Hibrit Bulut (Hybrid Cloud) ve Çoklu Bulut (Multi-Cloud) Tanımları

- Hibrit bulut, bir organizasyonun kendi özel bulutunu (private cloud) ve üçüncü taraf kamu bulutunu (public cloud) birleştirerek tek bir altyapı (infrastructure) oluşturmasını sağlar.
- Çoklu bulut, farklı hizmet sağlayıcılarından (service providers) çeşitli bulut modellerini (cloud models) benimseyen bir stratejidir.

Kullanım Senaryoları (Use Cases)

- **Bulut Ölçeklendirme (Cloud Scaling)**: Örneğin, bir çiçek teslimat hizmeti, belirli tatillerde artan kullanıcı yükünü (user load) karşılamak için bulut kaynaklarını (cloud resources) kullanarak ölçeklenebilir.
- **Bileşik Bulut (Composite Cloud)**: Uygulamalar (applications) farklı bulut ortamları (cloud environments) arasında dağıtılarak, örneğin, Avrupa'daki kullanıcılar için bazı bileşenler (components) yerel sunucularda (on-premise) kalırken, Kuzey Amerika'daki kullanıcılar için diğer bileşenler bulut platformuna (cloud platform) taşınabilir.

Havayolu ve Seyahat Endüstrisi Örneği

- **Modernizasyon (Modernization)**: Havayolu şirketleri, mobil uygulamalar (mobile applications) geliştirerek kullanıcı deneyimlerini (user experiences) iyileştirmektedir.
- **Veri ve Yapay Zeka (Data and AI)**: Tarihsel veriler (historical data) kullanılarak, makine öğrenimi (machine learning) ile bakım (maintenance) sorunları önceden tahmin edilebilir.

Sonuç

- Hibrit çoklu bulut stratejisi, belirli bir bulut platformuna (cloud platform) bağımlılığı önlemek ve iş yüklerini (workloads) esnek bir şekilde yönetmek için benimsenmektedir.

Coach

Bu video, mikroservis mimarisinin (microservices architecture) uygulama geliştirme üzerindeki etkilerini ve bu yaklaşımın nasıl çalıştığını açıklamaktadır.

Mikroservis Mimarisi

- Mikroservis mimarisi, tek bir uygulamanın (application) birçok bağımsız ve dağıtılabilir küçük bileşen (components) veya hizmet (services) ile oluşturulmasıdır.
- Bu hizmetler, kendi yığınları (stacks) üzerinde çalışır ve API'ler (APIs), olay akışı (event streaming) ve mesaj aracıları (message brokers) aracılığıyla birbirleriyle iletişim kurar.

Geliştirici Çalışma Şekli

- Geçmişte, yazılımlar büyük monolitik uygulamalar (monolithic applications) olarak inşa edilirdi; bu, geliştiricilerin (developers) uzun süreler boyunca tek bir kod tabanı (code base) üzerinde çalışmasını gerektiriyordu.

- Günümüzde, geliştiriciler bağımsız küçük ekipler (independent teams) halinde çalışarak, mikroservisler (microservices) adı verilen daha küçük kod parçaları (code snippets) yazmaktadır.

Mikroservis Örneği: Dream Game

- Dream Game, kullanıcıların içerik (content) aramasını kolaylaştırmak için mikroservisler kullanmaktadır. Örneğin, içerik kataloğu (Content Catalog), arama fonksiyonu (Search Function) ve öneri (Recommendations) mikroservisleri birlikte çalışarak kullanıcı deneyimini (user experience) geliştirmektedir.
- Bu mikroservisler, hizmet keşfi (Service Discovery) ile birbirlerini bulur ve API'ler aracılığıyla iletişim kurar.

Sonuç

- Mikroservis yaklaşımı, geliştiricilerin uygulamaları (applications) hızlı bir şekilde yenilikçi hale getirmesine olanak tanır ve kullanıcıların (users) ilgi alanlarına odaklanmalarını sağlar.

Sunucusuz Bilişim

- Sunucusuz, geliştiricilerin (developers) uygulama yığınlarını (application stacks) yönetme sorumluluğunu bulut sağlayıcılarına (cloud providers) devrettiği bir yaklaşımdır. Bu, geliştiricilerin kod (code) ve iş mantığı (business logic) üzerinde daha fazla odaklanmasını sağlar.
- Sunucusuz, fiziksel veya sanal sunucuların (servers) olmadığını değil, altyapının (infrastructure) yönetiminin kullanıcıdan alındığını ifade eder.

Kaynak Yönetimi

- Sunucusuz ortam, uygulamalar için ihtiyaç duyulan kaynakları (resources) talep üzerine tahsis eder. Bu model, sunucuların (servers) önceden yapılandırılmasını veya yazılım yüklenmesini gerektirmez.
- Kod, yalnızca talep üzerine çalıştırılır ve gelen istek sayısına göre otomatik olarak ölçeklenir (scales).

Maliyet Avantajları

- Kullanıcılar, yalnızca kullandıkları kaynaklar için ödeme yapar (pay only for resources used), bu da sanal sunucularda (virtual servers) olduğu gibi boşta kalan kapasite (idle capacity) için ödeme yapmamalarını sağlar.

Uygulama Senaryoları

- Sunucusuz mimari, kısa süreli (short-running) ve durumsuz (stateless) işlevler (functions) için uygundur. Örneğin, metin dosyalarını çeviren ve bulut tabanlı depolama hizmetine (cloud storage) kaydeden bir uygulama senaryosu (application scenario) verilebilir.

- Sunucusuz mimari, veri işleme (data processing), IoT (Internet of Things), mikro hizmetler (microservices) ve mobil arka uçlar (mobile backends) gibi kullanım durumları (use cases) için idealdir.

Zorluklar

- Uzun süreli işlemler (long-running processes) için sunucusuz mimari, geleneksel sunucu ortamlarını (traditional server environments) yönetmekten daha karmaşık ve maliyetli olabilir.
- Sunucusuz mimariler, belirli bir bulut sağlayıcısına (cloud provider) bağımlı olabilir ve bu da potansiyel olarak tedarikçi kilitlenmesine (vendor lock-in) yol açabilir.

Coach

Bu video, bulut yerel (cloud native) uygulamaların (applications) temel özelliklerini ve avantajlarını açıklamaktadır.

Bulut Yerel Uygulamalar

- Bulut yerel uygulamalar, başlangıçtan itibaren yalnızca bulut ortamında (cloud environment) çalışacak şekilde geliştirilmiş uygulamalardır. Mevcut bir uygulama, bulut yerel ilkeler (cloud native principles) ile yeniden yapılandırılabilir.
- Bu uygulamalar, bağımsız olarak ölçeklenebilen (scalable) ve otomasyon (automation) süreçleriyle güncellenebilen mikro hizmetlerden (microservices) oluşur.

Mikro Hizmetler

- Her mikro hizmet, uygulamanın (application) bir parçasını temsil eder ve bağımsız olarak geliştirilip dağıtılabilir. Örneğin, bir seyahat web sitesinde (travel website) uçuşlar, oteller ve araçlar gibi her konu kendi mikro hizmeti olarak ele alınabilir.
- Bu bağımsızlık, kullanıcı deneyimini (user experience) etkilemeden sık sık güncellemeler (updates) yapılmasına olanak tanır.

Geliştirme İlkeleri

- Geliştiriciler, bulut yerel uygulamalar oluştururken belirli geliştirme ilkelerine (development principles) uymalıdır:
    - Mikro hizmet mimarisi (microservices architecture) izlenmeli ve uygulamalar tek işlevli mikro hizmetlere (single-function microservices) bölünmelidir.
    - Maksimum esneklik (flexibility), ölçeklenebilirlik (scalability) ve taşınabilirlik (portability) için konteynerler (containers) kullanılmalıdır.
    - Kullanıcı geri bildirimine (user feedback) dayalı hızlı, yinelemeli güncellemeler (iterative updates) için Agile yöntemler (Agile methods) benimsenmelidir.

Sonuç

- Bulut yerel uygulamalar, yenilik (innovation) ve iş çevikliği (business agility) sağlarken, geliştiricilere uygulamaları ölçeklendirme ve iyileştirme konusunda büyük avantajlar sunar.

DevOps Nedir?

- DevOps, yazılım geliştirme (software development) ve IT operasyonları (IT operations) arasındaki işbirliğini (collaboration) artırmayı amaçlayan bir yaklaşımdır. Bu, yazılımın daha hızlı ve güvenilir bir şekilde teslim edilmesini sağlar.
- DevOps, sürekli entegrasyon (continuous integration), sürekli dağıtım (continuous delivery) ve sürekli izleme (continuous monitoring) gibi uygulamaları içerir.

Sürekli Entegrasyon ve Dağıtım

- Sürekli entegrasyon, geliştiricilerin (developers) kod değişikliklerini (code changes) sık sık birleştirmesini sağlar. Bu, hataların (bugs) erken tespit edilmesine yardımcı olur.
- Sürekli dağıtım, her yeni kod değişikliğinin otomatik olarak üretim ortamına (production environment) aktarılmasını sağlar. Bu, yazılımın (software) daha hızlı bir şekilde güncellenmesini mümkün kılar.

İzleme ve Geri Bildirim

- DevOps, uygulamaların performansını (performance) ve kullanılabilirliğini (availability) izlemek için araçlar (tools) kullanır. Bu, sorunların (issues) hızlı bir şekilde tespit edilmesine ve çözülmesine olanak tanır.
- Kullanıcı geri bildirimleri (user feedback), yazılım geliştirme sürecine (development process) entegre edilerek, ürünün (product) sürekli olarak iyileştirilmesini sağlar.

Sonuç

- DevOps, yazılım geliştirme süreçlerini (development processes) daha verimli hale getirirken, işbirliğini artırır ve müşteri memnuniyetini (customer satisfaction) yükseltir. Bu yaklaşım, modern yazılım geliştirme (modern software development) için kritik bir öneme sahiptir.

Uygulama Modernizasyonu

- Uygulama modernizasyonu, eski sistemlerin (legacy systems) güncellenmesi ve yeni teknolojilere (new technologies) uyum sağlaması sürecidir. Bu, organizasyonların dijital dönüşümlerini (digital transformations) hızlandırmalarına yardımcı olur.
- Modernizasyon, uygulamaların daha esnek (flexible) ve müşteri ihtiyaçlarına (customer needs) daha duyarlı hale gelmesini sağlar.

Mikroservis Mimarisi

- Uygulama modernizasyonu, mikroservis mimarisi (microservices architecture) ile desteklenir. Bu mimari, uygulamaların küçük, bağımsız hizmetler (independent services) olarak yapılandırılmasını sağlar.

- Mikroservisler, daha hızlı geliştirme (faster development) ve dağıtım (deployment) süreçleri sunar, bu da organizasyonların pazara (market) daha hızlı yanıt vermesine olanak tanır.

Bulut Altyapısı

- Bulut bilişim, uygulama modernizasyonunun (application modernization) temel bileşenlerinden biridir. Bulut altyapısı (cloud infrastructure), uygulamaların dinamik olarak ölçeklenmesini (scaling) ve yönetilmesini kolaylaştırır.
- Bulut, organizasyonların kaynakları (resources) daha verimli kullanmalarını sağlar ve maliyetleri (costs) düşürür.

DevOps ve Çalışma Yöntemleri

- DevOps, uygulama modernizasyonu sürecinde önemli bir rol oynar. Geliştiriciler (developers) ve operasyon ekipleri (operations teams) arasındaki işbirliğini artırarak, daha hızlı ve güvenilir yazılım teslimatı (software delivery) sağlar.
- Agile yöntemler (Agile methods) ve sürekli entegrasyon (continuous integration) uygulamaları, modernizasyon sürecini destekler.

Sonuç

- Uygulama modernizasyonu, organizasyonların rekabet avantajı (competitive advantage) elde etmelerine yardımcı olur. Bu süreç, bulut bilişim ve mikroservis mimarisi ile birleştiğinde, daha hızlı, esnek ve müşteri odaklı uygulamalar (customer-centric applications) geliştirilmesini sağlar.