

Daphne Kariyer online eğitim · Dijital yetkinlikler · Bulut bilişim ve altyapı

Bulut Bilişim ve Altyapı Temelleri

Modül	10
Önerilen süre	Yaklaşık 10–12 saat
Düzey	Başlangıç
Biçim	Online, kendi hızında
Ölçme	Daphne Kariyer ücretsiz testleri

İçindekiler

Modül 1	Bilgi işlem altyapısının temelleri
Modül 2	Bulut bilişim nedir?
Modül 3	Büyük sağlayıcılar ve temel hizmet kategorileri
Modül 4	Ağ temelleri
Modül 5	Linux ve Windows Server ile sistem yönetimine giriş
Modül 6	Bulutta güvenlik, kimlik ve erişim
Modül 7	Maliyet yönetimi
Modül 8	Konteynerler ve Docker
Modül 9	Kubernetes'e giriş ve DevOps kültürü
Modül 10	Buluta geçiş, yedekleme, felaket kurtarma ve veri konumu
Kendini dene cevapları · Sözlük · Sonraki adım	

Bu eğitim hakkında

Bu eğitim, bilgi işlem altyapısına ve bulut bilişime yeni başlayanlar için hazırlandı. BT destek ekibinde çalışan ya da çalışmak isteyen, sistem yöneticisi olmaya hazırlanan veya şirketinin buluta geçişinde görev alan kişiler düşünülerek yazıldı. Önceden sunucu kurmuş olman gerekmez. Bilgisayar kullanmayı, dosya ve klasör mantığını, internete bağlanmanın ne demek olduğunu bilmen yeterli.

Eğitim on modülden oluşur. İlk modüller sunucu, depolama, ağ ve sanallaştırma gibi temel kavramları anlatır. Ardından bulut bilişimin ne olduğuna, büyük sağlayıcılara, ağ yapısına ve sunucu yönetimine geçer. Son modüller güvenlik, maliyet, konteynerler, Kubernetes, DevOps ve buluta geçiş planlamasıyla ilgilidir. Her modülde gerçekçi bir iş örneği, adım adım bir uygulama, sık yapılan hatalar ve beş soruluk bir "Kendini dene" bölümü var. Cevaplar eğitimin sonunda. Komut ve yapılandırma örnekleri kısa tutuldu; hepsi bilgi okuyan ya da kendi deneme ortamında güvenle çalıştırabileceğin türden.

Eğitim kendi hızında okunmak için hazırlandı. Örnek komutları denemek için bir sanal makine ya da sağlayıcıların ücretsiz deneme hesaplarından biri yeterlidir; deneme hesabı açarsan bütçe uyarısını ilk gün kur (Modül 7).

Önerilen süre: yaklaşık 10–12 saat (okuma, örnekler ve alıştırımlar dahil).

Eğitimin sonunda şunları yapabileceksin:

- Sunucu, depolama, ağ ve sanallaştırmanın ne işe yaradığını ve birbirine nasıl bağlandığını açıklamak.
- Bulut bilişimin temel özelliklerini, IaaS, PaaS ve SaaS farkını, genel, özel ve hibrit bulutu ayırt etmek.
- AWS, Microsoft Azure ve Google Cloud'daki temel hizmet kategorilerini doğru adlarıyla tanımak.
- IP adresi, alt ağ, DNS ve yük dengeleyici kavramlarını kullanarak basit bir bulut ağını okumak.
- Linux ve Windows Server'da temel durum kontrolü komutlarını çalıştırmak ve çıktısını yorumlamak.
- Paylaşılan sorumluluk modelini, en az yetki ilkesini ve çok faktörlü kimlik doğrulamayı uygulamak.
- Etiketleme ve bütçe uyarısıyla bulut maliyetini izlemek.
- Konteyner, Docker imajı ve Kubernetes'in temel nesnelerini açıklamak; basit bir Dockerfile okumak.
- CI/CD ve DevOps kültürünün ekip çalışmasına ne kattığını anlatmak.
- Buluta geçiş, yedekleme ve felaket kurtarma için temel bir plan çıkarmak; KVKK açısından veri konumunu sorgulamak.

Bilgi işlem altyapısının temelleri

Bir şirketin e-postası, muhasebe programı, web sitesi ya da stok takip sistemi bir yerde çalışır. O "yer", donanım ve yazılımdan oluşan bilgi işlem altyapısıdır. Bulut bilişimi anlamak için önce bu altyapının parçalarını tanıman gerekir, çünkü bulut bu parçaları ortadan kaldırmaz; onları başkasının veri merkezinde, hizmet olarak sunar.

Sunucu nedir?

Sunucu (server), başka bilgisayarlara ya da kullanıcılara hizmet veren bilgisayardır. Masaüstü bilgisayarla aynı temel parçalara sahiptir ama kesintisiz çalışmak üzere tasarlanır.

- **İşlemci (CPU):** Hesaplamayı yapan birim. Sunucularda çekirdek sayısı yüksektir; aynı anda çok sayıda isteği karşılar.
- **Bellek (RAM):** Çalışan programların ve verinin geçici olarak tutulduğu hızlı alan. Sunucu kapanınca içeriği silinir.
- **Disk:** Kalıcı depolama. HDD (dönen plakalı disk) daha ucuz ve yavaş, SSD (yarı iletken disk) daha hızlıdır.
- **Ağ kartı (NIC):** Sunucuyu ağa bağlar. Sunucularda genellikle birden fazla bulunur.
- **Yedekli güç kaynağı:** Biri arızalanırsa diğeri devreye girer.

Sunucular genellikle dolap (rack) tipindedir: standart genişlikte, yatay olarak kabinlere vidalanan kasalar. Bir kabinde onlarca sunucu, ağ anahtarı ve güç dağıtım birimi bulunabilir.

Sunucunun yaptığı işe göre adı değişir: dosya sunucusu, veritabanı sunucusu, web sunucusu, e-posta sunucusu, etki alanı denetleyicisi (domain controller). Aynı fiziksel makine birden çok rol üstlenebilir ama rolleri ayırmak arızada etkilenen alanı küçültür.

Veri merkezi

Veri merkezi, sunucuların bulunduğu ve kesintisiz çalışmaları için gerekli koşulların sağlandığı tesistir. Küçük bir şirkette bu bir "sistem odası" olabilir; büyük kurumlarda ayrı binalar kullanılır.

Bir veri merkezinde şunlar bulunur:

- Kesintisiz güç kaynağı (UPS) ve jeneratör: Elektrik kesildiğinde sistemleri ayakta tutar.
- Soğutma: Sunucular ısı üretir; sıcaklık yükselirse donanım kendini korumak için yavaşlar ya da kapanır.
- Fiziksel güvenlik: Kartlı giriş, kamera, ziyaretçi kaydı.
- Yangın algılama ve söndürme sistemleri.
- Birden fazla internet bağlantısı ve operatör.

Bütün bunlar paraya, alana ve uzman personele mal olur; bulut bu yükü sağlayıcıya devreder.

Depolama türleri

Veri farklı biçimlerde saklanır. Bulut hizmetlerini seçerken de aynı ayrımla karşılaşacaksın.

- **Blok depolama:** Disk gibi davranır. İşletim sistemi onu biçimlendirir ve üzerine dosya sistemi kurar. Veritabanları ve işletim sistemi diskleri için uygundur.

- **Dosya depolama:** Klasör ve dosya yapısıyla, ağ üzerinden paylaşılır. Ofisteki ortak "Muhasebe" klasörü buna örnektir. Windows ağlarında SMB, Linux ağlarında NFS protokolü yaygındır.
- **Nesne depolama:** Her dosya bir "nesne" olarak, benzersiz bir anahtar ve ek bilgilerle (metadata) saklanır. Klasör hiyerarşisi yoktur ya da yalnız görünüştedir. Yedekler, fotoğraflar, belgeler, log arşivleri gibi büyük ve çok sayıda dosya için uygundur.

Şirket içi altyapıda bu türler farklı cihazlarla sağlanır:

- **DAS (doğrudan bağlı depolama):** Diskler doğrudan sunucuya takılıdır.
- **NAS (ağa bağlı depolama):** Dosya paylaşımı yapan, ağa bağlı depolama cihazı.
- **SAN (depolama alan ağı):** Sunuculara blok düzeyinde disk sunan, genellikle ayrı bir ağ üzerinde çalışan depolama sistemi.

RAID: disk arızasına karşı önlem

RAID, birden fazla diski tek bir mantıksal disk gibi kullanma yöntemidir. Amaç hız, arızaya dayanıklılık ya da ikisidir.

- **RAID 1 (ayna):** Veri iki diske aynen yazılır. Bir disk bozulursa diğeri çalışmaya devam eder.
- **RAID 5:** Veri ve eşlik (parity) bilgisi en az üç diske dağıtılır. Bir disk arızasına dayanır.
- **RAID 10:** Aynalama ve şeritlemenin birleşimi. Hem hızlı hem dayanıklıdır, ama disklerin yarısı kopya için kullanılır.

RAID yedek değildir. Yanlışlıkla silinen bir dosya, fidye yazılımının şifrelediği bir klasör ya da bozulan bir veritabanı bütün disklere aynı anda yansır. RAID yalnız disk arızasına karşı korur.

Ağ cihazları

- **Anahtar (switch):** Aynı yerel ağdaki cihazları birbirine bağlar.
- **Yönlendirici (router):** Farklı ağlar arasında trafiği yönlendirir; ofis ağını internete bağlar.
- **Güvenlik duvarı (firewall):** Hangi trafiğin geçeceğine kurallara göre karar verir.
- **Kablosuz erişim noktası:** Kablosuz cihazları ağa bağlar.

Ağ konusuna Modül 4'te ayrıntılı döneceğiz.

Sanallaştırma

Eskiden her uygulama için ayrı bir fiziksel sunucu alınırdı. Bu sunucuların çoğu zamanın büyük kısmında kapasitesinin küçük bir bölümünü kullanırdı. Sanallaştırma bu israfı çözer.

Hipervizör (hypervisor), tek bir fiziksel makinenin kaynaklarını (işlemci, bellek, disk, ağ) birden fazla **sanal makineye (VM)** bölen yazılımdır. Her sanal makine kendi işletim sistemini çalıştırır ve kendini ayrı bir bilgisayar sanır.

- **Tip 1 hipervizör:** Doğrudan donanım üzerinde çalışır. VMware ESXi, Microsoft Hyper-V ve Linux'taki KVM bu gruptadır. Veri merkezlerinde ve bulut sağlayıcılarında kullanılan budur.
- **Tip 2 hipervizör:** Normal bir işletim sisteminin üstünde, uygulama olarak çalışır. VirtualBox ve VMware Workstation örnektir. Eğitim ve deneme için uygundur.

Sanallaştırmanın getirdikleri:

- Bir fiziksel sunucuda çok sayıda sanal makine çalışır; donanım daha verimli kullanılır.
- Yeni sunucu ihtiyacı satın alma beklemeden, dakikalar içinde karşılanır.

- **Anlık görüntü (snapshot)** ile bir sanal makinenin o anki durumu kaydedilir; güncelleme bozulursa geri dönlür.
- Sanal makine bir fiziksel sunucudan diğetine taşınabilir; donanım bakımı için kesinti gerekmez.

Bulut bilişim, sanallaştırmanın büyük ölçekte, otomasyonla ve self servis olarak sunulmuş hâlidir.

Sermaye gideri ve işletme gideri

Şirket içi altyapıda sunucu, depolama ve ağ cihazı peşin satın alınır. Bu bir **sermaye gideridir (CapEx)**. Donanım birkaç yıl kullanılır, sonra yenilenir. Kapasite ihtiyacı tahmin edilerek alınır: az alırsan yetmez, çok alırsan boşa kalır.

Bulutta ise kaynak kiralanır ve kullanıldığı kadar ödenir. Bu bir **işletme gideridir (OpEx)**. Bu fark yalnız muhasebe meselesi değildir; planlama biçimini de değiştirir. Modül 7'de maliyet yönetimine ayrıca bakacağız.

Örnek vaka

Bursa'da 35 kişilik bir tekstil firmasının sistem odasında üç fiziksel sunucu var: biri dosya paylaşımı ve etki alanı denetleyicisi, biri muhasebe programının veritabanı, biri de eski bir üretim takip uygulaması. Sunucular yedi yıllık. Muhasebe sunucusunun bir diski geçen ay bozuldu; RAID 1 olduğu için veri kaybolmadı ama yeni diski bulmak bir hafta sürdü. Bu süre boyunca sunucu tek diskle çalıştı ve ikinci bir arıza tüm veriyi götürebilirdi.

Firmanın BT sorumlusu yönetime şu tespitleri sunuyor:

1. Donanım ömrünü doldurmuş; parça bulmak zorlaşıyor.
2. Üç sunucu da kapasitesinin küçük bir kısmını kullanıyor.
3. Yedekler aynı odadaki bir harici diske alınıyor; yangın ya da hırsızlıkta yedek de kaybolur.
4. Sistem odasının kliması tek ve bakım kaydı yok.

Önerilen ilk adım, üç sunucuyu tek bir yeni fiziksel sunucuda sanal makine olarak çalıştırmak ya da doğrudan buluta taşımaktır. Hangisinin seçileceği maliyet, internet bağlantısının kalitesi ve uygulamaların bulutta çalışıp çalışmadığına bağlıdır. Bu soruların nasıl yanıtlanacağını Modül 10'da göreceksin.

Uygulama

Bulunduğun iş yerindeki ya da kendi bilgisayarındaki altyapıyı tanıyarak başla. Aşağıdaki envanter listesini doldur:

1. Hangi sunucular ya da bilgisayarlar sürekli açık duruyor? Her biri ne iş yapıyor?
2. Her birinde kaç çekirdek, ne kadar bellek ve ne kadar disk var?
3. Diskler RAID ile mi korunuyor? Hangi RAID düzeyi?
4. Yedek nereye, ne sıklıkla alınıyor? Son kez ne zaman geri yükleme denendi?
5. Sanallaştırma kullanılıyor mu? Hangi hipervizör?
6. Donanım ne zaman alındı, garanti ne zaman bitiyor?

Bu liste, ileride buluta geçiş planı yaparken başlangıç noktası olacak. Kendi bilgisayarında denemek istersen VirtualBox gibi bir tip 2 hipervizör kur ve bir Linux sanal makinesi oluştur. Sonraki modüllerdeki komutları bu makinede güvenle deneyebilirsin.

Sık yapılan hatalar

- RAID'i yedek sanmak. RAID disk arızasına karşı korur; silme, bozulma ve fidye yazılımına karşı korumaz.
- Snapshot'ı uzun süreli yedek olarak kullanmak. Anlık görüntüler genellikle aynı depolamada durur ve uzun süre tutulduğunda performansı düşürebilir.
- Sistem odasındaki tek bir cihaza (tek klima, tek UPS, tek internet hattı) bağımlı kalmak.
- Sunucuların hangi işi yaptığını ve kimin sorumlu olduğunu yazılı tutmamak. Envanteri olmayan altyapı ne güvenle yönetilir ne de taşınır.
- Kapasiteyi "nasılsa yeter" diye tahminle belirlemek; kullanım verisini ölçmemek.

Kendini dene

1. Bellek (RAM) ile disk arasındaki temel fark nedir?
2. Blok, dosya ve nesne depolamadan hangisi binlerce fotoğraf ve yedek dosyasını saklamak için en uygundur?
3. RAID 1 kullanan bir sunucuda yanlışlıkla silinen bir dosya kurtarılabilir mi? Neden?
4. Tip 1 ve tip 2 hipervizör arasındaki fark nedir?
5. Peşin donanım satın almak hangi gider türüne girer?

Bulut bilişim nedir?

Bulut bilişim, bilgi işlem kaynaklarının (sunucu, depolama, veritabanı, ağ, yazılım) internet üzerinden, ihtiyaç duyulduğunda kiralanması ve kullanıldığı kadar ödenmesidir. Kaynaklar sağlayıcının veri merkezlerinde durur; sen bir web konsolundan, komut satırından ya da programla onları oluşturur, değiştirir ve silersin.

"Bulut" sözcüğü yanıltıcı olabilir. Veri havada durmaz; belirli bir ülkedeki, belirli bir veri merkezindeki fiziksel disklerde durur.

Beş temel özellik

ABD Ulusal Standartlar ve Teknoloji Enstitüsü (NIST), bulut bilişimi beş temel özelliklerle tanımlar. Bu tanım sektörde yaygın olarak kabul edilir:

1. **İsteğe bağlı self servis:** Kaynağı kimseyle görüşmeden, kendin oluşturursun. Yeni bir sanal sunucu için satış temsilcisine e-posta atmazsın; konsolda birkaç tıklama yaparsın.
2. **Geniş ağ erişimi:** Hizmetlere standart yöntemlerle, internet üzerinden, farklı cihazlardan ulaşılır.
3. **Kaynak havuzu:** Sağlayıcının donanımı çok sayıda müşteri arasında paylaşılır. Senin sanal makinenin hangi fiziksel sunucuda çalışıyor, genellikle bilmezsin ve bilmen gerekmez.
4. **Hızlı esneklik:** Kaynak ihtiyaca göre hızla artırılır ya da azaltılır. Kampanya döneminde sunucu sayısı artar, kampanya bitince azalır.
5. **Ölçülen hizmet:** Kullanım ölçülür ve faturalanır. Ne kadar işlemci saati, ne kadar depolama, ne kadar veri transferi kullandığını görebilirsin.

Hizmet modelleri: IaaS, PaaS, SaaS

Bulut hizmetleri, sağlayıcının ne kadarını yönettiğine göre üç ana modele ayrılır. Bunu bir yemek örneğiyle düşünebilirsin: mutfağı kiralayıp yemeği kendin pişirmek (IaaS), hazır malzemeye yalnız son dokunuşu yapmak (PaaS), restoranda oturup yemeği yemek (SaaS).

- **IaaS (Infrastructure as a Service, hizmet olarak altyapı):** Sağlayıcı sana sanal makine, disk ve ağ verir. İşletim sistemini, güncellemeleri, uygulamayı ve veriyi sen yönetirsin. Şirket içindeki sunucuna en çok benzeyen modeldir. Örnek: Amazon EC2, Azure Virtual Machines, Google Compute Engine.
- **PaaS (Platform as a Service, hizmet olarak platform):** Sağlayıcı işletim sistemini ve çalışma ortamını da yönetir. Sen yalnız uygulamayı ve verini getirirsin. Veritabanı yedekleme, işletim sistemi yaması gibi işlerle uğraşmazsın. Örnek: AWS Elastic Beanstalk, Azure App Service, Google App Engine; yönetilen veritabanları olarak Amazon RDS, Azure SQL Database, Cloud SQL.
- **SaaS (Software as a Service, hizmet olarak yazılım):** Hazır bir yazılımı tarayıcıdan ya da uygulamadan kullanırsın. Altyapının hiçbir kısmını yönetmezsin; yalnız kullanıcıları, ayarları ve verini yönetirsin. Örnek: Microsoft 365, Google Workspace, çevrim içi muhasebe ve İK yazılımları.

Bu üç modelin arasında da hizmetler vardır. **Sunucusuz (serverless)** hizmetlerde (AWS Lambda, Azure Functions, Google Cloud Run functions) yalnız kodu yüklersin; kod bir olay olduğunda çalışır, çalıştığı süre kadar ödersin. Sunucu hâlâ vardır ama onu hiç görmezsin.

Model yukarı çıktıkça (IaaS → PaaS → SaaS) yönetim yükün azalır ama kontrolün de azalır. Eski bir uygulama belirli bir işletim sistemi sürümünü istiyorsa IaaS gerekebilir. Yeni bir web uygulaması için PaaS daha az iş çıkarır. E-posta için kendi sunucunu kurmak yerine SaaS kullanmak çoğu KOBİ için daha mantıklıdır.

Dağıtım modelleri: genel, özel, hibrit

Bulutun kimin için ve nerede çalıştığına göre de ayırım yapılır.

- **Genel bulut (public cloud):** Altyapı sağlayıcıya aittir ve çok sayıda müşteriye hizmet verir. AWS, Microsoft Azure ve Google Cloud genel bulut sağlayıcılarıdır. Müşteriler birbirinden mantıksal olarak ayrılır.
- **Özel bulut (private cloud):** Altyapı yalnız bir kurum için çalışır. Kurumun kendi veri merkezinde ya da bir hizmet sağlayıcıda olabilir. Self servis, otomasyon ve ölçülen hizmet gibi bulut özelliklerini taşır; yalnız sanallaştırılmış bir sistem odası özel bulut sayılmaz.
- **Hibrit bulut (hybrid cloud):** Şirket içi ya da özel bulut ile genel bulutun birlikte, birbirine bağlı kullanılmasıdır. Örneğin hasta kayıtları hastanenin kendi veri merkezinde tutulur, randevu web sitesi genel bulutta çalışır ve ikisi güvenli bir bağlantıyla konuşur.
- **Topluluk bulutu (community cloud):** Benzer ihtiyaçları olan kurumların (örneğin aynı sektördeki kurumlar) ortak kullandığı bulut. Daha az görülür.

Birden fazla genel bulut sağlayıcısını birlikte kullanmaya **çoklu bulut (multi-cloud)** denir. Örneğin e-posta ve ofis için Microsoft 365, veri analizi için Google Cloud, web uygulaması için AWS kullanan bir şirket çoklu bulut kullanıyordur. Çoklu bulut tek sağlayıcıya bağımlılığı azaltabilir ama yönetimi, güvenliği ve maliyet takibini zorlaştırır.

Bulutun getirdikleri

- **Hız:** Yeni bir ortam dakikalar içinde hazırlanır.
- **Esneklik:** Talep artınca kaynak eklenir, azalınca kaldırılır.
- **Peşin yatırımın azalması:** Donanım satın alınmaz.
- **Coğrafi yaygınlık:** Farklı ülkelerdeki veri merkezlerinde çalışmak mümkündür.
- **Yönetilen hizmetler:** Veritabanı, yedekleme, izleme gibi işler hazır hizmet olarak alınır.
- **Dayanıklılık seçenekleri:** Aynı bölgedeki farklı veri merkezlerine dağıtılarak tek bir binadaki arızadan etkilenmemek mümkündür.

Bulutun sınırları

Bulut her sorunu kendiliğinden çözmez.

- **Maliyet kontrolü:** Kolay açılan kaynak kolay unutulur. Unutulan kaynaklar her ay fatura üretir.
- **İnternet bağımlılığı:** Ofisteki kullanıcılar buluttaki uygulamaya internetle ulaşır. Bağlantı yavaş ya da kesikse iş durur.
- **Beceri ihtiyacı:** Bulut farklı bir yönetim biçimi ister; ekibin öğrenmesi gerekir.
- **Sağlayıcıya bağımlılık (vendor lock-in):** Bir sağlayıcıya özgü hizmetlere çok bağlanırsan başka yere taşınmak zorlaşır.
- **Mevzuat:** Kişisel verinin nerede tutulduğu ve yurt dışına aktarılıp aktarılmadığı hukuki sonuç doğurur (Modül 10).

Bölge ve erişilebilirlik alanı

Büyük sağlayıcılar dünyanın farklı yerlerinde **bölgeler (region)** işletir. Her bölge, birbirine yakın ama ayrı binalarda, ayrı güç ve soğutma altyapısıyla çalışan veri merkezi gruplarından oluşur. Bunlara AWS ve Azure'da **erişilebilirlik alanı (Availability Zone)**, Google Cloud'da **bölge içi alan (zone)** denir.

Bir uygulamayı aynı bölgedeki iki ya da üç erişilebilirlik alanına dağıtırsan, bir veri merkezindeki elektrik ya da ağ arızası uygulamayı durdurmaz. Bölge seçerken üç şeye bakılır:

1. Kullanıcılara yakınlık. Uzaklık arttıkça gecikme artar.
2. Mevzuat. Verinin hangi ülkede tutulduğu.
3. Hizmetin o bölgede sunulup sunulmadığı ve fiyatı. Her hizmet her bölgede yoktur; fiyatlar bölgeye göre değişir.

Sağlayıcıların bölge listesi zamanla değişir. Güncel listeyi her zaman sağlayıcının kendi belgelerinden kontrol et.

Örnek vaka

Antalya'da sezonluk çalışan bir otel grubu online rezervasyon sistemini kendi sistem odasındaki iki sunucuda çalıştırıyor. Yaz aylarında, özellikle kampanya duyurularından sonra site yavaşlıyor ve zaman zaman yanıt vermiyor. Kış aylarında ise sunucular neredeyse boşta.

BT sorumlusu üç seçeneği değerlendiriyor:

1. Daha güçlü iki sunucu almak. Yaz yoğunluğunu karşılar ama kışın kapasitenin büyük kısmı boşta kalır.
2. Rezervasyon sitesini genel buluta taşıyıp yoğunluğa göre otomatik ölçeklemek (IaaS ya da PaaS). Yazın sunucu sayısı artar, kışın azalır.
3. Hazır bir otel rezervasyon yazılımına (SaaS) geçmek. Altyapı yönetimi tamamen biter ama özel ihtiyaçlar için esneklik azalır.

Değerlendirmede hızlı esneklik ve ölçülen hizmet özellikleri öne çıkar. Ayrıca misafir verisi kişisel veri olduğu için seçilecek bölgenin ve sözleşmenin KVKK açısından incelenmesi gerekir. Sonuçta grup, web sitesini PaaS üzerinde çalıştırmaya, muhasebe sistemini ise bir süre daha şirket içinde tutmaya karar verir. Bu bir hibrit bulut yapısıdır.

Uygulama

Kendi iş yerinde ya da tanıdığın bir şirkette kullanılan beş yazılımı listele. Her biri için şu soruları yanıtla:

1. Yazılım nerede çalışıyor: şirket içinde mi, bulutta mı?
2. Buluttaysa hangi modelde: IaaS, PaaS, SaaS?
3. Güncellemeleri ve yedeklemeyi kim yapıyor: şirket mi, sağlayıcı mı?
4. Bu yazılım bir gün çalışmazsa hangi iş durur?

Örneğin "e-posta: Microsoft 365, SaaS, güncellemeleri Microsoft yapıyor, kullanıcı ve ayar yönetimi bizde" gibi. Bu alıştırmayı, hizmet modellerinin sorumluluğu nasıl paylaştığını somut olarak görmeni sağlar ve Modül 6'daki paylaşılan sorumluluk modeline hazırlık olur.

Sık yapılan hatalar

- "Bulutta veri kaybolmaz" diye düşünmek. Sağlayıcı altyapıyı korur; yanlış silme, yanlış yapılandırma ve hesap ele geçirilmesine karşı yedek ve önlem senin işindir.
- Sanallaştırılmış her ortamı bulut sanmak. Self servis, esneklik ve ölçülen hizmet yoksa bu klasik sanallaştırma değildir.

- Bütün uygulamaları aynı modelle taşımaya çalışmak. Her uygulama için IaaS, PaaS ya da SaaS ayrı ayrı değerlendirilir.
- Bölge seçimini rastgele yapmak. Gecikme, fiyat ve mevzuat bölgeye göre değişir.
- Tek erişilebilirlik alanına kurulu sistemi "yüksek erişilebilir" sanmak.

Kendini dene

1. NIST'in tanımladığı beş temel bulut özelliğini say.
2. Sanal makineyi sağlayıcıdan alıp işletim sistemini kendin yönettiğin model hangisidir?
3. Microsoft 365 hangi hizmet modeline örnektir?
4. Şirket içi veri merkezi ile genel bulutun birbirine bağlı kullanılmasına ne ad verilir?
5. Bir uygulamayı aynı bölgedeki birden fazla erişilebilirlik alanına dağıtmanın amacı nedir?

Büyük sağlayıcılar ve temel hizmet kategorileri

Genel bulut pazarında en yaygın kullanılan üç sağlayıcı Amazon Web Services (AWS), Microsoft Azure ve Google Cloud'dur. Üçü de yüzlerce hizmet sunar. İlk bakışta bu liste ürktücü görünür ama hizmetlerin büyük çoğunluğu birkaç ana kategoriye girer ve her kategoride üç sağlayıcının karşılığı vardır. Bir sağlayıcıda kavramı öğrendiğinde diğerine geçmek, büyük ölçüde isimleri eşleştirmekten ibarettir.

Bu modülde hizmet adlarını doğru yazıyoruz ama fiyat, kota ve sürüm gibi değişen bilgileri vermiyoruz. Sağlayıcılar hizmetlerini sık günceller, zaman zaman adlarını değiştirir. Bir karar vermeden önce her zaman sağlayıcının güncel belgelerine bak.

Üç sağlayıcıyı kısaca tanı

- **AWS:** Amazon'un bulut birimi. Genel bulut hizmetlerini en erken yaygınlaştıran sağlayıcıdır. Hizmet yelpazesi çok geniştir; birçok bulut kavramı ilk olarak AWS'deki adlarıyla bilinir.
- **Microsoft Azure:** Microsoft'un bulut platformu. Windows Server, SQL Server, Active Directory ve Microsoft 365 kullanan kurumlar için doğal bir devam yoludur. Kimlik yönetimi Microsoft Entra ID (eski adıyla Azure Active Directory) üzerinden yürür.
- **Google Cloud:** Google'ın bulut platformu. Veri analizi (BigQuery), konteyner yönetimi (Kubernetes ilk olarak Google'da geliştirildi) ve makine öğrenmesi hizmetleriyle öne çıkar.

Türkiye'de de bu üç sağlayıcıyı kullanan çok sayıda şirket vardır. Bunların yanında yerel bulut ve veri merkezi hizmeti veren firmalar da bulunur. Seçim, ihtiyaçlara, ekibin bilgisine, mevcut lisanslara ve mevzuata göre yapılır.

Hesap ve kaynak düzeni

Her sağlayıcıda kaynaklar bir hiyerarşi içinde durur. Faturalama, yetki ve politika bu hiyerarşiye göre uygulanır.

- **AWS:** Temel birim **hesaptır (AWS account)**. Birden fazla hesap **AWS Organizations** ile tek çatı altında toplanır. Üretim, test ve geliştirme ortamları için ayrı hesaplar açmak yaygın ve önerilen bir yöntemdir.
- **Azure:** En üstte kuruluşun Microsoft Entra **kiracısı (tenant)** bulunur. Altında **yönetim grupları (management groups)**, **abonelikler (subscriptions)** ve aboneliklerin içinde **kaynak grupları (resource groups)** yer alır. Her kaynak bir kaynak grubuna aittir.
- **Google Cloud:** En üstte **kuruluş (organization)**, altında **klasörler (folders)** ve **projeler (projects)** bulunur. Kaynaklar projelerin içinde oluşturulur; faturalama hesabı projelere bağlanır.

Bu düzeni baştan doğru kurmak sonradan çok iş kurtarır. Örneğin "test" ile "üretim" aynı yerde karışık durursa, bir test kaynağını silerken üretimdeki bir kaynağı silme riski doğar.

Hizmet kategorileri ve karşılıkları

Aşağıdaki listede her kategori için üç sağlayıcının temel hizmetleri AWS, Azure, Google Cloud sırasıyla verilmiştir.

- **Sanal sunucu (IaaS işlem gücü):** Amazon EC2; Azure Virtual Machines; Compute Engine.
- **Otomatik ölçeklenen sunucu grupları:** Amazon EC2 Auto Scaling; Azure Virtual Machine Scale Sets; Compute Engine yönetilen örnek grupları (managed instance groups).

- **Sunucusuz işlev:** AWS Lambda; Azure Functions; Cloud Run functions (eski adıyla Cloud Functions).
- **Nesne depolama:** Amazon S3; Azure Blob Storage; Cloud Storage.
- **Blok depolama (sanal makine diski):** Amazon EBS; Azure Managed Disks; Persistent Disk.
- **Dosya paylaşımı:** Amazon EFS ve Amazon FSx; Azure Files; Filestore.
- **İlişkisel veritabanı:** Amazon RDS ve Amazon Aurora; Azure SQL Database ve Azure Database for PostgreSQL / MySQL; Cloud SQL ve Spanner.
- **NoSQL veritabanı:** Amazon DynamoDB; Azure Cosmos DB; Firestore ve Bigtable.
- **Veri ambarı ve analiz:** Amazon Redshift; Azure Synapse Analytics; BigQuery.
- **Sanal ağ:** Amazon VPC; Azure Virtual Network (VNet); VPC.
- **DNS:** Amazon Route 53; Azure DNS; Cloud DNS.
- **Yük dengeleme:** Elastic Load Balancing; Azure Load Balancer ve Application Gateway; Cloud Load Balancing.
- **İçerik dağıtım ağı (CDN):** Amazon CloudFront; Azure Front Door; Cloud CDN.
- **Kimlik ve erişim:** AWS IAM; Microsoft Entra ID ve Azure RBAC; Cloud IAM.
- **Anahtar ve gizli bilgi yönetimi:** AWS KMS ve AWS Secrets Manager; Azure Key Vault; Cloud KMS ve Secret Manager.
- **İzleme ve log:** Amazon CloudWatch ve AWS CloudTrail; Azure Monitor; Cloud Monitoring ve Cloud Logging.
- **Konteyner kayıt deposu:** Amazon ECR; Azure Container Registry; Artifact Registry.
- **Kubernetes:** Amazon EKS; Azure Kubernetes Service (AKS); Google Kubernetes Engine (GKE).
- **Sunucusuz konteyner:** AWS Fargate (ECS ya da EKS ile); Azure Container Apps ve Azure Container Instances; Cloud Run.
- **Mesajlaşma ve kuyruk:** Amazon SQS ve Amazon SNS; Azure Service Bus; Pub/Sub.
- **Makine öğrenmesi platformu:** Amazon SageMaker; Azure Machine Learning; Vertex AI.
- **Yedekleme:** AWS Backup; Azure Backup; Backup and DR Service.

Bu listeyi ezberlemen gerekmez; önemli olan bir ihtiyacın hangi kategoriye girdiğini düşünebilmektir. Örneğin "kullanıcıların yüklediği faturaların PDF'lerini saklayacağız" ihtiyacı nesne depolamaya, "her gece bir rapor üretecek küçük bir kod çalışacak" ihtiyacı sunucusuz işleve karşılık gelir.

Depolama sınıfları

Nesne depolamada verinin ne sıklıkla okunacağına göre farklı sınıflar ya da katmanlar vardır. Sık erişilen veri daha pahalı ama hızlı katmanda, nadiren erişilen arşiv verisi daha ucuz ama geri çağırması yavaş ya da ücretli katmanda tutulur.

- AWS'de S3 Standard, S3 Standard-IA (seyrek erişim) ve S3 Glacier sınıfları bulunur.
- Azure Blob Storage'da Hot, Cool, Cold ve Archive erişim katmanları vardır.
- Cloud Storage'da Standard, Nearline, Coldline ve Archive sınıfları vardır.

Yaşam döngüsü kuralları (lifecycle rules) ile eski dosyalar belirli bir süre sonra otomatik olarak daha ucuz katmana taşınabilir ya da silinebilir. Örneğin bir muhasebe bürosu, taranmış belgeleri ilk birkaç ay sık erişilen katmanda, sonra arşiv katmanında tutabilir. Saklama süresi ise yasal yükümlülöklere göre belirlenir.

Hizmetlere erişim yolları

Bulut kaynaklarını yönetmenin dört yolu vardır:

1. **Web konsolu:** AWS Management Console, Azure portal, Google Cloud console. Öğrenmek ve tek seferlik işler için uygundur.
2. **Komut satırı (CLI):** AWS CLI (`aws`), Azure CLI (`az`), Google Cloud CLI (`gcloud`). Tekrarlanan işler ve betikler için uygundur.
3. **Yazılım geliştirme kiti (SDK):** Python, Java, JavaScript gibi dillerden bulut hizmetlerini çağırmak için.
4. **Kod olarak altyapı (Infrastructure as Code, IaC):** Altyapıyı bir metin dosyasında tanımlayıp otomatik oluşturmak. AWS CloudFormation, Azure Resource Manager şablonları ve Bicep, Google Cloud'da Terraform yaygın olarak kullanılır. Terraform üç sağlayıcıda da çalışan, HashiCorp'un geliştirdiği bir araçtır.

Aşağıdaki komutlar yalnız bilgi listeler, hiçbir şeyi değiştirmez. Her biri, ilgili CLI kurulu ve oturum açılmış olduğunda çalışır.

```
# AWS: erişebildiğin S3 kovalarını listeler
aws s3 ls

# Azure: aboneliğindeki kaynak gruplarını tablo hâlinde listeler
az group list --output table

# Google Cloud: erişebildiğin projeleri listeler
gcloud projects list
```

Konsolda elle yapılan değişiklikler kayıt dışı kalır ve tekrarlanamaz; ekipler büyüdükçe IaC'ye geçer (Modül 9).

Sağlayıcı seçerken sorulacak sorular

1. Kurumun hâlihazırda hangi yazılımları ve lisansları kullanıyor? Microsoft ağırlıklı bir yapıda Azure ile entegrasyon kolay olabilir.
2. Ekibin hangi sağlayıcıda deneyimi var?
3. İhtiyaç duyduğun hizmetler istediğin bölgede sunuluyor mu?
4. Verinin tutulacağı ülke ve yurt dışına aktarım konusunda hukuk biriminin görüşü ne?
5. Destek planları, sözleşme koşulları ve hizmet düzeyi taahhütleri (SLA) ihtiyacını karşılıyor mu?
6. Gelecekte başka sağlayıcıya geçmek gerekirse ne kadar zor olur?

Örnek vaka

Ankara'da 60 kişilik bir yazılım danışmanlık firması müşteri projeleri için test ortamları kuruyor. Şimdiye kadar her proje için ofisteki sanal sunucu üzerinde yeni makineler açılıyor; kapasite dolduğu için yeni projeler bekliyor.

Firma, ekipteki deneyime ve müşterilerinin çoğunun Microsoft ürünleri kullanmasına bakarak Azure'u seçiyor. İlk iş olarak şu düzeni kuruyorlar:

1. Kuruluşun Microsoft Entra kiracısı altında "Üretim" ve "Test" adlı iki yönetim grubu.

2. Her müşteri projesi için ayrı bir abonelik ya da en azından ayrı bir kaynak grubu.
3. Her kaynak grubuna "proje", "sahip" ve "ortam" etiketleri (Modül 7).
4. Test kaynak gruplarına bütçe uyarısı.

Bir müşteri projesi bittiğinde ilgili kaynak grubunu bütünüyle kaldırmak yeterli oluyor.

Uygulama

Üç sağlayıcının ücretsiz deneme ya da ücretsiz katman olanaklarından birini seç. Hesap açarken kendi kişisel bilgilerini kullanacaksan şirket verisi yüklediğinden emin ol. Ardından şunları yap:

1. Hesabın ana kullanıcısında çok faktörlü kimlik doğrulamayı aç (Modül 6).
2. Küçük bir tutar için bütçe uyarısı oluştur (Modül 7).
3. Konsolda yukarıdaki kategorilerden beşini bul: sanal sunucu, nesne depolama, ilişkisel veritabanı, sanal ağ ve izleme. Her birinin ana sayfasını aç ve hangi bölgede olduğunu kontrol et.
4. Hesap hiyerarşisini incele: AWS'de hesap ve Organizations, Azure'da abonelik ve kaynak grupları, Google Cloud'da proje.
5. Hiçbir kaynak oluşturmadıysan bu adımda bırak. Kaynak oluşturduysan iş bitince sil.

Sık yapılan hatalar

- Hizmet adlarını karıştırmak. Örneğin Azure'daki "kaynak grubu" ile AWS'deki "güvenlik grubu" (security group) tamamen farklı şeylerdir.
- Her şeyi tek hesapta ya da tek abonelikte toplamak. Üretim ve test ayrılmazsa yetki ve maliyet kontrolü zorlaşır.
- Bir sağlayıcıdaki hizmetin her bölgede olduğunu varsaymak.
- Konsolda elle oluşturulan kaynakların kaydını tutmamak.
- Hizmetleri yalnız adına göre seçmek; yönetim yükünü, maliyet yapısını ve taşınabilirliği düşünmemek.

Kendini dene

1. AWS, Azure ve Google Cloud'daki sanal sunucu hizmetlerinin adları nedir?
2. Azure'da her kaynağın ait olması gereken mantıksal kapsayıcıya ne denir?
3. Google Cloud'da kaynakların oluşturulduğu temel birim nedir?
4. Nadiren erişilen arşiv verisi için nesne depolamada neden farklı bir sınıf kullanılır?
5. Altyapıyı metin dosyalarıyla tanımlayıp otomatik oluşturma yaklaşımına ne denir?

Ağ temelleri

Buluttaki her kaynak bir ağın parçasıdır. Sanal sunucu bir alt ağda durur, bir IP adresi alır, bir güvenlik kuralına göre trafik kabul eder ve kullanıcılar ona çoğunlukla bir alan adı üzerinden ulaşır. Bulutla ilgili sorunların önemli bir kısmı aslında ağ sorunudur: "sunucuya bağlanamıyorum" şikâyeti çoğu zaman yanlış bir güvenlik kuralından, yanlış bir DNS kaydından ya da çakışan bir adres aralığından çıkar.

IP adresi

IP adresi, ağdaki bir cihazın adresidir. Bugün iki sürüm kullanılır:

- **IPv4:** 32 bitlik adres. Nokta ile ayrılmış dört sayı biçiminde yazılır: 192.168.1.10. Her sayı 0 ile 255 arasındadır.
- **IPv6:** 128 bitlik adres. IPv4 adreslerinin tükenmesi nedeniyle geliştirildi. Onaltılık sayılarla ve iki nokta üst üste ile yazılır: 2001:db8::1.

IPv4 adresleri ikiye ayrılır:

- **Özel (private) adresler:** Yalnız iç ağda kullanılır, internette yönlendirilmez. Üç aralık bu amaçla ayrılmıştır: 10.0.0.0–10.255.255.255, 172.16.0.0–172.31.255.255 ve 192.168.0.0–192.168.255.255. Ofis ağları ve bulut sanal ağları bu aralıkları kullanır.
- **Genel (public) adresler:** İnternette benzersizdir ve yönlendirilir. Bir web sitesinin internetten erişilen adresi genel adrestir.

Özel adresli cihazlar internete **NAT (ağ adresi çevirisi)** aracılığıyla çıkar. NAT, iç ağdaki çok sayıda cihazın tek bir genel adresi paylaşmasını sağlar. Evdeki modem de bunu yapar.

Alt ağ ve CIDR gösterimi

Bir ağ, daha küçük parçalara, yani **alt ağlara (subnet)** bölünür. Alt ağın büyüklüğü **CIDR gösterimiyle** yazılır: 10.0.1.0/24 gibi. Eğik çizgiden sonraki sayı, adresin kaç bitinin ağ kısmına ayrıldığını gösterir. Geriye kalan bitler cihaz adresleri içindir.

Hesap basittir: toplam adres sayısı 2 üzeri (32 eksi CIDR sayısı) kadardır.

- /16: $2^{16} = 65.536$ adres. Örneğin 10.20.0.0/16. Bir sanal ağın tamamı için sık kullanılır.
- /24: $2^8 = 256$ adres. Örneğin 10.20.1.0/24. Tipik bir alt ağ.
- /27: $2^5 = 32$ adres.
- /28: $2^4 = 16$ adres. Küçük, özel amaçlı alt ağlar için.

Her alt ağda bazı adresler kullanılamaz. Geleneksel ağlarda ilk adres ağın kendisini, son adres yayın (broadcast) adresini gösterir. Bulut sağlayıcıları ise her alt ağda birkaç adresi kendi hizmetleri için ayırır; AWS ve Azure'da bu sayı 5'tir. Yani bulutta bir /24 alt ağda 251 adres kullanılabilir.

Bulutta sanal ağ

Her sağlayıcı müşterilerine yalıtılmış bir sanal ağ verir: AWS'de ve Google Cloud'da VPC (Virtual Private Cloud), Azure'da VNet (Virtual Network). Sanal ağ, kendi adres aralığının ve alt ağlarınla buluttaki "ofis ağın" gibidir.

Önemli bir ayrıntı: AWS'de VPC bir bölgeye, alt ağlar ise bir erişilebilirlik alanına bağlıdır. Azure'da VNet bir bölgeye bağlıdır, alt ağları bölgedeki alanlara yayılabilir. Google Cloud'da VPC küreseldir, alt ağları bölgeseldir.

Bir sanal ağda alt ağlar genellikle iki türde düşünülür:

- **Genel alt ağ (public subnet):** İnternette gelen trafiği kabul edebilen kaynaklar burada durur; örneğin yük dengeleyici. AWS'de bir alt ağ "genel" yapan, yönlendirme tablosunda internet ağ geçidine (internet gateway) giden bir rota olmasıdır.
- **Özel alt ağ (private subnet):** İnternette doğrudan erişilemeyen kaynaklar; uygulama sunucuları ve veritabanları. Bu kaynaklar güncelleme indirmek gibi işler için internete NAT ağ geçidi üzerinden çıkabilir ama dışarıdan onlara bağlantı başlatılamaz.

Yönlendirme tablosu (route table), bir alt ağdaki trafiğin nereye gideceğini belirler: aynı ağ içine mi, internete mi, VPN üzerinden şirket ağına mı.

Güvenlik kuralları ve portlar

Bir sunucuda çalışan her hizmet bir **port** numarası üzerinden dinler. Sık karşılaşılabilecek portlar:

- 22: SSH (Linux sunucuya uzaktan bağlantı)
- 3389: RDP (Windows sunucuya uzak masaüstü)
- 80: HTTP
- 443: HTTPS
- 53: DNS
- 3306: MySQL
- 5432: PostgreSQL
- 1433: Microsoft SQL Server

Bulutta trafiğe izin veren kurallar şöyle adlandırılır:

- **AWS:** Güvenlik grubu (security group), kaynak düzeyinde çalışır ve durum bilgilidir (stateful): içeri izin verilen bir bağlantının yanıtı otomatik olarak dışarı çıkar. Ayrıca alt ağ düzeyinde, durum bilgisi tutmayan ağ erişim denetim listeleri (network ACL) vardır.
- **Azure:** Ağ güvenlik grubu (Network Security Group, NSG). Alt ağa ya da ağ arayüzüne bağlanır.
- **Google Cloud:** VPC güvenlik duvarı kuralları (firewall rules).

Temel kural: yalnız gerekeni aç. Web sunucusu için 443 portu herkese açık olabilir. Ama SSH (22) ve RDP (3389) portları asla 0.0.0.0/0 (bütün internet) için açılmamalıdır. Bunun yerine yalnız şirket ağının adresine izin verilir, VPN kullanılır ya da sağlayıcının güvenli bağlantı hizmetleri (AWS Systems Manager Session Manager, Azure Bastion, Google Cloud'da IAP ile TCP yönlendirme) tercih edilir. Veritabanı portu yalnız uygulama sunucularına açılır.

DNS: adlardan adreslere

İnsanlar adları, bilgisayarlar adresleri kullanır. **DNS (Domain Name System),** alan adlarını IP adreslerine çeviren sistemdir. Tarayıcına bir alan adı yazdığında bilgisayarın önce bir DNS çözümleyicisine sorar; çözümleyici gerekirse kök sunuculardan başlayarak alan adının yetkili sunucusuna kadar sorgu yapar ve cevabı getirir.

Sık kullanılan kayıt türleri:

- **A:** Alan adını bir IPv4 adresine bağlar.

- **AAAA:** Alan adını bir IPv6 adresine bağlar.
- **CNAME:** Bir adı başka bir ada yönlendirir (takma ad).
- **MX:** Alan adına gelen e-postanın hangi sunucuya gideceğini gösterir.
- **TXT:** Metin bilgisi. Alan adı sahipliğini doğrulamak ve e-posta güvenliği (SPF, DKIM, DMARC) için kullanılır.
- **NS:** Alan adının hangi DNS sunucularında yönetildiğini gösterir.

Her kaydın bir **TTL (time to live)** değeri vardır. TTL, cevabın saniye cinsinden ne kadar süre önbellekte tutulacağını söyler. Bir taşıma öncesinde TTL'i birkaç gün önceden düşürmek, adres değiştiğinde yeni adrese geçişin hızlı olmasını sağlar.

Aşağıdaki komutlar Linux ya da macOS'ta zararsız ağ kontrolleri yapar. Örnek alan adı olarak bu amaçla ayrılmış example.com kullanılmıştır.

```
# Hedefe 4 paket gönderip ulaşılabilirliği ve gecikmeyi ölçer
ping -c 4 example.com

# Alan adının A kaydını sorgular
dig example.com A +short

# Aynı sorgu, Windows'ta da çalışan araçla
nslookup example.com

# Web sunucusunun yalnız yanıt başlıklarını getirir
curl -I https://example.com
```

Yük dengeleyici

Yük dengeleyici (load balancer), gelen istekleri birden fazla sunucuya dağıtan hizmettir. Üç işi vardır:

1. Yükü paylaştırır; tek bir sunucu darboğaz olmaz.
2. **Sağlık kontrolü (health check)** yapar: düzenli aralıklarla sunuculara istek gönderir, yanıt vermeyen sunucuya trafik göndermeyi bırakır.
3. Kullanıcıya tek bir giriş noktası sunar; arkadaki sunucular eklenip çıkarılırken kullanıcı bunu fark etmez.

İki ana tür vardır:

- **Katman 4 (taşıma katmanı):** IP adresi ve porta göre dağıtım yapar; içeriğe bakmaz. Hızlıdır. AWS Network Load Balancer, Azure Load Balancer bu türdendir.
- **Katman 7 (uygulama katmanı):** HTTP isteğinin içeriğine bakar; örneğin /api ile başlayan istekleri bir gruba, diğerlerini başka gruba yönlendirebilir. HTTPS sertifikasını da üzerinde taşıyabilir. AWS Application Load Balancer ve Azure Application Gateway bu türdendir. Google Cloud Load Balancing her iki türü de sunar.

Yük dengeleyici genellikle otomatik ölçekleme ile birlikte kullanılır: yük artınca yeni sunucular eklenir, yük dengeleyici onları sağlık kontrolünden geçtikten sonra trafiğe alır.

Şirket ağını buluta bağlamak

Hibrit yapılarda ofis ağı ile bulut ağı birbirine bağlanır. İki yol vardır:

- **Siteden siteye VPN:** İnternet üzerinden şifreli tünel. Hızlı kurulum, maliyeti düşüktür, performansı internet bağlantısına bağlıdır.
- **Özel bağlantı:** Bir veri merkezi ya da operatör üzerinden buluta doğrudan, internette geçmeyen hat. AWS Direct Connect, Azure ExpressRoute ve Google Cloud Interconnect bu hizmetlerdir. Daha öngörülebilir performans verir ama kurulumu daha uzun ve maliyetlidir.

Bağlantı kurulacaksa ofis ağının ve bulut ağının adres aralıkları **çakışmamalıdır**. Ofiste 192.168.1.0/24 kullanılıyorsa bulutta aynı aralık seçilmez.

Uygulama

İstanbul'da bir e-ticaret mağazası için AWS'de bir VPC planla. Ofis ağı 192.168.10.0/24 adresini kullanıyor. İhtiyaçlar: internete açık bir yük dengeleyici, iki uygulama sunucusu, bir veritabanı ve iki erişilebilirlik alanı.

1. VPC için ofisle çakışmayan bir aralık seç: 10.20.0.0/16.
2. Her erişilebilirlik alanında bir genel alt ağ aç: 10.20.1.0/24 ve 10.20.2.0/24. Yük dengeleyici burada durur.
3. Her alanda bir uygulama alt ağı aç: 10.20.11.0/24 ve 10.20.12.0/24. Uygulama sunucuları burada, özel alt ağda durur.
4. Her alanda bir veritabanı alt ağı aç: 10.20.21.0/24 ve 10.20.22.0/24.
5. Güvenlik kurallarını yaz: yük dengeleyiciye 443 portundan herkes; uygulama sunucularına yalnız yük dengeleyiciden; veritabanına yalnız uygulama sunucularından 5432 portu.
6. Yönetim erişimi için 22 portunu internete açma; Session Manager ya da VPN kullan.
7. Alan adının A kaydını (ya da sağlayıcının önerdiği takma ad kaydını) yük dengeleyiciye yönlendir.

Bu planı kâğıda çiz: kutular alt ağlar, oklar izin verilen trafik.

Sık yapılan hatalar

- SSH ya da RDP portunu "geçici olarak" bütün internete açıp kapatmayı unutmak. Açık uzak erişim portları otomatik tarayıcılar tarafından kısa sürede bulunur.
- Bulut ağının adres aralığını ofis ağıyla ya da başka bir bulut ağıyla çakışacak şekilde seçmek. Sonradan düzeltmek çoğu zaman ağı yeniden kurmayı gerektirir.
- Alt ağları çok küçük seçmek. Otomatik ölçekleme ve konteynerler beklenenden çok adres tüketebilir.
- Taşıma öncesinde DNS TTL değerini düşürmeyi unutmak; eski adrese saatlerce trafik gitmesi.
- Veritabanını genel alt ağa koymak.
- Yük dengeleyicinin sağlık kontrolü yolunu yanlış ayarlamak; sağlıklı sunucuların trafikten çıkarılması.

Kendini dene

1. 10.20.0.0/16 aralığında toplam kaç IPv4 adresi vardır?
2. Özel alt ağdaki bir sunucu güncelleme indirmek için internete nasıl çıkar?
3. Bir alan adını IPv4 adresine bağlayan DNS kaydı türü hangisidir?
4. Katman 4 ve katman 7 yük dengeleyici arasındaki temel fark nedir?
5. Ofis ağı buluta bağlanacaksa adres aralıkları konusunda neye dikkat edilir?

Linux ve Windows Server ile sistem yönetimine giriş

Bulutta sanal sunucu açtığında karşına iki işletim sistemi ailesinden biri çıkar: Linux ya da Windows Server. Web sunucularının, konteynerlerin ve bulut araçlarının büyük kısmı Linux üzerinde çalışır. Windows Server ise Active Directory, dosya paylaşımı, Microsoft SQL Server ve Windows'a özgü kurumsal uygulamalar için yaygındır. Bir sistem yöneticisinin ikisinde de temel durum kontrolünü yapabilmesi beklenir.

Bu modüldeki komutlar bilgi okur ya da kendi deneme makinende güvenle çalıştırabileceğin türdendir. Üretim sunucusunda komut çalıştırmadan önce ne yaptığını bildiğinden emin ol ve yetkin olmayan sistemlerde komut çalıştırma.

Linux'a kısa bakış

Linux, açık kaynaklı bir işletim sistemi çekirdeğidir. Bu çekirdeğin etrafında farklı **dağıtımlar** oluşmuştur. Sunucu dünyasında iki büyük aile öne çıkar:

- **Debian ailesi:** Debian ve Ubuntu. Paket yöneticisi `apt`.
- **Red Hat ailesi:** Red Hat Enterprise Linux (RHEL) ve onunla uyumlu dağıtımlar (Rocky Linux, AlmaLinux gibi), Fedora. Paket yöneticisi `dnf`.

Bulut sağlayıcıları da kendi Linux dağıtımlarını sunabilir; örneğin AWS'nin Amazon Linux'u Red Hat ailesine yakındır.

Linux sunucular genellikle grafik arayüz olmadan, **komut satırından (shell)** yönetilir. En yaygın kabuk `bash` 'tir. Sunucuya uzaktan **SSH** ile bağlanılır. Bulutta SSH bağlantısı için parola yerine **anahtar çifti** kullanılır: özel anahtar sende kalır, genel anahtar sunucuya yüklenir.

```
# Kendi bilgisayarında yeni bir SSH anahtar çifti üretir
ssh-keygen -t ed25519 -C "deneme-anahtari"

# Sunucuya bağlanır (kullanıcı adı ve adres örnektir)
ssh yonetici@203.0.113.10
```

Özel anahtar dosyasını kimseyle paylaşma, e-postayla gönderme, ortak klasöre koyma. Kaybolursa ya da sızarsa sunucudan ilgili genel anahtarı kaldırmak gerekir.

Dosya sistemi düzeni

Linux'ta her şey kök dizin `/` altından başlayan tek bir ağaçta durur. Windows'taki gibi C:, D: sürücü harfleri yoktur; ek diskler bir klasöre **bağlanır (mount)**.

- `/etc` : Yapılandırma dosyaları.
- `/var/log` : Log dosyaları.
- `/home` : Kullanıcıların kişisel klasörleri.
- `/tmp` : Geçici dosyalar.
- `/usr` : Programlar ve kütüphaneler.

- `/root` : Yönetici kullanıcının (root) kişisel klasörü.

Temel durum kontrolü

Bir sunucuya ilk bağlandığında ya da bir şikâyet geldiğinde önce genel duruma bakarsın: disk dolu mu, bellek yetiyor mu, sunucu ne zamandır açık, hangi işletim sistemi kurulu?

```
# Hangi klasördesin ve içinde ne var
pwd
ls -l
# Disklerin doluluk oranı, okunabilir birimlerle
df -h
# Bellek kullanımı
free -h
# Sunucu ne zamandır açık, yük ortalaması ne
uptime
# İşletim sistemi adı ve sürümü
cat /etc/os-release
```

`df -h` çıktısında "Use%" sütunu disklerin doluluk yüzdesini gösterir. Bir disk dolduğunda uygulamalar yazamaz, loglar kesilir, veritabanı durabilir. "Disk doldu" sunucu sorunlarının en sık nedenlerinden biridir.

`uptime` çıktısındaki yük ortalaması (load average) son 1, 5 ve 15 dakikadaki ortalama iş kuyruğunu gösterir. Çekirdek sayısına göre yorumlanır: 4 çekirdekli bir sunucuda sürekli 4'ün çok üstünde bir değer, işlerin sıra beklediğini gösterir. Anlık işlem listesini görmek için `top` komutu kullanılır; çıkmak için `q` tuşuna basılır.

Hizmetler ve loglar

Çoğu modern Linux dağıtımında hizmetler **systemd** ile yönetilir. Bir web sunucusu, veritabanı ya da zamanlanmış iş birer hizmettir.

```
# nginx hizmetinin çalışıp çalışmadığını gösterir
systemctl status nginx
# Bu hizmetin son bir saatlik loglarını gösterir
journalctl -u nginx --since "1 hour ago"
# Log klasörünün toplam boyutunu gösterir
sudo du -sh /var/log
# Genel sistem logunun son 20 satırı (Debian ve Ubuntu'da)
tail -n 20 /var/log/syslog
```

Red Hat ailesinde genel sistem logu `/var/log/messages` dosyasındadır. Bir hizmet çalışmıyorsa sırası şöyledir: önce `systemctl status` ile durumunu ve son hata satırlarını gör, sonra `journalctl` ile ayrıntılı loga bak, sonra disk ve bellek durumunu kontrol et.

Kullanıcılar ve izinler

Linux'ta her dosyanın bir sahibi, bir grubu ve üç izin kümesi vardır: sahip, grup ve diğerleri. Her kümede okuma (r), yazma (w) ve çalıştırma (x) izni bulunur. `ls -l` çıktısında şöyle görünür:

```
-rw-r----- 1 ayse muhasebe 2048 Oct 12 09:15 rapor.txt
```

Bu satırın anlamı: dosyanın sahibi "ayse" okuyup yazabilir, "muhasebe" grubundakiler yalnız okuyabilir, diğer herkes hiçbir şey yapamaz. İzinler sayıyla da yazılır: okuma 4, yazma 2, çalıştırma 1. Yukarıdaki izin 640'tır ve `chmod 640 rapor.txt` komutuyla verilir.

Yönetici yetkisi gerektiren işler için doğrudan root kullanıcısıyla oturum açılmaz; normal kullanıcıyla bağlanılıp gerektiğinde komutun başına `sudo` eklenir. Böylece kimin hangi yönetici komutunu çalıştırdığı loglarda görünür.

Paket güncelleme de yönetici işidir. Ubuntu'da `sudo apt update` paket listesini yeniler, `sudo apt upgrade` kurulu paketleri günceller. Üretim sunucusunda güncelleme önce test ortamında denir, bakım saatinde yapılır.

Windows Server'a kısa bakış

Windows Server, Microsoft'un sunucu işletim sistemidir. Kurulumda iki seçenek sunulur:

- **Masaüstü deneyimi (Desktop Experience):** Tanıdık grafik arayüzle gelir.
- **Server Core:** Grafik arayüz olmadan, daha az bileşenle gelir. Daha az güncelleme ister ve saldırı yüzeyi daha küçüktür; yönetimi PowerShell ve uzaktan araçlarla yapılır.

Windows Server'da bir sunucunun işi **rollerle** belirlenir. Sık kullanılan roller:

- **Active Directory Domain Services (AD DS):** Kullanıcı, bilgisayar ve grupları merkezi olarak yönetir. Etki alanı denetleyicisi bu rolü çalıştırır.
- **DNS Sunucusu ve DHCP Sunucusu:** Ad çözümleme ve cihazlara otomatik IP adresi dağıtma.
- **Dosya ve Depolama Hizmetleri:** Paylaşılan klasörler.
- **Web Sunucusu (IIS):** Web uygulamaları.
- **Hyper-V:** Sanallaştırma.

Grup İlkesi (Group Policy) ile etki alanındaki bilgisayarlara merkezi ayarlar uygulanır: parola kuralları, ekran kilidi süresi, yazıcı eşleme gibi. Rollerin eklenmesi Sunucu Yöneticisi (Server Manager) ya da PowerShell ile yapılır. Birden çok sunucuyu tarayıcıdan yönetmek için Windows Admin Center kullanılabilir. Windows sunuculara uzaktan grafik bağlantı RDP ile, komut satırı bağlantısı PowerShell Remoting ile yapılır.

PowerShell ile temel durum kontrolü

PowerShell, Windows'un komut satırı ve betik dilidir. Komutlar "Fiil-İsim" biçimindedir: Get-Service, Get-Process, Restart-Service gibi. Aşağıdaki komutlar yalnız bilgi okur.

```
# Bilgisayar adı ve işletim sistemi adı
Get-ComputerInfo -Property CsName, OsName
# Disklerin boyutu ve boş alanı
Get-Volume
# Çalışan hizmetler
Get-Service | Where-Object Status -eq 'Running'
# Sistem olay günlüğünün son 10 kaydı
Get-WinEvent -LogName System -MaxEvents 10
# Bir sunucunun 443 portuna erişilebiliyor mu
Test-NetConnection -ComputerName example.com -Port 443
```

`Test-NetConnection` ağ sorunlarında çok işe yarar: çıktısındaki "TcpTestSucceeded : True" satırı porta ulaşılabilirdiğini, "False" ise arada bir güvenlik kuralının ya da kapalı bir hizmetin olduğunu gösterir. Klasik `ipconfig /all` komutu da ağ kartlarının adreslerini, DNS sunucularını ve ağ geçidini gösterir.

Örnek vaka

Kayseri'de bir mobilya üreticisinin e-ticaret sitesi bulutta, bir Ubuntu sanal sunucusunda çalışıyor. Sabah müşteri hizmetleri "site açılmıyor" diye bildiriyor. BT destek çalışanı şu adımları izliyor:

1. Kendi bilgisayarından `curl -I` ile siteye istek gönderiyor; bağlantı kuruluyor ama sunucu hata kodu (502) dönüyor. Demek ki ağ ve DNS çalışıyor, sorun sunucuda.
2. Sunucuya Session Manager ile bağlanıyor. `systemctl status nginx` web sunucusunun çalıştığını gösteriyor.
3. Uygulama hizmetinin durumuna bakıyor; hizmet durmuş. `journalctl` çıktısında "No space left on device" (aygıtta yer kalmadı) hatası görünüyor.
4. `df -h` ile kök diskin %100 dolu olduğunu görüyor. `sudo du -sh /var/log` log klasörünün büyüdüğünü gösteriyor; bir uygulama logu aylardır dönmeden (rotate edilmeden) yazılıyor.
5. Ekip lideriyle konuşarak eski log dosyalarını arşivleyip yer açıyor, uygulama hizmetini yeniden başlatıyor. Site açılıyor.
6. Kalıcı çözüm olarak log döndürme (logrotate) ayarı yapılıyor ve disk doluluğu %80'i geçince uyarı veren bir izleme kuralı ekleniyor.

Asıl kazanım altıncı adımdır: izleme ve uyarı kurulmasaydı sorun birkaç ay sonra yine gelirdi.

Uygulama

Kendi deneme sanal makinende (Modül 1'de kurduğun Linux makinesi ya da bir bulut deneme sunucusu) şunları yap ve çıktıları bir dosyaya not al:

1. `cat /etc/os-release` ile dağıtım adını ve sürümünü öğren.
2. `df -h` ile disk doluluğunu, `free -h` ile belleği, `uptime` ile yük ortalamasını kontrol et.
3. `ls -l /etc` ile yapılandırma dosyalarının sahiplerini ve izinlerini incele.
4. Kendi ev klasöründe bir deneme dosyası oluştur, `chmod 640` ile iznini değiştir ve `ls -l` ile sonucu gör.
5. `systemctl status` komutuyla çalışan bir hizmetin (örneğin ssh) durumuna bak.
6. Bir Windows makinen varsa PowerShell'i aç; `Get-Volume`, `Get-Service` ve `Test-NetConnection` komutlarını dene.

Bu notlar, ileride kısa bir "sunucu sađlık kontrolü" listesine dönüşebilir.

Sık yapılan hatalar

- Her işi root kullanıcısıyla yapmak. Bir yazım hatası bütün sistemi etkileyebilir ve kimin ne yaptığı izlenemez.
- İnternette bulunan bir komutu ne yaptığını anlamadan üretim sunucusunda çalıştırmak.
- İzin sorununu "herkese her izni ver" (chmod 777) ile çözmeye çalışmak.
- Log döndürme ayarlamamak; diskin loglarla dolması.
- Güncellemeleri test etmeden doğrudan üretime uygulamak ya da hiç uygulamamak.
- SSH özel anahtarını ya da yönetici parolasını ekip içinde e-postayla paylaşmak.
- Windows sunucuda gerekmediğı hâlde masaüstü deneyimi kurmak ve kullanılmayan rolleri açık bırakmak.

Kendini dene

1. Linux'ta disklerin doluluk oranını okunabilir birimlerle gösteren komut hangisidir?
2. Debian ve Ubuntu'da kullanılan paket yöneticisinin adı nedir?
3. "-rw-r-----" izninin sayısal karşılığı nedir?
4. Windows Server'da kullanıcı ve bilgisayarları merkezi olarak yöneten rol hangisidir?
5. PowerShell'de bir sunucunun belirli bir portuna erişilip erişilemediğini hangi komutla test edersin?

Bulutta güvenlik, kimlik ve erişim

Bulutta güvenlik ihlallerinin önemli bir kısmı sağlayıcının altyapısındaki bir açıktan değil, müşterinin yaptığı bir hatadan kaynaklanır: herkese açık bırakılmış bir depolama alanı, ele geçirilmiş bir parola, koda gömülmüş bir erişim anahtarı, gereğinden geniş verilmiş bir yetki. Bu modül, bu hataları önlemek için bilmen gereken temel kavramları anlatır.

Paylaşılan sorumluluk modeli

Bulutta güvenlik, sağlayıcı ile müşteri arasında paylaşılır. AWS bunu "bulutun güvenliği" ve "buluttaki güvenlik" ayrımıyla anlatır:

- **Sağlayıcının sorumluluğu (bulutun güvenliği):** Veri merkezlerinin fiziksel güvenliği, donanım, ağ altyapısı, hipervizör ve yönetilen hizmetlerin alt katmanları.
- **Müşterinin sorumluluğu (buluttaki güvenlik):** Verinin kendisi, kullanıcı hesapları ve yetkiler, ağ ve güvenlik duvarı kuralları, şifreleme ayarları, IaaS'ta işletim sistemi ve uygulama güncellemeleri.

Sorumluluğun sınırı hizmet modeline göre kayar:

- **IaaS'ta** işletim sistemi yaması, güvenlik duvarı ve uygulama sende.
- **PaaS'ta** işletim sistemi ve çalışma ortamı sağlayıcıda; uygulama kodu, yapılandırma ve veri sende.
- **SaaS'ta** yazılımın neredeyse tamamı sağlayıcıda; ama kullanıcı hesapları, yetkiler, paylaşım ayarları ve veri yine sende.

Hangi model olursa olsun **veri, kimlikler ve erişim yetkileri her zaman müşterinin sorumluluğundadır**. Microsoft 365 kullanan bir şirkette, bir çalışanın parolası çalınıp e-postaları okunursa bu Microsoft'un değil, hesabı korumayan şirketin sorunudur.

Kimlik ve erişim yönetimi (IAM)

Bulutta "kim, neye, ne yapabilir?" sorusunun cevabı **kimlik ve erişim yönetimi (Identity and Access Management, IAM)** ile verilir. Temel kavramlar:

- **Kimlik doğrulama (authentication):** Kişinin iddia ettiği kişi olduğunu kanıtlaması. Parola, çok faktörlü doğrulama, sertifika.
- **Yetkilendirme (authorization):** Kimliği doğrulanmış kişinin hangi işlemi yapabileceği.
- **Kullanıcı:** Bir kişiyi temsil eden kimlik.
- **Grup:** Aynı yetkilere sahip kullanıcıların topluluğu. Yetkiler kişilere tek tek değil, gruplara verilir.
- **Rol:** Belirli yetkiler kümesi. Bir kişi, uygulama ya da hizmet bu rolü geçici olarak üstlenebilir.
- **Politika (policy):** Hangi işlemin hangi kaynak üzerinde izinli olduğunu tanımlayan kural belgesi.
- **Hizmet hesabı / yönetilen kimlik:** İnsan değil, bir uygulamanın kullandığı kimlik. Google Cloud'da service account, Azure'da managed identity, AWS'de bir hizmete atanan IAM rolü.

Sağlayıcılardaki karşılıklar:

- **AWS:** AWS IAM; kullanıcılar, gruplar, roller ve JSON biçiminde politikalar. Kurumsal kullanıcıların merkezi girişi için AWS IAM Identity Center.
- **Azure:** Kimlikler Microsoft Entra ID'de tutulur. Kaynaklara erişim Azure rol tabanlı erişim denetimi (Azure RBAC) ile verilir. Yerleşik roller arasında Owner (Sahip), Contributor (Katkıda Bulunan) ve Reader (Okuyucu) en bilinenleridir.

- **Google Cloud:** Cloud IAM; kimliklere (kullanıcı, grup, hizmet hesabı) kaynaklar üzerinde roller verilir. Önceden tanımlanmış (predefined) roller, geniş kapsamlı temel (basic) rollere tercih edilir.

En az yetki ilkesi

En az yetki ilkesi (least privilege), her kimliğe işini yapması için gereken en az yetkinin verilmesidir. Muhasebe raporlarını okuyan bir uygulama, raporları silebilmemelidir. Destek ekibindeki bir çalışan sunucuları izleyebilir ama silememelidir.

Uygulamada şöyle yapılır:

1. Önce "Bu kişi ya da uygulama tam olarak hangi işleri yapacak?" sorusunu yanıtla.
2. Sağlayıcının hazır rollerinden bu işe en dar olanı seç; gerekirse özel rol ya da politika yaz.
3. Yetkiyi en dar kapsamda ver: bütün hesap yerine tek bir kaynak grubu, tek bir proje ya da tek bir depolama alanı.
4. Yetkileri düzenli aralıklarla gözden geçir; işten ayrılan ya da görev değiştiren kişilerin yetkilerini kaldır.

Aşağıda AWS'de, yalnız tek bir S3 kovalarındaki dosyaları listeleme ve okuma izni veren bir politika örneği var. Kova adı örnektir. "Version" alanındaki tarih, AWS politika dilinin sürümünü gösteren sabit bir değerdir.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["s3:ListBucket", "s3:GetObject"],
      "Resource": [
        "arn:aws:s3::ornek-fatura-arsivi",
        "arn:aws:s3::ornek-fatura-arsivi/*"
      ]
    }
  ]
}
```

Bu politikayı alan kimlik dosyaları görebilir ve indirebilir ama silemez, değiştiremez, başka kovaya erişemez.

Kök hesap ve çok faktörlü doğrulama

Her bulut hesabının en geniş yetkiye sahip bir sahibi vardır. AWS'de buna **kök kullanıcı (root user)** denir; hesabı açarken kullanılan e-posta adresidir. Azure'da kiracının Genel Yönetici (Global Administrator) rolü, Google Cloud'da kuruluş yöneticisi benzer bir güce sahiptir.

Bu hesaplar için kurallar:

- **Çok faktörlü kimlik doğrulama (MFA)** mutlaka açılır. Parolanın yanına bir doğrulama uygulaması, donanım anahtarı ya da benzeri ikinci bir faktör eklenir.
- Günlük işler kök hesapla yapılmaz. Yönetim için ayrı, adı belli kişilere ait hesaplar açılır.

- Kök hesaba erişim bilgileri kasada ya da kurumun onaylı parola yöneticisinde tutulur; kimin erişebildiği bellidir.
- Kök hesaba ait erişim anahtarları (programatik erişim için) oluşturulmaz.

MFA yalnız yöneticiler için değil, bulut konsoluna giren herkes için açılmalıdır. MFA açık bir hesapta, çalınan bir parola tek başına hesaba girmeye yetmez.

Erişim anahtarları ve gizli bilgiler

Uygulamaların bulut hizmetlerine erişmesi için kimlik bilgisi gerekir. En riskli yöntem, uzun ömürlü bir erişim anahtarını kodun içine ya da yapılandırma dosyasına yazmaktır. Kod bir depoya gönderildiğinde anahtar da gider; herkese açık bir depoya gönderilirse kötü niyetli kişiler tarafından kısa sürede bulunabilir.

Daha güvenli yollar:

- Bulutta çalışan uygulamaya **rol ya da yönetilen kimlik** atamak. Uygulama geçici kimlik bilgilerini otomatik alır; kodda anahtar olmaz.
- Veritabanı parolası gibi gizli bilgileri **gizli bilgi yönetim hizmetinde** tutmak: AWS Secrets Manager, Azure Key Vault, Google Cloud Secret Manager.
- Gizli bilgileri düzenli olarak değiştirmek (rotation).
- Kod deposuna gizli bilgi gönderilmesini engelleyen tarama araçları kullanmak.

Şifreleme

- **Durağan veride şifreleme (encryption at rest):** Diskte, veritabanında, nesne depolamada duran verinin şifrlenmesi. Büyük sağlayıcılar birçok depolama hizmetinde bunu varsayılan olarak yapar. Anahtarları kendin yönetmek istersen AWS KMS, Azure Key Vault ya da Cloud KMS kullanılır.
- **Aktarılan veride şifreleme (encryption in transit):** Verinin ağda giderken şifrlenmesi. Web trafiği için HTTPS, yani TLS kullanılır. Şifresiz HTTP ile kişisel veri taşınmaz.

Şifreleme anahtarına erişim de bir yetkidir. Anahtarı silen ya da kaybeden, şifreli veriye de erişimini kaybeder.

Kayıt ve izleme

Bir olay olduğunda "kim, ne zaman, ne yaptı?" sorusuna cevap verebilmek gerekir. Bunun için yönetim işlemleri loglanır:

- AWS'de **AWS CloudTrail**, hesapta yapılan API çağrılarını kaydeder.
- Azure'da **etkinlik günlüğü (Activity Log)** abonelikteki yönetim işlemlerini kaydeder; Azure Monitor ile toplanır.
- Google Cloud'da **Cloud Audit Logs** yönetim ve erişim işlemlerini kaydeder.

Loglar yalnız tutulmakla kalmamalı; kök hesaba giriş, güvenlik kuralının değiştirilmesi, olağan dışı bölgede kaynak açılması gibi olaylar için uyarı kurulmalıdır. Sağlayıcıların güvenlik durumu değerlendirme araçları (AWS Security Hub ve Amazon GuardDuty, Microsoft Defender for Cloud, Google Cloud Security Command Center) yanlış yapılandırmaları ve şüpheli etkinlikleri raporlar.

Standartlar ve mevzuat

Bilgi güvenliği yönetimi için uluslararası alanda en yaygın standart **ISO/IEC 27001**'dir. Büyük bulut sağlayıcıları bu ve benzeri standartlara uyum belgelerini kendi uyum (compliance) sayfalarında yayımlar. Sağlayıcının belgesi senin sorumluluğundaki kısmı kapsamaz; kendi yapılandırmanın güvenliğini sen sağlarsın.

Kişisel veri işleyen kurumlar için 6698 sayılı **KVKK**'nın veri güvenliğine ilişkin yükümlülükleri, uygun teknik ve idari tedbirlerin alınmasını ister. Erişim yetkilerinin sınırlandırılması, loglama ve şifreleme bu tedbirlerin parçasıdır.

Örnek vaka

İzmir'de bir diş kliniği zinciri randevu ve hasta görüntülerini bulutta saklıyor. Yeni gelen BT sorumlusu ilk hafta bir güvenlik gözden geçirmesi yapıyor ve şunları buluyor:

1. Altı kişi aynı yönetici hesabını, aynı parolayla kullanıyor. MFA kapalı.
2. Hasta röntgenlerinin tutulduğu depolama alanında "herkese açık okuma" ayarı açık; eski bir web uygulaması için açılmış, uygulama artık kullanılmıyor.
3. Randevu uygulamasının kodunda veritabanı parolası açık metin olarak yazılı.
4. Yönetim işlemleri loglanıyor ama loglara kimse bakmıyor.

Yapılan düzeltmeler sırasıyla şöyle:

1. Ortak hesap kapatılıyor; her kişiye kendi adına hesap açılıyor ve MFA zorunlu hâle getiriliyor. Yönetici yetkisi yalnız iki kişide kalıyor; diğerlerine görevlerine uygun roller veriliyor.
2. Depolama alanındaki genel erişim kaldırılıyor; hesap düzeyinde genel erişim engeli açılıyor. Hasta verisi kişisel veri olduğu için, açık kalınan sürede erişim olup olmadığı loglardan inceleniyor ve hukuk birimine bilgi veriliyor.
3. Veritabanı parolası değiştiriliyor ve gizli bilgi yönetim hizmetine taşınıyor; uygulama parolayı oradan okuyor.
4. Kök hesap girişi, güvenlik kuralı değişikliği ve genel erişim ayarı değişikliği için e-posta uyarıları kuruluyor.

Sık yapılan hatalar

- Bulutta güvenliğin tamamen sağlayıcının işi olduğunu düşünmek.
- Ortak hesap kullanmak; kimin ne yaptığının izlenememesi.
- "Şimdilik" diye geniş yetki verip daraltmayı unutmak.
- Erişim anahtarlarını koda, yapılandırma dosyasına ya da sohbet mesajlarına yazmak.
- İşten ayrılan çalışanların bulut hesaplarını kapatmamak.
- Logları açıp hiç incelememek.

Kendini dene

1. Paylaşılan sorumluluk modelinde, hizmet modeli ne olursa olsun müşteride kalan sorumluluklar nelerdir?
2. En az yetki ilkesi ne demektir?
3. AWS'de hesabı açan e-posta adresine bağlı en yetkili kullanıcıya ne ad verilir ve bu hesap için ilk yapılacak iş nedir?
4. Bir uygulamanın veritabanı parolasını kodun içine yazmak yerine nerede saklaması önerilir?
5. AWS'de hesapta yapılan API çağrılarını kaydeden hizmet hangisidir?

Maliyet yönetimi

Bulutta kaynak açmak birkaç dakika sürer ve kimseden onay istemez. Bu hız büyük bir avantajdır ama maliyet açısından bir risk de taşır: açılan her kaynak, silinene kadar fatura üretir. Şirket içi altyapıda bir sunucu satın almak için bütçe onayı gerekir; bulutta ise bir geliştiricinin unuttuğu test sunucusu aylarca çalışabilir. Maliyet yönetimi, bu nedenle bulutta teknik işin bir parçasıdır.

Bu modülde fiyat vermiyoruz. Fiyatlar bölgeye, hizmete, sözleşmeye ve zamana göre değişir. Önemli olan, maliyetin nereden doğduğunu bilmek ve onu ölçüp kontrol edebilmektir. Güncel fiyatlar için sağlayıcıların fiyatlandırma sayfalarını ve fiyat hesaplayıcılarını (AWS Pricing Calculator, Azure Pricing Calculator, Google Cloud Pricing Calculator) kullan.

Kullandıkça öde ve faturalandırma birimleri

Bulut hizmetlerinin çoğu **kullandıkça öde (pay as you go)** modeliyle ücretlendirilir. Ölçü birimi hizmete göre değişir:

- **Sanal sunucu:** Çalıştığı süre (saniye, dakika ya da saat bazında) ve boyutu (işlemci, bellek). Kapalı (durdurulmuş) bir sanal makine işlem ücreti üretmez ama bağlı diskleri ücret üretmeye devam eder.
- **Disk ve depolama:** Ayrılan ya da kullanılan kapasite ve süre. Nesne depolamada ayrıca okuma, yazma ve listeleme istek sayısı da ücretlendirilebilir.
- **Veri transferi:** Bulut içine giren veri genellikle ücretsizdir; buluttan internete çıkan veri (egress) ve bölgeler arası transfer çoğunlukla ücretlidir.
- **Yönetilen hizmetler:** Veritabanı, yük dengeleyici, NAT ağ geçidi gibi hizmetler hem çalıştıkları süre hem işledikleri veri üzerinden ücretlendirilebilir.
- **Sunucusuz hizmetler:** İstek sayısı ve çalışma süresi.
- **Lisanslar:** Windows Server ya da SQL Server gibi lisanslı yazılımlar sanal makine fiyatına eklenebilir ya da mevcut lisans getirilebilir.

Maliyetin gizli kaynakları

Faturada şaşırtan kalemler genellikle şunlardır:

- Silinen sanal makineden geriye kalan, hiçbir yere bağlı olmayan diskler.
- Yıllar önce alınmış ve hiç silinmemiş anlık görüntüler (snapshot).
- Kullanılmayan ama ayrılmış genel IP adresleri.
- Test için açılıp unutulmuş büyük boyutlu sanal makineler ve veritabanları.
- Özel alt ağdaki sunucuların internete çıktığı NAT ağ geçidinin işlediği veri.
- Bölgeler arası ya da internete giden yüksek veri transferi.
- Log ve izleme hizmetlerine gönderilen çok büyük hacimli loglar.

İndirim modelleri

Kullandıkça öde en esnek modeldir. Uzun süre sabit çalışacak iş yükleri için sağlayıcılar taahhüt karşılığında indirim sunar:

- **Taahhütlü kullanım:** Bir ya da üç yıllık kullanım taahhüdü karşılığında indirim. AWS'de Savings Plans ve Reserved Instances, Azure'da Azure Reservations ve Azure savings plan for compute, Google

Cloud'da committed use discounts. Taahhüt edilen tutar, kullansan da kullanmasan da ödenir.

- **Kesintiye uğrayabilen kapasite:** Sağlayıcının boşta kalan kapasitesi, belirgin indirimle ama her an geri alınabilecek şekilde sunulur. AWS'de Spot Instances, Azure'da Azure Spot Virtual Machines, Google Cloud'da Spot VMs. Kesintiye dayanıklı işler için uygundur: toplu veri işleme, test ortamları, görüntü işleme. Bir veritabanı ya da tek bir web sunucusu için uygun değildir.

Doğru sıra şudur: önce kaynakları doğru boyuta getir ve gereksizleri kapat, sonra sabit kalan kısım için taahhüt al. Fazla büyük bir sunucu için taahhüt almak, israfı üç yıllığına kilitlemek demektir.

Etiketleme

Etiket (tag), kaynaklara eklenen anahtar-değer çiftidir. AWS ve Azure'da "tag", Google Cloud'da "label" adı kullanılır. Etiketler, faturayı proje, ortam, birim ya da sahip bazında bölmeyi sağlar. Etiketsiz bir faturada "bu ayki artış nereden geldi?" sorusunun cevabını bulmak çok zordur.

Basit ve tutarlı bir etiket şeması yeterlidir:

```
proje          = eticaret-site
ortam          = uretim | test | gelistirme
sahip          = bt-altyapi
maliyet-merkezi = pazarlama
```

Etiketleme kuralları:

1. Etiket adlarını ve izin verilen değerleri yazılı bir belgeyle belirle. "Ortam", "ortam", "env" gibi farklı yazımlar raporu bozar.
2. Etiket kullanımını mümkünse politikayla zorunlu kıl ya da denetle. AWS'de Organizations etiket politikaları (tag policies) etiket yazımını standartlaştırır; Azure Policy etiketsiz kaynak oluşturulmasını engelleyebilir ya da eksik etiketi otomatik ekleyebilir.
3. Etiketleri faturada kullanılabilir hâle getir. AWS'de etiketlerin maliyet raporlarında görünmesi için "maliyet dağıtım etiketi" (cost allocation tag) olarak etkinleştirilmesi gerekir.
4. Etiketlenmemiş kaynakları düzenli olarak raporla.

Aşağıdaki komutlar etiket kullanımına iki örnektir. Birincisi Azure'da bir test kaynak grubuna etiket ekler, ikincisi AWS'de "ortam=test" etiketli sunucuların kimliklerini listeler.

```
# Azure: kaynak grubuna etiket ekler (grup adı örnektir)
az group update --name rg-eticaret-test --tags ortam=test proje=eticaret sahip=bt-
altyapi

# AWS: ortam=test etiketli EC2 sunucularının kimliklerini listeler
aws ec2 describe-instances \
  --filters "Name=tag:ortam,Values=test" \
  --query "Reservations[].Instances[].InstanceId" --output text
```

Bütçe ve uyarılar

Her bulut hesabında, ilk gün, bir **bütçe uyarısı** kurulmalıdır. Bütçe, belirli bir dönem için beklenen harcamadır; uyarı, gerçek ya da tahmini harcama bütçenin belirli bir oranına ulaştığında e-posta ya da mesaj gönderir.

- AWS'de **AWS Budgets**,
- Azure'da **Microsoft Cost Management** içindeki bütçeler,
- Google Cloud'da **Cloud Billing bütçeleri** kullanılır.

Önemli ayrıntı: **bütçe uyarısı harcamayı kendiliğinden durdurmaz**. Yalnız haber verir. Uyarıyı alan birinin harekete geçmesi gerekir. Bazı sağlayıcılarda bütçe aşımında otomatik eylem tanımlamak mümkündür ama bunun nasıl çalıştığı dikkatle test edilmelidir; üretimdeki bir hizmeti otomatik olarak durdurmak işi kesebilir.

İyi bir bütçe düzeni şöyle kurulur:

1. Her hesap, abonelik ya da proje için ayrı bütçe.
2. Gerçek harcama için %50, %80 ve %100 eşiklerinde uyarı.
3. Tahmini harcama (dönem sonu öngörüsü) bütçeyi aşacaksa erken uyarı.
4. Uyarıların tek bir kişiye değil, bir ekip adresine gitmesi.

Maliyeti izlemek ve düşürmek

Harcamayı incelemek için sağlayıcıların araçları kullanılır: **AWS Cost Explorer**, **Microsoft Cost Management** maliyet analizi ve **Google Cloud Billing raporları**. Bu araçlarda harcama hizmete, bölgeye, etikete ve zamana göre gruplanabilir.

Maliyeti düşürmenin sık kullanılan yolları:

- **Doğru boyutlandırma (rightsizing)**: Kullanım ölçümlerine bakarak gereğinden büyük sunucuları küçültmek. Sağlayıcıların öneri araçları (AWS Compute Optimizer, Azure Advisor, Google Cloud Recommender) bu konuda öneri sunar.
- **Zamanlama**: Geliştirme ve test sunucularını mesai dışında ve hafta sonu kapatmak.
- **Sahipsiz kaynakları temizlemek**: Bağlı olmayan diskler, eski anlık görüntüler, kullanılmayan IP adresleri.
- **Depolama katmanlarını kullanmak**: Eski verileri yaşam döngüsü kurallarıyla daha ucuz katmanlara taşımak.
- **Otomatik ölçekleme**: Sabit en yüksek kapasite yerine talebe göre kaynak eklemek ve çıkarmak.
- **Yönetilen ya da sunucusuz hizmete geçmek**: Sürekli açık duran küçük işleri olay tetiklemeli hizmetlere taşımak.

Örnek vaka

Gaziantep'te bir gıda toptancısı altı ay önce sipariş sistemini buluta taşıdı. İlk aylarda fatura beklentiye uygundu; son iki ayda ise belirgin şekilde arttı. Kimse nedenini açıklayamıyor, çünkü kaynakların hiçbir etiketli değil.

BT sorumlusu şu adımları izliyor:

1. Maliyet aracında harcamayı hizmete göre grupluyor. Artışın büyük kısmı sanal makinelerden ve disklerden geliyor.
2. Sanal makineleri tek tek inceliyor. Bir yazılım firmasının kurulum sırasında açtığı iki büyük test sunucusu kurulum bittikten sonra da çalışmaya devam ediyor.
3. Hiçbir sunucuya bağlı olmayan dört disk ve aylardır alınan, hiç silinmeyen günlük anlık görüntüler buluyor.

4. Test sunucularının sahibini ve hâlâ gerekli gerekmediğini yazılım firmasına soruyor. Biri gerekmiyor; yedeği alınıp siliniyor. Diğerleri yalnız mesai saatinde açık kalacak şekilde zamanlanıyor.
5. Bağlı olmayan disklerin içeriğini kontrol edip, gerekeni arşivleyerek siliyor; anlık görüntüler için saklama süresi belirliyor.
6. Bütün kaynaklara proje, ortam ve sahip etiketleri ekliyor; aylık bütçe ve uyarılar kuruyor.

Sonraki ayın faturası beklenen düzeye iniyor; artık her artışın nedeni etiketlerden hızla görülebiliyor.

Uygulama

Deneme hesabında ya da sorumlu olduğun bir hesapta şunları yap:

1. Aylık, küçük bir tutarla bir bütçe oluştur. %50, %80 ve %100 eşiklerinde e-posta uyarısı ekle.
2. Yukarıdaki dört etiketten oluşan bir etiket şeması yaz ve ekibinle paylaş.
3. Maliyet aracında son ayın harcamasını hizmete göre grupta. En büyük üç kalemi not et.
4. Bağlı olmayan diskleri, eski anlık görüntüleri ve kullanılmayan genel IP adreslerini ara.
5. Test ortamındaki sunucular için bir kapatma saati belirle.
6. Bulduklarını kısa bir rapor hâlinde yöneticine sun: ne bulundu, ne yapıldı, ne bekliyor.

Sık yapılan hatalar

- Bütçe uyarısını harcama sınırı sanmak.
- Taşımadan önce maliyet tahmini yapmamak ya da yalnız sunucu maliyetini hesaplayıp veri transferi, yedek ve lisans kalemlerini unutmak.
- Etiketlemeyi sonraya bırakmak. Yüzlerce kaynak birikince geriye dönük etiketlemek zahmetlidir.
- Doğru boyutlandırma yapmadan taahhütlü kullanım satın almak.
- Kesintiye uğrayabilen kapasiteyi kesintiye dayanıksız işler için kullanmak.
- Sanal makineyi durdurunca bütün maliyetin bittiğini sanmak; disk ve IP ücretlerini unutmak.

Kendini dene

1. Durdurulmuş bir sanal makine için hangi kalemler ücret üretmeye devam edebilir?
2. Bütçe uyarısı harcamayı otomatik olarak durdurur mu?
3. Etiketlemenin maliyet yönetimine katkısı nedir?
4. Sağlayıcının boşta kalan kapasitesini indirimli ama kesintiye uğrayabilir şekilde kullanmak hangi işler için uygundur?
5. Taahhütlü kullanım satın almadan önce neden doğru boyutlandırma yapılmalıdır?

Konteynerler ve Docker

Bir uygulamanın geliştiricinin bilgisayarında çalışıp sunucuda çalışmaması, BT ekiplerinin en bilinen sorunlarından biridir. Nedeni genellikle ortam farkıdır: farklı bir kütüphane sürümü, eksik bir paket, farklı bir yapılandırma. Konteynerler bu sorunu, uygulamayı ihtiyaç duyduğu her şeyle birlikte paketleyerek çözer.

Konteyner nedir?

Konteyner (container), bir uygulamayı ve çalışması için gereken kütüphane, araç ve ayarları tek bir paket hâlinde, işletim sisteminin geri kalanından yalıtılmış olarak çalıştıran yöntemdir. Konteynerler, üzerinde çalıştıkları makinenin işletim sistemi çekirdeğini paylaşır.

Sanal makineyle farkı şöyledir:

- **Sanal makine** kendi tam işletim sistemini taşır. Başlaması dakikalar alabilir, boyutu gigabaytlarla ölçülür. Yalıtım güçlüdür.
- **Konteyner** yalnız uygulamayı ve bağımlılıklarını taşır; çekirdeği ana makineyle paylaşır. Saniyeler içinde başlar, boyutu genellikle çok daha küçüktür. Aynı makinede çok sayıda konteyner çalışabilir.

İkisi birbirinin rakibi değildir. Bulutta konteynerler çoğu zaman sanal makinelerin üzerinde çalışır.

Linux konteynerleri Linux çekirdeği ister; Windows ve macOS'ta Docker Desktop gibi araçlar bunun için arka planda küçük bir Linux sanal makinesi kullanır. Windows uygulamaları için ayrıca Windows konteynerleri de vardır.

Docker ve temel kavramlar

Docker, konteyner oluşturmayı ve çalıştırmayı yaygınlaştıran araçtır. Bugün konteyner imajlarının biçimi Open Container Initiative (OCI) standartlarıyla tanımlanır; bu yüzden Docker ile oluşturulan imajlar Podman gibi başka araçlarla ve Kubernetes'te de çalışır.

- **İmaj (image):** Konteynerin salt okunur şablonu. Uygulamayı, bağımlılıklarını ve nasıl başlatılacağını içerir. Katmanlardan oluşur.
- **Konteyner:** Bir imajın çalışan örneği. Aynı imajdan çok sayıda konteyner başlatılabilir.
- **Dockerfile:** İmajın nasıl oluşturulacağını adım adım tarif eden metin dosyası.
- **Kayıt deposu (registry):** İmajların saklandığı ve paylaşıldığı yer. Docker Hub herkese açık bir kayıt deposudur. Bulutta Amazon ECR, Azure Container Registry ve Artifact Registry kullanılır.
- **Etiket (tag):** İmajın sürümünü gösteren ad parçası. "kampanya-sayfasi:1.0" ifadesinde "1.0" etikettir.
- **Birim (volume):** Konteyner silindiğinde kaybolmaması gereken verinin saklandığı alan.

Konteynerin içindeki dosya sistemi geçicidir: konteyner silinince içine yazılan veri de gider. Kalıcı veri (veritabanı dosyaları, kullanıcı yüklemeleri) birimlerde ya da harici depolama hizmetlerinde tutulur.

İlk Dockerfile

Aşağıdaki Dockerfile, statik bir kampanya sayfasını nginx web sunucusuyla yayınlayan bir imaj tanımlar. Aynı klasörde bir index.html dosyası olduğu varsayılır.

```
# Küçük boyutlu resmî nginx imajını temel al
FROM nginx:alpine
# Kendi sayfamızı nginx'in yayın klasörüne kopyala
COPY index.html /usr/share/nginx/html/index.html
# Konteynerin 80 portunu kullandığını belgele
EXPOSE 80
```

İmajı oluşturup çalıştırmak için:

```
# Bulunduğun klasördeki Dockerfile'dan imaj oluşturur
docker build -t kampanya-sayfasi:1.0 .
# Konteyneri arka planda başlatır; bilgisayarın 8080 portunu konteynerin 80 portuna
bağlar
docker run -d --name kampanya -p 8080:80 kampanya-sayfasi:1.0
# Çalışan konteynerleri listeler
docker ps
# Konteynerin loglarını gösterir
docker logs kampanya
# Konteyneri durdurur ve siler
docker stop kampanya
docker rm kampanya
```

Konteyner çalışırken tarayıcıda localhost:8080 adresini açarsan sayfayı görürsün.

Bir uygulama imajı örneği

Basit bir Python uygulaması için Dockerfile şöyle olabilir:

```
# Küçük boyutlu resmî Python imajı
FROM python:3-slim
# Uygulama klasörü
WORKDIR /app
# Önce bağımlılık listesini kopyala ve kur (katman önbelleği için)
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt
# Uygulama kodunu kopyala
COPY . .
# Uygulamayı yetkisiz bir kullanıcıyla çalıştır
USER nobody
CMD ["python", "app.py"]
```

Burada iki iyi uygulama var. Birincisi, bağımlılıklar koddan önce kopyalanır; kod değiştiğinde bağımlılık katmanı yeniden kurulmaz, imaj daha hızlı oluşur. İkincisi, uygulama root yerine yetkisiz bir kullanıcıyla çalışır. Üretimde temel imajın etiketi daha kesin bir sürüme sabitlenir; böylece aynı Dockerfile her seferinde aynı temel imajla derlenir.

Birden fazla konteyner: Docker Compose

Gerçek uygulamalar çoğu zaman birden fazla parçadan oluşur: web uygulaması, veritabanı, önbellek.

Docker Compose, bu parçaları tek bir YAML dosyasında (compose.yaml) tanımlayıp `docker compose up` komutuyla birlikte başlatmayı sağlar. Geliştirme ortamı kurmak ve küçük deneme ortamları için çok kullanışlıdır. Üretimde, çok sunuculu ortamlarda ise genellikle Kubernetes ya da bulut sağlayıcısının konteyner hizmetleri tercih edilir.

Bulutta konteyner çalıştırma

Konteyner imajını bir kayıt deposuna gönderdikten sonra bulutta çalıştırmanın birkaç yolu vardır:

- **Sunucusuz konteyner hizmetleri:** Sunucu yönetmeden konteyner çalıştırırsın. Google Cloud Run, Azure Container Apps, AWS'de Fargate ile Amazon ECS. Küçük ve orta ölçekli web uygulamaları ve API'ler için kolay bir başlangıçtır.
- **Konteyner orkestrasyon hizmetleri:** Çok sayıda konteyneri birden fazla sunucuda yöneten hizmetler. Amazon ECS ve yönetilen Kubernetes hizmetleri (Amazon EKS, AKS, GKE).
- **Sanal makinede Docker:** Bir sanal sunucuya Docker kurup konteyneri orada çalıştırmak. Basittir ama ölçekleme, güncelleme ve izleme işleri sende kalır.

Konteyner güvenliği

- Güvenilir ve resmî temel imajlar kullan; küçük imajlar daha az gereksiz bileşen ve daha az açık taşır.
- İmajlara parola, erişim anahtarı ya da sertifika koyma. Dockerfile'a yazılan gizli bilgi imaj katmanlarında kalır ve imajı alan herkes görebilir.
- Uygulamayı root kullanıcısıyla çalıştırma.
- İmajları bilinen güvenlik açıklarına karşı tara. Bulut sağlayıcılarının kayıt depolarında tarama özellikleri bulunur.
- İmajları düzenli olarak yeniden oluştur; temel imajdaki güvenlik güncellemeleri ancak böyle uygulamana gelir.

Ticari ortamda Docker Desktop kullanılacaksa lisans koşullarını kontrol et; kurumun büyüklüğüne göre ücretli abonelik gerekebilir. Sunucularda kullanılan Docker Engine ve açık kaynak alternatifler bu konuda farklı koşullara sahiptir.

Örnek vaka

Eskişehir'de bir belediyenin bilgi işlem birimi, vatandaşların su arıza bildirimini yaptığı küçük bir web uygulaması geliştirmiş. Uygulama geliştiricinin bilgisayarında çalışıyor ama test sunucusunda farklı bir Python sürümü olduğu için hata veriyor. Her kurulumda sunucuya elle paket yükleniyor ve kimin neyi kurduğu bilinmiyor.

Birim şu adımlarla konteynere geçiyor:

1. Uygulama için yukarıdakine benzer bir Dockerfile yazılıyor; bağımlılıklar requirements.txt dosyasında sürümleriyle belirtiliyor.
2. İmaj geliştiricinin bilgisayarında oluşturulup test ediliyor.
3. Aynı imaj, kurumun kayıt deposuna "1.0" etiketiyle gönderiliyor.
4. Test sunucusu ve üretim sunucusu bu imajı çekip çalıştırıyor. İki ortamda aynı imaj çalıştığı için "bende çalışıyordu" sorunu ortadan kalkıyor.

5. Veritabanı konteynerin içine değil, ayrı bir yönetilen veritabanı hizmetine konuyor; veritabanı parolası ortam değişkeniyle ve gizli bilgi yönetim hizmetinden veriliyor.
6. Yeni sürümde sorun çıkarsa önceki etiketli imaja dönülüyor.

Uygulama

Bilgisayarına Docker'ı ya da açık kaynaklı bir alternatifi kur (kurum bilgisayarında bunu yapmadan önce BT biriminin onayını al). Sonra:

1. Boş bir klasör aç; içine tek satırlık bir index.html ve yukarıdaki ilk Dockerfile'ı koy.
2. `docker build` ile imajı oluştur, `docker images` ile listede gör.
3. `docker run` ile başlat, tarayıcıdan localhost:8080 adresini aç.
4. index.html'i değiştir; çalışan konteynerde değişikliğin görünmediğine dikkat et. İmajı "1.1" etiketiyle yeniden oluşturup yeni konteyner başlat.
5. `docker logs` ile istek loglarını incele.
6. İş bitince konteynerleri durdur ve sil.

Dördüncü adımın dersi: konteyner değiştirilmez, yeni imajla yeniden oluşturulur.

Sık yapılan hatalar

- Kalıcı veriyi konteynerin içine yazmak; konteyner silinince verinin kaybolması.
- "latest" etiketine güvenmek; hangi sürümün çalıştığının belirsiz kalması.
- Gizli bilgileri Dockerfile'a ya da imaja koymak.
- Çalışan konteynere bağlanıp elle değişiklik yapmak. Bu değişiklikler bir sonraki yeniden başlatmada kaybolur.
- Gereksiz büyük temel imajlar kullanmak; imajların yavaş indirilmesi ve daha fazla açık taşınması.
- Konteyneri sanal makine gibi kullanıp içine birçok ilgisiz hizmet doldurmak. Her konteyner tek bir işi yapmalıdır.

Kendini dene

1. Konteyner ile sanal makine arasındaki temel fark nedir?
2. İmaj ile konteyner arasındaki ilişki nedir?
3. Bir imajın nasıl oluşturulacağını tarif eden dosyanın adı nedir?
4. Konteyner silindiğinde kaybolmaması gereken veri nerede tutulmalıdır?
5. Dockerfile'a veritabanı parolası yazmak neden sakıncalıdır?

Kubernetes'e giriş ve DevOps kültürü

Tek bir sunucuda birkaç konteyner çalıştırmak kolaydır. Ama onlarca hizmet, yüzlerce konteyner ve birden fazla sunucu söz konusu olduğunda yeni sorular doğar: Bir konteyner çökerse onu kim yeniden başlatacak? Bir sunucu arızalanırsa üzerindeki konteynerler nereye taşınacak? Yeni sürüm kesinti olmadan nasıl yayınlanacak? Yük artınca konteyner sayısı nasıl artırılacak? Kubernetes bu soruları yanıtlamak için geliştirildi. DevOps ise bu teknik araçları kullanan ekiplerin nasıl çalışacağıyla ilgilidir.

Kubernetes nedir?

Kubernetes, konteynerleri birden fazla sunucu üzerinde otomatik olarak dağıtan, çalıştıran, ölçekleyen ve iyileştiren açık kaynaklı bir **orkestrasyon** platformudur. İlk olarak Google'da geliştirildi; bugün Cloud Native Computing Foundation (CNCF) çatısı altında, geniş bir toplulukla geliştiriliyor. Kısaltması "K8s"tir.

Kubernetes'in temel fikri **istenen durum (desired state)** yaklaşımıdır. Sen "bu uygulamadan 3 kopya çalışsın" dersin; Kubernetes sürekli olarak gerçek durumu istenen durumla karşılaştırır. Bir kopya çökerse yenisini başlatır. Bir sunucu giderse kopyaları diğer sunuculara yerleştirir. Ne yapılacağını adım adım değil, sonucun ne olması gerektiğini tarif edersin. Buna **bildirimsel (declarative)** yönetim denir.

Temel yapı ve nesneler

- **Küme (cluster):** Kubernetes'in yönettiği makinelerin tamamı.
- **Kontrol düzlemi (control plane):** Kümenin beyni. İstenen durumu saklar, kararları verir, konteynerleri düğümlere yerleştirir. Yönetilen Kubernetes hizmetlerinde kontrol düzlemini sağlayıcı işletir.
- **Düğüm (node):** Konteynerlerin çalıştığı sunucu; genellikle bir sanal makine.
- **Pod:** Kubernetes'te çalıştırılan en küçük birim. Bir ya da birbirine sıkı bağlı birkaç konteyner içerir. Pod'lar geçicidir; silinip yeniden oluşturulabilirler ve her seferinde farklı IP adresi alabilirler.
- **Deployment:** Belirli sayıda pod'un çalışmasını ve sürüm güncellemelerinin kademeli yapılmasını sağlayan nesne.
- **Service:** Değişen pod'ların önünde sabit bir ad ve adres sunan, trafiği pod'lara dağıtan nesne. ClusterIP türü küme içinden, LoadBalancer türü dışarıdan erişim için kullanılır.
- **Ingress:** Dışarıdan gelen HTTP ve HTTPS trafiğini alan adına ya da yola göre doğru Service'e yönlendiren kurallar.
- **Namespace:** Kümeyi mantıksal bölümlere ayırır; örneğin "test" ve "uretim" ya da her ekip için ayrı bir alan.
- **ConfigMap ve Secret:** Uygulama ayarlarını ve gizli bilgileri imajın dışında tutmak için kullanılır. Secret'ların varsayılan olarak yalnız kodlanmış (base64) olduğunu, şifrelenmiş olmadığını bil; küme ayarlarında şifreleme ve erişim kısıtlaması ayrıca yapılır.
- **Kalıcı birim (PersistentVolume):** Pod'lar silindiğinde kaybolmaması gereken veri için depolama.

Bir Deployment tanımı

Kubernetes nesneleri genellikle YAML dosyalarıyla tanımlanır. Aşağıdaki dosya, nginx imajından 3 kopya çalıştıran bir Deployment tanımlar.

```
# web uygulamasından 3 kopya çalıştırır
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web
spec:
  replicas: 3
  selector:
    matchLabels:
      app: web
  template:
    metadata:
      labels:
        app: web
    spec:
      containers:
        - name: web
          image: nginx:alpine
```

"selector" ile "template" altındaki "labels" eşleşmelidir; Deployment, yönettiği pod'ları bu etiketle tanır. Bu dosya `kubectl apply -f web.yaml` komutuyla kümeye uygulanır. Dosyadaki "replicas" değerini 3'ten 5'e çıkarıp yeniden uygularsan Kubernetes iki pod daha başlatır.

kubectl ile kümeyi okumak

kubectl, Kubernetes kümesini yönetmek için kullanılan komut satırı aracıdır. Aşağıdaki komutlar yalnız bilgi okur.

```
# Kümedeki düğümleri ve durumlarını listeler
kubectl get nodes
# Bir namespace'teki pod'ları listeler
kubectl get pods -n üretim
# Bir pod'un ayrıntılarını ve son olaylarını gösterir (pod adı örnektir)
kubectl describe pod web-7d9c5b8f6-abcde -n üretim
# Pod'daki konteynerin loglarını gösterir
kubectl logs web-7d9c5b8f6-abcde -n üretim
```

Bir pod çalışmıyorsa sırası şudur: `kubectl get pods` ile durumuna bak ("Pending", "CrashLoopBackOff", "ImagePullBackOff" gibi), `kubectl describe pod` ile olaylar bölümünü oku, sonra `kubectl logs` ile uygulama loguna geç. "ImagePullBackOff" çoğunlukla imaj adının ya da kayıt deposu yetkisinin yanlış olduğunu, "CrashLoopBackOff" uygulamanın başlar başlamaz çöktüğünü gösterir.

Yönetilen Kubernetes ve ne zaman gerekli olduğu

Kubernetes'in kontrol düzlemini kendin kurup işletmek uzmanlık ister. Bu yüzden çoğu kurum yönetilen hizmetleri kullanır: **Amazon EKS**, **Azure Kubernetes Service (AKS)** ve **Google Kubernetes Engine (GKE)**. Bu hizmetlerde kontrol düzleminin kurulumu, güncellenmesi ve yüksek erişilebilirliği büyük ölçüde sağlayıcıdadır; düğümler, uygulamalar ve yapılandırma sende kalır.

Birkaç hizmetten oluşan küçük bir uygulama için Cloud Run, Azure Container Apps ya da ECS ile Fargate gibi daha basit sunucusuz konteyner hizmetleri çoğu zaman yeterlidir. Kubernetes'e; çok sayıda hizmet, birden fazla ekip, taşınabilirlik ihtiyacı ve onu işletecek bilgi birikimi olduğunda geçmek mantıklıdır.

DevOps nedir?

DevOps, yazılım geliştirme (development) ve BT operasyonları (operations) ekiplerinin ayrı duvarlar arkasında değil, ortak hedefle birlikte çalışmasını amaçlayan bir kültür ve uygulamalar bütünüdür. Geleneksel yapıda geliştirici kodu yazar ve "duvarın öbür tarafına" atar; operasyon ekibi onu yayına almaya ve çalışır tutmaya çalışır. Sorun çıktığında iki ekip birbirini suçlar.

DevOps'un temel ilkeleri:

- **Ortak sorumluluk:** Uygulamanın üretimde sağlıklı çalışması herkesin işidir.
- **Otomasyon:** Test, derleme, dağıtım ve altyapı kurulumu elle değil, tekrarlanabilir otomasyonla yapılır.
- **Küçük ve sık değişiklik:** Altı ayda bir büyük sürüm yerine sık aralıklarla küçük değişiklikler yayınlanır. Küçük değişikliğin hatası daha kolay bulunur ve geri alınır.
- **Ölçme ve izleme:** Uygulamanın durumu ölçümler, loglar ve uyarılarla sürekli izlenir.
- **Suçlamasız değerlendirme:** Bir olaydan sonra "kim hata yaptı?" yerine "sistem bu hataya nasıl izin verdi, nasıl önleriz?" sorusu sorulur.

DevOps bir unvan ya da tek bir araç değildir. "DevOps mühendisi" iş ilanları genellikle otomasyon, CI/CD, bulut ve konteyner bilgisi isteyen pozisyonlardır; ama kültür değişmeden araç eklemek beklenen sonucu vermez.

Sürüm kontrolü

DevOps'un temeli **sürüm kontrolüdür**. Kod, yapılandırma dosyaları ve altyapı tanımları bir sürüm kontrol sisteminde tutulur. Bugün en yaygın sistem **Git**'tir; GitHub, GitLab, Bitbucket ve Azure Repos gibi hizmetler Git depolarını barındırır.

Temel çalışma biçimi şöyledir: her değişiklik ayrı bir dalda (branch) yapılır, bir **birleştirme isteği** (pull request ya da merge request) açılır, ekipten biri değişikliği inceler, otomatik testler geçerse değişiklik ana dala alınır. Böylece her değişikliğin kim tarafından, ne zaman ve neden yapıldığı kayıtlı olur.

CI/CD

- **Sürekli entegrasyon (Continuous Integration, CI):** Geliştiricilerin değişiklikleri sık sık ortak koda eklemesi ve her eklemede kodun otomatik olarak derlenip test edilmesi. Hata, ortaya çıktığı gün yakalanır.
- **Sürekli teslim (Continuous Delivery):** Testten geçen her değişikliğin yayına hazır bir paket hâline getirilmesi. Üretime alma kararı bir onayla verilir.
- **Sürekli dağıtım (Continuous Deployment):** Testlerden geçen değişikliğin insan onayı olmadan otomatik olarak üretime alınması.

Tipik bir CI/CD hattı (pipeline) şu aşamalardan geçer:

1. Geliştirici değişikliği Git deposuna gönderir.
2. Hat otomatik başlar; kod derlenir, birim testleri çalışır.
3. Güvenlik ve kod kalitesi taramaları yapılır.

4. Konteyner imajı oluşturulur, sürüm etiketiyle kayıt deposuna gönderilir.
5. İmaj test ortamına dağıtılır; entegrasyon testleri çalışır.
6. Onaydan sonra (ya da otomatik olarak) üretime dağıtılır.
7. Dağıtım sonrası izleme kontrol edilir; sorun varsa önceki sürüme dönülür.

Yaygın CI/CD araçları arasında GitHub Actions, GitLab CI/CD, Jenkins, Azure Pipelines, AWS CodePipeline ve Google Cloud Build bulunur.

Kod olarak altyapı

DevOps yaklaşımında altyapı da kod gibi yönetilir. Sanal ağ, sunucu, veritabanı ve güvenlik kuralları **Terraform**, AWS CloudFormation ya da Azure Bicep gibi araçlarla dosyalarda tanımlanır. Bu dosyalar Git'te tutulur, değişiklikler incelenir ve CI/CD hattıyla uygulanır. Kazanımlar:

- Aynı ortam tekrar tekrar, aynı şekilde kurulabilir.
- Altyapıdaki her değişikliğin geçmişi görülür.
- Felaket durumunda ortam kod üzerinden yeniden oluşturulabilir.
- Konsolda elle yapılan, kimsenin hatırlamadığı değişiklikler azalır.

Örnek vaka

Konya'da bir tarım makineleri üreticisinin bayi sipariş portalı, ayda bir, cumartesi gecesi yapılan büyük güncellemelerle yayınlanıyor. Her güncellemede bir şey bozuluyor, pazartesi sabahı bayiler arıyor, ekip hatayı aramak için bir aylık değişiklik yığınının içinde kayboluyor.

Ekip altı ay içinde şu değişiklikleri yapıyor:

1. Bütün kod ve yapılandırma tek bir Git deposunda toplanıyor; her değişiklik birleştirme isteğiyle ve bir kişinin incelemesiyle ana dala alınıyor.
2. Her birleştirme isteğinde testler otomatik çalışıyor.
3. Uygulama konteynere taşınıyor ve yönetilen bir Kubernetes kümesinde çalıştırılıyor. Güncellemeler pod'ları kademeli değiştirerek, kesintisiz yapılıyor.
4. Aylık büyük güncelleme yerine haftada birkaç küçük güncelleme yapılıyor.
5. Hata oranı ve yanıt süresi için panolar ve uyarılar kuruluyor; bir güncellemeden sonra hata oranı artarsa önceki sürüme dönülüyor.
6. Her olaydan sonra kısa, suçlamasız bir değerlendirme yazılıyor.

Güncellemeler sıradan bir iş hâline geliyor; bir hata çıktığında hangi küçük değişiklikten geldiği kolayca bulunuyor.

Uygulama

1. Bilgisayarına Git kur. Bir deneme deposu oluştur; Modül 8'deki Dockerfile ve index.html dosyalarını ekle ve ilk kaydı (commit) yap.
2. Yeni bir dal aç, index.html'de küçük bir değişiklik yap, kaydet ve ana dala birleştir. `git log` ile geçmişi incele.
3. Çalıştığın ya da tanıdığın bir şirketin yazılım yayınlama sürecini yukarıdaki yedi aşamalı CI/CD hattıyla karşılaştı. Hangi aşamalar elle yapılıyor?
4. İstersen minikube ya da kind gibi bir araçla yerel deneme kümesi kur, yukarıdaki Deployment dosyasını uygula ve `kubectl get pods` ile üç pod'u gör.

Sık yapılan hatalar

- Küçük ve basit bir uygulama için ihtiyaç olmadan Kubernetes kurmak; yönetim yükünü gereksiz yere artırmak.
- Kubernetes Secret'lerini şifreli sanmak ve ek önlem almamak.
- Kümede kaynak sınırları (işlemci ve bellek istek ve limitleri) tanımlamamak; bir uygulamanın düğümü tüketmesi.
- DevOps'u bir araç satın almak ya da bir kişiyi "DevOps" unvanıyla işe almak olarak görmek.
- CI/CD hattında testleri atlamak ya da kırmızıya dönen testleri görmezden gelmek.
- Altyapıyı kodla tanımladıktan sonra konsoldan elle değiştirmek; kod ile gerçek durumun ayrışması.

Kendini dene

1. Kubernetes'te çalıştırılan en küçük birim nedir?
2. Değişen pod'ların önünde sabit bir adres sunan Kubernetes nesnesi hangisidir?
3. Kubernetes'in "istenen durum" yaklaşımını kısaca açıkla.
4. Sürekli teslim ile sürekli dağıtım arasındaki fark nedir?
5. Bir pod'un "ImagePullBackOff" durumunda olması genellikle neyi gösterir?

Buluta geçiş, yedekleme, felaket kurtarma ve veri konumu

Buluta geçiş bir hafta sonu işi değildir. Uygulamaları tanımayı, önceliklendirmeyi, test etmeyi, kullanıcıları hazırlamayı ve geri dönüş yolunu baştan planlamayı gerektiren bir projedir. Bu modülde geçişin aşamalarını, yedekleme ve felaket kurtarmanın temel kavramlarını ve kişisel verinin nerede tutulduğunun KVKK açısından neden önemli olduğunu ele alıyoruz.

Geçişin aşamaları

Sağlayıcıların geçiş rehberleri farklı adlar kullansa da bir buluta geçiş projesi genellikle şu aşamalardan oluşur:

1. **Değerlendirme:** Mevcut sistemlerin envanteri çıkarılır. Her uygulamanın ne iş yaptığı, kimin kullandığı, hangi sunucularda çalıştığı, ne kadar kaynak tükettiği, hangi diğer sistemlere bağlı olduğu ve lisans durumu belirlenir. Modül 1'deki envanter listesi bu aşamanın başlangıcıdır.
2. **Planlama:** Her uygulama için geçiş stratejisi seçilir, uygulamalar dalgalara (wave) ayrılır, hedef mimari ve maliyet tahmini hazırlanır.
3. **Temel hazırlık (landing zone):** Bulutta hesap ve abonelik yapısı, ağ, kimlik, güvenlik kuralları, loglama, etiketleme ve bütçeler kurulur. Uygulamalar ancak bu temel hazır olunca taşınır.
4. **Pilot:** Kritik olmayan, bağımlılığı az bir uygulama ilk olarak taşınır. Ekip deneyim kazanır, süreçteki eksikler görülür.
5. **Taşıma dalgaları:** Uygulamalar planlanan sırayla taşınır, test edilir ve kullanıma açılır.
6. **Optimizasyon:** Taşınan sistemler izlenir; boyutlandırma, maliyet ve güvenlik ayarları iyileştirilir.

Envanter çıkarma ve taşıma için sağlayıcıların araçları vardır: Azure Migrate, AWS Application Migration Service, AWS Database Migration Service ve Google Cloud'un Migrate to Virtual Machines hizmeti bunlardan bazılarıdır.

Geçiş stratejileri

Her uygulama aynı şekilde taşınmaz. Sık kullanılan sınıflandırma şöyledir:

- **Yeniden barındırma (rehost, "kaldır ve taşı"):** Sunucu olduğu gibi bulutta sanal makineye taşınır. En hızlı yoldur; uygulama değişmez ama bulutun avantajlarından da az yararlanır.
- **Platform değiştirme (replatform):** Küçük değişikliklerle yönetilen hizmetlere geçilir. Örneğin sanal makinedeki veritabanı yerine yönetilen veritabanı hizmeti kullanılır.
- **Yeniden yapılandırma (refactor):** Uygulama buluta uygun biçimde yeniden tasarlanır; konteynerler, sunucusuz hizmetler kullanılır. En çok emek isteyen ama en çok fayda sağlayabilen yoldur.
- **Yeniden satın alma (repurchase):** Uygulama bırakılıp SaaS bir ürüne geçilir. Örneğin şirket içi e-posta sunucusu yerine Microsoft 365 ya da Google Workspace.
- **Emekliye ayırma (retire):** Artık kullanılmayan uygulama kapatılır. Envanter çıkarıldığında, kimsenin kullanmadığı sistemler bulunması şaşırtıcı derecede yaygındır.
- **Yerinde tutma (retain):** Uygulama şimdilik taşınmaz; teknik, hukuki ya da maliyet nedeniyle yerinde kalır.

Geçiş günü: kesme planı

Bir uygulamanın eski sistemden yeni sisteme geçtiği ana **kesme (cutover)** denir. Kesme planında şunlar yazılı olmalıdır:

- Kesmenin tarihi, saati ve beklenen kesinti süresi; kullanıcılara ne zaman duyurulacağı.
- Verinin son kez hangi yöntemle eşitleneceği.
- DNS değişikliği ve TTL'in önceden düşürülmesi (Modül 4).
- Kesmeden sonra yapılacak testler ve testi kimin onaylayacağı.
- **Geri dönüş planı:** Bir sorun çıkarsa eski sisteme nasıl ve hangi süre içinde dönüleceği. Eski sistem, yeni sistem bir süre sorunsuz çalışana kadar kapatılmaz.

Yedekleme

Bulut sağlayıcısı altyapıyı korur ama senin verini yanlış silmeye, bozulmaya ya da fidye yazılımına karşı korumak senin işindir (Modül 6). Bu yüzden bulutta da yedek alınır.

Yedekleme türleri:

- **Tam yedek:** Bütün verinin kopyası. En kolay geri yüklenen, en çok yer kaplayan türdür.
- **Artımlı yedek (incremental):** Son yedekten (tam ya da artımlı) bu yana değişen veri.
- **Fark yedeği (differential):** Son tam yedekten bu yana değişen veri.

Yaygın kabul gören **3-2-1 kuralı:** verinin en az **3** kopyası olsun, bunlar en az **2** farklı ortamda dursun ve en az **1** kopya başka bir konumda (örneğin başka bir bölgede ya da başka bir hesapta) olsun. Fidye yazılımlarına karşı ayrıca **değiştirilemez (immutable)** yedek önerilir: belirli bir süre boyunca kimsenin, yönetici hesabı ele geçirilse bile, silemediği ya da değiştiremediği yedek.

Bulutta yedekleme için AWS Backup, Azure Backup ve Google Cloud Backup and DR Service gibi hizmetler kullanılır. Yedeklerin, üretim hesabından ayrı bir hesapta ya da abonelikte, farklı yetkilerle tutulması önemlidir: üretim hesabını ele geçiren biri yedeklere de ulaşamamalıdır.

Geri yüklemesi denenmemiş yedek, güvenilir yedek değildir. Düzenli aralıklarla örnek bir dosyayı, bir veritabanını ya da bütün bir sunucuyu yedekten geri yükleyip süreyi ölç.

RPO ve RTO

Felaket kurtarma planı iki soruyla başlar:

- **RPO (Recovery Point Objective, kurtarma noktası hedefi):** En fazla ne kadarlık veri kaybını kabul edebilirsin? RPO 24 saat ise gece alınan yedek yeterlidir. RPO 15 dakika ise çok daha sık yedek ya da sürekli kopyalama gerekir.
- **RTO (Recovery Time Objective, kurtarma süresi hedefi):** Sistem en geç ne kadar sürede yeniden çalışmalıdır? RTO 2 gün ise yedekten yeniden kurmak yeterli olabilir. RTO 1 saat ise hazırda bekleyen bir ortam gerekir.

Bu değerleri teknik ekip tek başına belirlemez; iş birimleriyle birlikte, kesintinin işe maliyeti düşünülerek belirlenir. RPO ve RTO küçüldükçe çözüm pahalılaşır.

Felaket kurtarma stratejileri

Maliyet ve hız sırasıyla dört temel yaklaşım vardır:

1. **Yedekle ve geri yükle:** Yedekler başka bir bölgede tutulur; felakette ortam yedekten yeniden kurulur. En ucuz, en yavaş yoldur.
2. **Pilot ışığı (pilot light):** Kurtarma bölgesinde yalnız en kritik parçalar (örneğin veritabanı kopyası) sürekli çalışır; diğer kaynaklar felakette hızla açılır.
3. **Sıcak bekleme (warm standby):** Kurtarma bölgesinde ortamın küçültülmüş ama çalışan bir kopyası durur; felakette büyütülür.
4. **Çoklu etkin (multi-site active/active):** Sistem birden fazla bölgede aynı anda tam kapasiteyle çalışır. En hızlı, en pahalı yoldur.

Kod olarak altyapı (Modül 9), felaket kurtarmayı kolaylaştırır: ortamı tarif eden kod hazırsa yeni bir bölgede aynı ortam hızla kurulabilir. Azure Site Recovery ve AWS Elastic Disaster Recovery gibi hizmetler sunucuların başka bir bölgeye sürekli kopyalanmasını sağlar. Felaket kurtarma planı da yedek gibi düzenli olarak tatbikatla denenmelidir.

Yüksek erişilebilirlik ile **felaket kurtarma** aynı şey değildir. Yüksek erişilebilirlik, tek bir sunucu ya da veri merkezi arızasında sistemin kesintisiz çalışmasıdır (örneğin birden fazla erişilebilirlik alanı). Felaket kurtarma, bütün bir bölgenin ya da sistemin kaybı gibi büyük olaylardan sonra geri dönmektir.

KVKK açısından veri konumu

Türkiye'de kişisel verilerin işlenmesi 6698 sayılı Kişisel Verilerin Korunması Kanunu (KVKK) ile düzenlenir. Buluta geçişte şu sorular hukuk ve uyum birimiyle birlikte ele alınmalıdır:

- **Hangi veriler kişisel veri?** Müşteri adı, telefon, adres, TC kimlik numarası, çalışan bilgileri. Sağlık verisi gibi **özel nitelikli kişisel veriler** daha sıkı korumaya tabidir.
- **Roller:** Veriyi işleme amacını ve yöntemini belirleyen kurum **veri sorumlusudur**. Bulut sağlayıcısı genellikle kurum adına veri işleyen konumundadır. Sorumluluk bulut sağlayıcısına geçmez; veri sorumlusu, veri işleyenle birlikte gerekli teknik ve idari tedbirlerden sorumludur.
- **Veri nerede duruyor?** Seçtiğin bölge verinin hangi ülkedeki veri merkezinde tutulduğunu belirler. Yurt dışındaki bir bölgede kişisel veri saklamak, yurt dışına aktarım sayılabilir. Ayrıca destek, yedek ya da kopyalama için verinin başka ülkelere gidip gitmediği sözleşmeden ve sağlayıcı belgelerinden kontrol edilir.
- **Yurt dışına aktarım koşulları:** KVKK'nın yurt dışına aktarımı düzenleyen 9. maddesi 2024'te değiştirildi; aktarımın hangi koşullarda ve hangi güvencelerle (örneğin Kişisel Verileri Koruma Kurulu'nun belirlediği standart sözleşmeler) yapılabileceği yeniden tanımlandı. Ayrıntılar ve güncel ikincil düzenlemeler için Kişisel Verileri Koruma Kurumu'nun yayımladığı rehberlere ve kurumunun hukuk birimine başvurun.
- **Sektörel kurallar:** Bankacılık, sağlık, kamu ve telekomünikasyon gibi bazı alanlarda verinin yurt içinde tutulmasını ya da belirli güvenlik önlemlerini isteyen ek düzenlemeler bulunabilir. Kamu kurumları için Cumhurbaşkanlığı Dijital Dönüşüm Ofisi'nin Bilgi ve İletişim Güvenliği Rehberi de dikkate alınır.
- **Aydınlatma ve kayıt:** Veri işleme süreçleri değiştiğinde aydınlatma metinlerinin ve kurumun kişisel veri işleme envanterinin güncellenmesi gerekebilir.

Teknik ekibin işi hukuki kararı vermek değil, verinin nerede durduğunu, nereye gittiğini ve nasıl korunduğunu açıkça belgelemektir.

Örnek vaka

Adana'da 120 yataklı özel bir hastane, hasta randevu sistemini, personel e-postasını ve tıbbi görüntü arşivini değerlendirmeye alıyor. BT ekibi hukuk birimiyle birlikte şu planı çıkarıyor:

1. **Envanter:** 14 uygulama listeleniyor. Bunlardan ikisinin yıllardır kullanılmadığı görülüyor; bu ikisi emekliye ayrılıyor.
2. **E-posta:** Şirket içi e-posta sunucusu SaaS bir ürüne taşınıyor (yeniden satın alma). Sözleşmedeki veri konumu ve aktarım koşulları hukuk birimince inceleniyor.
3. **Randevu web sitesi:** Hasta adı ve telefonu içeriyor. Uygun bölge ve sözleşme koşulları değerlendirildikten sonra PaaS üzerinde çalıştırılmasına karar veriliyor (platform değiştirme).
4. **Tıbbi görüntü arşivi ve hasta kayıt sistemi:** Özel nitelikli kişisel veri içeriyor, sektörel kurallar ve büyük veri hacmi nedeniyle şimdilik hastanenin veri merkezinde kalıyor (yerinde tutma). Bu sistemlerin şifreli yedeklerinin nerede tutulacağı ayrıca değerlendiriliyor.
5. **Yedekleme ve felaket kurtarma:** Randevu sistemi için RPO 1 saat, RTO 4 saat olarak belirleniyor. Yedekler ayrı bir hesapta, değiştirilemez olarak tutuluyor. Her üç ayda bir geri yükleme tatbikatı yapılıyor.
6. **Pilot ve kesme:** İlk olarak iç duyuru panosu taşınıyor. Randevu sistemi, hafta sonu gece yarısı, DNS TTL'i önceden düşürülerek ve geri dönüş planıyla taşınıyor; eski sistem iki hafta daha hazırda tutuluyor.

Sonuç bilinçli olarak seçilmiş bir hibrit yapıdır: her sistem, riskine ve ihtiyacına uygun yerde çalışıyor.

Uygulama

Kendi iş yerin ya da hayalî bir KOBİ için tek sayfalık bir geçiş ve kurtarma planı taslağı hazırla:

1. Beş uygulama seç. Her biri için kullanıcı sayısı, bağımlı olduğu sistemler ve kişisel veri içerip içermediğini yaz.
2. Her uygulama için bir geçiş stratejisi seç ve nedenini bir cümleyle açıkla.
3. En kritik iki uygulama için RPO ve RTO değeri öner; bu değerleri bir iş birimi yöneticisiyle konuşarak kontrol et.
4. Bu değerlere uygun yedekleme sıklığını ve felaket kurtarma stratejisini seç.
5. Kişisel veri içeren uygulamalar için verinin hangi bölgede duracağını, kopyaların nereye gideceğini ve hukuk birimine sorulacak soruları listele.
6. Pilot olarak taşınacak uygulamayı ve geri dönüş planını yaz.

Sık yapılan hatalar

- Envanter çıkarmadan taşımaya başlamak; geçişten sonra unutulmuş bir bağımlılığın ortaya çıkması.
- Bütün uygulamaları tek seferde taşımaya çalışmak.
- Geri dönüş planı yapmamak ve eski sistemi kesmenin hemen ardından kapatmak.
- Yedekleri üretimle aynı hesapta ve aynı yetkilerle tutmak.
- Yedekten geri yüklemeyi hiç denememek.
- RPO ve RTO'yu iş birimlerine sormadan, yalnız teknik ekip olarak belirlemek.
- Bölge seçimini yalnız fiyat ve hıza göre yapmak; kişisel verinin konumunu ve aktarım koşullarını hukuk birimine danışmamak.

Kendini dene

1. Bir sunucuyu hiç deęiřtirmeden bulutta sanal makineye taşıma stratejisine ne ad verilir?
2. 3-2-1 yedekleme kuralını açıkla.
3. RPO ile RTO arasındaki fark nedir?
4. Kurtarma bölgesinde ortamın küçültölmüş ama çalışan bir kopyasının tutulduęu felaket kurtarma stratejisinin adı nedir?
5. KVKK açısından bulut sağlayıcısı genellikle hangi konumdadır ve kişisel veriden sorumluluk ona geçer mi?

Kendini dene cevapları

Modül 1: 1) RAM geçici ve hızlıdır, sunucu kapanınca içeriği silinir; disk kalıcı depolamadır. 2) Nesne depolama. 3) Hayır; RAID silme işlemini de bütün disklerle yansıtır, yalnız disk arızasına karşı korur. Dosya ancak yedekten kurtarılır. 4) Tip 1 doğrudan donanım üzerinde, tip 2 bir işletim sisteminin üstünde uygulama olarak çalışır. 5) Sermaye gideri (CapEx).

Modül 2: 1) İsteğe bağlı self servis, geniş ağ erişimi, kaynak havuzu, hızlı esneklik, ölçülen hizmet. 2) IaaS. 3) SaaS. 4) Hibrit bulut. 5) Bir veri merkezindeki arızanın uygulamayı durdurmasını önlemek (yüksek erişilebilirlik).

Modül 3: 1) Amazon EC2, Azure Virtual Machines, Compute Engine. 2) Kaynak grubu (resource group). 3) Proje (project). 4) Arşiv sınıfları daha ucuzdur; sık erişilmeyen veri için yüksek performanslı sınıfa ödeme yapılmaz. 5) Kod olarak altyapı (Infrastructure as Code, IaC).

Modül 4: 1) 65.536. 2) NAT ağ geçidi üzerinden. 3) A kaydı. 4) Katman 4 IP ve porta göre dağıtır, içeriğe bakmaz; katman 7 HTTP isteğinin içeriğine (yol, alan adı) göre yönlendirme yapabilir. 5) Ofis ve bulut ağlarının adres aralıkları çakışmamalıdır.

Modül 5: 1) df -h. 2) apt. 3) 640. 4) Active Directory Domain Services (AD DS). 5) Test-NetConnection.

Modül 6: 1) Veri, kimlikler (kullanıcı hesapları) ve erişim yetkileri. 2) Her kimliğe işini yapması için gereken en az yetkinin, en dar kapsamda verilmesi. 3) Kök kullanıcı (root user); ilk iş MFA'yı açmaktır. 4) Gizli bilgi yönetim hizmetinde (AWS Secrets Manager, Azure Key Vault, Secret Manager) ya da uygulamaya atanan rol veya yönetilen kimlikle. 5) AWS CloudTrail.

Modül 7: 1) Bağlı diskler, anlık görüntüler ve ayrılmış genel IP adresleri. 2) Hayır, yalnız haber verir. 3) Harcamanın proje, ortam, birim ve sahip bazında görülmesini ve artışların nedeninin bulunmasını sağlar. 4) Kesintiye dayanıklı işler: toplu veri işleme, test ortamları, görüntü işleme. 5) Gereğinden büyük kaynak için taahhüt almak israfı taahhüt süresi boyunca kilitler.

Modül 8: 1) Sanal makine kendi tam işletim sistemini taşır; konteyner ana makinenin çekirdeğini paylaşır, daha hafiftir ve daha hızlı başlar. 2) İmaj salt okunur şablondur; konteyner o imajın çalışan örneğidir. 3) Dockerfile. 4) Birimde (volume) ya da harici bir depolama veya veritabanı hizmetinde. 5) Parola imaj katmanlarında kalır ve imaja erişen herkes onu görebilir.

Modül 9: 1) Pod. 2) Service. 3) Kullanıcı sonucun ne olması gerektiğini tanımlar; Kubernetes gerçek durumu sürekli bununla karşılaştırır ve farkı kendiliğinden kapatır. 4) Sürekli teslimde üretime alma bir onayla yapılır; sürekli dağıtımda testlerden geçen değişiklik onaysız, otomatik olarak üretime alınır. 5) İmaj adının ya da etiketinin yanlış olduğunu veya kayıt deposuna erişim yetkisi bulunmadığını.

Modül 10: 1) Yeniden barındırma (rehost, kaldır ve taşı). 2) Verinin en az 3 kopyası, en az 2 farklı ortamda, en az 1'i başka bir konumda tutulur. 3) RPO kabul edilebilir en fazla veri kaybını, RTO sistemin en geç ne kadar sürede yeniden çalışması gerektiğini gösterir. 4) Sıcak bekleme (warm standby). 5) Genellikle veri işleyen konumundadır; veri sorumlusunun sorumluluğu ona geçmez.

Sözlük

Alt ağ (subnet): Bir ağın, kendi adres aralığına sahip daha küçük parçası.

Anlık görüntü (snapshot): Bir diskin ya da sanal makinenin belirli bir andaki durumunun kaydı.

Bölge (region): Bulut sağlayıcısının belirli bir coğrafyadaki veri merkezleri grubu.

CI/CD: Kod değişikliklerinin otomatik olarak test edilmesi (sürekli entegrasyon) ve yayına hazırlanması ya da yayınlanması (sürekli teslim ve dağıtım).

CIDR: IP adres aralığını adres ve eğik çizgiden sonra yazılan ağ biti sayısı ile gösterme biçimi (10.0.0.0/16 gibi).

DevOps: Geliştirme ve operasyon ekiplerinin ortak sorumluluk, otomasyon ve sık küçük değişikliklerle birlikte çalışma kültürü.

DNS: Alan adlarını IP adreslerine çeviren sistem.

Docker: Konteyner imajı oluşturmak ve konteyner çalıştırmak için kullanılan araç.

Dockerfile: Bir konteyner imajının nasıl oluşturulacağını tarif eden dosya.

En az yetki ilkesi: Her kimliğe yalnız işini yapması için gereken yetkinin verilmesi.

Erişilebilirlik alanı (Availability Zone): Bir bölge içindeki, ayrı güç ve soğutma altyapısına sahip veri merkezi ya da merkezler grubu.

Etiket (tag): Kaynaklara eklenen, maliyet ve yönetim için kullanılan anahtar-değer bilgisi.

Felaket kurtarma: Büyük bir arıza ya da kayıptan sonra sistemleri ve veriyi yeniden çalışır hâle getirme planı ve süreci.

Genel bulut: Sağlayıcıya ait, çok sayıda müşteriye hizmet veren bulut altyapısı.

Hibrit bulut: Şirket içi ya da özel bulut ile genel bulutun birbirine bağlı kullanılması.

Hipervizör: Fiziksel makinenin kaynaklarını sanal makinelere bölen yazılım.

IaaS: Sanal makine, disk ve ağın hizmet olarak sunulduğu bulut modeli.

IAM: Kimlik ve erişim yönetimi; kimin hangi kaynakta ne yapabileceğini belirleyen sistem.

İmaj (image): Konteynerin salt okunur şablonu.

Kod olarak altyapı (IaC): Altyapının metin dosyalarıyla tanımlanıp otomatik kurulması.

Konteyner: Uygulamayı bağımlılıklarıyla birlikte, ana makinenin çekirdeğini paylaşarak yalıtılmış çalıştıran paket.

Kubernetes: Konteynerleri birden fazla sunucuda dağıtan, ölçekleyen ve iyileştiren açık kaynaklı orkestrasyon platformu.

MFA: Çok faktörlü kimlik doğrulama; parolanın yanında ikinci bir doğrulama faktörü istenmesi.

NAT: Özel adresli cihazların ortak bir genel adres üzerinden internete çıkmasını sağlayan adres çevirisi.

Nesne depolama: Dosyaların anahtar ve ek bilgilerle nesne olarak saklandığı depolama türü.

PaaS: İşletim sistemi ve çalışma ortamının sağlayıcıya yönetildiği, kullanıcının uygulamasını getirdiği bulut modeli.

Paylaşılan sorumluluk modeli: Bulutta güvenlik görevlerinin sağlayıcı ile müşteri arasında paylaşılması.

Pod: Kubernetes'te bir ya da birkaç konteyner içeren en küçük çalıştırma birimi.

RAID: Birden fazla diski hız ya da arızaya dayanıklılık için birlikte kullanma yöntemi.

RPO: Kabul edilebilir en fazla veri kaybı süresi.

RTO: Sistemin bir kesintiden sonra en geç ne kadar sürede yeniden çalışması gerektiği.

SaaS: Hazır yazılımın internet üzerinden hizmet olarak kullanıldığı model.

Sanal makine (VM): Hipervizör üzerinde, kendi işletim sistemiyle çalışan yazılımsal bilgisayar.

Sunucusuz (serverless): Sunucu yönetmeden kodun ya da konteynerin olaylara göre çalıştırıldığı hizmet türü.

Veri sorumlusu: KVKK'ya göre kişisel verinin işleme amacını ve yöntemini belirleyen kişi ya da kurum.

VPC / VNet: Bulut sağlayıcısında müşteriye ayrılmış yalıtılmış sanal ağ.

Yük dengeleyici: Gelen istekleri birden fazla sunucuya dağıtan ve sağlık kontrolü yapan hizmet.

Sonraki adım

Eğitimi bitirdiysen bilgini ölç: Daphne Kariyer'de Bulut Sistemleri, AWS, Microsoft Azure, Google Cloud, DevOps, Docker, Kubernetes ve Linux Sistem Yönetimi testleri ücretsizdir. Her test 20 sorudan oluşur. Önce Bulut Sistemleri testiyle genel düzeyini gör; ardından bu eğitimde en çok ilgini çeken alanın testine geç. 55 ve üzeri puan alırsan isteğe bağlı olarak doğrulanabilir dijital sertifika alabilirsin.

daphnekariyer.com/dijital-yetkinlikler/