CodeCommit: Veriyi depolama aşamasında şifreler. Örneğin, bir "push" işlemi gerçekleştirdiğimizde, CodeCommit, depolanan veriyi şifreler. Veri çekildiğinde ise CodeCommit bu veriyi şifresini çözer. CodeCommit ile iletişim kurarken SSH veya HTTPS protokollerini kullanırız ve bu iletişim sırasında veri şifrelenir. AWS Yönetim Konsolu'nda "Key Management Service" bölümüne giderseniz, CodeCommit için bir anahtar olduğunu görebilirsiniz. KMS anahtarları bölgesel bazda çalışır, yani bir bölgede bir veritabanı oluşturduğunuzda o bölgede bir anahtar oluşturulur. Başka bir bölgede veritabanı oluşturmadığınız sürece o bölgede bir anahtar olmaz.

B**uildspec:** Build sürecinin adımlarını net bir şekilde tanımlamanıza ve her adımın hangi komutları çalıştıracağını belirlemenize olanak tanır. Bu sayede build sürecinizi özelleştirebilir ve uygulamanızı başarıyla derleyip dağıtabilirsiniz.

CodeDeploy: CodeDeploy'un ana amacı dağıtımları otomatikleştirmektir. CodeDeploy'a erişimi olan bir kullanıcı sağlamak isteyeceğiz. Tekrar CodeCommit'e atıfta bulunacağım, kod CodeCommit'te olduğu için CodeBuild bu kodu alır ve derleme yapar. Çıktı artefaktımız, derleme paketi S3'e yerleştirilir. Bu yüzden yukarıdaki diyagramda biraz karmaşık hale getirdim, çünkü benim aklım bu durumda CodeCommit'tden CodeBuild'a geçmek istiyor. Son olarak, dağıtımın nasıl gerçekleşeceğini belirlemek için bir AppSpec dosyası oluştururuz.

AppSpec: Dağıtım sırasında yapılması gereken adımları tanımlar. Bu adımları tamamladıktan sonra, dağıtımı başlatmak için CodeDeploy konsolunda bir dağıtım oluşturabiliriz. Revizyon dosyaları AppSpec.yml dosyasıyla aynı dizinde olmalıdır. AppSpec.yml dosyası, dağıtımın nasıl yapılacağını ve hangi kaynak dosyalarının hedef sistemlere yerleştirileceğini belirten bir konfigürasyon dosyasıdır.

CodePipeline: Yazılımınızı otomatik olarak derleme, test etme ve dağıtma konularında size yardımcı olabilir. Pipeline, kodumuzu otomatikleştirerek, kodumuzu ortamlarımıza, örneğin üretim ortamlarına hızlıca dağıtmamızı sağlar.

EventBridge, CodePipeline ile birlikte kullanılabilir ve EventBridge bir Lambda işlevini tetikleyebilir. Bir pipeline'ı oluşturmak için farklı aşamaları birleştirmemiz gerekmektedir. Kaynak aşaması, yapılan değişiklikleri içerir ve bu değişiklikler otomatik olarak veya manuel onaylarla tetiklenebilir.

Sonuç olarak, CodePipeline sürekli dağıtım süreçlerini otomatikleştirmemize yardımcı olan bir hizmettir. Pipeline'ımızı oluştururken kaynak, derleme, dağıtma gibi aşamaları bir araya getirir ve kodumuzun otomatik olarak derlenmesini, test edilmesini ve hedef ortama dağıtılmasını sağlar.

CodePipeline, varsayılan olarak hala CloudWatch Events'i kullanıyor. Bu nedenle olaylar ve EventBridge genel anlamda birbirinin yerine kullanılabilir. CodePipeline, bir kaynağın durumu değiştiğinde CloudWatch Events'e bir olay bildirir. Bu olaylar, pipeline yürütme ve aşama yürütme için garanti edilmiş en az bir kez temelinde gönderilir. Olaylar, aşama yürütmelerindeki eylemler gibi pipeline içindeki aksiyonlarla birlikte çalışabilir.

Jenkins: CodePipeline ile etkileşimde olduğunda, geliştirici kodu AWS CodeCommit'e gönderir. CodePipeline, yeni kodu algılar ve Jenkins'i çağırarak uygulamayı derler. Ardından CodePipeline, CodeDeploy'u tetikleyerek uygulamayı EC2 örneğine dağıtır.

Sınav ipuçları olarak, mevcut Jenkins yapı işlerini CodeBuild ile nasıl entegre edebileceğimiz ve Jenkins'in bir iş akışında hangi aşamalarda çalışabileceği gibi konulara dikkat etmek önemlidir.

AWS CodeArtifact: CodeArtifact bir yazılım sanal deposudur. Daha basit bir şekilde ifade etmek gerekirse, CodeArtifact bir yazılım paketi yönetim aracıdır ve geliştirme ve uygulamalarımızda kullanılmak üzere yazılım paketlerini ve kütüphanelerini depolamamıza olanak sağlar. Paketler birden çok farklı yerden gelebilir. Örneğin, Java, .NET, C#, JavaScript ya da Python gibi in-house geliştirmelerimizden kaynaklanan paketleri CodeArtifact'ta depolayabiliriz. Maven için Java, Gradle için .NET (Gradle birden çok dil ile çalışabilir), NuGet npm için JavaScript ve pip için Python örneklerini gösteriyorum. Ayrıca, CodeArtifact, genel paket depoları için bir ayna olarak da işlev görebilir. Bu da CodeArtifact'in gücünü genişletir. Yani, genel paket depolarından paketleri çekebiliriz.

EC2 Image Builder kullanarak EC2 İmaj Oluşturmayı Otomatikleştirme: EC2 Image Builder ile Amazon Makine İmajı (AMI) olarak adlandırılan temel imajları otomatik olarak oluşturabiliriz. Bu imajlar, güncel güvenlik düzeltmeleri ve sıkılaştırmalarla standart hale getirilebilir ve her zaman kullanmak istediğimiz imajlardır.

EC2 Image Builder'ı kullanmayı daha iyi anlamak için denemek en iyi yoldur. Yönetim Konsolu'na giderek Image Builder'a girip bir imaj süreci oluşturabiliriz. İmaj süreci oluştururken belirli bir zaman diliminde çalışan bir zamanlanmış süreç oluşturabiliriz.

Amazon CodeGuru: Amazon CodeGuru, yazılım geliştiricilere kod kalitesini iyileştirmek ve uygulamanın en maliyetli kod satırlarını belirlemek için akıllı öneriler sunan bir yazılım geliştirme aracıdır.

CodeGuru Profiler, uygulama performansını ayarlamak ve operasyonel sorunları tespit etmek için etkileşimli görselleştirmeler ve öneriler sunar. Profiler, yapı ve test aşamalarında veya canlı ortamda kullanılabilir. Kod kalitesini artırmak ve performansı optimize etmek için geliştirme sürecinin farklı aşamalarında kullanılabilir. Geliştiriciler kod yazdıktan sonra, CodeGuru Reviewer kod incelemesi yapabilir.

CodeGuru'nun bir diğer önemli özelliği de Secrets Detector'dır. Secrets Detector, kod incelemesi yaparken yapılandırma dosyalarında veya kodda saklanan sırları (örneğin, şifreler veya kimlik bilgileri) tespit edebilir ve bu sırların Secrets Manager gibi güvenli bir ortamda saklanması gerektiğini önerir.

SDLC (Yazılım Geliştirme Yaşam Döngüsü): AWS App Runner, önceden altyapı veya konteyner deneyimi gerektirmeksizin konteynerize web uygulamalarını ve API hizmetlerini oluşturmanıza, dağıtmanıza ve çalıştırmanıza olanak tanıyan tamamen yönetilen bir konteyner uygulama

hizmetidir. Bu, AWS'in altyapıyı yönettiği anlamına gelir. App Runner, AWS Fargate ile benzerlik gösterir. Ancak, App Runner'da ölçeklendirme için kullanılan temel boyut aynı anda çalışan isteklerdir.

CloudShell: AWS yönetim konsolundan doğrudan başlatabileceğiniz tarayıcı tabanlı bir ön yetkilendirilmiş kabuktur. CloudShell'i kullanarak Bash, PowerShell veya Z Shell gibi tercih ettiğiniz kabukta AWS CLI komutlarını çalıştırabilirsiniz. CloudShell, tarayıcı tabanlı olduğu için bir EC2 örneğine ihtiyaç duymaz ve SSH veya örnek bağlantısı yapmanıza gerek kalmaz.

CodeStar: AWS üzerinde hızla uygulama geliştirmeniz için gerekli araçları sağlayan bulut tabanlı bir geliştirme hizmetidir. CodeStar, CodeCommit, CodeBuild, CodePipeline ve CodeDeploy'ı önceden tanımlanmış şablonlar kullanarak birbirine bağlar. CodeStar, bu hizmetlerin üzerinde bir çatı görevi görür ve bir kontrol paneli gibi düşünülebilir. CodeStar, EC2, AWS Lambda ve Elastic Beanstalk gibi çeşitli proje şablonlarını kullanarak uygulama geliştirmenize olanak tanır.

Son olarak, S3'ü bir depo olarak kullanma konusu gündeme geliyor. Adayın belirttiği gibi, S3 bir depo olarak kullanılabilir, ancak depo özelliğini etkinleştirmeniz gerekebilir. Mülakatçı, S3'ü otomatik olarak bir depo olarak kullanamayacağınızı belirtiyor. Bu durumu göz önünde bulundurarak doğru cevabı bulabilirsiniz.

Elastic Beanstalk: Elastic Beanstalk ile uygulamalarımızı yüklemek için öncelikle bir uygulama oluştururuz. Bu uygulama, bileşenler, ortamlar, sürümler ve yapılandırmalar gibi unsurların bir koleksiyonudur. Daha sonra bir uygulama sürümü yükleriz, bu genellikle bir uygulama kaynak paketi olarak düşünülebilir, örneğin bir WAR dosyası veya JAR dosyası. Elastic Beanstalk, bu uygulama sürümüne göre otomatik olarak ortamı oluşturur ve AWS kaynaklarını yapılandırır ve oluşturur.

Elastic Beanstalk, kapasite sağlama ve yük dengelemesi gibi konular üzerinde bazı kontroller sunar, ancak CloudFormation gibi detaylı bir şekilde yapılandırmanızı gerektirmez. Varsayılan seçeneklerle başlayabilir ve isterseniz altyapı kaynaklarını yapılandırabilirsiniz. Elastic Beanstalk'in temel kullanım durumlarından biri, AWS hakkında deneyimi olmayan bir yazılım şirketinin hızlı bir şekilde uygulamalarını dağıtabilmesidir. Elastic Beanstalk, altyapıyla ilgilenmek yerine sadece uygulamanızı yüklemenizi sağlar. Ayrıca test ortamları için hızlı bir şekilde ortam oluşturmak da idealdir.

Ancak Elastic Beanstalk'in bazı durumlarda uygun olmadığı durumlar da vardır. Örneğin, kaynak yapılandırmalarını tam kontrol etmek istediğinizde veya karmaşık bir dağıtım süreciniz varsa Elastic Beanstalk yerine OpsWorks veya CloudFormation gibi hizmetleri tercih edebilirsiniz.

All at Once: Bir güncelleme tüm örneklerde aynı anda gerçekleştirilir. Bu, mevcut altyapının hiç değişmediği yerinde bir dağıtımdır. Bu yöntemin avantajı hızlı olması ve DNS değişikliklerine ihtiyaç duyulmamasıdır. Ancak, bir sorun oluşursa tüm örneklerde sorun oluşur ve geri alma süreci karmaşık hale gelebilir. Bu nedenle, hiçbir kesinti olmaması gereken bir gereksiniminiz varsa "All at Once" stratejisini kullanmamalısınız.

Rolling: Dağıtımında en az bir örneğin her zaman çalışır durumda olması garantilenir. Güncelleme tamamlandığında, örnekler yük dengeleyiciden ayrılır, güncelleme tamamlandıktan sonra tekrar yük dengeleyiciye eklenir. Bu yöntemin dezavantajı, güncelleme sırasında yük dengeleyici sağlık kontrollerinin başarısız olması durumunda farklı sürümlerde örneklerin oluşması ve güncelleme başarısız olursa kapasite sorunları yaşanmasıdır.

Rolling with Additional Batch: Dağıtımında güncelleme öncelikle mevcut döngüde değil, yeni örneklerde gerçekleştirilir. Bu yöntemin avantajı kapasite sorunları yaşanmamasıdır. Dezavantajı ise normal bir "Rolling" güncellemeden daha uzun sürmesidir.

Immutable: Dağıtımında mevcut örnekler değiştirilmez, yerine tamamen yeni örnekler oluşturulur. Bu yöntemin avantajı, kesintisizlik sağlaması ve geri alma işleminin kolay olmasıdır. Dezavantajı ise geçici olarak iki katı örnek oluşturmasıdır.

Blue/Green: Dağıtımında tüm ortam klonlanır. Yeni sürüm yeşil ortama dağıtılır ve tamamlandığında DNS değişikliği yapılır. Bu yöntemin avantajı, kesintisiz bir geçiş sağlaması ve geri alma işleminin kolay olmasıdır. Dezavantajı ise çift infrastrüktür gerektirmesidir.

Canary (Kanarya): Dağıtımı da Blue/Green dağıtımına oldukça benzer. Bu durumda, yeni sürümü test etmek için sınırlı bir miktar trafiği yeşil ortama yönlendiririz. Eğer testler başarılıysa, daha fazla trafiği yeşil ortama yönlendirerek testlere devam ederiz. Yeşil ortamda her şey yolunda giderse, sonunda tüm trafiği yeşil ortama yönlendiririz. Canary dağıtımı da Blue/Green dağıtımı gibi kesintisiz bir geçiş sağlar, ancak daha küçük bir ölçekte gerçekleşir ve genellikle daha az riskli durumlar için tercih edilir.

Docker: Docker, Elastic Beanstalk ile kullanılabilecek iki farklı dağıtım seçeneği sunar: tek konteyner dağıtımları ve çoklu konteyner dağıtımları. Tek konteyner dağıtımları, Elastic Beanstalk tarafından yönetilen bir EC2 örneğinde tek bir Docker konteynerinin çalıştırılmasını sağlar. Bu şekilde, uygulamanızı hızlı ve kolay bir şekilde dağıtabilirsiniz.

Tek konteyner dağıtımları için Dockerfile zorunlu iken, çoklu konteyner dağıtımları için Docker run dosyası da zorunludur. Docker konteynerleri, Docker görüntü dosyalarını barındırır ve Dockerfile veya Docker run dosyaları bu konteynerlerin yapılandırmasını belirler.

AWS Lambda: Lambda, sunucu tahsisi yapmadan kodunuzu çalıştırmanıza olanak sağlayan bir hesaplama hizmetidir. Elastic Beanstalk ile karşılaştırıldığında, Elastic Beanstalk'ta sunucuları ve veritabanlarını yapılandırmamız gereken durumlar olabiliyor. Ancak Lambda'da bu tür yapılandırmalarla uğraşmamıza gerek yok, tüm bu işlemler AWS tarafından yönetiliyor. Sadece kodumuzu yüklüyoruz ve bazı metrikler sağlıyoruz. Lambda'nın farklılaştığı nokta, kodumuz çalışırken yalnızca bu süre için ücretlendirilmesidir.

Lambda SAM (Serverless Application Model): SAM, açık kaynaklı bir çerçeve olup serverless uygulamalar oluşturmak için kullanılır. Lambda fonksiyonları, API'ler, veritabanları ve etkinlik kaynağı eşlemelerini ifade etmek için YAML kullanarak kısaltılmış bir sözdizimi sunar. SAM

çerçevesini kullanmak için SAM CLI'nin yüklenmesi gerekmektedir. Bu CLI, Lambda benzeri bir yürütme ortamı sağlar ve uygulamaları yerel olarak oluşturmanıza, test etmenize ve hata ayıklamanıza olanak tanır. Ayrıca, CloudFormation ile entegrasyon sağlar.

AWS Step Functions: Birden fazla AWS hizmetini bir araya getirerek serverless iş akışlarını koordine etmemizi sağlar. Lambda gibi hizmetlerin yanı sıra Fargate, Amazon SageMaker gibi hizmetleri birleştirerek zengin özellikli uygulamalar oluşturabilir ve güncelleyebiliriz. Adımlar, görevler, seçenekler, paralel yürütmeler ve zaman aşımı gibi adımlardan oluşan bir dizi olarak iş akışını tanımlarız. Ardından, görevlerimizi, işlevlerde barındırılan kodlara, örneklemelere ve yerel sunuculara bağlarız. 15 dakikadan daha uzun süren Lambda fonksiyonlarıyla uğraşıyorsak Step Functions kullanmaya başlamamız gerekebilir. Ayrıca, her bir görev için bir fonksiyon kullanma prensibine dikkat etmek istiyoruz. Bir Lambda fonksiyonunda çok fazla iş yapmaya çalışıyorsak 15 dakikayı aşmamızın nedeni olabilir. İyi yazılım uygulamalarını uygulamak önemlidir. Step Functions, iş süreçleri için uygundur. Kullanıcı müdahalesinin gerektiği durumlarla karşılaşabiliriz, bu yüzden karışık bir yapı olabilir. Ayrıca, Lambda fonksiyonlarının paralel yürütülmesi.

API Gateway: API Gateway'nin en yaygın kullanım durumu, Lambda'nın ön ucu olarak kullanılmasıdır. API Gateway üzerinden gelen istekler, Lambda işlevinin tetikleyicisi olarak kullanılır. API Gateway, Rest ve WebSocket API'larıyla iletişim için HTTP Get ve Post yöntemlerini kullanır. Rest API'ları tek yönlü iletişim sağlarken, WebSocket API'ları iki yönlü iletişim sağlar.

WebSocket API'si, istemci uygulamaları ile arka uç arasında gerçek zamanlı iletişimi destekler. Bir örnek kullanım senaryosu gerçek zamanlı bir sohbet uygulamasıdır. WebSocket API'si bu iki yönlü iletişimi sağlayarak bu tür senaryolara uygun bir çözüm sunar. İstemci, API Gateway'e bir mesaj gönderir ve API Gateway bu mesajı ilgili Lambda işlevine yönlendirir. İlk bağlantıda, Lambda işlevi istemciyi sohbet odasına bağlar. Bağlantı kurulduktan sonra, istemci sohbet odasına mesajlar gönderebilir. İstemci, bağlandığında arka uç, istemciye bir geri çağırma URL'si gönderir ve bu URL üzerinden mesajlar gönderebilir. Sohbet odasından ayrılmak isteyen istemci, API Gateway bu mesajı OnDisconnect işlevine yönlendirir ve işlev istemciyi sohbet odasından çıkarır. Bu senaryoda DynamoDB, istemci tanımlayıcılarını depolamak için kullanılır.

Özetlemek gerekirse, REST API'ları ve HTTP API'ları bir yönlü iletişimi desteklerken, WebSocket API'si iki yönlü iletişimi destekler. WebSocket API'si, arka uç tarafından bağımsız olarak ileti gönderebilir. Sınav ipuçlarına gelince, HTTP ve REST API'ları, ek işlevselliğe ihtiyaç duymadıkça daha ucuz ve daha hızlıdır. İki yönlü trafiğe ihtiyaç duyuluyorsa, WebSocket API'si tercih edilir.

CloudFormation: CloudFormation şablonları, AWS kaynaklarını tek bir yerden yönetmenizi sağlar ve altyapıyı kod olarak tanımlayarak tekrarlanabilirliği ve otomasyonu artırır. Ayrıca, felaket kurtarma senaryoları için kullanılabilir ve şablonların bütünlüğünü korumak için kaynak kontrolü ve drift tespiti gibi yöntemlerle desteklenir.

Başka bir önemli kullanım durumu ise bir hibrit ortamda çalışıyorsak. Hala bir veritabanı örneği oluşturuyoruz, ancak bu örnek AWS ortamının bir parçası bile değil. Bu durumda sinyalleri göndermek için yollar bulmamız gerekiyor. Bir sonraki derste yardımcı betikler hakkında

konuşacağız. Ancak "cfn-signal" işlemi, yapılandırmanın tamamlandığını belirtmek için kullanılır. Ancak bu, belirtilen kaynak tamamlanana kadar bekleme koşulunun yığım oluşturmasını duraklatmasına bağlıdır. Ardından "cfn-signal" işlemiyle sinyal gönderebiliriz. Genellikle yığın, birden çok başarılı sinyali bekler ve ardından yığın oluşturmayı tamamlar.

cfn-signal komut dosyası ise bekleyen bir durum (wait condition) ile stack arasında sinyal iletişimi sağlar. Stack, bekleyen durumda beklerken sinyalleri bekler. Eğer zaman aşımı süresi içinde başarı sinyallerini alamazsa, stack oluşturma başarısız olur.

cfn-hop, metadata güncellemelerini kontrol eden bir daemon'dur. Örneği oluşturulduktan sonra bu daemon çalışmaya devam eder ve metadata'da değişiklikler algılandığında özel işlevleri çalıştırır.

CloudFormation StackSets: Birden fazla hesap ve bölgede tek bir işlemle yığınları dağıtmamıza olanak sağlar. Bu dağıtım tek bir hesapta ve tek bir bölgede yapmaktan daha uzun sürebilir, çünkü birden fazla hesap ve bölgede yığınları dağıtıyoruz. StackSets, bir şablonda birden fazla yığın oluşturmayı sağlar ve bu yığınlar farklı hesaplara ve bölgelere dağıtılır.

StackSets kullanırken, bir yönetici hesabı ve hedef hesaplar arasında bir ilişki kurmamız gerekmektedir. Yönetici hesap, StackSets'i oluşturduğumuz hesaptır ve StackSets'i yönetmek için bu hesaba giriş yaparız. Hedef hesaplar ise StackSets içinde bir veya daha fazla yığın oluşturulan hesaplardır. Hedef hesaplar arasında tüm yığınları aynı anda oluşturmak zorunda değiliz, istediğimiz yığınları tek tek oluşturabiliriz.

CloudFormation Drift (sapma): Drift, CloudFormation şablonuyla oluşturulmuş kaynakların dışında yapılan değişikliklerdir. Bu değişiklikler, CloudFormation şablonuyla oluşturulan yığının mevcut durumunu doğru bir şekilde yansıtmaz. Drift oluştuğunda, şablon artık yığını doğru bir şekilde temsil etmez ve bazı sorunlara yol açabilir.

Drift oluştuğunda, yığında değişiklik yapmak veya yığını silmek zorlaşır. Örneğin, bir şablonun içindeki bir öğeyi dışarıdan değiştirirseniz (örneğin, bir EC2 örneği boyutunu değiştirirseniz), şablon artık doğru bilgileri yansıtmaz. Bu durumda, şablonu güncellemek ve değişiklikleri yığına uygulamak gerekebilir. Drift tespiti için farklı yöntemler vardır. Management Console üzerinden tüm yığında veya belirli kaynaklarda drift tespiti yapabilirsiniz. Ayrıca AWS Config kullanarak da drift tespiti yapabilirsiniz. AWS Config'deki "stack drift detection" (yığın sapma tespiti) kuralı, drifti tespit edebilir.

Yığına sonlandırma koruması ekleyebiliriz. Bu korumayı Yönetim Konsolu veya Komut Satırı Arayüzü (CLI) aracılığıyla yapabiliriz. Sonlandırma koruması etkinleştirildiğinde, yığın üzerindeki silme işlemleri reddedilecektir.

OpsWorks: OpsWorks, Chef ve Puppet'in yönetilen örneklerini sağlayan bir yapılandırma yönetimi aracıdır. Chef ve Puppet, kod kullanarak örneklerinizin yapılandırmasını otomatikleştirmenize olanak tanıyan otomasyon platformlarıdır.

OpsWorks'un temel bileşeni OpsWorks Stack'tir. Bir CloudFormation Stack'e benzer kavramlar vardır, ancak farklı uygulamaları vardır. Bir stack içinde katmanlar bulunur ve daha fazla katman gördükçe, çok belirli olduklarını görürsünüz. Bir Yük Dengeleyici Katmanı, bir Uygulama Sunucusu Katmanı ve bir RDS Veritabanı Katmanı gibi düşünebilirsiniz. Bunlar yalnızca sahip olabileceğimiz katmanlar değil, sadece iyi bir temsil şeklidir. Bu, çok katmanlı bir mimariye benzer, tamamen aynı olmasa da bazı benzerlikler çizebiliriz.

Genellikle örneklerimizin boyutu veya veritabanı sunucusunun boyutu farklıdır. Maliyeti geliştirme sunucularında düşürebilir ve üretime doğru hareket ettikçe artırabiliriz. Katmanlarımız, bir yığının bileşenlerini temsil etmek ve yapılandırmak için kullanılır. Yük Dengeleyici, Web uygulaması, veritabanı gibi bir katman için düşünebilirsiniz. Ardından, Örneklerimiz var, bir örneğin en az bir katmanla ilişkilendirilmesi gerekir ve örneklerimizi 7/24, yük bazlı veya zaman bazlı olarak çalıştırabiliriz, neredeyse Otomatik Ölçeklendirme grubunda olduğu gibi benzer bir kavramdır. Gördüğünüz gibi, bu konuya daha da derinlemesine iniyoruz. Yığınlarımızda, katmanlarımızda örneklerimiz var, bu örneklerle ne yapıOpsWorks, AWS'nin yapılandırma yönetimi aracıdır. OpsWorks, önceden yapılandırılmış bir önyüz üzerinden Chef veya Puppet gibi yapılandırma yönetimi araçlarını kullanarak uygulamalarınızın dağıtımını ve yönetimini otomatikleştirir.

Elastic Container Service: ECS, Docker konteynerlerini destekleyen, yüksek ölçeklenebilir ve yüksek performanslı bir konteyner yönetim hizmetidir. Bu hizmet, yönetilen EC2 örnek kümesinde uygulamaları kolayca çalıştırmanızı sağlar. Yazılım arka planından gelen biri için ECS'yi Docker'ın bir sarmalayıcısı olarak düşünmek iyi bir yoldur. ECS'nin temelinde Docker bulunur. Bu derste, ECS ile yakından ilişkili olan Elastic Container Registry'ye (ECR) bakacağız ve nasıl çalıştığını göreceğiz.

ECR, Docker görüntülerini depolayan bir konteyner kaynağını temsil eder. ECS ise bu görüntüleri dağıtan hizmettir. ECR, geliştiricilerin Docker görüntülerini paylaşmasını ve dağıtmasını kolaylaştıran tamamen yönetilen bir konteyner kaynağıdır. ECR, ECS ve Elastic Kubernetes Service (EKS) ile entegre çalışır. Ayrıca AWS Lambda ile de entegrasyon sağlar.

ECS'ye daha derinlemesine bakacak olursak, ECS kümesi içinde görevler ve ENI'lar (Elastic Network Interface) bulunur. Bir konteyner görüntüsü oluşturmak için Dockerfile kullanılır. Bu görüntüyü ECR'ye yayınlarız ve ardından görev tanımı adı verilen parametreleri belirterek uygulamayı dağıtırız.

ECS'yi yönetim konsolunda kullanarak bu kavramları bir araya getirelim. Konteyner tanımı ve görev tanımı gibi adımları takip ederek örnek bir uygulama oluşturabiliriz. Bu işlem sırasında ECS kümesi oluşturulur ve Docker görüntüsü Fargate üzerinde çalıştırılır. Bu süreç, ECS'nin nasıl çalıştığına dair iyi bir anlayış sağlar.

ECS hakkında bazı sınav ipuçlarına da değinildi. Örneğin, Docker görüntülerini kullanarak konteynerleri başlatan hizmet ECS'dir. Docker görüntüleri oluşturmak için Dockerfile kullanılır. DockerHub ise halka açık bir konteyner kaynağıdır. ECR ise özel bir konteyner kaynağıdır ve DockerHub'ın AWS versiyonu olarak düşünülebilir.

AWS Fargate: Fargate, sunucusuz bir konteyner hizmetidir ve AWS'nin Elastic Container Service (ECS) ile çalışır. Fargate, konteynerlerle ilgilenir ve konteynerleri temel bir hesaplama birimi olarak kullanmanızı sağlar. Bu sayede sanal makineleri yönetmek zorunda kalmadan konteynerlerinizi oluşturabilir ve dağıtabilirsiniz.

AWS Cloud Development Kit (CDK): Bulut altyapısını kodla tanımlamanızı ve sağlamanızı sağlayan AWS tarafından geliştirilen bir çerçevedir. CDK, CloudFormation'ın üzerine inşa edilmiş bir wrapper olarak düşünebilirsiniz. Başka bir deyişle, CDK, CloudFormation'ın altında çalışır. CDK, Java, C#, Go, Python, TypeScript, JavaScript gibi yüksek seviyeli dilleri kullanırken, altta hala bir CloudFormation şablonuna çözülür ve bu şablondan yola çıkarak yığın oluşturulur.

CDK, uygulamaları, yığınları ve yapılandırmaları kullanarak altyapıyı sağlar. Uygulama, yapılandırmaların ve yığınların birleştirildiği bir ağaç yapısıdır ve AWS üzerinde dağıtılabilen tüm yapılandırmaları içerir. Yığın, tek bir birim olarak düşünebileceğiniz AWS kaynaklarını tutar ve geliştirme için kullanılabilir. Yapılandırmalar ise AWS kaynaklarını içeren yapı taşlarıdır. Bir yapılandırma, tek bir AWS kaynağı veya birden fazla AWS kaynağını bir araya getirebilir. CDK'da yapılandırmalar yapı taşı kütüphanelerinde saklanır ve bu kodu yeniden kullanabilirsiniz.

AWS AppConfig: AppConfig, AWS Systems Manager'ın bir parçasıdır ve uygulama yapılandırmalarını oluşturmayı, yönetmeyi ve hızlı bir şekilde dağıtmayı sağlamak için kullanılır. Bu derste AWS AppConfig hakkında genel bir bakış atacak, AppConfig ile CodeDeploy arasındaki farkları tartışacak, bazı kullanım durumlarını ve faydalarını ele alacak ve AppConfig'in nasıl kurulduğunu gösteren bir demo yapacağız.

AppConfig'i kurmak için öncelikle bir uygulama oluşturmanız gerekmektedir. Uygulama, yapılandırma verilerini içeren bir organizasyonel yapıdır. Daha sonra bir ortam oluşturulur, bu ortam AppConfig hedeflerinin mantıksal bir dağıtım grubudur. Ardından, yapılandırma profilleri ve özellik bayrakları oluşturulabilir. Yapılandırma değişikliklerini dağıtmak için bir dağıtım stratejisi belirlenir ve son olarak yapılandırmayı hedefe dağıtabilirsiniz.

AWS Control Tower: Bir AWS çoklu hesap ortamını kurmak ve yönetmek için bir yol sağlayan bir hizmettir. Control Tower, AWS Organizations dahil olmak üzere birkaç diğer hizmetle birlikte çalışır. Control Tower, organizasyonlarla sıkı bir entegrasyona sahiptir ve çoklu hesap ortamlarında ek kontrol imkanı sağlar.

Control Tower'ın temel amacı, çoklu hesaplara uygulamaları dağıttığımızda ortaya çıkan karmaşıklığı yönetmek ve çevremizi yönetmektir. Control Tower, organizasyonların sağladığı birçok özelliği devralır ve örgüt birimleri ve hizmet kontrol politikaları gibi özellikler sunar. Control Tower, organizasyonların bir genişlemesi olarak düşünülebilir ve çoklu hesap ortamlarında ek kontrol sağlar.

Control Tower'ın ana amacı, çoklu hesap ortamlarında güvenlik ve yönetişimi sağlamaktır. İniş bölgesi oluşturarak bu amaca ulaşabiliriz ve çoklu hesaplardan gelen kayıtları tek bir yerde toplayabiliriz.

Amazon Detective: Amazon Detective, güvenlik bulgularını veya şüpheli aktiviteleri analiz etmenize, araştırmanıza ve hızlı bir şekilde kök nedenini belirlemenize yardımcı olan bir hizmettir.

AWS'de birçok güvenlik hizmeti bulunmaktadır, bunlardan bazıları GuardDuty ve Security Hub'dır. Bu hizmetler, sorunları bildirebilir veya Security Hub gibi bazıları değişiklikler önerir. Ancak Detective, bir güvenlik bulgusu olduğunda daha derine inmemiz ve bu sorunun kök nedenini bulmamız gerekmektedir.

Detective, CloudTrail günlükleri, EKS denetim günlükleri, VPC Akış günlükleri gibi günlük verilerini toplar. Detective otomatik olarak bu günlüklerden oturum açma girişimleri, API çağrıları, ağ trafiği gibi zaman temelli olayları çıkarır. Ayrıca GuardDuty tarafından tespit edilen bulguları da alır. Detective ile çalışırken GuardDuty'yi etkinleştirmiş olmanız gerektiğini ve Detective'e başlamadan önce GuardDuty'yi en az 48 saat etkinleştirmeniz gerektiğini göreceğiz. Ayrıca GuardDuty'ye ek olarak Security Hub ile de entegre olur. Tüm bu hizmetler birbirleriyle entegre çalışır ve birbirlerinin amacını hatırlamak önemlidir.

güvenlik araştırmalarına yardımcı olur. Makine öğrenimi ve graf teorisi gibi teknikleri kullanarak güvenlik sorunlarının kök nedenlerini belirlemeye yardımcı olur. Detective, CloudTrail günlükleri, VPC Akış günlükleri, GuardDuty ve Security Hub gibi hizmetlerden veri alır ve bu verileri birleştirerek etkileşimli grafikler oluşturur. AWS Organizations ile entegre olarak birden fazla hesaptan gelen verileri merkezi olarak yönetebilir.

CloudWatch: CloudWatch alarmaları, İzleme ve Günlük Kaydı alanında önemli bir rol oynar. Özellikle CloudWatch alarmaları, genellikle DevOps ve AWS genelinde Auto Scaling için alarmalar olarak düşünülür. Bu yüzden önce bunu tekrar gözden geçireceğiz. İzlememiz gereken bazı metrikleri inceleyeceğiz ve ardından bir demo yapacağız. Ayrıca DevOps perspektifinden yararlı olabilecek ancak az kullanılan bir özelliği göstereceğiz. Son olarak, bazı sınav ipuçlarına göz atacağız.

EventBridge ve SNS: EventBridge, çevremizdeki bazı etkinlikleri izleyen ve bu etkinliklere dayalı olarak belirli hedeflere otomatikleşme sağlayan bir hizmet sunar.

Otomasyonları bir araya getirebilir ve aynı zamanda EventBridge'i ve SNS'i birlikte kullanarak birden fazla otomasyonu zincirleyebiliriz. EventBridge, CloudWatch Events'in yeni bir sürümüdür ve arayüzü biraz farklıdır.

AWS CloudTrail: AWS CloudTrail, AWS hesabınızın yönetişimini, uyumluluğunu ve risk denetimini sağlayan bir hizmettir. DevOps Pro sınavındaki anahtar kelimelerden biri "denetim"dir ve denetimle ilgili bir durum olduğunda genellikle CloudTrail devreye girer. Bu yüzden CloudTrail'a dikkat etmek önemlidir.

CloudTrail ile nasıl çalışabilir ve erişebiliriz? Yönetim konsolu üzerinden, AWS araçları ve SDK'ları ile programatik olarak erişebiliriz. Örneğin, Java gibi bir programlama dilinde AWS SDK'sını kullanarak Java kodumuzda CloudTrail'a API çağrıları yapabiliriz. Ayrıca AWS CLI'yı da kullanabiliriz.

AWS Kinesis: Kinesis Data Analytics'e geçelim. Kinesis Data Analytics, gerçek zamanlı veri analizi yapmak için kullanılır. Veri akışını alır, analiz eder ve sonuçları hedeflere gönderir. SQL sorguları kullanarak veri akışını analiz edebilir ve sonuçları gerçek zamanlı olarak elde edebilirsiniz. Kinesis Data Analytics, anlık metrikler, gerçek zamanlı raporlar ve uyarılar oluşturmanıza olanak tanır. Bu, işletmelerin hızlı kararlar almasına ve gerçek zamanlı tepkiler vermesine yardımcı olabilir.

Sınavda Kinesis ile ilgili sorulara yanıt verirken, her bir Kinesis hizmetinin temel özelliklerini ve kullanım durumlarını anlamak önemlidir. Kinesis Data Firehose, veri akışını otomatik olarak hedeflere iletmek için kullanılırken, Kinesis Data Streams, gerçek zamanlı veri işleme ve analiz yapmak için kullanılır. Kinesis Video Streams, video akışlarını toplar ve analiz etmek için kullanılabilirken, Kinesis Data Analytics, gerçek zamanlı veri analizi yapmak için SQL sorgularını kullanır. Kısacası, her bir Kinesis hizmetinin ne yaptığını ve hangi senaryolarda kullanılabileceğini anlamak önemlidir.

AWS Lambda ile Kinesis: Kinesis veri akışlarını veri alıcısı ve veri göndericisi olarak kullanmayı ele alacağız. Özel bir uygulama oluşturacağız ve bu uygulama Kinesis veri akışımıza veri oluşturacak. Bu veri akışı veriyi kabul edecek ve bir Lambda fonksiyonunu tetikleyecek. Burada yaygın olan bazı desenleri göreceksiniz. Öncelikle, her zamanki gibi bir yürütme rolüne ihtiyacımız var. Çünkü fonksiyonlar adımımız adına işlem yapacaklar. Son olarak, Lambda fonksiyonumuz bilgileri CloudWatch günlüklerine aktaracak. Yani tanıdık bir yapıya sahibiz. İlk olarak, IAM'den yürütme rolümüzü oluşturmamız gerekiyor. Lambda'nın adımımız adına eylemler gerçekleştirebilmesi için bu rolü oluşturmamız gerekiyor.

Kinesis veri akışı, gelen veriyi Lambda fonksiyonuna iletecektir. Lambda fonksiyonu, gelen veriyi işleyecek ve istenen işlemleri gerçekleştirecektir. İşlemler tamamlandıktan sonra isteğe bağlı olarak sonuçları başka bir hedefe kaydedebilirsiniz. Örneğin, CloudWatch Logs gibi bir hedef kullanarak logları kaydedebilirsiniz.

Son olarak, senaryonun doğru çalıştığını doğrulamak için Lambda fonksiyonunu test edebilirsiniz. Bu, örnek bir veri göndererek veya manuel olarak tetikleyerek yapılabilir. Lambda fonksiyonunun beklenen çıktıyı ürettiğinden ve işlemleri doğru bir şekilde gerçekleştirdiğinden emin olun.

Bu adımları takip ederek, AWS Lambda ve Kinesis veri akışlarını bir araya getirerek veriyi işleyebilir ve istenen işlemleri gerçekleştirebilirsiniz. İhtiyaçlarınıza ve senaryonuza göre uygun yapılandırmaları yapmayı unutmayın.

AWS X-Ray: Dağıtık uygulamalarımızda sorunları tespit etmek ve performansı izlemek için kullanılan bir hizmettir. Mikro hizmetlerden oluşan bir uygulama içerisinde, her bir bileşenin izlenmesi ve sorunların bulunması zorlaşabilir. İşte bu noktada X-Ray devreye girer.

Son olarak, log dosyalarımızı şifreleme konusunda birkaç sınav ipucu vermek istedik. Log dosyalarıyla kullanılabilecek farklı şifreleme türlerini anlamak önemlidir. Varsayılan olarak, CloudWatch logları sunucu tarafında şifreleme anahtarlarını yönetir ve aynı durumu CloudTrail'de de gördük. Ancak başka seçenekler de vardır. Kendi anahtarımızı kullanarak KMS ile müşteri

anahtarlı şifreleme yapabiliriz. CloudTrail logları için de aynı durum geçerlidir. Log dosyaları otomatik olarak S3'de SSE-S3 kullanılarak şifrelenir. Bunun yönetimini yapmamız gerekiyorsa SSE-KMS ile bunu gerçekleştirebiliriz.

AWS OpenSearch: Öncelikle belirtmek istediğim bir nokta, bu hizmetin eskiden Elasticsearch olarak bilindiği, ancak 2021 yılında AWS tarafından isminin OpenSearch olarak değiştirildiği. Bu yüzden sınavlarda Elasticsearch terimini görebilirsiniz, bunun OpenSearch ile aynı şeyi ifade ettiğini unutmayın.

AWS OpenSearch, AWS bulutunda OpenSearch kümeleme işlemlerini kolayca dağıtmak, işletmek ve ölçeklendirmek için tasarlanmış bir yönetilen hizmettir. Adından da anlaşılacağı gibi, veri araması ve analizi yapmak için kullanılır ve genellikle Logstash ve Kibana ile ilişkilendirilir. Kibana, Elasticsearch ile çalışmak üzere tasarlanmış popüler bir açık kaynak görselleştirme aracıdır. Logstash ise logları depolamak ve OpenSearch alanına yüklemek için kullanılır.

Son olarak, AWS OpenSearch'in adının 2021 yılında değiştirildiğini ve eski adının Elasticsearch olduğunu unutmayın. Sınavlarda bu iki terimin birbirini ifade ettiğini bilmeniz önemlidir.

Etiketleme (tagging) ve kaynak grupları (resource groups) hakkında konuşacağız. Etiketleme ve kaynak gruplarıyla birçok şey yapabiliriz, ancak bu kursun odak noktası olan DevOps için özellikle otomasyonlar için etiketleme ve kaynak gruplarını kullanabiliriz. Bu derste genel olarak etiketlemeye bir göz atacak, hangi kaynakları etiketleyebileceğimizi ve bazı etiketleme kategorilerini ve stratejilerini inceleyeceğiz. Ayrıca yönetim konsolundan hızlı bir demo yapacağız ve sınav ipuçlarını gözden geçireceğiz.

Sonuç olarak, bu derste etiketleme ve kaynak grupları hakkında genel bir bakış yaptık. Otomasyonlarımız için bunları nasıl kullanabileceğimizi gördük. Sınav ipuçlarını ve stratejilerini gözden geçirdik. Kaynaklarımızı etiketlemek iyi bir uygulama olup stratejimiz olmalıdır. Otomasyon, maliyet tahsisi, erişim kontrolü ve yönetişim gibi farklı alanlarda etiketleme stratejileri kullanabiliriz.

Prometheus: Ölçeklenebilir bir şekilde konteynerleştirilmiş uygulamaları ve altyapıyı izler ve uyarılar sağlar. Amazon Managed Service for Prometheus, Elastic Kubernetes Service, Elastic Container Service ve AWS Distro ile entegre edilebilir. Yapmamız gereken bağlantı, Prometheus ile konteynerleştirilmiş uygulamalar arasında yapılır. Ve eğer bilmiyorsanız, Amazon Managed Service for Prometheus adında bir açık kaynak sürümünün olabileceğini düşündürür. Ve gerçekten öyle, Prometheus, olay izleme ve uyarı için kullanılan bir yazılım uygulamasıdır. HTTP havuz modelini kullanarak gerçek zamanlı metrikleri bir zaman serisi veritabanına kaydMerhaba! Bu derste, izleme ve günlük kaydetme alanındaki destekleyici hizmetlere odaklanacağız. Sınavda çok önemli olmasalar da, karşılaşabileceğiniz bazı hizmetlere bakacağız. Bu yüzden, bunların ne olduğunu, ne yaptıklarını ve DevOps resminde nasıl yer aldıklarını bilmek faydalı olacaktır.

İlk olarak, AWS Compute Optimizer'a bakalım. Compute Optimizer, çalışma yüklerimizi inceleyerek optimizasyon önerileri sunan bir hizmettir. Makine öğrenmesini kullanarak

kaynaklarımızın nasıl yapılandırıldığını analiz eder ve aşırı veya yetersiz kaynak kullanımını önlememize yardımcı olur. Aşırı kaynak kullanıyorsak para israf ediyoruz demektir, yetersiz kaynak kullanıyorsak performansımız etkilenir. Compute Optimizer, EC2, EBS birimleri, Fargate üzerinde ECS hizmetleri ve Lambda işlevleri gibi dört tür kaynağı analiz edebilir. CloudWatch ile entegre çalışarak kaynak kullanım verilerini analiz edebilir. Bu sayede aşırı veya yetersiz kaynak kullanımını belirleyebiliriz. Bu şekilde kaynakları doğru boyutta kullanarak maliyeti düşürebilir veya performansı artırabiliriz.

AWS License Manager: Yazılım lisanslarımızı yönetmemizi sağlayan bir hizmettir. Microsoft, SAP, Oracle ve IBM gibi yazılım sağlayıcılarından lisansları yönetebiliriz. Bu hizmetle lisanslı yazılımlar için kurallar belirleyebilir, yazılım başlatılırken bu kuralları uygulayabilir, lisans envanterini takip edebilir ve AWS hesaplarımızın yanı sıra yerinde ortamlardaki lisansları merkezi olarak yönetebiliriz. Yani License Manager, lisans yönetimini kolaylaştırır ve lisans uyumluluğunu sağlar.

AWS Managed Service for Prometheus (AMP) ve Amazon Managed Grafana ise izleme ve analiz hizmetleridir. Prometheus, konteyner tabanlı uygulamalar ve altyapılar için izleme ve uyarı sağlar. AWS Managed Service for Prometheus, Elastic Kubernetes Service (EKS), Elastic Container Service (ECS) ve AWS Distro ile entegre çalışabilir. Böylece konteyner tabanlı uygulamaların metriklerini toplayabilir ve izleyebiliriz. Amazon Managed Grafana ise metrikleri görselleştirmek ve alarm oluşturmak için kullanılan bir hizmettir. Grafana, AMP ile entegre çalışarak verileri grafikler ve tablolar halinde sunar.

Son olarak, Amazon AppFlow hakkında konuşalım. AppFlow, SaaS (Yazılım olarak Hizmet) uygulamaları ile AWS hizmetleri arasında çift yönlü veri akışını otomatikleştiren bir hizmettir. Salesforce, SAP, Google Analytics gibi SaaS uygulamalarından veri alabilir, bu verileri filtreleyebilir, doğrulayabilir ve AWS veri depolarına (S3) veya daha derin analizler için Redshift'e aktarabiliriz. AppFlow, yönetilen bir hizmet olduğu için ölçeklenebilir ve otomatik olarak ayarlanır.

QuickSight: Etkileşimli gösterge panoları oluşturmanızı ve yayınlamanızı sağlayan bir hizmettir. Makine öğrenimi içeren veri gösterge panoları oluşturabilir. QuickSight, buluttaki verilerinize bağlanır ve AWS verileri, üçüncü taraf verileri, büyük veri, elektronik tablo verileri, SaaS verileri, B2B verileri gibi birçok farklı kaynaktan verileri birleştirebilir.

QuickSight, Athena, Aurora, OpenSearch, Redshift, S3, RDS gibi birçok farklı veri kaynağını destekler. Bu veri kaynaklarından verileri analiz için sağlayabilirsiniz. Örneğin, CloudWatch günlüklerini S3 kovasına ileten bir Lambda işlevi gibi otomasyonlarla bu veri kaynakları doldurulabilir.

Sınav ipuçlarına gelince, QuickSight'ın ne yaptığını, hangi hizmetlerle etkileşimde bulunduğunu ve veri kaynaklarını anladığınız sürece sınav için iyi bir durumda olursunuz. QuickSight'ın bir DevOps ortamında nasıl kullanılabileceğini anlamak için S3, RDS, Athena, Aurora gibi veri kaynaklarına bağlanabilir ve otomasyonlarla bu veri kaynaklarını doldurabilirsiniz.

Amazon GuardDuty: GuardDuty, sürekli olarak log dosyalarını analiz eden bir hizmettir. Bu log dosyaları arasında VPC Flow Logs, CloudTrail Logs ve bazı DNS logları bulunur. Otomasyon temasına bağlı kalarak, GuardDuty ile ilgili olarak EventBridge olayını ekleyebilir ve GuardDuty'deki yeni bulguların tetiklenmesini sağlayabiliriz. Her zamanki gibi, sorunların giderilmesi için bir Lambda işlevi veya bir SNS konusu kullanabiliriz.

GuardDuty, AWS hesaplarınızı ve yüklerinizi sürekli olarak izlemenizi ve korumanızı sağlayan bir tehdit tespiti sunar. GuardDuty, hesabınızdan oluşturulan meta veri akışlarını, CloudTrail, VPC Flow Logs ve DNS loglarında bulunan ağ etkinliklerini analiz eder. Entegre tehdit istihbaratı ve makine öğrenimi kullanarak tehditleri daha doğru bir şekilde tespit eder. GuardDuty, ek güvenlik yazılımı veya altyapıya ihtiyaç duymadan tüm AWS hesaplarımızı izleyebilir ve kötü niyetli veya izinsiz davranışları tespit eder. Bulguları yönetim konsolunda inceleyebilir veya EventBridge, Lambda gibi etkinlik yönetimiyle birlikte kullanabiliriz.

GuardDuty'nin örnek bulgular oluşturabileceği yönetim konsolunda da gösterildi. Bu örnek bulgular, yetkisiz IAM kullanıcısı, zararlı alan adı isteği, hizmet reddi gibi tehditleri içerebilir. Bulgular, düşük, orta ve yüksek şiddet seviyelerini temsil eden simgelerle gösterilir.

Amazon Inspector: Sertifikasyon sınavında Inspector'ın kapsamı, Inspector'ın kullanım durumlarını bilmek ve sıklıkla karıştırılan GuardDuty, Trusted Advisor ve bazen de Macie gibi yaygın yanıltıcılarının kullanım durumlarını bilmek gerektiğidir. Inspector hakkında bir ana cümlemiz var: Inspector, EC2 örneklerinin ağ erişilebilirliğini ve bu örneklerde çalışan uygulamaların güvenlik durumunu test eder. Eğer bu bilgilere sahipseniz sınavda sorun yaşamazsınız.

Inspector, örneklerimizi ve uygulamalarımızı zafiyetler açısından otomatik olarak analiz etmek üzere yapılandırılabilir. Otomasyon sürecini devam ettirebiliriz. Örneğin, CodePipeline üzerinden bir uygulama dağıtırsak, bir Lambda işlevini tetikleyebiliriz. Lambda Inspector'ı tetikleyebilir. Lambda EC2 üzerinde görevler gerçekleştirebilir. Inspector EC2 ve uygulamalarımızı analiz edebilir. Son olarak, SNS ile bir Lambda işlevi tetiklenebilir ve Slack kanalına bildirim gönderilebilir. Inspector oldukça basittir. Çok derinlemesine gitmemize gerek yoktur. Sadece ne yaptığını hatırlamamız yeterlidir. Tekrar ediyorum, Inspector EC2 örneklerimizi ve bu örneklerde çalışan uygulamaları zafiyetler açısından analiz eder. Bunları bilirseniz, Inspector ile ilgili sınavda sorun yaşamazsınız.

Security Hub: güvenlik en iyi uygulama kontrolleri yapabilen bir bulut güvenlik durumu yönetimi hizmetidir, uyarıları bir araya getirir ve otomatik düzeltmeyi sağlar. Gelişmiş güvenlik önerileri sunan Trusted Advisor var. Ve son olarak, Config. AWS Config ile uyumluluk ve denetim kurabiliriz. Güvenlik bilgi ve olay yönetimine bir göz atalım. Güvenlik olaylarına ve olaylara yanıtımızı otomatikleştirebilir miyiz? Ve şimdiye kadar muhtemelen cevabı biliyorsunuz, kesinlikle. Bu, kullanabileceğimiz bazı araçları ve bu araçları nasıl kullanarak otomatikleştireceğimizi bilmekle ilgilidir. Bu hizmetlerin çoğunu zaten gördük.

OpsCenter: AWS Systems Manager'ın bir yeteneğidir ve operasyonel iş kalemlerini yönetmek için bir merkezi bir konum sağlar. Bir iş kalemi veya operasyonel sorun, araştırılması ve giderilmesi gereken herhangi bir operasyonel sorundur. OpsCenter, tüm bu işlemleri tek bir yerde, bir gösterge tablosunda yapmanızı sağlar.

OpsCenter, AWS kaynaklarını etkileyen sorunların çözülme süresini azaltmanızı sağlar ve hizmetler arasında birleşik ve standart bir iş kalemi oluşturabilir. Sistem Yöneticisi otomasyonunu kullanarak sorunları hızlı bir şekilde gidermek için çalışma kitapları kullanılabilir. Ayrıca OpsCenter'ı CloudWatch Alarmları ve EventBridge olaylarıyla entegre edebilirsiniz.

OpsCenter, CloudWatch ve EventBridge ile entegrasyon sağlar ve bu entegrasyonlar sayesinde otomatik olarak bir iş kalemi oluşturur. Örneğin, CloudWatch alarmları veya EventBridge olayları tetiklendiğinde OpsCenter'da bir iş kalemi oluşturulur. CloudWatch alarmlarıyla ilgili yaygın senaryolar arasında veritabanı okuma ve yazma eylemlerinin bir eşiğe ulaşması, CPU kullanımı, bellek kullanımı, faturalandırma eşiği ve EC2 disk alanı durum kontrolü başarısızlıkları yer alır. EventBridge ise Security Hub uyarıları, DynamoDB, Otomatik Ölçeklendirme sağlık uyarıları ve örneğin duran örnekler gibi birçok hizmetle entegre olabilir.

Secrets Manager: Sırların güvenli bir şekilde depolanması ve yönetilmesi için bir hizmettir.

Ayrıca, politikalar ve izinler, izin sınırları ve sınav ipuçları gibi konuları ele aldık. Bu ders, IAM'ın ölçekte nasıl yönetileceği ve güvenli bir şekilde kullanılacağı konularında genel bir bakış sunmaktadır.

AWS WAF, OSI modelinin 7. katmanında çalışan bir web uygulama güvenlik duvarıdır. HTTP, HTTPS ve web uygulama güvenlik açıklarına karşı koruma sağlar. Örneğin SQL enjeksiyonu, cross-site scripting ve DDoS saldırıları gibi saldırıları engellemek için kullanılır.

AWS Shield ise dağıtık bir hizmet reddetme saldırılarına (DDoS) karşı koruma sağlayan bir hizmettir. Shield da aynı şekilde 3. ve 4. katmanlarda çalışır. WAF DDoS saldırılarını da engelleyebilir.

AWS Organizations ve Control Tower gibi hizmetler, çoklu hesap ve çoklu bölge ortamlarında güvenliği ölçeklendirmek için kullanılabilir. Organizasyonlar, hesapların merkezi düzeyde güvenliğini sağlamak için faydalı olabilir. Control Tower, merkezi bir yerleşim yeri olarak konfigüre edilebilir ve güvenlik ve yönetişim için kullanılabilir.

Sınav ipuçlarına gelince, AWS Network Firewall'ın İnternet Ağ Geçidi ile müşteri altyapısı arasındaki bağlantıyı nasıl kolaylaştırdığını bilmek önemlidir. WAF, Network Firewall ve Shield'ın hangi OSI katmanlarında çalıştığına dikkat etmek gerekmektedir. AWS Organizations OUs'un Control Tower'a nasıl yardımcı olduğunu bilmek de önemlidir.

AWS CloudTrail: AWS CloudTrail, AWS hesabınızdaki etkinlikleri izlemek ve denetlemek için kullanılır. CloudTrail, API çağrılarını, yönetici etkinliklerini, kaynak değişikliklerini ve diğer olayları

kaydeder. Bu kayıtlar, güvenlik tehditlerini tespit etmek, uyumluluk gereksinimlerini karşılamak ve olayları soruşturmak için kullanılabilir.

AWS Config: AWS Config, kaynak yapılandırmasını sürekli olarak izleyen ve kaynaklardaki değişiklikleri kaydeden bir hizmettir. AWS Config, yapılandırma değişikliklerini izler ve kaydeder, ardından bu bilgileri kullanarak güvenlik değerlendirmeleri yapabilir ve yapılandırma hatalarını tespit edebilir.

VPC Flow Logs: VPC Flow Logs, Amazon Virtual Private Cloud (VPC) içindeki ağ trafiğini izlemek için kullanılır. VPC Flow Logs, gelen ve giden ağ trafiğini kaydeder ve güvenlik tehditlerini tespit etmek, ağ performansını analiz etmek ve uyumluluk gereksinimlerini karşılamak için kullanılabilir.

AWS Security Hub: AWS Security Hub, AWS hesabınızdaki güvenlik olaylarını merkezi bir konumda toplar ve bunları otomatik olarak analiz eder. Security Hub, AWS Config, AWS CloudTrail, Amazon GuardDuty ve diğer güvenlik hizmetleriyle entegre çalışır. Bu sayede, güvenlik tehditlerini daha iyi tespit edebilir ve yönetebilirsiniz.

Amazon GuardDuty: Amazon GuardDuty, AWS hesabınızdaki güvenlik tehditlerini sürekli olarak izler ve tehditleri tespit eder. GuardDuty, anomali tabanlı tespit yöntemleri ve makine öğrenimi teknikleri kullanır. Güvenlik tehditleri, kötü amaçlı yazılım aktiviteleri, kimlik hırsızlığı girişimleri ve diğer potansiyel tehditler gibi çeşitli kategorilerde olabilir.

Sınav ipuçları olarak, AWS'ye hangi kimlik sağlayıcılarıyla giriş yapabileceğimiz, kimlik sağlayıcının hangi formatlarla uyumlu olması gerektiği ve AWS erişimi için geçici belgeler sağlayan hizmetin AWS Security Token Service (STS) olduğu bilgilerine dikkat etmek önemlidir.

AWS Organizations: Birden çok AWS hesabını bir araya getirerek merkezi olarak yönetmenizi sağlayan bir hesap yönetimi hizmetidir. Büyük bir şirketin birden çok ofisi olan ve birden çok AWS hesabına sahip olduğunu düşünün. Tüm bunları nasıl yöneteceksiniz? Cevap AWS Organizations'tır. Organizations, hesap yönetimi birleşik faturalama özelliklerini içerir ve bu da bütçe, güvenlik ve uyumluluk ihtiyaçlarınızı daha iyi karşılamanızı sağlar.

AWS Organizations'ı kullanmak için öncelikle bir yönetim hesabına ihtiyacınız vardır. Bu yönetim hesabından AWS organizasyonunuzu oluşturabilirsiniz. Ardından, diğer hesapları organizasyona davet etmeniz gerekmektedir. Sınav için organizasyon oluşturma sürecini ayrıntılı olarak bilmeniz gerekmemekle birlikte, ilgileniyorsanız organizasyon oluşturma sürecini içeren bir ek dersimiz bulunmaktadır.

AWS Organizations, birden çok AWS hesabının merkezi olarak yönetilmesi ve organizasyonel gereksinimlerin karşılanması için güçlü bir hizmettir. CloudFormation StackSets, AWS Service Catalog, CloudTrail, EventBridge olayları ve AWS Config gibi birçok AWS hizmetiyle entegre çalışır.

Trusted Advisor: Trusted Advisor, AWS ortamımızı sürekli olarak analiz eden ve bize en iyi uygulamalar konusunda geri bildirim sağlayan bir hizmettir.

İşletim maliyeti optimizasyonu, performans, güvenlik, hata tolere edilebilirliği ve hizmet sınırları gibi alanlarda geri bildirim sağlar. Ayrıca, öneriler ve eylem bağlantıları da sunar. Örneğin, bir güvenlik grubunun çok açık olduğunu belirtebilir ve bu durumu düzeltmek için bir bağlantı sağlayabilir.

Trusted Advisor'ın 7 temel denetimi nelerdir? Bu denetimler, varsayılan olarak Basic ve Developer planlarıyla birlikte gelir. Bu denetimler: kova izinleri, güvenlik grupları, IAM kullanımı, kök hesapta MFA kullanımı, halka açık EBS görüntüleri, RDS halka açık görüntüleri ve hizmet sınırlarıdır.

Trusted Advisor'ı AWS Yönetim Konsolu'ndan nasıl kullanabiliriz? Yönetim konsolunda Trusted Advisor'a erişebilir ve mevcut denetim sonuçlarını görüntüleyebiliriz. Bu sonuçlar, maliyet optimizasyonu, performans, güvenlik, hata tolere edilebilirliği ve hizmet sınırları gibi alanlarda bulunabilir. Daha fazla denetim sonucunu görmek için destek planınızı yükseltebilirsiniz.

Sınav sırasında dikkate almanız gereken bazı noktalar şunlardır: Trusted Advisor, en iyi uygulamalar ve öneriler gibi terimlerle ilişkilendirilebilir. Sınav sorularında AWS Inspector, Cost Explorer, GuardDuty ve CloudWatch gibi hizmetler, Trusted Advisor ile birlikte kullanılmak üzere yanıltıcı seçenekler olarak sunulabilir. Trusted Advisor'ın kontrol ettiği kategorileri bilmek de önemlidir: İşletim maliyeti optimizasyonu, performans, güvenlik, hata tolere edilebilirliği ve hizmet sınırları.

AWS Service Catalog: Service Catalog, büyük bir kuruluşta kullanıcıların kuruluşunuzdaki uygulamalara erişimine odaklanan bir hizmettir. Farklı türlerde kullanıcılara sahip olabilirsiniz: bazıları her şeyi indirip denemek isteyen çok istekli kullanıcılar olabilirken, diğerleri ise ne kullanacaklarını veya nasıl kullanacaklarını bilmeyen kullanıcılardır. İşte burada Service Catalog devreye girer.

Service Catalog, uygulamalarınızın ve diğer hizmetlerinizin bir kataloğu gibi düşünebilirsiniz. Kullanıcıların bakış açısından, yönetici olarak ise düzenlememize, izinlerimizi düşünmemize ve uygulama yığınlarını dağıtmamıza yardımcı olur. Service Catalog'da bu uygulama yığınlarına "ürünler" denir.

Service Catalog, kurumsal standartlara uyumu sağlama, ürünleri merkezi olarak yönetme ve dağıtma, AWS üzerindeki tüm uygulamaları yönetme gibi konularda yardımcı olur. Ayrıca, kullanıcıların ürünleri keşfetmelerini ve başlatmalarını sağlar. Son kullanıcılar, kendi başlarına ihtiyaç duydukları ürünlere göz atabilir ve erişebilirler. Bu sayede kullanıcılarımız, gereksinimleri dışındaki uygulamalara erişmeden kendi kendilerine yeterli olabilirler.

Systems Manager: Systems Manager'ın girişini, yönetim konsolunda yapılandırmayı ve bazı ayarları göreceğiz. Systems Manager'ın giriş seviyesindeki bir bakış açısını ele alacağız. Kaynak grupları hakkında daha fazla bilgi vereceğiz ve Systems Manager'ın sunabileceği yeteneklerden bahsedeceğiz. Ardından yönetim konsolunda Systems Manager'ı canlı olarak görmek için hızlı bir gösteri yapacağız. Son olarak, organizasyonel kontroller ve sınav ipuçları hakkında konuşacağız.

Sanal makinelerimizi yönetmek için kullanabileceğimiz bir yetenekler koleksiyonudur. Hem bulutta bulunan EC2 örneklerimizi hem de yerel sunucularımızı yönetebiliriz. Ayrıca Systems Manager ile diğer AWS kaynaklarını da yönetebiliriz. Systems Manager'ı kısaca tanımlamak gerekirse, Systems Manager web sunucularınızı yönetmenize yardımcı olur. Tabii ki, bununla sınırlı değildir ve EC2 örnekleri veya yerel sunucularımızla ilgili senaryoları ele alırken, Systems Manager'ı çözümün bir parçası olarak düşünebiliriz. Sunucularımızı yönetmek için yapabileceğimiz bazı şeyler arasında yazılım envanteri toplamak, işletim sistemlerini yapılandırmak, sistem görüntüleri oluşturmak, hibrit bulut yönetimi yapmak (hatırlayın, yerel sunucularımız) yer alır. Bunları göz önünde bulundurarak, Systems Manager'ı düşünebiliriz.

Yönetim konsolunda Systems Manager'ı yapılandırabilir ve kullanabilirsiniz. Systems Manager, EC2 örnekleri, sanal özel sunucular (VPC'ler), Lambda fonksiyonları, S3 depolama kovaları ve diğer AWS kaynakları gibi birçok farklı kaynağı yönetme yeteneğine sahiptir.

Ayrıca, Systems Manager'ı kullanarak örneklerinizin yapılandırmasını da yönetebilirsiniz. Örneğin, belirli bir konfigürasyon dosyasını örneklerinize dağıtabilir veya örneklerinizin belirli bir yapılandırma durumunu izleyebilirsiniz.

State Manager: Sistem Yöneticisi'nin bir yeteneğidir ve örneklerimiz için standart bir yapılandırma belirleyip yapılandırabiliriz. State Manager ile yapabileceğimiz bazı şeyler: örneklerimizi başlatırken belirli yazılımlarla başlatma, tanımlanan bir zaman çizelgesine göre ajanları indirme ve yükleme, ağ ayarlarını yapılandırma, örnekleri Microsoft Active Directory etki alanına katılma, Linux, MacOS ve Windows üzerindeki ömür boyu süresince komut dosyalarını çalıştırma gibi işlemler yapabiliriz. Böylece örneklerimizi belirli bir yapılandırma durumuna getiriyoruz ve bu durum State Manager tarafından belirleniyor. Uyumluluk, State Manager ile birlikte çalışır. Uyumluluk, Yama Yöneticisi'nde ve State Manager'da gösterilen yama uygulama durumu ve ilişkilerini görüntüler. Yani, her şey uyumluluğa uygun mu? Bu bilgiyi tarama yaparak elde eder ve sonuçlarını Yama Yöneticisi ve State Manager'da görüntüler.

Sistem Yöneticisi Run Command hakkında konuşacağız. Run Command sayesinde sunucularımızın yapılandırmasını uzaktan yönetebiliriz, hatta bu sunucular yerelde olsalar bile. Bu sayede, yerel sunucularla bulut ortamında yer alan EC2 örneklerini aynı şekilde yönetebiliriz. Yerel sunucularımız için bir benzetme yapacak olursak, eğer yerel sunucularımızın CloudWatch ile çalışmasını istiyorsak, CloudWatch ajanını yükleriz. Aynı şekilde, yerel sunucularımızın Sistem Yöneticisi ile çalışmasını istiyorsak, bu sunuculara Sistem Yöneticisi ajanını yükleriz ve böylece Sistem Yöneticisi ile çalışmaya hazır hale geliriz.

Run Command için IAM yetkilerini nasıl yapılandırabiliriz? Bunun için bir örnek profili yapılandırmamız gerekiyor. Örnek profili, EC2 örneklerimize başlatma sırasında bir IAM rolü eklememizi sağlar. Bu rolün içinde tabii ki IAM politikaları bulunur. Ana politika olan "AmazonSSMManagedInstanceCore", bir örneğin Sistem Yöneticisi'nin temel işlevselliğini kullanmasına izin verir, ancak politikaları özelleştirerek ihtiyaçlarımıza göre ayarlayabiliriz. Sistem Yöneticisi'nden Run Command çıkışı verebiliriz. Bu, Sistem Yöneticisi belgelerini gösteriyor. Bu belgeleri önceden oluşturulmuş belgeler olarak kullanabiliriz ve yönetim konsolunda bu önceden

oluşturulmuş belgeleri göreceğiz. Ya da kendi özel belgelerimizi oluşturabiliriz ve bu belgelere "komut belgeleri" denir. Demo'da da bunlardan birini kullanacağız.

Bu şekilde Sistem Yöneticisi Run Command ile ne yapabileceğimize dair basit bir bakış attık. Şimdi yönetim konsolundaki bir demo ile bu yetenekleri göstereceğiz ve bazı sınav ipuçlarıyla tamamlayacağız.

Systems Manager Parameter Store: Parameter Store, Systems Manager'ın sağladığı bir özellik olup yapılandırma verileri yönetimi ve gizli bilgi yönetimi için güvenli, hiyerarşik depolama sağlar. Bu dersde problemi tanımlayacağız. Şu anda üzerinde konuştuğumuz gibi bir problem olduğunu zaten belirttik. Bu diyagram da problemi basitçe özetliyor. Bir pipeline'imiz var, CodeBuild ve CodeDeploy. Bir buildspec dosyası kullanıyoruz ve bu buildspec dosyasında bir erişim anahtarı, gizli erişim anahtarı ve veritabanı parolası saklıyoruz. Hassas veri dosyalarını buildspec dosyamıza saklamak iyi bir fikir mi? Cevabı biliyoruz - hiç iyi bir fikir değil. İşte bu noktada Parameter Store devreye giriyor. Temel olarak, hassas bilgilerimizi Parameter Store'da depolayabilir ve kodumuzdan bu bilgilere referans verebiliriz. Özetlemek gerekirse, Parameter Store bize bunu yapma imkanı sağlar.

Sistem Yöneticisi ile Otomasyon: Tüm kurs boyunca otomasyon temasını işledik ve Sistem Yöneticisi'nde otomasyon da bir özellik olarak karşımıza çıkıyor. Sistem Yöneticisi içinde otomasyon, Run Command ve Uyumluluk gibi diğer yetenekler gibi bir yetenektir. Otomasyon, EC2 gibi hizmetler için yaygın görevleri, bakım görevlerini, dağıtım ve düzeltme görevlerini basitleştirebilir. Ancak sadece EC2 ile sınırlı değildir, RDS, Redshift, S3 gibi diğer hizmetlerle de çalışabilir. Bu derste otomasyonu ayrıntılı olarak inceleyeceğiz. Öncelikle bir genel bakış yapacak, otomasyonu inceleyeceğiz, ardından bu kavramları pekiştirmek için bir kullanım durumuna bakacağız. Daha sonra hızlı bir demo yapacağız ve sınav ipuçlarıyla derse son vereceğiz.

Sistem Yöneticisi Otomasyonu'nun temelinde otomasyon belgeleri (artık çalışma kitapları olarak adlandırılıyor) çalışır. Örneğin, ortamımızda bazı özel toplu işlem süreçleri için EC2 örnekleri var diyelim. Bu işlem süreci tamamlandığında, bu örnekleri kaldırmak istiyoruz. Belki hafta sonu çalışıyorlar ve kimse çalışmıyor, ancak EventBridge'i bu örnekleri izlemek üzere yapılandırabiliriz. Uygulama tamamlandığında EventBridge devreye girer ve bir Sistem Yöneticisi Otomasyonunu tetikler. Bu durumda otomasyon, bu örnekleri sonlandırmak olabilir. Yapay zeka destekli otomasyonumuz sayesinde hiçbir manuel kullanıcı müdahalesine gerek kalmaz. Tabii ki, otomasyonu yapılandırmamız gerekiyor, bunu demo sırasında inceleyeceğiz.

Otomasyon ile EC2 örneklerini durdurma, örnekleri sonlandırma gibi ortak IT görevlerini gerçekleştirebiliriz. Demoda farklı bir şey yapmak için EC2 örneklerini yeniden başlatacağız. Diğer görevler arasında, toplu olarak müdahale gerektiren görevleri gerçekleştirme bulunur. Örneğin, CloudFormation yığınlarını aynı anda güncelleyebiliriz. Yine EC2 örneklerini yeniden başlatma, AMI'ları güncelleme gibi görevler yapabiliriz. Bu otomasyonları yönetim konsoluna girerek çalıştırabileceğimizi unutmayın, ancak aynı zamanda tetiklenebilir ve otomatik olarak çalıştırılabilirler. Şimdi otomasyon için bir kullanım durumuna göz atalım.

Patch Manager: Patch Manager, Systems Manager'ın bir parçasıdır ve yönetilen düğümleri güvenlikle ilgili ve diğer tür güncellemelerle otomatik olarak yamalama işlemini gerçekleştirmemizi sağlar. Daha önceki Uyumluluk dersinden hatırlayabileceğiniz gibi, Patch Manager'da elde ettiğimiz uyumluluk bilgilerini Security Hub'a gönderebiliriz. Güvenliğimizi sıkılaştırmada yamalama kesinlikle önemli bir rol oynar.

Patch Manager'a genel bir bakış yapalım. Systems Manager'ın bir yeteneği olan "Bakım Penceresi" ile, düğümlerimizde potansiyel olarak kesintiye neden olabilecek işlemleri zamanlamamıza yardımcı olabiliriz. Bu işlemler, işletim sistemlerini yamalama, sürücüleri güncelleme, yazılım kurulumu gibi şeyler olabilir. Bu derste ise yamalar üzerinde duruyoruz. Bakım Penceresi, S3 depoları, SQS, Anahtar Yönetimi Hizmeti gibi diğer hizmetlerle birlikte çalışır, yalnızca EC2 örnekleriyle sınırlı değildir.

Yamalama işlemi için doğal olarak örneklerimizi yamalayacağız. Ayrıca, On-Premises sunucularımıza Systems Manager ile işlem yapabilmek için sadece Systems Manager Agent'ı yüklememiz gerekmektedir. Agent'ı eklediğimizde, bazı IAM yapılandırmalarının da yapılması gerekmektedir.

Patch Manager, yamaların otomatik olarak uygulanmasını sağlar ve yama geçmişi, uyumluluk durumu ve yama raporları gibi bilgileri sağlar. Bunlar, sistem yöneticilerinin yamaların durumunu izlemesine ve uyumluluk sorunlarını belirlemesine yardımcı olur.

AWS Config: AWS kaynaklarımızın yapılandırmalarını değerlendirmemize, denetlememize ve incelememize olanak sağlayan bir hizmettir. Config, AWS'deki kaynak yapılandırmalarını sürekli olarak izler ve kaydeder ve yapılandırmalarımızın değerlendirmesine dayalı otomasyonlar kurmamıza olanak tanır. Bu derste AWS Config'a genel bir bakış atacak, daha derinlere inecek ve yönetim konsolundan bir demo ile bilgilerimizi pekiştireceğiz. Ayrıca sınav ipuçları da paylaşacağız.

AWS Config ile AWS hesaplarımızda bir varlık envanteri oluşturabilir ve Config, yapılandırmaları kaydedip değerlendirmemize olanak sağlar. Değerlendirmeyi nasıl yaparız? Tabii ki Config izleyicisine giderek sonuçları görüntüleyebiliriz, ancak aynı zamanda yapılandırmalarımızın değerlendirmesine dayalı bazı otomasyonları da kurabiliriz. Config, sistemimizden geçen tüm yapılandırma verilerini kaydeder ve Config ile uyumluluğu sağlamak için kurallar oluşturabiliriz.

AWS Config'i birden çok hesap ve bölge üzerinde kullanabilir miyiz? Cevap kesinlikle evettir. Birleştirici, birden çok hesap ve bölge üzerinde yapılandırma ve uyumluluk verilerini toplayan bir AWS Config kaynağıdır. Bu verileri birden çok hesap ve bölgeden alarak birleştirir. Dolayısıyla AWS Config'i birden çok hesap ve bölgede kullanabilir ve verileri birleştirebiliriz.

Uyumluluk Değerlendirmeleri: AWS Config, yapılandırmalarınızı belirli kurallara göre değerlendirebilir. Örneğin, güvenlik kuralları, maliyet optimizasyonu kuralları, performans kuralları vb. oluşturabilirsiniz. Bu kurallar, kaynak yapılandırmalarınızı analiz eder ve belirli bir uyumluluk durumunu kontrol eder.

Olay Tetiklemeleri: AWS Config, yapılandırma değişiklikleri veya uyumluluk durumu değişiklikleri gibi olaylara dayalı tetikleyiciler sağlar. Bu sayede, belirli bir olay gerçekleştiğinde otomatik tepkiler oluşturabilirsiniz. Örneğin, belirli bir kaynak yapılandırması değiştirildiğinde bir SNS bildirimi göndermek gibi.

Raporlama ve Analiz: AWS Config, kaynaklarınızın geçmiş durumunu anlamak, değişiklikleri izlemek ve raporlamak için güçlü analiz ve sorgulama yetenekleri sunar. Bu sayede, kaynak yapılandırmalarınızı daha iyi yönetebilir ve kontrol edebilirsiniz.

Amazon Macie: Tamamen yönetilen bir veri güvenliği ve veri gizliliği hizmetidir. Makine öğrenimi ve desen eşleştirme kullanarak AWS üzerindeki hassas verilerinizi keşfeder ve korur. Özellikle Amazon S3 kovalarındaki verileri analiz eder ve korur.

Amazon Macie, hassas verileri tanımlayabilme yeteneğine sahiptir. Örneğin, kişisel tanımlanabilir bilgiler gibi hassas verileri tespit edebilir ve size düzeltici önlemler alma konusunda yardımcı olur. Ayrıca, veriye nasıl erişildiği konusunda görünürlük sağlayan panolar ve uyarılar sunar.

Macie'yi etkinleştirdiğinizde, Macie otomatik olarak bir hizmete bağlı rol oluşturur. Bu rol, Macie'nin S3'te depoladığınız veriler hakkında bilgi toplama, S3 kovalarınızı değerlendirme ve izleme gibi görevleri gerçekleştirmesine izin verir.

AWS CloudHSM: AWS Directory Service for Microsoft Active Directory ve Resource Access Manager gibi hizmetleri inceleyeceğiz.

İlk olarak, AWS CloudHSM'ye bakalım. AWS CloudHSM, güvenli şifreleme anahtarları yönetimi ve depolama için kullanılan bir kurumsal sınıf hizmettir. Bu hizmet, sizin sahip olduğunuz ve tek kiracı olduğunuz bir donanım cihazıdır. Yani CloudHSM, gerçek bir donanım cihazıdır ve müşteriye aittir. Şifreleme anahtarlarınızı güvenli bir şekilde işleyen ve depolayan bir hesaplama cihazıdır.

Bu noktada, paylaşılan sorumluluk modeline bir göz atalım. Donanım cihazını kim yönetir? Unutmayın, bu cihaz size aittir ve müşteri tarafından yönetilir. Şifreleme anahtarlarını kim yönetir? Siz, yani cihazın sahibi.

Son olarak, AWS Resource Access Manager'a bakalım. Bu hizmet, bir kuruluş veya organizasyon birimleri içindeki AWS hesapları arasında kaynakları güvenli bir şekilde paylaşmanıza yardımcı olur. Örneğin, Account A'da kaynaklarınızı oluşturduktan sonra, diğer hesapların bu kaynakları kullanmasını istiyorsanız, AWS Resource Access Manager'ı kullanabilirsiniz. Bu hizmetle, paylaşmak istediğiniz kaynakları seçer, uygun izinleri belirlersiniz ve ilgili kullanıcıları (IAM kullanıcıları, organizasyonlar veya IAM grupları) belirtirsiniz.

Lifecycle Hooks: Auto Scaling grupları, CPU kullanımına bağlı olarak ölçeklendirme yapabilen yapılar olarak düşünelim. Örneğin, CPU kullanımı %80'e çıktığında ölçeklendirme yaparak yeni bir örneği devreye alıyoruz ve CPU kullanımı %20'ye düştüğünde ise ölçeklendirmeyi geri alarak bir örneği kapatıyoruz.

Auto Scaling lifecycle hook'larını kullanarak hangi sorunu çözmeye çalışıyoruz? Aslında, lifecycle hook'ları kullanarak iki şey yapabiliriz:

Lifecycle hook'lar sayesinde örneği başlatma veya sonlandırma aşamalarında değişiklikler yapabiliriz. Önemli olan, örneğin oluşturulmasını veya sonlandırılmasını duraklatma gücünü elde etmek ve ardından lifecycle hook'ların gerektiği süre boyunca işleri kontrol etmesini sağlamaktır.

Lifecycle hook'ları kullanmanın iki yolu vardır: başlatma (launch) lifecycle hook'ları ve sonlandırma (termination) lifecycle hook'ları.

Başlatma aşamasında örneğin oluşturulması duraklatılabilir ve "Beklemede: Bekleme" durumunda tutulabilir. Bu durumda, örneği kullanıma hazır hale getirmek için gerekli hazırlıkları yapabiliriz. Örneğin, yazılımı yükleyebiliriz. Yazılım yüklenmesi 5 ila 10 dakika kadar sürebilir, ancak load balancer örneği kullanıma hazır olarak işaretleyebilir ve bu durumda kullanıcılarımız sorunlarla karşılaşabilir. Bu nedenle, lifecycle hook örneği duraklatır ve yazılımın yüklenmesini bekler. Yazılım yüklendikten sonra, lifecycle hook Auto Scaling grubuna "Başarılı!" şeklinde bir sinyal gönderir ve durum "Beklemede: İlerleme" haline gelir. Artık örnek hizmet vermek üzere hazırdır.

Sonlandırma aşamasında ise, örneği sonlandırmadan önce yapmak istediğimiz işlemler için örneği duraklatabiliriz. Örneğin, log dosyalarını almak istiyor olabiliriz. Bu durumda, lifecycle hookörneği duraklatır ve log dosyalarını almak için gerekli işlemleri yapar. İşlemler tamamlandıktan sonra, lifecycle hook Auto Scaling grubuna "Tamamlandı!" şeklinde bir sinyal gönderir ve örneğin sonlandırılmasına devam edilir.

Lifecycle hook'larla ilişkili zaman aşımı süreleri vardır. Örneğin, başlatma aşamasında belirli bir süre içinde örneği kullanıma hazır hale getirmeniz beklenir. Aksi takdirde, Auto Scaling grubu örneği başarısız olarak işaretleyebilir ve yeni bir örnek oluşturabilir. Bu nedenle, işlemlerinizin zaman aşımı sürelerini dikkate almalı ve gerektiğinde süreyi uzatmalısınız.

Lifecycle hook'lar, Auto Scaling grubunun olaylarını dinleyen bir hedefle ilişkilendirilir. Bu hedef genellikle AWS Lambda işlevi olabilir. Lifecycle hook, durumu değiştirdiğinde belirli bir Lambda işlevini tetikleyecektir. Bu işlev, örneği duraklatma, işlemleri yapma ve durumu güncelleme gibi işlemleri gerçekleştirebilir.

DynamoDB Streams, bir tabloya yapılan değişikliklerin olay tabanlı bir şekilde yakalanmasını sağlar. Örneğin, bir tabloya yeni bir kayıt eklediğimizde veya mevcut bir öğeyi değiştirdiğimizde, bu değişikliklerin bir olay olarak yayınlanmasını sağlar. Bu olaylar, Lambda fonksiyonlarını tetikleyebilir veya Kinesis veri akışlarına beslenebilir. DynamoDB Streams ayrıca Global Tablolar ile birlikte kullanılabilir. Global Tablolar, birden çok AWS bölgesindeki replika tablolardan oluşur ve DynamoDB bunları tek bir birim gibi yönetir. Bir tabloya yazılan bilgiler diğer tablolara da replike edilir ve bu işlemleri DynamoDB otomatik olarak gerçekleştirir. Bu, kullanıcı deneyimini iyileştirir

ve yanıt sürelerini hızlandırır. Global Tablolar ve DynamoDB Streams, felaket kurtarma stratejimizi güçlendirmek için büyük bir avantaj sağlar.

DAX (DynamoDB Accelerator), DynamoDB için bir önbellek çözümüdür. DAX, tamamen yönetilen bir hizmettir ve DynamoDB için yüksek performanslı bir in-memory önbellek sağlar. Bu sayede DynamoDB'nin yanıt sürelerinde 10 kata kadar performans artışı elde edilebilir. DAX, özellikle tekrarlayan büyük veri okumaları için kullanılır ve Microseconds (mikrosaniye) performans gerektiren durumları destekler.

TTL (Time to Live), bir DynamoDB tablosundaki öğelerin belirli bir süre sonra otomatik olarak silinmesini sağlar. Bir öğe için bir TTL değeri belirlendiğinde, bu öğe belirtilen süre sonunda tablodan otomatik olarak kaldırılır. TTL olayları, Lambda fonksiyonlarını tetikleyebilir ve bu sayede silinen veri üzerinde temizlik veya analiz işlemleri gerçekleştirilebilir. TTL ile silinen öğeler tabloda hâlâ mevcuttur, ancak tamamen silinene kadar kullanılabilirler. Örneğin, bir e-ticaret sitesinde alışveriş sepetine eklenen bir ürün, belirli bir süre sonra otomatik olarak sepetten kaldırılabilir.

Sınav ipuçları olarak, DynamoDB Streams'in nasıl çalıştığını, Global Tabloların nasıl yapılandırıldığını, DAX'in mikrosaniye performans gerektiren durumlar için kullanıldığını ve TTL'nin olay tabanlı otomasyon için nasıl kullanılabileceğini anlamamız önemlidir. Ayrıca, Lambda fonksiyonlarının güçlü bir araç olduğunu ve Python ile yazılan her şeyi gerçekleştirebileceğimizi unutmamalıyız. Bu bilgiler, DynamoDB'nin önemli araçları hakkında genel bir bakış sağlamaktadır.

Felaket Kurtarma: Felaket kurtarma, bir felaket durumunda geri dönme noktası ve kurtarma süresi gibi iki ana kavramı içerir. Geri dönme noktası, felaketin meydana geldiği zamandan geriye doğru bir zaman çizelgesidir ve veri kaybımızı belirler. Kurtarma süresi ise normal işleyişe geri dönme süresidir. AWS'de, bu iki kavrama dayanarak felaket kurtarma için çeşitli teknikler kullanabiliriz.

Sınav ipuçlarına gelince, felakete karşı toleransımız, felaket kurtarma için kullanacağımız araçları belirleyecektir. Kurtarma süresi hedefimize (RTO) ve veri kaybı toleransımıza (RPO) bağlı olarak farklı teknikler kullanabiliriz. Örneğin, RTO çok düşükse, verilerimizi çoklu bölgeler arasında senkronize olarak replike etmemiz gerekebilir. Ancak RTO daha yüksekse, otomatik yedekleme ve kurtarma çözümleri daha uygun ve ekonomik olabilir.

CloudFormation şablonları, felaket kurtarma için mükemmel bir araçtır, ancak bu çözümün düşmanı "stack drift"tir. Bu nedenle, şablonlarımızı başka bir bölgede depolarken, mevcut mimarimizi doğru bir şekilde temsil ettiğinden emin olmalıyız. Ayrıca, RTO hedefimiz orta seviyede ise, AMI'ler aracılığıyla başka bir bölgede örneklerimizi çoğaltabiliriz.

ROSA: Red Hat OpenShift Service on AWS'nin (AWS üzerindeki Red Hat OpenShift Hizmeti) kısaltmasıdır. Red Hat OpenShift, geliştiricilerin uygulama geliştirmesine yardımcı olan bir bulut tabanlı Kubernetes platformudur. ROSA, bu platformun AWS versiyonunu sağlayarak OpenShift ile entegre bir deneyim sunar. AWS'deki çeşitli hizmetlerle entegrasyon sağlayabiliriz. ROSA, Red Hat ve AWS tarafından desteklenen tamamen yönetilen bir OpenShift hizmetidir.

AWS Veri Taşıma Hizmeti (DMS), ilişkisel veritabanları, veri depolarını, NoSQL veritabanlarını ve diğer veri depolama türlerini taşımayı mümkün kılan bir bulut hizmetidir. DMS ile buluta (AWS'ye) veya bulut ile On-Premises kombinasyonları arasına taşıma yapabiliriz. Örneğin, bir On-Premises MySQL veritabanını Amazon Aurora'ya taşımak için DMS kullanabiliriz. Taşıma görevi, kaynaktan hedefe veriyi dönüştürür ve yükler. Aynı türden bir taşıma olduğunda (örneğin MySQL'den MySQL'e), bu homojen bir taşınma olarak adlandırılır. Farklı türler arasındaki taşıma (örneğin Oracle veritabanından Amazon Aurora'ya) ise heterojen bir taşınma olarak adlandırılır.

AWS Transit Gateway, VPC'leri ve On-Premises ağları merkezi bir hub üzerinden birbirine bağlar. Bu bağlantılar ağımızı basitleştirir. Transit Gateway, VPC peering'in ölçeklenebilirliğini geliştiren bir iyileştirmedir. Transit Gateway ile birçok VPC'yi ve On-Premises'ı birbirine bağlayabiliriz. Örneğin, bir On-Premises veri merkezi ile birçok VPC'yi Transit Gateway üzerinden bağlayabiliriz. Transit Gateway, ölçekte daha basit bir çözümdür. Ancak, VPC peering daha ucuzdur.

Sınav ipuçları bölümünde, EKS Distro'nun kısa ismi "EKS-D" olarak geçer. ROSA'nın desteklediği hizmet Kubernetes'tir ve ROSA ile Red Hat ve AWS birlikte çalışır. AWS DMS'nin tipik kaynakları On-Premises ve hedefleri ise AWS bulutu veya buluttan buluta taşımadır. AWS Transit Gateway'in VPC peering'e göre avantajı, ölçeklenebilir organizasyonlarda, çok sayıda bağlantı ve çoklu bölgelerde daha basit bir çözüm sunmasıdır.