AWS'de Mikro Hizmetlerin Uygulanması

İlk Yayın Tarihi 1 Aralık 2016

Güncellendi 9 Kasım 2021

Bu sürüm arşivlenmiştir.

Bu belgenin en son sürümü için bkz.

https://docs.aws.amazon.com/whitepapers/latest/microservices-on-aws/microservices-on-aws.pdf

Bildirimler

Müşteriler, bu belgedeki bilgileri kendi bağımsız değerlendirmeleriyle yapmaktan sorumludur. Bu belge: (a) yalnızca bilgilendirme amaçlıdır, (b) önceden bildirilmeksizin değiştirilebilen mevcut AWS ürün tekliflerini ve uygulamalarını temsil eder ve (c) AWS ve bağlı kuruluşları, tedarikçileri veya lisans verenlerinden herhangi bir taahhüt veya güvence oluşturmaz.AWS ürünleri veya hizmetleri, açık veya zımni herhangi bir garanti, beyan veya koşul olmaksızın "olduğu gibi" sağlanır. AWS'nin müşterilerine karşı sorumlulukları ve yükümlülükleri AWS sözleşmeleri tarafından kontrol edilir ve bu belge AWS ile müşterileri arasındaki herhangi bir sözleşmenin parçası değildir veya bu sözleşmeyi değiştirmez.

İçindekiler

Özet

Mikro hizmetler, dağıtım döngülerini hızlandırmak, yenilikçiliği ve sahiplenmeyi teşvik etmek, yazılım uygulamalarının sürdürülebilirliğini ve ölçeklenebilirliğini geliştirmek ve ekiplerin bağımsız çalışmasına yardımcı olan çevik bir yaklaşım kullanarak yazılım ve hizmet sunan kuruluşları ölçeklendirmek için oluşturulan yazılım geliştirmeye yönelik mimari ve organizasyonel bir yaklaşımdır. Mikro hizmetler yaklaşımı ile yazılım, bağımsız olarak dağıtılabilen iyi tanımlanmış uygulama programlama arayüzleri (API'ler) üzerinden iletişim kuran küçük hizmetlerden oluşur. Bu hizmetlerin sahibi küçük otonom ekiplerdir.
Bu çevik yaklaşım, kuruluşunuzu başarılı bir şekilde ölçeklendirmenin anahtarıdır.

AWS müşterileri mikro hizmetler oluştururken üç ortak model gözlemlenmiştir: API odaklı, olay odaklı ve veri akışı. Bu teknik dokümanda her üç yaklaşım da tanıtılmakta ve mikro hizmetlerin ortak özellikleri özetlenmekte, mikro hizmet oluşturmanın temel zorlukları tartışılmakta ve ürün ekiplerinin bu zorlukların üstesinden gelmek için Amazon Web Services'i (AWS) nasıl kullanabilecekleri açıklanmaktadır.

Veri deposu, eşzamansız iletişim ve hizmet keşfi de dahil olmak üzere bu teknik dokümanda ele alınan çeşitli konuların oldukça karmaşık yapısı nedeniyle, okuyucunun mimari seçimler yapmadan önce sağlanan kılavuza ek olarak kendi uygulamalarının özel gereksinimlerini ve kullanım durumlarını göz önünde bulundurması teşvik edilmektedir.

Giriş

Mikro hizmet mimarileri, yazılım mühendisliğine tamamen yeni bir yaklaşım değil, daha ziyade aşağıdaki gibi çeşitli başarılı ve kanıtlanmış kavramların bir kombinasyonudur:

- Çevik yazılım geliştirme
- Hizmet odaklı mimariler
- API öncelikli tasarım
- Sürekli entegrasyon/sürekli teslimat (CI/CD)

Birçok durumda, mikro hizmetler için On İki Faktör Uygulamasının tasarım kalıpları kullanılır.

Bu teknik dokümanda ilk olarak yüksek düzeyde ölçeklenebilir, hataya dayanıklı bir mikro hizmet mimarisinin (kullanıcı arayüzü, mikro hizmet uygulaması ve veri deposu) farklı yönleri ve konteyner teknolojileri kullanılarak AWS üzerinde nasıl oluşturulacağı açıklanmaktadır. Ardından, operasyonel karmaşıklığı azaltmak amacıyla tipik bir sunucusuz mikro hizmet mimarisini uygulamak için AWS hizmetlerini önermektedir.

Sunucusuz, aşağıdaki ilkelere göre bir operasyonel model olarak tanımlanır:

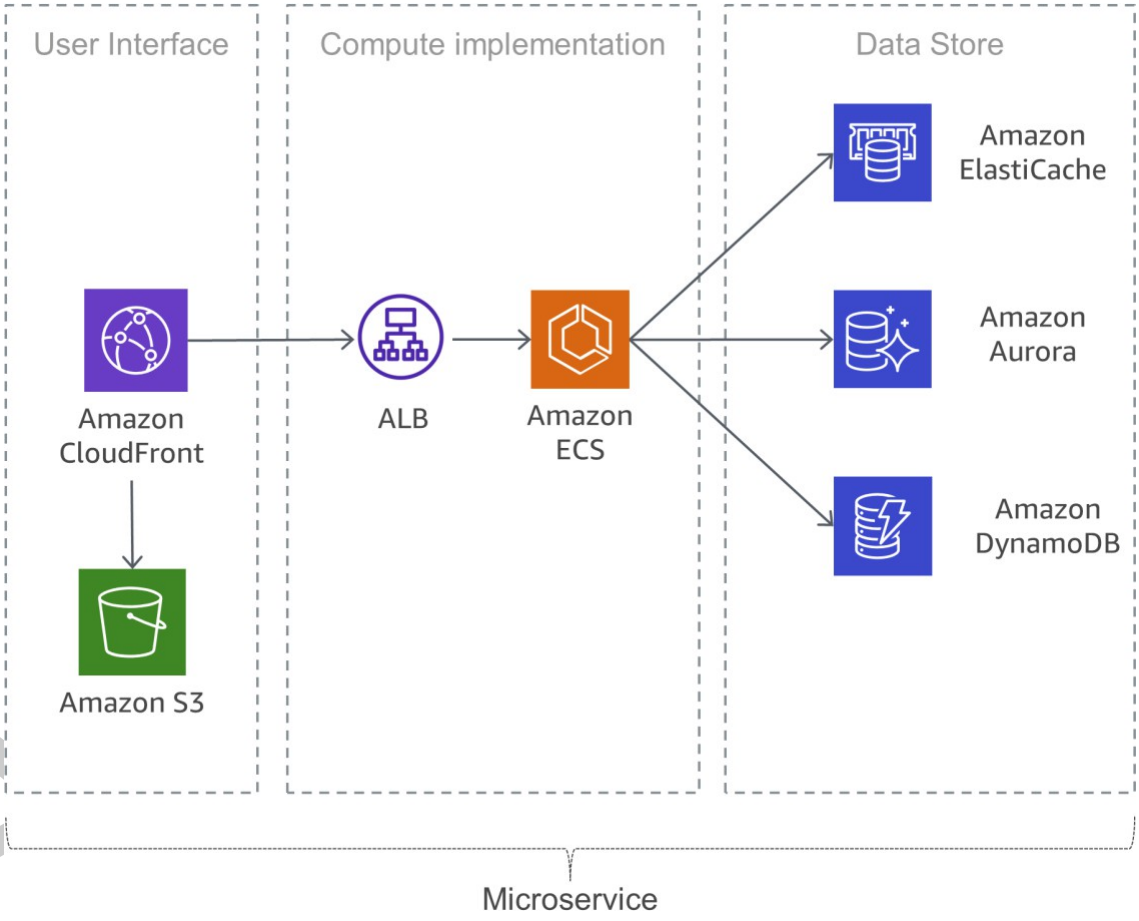
- Sağlayacak veya yönetecek altyapı yok
- Tüketim birimine göre otomatik olarak ölçeklendirme
- *Değer için ödeme* faturalama modeli
- Yerleşik kullanılabilirlik ve hata toleransı

Son olarak, bu teknik doküman sistemin genelini kapsamakta ve dağıtık izleme ve denetim, veri tutarlılığı ve eşzamansız iletişim gibi mikro hizmetler mimarisinin çapraz hizmet yönlerini tartışmaktadır.

Bu teknik doküman yalnızca AWS Cloud'da çalışan iş yüklerine odaklanmaktadır. Hibrit senaryoları veya geçiş stratejilerini kapsamaz. Geçiş hakkında daha fazla bilgi için Konteyner Geçiş Metodolojisi teknik incelemesine bakın.

AWS üzerinde mikro hizmet mimarisi

Tipik monolitik uygulamalar, kullanıcı arayüzü (UI) katmanı, iş katmanı ve kalıcılık katmanı gibi farklı katmanlar kullanılarak oluşturulur. Mikro hizmet mimarisinin ana fikri, işlevleri teknolojik katmanlara göre *değil,* belirli bir etki alanını uygulayarak uyumlu dikeylere bölmektir. Aşağıdaki şekilde AWS'deki tipik bir mikro hizmet uygulaması için bir referans mimari gösterilmektedir.



AWS üzerinde tipik mikro hizmet uygulaması

Kullanıcı arayüzü

Modern web uygulamaları, temsili durum aktarımı (REST) veya RESTful ile iletişim kuran tek sayfalık bir uygulamayı hayata geçirmek için genellikle JavaScript çerçevelerini kullanır.

API. Statik web içeriği Amazon Simple Storage Service (Amazon S3) ve Amazon CloudFront kullanılarak sunulabilir.

Bir mikro hizmetin istemcileri en yakın uç konumdan sunulduğundan ve yanıtları bir önbellekten veya kaynağa optimize edilmiş bağlantılara sahip bir proxy sunucusundan aldığından, gecikmeler önemli ölçüde azaltılabilir. Ancak, birbirine yakın çalışan mikro hizmetler bir içerik dağıtım ağından faydalanamaz. Bazı durumlarda, bu yaklaşım aslında ek gecikme ekleyebilir. En iyi uygulama, sohbeti azaltmak ve gecikmeleri en aza indirmek için diğer önbelleğe alma mekanizmalarını uygulamaktır. Daha fazla bilgi için Chattiness konusuna bakın.

Mikro Hizmetler

API'ler mikro hizmetlerin ön kapısıdır; yani API'ler, tipik olarak RESTful web hizmetleri API'si olan bir dizi programatik arayüzün arkasındaki uygulama mantığı için giriş noktası görevi görür. Bu API istemcilerden gelen çağrıları kabul eder ve işler ve trafik yönetimi, istek filtreleme, yönlendirme, önbelleğe alma, kimlik doğrulama ve yetkilendirme gibi işlevleri uygulayabilir.

Mikro hizmet uygulaması

AWS, mikro hizmetlerin geliştirilmesini destekleyen entegre yapı taşlarına sahiptir. İki popüler yaklaşım AWS Lambda ve AWS Fargate ile Docker konteynerlerini kullanmaktır.

AWS Lambda ile kodunuzu yüklersiniz ve Lambda'nın yüksek kullanılabilirlikle gerçek talep eğrinizi karşılamak üzere uygulamayı çalıştırmak ve ölçeklendirmek için gereken her şeyi yapmasına izin verirsiniz. Altyapı yönetimine gerek yoktur. Lambda çeşitli programlama dillerini destekler ve diğer AWS hizmetlerinden çağrılabilir veya doğrudan herhangi bir web veya mobil uygulamadan çağrılabilir. AWS Lambda'nın en büyük avantajlarından biri hızlı hareket edebilmenizdir: güvenlik ve ölçeklendirme AWS tarafından yönetildiği için iş mantığınıza odaklanabilirsiniz. Lambda'nın fikirsel yaklaşımı ölçeklenebilir platformu yönlendirir.

Dağıtım için operasyonel çabaları azaltmaya yönelik yaygın bir yaklaşım konteyner tabanlı dağıtımdır. Docker gibi konteyner teknolojileri taşınabilirlik, üretkenlik ve verimlilik gibi avantajlar nedeniyle son birkaç yılda popülerliğini artırmıştır. Konteynerler ile öğrenme eğrisi dik olabilir ve Docker imajlarınız için güvenlik düzeltmeleri ve izleme hakkında düşünmeniz gerekir. Amazon Elastic Container Service (Amazon ECS) ve Amazon

Elastic Kubernetes Service (Amazon EKS), kendi küme yönetimi altyapınızı kurma, işletme ve ölçeklendirme ihtiyacını ortadan kaldırır. API çağrıları ile Docker özellikli uygulamaları başlatıp durdurabilir, kümenizin tüm durumunu sorgulayabilir ve güvenlik grupları, Yük Dengeleme, Amazon Elastic Block Store (Amazon EBS) birimleri ve AWS Kimlik ve Erişim Yönetimi (IAM) rolleri gibi birçok tanıdık özelliğe erişebilirsiniz.

AWS Fargate, hem Amazon ECS hem de Amazon EKS ile çalışan konteynerler için sunucusuz bir hesaplama motorudur. Fargate ile artık konteyner uygulamalarınız için yeterli bilgi işlem kaynağı sağlama konusunda endişelenmenize gerek yok. Fargate on binlerce konteyner başlatabilir ve en kritik görev uygulamalarınızı çalıştırmak için kolayca ölçeklenebilir.

Amazon ECS, Amazon ECS'nin görevleri yerleştirme ve sonlandırma şeklini özelleştirmek için kapsayıcı yerleştirme stratejilerini ve kısıtlamalarını destekler. Görev yerleştirme kısıtlaması, görev yerleştirme sırasında dikkate alınan bir kuraldır. Temelde anahtar-değer çiftleri olan öznitelikleri konteyner örneklerinizle ilişkilendirebilir ve ardından görevleri bu özniteliklere göre yerleştirmek için bir kısıtlama kullanabilirsiniz. Örneğin, GPU destekli örnekler gibi belirli mikro hizmetleri örnek türüne veya örnek özelliğine göre yerleştirmek için kısıtlamalar kullanabilirsiniz.

Amazon EKS, açık kaynaklı Kubernetes yazılımının güncel sürümlerini çalıştırır, böylece Kubernetes topluluğunun mevcut tüm eklentilerini ve araçlarını kullanabilirsiniz. Amazon EKS üzerinde çalışan uygulamalar, ister şirket içi veri merkezlerinde ister genel bulutlarda çalışsın, herhangi bir standart Kubernetes ortamında çalışan uygulamalarla tamamen uyumludur. Amazon EKS, IAM'yi Kubernetes ile entegre ederek IAM varlıklarını Kubernetes'teki yerel kimlik doğrulama sistemine kaydetmenizi sağlar. Kubernetes kontrol düzlemi ile kimlik doğrulaması yapmak için kimlik bilgilerini manuel olarak ayarlamanıza gerek yoktur. IAM entegrasyonu, kontrol düzleminin kendisiyle doğrudan kimlik doğrulaması yapmak ve Kubernetes kontrol düzleminizin genel uç noktasına ayrıntılı erişim sağlamak için IAM'yi kullanmanıza olanak tanır.

Amazon ECS ve Amazon EKS'de kullanılan Docker görüntüleri Amazon Elastic Container Registry'de (Amazon ECR) depolanabilir. Amazon ECR, konteyner kayıt defterinize güç sağlamak için gereken altyapıyı çalıştırma ve ölçeklendirme ihtiyacını ortadan kaldırır.

Sürekli entegrasyon ve sürekli teslimat (CI/CD) en iyi uygulamalardır ve sistem kararlılığını ve güvenliğini korurken hızlı yazılım değişikliklerine olanak tanıyan bir DevOps girişiminin hayati bir parçasıdır. Ancak bu konu bu teknik incelemenin kapsamı dışındadır. Daha fazlası için

bilgi için AWS'de Sürekli Entegrasyon ve Sürekli Teslimat Uygulama teknik incelemesine bakın.

Özel bağlantılar

AWS PrivateLink, sanal özel bulutunuzu (VPC) desteklenen AWS hizmetlerine, diğer AWS hesapları tarafından barındırılan hizmetlere (VPC uç nokta hizmetleri) ve desteklenen AWS Marketplace iş ortağı hizmetlerine özel olarak bağlamanızı sağlayan yüksek oranda kullanılabilir, ölçeklenebilir bir teknolojidir. Hizmetle iletişim kurmak için bir internet ağ geçidine, ağ adresi çeviri cihazına, genel IP adresine, AWS Direct Connect bağlantısına veya VPN bağlantısına ihtiyacınız yoktur. VPC'niz ile hizmet arasındaki trafik Amazon ağından çıkmaz.

Özel bağlantılar, mikro hizmet mimarisinin izolasyonunu ve güvenliğini artırmanın harika bir yoludur. Örneğin bir mikro hizmet, tamamen ayrı bir VPC'de konuşlandırılabilir, bir yük dengeleyici tarafından önlenebilir ve bir AWS PrivateLink uç noktası aracılığıyla diğer mikro hizmetlere maruz bırakılabilir. Bu kurulumla, AWS PrivateLink kullanılarak, mikro hizmete giden ve mikro hizmetten gelen ağ trafiği asla genel internetten geçmez. Bu izolasyonun kullanım alanlarından biri PCI, HIPPA ve EU/US Privacy Shield gibi hassas verileri işleyen hizmetler için yasal uyumluluktur. Ayrıca AWS PrivateLink, güvenlik duvarı kurallarına, yol tanımlarına veya rota tablolarına gerek kalmadan mikro hizmetlerin farklı hesaplar ve Amazon VPC'leri arasında bağlanmasına olanak tanıyarak ağ yönetimini basitleştirir. PrivateLink'i kullanarak, hizmet olarak yazılım (SaaS) sağlayıcıları ve ISV'ler de mikro hizmet tabanlı çözümlerini tam operasyonel izolasyon ve güvenli erişim ile sunabilirler.

Veri deposu

Veri deposu, mikro hizmetlerin ihtiyaç duyduğu verileri kalıcı hale getirmek için kullanılır. Oturum verileri için popüler depolar Memcached veya Redis gibi bellek içi önbelleklerdir. AWS, her iki teknolojiyi de yönetilen Amazon ElastiCache hizmetinin bir parçası olarak sunar.

Uygulama sunucuları ile veritabanı arasına bir önbellek yerleştirmek, veritabanı üzerindeki okuma yükünü azaltmak için yaygın bir mekanizmadır ve bu da kaynakların daha fazla yazmayı desteklemek için kullanılmasını sağlayabilir. Önbellekler gecikme süresini de iyileştirebilir.

İlişkisel veritabanları, yapılandırılmış verileri ve iş nesnelerini depolamak için hala çok

populerdir. AWS altı veritabanı motoru sunmaktadır (Microsoft SQL Server, Oracle, MySQL,

MariaDB, PostgreSQL ve Amazon Aurora) Amazon İlişkisel Veritabanı Hizmeti (Amazon RDS) aracılığıyla yönetilen hizmetler olarak.

Bununla birlikte, ilişkisel veritabanları sonsuz ölçek için tasarlanmamıştır, bu da çok sayıda sorguyu desteklemek için tekniklerin uygulanmasını zorlaştırabilir ve yoğun zaman alabilir.

NoSQL veritabanları, ilişkisel veritabanlarının tutarlılığı yerine ölçeklenebilirliği, performansı ve kullanılabilirliği tercih edecek şekilde tasarlanmıştır. NoSQL veritabanlarının önemli bir unsuru, genellikle katı bir şema uygulamamalarıdır. Veriler yatay olarak ölçeklendirilebilen bölümlere dağıtılır ve bölüm anahtarları kullanılarak alınır.

Bireysel mikro hizmetler tek bir şeyi iyi yapmak için tasarlandığından, genellikle NoSQL kalıcılığına çok uygun olabilecek basitleştirilmiş bir veri modeline sahiptirler. NoSQL veritabanlarının ilişkisel veritabanlarından farklı erişim modellerine sahip olduğunu anlamak önemlidir. Örneğin, tabloları birleştirmek mümkün değildir. Bu gerekliyse, mantığın uygulama içinde uygulanması gerekir. Herhangi bir miktarda veriyi depolayabilen ve alabilen ve herhangi bir istek trafiği seviyesine hizmet verebilen bir veritabanı tablosu oluşturmak için Amazon DynamoDB'yi kullanabilirsiniz. DynamoDB tek haneli milisaniye performansı sunar, ancak mikrosaniye cinsinden yanıt süreleri gerektiren belirli kullanım durumları vardır. Amazon DynamoDB Accelerator (DAX), verilere erişim için önbelleğe alma özellikleri sağlar.

DynamoDB ayrıca gerçek trafiğe yanıt olarak verim kapasitesini dinamik olarak ayarlamak için otomatik bir ölçeklendirme özelliği sunar. Ancak, uygulamanızdaki kısa süreli büyük aktivite artışları nedeniyle kapasite planlamasının zor olduğu veya mümkün olmadığı durumlar vardır. Bu tür durumlar için DynamoDB, talep başına basit ödeme fiyatlandırması sunan isteğe bağlı bir seçenek sunar.DynamoDB isteğe bağlı, kapasite planlaması olmadan anında saniyede binlerce istek sunabilir.

Operasyonel karmaşıklığının azaltılması

Bu whitepaper'da daha önce açıklanan mimari zaten yönetilen hizmetleri kullanıyor, ancak Amazon Elastic Compute Cloud (Amazon EC2) örneklerinin hala yönetilmesi gerekiyor. Mikro hizmetleri çalıştırmak, sürdürmek ve izlemek için gereken operasyonel çabalar, tamamen sunucusuz bir mimari kullanılarak daha da azaltılabilir.

API uygulaması

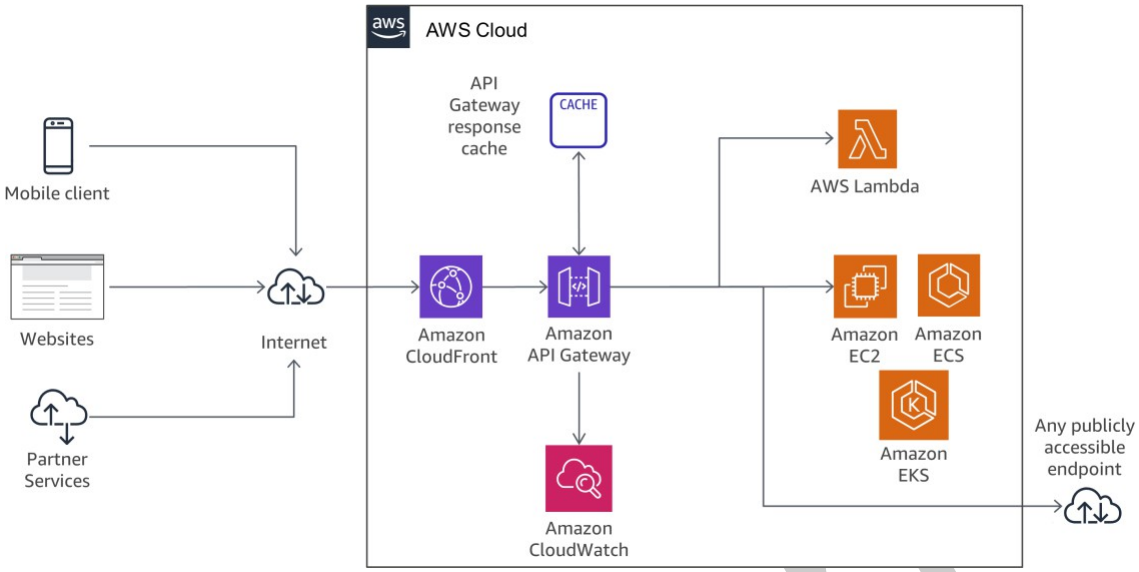
Bir API'yi tasarlamak, dağıtmak, izlemek, sürekli iyileştirmek ve bakımını yapmak zaman alıcı bir iş olabilir. Bazen tüm istemciler için geriye dönük uyumluluğu sağlamak için API'lerin farklı sürümlerinin çalıştırılması gerekir. Geliştirme döngüsünün farklı aşamaları (örneğin, geliştirme, test ve üretim) operasyonel çabaları daha da artırır.

Yetkilendirme tüm API'ler için kritik bir özelliktir, ancak genellikle oluşturulması karmaşıktır ve tekrarlayan işler içerir. Bir API yayınlandığında ve başarılı olduğunda, bir sonraki zorluk API'leri kullanan üçüncü taraf geliştiricilerin ekosistemini yönetmek, izlemek ve bunlardan para kazanmaktır.

Diğer önemli özellikler ve zorluklar arasında arka uç hizmetlerini korumak için istekleri azaltma, API yanıtlarını önbelleğe alma, istek ve yanıt dönüşümünü işleme ve Swagger gibi araçlarla API tanımları ve belgeleri oluşturma yer alır.

Amazon API Gateway bu zorlukların üstesinden gelir ve RESTful API'lerin oluşturulması ve sürdürülmesindeki operasyonel karmaşıklığı azaltır. API Gateway, AWS API veya AWS Management Console kullanarak Swagger tanımlarını içe aktararak API'lerinizi programlı olarak oluşturmanıza olanak tanır. API Gateway, Amazon EC2, Amazon ECS, AWS Lambda veya herhangi bir şirket içi ortamda çalışan herhangi bir web uygulaması için bir ön kapı görevi görür. Temel olarak API Gateway, sunucuları yönetmek zorunda kalmadan API'leri çalıştırmanıza olanak tanır.

Aşağıdaki şekilde API Gateway'in API çağrılarını nasıl işlediği ve diğer bileşenlerle nasıl etkileşime girdiği gösterilmektedir. Mobil cihazlardan, web sitelerinden veya diğer arka uç hizmetlerinden gelen istekler, gecikmeyi en aza indirmek ve optimum kullanıcı deneyimi sağlamak için en yakın CloudFront Varlık Noktasına yönlendirilir.



API Gateway çağrı akışı

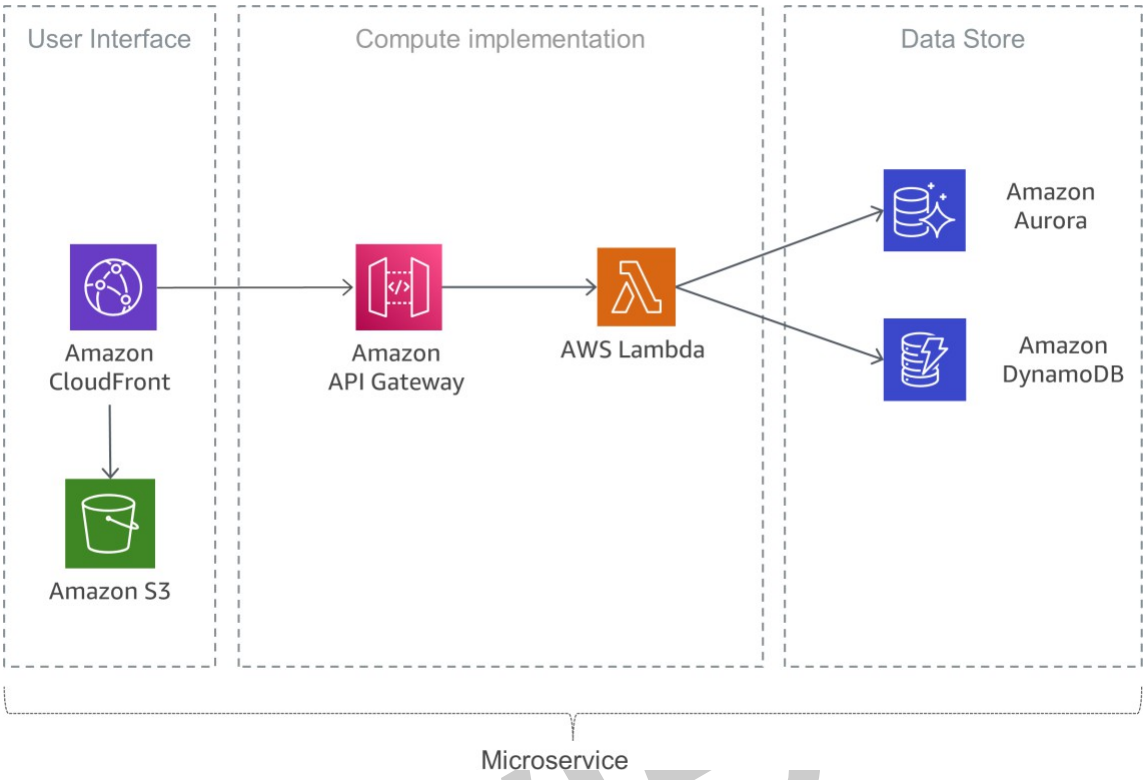
Sunucusuz mikro hizmetler

> *"Hiçbir sunucuyu yönetmek, hiç sunucu olmamasından daha kolay değildir."* - *AWS re:Invent*

Sunuculardan kurtulmak, operasyonel karmaşıklığı ortadan kaldırmak için harika bir yoldur.

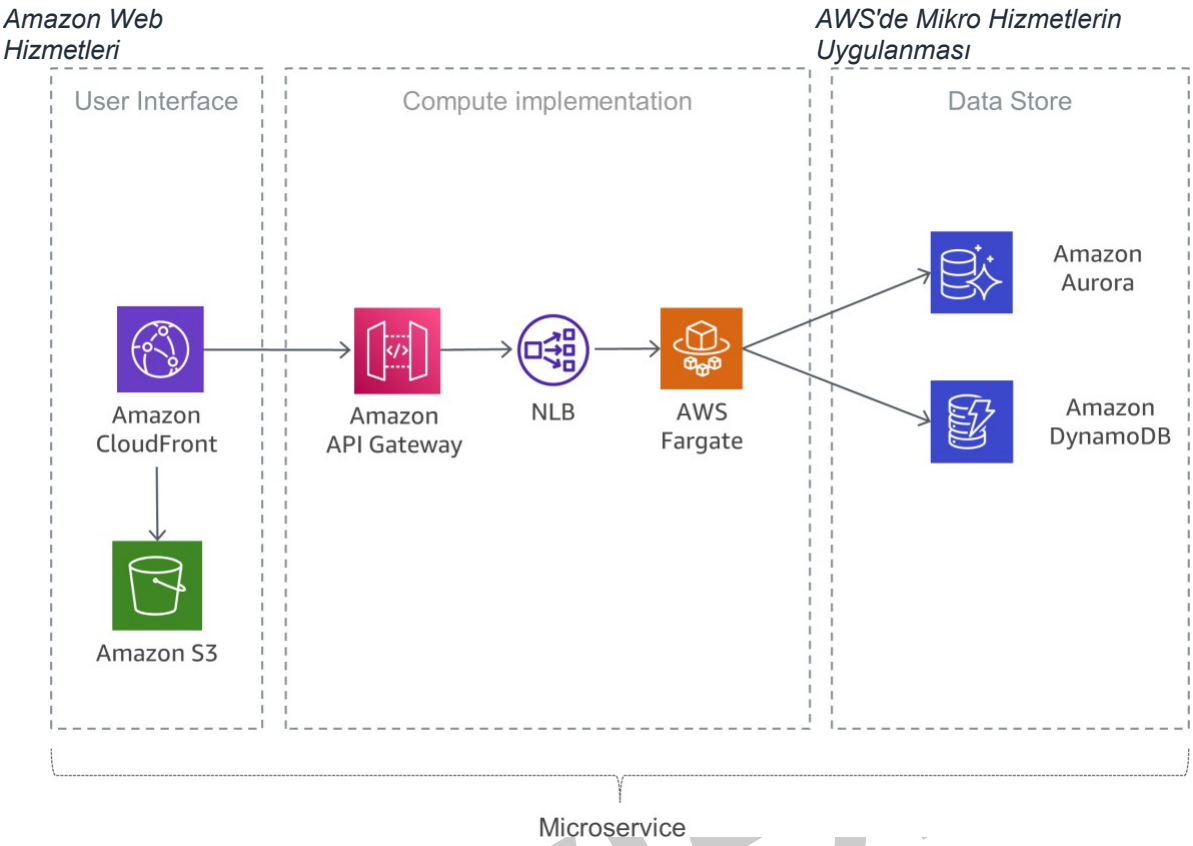
Lambda, API Gateway ile sıkı bir şekilde entegre edilmiştir. API Gateway'den Lambda'ya senkronize çağrılar yapabilme özelliği, tamamen sunucusuz uygulamaların oluşturulmasını sağlar ve Amazon API Gateway *Geliştirici Kılavuzu'nda* ayrıntılı olarak açıklanmıştır.

Aşağıdaki şekilde AWS Lambda ile sunucusuz bir mikro hizmetin mimarisi gösterilmektedir; burada tüm hizmet yönetilen hizmetlerden oluşturulmaktadır, bu da ölçek ve yüksek kullanılabilirlik için tasarım yapma konusundaki mimari yükü ortadan kaldırmakta ve mikro hizmetin temel altyapısını çalıştırma ve izleme konusundaki operasyonel çabaları ortadan kaldırmaktadır.



AWS Lambda kullanarak sunucusuz mikro hizmet

Sunucusuz hizmetlere dayanan benzer bir uygulama da aşağıdaki şekilde gösterilmektedir. Bu mimaride Fargate ile birlikte Docker konteynerleri kullanıldığından altta yatan altyapıyı önemsemeye gerek yoktur. DynamoDB'ye ek olarak, Aurora (MySQL uyumlu sürüm) için isteğe bağlı, otomatik ölçeklendirme yapılandırması olan ve veritabanının uygulamanızın ihtiyaçlarına göre otomatik olarak başlayacağı, kapanacağı ve kapasiteyi yukarı veya aşağı ölçeklendireceği Amazon Aurora Serverless kullanılır.



Fargate kullanarak sunucusuz mikro hizmet

Felaket kurtarma

Bu whitepaper'ın giriş bölümünde daha önce belirtildiği gibi, tipik mikro hizmet uygulamaları On İki Faktörlü Uygulama kalıpları kullanılarak uygulanmaktadır. Süreçler bölümünde "On iki faktörlü süreçler durumsuzdur ve hiçbir şey paylaşmaz. Kalıcı olması gereken tüm veriler durum bilgisi olan bir destek hizmetinde, tipik olarak bir veritabanında saklanmalıdır."

Tipik bir mikro hizmet mimarisi için bu, felaket kurtarma için ana odağın uygulamanın durumunu koruyan aşağı akış hizmetlerinde olması gerektiği anlamına gelir. Örneğin bunlar dosya sistemleri, veritabanları ya da kuyruklar olabilir. Bir felaket kurtarma stratejisi oluştururken kuruluşlar en yaygın olarak kurtarma zamanı hedefi ve kurtarma noktası hedefi için plan yapar.

Kurtarma süresi hedefi, hizmet kesintisi ile hizmetin geri yüklenmesi arasındaki kabul edilebilir maksimum gecikmedir. Bu hedef, hizmet kullanılamadığında neyin kabul edilebilir bir zaman aralığı olarak görüldüğünü belirler ve kuruluş tarafından tanımlanır.

Kurtarma noktası hedefi, son veri kurtarma noktasından bu yana geçen kabul edilebilir maksimum süredir. Bu hedef, son kurtarma noktası ile hizmet kesintisi arasında neyin kabul edilebilir bir veri kaybı olarak görüldüğünü belirler ve kuruluş tarafından tanımlanır.

Daha fazla bilgi için AWS'de İş Yüklerinin Felaket Kurtarması'na bakın: Bulutta Kurtarma teknik incelemesine bakın.

Yüksek kullanılabilirlik

Bu bölüm, farklı bilgi işlem seçenekleri için yüksek kullanılabilirliğe daha yakından bakmaktadır.

Amazon EKS, yüksek kullanılabilirlik sağlamak için Kubernetes kontrol ve veri düzlemi örneklerini birden fazla Kullanılabilirlik Bölgesinde çalıştırır. Amazon EKS, sağlıksız kontrol düzlemi örneklerini otomatik olarak algılar ve değiştirir ve bunlar için otomatik sürüm yükseltmeleri ve düzeltme eki sağlar. Bu kontrol düzlemi, bir bölgedeki üç Kullanılabilirlik Bölgesinde çalışan en az iki API sunucu düğümü ve üç etcd düğümünden oluşur. Amazon EKS, yüksek kullanılabilirliği korumak için AWS Bölgeleri mimarisini kullanır.

Amazon ECR, imajları yüksek oranda kullanılabilir ve yüksek performanslı bir mimaride barındırarak, konteyner uygulamaları için imajları Kullanılabilirlik Bölgeleri arasında güvenilir bir şekilde dağıtmanıza olanak tanır. Amazon ECR; Amazon EKS, Amazon ECS ve AWS Lambda ile birlikte çalışarak geliştirmeden üretime iş akışını basitleştirir.

Amazon ECS, bir AWS Bölgesi içindeki birden fazla Kullanılabilirlik Bölgesinde konteynerleri yüksek kullanılabilirlikte çalıştırmayı basitleştiren bölgesel bir hizmettir. Amazon ECS, kaynak ihtiyaçlarınıza (örneğin, CPU veya RAM) ve kullanılabilirlik gereksinimlerinize göre kapsayıcıları kümelerinize yerleştiren çoklu zamanlama stratejileri içerir.

AWS Lambda, tek bir bölgede hizmet kesintisi olması durumunda olayları işlemek için kullanılabilir olmasını sağlamak için işlevinizi birden çok Kullanılabilirlik Bölgesinde çalıştırır. İşlevinizi hesabınızdaki bir sanal özel buluta (VPC) bağlanacak şekilde yapılandırırsanız, yüksek kullanılabilirlik sağlamak için birden çok Kullanılabilirlik Bölgesinde alt ağlar belirtin.

Lambda tabanlı uygulamalarını dağıtma

Sunucusuz uygulamaları tanımlamak, dağıtmak ve yapılandırmak için AWS

CloudFormation'ı kullanabilirsiniz.

AWS Sunucusuz Uygulama Modeli (AWS SAM), sunucusuz uygulamaları tanımlamak için kullanışlı bir yoldur. AWS SAM, CloudFormation tarafından yerel olarak desteklenir ve sunucusuz kaynakları ifade etmek için basitleştirilmiş bir sözdizimi tanımlar. Uygulamanızı dağıtmak için uygulamanızın bir parçası olarak ihtiyaç duyduğunuz kaynakları, ilişkili izin politikalarıyla birlikte bir CloudFormation şablonunda belirtin, dağıtım yapılarınızı paketleyin ve şablonu dağıtın. AWS SAM'i temel alan SAM Local, sunucusuz uygulamalarınızı Lambda çalışma zamanına yüklemeden önce yerel olarak geliştirmeniz, test etmeniz ve analiz etmeniz için bir ortam sağlayan bir AWS Komut Satırı Arayüzü aracıdır.AWS çalışma zamanı ortamını simüle eden yerel bir test ortamı oluşturmak için SAM Local'i kullanabilirsiniz.

Dağıtık sistemler bileşenler

AWS'nin bireysel mikro hizmetlerle ilgili zorlukları nasıl çözebileceğine baktıktan sonra, odak noktası hizmet keşfi, veri tutarlılığı, eşzamansız iletişim ve dağıtılmış izleme ve denetim gibi çapraz hizmet zorluklarına geçer.

Hizmet keşif

Mikro hizmet mimarilerindeki temel zorluklardan biri, hizmetlerin birbirlerini keşfetmelerini ve birbirleriyle etkileşime girmelerini sağlamaktır. Mikro hizmet mimarilerinin dağıtık özellikleri hizmetlerin iletişim kurmasını zorlaştırmakla kalmaz, aynı zamanda bu sistemlerin sağlığını kontrol etmek ve yeni uygulamalar kullanıma sunulduğunda bunları duyurmak gibi başka zorluklar da ortaya çıkarır. Ayrıca uygulamalar tarafından kullanılabilecek yapılandırma verileri gibi meta bilgileri nasıl ve nerede saklayacağınıza da karar vermeniz gerekir. Bu bölümde mikro hizmet tabanlı mimariler için AWS'de hizmet keşfi gerçekleştirmeye yönelik çeşitli teknikler incelenmektedir.

DNS tabanlı hizmet keşfi

Amazon ECS artık konteynerli hizmetlerinizin birbirlerini keşfetmelerini ve birbirleriyle bağlantı kurmalarını sağlayan entegre hizmet keşfi içeriyor.

Önceden, hizmetlerin birbirlerini keşfedebilmelerini ve birbirleriyle bağlantı kurabilmelerini sağlamak için Amazon Route 53, AWS Lambda ve ECS olay akışlarını temel alan kendi hizmet keşif sisteminizi yapılandırmanız ve çalıştırmanız ya da her hizmeti bir yük dengeleyiciye bağlamanız gerekiyordu.

Amazon ECS, Route 53 Auto Naming API'sini kullanarak bir hizmet adları kaydı oluşturur ve yönetir. Adlar otomatik olarak bir dizi DNS kaydıyla eşleştirilir, böylece kodunuzda bir hizmete adıyla başvurabilir ve adın çalışma zamanında hizmetin uç noktasına çözümlenmesini sağlamak için DNS sorguları yazabilirsiniz. Bir hizmetin görev tanımında sağlık kontrolü koşullarını belirtebilirsiniz ve Amazon ECS, bir hizmet araması tarafından yalnızca sağlıklı hizmet uç noktalarının döndürülmesini sağlar.

Ayrıca, Kubernetes tarafından yönetilen hizmetler için birleşik hizmet bulmayı da kullanabilirsiniz. Bu entegrasyonu etkinleştirmek için AWS, bir Kubernetes kuluçka projesi olan External DNS projesine katkıda bulundu.

Diğer bir seçenek de AWS Cloud Map'in yeteneklerini kullanmaktır. AWS Cloud Map, İnternet Protokolleri (IP'ler), Tekdüzen Kaynak Konum Belirleyicileri (URL'ler) ve Amazon Kaynak Adları (ARN'ler) gibi kaynaklar için bir hizmet kayıt defteri sağlayarak ve daha hızlı bir değişiklik yayılımı ve keşfedilen kaynaklar kümesini daraltmak için öznitelikleri kullanma yeteneği ile API tabanlı bir hizmet keşif mekanizması sunarak Otomatik Adlandırma API'lerinin yeteneklerini genişletir. Mevcut Route 53 Auto Naming kaynakları otomatik olarak AWS Cloud Map'e yükseltilir.

Üçüncü taraf yazılımlar

Hizmet keşfini uygulamaya yönelik farklı bir yaklaşım da HashiCorp Consul, etcd veya Netflix Eureka gibi üçüncü taraf yazılımları kullanmaktır. Her üç örnek de dağıtılmış, güvenilir anahtar-değer depolarıdır. HashiCorp Consul için esnek, ölçeklenebilir bir AWS Bulut ortamı kuran ve HashiCorp Consul'u seçtiğiniz bir yapılandırmada otomatik olarak başlatan bir AWS Hızlı Başlangıç vardır.

Servis ağları

Gelişmiş bir mikro hizmet mimarisinde, gerçek uygulama yüzlerce, hatta binlerce hizmetten oluşabilir. Genellikle uygulamanın en karmaşık kısmı gerçek hizmetlerin kendileri değil, bu hizmetler arasındaki iletişimdir. Hizmet ağları, mikro hizmet mimarilerinde trafiği izlemek ve kontrol etmekten sorumlu olan, hizmetler arası iletişimi ele almak için ek bir katmandır. Bu, hizmet keşfi gibi görevlerin tamamen bu katman tarafından ele alınmasını sağlar.

Tipik olarak, bir hizmet ağı bir veri düzlemi ve bir kontrol düzlemine ayrılır. Veri düzlemi, uygulama koduyla birlikte bir dizi akıllı proxy'den oluşur.

Mikro hizmetler arasındaki tüm ağ iletişimini kesen özel yan vekil. Kontrol düzlemi proxy'lerle iletişimden sorumludur.

Hizmet ağları şeffaftır, yani uygulama geliştiricilerin bu ek katmandan haberdar olması ve mevcut uygulama kodunda değişiklik yapması gerekmez. AWS App Mesh, hizmetlerinizin birden fazla bilgi işlem altyapısı türünde birbirleriyle iletişim kurmasını sağlamak için uygulama düzeyinde ağ sağlayan bir hizmet ağıdır. App Mesh, hizmetlerinizin iletişim şeklini standartlaştırarak size tam görünürlük sağlar ve uygulamalarınız için yüksek kullanılabilirlik sağlar.

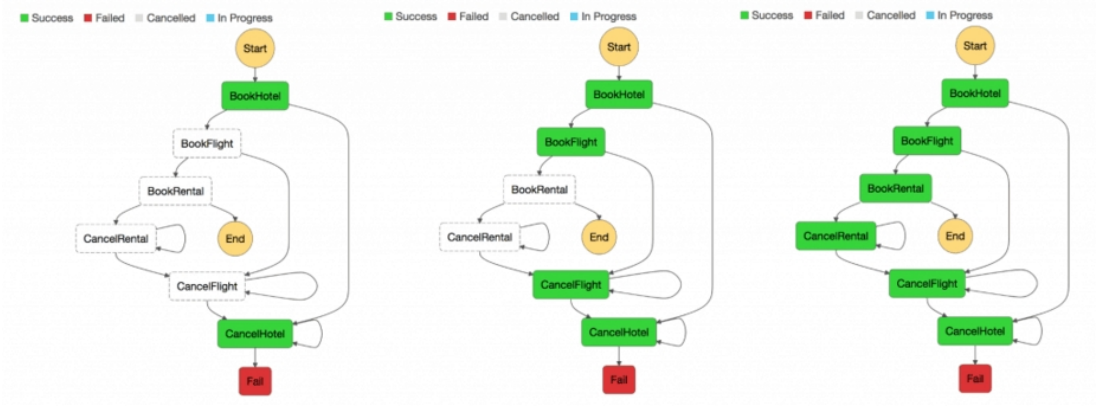
App Mesh'i Amazon EC2, Fargate, Amazon ECS, Amazon EKS ve AWS'de kendi kendini yöneten Kubernetes üzerinde çalışan mevcut veya yeni mikro hizmetlerle kullanabilirsiniz. App Mesh, herhangi bir kod değişikliği yapmadan tek bir uygulama olarak kümeler, orkestrasyon sistemleri veya VPC'ler arasında çalışan mikro hizmetlerin iletişimini izleyebilir ve kontrol edebilir.

Dağıtılmış veri yönetimi

Monolitik uygulamalar genellikle tüm uygulama bileşenleri için ortak olan tek bir veri modelini tanımlayan büyük bir ilişkisel veritabanı tarafından desteklenir. Bir mikro hizmet yaklaşımında, böyle bir merkezi veritabanı, merkezi olmayan ve bağımsız bileşenler oluşturma hedefini engelleyecektir. Her mikro hizmet bileşeni kendi veri kalıcılığı katmanına sahip olmalıdır.

Ancak dağıtık veri yönetimi yeni zorlukları da beraberinde getirmektedir. CAP teoreminin bir sonucu olarak, dağıtık mikro hizmet mimarileri doğal olarak performans için tutarlılıktan ödün verir ve nihai tutarlılığı benimsemeleri gerekir.

Dağıtılmış bir sistemde, ticari işlemler birden fazla mikro hizmeti kapsayabilir. Tek bir ACID işlemi kullanamadıkları için, kısmi yürütmelerle karşılaşabilirsiniz. Bu durumda, halihazırda işlenmiş işlemleri yeniden yapmak için bazı kontrol mantığına ihtiyacımız olacaktır. Bu amaçla, dağıtılmış Saga modeli yaygın olarak kullanılır. Başarısız bir ticari işlem durumunda Saga, önceki işlemler tarafından yapılan değişiklikleri geri alan bir dizi telafi edici işlem düzenler. AWS Step Functions, aşağıdaki şekilde gösterildiği gibi bir Saga yürütme koordinatörünün uygulanmasını kolaylaştırır.



Saga yürütme koordinatörü

Temel veri yönetimi araçları ve prosedürleri tarafından yönetilen merkezi bir kritik referans veri deposu oluşturmak, mikro hizmetlerin kritik verilerini senkronize etmeleri ve muhtemelen durumu geri almaları için bir araç sağlar. Zamanlanmış Amazon CloudWatch Olayları ile AWS Lambda kullanarak basit bir temizleme ve veri tekilleştirme mekanizması oluşturabilirsiniz.

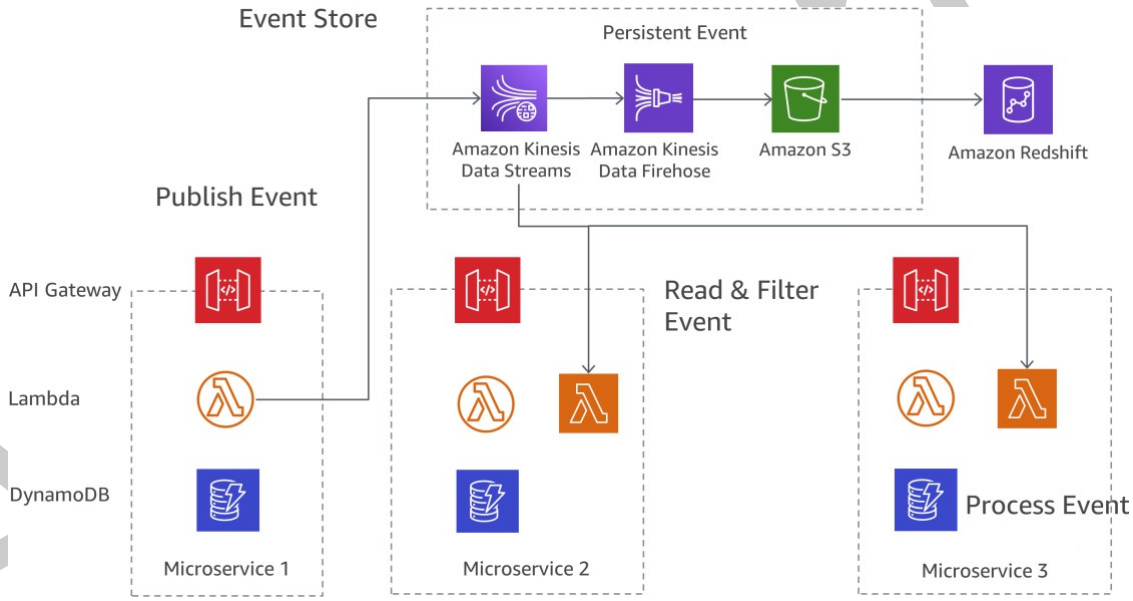
Durum değişikliklerinin tek bir mikro hizmetten daha fazlasını etkilemesi çok yaygındır. Bu gibi durumlarda olay kaynağının kullanışlı bir model olduğu kanıtlanmıştır. Olay kaynağının arkasındaki temel fikir, her uygulama değişikliğini bir olay kaydı olarak temsil etmek ve sürdürmektir.
Uygulama durumunu kalıcı hale getirmek yerine, veriler bir olay akışı olarak depolanır. Veritabanı işlem günlüğü ve sürüm kontrol sistemleri, olay kaynak kullanımı için iyi bilinen iki örnektir. Olay kaynağının birkaç faydası vardır: durum, zaman içinde herhangi bir nokta için belirlenebilir ve yeniden oluşturulabilir. Doğal olarak kalıcı bir denetim izi oluşturur ve ayrıca hata ayıklamayı kolaylaştırır.

Mikro hizmet mimarileri bağlamında olay kaynağı, bir yayınlama ve abone olma modeli kullanarak bir uygulamanın farklı bölümlerinin ayrıştırılmasını sağlar ve aynı olay verilerini ayrı mikro hizmetler için farklı veri modellerine besler. Olay kaynağı, okuma ve yazma iş yüklerini birbirinden ayırmak ve her ikisini de performans, ölçeklenebilirlik ve güvenlik açısından optimize etmek için sıklıkla Komut Sorgusu Sorumluluk Ayrımı (CQRS) deseniyle birlikte kullanılır. Geleneksel veri yönetim sistemlerinde komutlar ve sorgular aynı veri havuzunda çalıştırılır.

Aşağıdaki şekilde olay kaynağı modelinin AWS'de nasıl uygulanabileceği gösterilmektedir. Amazon Kinesis Data Streams, uygulama değişikliklerini olay olarak yakalayan ve bunları AWS'de kalıcı hale getiren merkezi olay deposunun

ana bileşeni olarak hizmet vermektedir.

Amazon S3. Şekilde API Gateway, AWS Lambda ve DynamoDB'den oluşan üç farklı mikro hizmet gösterilmektedir. Oklar olayların akışını göstermektedir: Mikro Hizmet 1 bir olay durumu değişikliği yaşadığında, Kinesis Data Streams'e bir mesaj yazarak bir olay yayınlar. Tüm mikro hizmetler, AWS Lambda'da mesajın bir kopyasını okuyan, mikro hizmet için alaka düzeyine göre filtreleyen ve muhtemelen daha fazla işlem için ileten kendi Kinesis Data Streams uygulamasını çalıştırır. İşleviniz bir hata döndürürse Lambda, işleme başarılı olana veya verilerin süresi dolana kadar yığını yeniden dener. Parçaların durmasını önlemek için olay kaynağı eşlemesini daha küçük bir yığın boyutuyla yeniden deneyecek, yeniden deneme sayısını sınırlayacak veya çok eski kayıtları atacak şekilde yapılandırabilirsiniz. Atılan olayları saklamak için olay kaynağı eşlemesini, başarısız toplu işlerle ilgili ayrıntıları bir Amazon Simple Queue Service (SQS) kuyruğuna veya Amazon Simple Notification Service (SNS) konusuna gönderecek şekilde yapılandırabilirsiniz.



AWS'de olay kaynağı modeli

Amazon S3, tüm mikro hizmetlerde tüm olayları dayanıklı bir şekilde depolar ve hata ayıklama, uygulama durumunu kurtarma veya uygulama değişikliklerini denetleme söz konusu olduğunda tek gerçek kaynağıdır. Kayıtların Kinesis Data Streams uygulamanıza birden fazla kez teslim edilmesinin iki temel nedeni vardır: üretici yeniden denemeleri ve tüketici yeniden denemeleri.
Uygulamanız, bireysel kayıtların birden çok kez işlenmesini öngörmeli ve uygun şekilde ele almalıdır.

Yapılandırma yönetimi

Düzinelerce farklı hizmetin bulunduğu tipik bir mikro hizmet mimarisinde, her hizmetin çeşitli alt hizmetlere ve hizmete veri sunan altyapı bileşenlerine erişmesi gerekir. Örnekler mesaj kuyrukları, veritabanları ve diğer mikro hizmetler olabilir. Temel zorluklardan biri, her bir hizmeti, alt hizmetlere ve altyapıya bağlantı hakkında bilgi sağlamak için tutarlı bir şekilde yapılandırmaktır. Ayrıca yapılandırma, hizmetin çalıştığı ortam hakkında da bilgi içermeli ve yeni yapılandırma verilerini kullanmak için uygulamanın yeniden başlatılması gerekmemelidir.

On İki Faktörlü Uygulama modellerinin üçüncü ilkesi bu konuyu kapsamaktadır: "*On iki faktörlü uygulama yapılandırmayı ortam değişkenlerinde (genellikle env vars veya env olarak kısaltılır) saklar.*" Amazon ECS için ortam değişkenleri, docker run'daki `--env` seçeneğiyle eşleşen environment container definition parametresi kullanılarak konteynere aktarılabilir. Ortam değişkenleri, ortam değişkenlerini içeren bir veya daha fazla dosyayı listelemek için `environmentFiles kapsayıcı` tanım parametresi kullanılarak kapsayıcılarınıza toplu olarak aktarılabilir. Dosya Amazon S3'te barındırılmalıdır. AWS Lambda'da çalışma zamanı, ortam değişkenlerini kodunuz için kullanılabilir hale getirir ve işlev ve çağırma isteği hakkında bilgi içeren ek ortam değişkenleri ayarlar. Amazon EKS için, ortam değişkenlerini ilgili pod'un yapılandırma bildiriminin env alanında tanımlayabilirsiniz. Ortam değişkenlerini kullanmanın farklı bir yolu da ConfigMap kullanmaktır.

Asenkron iletişim ve hafif mesajlaşma

Geleneksel, monolitik uygulamalarda iletişim basittir - uygulamanın bir parçası diğer parçalarla iletişim kurmak için yöntem çağrılarını veya dahili bir olay dağıtım mekanizmasını kullanır. Aynı uygulama ayrıştırılmış mikro hizmetler kullanılarak uygulanırsa, uygulamanın farklı bölümleri arasındaki iletişim ağ iletişimi kullanılarak uygulanmalıdır.

REST tabanlı iletişim

HTTP/S protokolü, mikro hizmetler arasında eşzamanlı iletişimi uygulamanın en popüler yoludur. Çoğu durumda, RESTful API'ler HTTP'yi bir

taşıma katmanı. REST mimari tarzı, durum bilgisi olmayan iletişime, tek tip arayüzlere ve standart yöntemlere dayanır.

API Gateway ile, uygulamaların arka uç hizmetlerinizdeki verilere, iş mantığına veya işlevselliğe erişmesi için "ön kapı" görevi gören bir API oluşturabilirsiniz. API geliştiricileri, AWS veya diğer web hizmetlerinin yanı sıra AWS Cloud'da depolanan verilere erişen API'ler oluşturabilir. API Gateway hizmeti ile tanımlanan bir API nesnesi, bir grup kaynak ve yöntemdir.

Bir kaynak, bir API'nin etki alanı içindeki tiplendirilmiş bir nesnedir ve diğer kaynaklarla ilişkili bir veri modeline veya ilişkilere sahip olabilir. Her kaynak bir veya daha fazla yönteme, yani GET, POST veya PUT gibi standart HTTP fiillerine yanıt verecek şekilde yapılandırılabilir. REST API'leri farklı aşamalara dağıtılabilir, sürümlendirilebilir ve yeni sürümlere klonlanabilir.

API Gateway, trafik yönetimi, yetkilendirme ve erişim kontrolü, izleme ve API sürüm yönetimi dahil olmak üzere yüz binlerce eş zamanlı API çağrısının kabul edilmesi ve işlenmesiyle ilgili tüm görevleri yerine getirir.

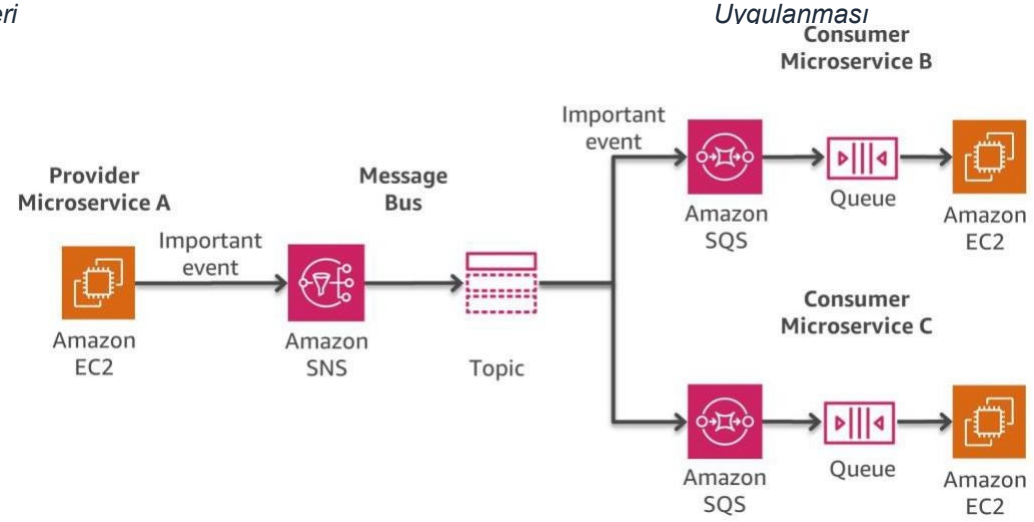
Asenkron mesajlaşma ve olay geçişi

Mesaj geçişi, mikro hizmetler arasındaki iletişimi uygulamak için kullanılan ek bir modeldir. Hizmetler bir kuyruk üzerinden mesaj alışverişi yaparak iletişim kurar. Bu iletişim tarzının en büyük faydalarından biri, bir hizmet keşfinin gerekli olmaması ve hizmetlerin gevşek bir şekilde bağlanmış olmasıdır.

Senkronize sistemler sıkı bir şekilde bağlıdır, bu da senkronize bir aşağı akış bağımlılığındaki bir sorunun yukarı akış arayanları anında etkilediği anlamına gelir. Yukarı yönlü arayanlardan gelen yeniden denemeler hızla yayılabilir ve sorunları büyütebilir.

Protokoller gibi belirli gereksinimlere bağlı olarak AWS, bu modelin uygulanmasına yardımcı olan farklı hizmetler sunar. Olası bir uygulamada Amazon Simple Queue Service (Amazon SQS) ve Amazon Simple Notification Service (Amazon SNS) bir arada kullanılır.

Her iki hizmet de yakın bir şekilde birlikte çalışır. Amazon SNS, uygulamaların bir push mekanizması aracılığıyla birden fazla aboneye mesaj göndermesini sağlar. Amazon SNS ve Amazon SQS birlikte kullanılarak, bir mesaj birden fazla tüketiciye iletilebilir. Aşağıdaki şekilde Amazon SNS ve Amazon SQS'nin entegrasyonu gösterilmektedir.



AWS'de mesaj veri yolu modeli

Bir SQS kuyruğunu bir SNS konusuna abone ettiğinizde, konuya bir mesaj yayınlayabilirsiniz ve Amazon SNS abone olunan SQS kuyruğuna bir mesaj gönderir. Mesaj, JSON biçiminde meta veri bilgileriyle birlikte konu ve konuya yayınlanan mesajı içerir.

Dahili uygulamaları, üçüncü taraf SaaS uygulamalarını ve AWS hizmetlerini kapsayan olay kaynaklarıyla olay odaklı mimariler oluşturmak için bir başka seçenek de Amazon EventBridge'dir. Tam olarak yönetilen bir olay veri yolu hizmeti olan EventBridge, farklı kaynaklardan olayları alır, bir yönlendirme kuralına göre bir hedef belirler ve AWS Lambda, Amazon SNS ve Amazon Kinesis Streams gibi diğerlerinin yanı sıra bu hedefe neredeyse gerçek zamanlı veriler sunar. Gelen bir olay, teslim edilmeden önce girdi dönüştürücüsü tarafından da özelleştirilebilir.

Olay odaklı uygulamaları önemli ölçüde daha hızlı geliştirmek için EventBridge şema kayıtları, AWS hizmetleri tarafından oluşturulan tüm olayların şemaları da dahil olmak üzere şemaları toplar ve düzenler. Müşteriler ayrıca özel şemalar tanımlayabilir veya şemaları otomatik olarak keşfetmek için şema çıkarma seçeneğini kullanabilir. Bununla birlikte, tüm bu özellikler için potansiyel bir değiş tokuş, EventBridge teslimatı için nispeten daha yüksek bir gecikme değeridir. Ayrıca, EventBridge için varsayılan verim ve kotalar, kullanım durumuna bağlı olarak bir destek talebi yoluyla artırılmasını gerektirebilir.

Farklı bir uygulama stratejisi, mevcut yazılım mesajlaşma için JMS, NMS, AMQP, STOMP, MQTT ve WebSocket dahil olmak üzere açık standart API'ler ve protokoller kullanıyorsa kullanılabilecek Amazon MQ'ya dayanmaktadır. Amazon SQS, özel bir

Bu da, geçiş yapmak istediğiniz mevcut bir uygulamanız varsa (örneğin, şirket içi bir ortamdan AWS'ye) kod değişikliklerinin gerekli olduğu anlamına gelir. Amazon MQ ile bu çoğu durumda gerekli değildir.

Amazon MQ, popüler bir açık kaynak mesaj aracısı olan ActiveMQ'nun yönetimini ve bakımını yönetir. Temel altyapı, uygulamalarınızın güvenilirliğini desteklemek için yüksek kullanılabilirlik ve mesaj dayanıklılığı için otomatik olarak sağlanır.

Orkestrasyon ve durum yönetimi

Mikro hizmetlerin dağıtılmış karakteri, birden fazla mikro hizmet söz konusu olduğunda iş akışlarını düzenlemeyi zorlaştırır. Geliştiriciler hizmetlerine doğrudan orkestrasyon kodu ekleme eğiliminde olabilirler. Bu durumdan kaçınılmalıdır çünkü daha sıkı bir bağlantıya neden olur ve tek tek hizmetlerin hızlı bir şekilde değiştirilmesini zorlaştırır.

Her biri ayrı bir işlevi yerine getiren bileşenlerden uygulamalar oluşturmak için AWS Step Functions'ı kullanabilirsiniz. Step Functions, hata işleme, serileştirme ve paralelleştirme gibi hizmet orkestrasyonunun karmaşıklıklarını gizleyen bir durum makinesi sağlar. Bu, hizmetlerin içinde ek koordinasyon kodundan kaçınırken uygulamaları hızla ölçeklendirmenize ve değiştirmenize olanak tanır.

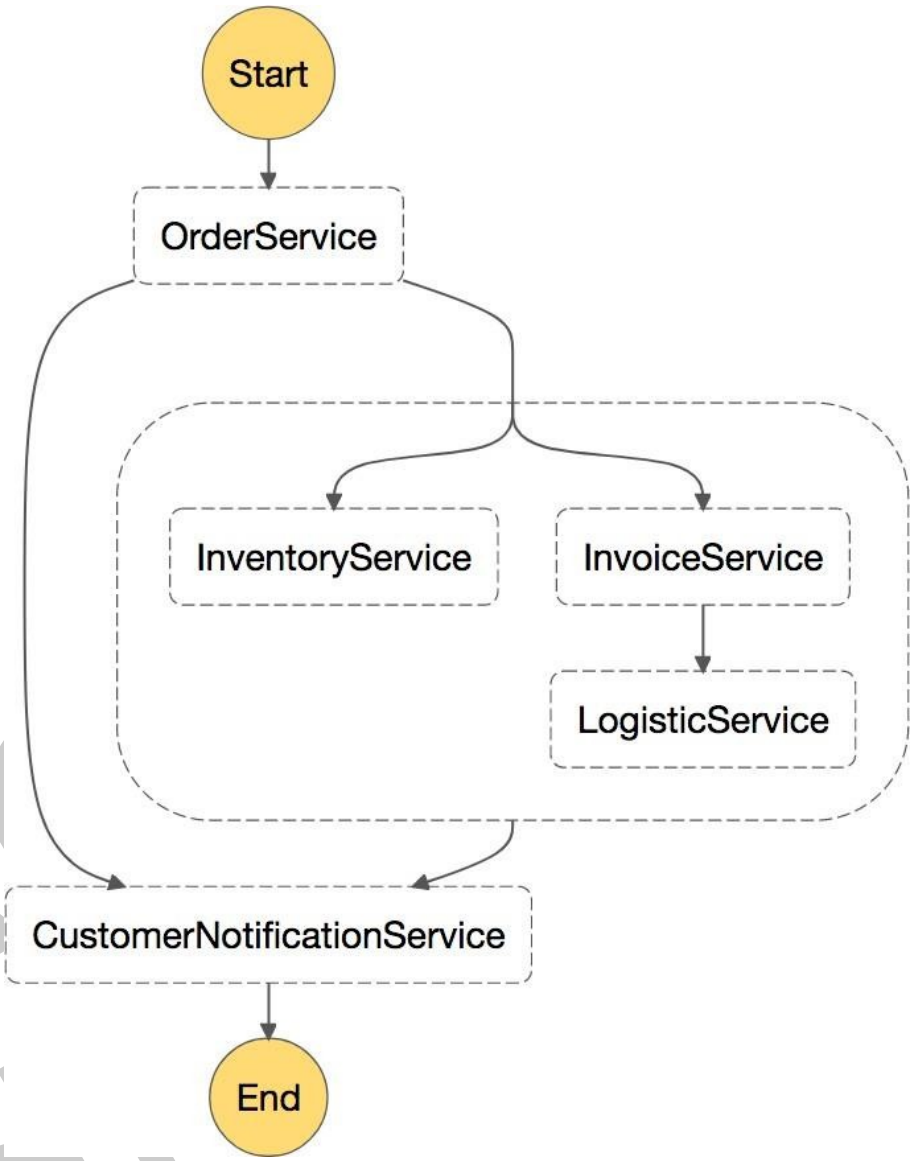
Step Functions, bileşenleri koordine etmenin ve uygulamanızın işlevlerini adım adım gerçekleştirmenin güvenilir bir yoludur. Step Functions, uygulamanızın bileşenlerini bir dizi adım olarak düzenlemek ve görselleştirmek için grafiksel bir konsol sağlar. Bu, dağıtılmış hizmetlerin oluşturulmasını ve çalıştırılmasını kolaylaştırır.

Step Functions her adımı otomatik olarak başlatır, izler ve hata olduğunda yeniden dener, böylece uygulamanız sırayla ve beklendiği gibi yürütülür. Step Functions her adımın durumunu günlüğe kaydeder, böylece bir şeyler ters gittiğinde sorunları hızlı bir şekilde teşhis edebilir ve hata ayıklayabilirsiniz. Uygulamanızı geliştirmek ve daha hızlı yenilik yapmak için kod bile yazmadan adımları değiştirebilir ve ekleyebilirsiniz.

Step Functions, AWS sunucusuz platformunun bir parçasıdır ve Lambda işlevlerinin yanı sıra Amazon EC2, Amazon EKS ve Amazon ECS gibi hesaplama kaynaklarına ve Amazon SageMaker ve AWS Glue gibi ek hizmetlere dayalı uygulamaların düzenlenmesini destekler. Step Functions, uygulamanızın her ölçekte kullanılabilir olmasını sağlamaya yardımcı olmak için işlemleri ve temel altyapıyı sizin için yönetir.

İş akışları oluşturmak için Step Functions, Amazon States Language'i kullanır. İş akışları sıralı veya paralel adımların yanı sıra dallanma adımları da içerebilir.

Aşağıdaki şekilde, sıralı ve paralel adımları birleştiren bir mikro hizmet mimarisi için örnek bir iş akışı gösterilmektedir. Böyle bir iş akışının çağrılması Step Functions API veya API Gateway ile yapılabilir.



Adım İşlevleri tarafından çağrılan bir mikro hizmet iş akışı örneği

Dağıtılmış izleme

Bir mikro hizmet mimarisi, izlenmesi gereken birçok farklı dağıtılmış parçadan oluşur. Metrikleri toplamak ve izlemek, günlük dosyalarını merkezileştirmek ve izlemek, alarmlar ayarlamak ve AWS ortamınızdaki değişikliklere otomatik olarak tepki vermek için Amazon CloudWatch'ı kullanabilirsiniz. CloudWatch, Amazon EC2 örnekleri, DynamoDB tabloları ve Amazon RDS DB örnekleri gibi AWS kaynaklarının yanı sıra uygulamalarınız ve hizmetleriniz tarafından oluşturulan özel metrikleri ve uygulamalarınızın oluşturduğu tüm günlük dosyalarını izleyebilir.

İzleme

Kaynak kullanımı, uygulama performansı ve operasyonel sağlık konularında sistem genelinde görünürlük elde etmek için CloudWatch'ı kullanabilirsiniz. CloudWatch, dakikalar içinde kullanmaya başlayabileceğiniz güvenilir, ölçeklenebilir ve esnek bir izleme çözümü sunar. Artık kendi izleme sistemlerinizi ve altyapınızı kurmanıza, yönetmenize ve ölçeklendirmenize gerek yok. Bir mikro hizmet mimarisinde, CloudWatch kullanarak özel metrikleri izleme özelliği ek bir avantajdır, çünkü geliştiriciler her hizmet için hangi metriklerin toplanması gerektiğine karar verebilir. Buna ek olarak, özel metriklere dayalı olarak dinamik ölçeklendirme uygulanabilir.

Amazon Cloudwatch'a ek olarak, kapsayıcılı uygulamalarınızdan ve mikro hizmetlerinizden metrikleri ve günlükleri toplamak, bir araya getirmek ve özetlemek için CloudWatch Container Insights'ı da kullanabilirsiniz. CloudWatch Container Insights, CPU, bellek, disk ve ağ gibi birçok kaynak için ölçümleri otomatik olarak toplar ve küme, düğüm, pod, görev ve hizmet düzeyinde CloudWatch ölçümleri olarak bir araya getirir. CloudWatch Container Insights'ı kullanarak CloudWatch Container Insights gösterge tablosu ölçümlerine erişebilirsiniz. Ayrıca, sorunları izole etmenize ve hızlı bir şekilde çözmenize yardımcı olmak için kapsayıcı yeniden başlatma hataları gibi tanılama bilgileri de sağlar. Container Insights'ın topladığı metrikler üzerinde CloudWatch alarmları da ayarlayabilirsiniz.

Container Insights, Amazon EC2 üzerindeki Amazon ECS, Amazon EKS ve Kubernetes platformları için kullanılabilir. Amazon ECS desteği Fargate için destek içerir.

Özellikle Amazon EKS için bir başka popüler seçenek de Prometheus kullanmaktır. Prometheus, toplanan metrikleri görselleştirmek için genellikle Grafana ile birlikte kullanılan açık kaynaklı bir izleme ve uyarı araç setidir. Birçok Kubernetes bileşeni `/metrics` adresinde metrikleri depolar ve Prometheus bu metrikleri düzenli aralıklarla kazıyabilir.

Amazon Managed Service for Prometheus (AMP), kapsayıcılı uygulamaları geniş ölçekte izlemenizi sağlayan Prometheus uyumlu bir izleme hizmetidir. AMP ile, operasyonel metriklerin alınması, depolanması ve sorgulanmasını yönetmek için gereken temel altyapıyı yönetmek zorunda kalmadan konteynerli iş yüklerinin performansını izlemek için açık kaynaklı Prometheus sorgu dilini (PromQL) kullanabilirsiniz. OpenTelemetry için AWS Distro veya Prometheus sunucularını toplama aracıları olarak kullanarak Amazon EKS ve Amazon ECS ortamlarından Prometheus metriklerini toplayabilirsiniz.

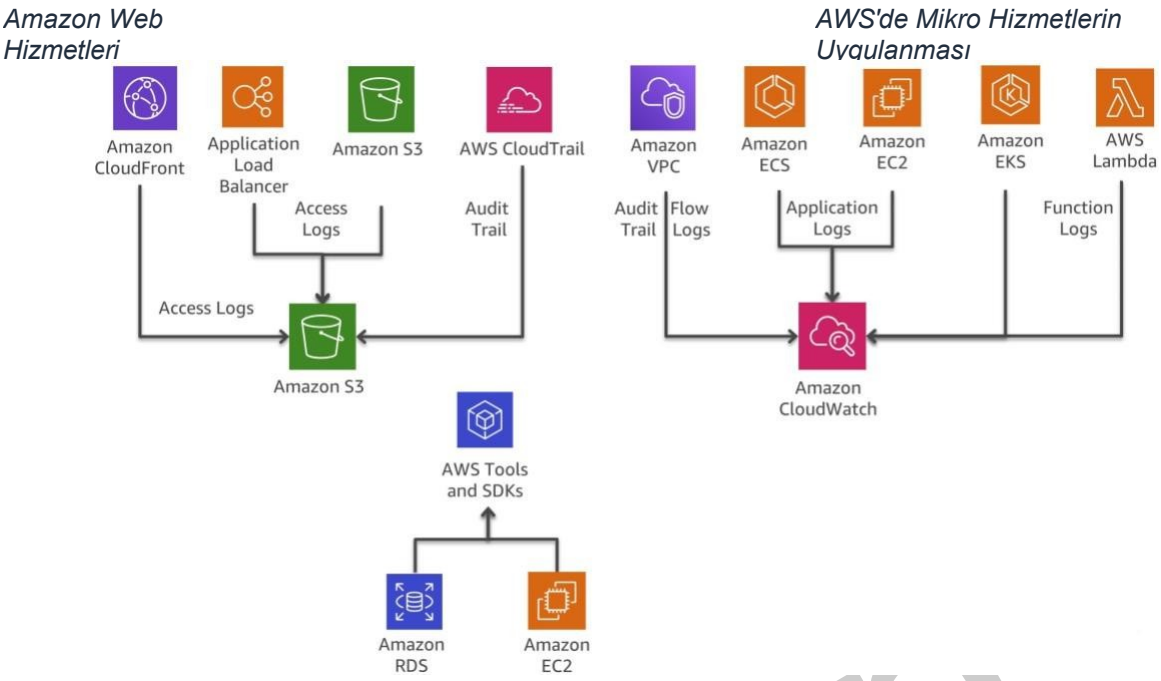
AMP genellikle Amazon Managed Service for Grafana (AMG) ile birlikte kullanılır. AMG, nerede depolandıklarından bağımsız olarak metriklerinizi sorgulamayı, görselleştirmeyi, uyarı vermeyi ve anlamayı kolaylaştırır. AMG ile sunucu sağlamak, yazılımı yapılandırmak ve güncellemek ya da Grafana'yı üretimde güvence altına almak ve ölçeklendirmek gibi ağır işleri yapmak zorunda kalmadan metriklerinizi, günlüklerinizi ve izlerinizi analiz edebilirsiniz.

Günlükleri merkezileştirme

Tutarlı günlük kaydı, sorun giderme ve sorunları tanımlama açısından kritik öneme sahiptir. Mikro hizmetler, ekiplerin her zamankinden çok daha fazla sürüm göndermesini sağlar ve mühendislik ekiplerini üretimde yeni özellikler üzerinde deneyler yapmaya teşvik eder. Müşteri etkisini anlamak, bir uygulamayı kademeli olarak iyileştirmek için çok önemlidir.

Varsayılan olarak, çoğu AWS hizmeti günlük dosyalarını merkezileştirir. AWS'de günlük dosyaları için birincil hedefler Amazon S3 ve Amazon CloudWatch Logs'tur. Amazon EC2 örneklerinde çalışan uygulamalar için, günlük dosyalarını CloudWatch Logs'a göndermek üzere bir daemon mevcuttur. Lambda işlevleri günlük çıktılarını yerel olarak CloudWatch Logs'a gönderir ve Amazon ECS, konteyner günlüklerinin CloudWatch Logs'a merkezileştirilmesini sağlayan awslogs günlük sürücüsü için destek içerir. Amazon EKS için Fluent Bit ya da Fluentd, kümedeki ayrı örneklerden gelen günlükleri, Amazon OpenSearch Service ve Kibana kullanılarak daha üst düzey raporlama için birleştirildikleri merkezi bir günlük kaydı CloudWatch Logs'a iletebilir. Daha az yer kaplaması ve performans avantajları nedeniyle Fluentd yerine Fluent Bit önerilir.

Aşağıdaki şekilde bazı hizmetlerin günlük tutma özellikleri gösterilmektedir. Ekipler daha sonra Amazon OpenSearch Service ve Kibana gibi araçları kullanarak bu günlükleri arayabilir ve analiz edebilir. Amazon Athena, Amazon S3'teki merkezi günlük dosyalarına karşı tek seferlik bir sorgu çalıştırmak için kullanılabilir.



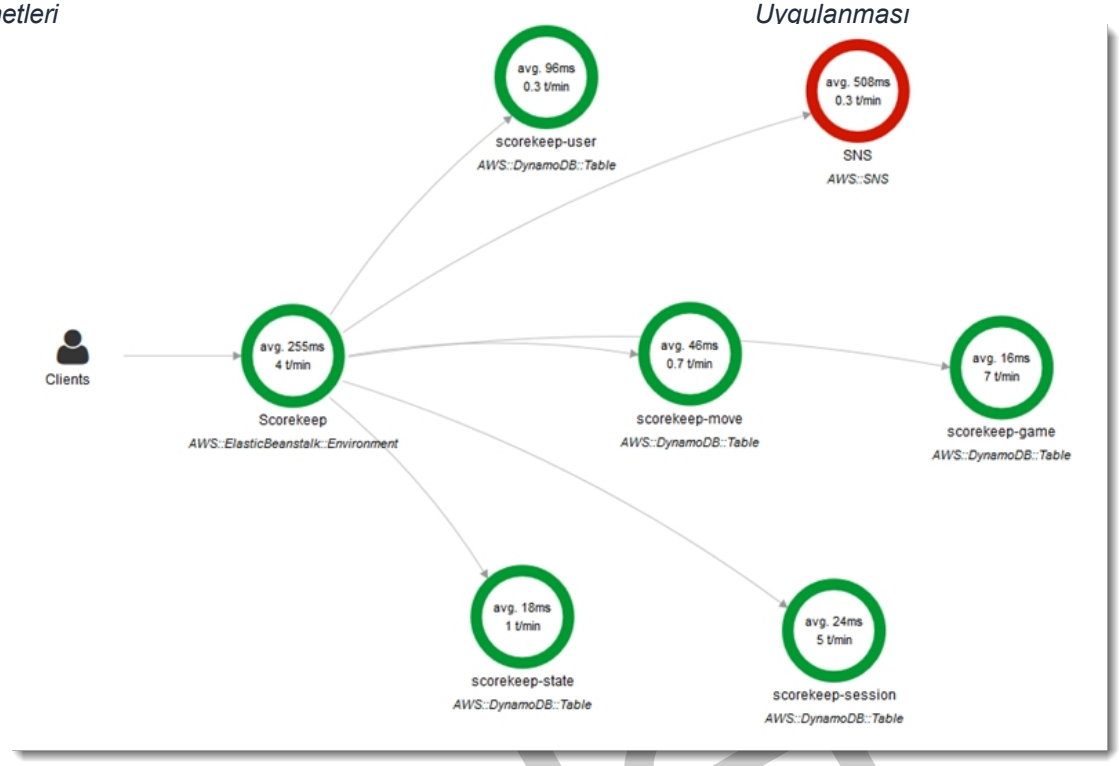
AWS hizmetlerinin günlük tutma özellikleri

Dağıtılmış izleme

Birçok durumda, bir dizi mikro hizmet bir isteği işlemek için birlikte çalışır. Çağrı zincirindeki hizmetlerden birinde bir hatanın meydana geldiği onlarca mikro hizmetten oluşan karmaşık bir sistem düşünün. Her mikro hizmet düzgün bir şekilde günlük tutuyor ve günlükler merkezi bir sistemde birleştiriliyor olsa bile, ilgili tüm günlük mesajlarını bulmak zor olabilir.

AWS X-Ray'in ana fikri, belirli bir olay zinciriyle ilgili tüm isteklere ve mesajlara eklenen benzersiz tanımlayıcılar olan korelasyon kimliklerinin kullanılmasıdır. İzleme kimliği, istek ilk X-Ray entegre hizmetine (örneğin, Uygulama Yük Dengeleyici veya API Ağ Geçidi) ulaştığında `X-Amzn-Trace-Id` adlı belirli izleme başlıklarında HTTP isteklerine eklenir ve yanıta dahil edilir. X-Ray SDK aracılığıyla herhangi bir mikro servis bu başlığı okuyabilir, aynı zamanda ekleyebilir veya güncelleyebilir.

X-Ray, Amazon EC2, Amazon ECS, AWS Lambda ve AWS Elastic Beanstalk ile çalışır. X-Ray'i bu hizmetlere dağıtılan Java, Node.js ve .NET ile yazılmış uygulamalarla kullanabilirsiniz.



X-Ray hizmet haritası

Epsagon, tüm AWS hizmetleri, üçüncü taraf API'leri (HTTP çağrıları aracılığıyla) ve Redis, Kafka ve Elastic gibi diğer yaygın hizmetler için izleme içeren tam olarak yönetilen bir SaaS'tır. Epsagon hizmeti, izleme yetenekleri, en yaygın hizmetlere yönelik uyarılar ve kodunuzun yaptığı her bir çağrıya yönelik yük görünürlüğü içerir.

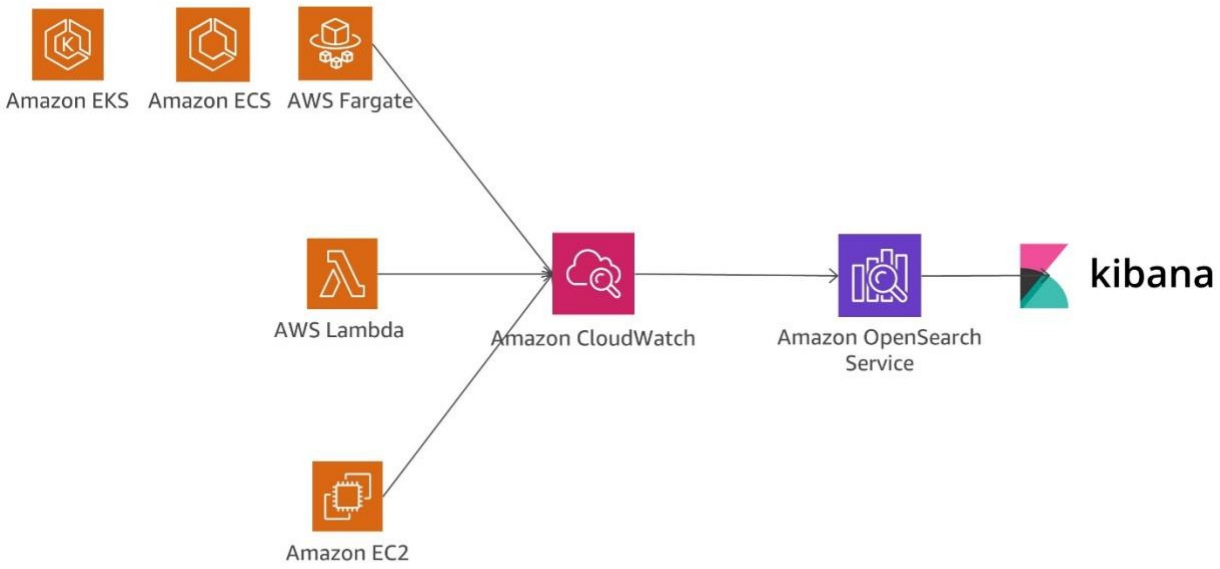
AWS Distro for OpenTelemetry, OpenTelemetry projesinin güvenli, üretime hazır, AWS destekli bir dağıtımıdır. Cloud Native Computing Foundation'ın bir parçası olan AWS Distro for OpenTelemetry, uygulama izlemeye yönelik dağıtılmış izleri ve ölçümleri toplamak için açık kaynaklı API'ler, kütüphaneler ve aracılar sağlar. AWS Distro for OpenTelemetry ile, birden fazla AWS ve iş ortağı izleme çözümüne ilişkili metrikler ve izler göndermek için uygulamalarınızı yalnızca bir kez enstrümante edebilirsiniz. Kodunuzu değiştirmeden izleri toplamak için otomatik enstrümantasyon aracılarını kullanın. AWS Distro for OpenTelemetry ayrıca AWS kaynaklarınızdan ve yönetilen hizmetlerden meta veriler toplayarak uygulama performans verilerini altta yatan altyapı verileriyle ilişkilendirir ve sorun çözümüne kadar geçen ortalama süreyi kısaltır. Amazon EC2, Amazon ECS, Amazon EKS on Amazon EC2, Fargate ve AWS Lambda üzerinde ve ayrıca şirket içinde çalışan uygulamalarınızı enstrümante etmek için AWS Distro for OpenTelemetry'yi kullanın.

AWS'de günlük analizi için seçenekler

Günlük verilerini aramak, analiz etmek ve görselleştirmek, dağıtık sistemleri anlamanın önemli bir yönüdür. Amazon CloudWatch Logs Insights, günlüklerinizi anında keşfetmenizi, analiz etmenizi ve görselleştirmenizi sağlar. Bu sayede operasyonel sorunları giderebilirsiniz. Log dosyalarını analiz etmek için bir başka seçenek de Amazon OpenSearch Service'i Kibana ile birlikte kullanmaktır.

Amazon OpenSearch Hizmeti tam metin arama, yapılandırılmış arama, analitik ve her üçü bir arada kullanılabilir. Kibana, Amazon OpenSearch Service ile sorunsuz bir şekilde entegre olan açık kaynaklı bir veri görselleştirme eklentisidir.

Aşağıdaki şekilde Amazon OpenSearch Service ve Kibana ile günlük analizi gösterilmektedir. CloudWatch Logs, bir CloudWatch Logs aboneliği aracılığıyla günlük girişlerini neredeyse gerçek zamanlı olarak Amazon OpenSearch Service'e aktaracak şekilde yapılandırılabilir. Kibana verileri görselleştirir ve Amazon OpenSearch Service'teki veri depolarına uygun bir arama arayüzü sunar. Bu çözüm, verilerde anormallikler, ani artışlar veya diğer ilgi çekici modeller tespit edilirse SNS bildirimleri ve e-postaları göndermek, JIRA biletleri oluşturmak vb. için bir uyarı sistemi uygulamak üzere ElastAlert gibi yazılımlarla birlikte kullanılabilir.



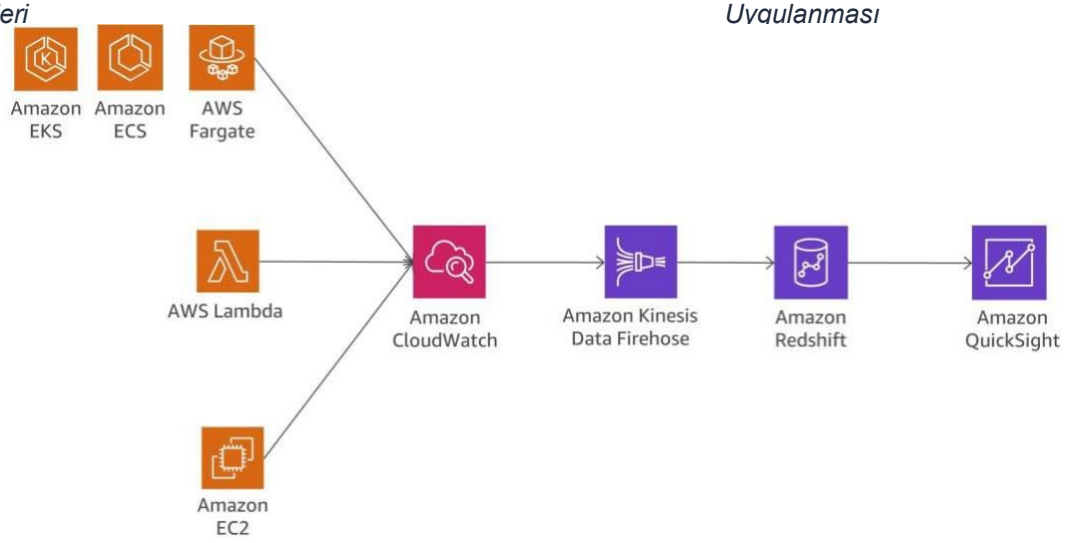
Amazon OpenSearch Service ve Kibana ile günlük analizi

Günlük dosyalarını analiz etmek için bir başka seçenek de Amazon QuickSight ile Amazon Redshift'i kullanmaktır.

QuickSight, Redshift, Amazon RDS, Aurora, Amazon EMR, DynamoDB, Amazon S3 ve Amazon Kinesis dahil olmak üzere AWS veri hizmetlerine kolayca bağlanabilir.

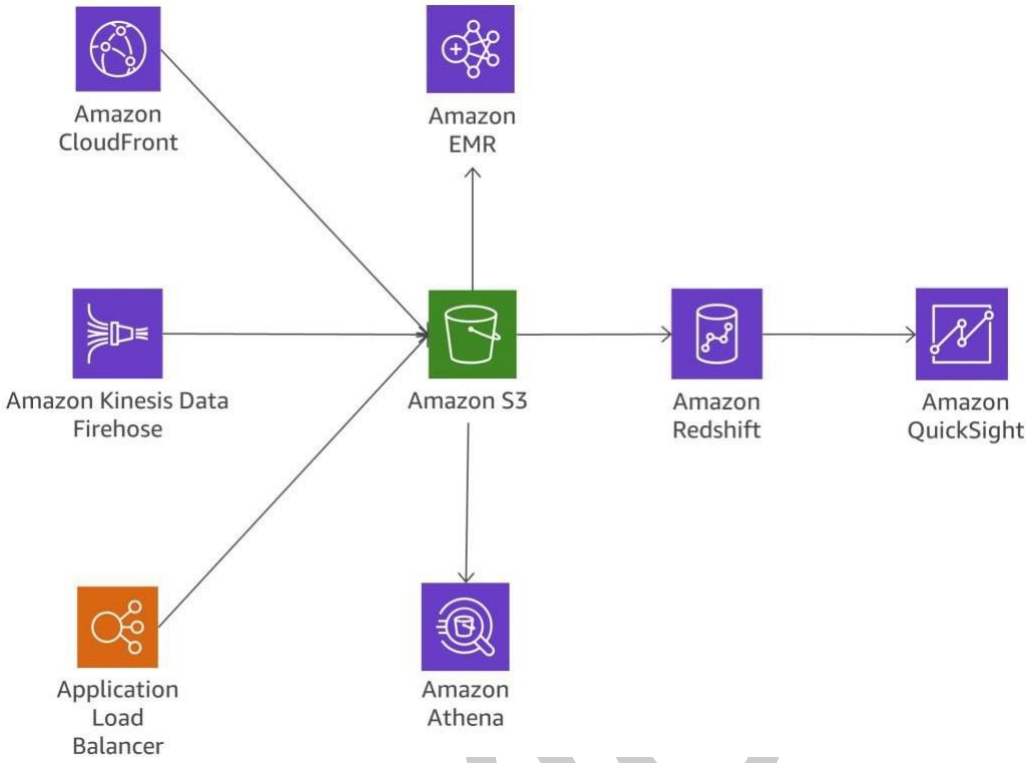
CloudWatch Logs, günlük verileri için merkezi bir depo görevi görebilir ve yalnızca verileri depolamanın yanı sıra günlük girdilerini Amazon Kinesis Data Firehose'a aktarmak da mümkündür.

Aşağıdaki şekilde, günlük girdilerinin CloudWatch Logs ve Kinesis Data Firehose kullanılarak farklı kaynaklardan Redshift'e aktarıldığı bir senaryo gösterilmektedir. QuickSight, Redshift'te depolanan verileri analiz, raporlama ve görselleştirme için kullanır.



Amazon Redshift ve Amazon QuickSight ile günlük analizi

Aşağıdaki şekilde Amazon S3 üzerinde bir log analizi senaryosu gösterilmektedir. Günlükler Amazon S3 kovalarında depolandığında, günlük verileri Redshift veya Amazon EMR gibi farklı AWS veri hizmetlerine yüklenerek günlük akışında depolanan veriler analiz edilebilir ve anormallikler bulunabilir.



Amazon S3 üzerinde günlük analizi

Sohbet

Monolitik uygulamaların küçük mikro hizmetlere bölünmesiyle, mikro hizmetlerin birbirleriyle konuşması gerektiğinden iletişim yükü artar. Birçok uygulamada, hafif bir iletişim protokolü olduğu için HTTP üzerinden REST kullanılır, ancak yüksek mesaj hacimleri sorunlara neden olabilir. Bazı durumlarda, ileri geri çok sayıda mesaj gönderen hizmetleri birleştirmeyi düşünebilirsiniz. Kendinizi sadece mesajlaşmayı azaltmak için artan sayıda hizmeti birleştirdiğiniz bir durumda bulursanız, sorunlu alanlarınızı ve alan modelinizi gözden geçirmelisiniz.

Protokoller

Bu whitepaper'ın başlarında, *Asenkron iletişim ve hafif mesajlaşma* bölümünde, farklı olası protokoller tartışılmıştır. Mikro hizmetler için HTTP gibi protokollerin kullanılması yaygındır. Hizmetler tarafından değiş tokuş edilen mesajlar, JSON veya YAML gibi insan tarafından okunabilir formatlar veya Avro veya Protokol Tamponları gibi verimli ikili formatlar gibi farklı şekillerde kodlanabilir.

Önbellekleme

Önbellekler, mikro hizmet mimarilerinin gecikmesini ve tıkanıklığını azaltmak için harika bir yoldur. Gerçek kullanım durumuna ve darboğazlara bağlı olarak çeşitli önbellekleme katmanları mümkündür. AWS üzerinde çalışan birçok mikro hizmet uygulaması, sonuçları yerel olarak önbelleğe alarak diğer mikro hizmetlere yapılan çağrıların hacmini azaltmak için ElastiCache kullanır. API Gateway, arka uç sunucular üzerindeki yükü azaltmak için yerleşik bir önbellekleme katmanı sağlar. Ayrıca önbelleğe alma, veri kalıcılığı katmanından gelen yükü azaltmak için de yararlıdır. Herhangi bir önbellekleme mekanizması için zorluk, iyi bir önbellek isabet oranı ile verilerin güncelliği ve tutarlılığı arasında doğru dengeyi bulmaktır.

Denetim

Potansiyel olarak yüzlerce dağıtılmış hizmete sahip olabilen mikro hizmet mimarilerinde ele alınması gereken bir diğer zorluk, her bir hizmet üzerindeki kullanıcı eylemlerinin görünürlüğünü sağlamak ve kurumsal düzeyde tüm hizmetler genelinde iyi bir genel görünüm elde edebilmektir. Güvenlik politikalarının uygulanmasına yardımcı olmak için hem kaynak erişimini hem de sistem değişikliklerine yol açan faaliyetleri denetlemek önemlidir.

Değişiklikler, bireysel hizmet düzeyinde ve daha geniş sistemde çalışan hizmetler arasında izlenmelidir. Tipik olarak, mikro hizmet mimarilerinde sık sık değişiklikler meydana gelir ve bu da değişikliklerin denetlenmesini daha da önemli hale getirir. Bu bölümde, AWS'de mikro hizmet mimarinizi denetlemenize yardımcı olabilecek temel hizmetler ve özellikler incelenmektedir.

Denetim izi

AWS CloudTrail, mikro hizmetlerdeki değişiklikleri izlemek için kullanışlı bir araçtır çünkü AWS Cloud'da yapılan tüm API çağrılarının günlüğe kaydedilmesini ve gerçek zamanlı olarak CloudWatch Günlüklerine veya birkaç dakika içinde Amazon S3'e gönderilmesini sağlar.

Tüm kullanıcı ve otomatik sistem eylemleri aranabilir hale gelir ve beklenmedik davranışlar, şirket politikası ihlalleri veya hata ayıklama için analiz edilebilir. Kaydedilen bilgiler arasında bir zaman damgası, kullanıcı ve hesap bilgileri, çağrılan hizmet, talep edilen hizmet eylemi, arayanın IP adresi, istek parametreleri ve yanıt öğeleri yer alır.

CloudTrail aynı hesap için birden fazla iz tanımlanmasına izin vererek güvenlik yöneticileri, yazılım geliştiriciler veya BT gibi farklı paydaşların

denetçiler, kendi izlerini oluşturmak ve yönetmek için. Mikro hizmet ekiplerinin farklı AWS hesapları varsa, izleri tek bir S3 kovasında toplamak mümkündür.

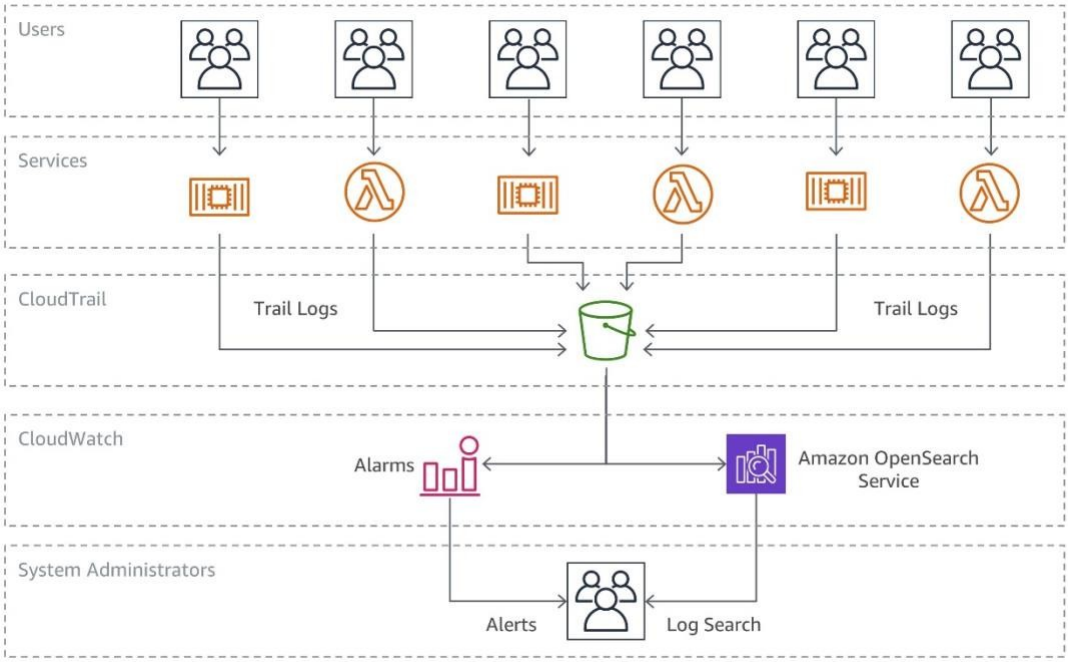
Denetim izlerini CloudWatch'ta depolamanın avantajları, denetim izi verilerinin gerçek zamanlı olarak yakalanması ve arama ve görselleştirme için bilgileri Amazon OpenSearch Hizmetine yeniden yönlendirmenin kolay olmasıdır. CloudTrail'i hem Amazon S3 hem de CloudWatch Günlüklerinde oturum açacak şekilde yapılandırabilirsiniz.

Olaylar ve gerçek zamanlı eylemler

Sistem mimarilerindeki belirli değişikliklere hızlı bir şekilde yanıt verilmeli ve durumu düzeltmek için harekete geçilmeli ya da değişikliği yetkilendirmek için belirli yönetişim prosedürleri başlatılmalıdır. Amazon CloudWatch Events'in CloudTrail ile entegrasyonu, tüm AWS hizmetlerindeki tüm değişen API çağrıları için olay oluşturmasına olanak tanır. Özel olaylar tanımlamak veya sabit bir programa göre olaylar oluşturmak da mümkündür.

Bir olay ateşlendiğinde ve tanımlanmış bir kuralla eşleştiğinde, kuruluşunuzdaki önceden tanımlanmış bir grup insan derhal bilgilendirilebilir, böylece uygun eylemi gerçekleştirebilirler. Gerekli eylem otomatikleştirilebiliyorsa, kural otomatik olarak yerleşik bir iş akışını tetikleyebilir veya sorunu çözmek için bir Lambda işlevini çağırabilir.

Aşağıdaki şekilde, CloudTrail ve CloudWatch Events'in bir mikro hizmet mimarisindeki denetim ve düzeltme gereksinimlerini karşılamak için birlikte çalıştığı bir ortam gösterilmektedir. Tüm mikro hizmetler CloudTrail tarafından izlenmekte ve denetim izi bir Amazon S3 kovasında saklanmaktadır. CloudWatch Events, meydana geldikçe operasyonel değişikliklerden haberdar olur. CloudWatch Events, ortama yanıt vermek için mesajlar göndererek, işlevleri etkinleştirerek, değişiklikler yaparak ve durum bilgilerini yakalayarak bu operasyonel değişikliklere yanıt verir ve gerektiğinde düzeltici eylemde bulunur.
CloudWatch Events, CloudTrail'in üzerine oturur ve mimarinizde belirli bir değişiklik yapıldığında uyarıları tetikler.



Denetim ve iyileştirme

Kaynak envanteri ve değişiklik yönetimi

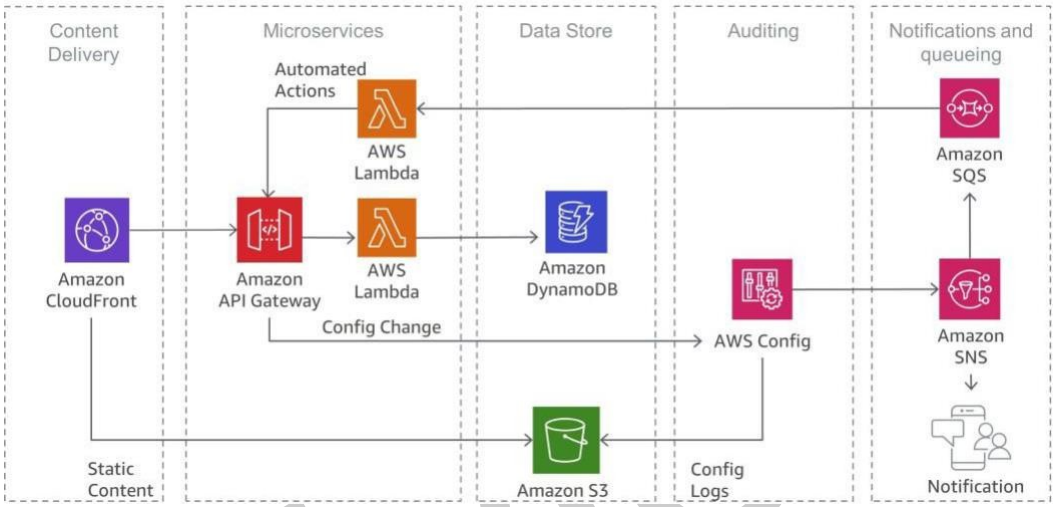
Çevik bir geliştirme ortamında hızla değişen altyapı yapılandırmaları üzerinde kontrol sağlamak için, mimarinizi denetlemek ve kontrol etmek için daha otomatik, yönetilen bir yaklaşıma sahip olmak çok önemlidir.

CloudTrail ve CloudWatch Events, mikro hizmetlerdeki altyapı değişikliklerini izlemek ve bunlara yanıt vermek için önemli yapı taşları olsa da AWS Config kuralları, bir şirketin politika ihlallerini otomatik olarak tespit etmek, izlemek ve sizi uyarmak için belirli kurallarla güvenlik politikaları tanımlamasına olanak tanır.

Bir sonraki örnek, mikro hizmet mimarinizdeki uyumlu olmayan yapılandırma değişikliklerini tespit etmenin, bilgilendirmenin ve bunlara otomatik olarak tepki vermenin nasıl mümkün olduğunu göstermektedir. Geliştirme ekibinin bir üyesi, uç noktanın yalnızca HTTPS isteklerine izin vermek yerine gelen HTTP trafiğini kabul etmesine izin vermek için bir mikro hizmet için API Ağ Geçidinde bir değişiklik yaptı.

Bu durum daha önce kuruluş tarafından bir güvenlik uyumluluğu endişesi olarak tanımlandığından, bir AWS Config kuralı zaten bu durumu izliyor.

Kural, değişikliği bir güvenlik ihlali olarak tanımlar ve iki eylem gerçekleştirir: denetim için bir Amazon S3 kovasında tespit edilen değişikliğin günlüğünü oluşturur ve bir SNS bildirimi oluşturur. Amazon SNS, senaryomuzda iki amaç için kullanılır: güvenlik ihlali hakkında bilgi vermek için belirli bir gruba e-posta göndermek ve bir SQS kuyruğuna mesaj eklemek. Ardından, mesaj alınır ve API Gateway yapılandırması değiştirilerek uyumlu durum geri yüklenir.



AWS Config ile güvenlik ihlallerini tespit etme

Kaynaklar

- AWS Mimari Merkezi
- AWS Beyaz Bültenleri
- Aylık AWS Mimarisi
- AWS Mimari Blogu
- This Is My Architecture videoları
- AWS Yanıtları
- AWS Dokümantasyonu

Sonuç

Mikro hizmetler mimarisi, geleneksel monolitik mimarilerin sınırlamalarının üstesinden gelmeyi amaçlayan dağıtılmış bir tasarım yaklaşımıdır. Mikro hizmetler, döngü sürelerini iyileştirirken uygulamaları ve kuruluşları ölçeklendirmeye yardımcı olur. Bununla birlikte, ek mimari karmaşıklık ve operasyonel yük getirebilecek birkaç zorluğu da beraberinde getirirler.

AWS, ürün ekiplerinin mikro hizmet mimarileri oluşturmasına ve mimari ve operasyonel karmaşıklığı en aza indirmesine yardımcı olabilecek geniş bir yönetilen hizmetler portföyü sunar. Bu teknik doküman, ilgili AWS hizmetleri ve hizmet keşfi veya olay kaynak kullanımı gibi tipik modellerin AWS hizmetleriyle yerel olarak nasıl uygulanacağı konusunda size rehberlik edecektir.

Belge Revizyonlar

Tarih	Açıklama
9 Kasım 2021	Amazon EventBridge, AWS OpenTelemetry, AMP, AMG, Container Insights entegrasyonu, küçük metin değişiklikleri.
1 Ağustos 2019	Küçük metin değişiklikleri.
1 Haziran 2019	Amazon EKS, AWS Fargate, Amazon MQ, AWS PrivateLink, AWS App Mesh, AWS Cloud Map entegrasyonu
1 Eylül 2017	AWS Step Functions, AWS X-Ray ve ECSevent akışlarının entegrasyonu.
1 Aralık 2016	İlk yayın

Katkıda Bulunanlar

Aşağıdaki kişiler bu belgeye katkıda bulunmuştur:

- Sascha Möllering, Çözüm Mimarisi, AWS
- Christian Müller, Çözüm Mimarisi, AWS
- Matthias Jung, Çözüm Mimarisi, AWS
- Peter Dalbhanjan, Çözüm Mimarisi, AWS
- Peter Chapman, Çözüm Mimarisi, AWS
- Christoph Kassen, Çözüm Mimarisi, AWS

- Umair Ishaq, Çözüm Mimarisi, AWS
- Rajiv Kumar, Çözüm Mimarisi, AWS