**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. KMS anahtarları bölgesel bazda çalışır, yani bir bölgede bir veritabanı oluşturduğunuzda o bölgede bir anahtar oluşturulur.

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. 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. 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.

**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.

**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.

**EC2 Image Builder kullanarak EC2 İmaj Oluşturmayı Otomatikleştirme:** EC2 Image Builder ile Amazon Makine İmajı (AMI) olarak adlandırılan temel imajları otomatik olarak oluşturabiliriz.

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. 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**: 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:** 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.

**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.

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.

**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.

**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. 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.

**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. 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. 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.

**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.

**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. 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.

**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.

**Elastic Container Service:** ECS, Docker konteynerlerini destekleyen, yüksek ölçeklenebilir ve yüksek performanslı bir konteyner yönetim hizmetidir. 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, 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.

**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. 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.

**AWS AppConfig:** AppConfig, AWS Systems Manager'ın bir parçasıdır ve uygulama yapılandırmalarını oluşturmayı, yönetmeyi ve hızlı bir şekilde dağıtmayı sağlamak için kullanılır. 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.

**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'ın temel amacı, çoklu hesaplara uygulamaları dağıttığımızda ortaya çıkan karmaşıklığı yönetmek ve çevremizi yönetmektir.

**Amazon Detective:** Amazon Detective, güvenlik bulgularını veya şüpheli aktiviteleri analiz etmenize, araştırmanıza ve hızlı bir şekilde kök nedenini belirlemenize yardımcı olan bir hizmettir. CloudTrail günlükleri, VPC Akış günlükleri, GuardDuty ve Security Hub gibi hizmetlerden veri alır ve bu verileri birleştirerek etkileşimli grafikler oluşturur.

**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.

**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.

**AWS CloudTrail:** AWS CloudTrail, AWS hesabınızın yönetişimini, uyumluluğunu ve risk denetimini sağlayan bir hizmettir. 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, 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. 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.

**AWS Lambda ile Kinesis:** 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.

**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. CloudTrail logları için de aynı durum geçerlidir.

**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.

**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.

**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.

**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.

**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. 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.

**Amazon Inspector:** EC2 örneklerinin ağ erişilebilirliğini ve bu örneklerde çalışan uygulamaların güvenlik durumunu test eder. 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.

**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. Ve şimdiye kadar muhtemelen cevabı biliyorsunuz, kesinlikle. Bu, kullanabileceğimiz bazı araçları ve bu araçları nasıl kullanarak otomatikleştireceğimizi bilmekle ilgilidir.

**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. 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. CloudWatch ve EventBridge ile entegrasyon sağlar ve bu entegrasyonlar sayesinde otomatik olarak bir iş kalemi oluşturur.

**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 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 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.

**AWS Organizations:** 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.

**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.

**Systems Manager:** 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.

**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.

**Sistem Yöneticisi ile Otomasyon:** 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.

**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. 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.

**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.

**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. 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.

**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. 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. 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'larla ilişkili zaman aşımı süreleri vardır. Örneğin, başlatma aşamasında belirli bir süre içinde örneği kullanıma hazır hale getirmeniz beklenir.

**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. Global Tablolar, birden çok AWS bölgesindeki replika tablolardan oluşur ve DynamoDB bunları tek bir birim gibi yönetir.

**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.

**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. 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.