızası durumunda hizmetin sürekliliğini sağlar.**
* **Veritabanı Yedekliliği: Veritabanı güncellemeleri, yedek veritabanına yansıtılır. Yedek veritabanı, operasyonel veritabanının bir kopyası olmalı, böylece bir arıza durumunda yedek sisteme geçiş yapılabilir.**
* **Sistem Monitörü: Bir sistem monitörü, sürekli olarak sistemin durumunu kontrol eder. Eğer operasyonel sistem arızalanırsa, otomatik olarak yedek sisteme geçiş yapılır. Bu, kesintisiz hizmetin sağlanmasında çok önemlidir.**

**Dayanıklılık:**

* **Fiziksel Sunucular ve Yedekleme: Yedek sanal sunucular aynı fiziksel bilgisayarda barındırılmamalıdır. Bunun yerine, farklı konumlarda bulunan yedek sunucular kullanılmalıdır. Bu sayede, bir fiziksel sunucunun arızalanması durumunda, diğer sunucularda çalışan yazılım kopyaları devreye girebilir.**
* **Fiziksel Sunucu Arızaları ve Veri Merkezleri: Eğer bir veri merkezi büyük bir arıza yaşarsa, yazılım kopyaları başka bir veri merkezine yönlendirilerek kesintisiz hizmet sağlanır. Bu tür yedeklilik, sistemin dayanıklılığını artırır.**

**Sistem Yapısı ve Monolitik Mimariler:**

* **Monolitik Sistemler: Geleneksel sistemler, genellikle monolitik yapılara sahip olup, tüm bileşenler tek bir sunucuda ya da birden fazla sunucuda tek bir büyük yazılım uygulaması olarak çalışır. Bu yapı, istemci-sunucu mimarisine dayalıdır.**
* **Hizmet Odaklı Yaklaşım: Bir alternatif olarak, sistemin hizmet odaklı bir yapıya dönüştürülmesi mümkündür. Bu, sistemi daha küçük, bağımsız, durumsuz hizmetlere ayırır. Her hizmet bağımsız çalışabilir, çoğaltılabilir ve başka sunuculara taşınabilir.**
  + **Durumsuz Hizmetler: Durumsuz hizmetler, her bir hizmetin kendi durumu hakkında bilgi tutmaması ve başka bir sunucuya kolayca taşınabilmesi anlamına gelir.**
  + **Hizmetlerin Konteynerlerde Çalışması: Hizmet odaklı yaklaşım, özellikle kapsayıcılar (containers) kullanılarak bulut tabanlı yazılımlarda uygulanabilir. Konteynerler, hizmetlerin bağımsız ve taşınabilir olmasını sağlar.**

Kilit noktalar 1

• Bulut, kendi kullanımınız için kiralayabileceğiniz çok sayıda sanal sunucudan oluşur.

Siz ve müşterileriniz bu sunuculara internet üzerinden uzaktan erişir ve kullanılan

sunucu süresi için ödeme yaparsınız.

• Sanallaştırma, birden fazla sunucu örneğinin aynı fiziksel bilgisayar üzerinde çalıştırılmasına olanak tanıyan bir teknolojidir. Bu, bulut üzerinde dağıtım için yazılımınızın izole edilmiş örneklerini oluşturabileceğiniz anlamına gelir.

• Sanal makineler, üzerinde kendi işletim sisteminizi, teknoloji yığınınızı ve uygulamalarınızı çalıştırdığınız fiziksel sunucu kopyalarıdır.

• Konteynerler, sanal sunucuların hızlı bir şekilde çoğaltılmasına ve dağıtılmasına olanak tanıyan hafif bir sanallaştırma teknolojisidir. Tüm konteynerler aynı işletim sistemini çalıştırır. Docker şu anda en yaygın kullanılan konteyner teknolojisidir.

• Bulutun temel bir özelliği, 'her şeyin' bir hizmet olarak sunulabilmesi ve internet üzerinden erişilebilmesidir. Bir hizmet sahip olunmak yerine kiralanır ve diğer kullanıcılarla paylaşılır.

Kilit noktalar 2

• Hizmet olarak altyapı (IaaS), bilgi işlem, depolama ve diğer hizmetlerin bulut üzerinden sunulması anlamına gelir. Kendi fiziksel sunucularınızı çalıştırmanıza gerek yoktur.

• Hizmet olarak platform (PaaS), yazılımınızı talebe göre otomatik ölçeklendirmeyi mümkün kılmak için bir bulut platformu satıcısı tarafından sağlanan hizmetleri kullanmak anlamına gelir.

• Hizmet olarak yazılım (SaaS), uygulama yazılımının kullanıcılara bir hizmet olarak sunulması anlamına gelmektedir. Bunun kullanıcılar için daha düşük sermaye maliyetleri ve yazılım satıcıları için yeni yazılım sürümlerinin daha kolay dağıtımı gibi

önemli faydaları vardır.

• Çok kullanıcılı sistemler, tüm kullanıcıların aynı veritabanını paylaştığı ve çalışma zamanında bireysel ihtiyaçlarına göre uyarlanabilen SaaS sistemleridir. Çok örnekli sistemler, her kullanıcının kendi ayrı veritabanına sahip olduğu SaaS uygulamalarıdır.

• Bulut tabanlı yazılımlar için temel mimari konular, kullanılacak bulut platformu, çok kiracılı veya çok örnekli bir veritabanı kullanılıp kullanılmayacağı, ölçeklenebilirlik ve esneklik gereksinimleri ve sistemdeki temel bileşenler olarak nesnelerin mi yoksa hizmetlerin mi kullanılacağıdır.