AWS 백서

AWS의 배포 옵션 개요

AWS의 배포 옵션 개요: AWS 백서

Table of Contents

AWS의 배포 옵션 개요

게시 날짜: 2024년 5월 31일(문서 수정)

요약

Amazon Web Services(AWS)는 인프라를 프로비저닝하고 애플리케이션을 배포하기 위한 여러 옵션을 제공합니다. 애플리케이션 아키텍처가 간단한 3계층 웹 애플리케이션이든 복잡한 워크로드 세트이든 상관없이 AWS는 애플리케이션 및 조직의 요구 사항을 충족하는 배포 서비스를 제공합니다.

이 백서는 AWS에서 제공하는 다양한 배포 서비스에 대한 개요를 찾고 있는 개인을 대상으로 합니다. 이러한 배포 서비스에서 사용할 수 있는 일반적인 기능을 설명하고 애플리케이션 스택을 배포하고 업데이트하기 위한 기본 전략을 설명합니다.

소개

애플리케이션을 위한 배포 솔루션을 설계하는 것은 AWS에서 잘 설계된 애플리케이션을 구축하는 데 중요한 부분입니다. 애플리케이션과 필요한 기본 서비스의 특성에 따라 AWS 서비스를 사용하여 애플리케이션과 조직의 요구 사항에 맞게 조정할 수 있는 유연한 배포 솔루션을 만들 수 있습니다.

지속적으로 증가하는 AWS 서비스 카탈로그는 애플리케이션 아키텍처를 구성할 서비스를 결정하는 프로세스뿐만 아니라 애플리케이션을 생성, 관리 및 업데이트하는 방법을 결정하는 프로세스를 복잡하게 만듭니다. AWS에서 배포 솔루션을 설계할 때는 솔루션이 다음 기능을 해결하는 방법을 고려해야 합니다.

- 프로비저닝 - 애플리케이션에 필요한 원시 인프라 또는 관리형 서비스 인프라를 생성합니다.
- 구성 - 환경, 런타임, 보안, 가용성, 성능, 네트워크 또는 기타 애플리케이션 요구 사항에 따라 인프라를 사용자 지정합니다.
- 배포 - 인프라 리소스에 애플리케이션 구성 요소를 설치 또는 업데이트하고 이전 애플리케이션 버전에서 새 애플리케이션 버전으로의 전환을 관리합니다.
- 규모 조정 - 사용자 정의 기준 집합에 따라 애플리케이션에 사용할 수 있는 리소스의 양을 사전에 또는 사후 대응적으로 조정합니다.
- 모니터링 - 애플리케이션 아키텍처의 일부로 시작되는 리소스에 대한 가시성을 제공합니다. 리소스 사용량, 배포 성공 또는 실패, 애플리케이션 상태, 애플리케이션 로그, 구성 드리프트 등을 추적합니다.

이 백서는 AWS에서 제공하는 배포 서비스를 강조하고 모든 유형의 애플리케이션에 대한 성공적인 배포 아키텍처를 설계하기 위한 전략을 간략하게 설명합니다.

AWS 배포 서비스

확장 가능하고 효율적이며 비용 효율적인 배포 솔루션을 설계하는 작업은 애플리케이션 버전을 업데이트하는 방법에 국한되지 않고 전체 애플리케이션 수명 주기 동안 지원 인프라를 관리하는 방법도 고려해야 합니다. 리소스 프로비저닝, 구성 관리, 애플리케이션 배포, 소프트웨어 업데이트, 모니터링, 액세스 제어 및 기타 문제는 모두 배포 솔루션을 설계할 때 고려해야 할 중요한 요소입니다.

AWS 서비스는 애플리케이션 수명 주기의 하나 이상의 측면에 대한 관리 기능을 제공할 수 있습니다. 원하는 제어 균형(리소스 수동 관리)과 편의성(리소스 AWS 관리) 및 애플리케이션 유형에 따라 이러한 서비스를 단독으로 사용하거나 결합하여 기능이 풍부한 배포 솔루션을 생성할 수 있습니다. 이 섹션에서는 조직이 애플리케이션을 보다 빠르고 안정적으로 구축하고 제공할 수 있도록 하는 데 사용할 수 있는 AWS 서비스에 대한 개요를 제공합니다.

AWS CloudFormation

AWS CloudFormation는 고객이 YAML 또는 JSON으로 표현된 사용자 지정 템플릿 언어를 사용하여 거의 모든 AWS 리소스를 프로비저닝하고 관리할 수 있는 서비스입니다. 템플릿은 CloudFormation 스택이라는 그룹에 인프라 리소스를 생성하고 이러한 리소스를 완전히 제어하면서 애플리케이션을 운영하는 데 필요한 모든 구성 요소를 정의하고 사용자 지정할 수 있습니다. 템플릿을 사용하면 인프라에서 버전 관리를 구현할 수 있는 기능과 인프라를 빠르고 안정적으로 복제할 수 있는 기능이 도입됩니다.

CloudFormation 는 라우팅 테이블 또는 서브넷 구성과 같은 하위 수준 구성 요소부터 CloudFront 배포와 같은 상위 수준 구성 요소에 이르기까지 모든 애플리케이션 인프라 구성 요소의 프로비저닝 및 관리에 대한 세분화된 제어를 제공합니다. CloudFormation 는 일반적으로 다른 AWS 배포 서비스 또는 타사 도구와 함께 사용되며, 보다 전문화된 배포 서비스와 결합하여 CloudFormation 인프라 구성 요소에 대한 애플리케이션 코드 배포를 관리합니다.

AWS는 기본 기능 외에도 CloudFormation 서비스에 대한 확장을 제공합니다.

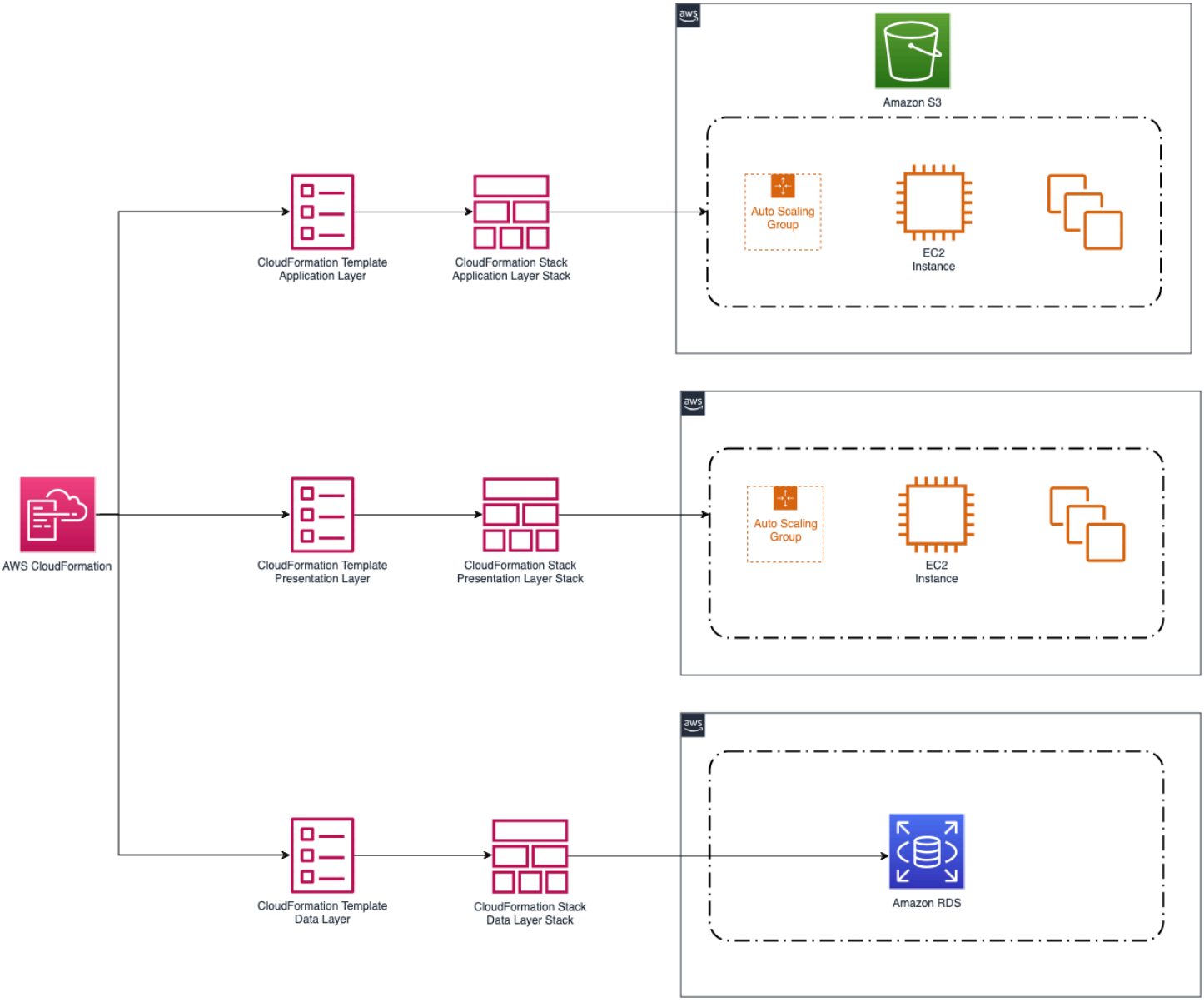
- AWS Cloud Development Kit (AWS CDK)는 TypeScript, JavaScript, Python, Java 또는 C#/를 사용하여 AWS 인프라를 프로그래밍 방식으로 모델링하는 오픈 소스 소프트웨어 개발 키트(SDK)입니다.NET.
- AWS Serverless Application Model (AWS SAM)는 AWS에서 서버리스 애플리케이션 구축을 간소화하는 오픈 소스 프레임워크입니다. 함수, API, 데이터베이스 및 이벤트 소스 매핑을 표현하는 속기 구문을 제공합니다.

표 1: AWS CloudFormation 배포 기능

기능	설명
프로비저닝	CloudFormation은 템플릿에 정의된 인프라 구성 요소를 자동으로 생성하고 업데이트합니다. CloudFormation 템플릿을 사용하여 인프라를 생성하는 방법에 대한 자세한 내용은 AWS CloudFormation 모범 사례를 참조하세요.
구성	CloudFormation 템플릿은 모든 인프라 구성 요소를 사용자 지정하고 업데이트할 수 있는 광범위한 유연성을 제공합니다. CloudFormation 템플릿 사용자 지정에 대한 자세한 내용은 템플릿 구조를 참조하세요.
배포	CloudFormation 템플릿을 업데이트하여 스택의 리소스를 변경합니다. 애플리케이션 아키텍처에 따라 인프라에서 실행되는 애플리케이션 버전을 업데이트하려면 추가 배포 서비스가 필요할 수 있습니다. 를 배포 솔루션으로 사용하는 방법에 대한 자세한 내용은를 사용하여 Amazon EC2에 애플리케이션 AWS CloudFormation 배포를 참조 CloudFormation 하세요.
규모 조정	CloudFormation 는 사용자를 대신하여 인프라 조정을 자동으로 처리하지 않지만 CloudFormation 템플릿에서 리소스에 대한 Auto Scaling 정책을 구성할 수 있습니다.
모니터링	CloudFormation 는 템플릿에 정의된 인프라 업데이트의 성공 또는 실패에 대한 기본 모니터링과 템플릿에 정의된 리소스가 사양을 충족하지 않는 경우 모니터링할 드리프트 감지를 제공합니다. 애플리케이션 수준 모니터링 및 지표를 위해 추가 모니터링 솔루션을 마련해야 합니다.

기능	설명
	가 인프라 업데이트를 모니터링하는 방법에 대한 자세한 내용은 스택 업데이트 진행 상황 CloudFormation 모니터링을 참조하세요.

다음 다이어그램은의 일반적인 사용 사례를 보여줍니다 CloudFormation. 여기서 CloudFormation 템플릿은 간단한 3계층 웹 애플리케이션을 생성하는 데 필요한 모든 인프라 구성 요소를 정의하기 위해 생성됩니다. 이 예제에서는에 정의된 부트스트랩 스크립트 CloudFormation 를 사용하여 최신 버전의 애플리케이션을 Amazon EC2 인스턴스에 배포하지만, 추가 배포 서비스를와 결합하는 것도 일반적인 관행입니다 CloudFormation (인프라 관리 및 프로비저닝 기능에 CloudFormation 만 사용). 인프라를 생성하는 데 두 개 이상의 CloudFormation 템플릿이 사용됩니다. 다이어그램에서 CloudFormation 는 IAM 역할, VPCs, 서브넷, 라우팅 테이블, 보안 그룹 및 Amazon S3 버킷 정책을 포함한 모든 인프라 구성 요소를 생성하는 데 사용됩니다. 별도의 CloudFormation 템플릿을 사용하여 애플리케이션 아키텍처의 각 도메인을 빌드합니다.



AWS CloudFormation 사용 사례

AWS Elastic Beanstalk

AWS Elastic Beanstalk는 Apache, Nginx, Passenger 및 IIS와 같은 친숙한 서버에서 Java, .NET, .NET Core, PHP, Node.js, Python, Ruby, Go 또는 Docker로 개발된 웹 애플리케이션 및 서비스를 배포하고 확장하기 위한 easy-to-use 서비스입니다. Elastic Beanstalk는 완벽한 애플리케이션 관리 솔루션으로, 사용자를 대신하여 모든 인프라 및 플랫폼 작업을 관리합니다.

Elastic Beanstalk를 사용하면 인프라 관리의 운영 부담 없이 애플리케이션을 빠르게 배포, 관리 및 확장할 수 있습니다. Elastic Beanstalk는 웹 애플리케이션의 관리 복잡성을 줄여 AWS를 처음 사용하거나 웹 애플리케이션을 최대한 빨리 배포하려는 조직에 적합합니다.

Elastic Beanstalk를 배포 솔루션으로 사용하는 경우 소스 코드를 업로드하면 Elastic Beanstalk가 서버, 데이터베이스, 로드 밸런서, 네트워크 및 오토 스케일링 그룹을 포함하여 필요한 모든 인프라를 프로비저닝하고 운영합니다. 이러한 리소스는 사용자를 대신하여 생성되지만 이러한 리소스를 완전히 제어할 수 있으므로 개발자가 필요에 따라 사용자 지정할 수 있습니다. Elastic Beanstalk는 HIPAA 자격 기준과 함께 ISO, PCI, SOC 1, SOC 2 및 SOC 3 규정 준수 기준을 충족합니다. 즉, Elastic Beanstalk에서 실행되는 애플리케이션은 규제 대상 금융 데이터 또는 보호 대상 건강 정보(PHI)를 처리할 수 있습니다.

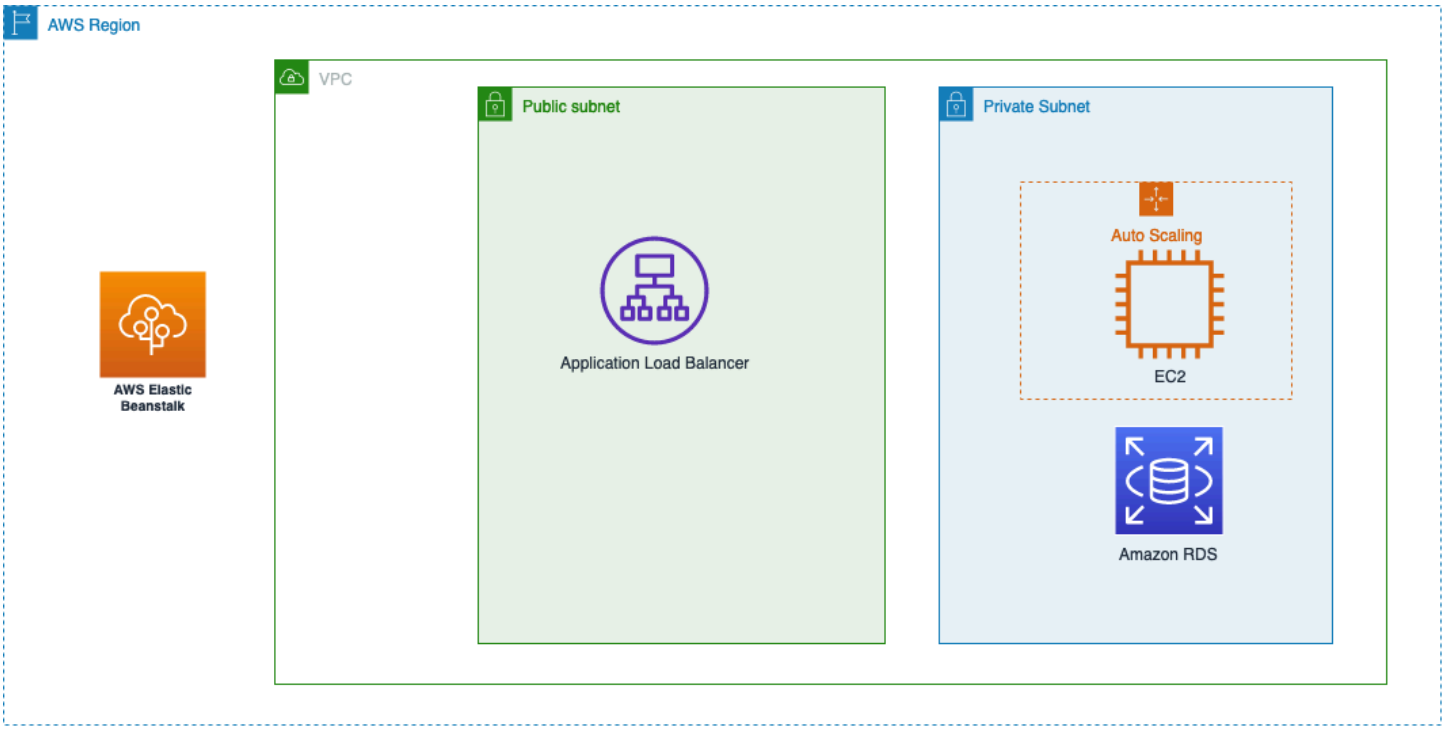
표 2: AWS Elastic Beanstalk 배포 기능

기능	설명
프로비저닝	Elastic Beanstalk는 지원되는 플랫폼 중 하나에서 실행되는 웹 애플리케이션 또는 서비스를 운영하는 데 필요한 모든 인프라 구성 요소를 생성합니다. 추가 인프라가 필요한 경우 Elastic Beanstalk 외부에서 생성해야 합니다. Elastic Beanstalk에서 지원하는 웹 애플리케이션 플랫폼에 대한 자세한 내용은 Elastic Beanstalk 플랫폼을 참조하세요.
구성	Elastic Beanstalk는 환경의 리소스를 사용자 지정하기 위한 다양한 옵션을 제공합니다. Elastic Beanstalk에서 생성한 리소스를 사용자 지정하는 방법에 대한 자세한 내용은 Elastic Beanstalk 환경 구성을 참조하세요.
배포	Elastic Beanstalk는 애플리케이션 배포를 자동으로 처리하고 기존 사용자에게 영향을 주지 않고 애플리케이션의 새 버전을 실행하는 환경을 생성합니다.

기능	설명
	Elastic Beanstalk를 사용한 애플리케이션 배포 AWS Elastic Beanstalk에 대한 자세한 내용은에 애플리케이션 배포를 참조하세요.
규모 조정	Elastic Beanstalk는 Elastic Load Balancing 및 Auto Scaling을 사용하여 특정 요구 사항에 따라 애플리케이션을 자동으로 확장 및 축소합니다. 여러 가용 영역에서 애플리케이션 안정성과 가용성을 개선할 수 있습니다. Elastic Beanstalk를 사용한 Auto Scaling에 대한 자세한 내용은 Elastic Beanstalk 환경의 Auto Scaling 그룹을 참조하세요.
모니터링	Elastic Beanstalk는 배포 성공/실패, 환경 상태, 리소스 성능 및 애플리케이션 로그를 포함한 애플리케이션에 대한 기본 제공 환경 모니터링을 제공합니다. Elastic Beanstalk를 사용한 전체 스택 모니터링에 대한 자세한 내용은 환경 모니터링을 참조하세요.
Graviton 지원	AWS Graviton arm64 기반 프로세서는 Amazon EC2에서 실행되는 클라우드 워크로드에 최상의 가격 대비 성능을 제공합니다. Elastic Beanstalk의 AWS Graviton을 사용하면 워크로드의 최적화 요구 사항을 충족하고 유사한 x86 기반 프로세서에 비해 향상된 가격 대비 성능의 이점을 누릴 수 있도록 Amazon EC2 인스턴스 유형을 선택할 수 있습니다.

Elastic Beanstalk를 사용하면 AWS에서 웹 애플리케이션을 빠르게 배포하고 관리할 수 있습니다. 다음 예제에서는 간단한 웹 애플리케이션을 배포하는 데 사용되므로 Elastic Beanstalk의 일반적인 사용 사례를 보여줍니다. 모든 애플리케이션 인프라(보안 그룹, IAM 역할 및 CloudWatch 경보 포함)는 Elastic Beanstalk에서 생성하고 관리합니다. Amazon EC2 인스턴스는 런타임 환경 및 배포 패키지로 자동으

로 프로비저닝됩니다. Elastic Beanstalk 환경은 Elastic Beanstalk 외부에서 생성된 Amazon Relational Database Service(RDS)와 같은 리소스와 통합할 수 있습니다.



AWS Elastic Beanstalk 사용 사례

AWS CodeDeploy

AWS CodeDeploy는 Amazon EC2, Amazon Elastic Container Service(Amazon ECS), AWS Lambda또는 온프레미스 서버와 같은 서비스를 계산하기 위해 애플리케이션 배포를 자동화하는 완전 관리형 배포 서비스입니다. 조직은 CodeDeploy를 사용하여 애플리케이션 배포를 자동화하고 배포 프로세스에서 오류가 발생하기 쉬운 수동 작업을 제거할 수 있습니다. CodeDeploy는 코드, 서버리스 함수, 구성 파일 등 다양한 애플리케이션 콘텐츠와 함께 사용할 수 있습니다.

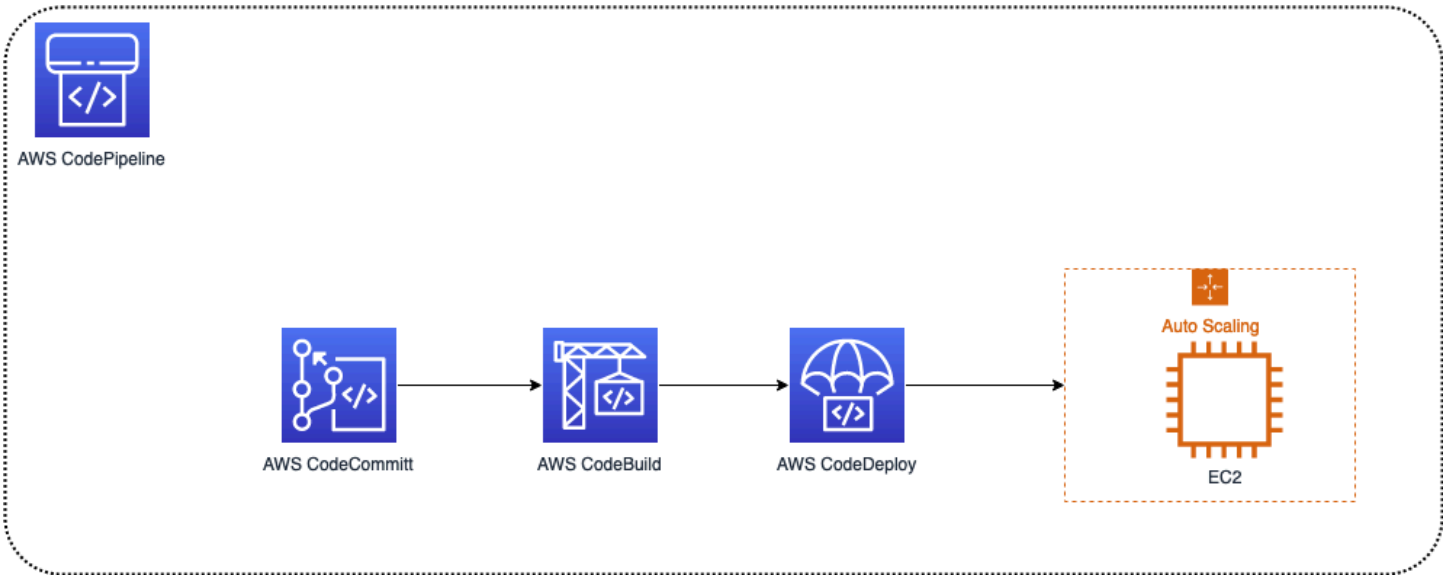
CodeDeploy는 애플리케이션 개발자가 기존 인프라에서 실행 중인 소프트웨어를 배포하고 업데이트할 수 있도록 지원하는 데 중점을 둔 빌딩 블록 서비스로 사용하기 위한 것입니다. end-to-end 애플리케이션 관리 솔루션이 아니며 전체 CI/CD 파이프라인의 일부로 AWS CodeStar, , 기타 AWS 개발자 도구 및 타사 서비스와 같은 AWS CodePipeline다른 AWS 배포 서비스와 함께 사용하기 위한 것입니다(AWS CodeDeploy 제품 통합의 전체 목록은 제품 통합 참조). https://aws.amazon.com/products/developer-tools/ 또한 CodeDeploy는 사용자를 대신하여 리소스 생성을 관리하지 않습니다.

표 3: AWS CodeDeploy 배포 기능

기능	설명
프로비저닝	CodeDeploy는 기존 컴퓨팅 리소스와 함께 사용하기 위한 것이며 사용자를 대신하여 리소스를 생성하지 않습니다. CodeDeploy에서는 애플리케이션 콘텐츠를 배포하기 위해 컴퓨팅 리소스를 배포 그룹이라는 구문으로 구성해야 합니다. CodeDeploy를 컴퓨팅 리소스에 연결하는 방법에 대한 자세한 내용은 CodeDeploy의 배포 그룹 작업을 참조하세요. CodeDeploy
구성	CodeDeploy는 애플리케이션 사양 파일을 사용하여 컴퓨팅 리소스에 대한 사용자 지정을 정의합니다. CodeDeploy를 사용한 리소스 사용자 지정에 대한 자세한 내용은 CodeDeploy AppSpec 파일 참조를 참조하세요. CodeDeploy
배포	CodeDeploy가 사용되는 컴퓨팅 리소스 유형에 따라 CodeDeploy는 애플리케이션을 배포하기 위한 다양한 전략을 제공합니다. 지원되는 배포 프로세스 유형에 대한 자세한 내용은 CodeDeploy에서 배포 작업을 참조하세요.
규모 조정	CodeDeploy는 기본 애플리케이션 인프라의 규모 조정을 지원하지 않지만 배포 구성에 따라 블루/그린 배포를 지원하는 추가 리소스를 생성할 수 있습니다.
모니터링	CodeDeploy는 배포의 성공 또는 실패를 모니터링할 수 있으며 모든 배포 기록을 제공하지만 성능 또는 애플리케이션 수준 지표를 제공하지는 않습니다.

기능	설명
	CodeDeploy에서 제공하는 모니터링 기능 유형에 대한 자세한 내용은 CodeDeploy에서 배포 모니터링을 참조하세요.

다음 다이어그램은 전체 CI/CD 솔루션의 일부로 CodeDeploy의 일반적인 사용 사례를 보여줍니다. 이 예제에서 CodeDeploy는 추가 AWS 개발자 도구, 즉 AWS CodePipeline (CI/CD 파이프라인 자동화), AWS CodeBuild (애플리케이션 구성 요소 빌드 및 테스트) 및 AWS CodeCommit (소스 코드 리포지토리)와 함께 사용하여 Amazon EC2 인스턴스 그룹에 애플리케이션을 배포합니다. CodeDeploy는 전체 CI/CD 파이프라인의 일부로 다른 도구와 함께 사용됩니다. CodeDeploy는 배포 그룹의 일부인 컴퓨팅 리소스에 대한 애플리케이션 구성 요소의 배포를 관리합니다. 모든 인프라 구성 요소는 CodeDeploy 외부에서 생성됩니다.



AWS CodeDeploy 사용 사례

AWS CodeDeploy 용 AWS Lambda

AWS CodeDeploy 용 AWS Lambda 를 사용하면 서버리스 배포를 자동화하여 애플리케이션 릴리스에 대한 제어 및 가시성을 높일 수 있습니다. CodeDeploy를 사용하여 서버리스 함수의 새 버전을 소수의 사용자 또는 트래픽에 배포하고 새 버전에 대한 신뢰도를 얻으면서 트래픽을 점진적으로 늘릴 수 있습니다. CodeDeploy를 사용하면 동일한 이벤트 소스에서 트래픽을 수신하는 Lambda 함수 세트를 나타내는 배포 그룹을 정의할 수 있습니다. 예를 들어 API Gateway 또는 Amazon EventBridge 규칙에 의해 시작되는 Lambda 함수 집합에 대한 배포 그룹을 생성할 수 있습니다. 그런 다음 erverless 함수의 새 버전을 지정된 배포 그룹에 배포하는 CodeDeploy를 사용하여 배포를 생성할 수 있습니다.

또한 CodeDeploy를 사용하면 배포 유형, 배포 전략 및 트래픽 이동 규칙과 같은 배포 설정을 지정하는 배포 구성을 정의할 수 있습니다. Canary 배포 전략을 사용하여 서버리스 함수의 새 버전을 적은 비율의 트래픽에 배포하고 새 버전에 대한 트래픽을 늘리기 전에 새 버전의 상태와 성능을 모니터링할 수 있습니다.

서버리스용 CodeDeploy를 사용하면 배포 프로세스를 자동화하고, 애플리케이션의 새 버전을 릴리스하는 데 필요한 시간과 노력을 줄이고, 서버리스 함수의 안정성과 신뢰성을 높일 수 있습니다.

Amazon Elastic Container Service

Amazon Elastic Container Service(Amazon ECS)는 Docker 컨테이너를 지원하고 관리형 클러스터에서 애플리케이션을 쉽게 실행할 수 있는 완전 관리형 컨테이너 오케스트레이션 서비스입니다. Amazon ECS는 컨테이너 관리 인프라를 설치, 운영 및 확장할 필요가 없으며 Security Groups, Elastic Load Balancing 및 AWS Identity and Access Management (IAM)와 같은 친숙한 AWS 핵심 기능을 갖춘 환경 생성을 간소화합니다.

Amazon ECS에서 애플리케이션을 실행할 때 Amazon EC2 인스턴스 또는 컨테이너용 서버리스 컴퓨팅 엔진AWS Fargate인를 사용하여 컨테이너에 기본 컴퓨팅 성능을 제공하도록 선택할 수 있습니다. 어느 경우든 Amazon ECS는 사용자가 정의한 구성에 따라 컨테이너를 클러스터에 자동으로 배치하고 확장합니다. Amazon ECS는 사용자를 대신하여 로드 밸런서 또는 IAM 역할과 같은 인프라 구성 요소를 생성하지 않지만 Amazon ECS 서비스는 Amazon ECS 클러스터에서 이러한 리소스의 생성 및 사용을 간소화하는 여러 APIs를 제공합니다.

Amazon ECS를 사용하면 개발자가 모든 인프라 구성 요소를 직접 세분화하여 사용자 지정 애플리케이션 아키텍처를 생성할 수 있습니다. 또한 Amazon ECS는 애플리케이션 컨테이너 이미지를 업데이트하기 위한 다양한 배포 전략을 지원합니다.

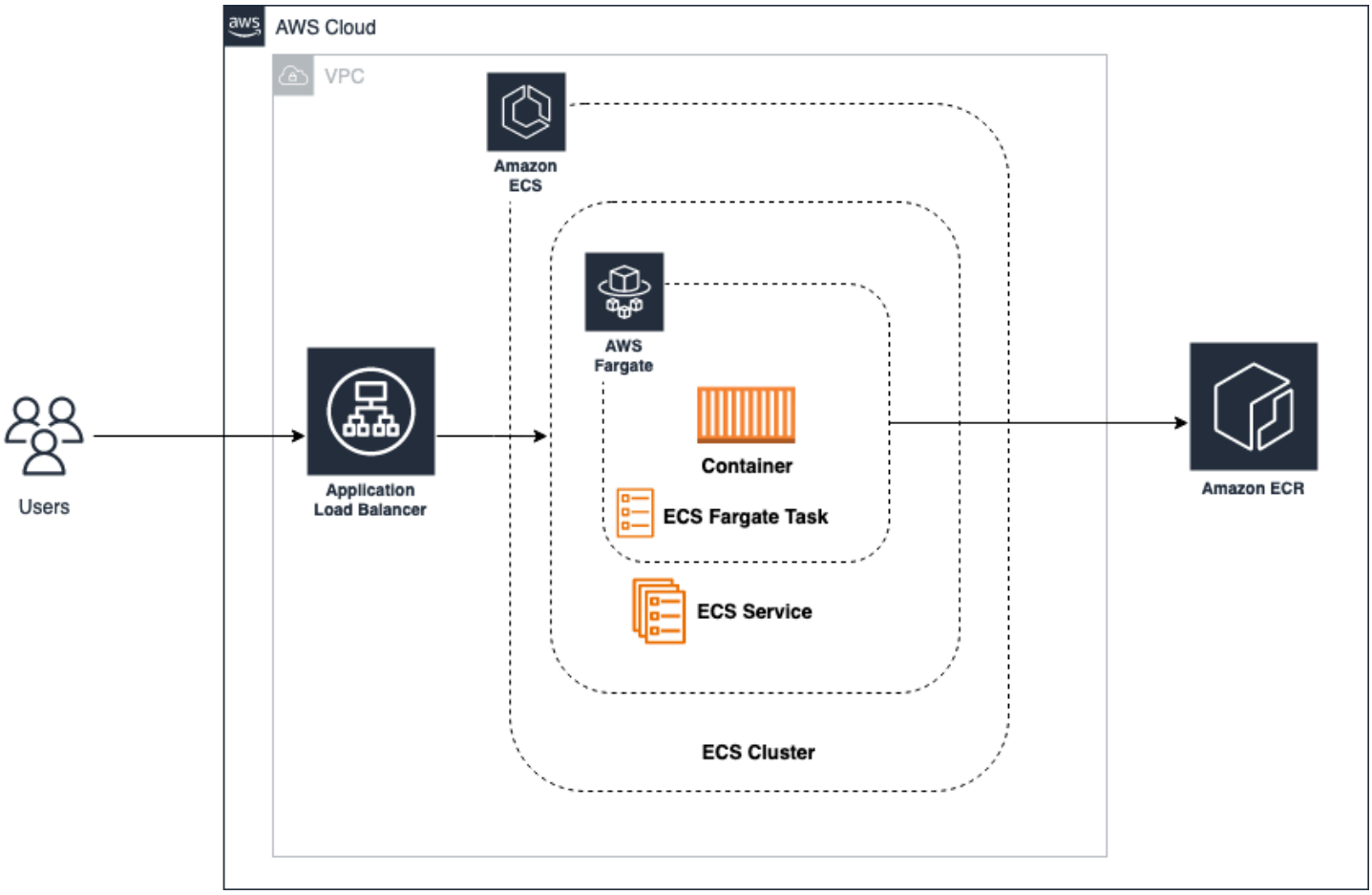
표 4: Amazon ECS 배포 기능

기능	설명
프로비저닝	Amazon ECS는 조정 정책 및 Amazon ECS 구성을 기반으로 새 애플리케이션 컨테이너 인스턴스와 컴퓨팅 리소스를 프로비저닝합니다. 로드 밸런서와 같은 인프라 리소스는 Amazon ECS 외부에서 생성해야 합니다.

기능	설명
	Amazon ECS로 생성할 수 있는 리소스 유형에 대한 자세한 내용은 Amazon ECS 시작하기를 참조하세요.
구성	Amazon ECS는 컨테이너화된 애플리케이션을 실행하기 위해 생성된 컴퓨팅 리소스의 사용자 지정과 애플리케이션 컨테이너의 런타임 조건(예: 환경 변수, 노출된 포트, 예약된 메모리/CPU)을 지원합니다. 기본 컴퓨팅 리소스의 사용자 지정은 Amazon EC2 인스턴스를 사용하는 경우에만 사용할 수 있습니다. 컨테이너화된 애플리케이션을 실행하도록 Amazon ECS 클러스터를 사용자 지정하는 방법에 대한 자세한 내용은 클러스터 생성을 참조하세요.
배포	Amazon ECS는 컨테이너화된 애플리케이션을 위한 여러 배포 전략을 지원합니다. 지원되는 배포 프로세스 유형에 대한 자세한 내용은 Amazon ECS 배포 유형을 참조하세요.
규모 조정	Amazon ECS를 Auto Scaling 정책과 함께 사용하여 Amazon ECS 클러스터에서 실행 중인 컨테이너 수를 자동으로 조정할 수 있습니다. Amazon ECS에서 컨테이너화된 애플리케이션의 Auto Auto Scaling 구성에 대한 자세한 내용은 Service Auto Scaling을 참조하세요.

기능	설명
모니터링	Amazon ECS는 CloudWatch를 사용한 컴퓨팅 리소스 및 애플리케이션 컨테이너 모니터링을 지원합니다. Amazon ECS에서 제공하는 모니터링 기능 유형에 대한 자세한 내용은 Amazon ECS 모니터링을 참조하세요.

다음 다이어그램은 간단한 컨테이너화된 애플리케이션을 관리하는 데 사용되는 Amazon ECS를 보여줍니다. 이 예에서 인프라 구성 요소는 Amazon ECS 외부에서 생성되며 Amazon ECS는 클러스터에서 애플리케이션 컨테이너의 배포 및 작업을 관리하는 데 사용됩니다.



Amazon ECS 사용 사례

> **Note**
>
> - 애플리케이션 인프라(Amazon Elastic Container Registry(Amazon ECR) 리포지토리, Amazon ECS 구성 및 로드 밸런서 포함)는 Amazon ECS 배포 외부에서 프로비저닝되고 관리됩니다.
> - Amazon ECS는 Amazon ECS 서비스 내에서 실행되는 애플리케이션 컨테이너의 배포를 Amazon ECR과 같은 컨테이너 레지스트리에서 가져온 작업으로 관리합니다.

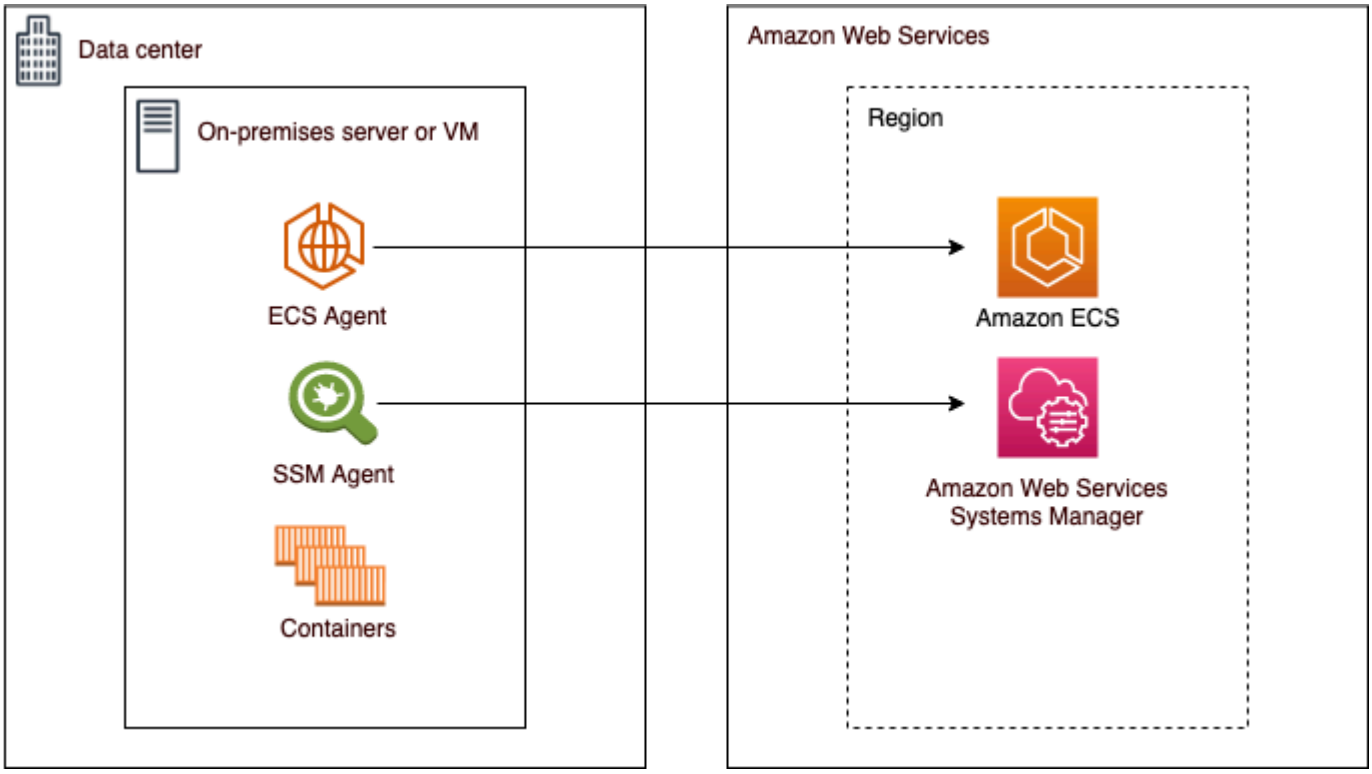
Amazon ECS는 Linux 및 Windows와 같은 여러 컨테이너 인스턴스 유형과 Amazon ECS Anywhere를 사용하는 온프레미스 가상 머신(VM)과 같은 외부 인스턴스 유형을 지원합니다.

Amazon ECS Anywhere

Amazon ECS Anywhere를 사용하면 온프레미스든 다른 클라우드 환경이든 관계없이 어디서나 Amazon ECS 작업을 실행할 수 있습니다. Amazon ECS Anywhere를 사용하면 일관된 운영 환경을 유지하면서 하이브리드 인프라 전체에 컨테이너화된 애플리케이션을 쉽게 배포하고 관리할 수 있습니다. 이 서비스는 Amazon ECS 플랫폼을 온프레미스 데이터 센터, 원격 사무실 및 기타 클라우드 환경을 포함한 모든 환경으로 확장하여 작동합니다. 이를 통해 기본 인프라에 대해 걱정할 필요 없이 동일한 친숙한 Amazon ECS APIs 및 도구를 사용하여 모든 환경에 컨테이너를 배포하고 관리할 수 있습니다.

Amazon ECS Anywhere는 Amazon ECS 에이전트를 사용하여 컨테이너의 배포 및 수명 주기를 관리하므로에서 사용하는 것과 동일한 Amazon ECS 작업 정의 및 구성 파일을 사용할 수 있습니다 AWS 클라우드. 이를 통해 하이브리드 인프라 전체에 컨테이너를 배포하고 관리하는 프로세스를 간소화하고 수동 구성 및 관리에 필요한 시간과 노력을 줄일 수 있습니다.

Amazon ECS Anywhere를 사용하면 IAM CloudFormation및 Amazon ECR과 같은 다른 AWS 서비스를 활용하여 컨테이너화된 애플리케이션을 관리할 수도 있습니다. 이를 통해 애플리케이션이 안전하고 규정을 준수하며 다른 AWS 서비스와 통합되도록 할 수 있습니다.



Amazon ECS Anywhere architecture

의 Amazon Elastic Container Service AWS Outposts

의 Amazon ECS AWS Outposts는에서 사용하는 것과 동일한 APIs 및 도구를 사용하여 온프레미스에서 Amazon ECS 작업을 실행할 수 있는 완전 관리형 AWS 서비스입니다 AWS 클라우드. 의 Amazon ECS를 사용하면 온프레미스에서 실행하든 클라우드에서 실행하든 컨테이너화된 애플리케이션을 일관되고 친숙한 방식으로 배포하고 관리할 AWS Outposts수 있습니다. AWS Outposts 는 AWS 인프라, 서비스, APIs 및 도구를 온프레미스 환경으로 확장하는 완전관리형 서비스입니다. Amazon ECS를 켜 AWS Outposts면 기본 인프라에 대해 걱정할 필요 없이 조직 전용 하드웨어에서 Amazon ECS 작업을 실행할 수 있습니다. 이를 통해 애플리케이션을 안전하고 규정을 준수하는 방식으로 배포하는 동시에 클라우드의 유연성과 확장성을 활용할 수 있습니다.

의 Amazon ECS AWS Outposts 는 온프레미스 환경에 AWS 서비스 및 APIs 세트를 배포하여 작동하므로 전용 하드웨어에서 Amazon ECS 작업을 실행할 수 있습니다. 여기에는 컨테이너의 배포 및 수명 주기를 관리하는 Amazon ECS 에이전트와 컨테이너화된 애플리케이션을 실행하기 위한 안전하고 규정을 준수하는 환경을 제공하는 AWS Outposts 인프라가 포함됩니다. Amazon ECS를 켜 AWS Outposts면에서 사용하는 것과 동일한 Amazon ECS APIs 및 도구를 사용하여 일관되고 친숙한 방식으로 컨테이너화된 애플리케이션을 AWS 클라우드쉽게 배포하고 관리할 수 있습니다. 이를 통해 수동 구성 및 관리에 필요한 시간과 노력을 줄이고 하이브리드 인프라 전반의 일관성과 신뢰성을 개선할 수 있습니다. AWS Outposts 또한의 Amazon ECS는 IAM CloudFormation및 Amazon ECR과 같은 다른

AWS 서비스와 통합되어 컨테이너화된 애플리케이션을 관리합니다. 이를 통해 애플리케이션이 안전하고 규정을 준수하며 다른 AWS 서비스와 통합되도록 할 수 있습니다.

Amazon Elastic Kubernetes Service:

Amazon Elastic Kubernetes Service(Amazon EKS)는 AWS에서 Kubernetes 클러스터를 구축, 보안, 운영 및 유지 관리하는 프로세스를 간소화하는 완전관리형 인증 Kubernetes 준수 서비스입니다. Amazon EKS는 CloudWatch, Auto Scaling 그룹 및 IAM과 같은 핵심 AWS 서비스와 통합되어 컨테이너화된 애플리케이션을 모니터링, 조정 및 로드 밸런싱할 수 있는 원활한 환경을 제공합니다.

Amazon EKS는 Kubernetes 워크로드를 위한 확장 가능하고 가용성이 높은 컨트롤 플레인을 제공합니다. Amazon ECS와 마찬가지로 Amazon EKS에서 애플리케이션을 실행할 때 Amazon EC2 인스턴스 또는를 사용하는 컨테이너에 기본 컴퓨팅 성능을 제공하도록 선택할 수 있습니다 AWS Fargate.

Amazon VPC Lattice는 여러 계정 및 Virtual Private Cloud(VPCs. Amazon EKS를 사용하면 Kubernetes Gateway API의 구현인 AWS Gateway API Controller를 사용하여 VPC Lattice를 활용할 수 있습니다. VPC Lattice를 사용하면 간단하고 일관된 방식으로 표준 Kubernetes 의미 체계를 사용하여 클러스터 간 연결을 설정할 수 있습니다.

다음 배포 옵션 중 하나와 함께 Amazon EKS를 사용할 수 있습니다.

- Amazon EKS Distro – Amazon EKS Distro는 Amazon EKS에 의해 클라우드에 배포된 동일한 오픈 소스 Kubernetes 소프트웨어 및 종속성을 배포한 것입니다. Amazon EKS 배포판은 Amazon EKS와 동일한 Kubernetes 버전 릴리스 주기를 따르며 오픈 소스 프로젝트로 제공됩니다. 자세한 내용은 Amazon EKS Distro를 참조하세요.
- Amazon EKS on AWS Outposts - 온프레미스 시설에서 네이티브 AWS 서비스, 인프라 및 운영 모델을 AWS Outposts 활성화합니다. Amazon EKS의 AWS Outposts경우 확장 또는 로컬 클러스터를 실행하도록 선택할 수 있습니다. 확장 클러스터의 경우 Kubernetes 컨트롤 플레인은에서 실행 AWS 리전 되고 노드는에서 실행됩니다 AWS Outposts. 로컬 클러스터를 사용하면 Kubernetes 컨트롤 플레인과 노드를 AWS Outposts포함하여 전체 Kubernetes 클러스터가 로컬에서 실행됩니다.
- Amazon EKS Anywhere – Amazon EKS Anywhere는 온프레미스에서 Kubernetes 클러스터를 쉽게 생성하고 운영할 수 있게 해주는 Amazon EKS를 위한 배포 옵션입니다. Amazon EKS 및 Amazon EKS Anywhere 모두 Amazon EKS Distro를 기반으로 구축됩니다. Amazon EKS Anywhere에 대해 자세히 알아보려면 Amazon Amazon EKS Anywhere, Amazon Amazon EKS Anywhere 개요 및 Amazon EKS Anywhere와 Amazon EKS 비교를 참조하세요.

Kubernetes 클러스터에 어떤 배포 옵션을 사용할지 선택할 때 다음 사항을 고려하세요.

표 5: Kubernetes 배포 기능

기능	Amazon EKS	의 Amazon EKS AWS Outposts	Amazon EKS Anywhere	Amazon EKS Distro
하드웨어	AWS에서 제공	AWS에서 제공	사용자 제공	사용자 제공
배포 위치	AWS 클라우드	사용자 데이터 센터	사용자 데이터 센터	사용자 데이터 센터
Kubernetes 제어 영역 위치	AWS 클라우드	AWS 클라우드 또는 데이터 센터	사용자 데이터 센터	사용자 데이터 센터
Kubernetes 데이터 영역 위치	AWS 클라우드	사용자 데이터 센터	사용자 데이터 센터	사용자 데이터 센터
Support	AWS 지원	AWS 지원	AWS 지원	OSS 커뮤니티 지원

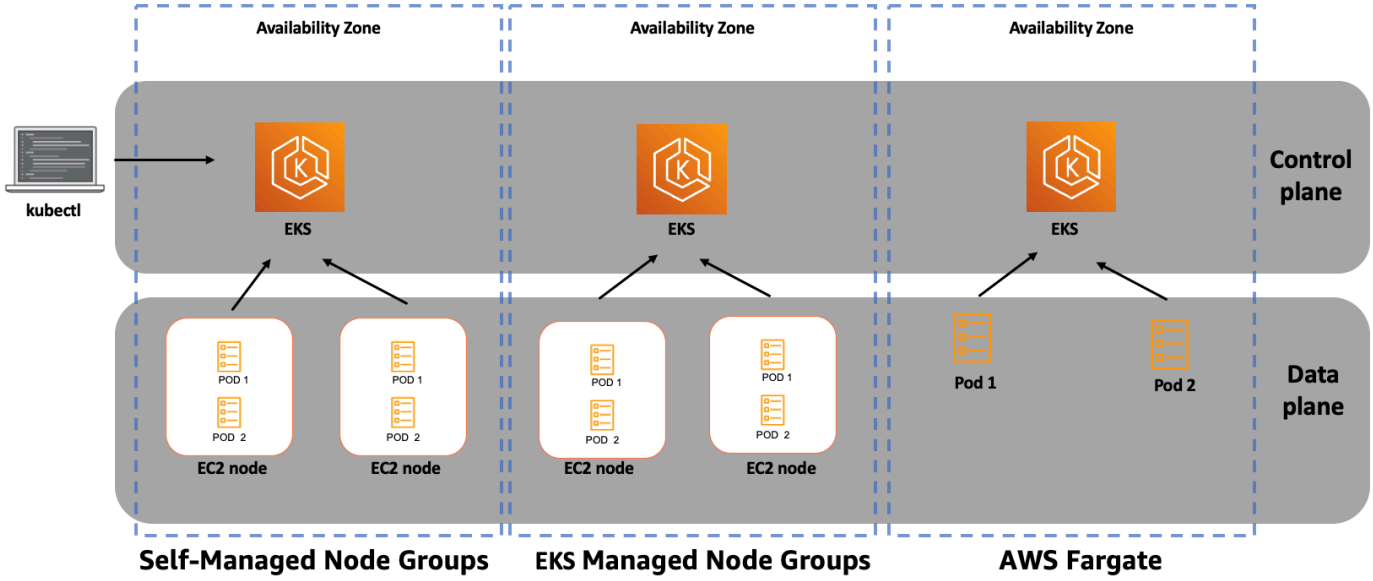
표 6: Amazon EKS 배포 기능

기능	설명
프로비저닝	Amazon EKS는 컨테이너화된 애플리케이션을 지원하기 위해 특정 리소스를 프로비저닝합니다. • 필요한 경우 로드 밸런서 • 컴퓨팅 리소스 또는 작업자(Amazon EKS는 Windows 및 Linux 지원) • 애플리케이션 컨테이너 인스턴스 또는 포드 Amazon EKS 클러스터 프로비저닝에 대한 자세한 내용은 Amazon EKS 시작하기를 참조하세요.

기능	설명
구성	Amazon EC2 인스턴스를 사용하여 컴퓨팅 성능을 제공하는 경우 Amazon EKS는 컴퓨팅 리소스(작업자)의 사용자 지정을 지원합니다. Amazon EKS는 애플리케이션 컨테이너(포드)의 런타임 조건 사용자 지정도 지원합니다. 자세한 내용은 작업자 노드 및 Fargate 포드 구성 설명서를 참조하세요.
배포	Amazon EKS는 Kubernetes와 동일한 배포 전략을 지원합니다. 자세한 내용은 Kubernetes 배포 사양 작성 -> 전략을 참조하세요.
규모 조정	Amazon EKS는 Kubernetes Cluster Autoscaler를 사용하여 작업자를 확장하고 Kubernetes Horizontal Pod Autoscaler 및 Kubernetes Vertical Pod Autoscaler를 사용하여 포드를 확장합니다. 또한 Amazon EKS는 오픈 소스의 유연한 고성능 Kubernetes 클러스터 오토스케일러인 Karpenter를 지원하여 애플리케이션 로드 변화에 대응하여 적절한 크기의 컴퓨팅 리소스를 빠르게 시작하여 애플리케이션 가용성과 클러스터 효율성을 개선하는 데 도움이 됩니다.
모니터링	Amazon EKS 컨트롤 플레인 로그는 감사 및 진단 정보를 CloudWatch Logs에 직접 제공합니다. 또한 Amazon EKS 컨트롤 플레인은와 통합되어 Amazon EKS에서 수행된 작업을 AWS CloudTrail 기록합니다. 자세한 내용은 Amazon EKS 로깅 및 모니터링을 참조하세요.

Amazon EKS를 사용하면 조직이 오픈 소스 Kubernetes 도구 및 플러그인을 활용할 수 있으며 기존 Kubernetes 환경을 사용하여 AWS로 마이그레이션하는 조직에 적합한 선택이 될 수 있습니다. 다음 다

이어그램은 일반적인 컨테이너화된 애플리케이션을 관리하는 데 사용되는 Amazon EKS를 보여줍니다.



Amazon EKS use case

Amazon EKS Anywhere

Amazon EKS Anywhere를 사용하면 자체 인프라에서 Kubernetes 클러스터를 생성하고 운영할 수 있습니다. Amazon EKS Anywhere는 Amazon EKS Distro의 강점을 기반으로 하며 최신 상태이고 패치가 적용된 오픈 소스 소프트웨어를 제공하므로 자체 관리형 Kubernetes 제품보다 안정적인 온프레미스 Kubernetes 환경을 구축할 수 있습니다.

Amazon EKS Anywhere는 선택한 공급자에 온프레미스로 Kubernetes 클러스터를 생성합니다. 지원되는 공급자에는 베어 메탈(Tinkerbell을 통해), CloudStack 및 vSphere가 포함됩니다. 해당 클러스터를 관리하기 위해 Ubuntu 또는 Mac Administrative 시스템에서 클러스터 생성 및 삭제 명령을 실행할 수 있습니다.

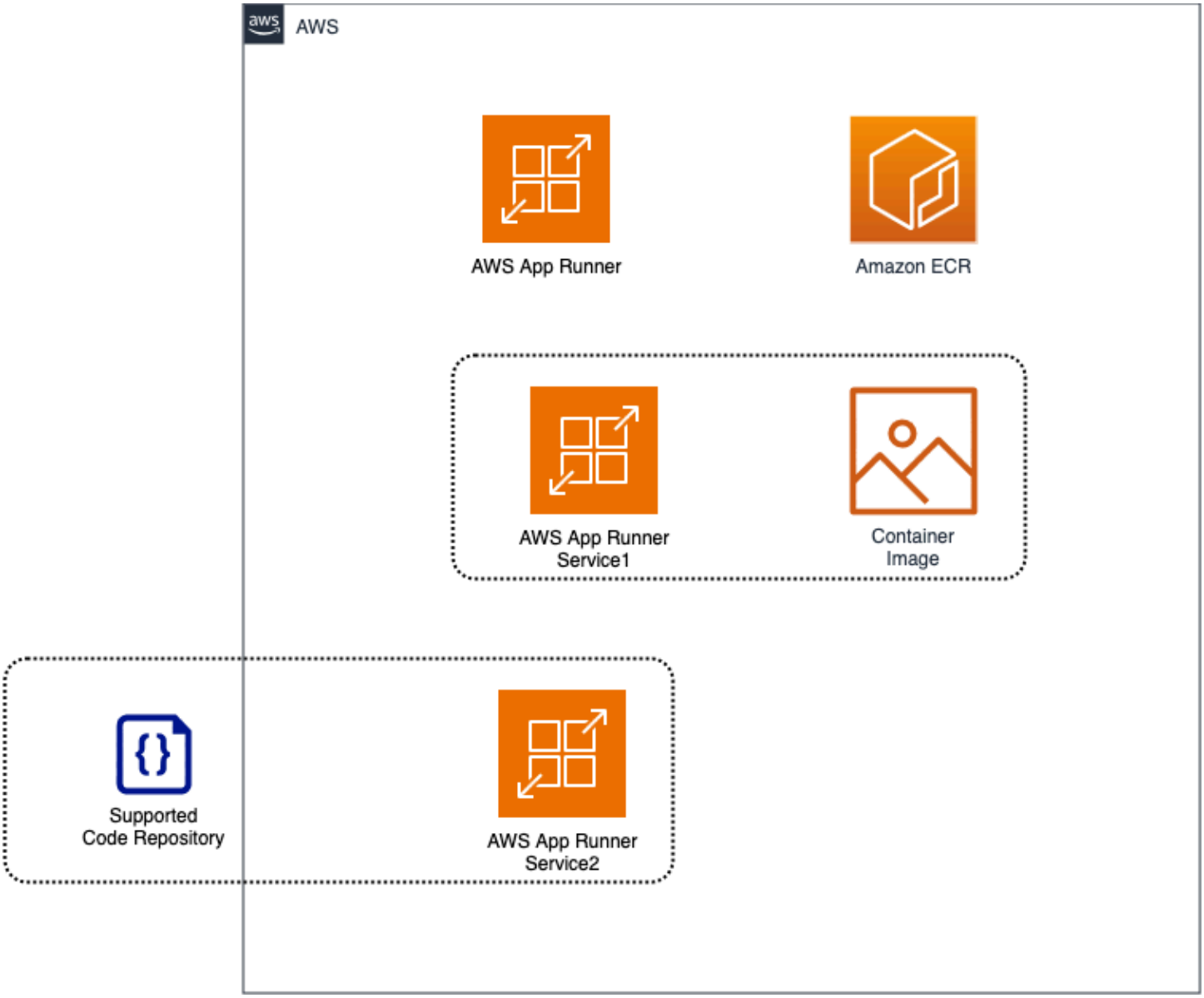
AWS App Runner

AWS App Runner는 사전 인프라 또는 컨테이너 경험 없이 컨테이너화된 웹 애플리케이션 및 API 서비스를 빌드, 배포 및 실행할 수 있는 완전 관리형 컨테이너 애플리케이션 서비스입니다. App Runner는 코드 또는 이미지 리포지토리에 직접 연결됩니다. 완전 관리형 운영, 고성능, 확장성 및 보안을 갖춘 자동 통합 및 전송 파이프라인을 제공합니다.

App Runner는 리포지토리에서 소스 코드 또는 소스 이미지를 가져온 다음에서 실행 중인 웹 서비스를 생성하고 유지 관리합니다 AWS 클라우드. 일반적으로 서비스를 생성하려면 App Runner 작업을 하나만 호출`CreateService`해야 합니다. 소스 이미지 리포지토리를 사용하면 App Runner가 웹 서비스를 실행하기 위해 배포할 수 있는 ready-to-use 가능한 컨테이너 이미지를 제공할 수 있습니다. 소스 코드 리포지토리를 사용하면 웹 서비스를 구축하고 실행하기 위한 코드와 지침을 제공하고 특정 런타임 환경을 대상으로 지정할 수 있습니다. App Runner는 플랫폼 메이저 버전에 대해 각각 하나 이상의 관리형 런타임이 있는 여러 프로그래밍 플랫폼을 지원합니다. App Runner는 컨테이너 이미지와 Node.js 및 Python을 포함한 런타임 및 웹 프레임워크를 지원합니다. App Runner는 애플리케이션으로 전송된 동시 요청 수를 모니터링하고 요청 볼륨을 기반으로 인스턴스를 자동으로 추가합니다. 애플리케이션이 수신 요청을 수신하지 못하면 App Runner는 컨테이너를 밀리초 이내에 수신 요청을 처리할 수 있는 CPU 제한 인스턴스인 프로비저닝된 인스턴스로 축소합니다.

현재 App Runner는 GitHub 리포지토리에서 소스 코드를 검색하거나의 Amazon ECR에서 소스 이미지를 검색할 수 있습니다 AWS 계정.

다음 다이어그램은 App Runner 서비스 아키텍처의 개요를 보여줍니다. 다이어그램에는 두 가지 예제 서비스가 있습니다. 하나는 GitHub에서 소스 코드를 배포하고 다른 하나는 Amazon ECR에서 소스 이미지를 배포합니다.



App Runner use case

App Runner는 HTTP 및 HTTPS 프로토콜을 사용하는 프런트엔드 및 백엔드 웹 애플리케이션을 포함하여 전체 스택 개발을 지원합니다. 이러한 애플리케이션에는 API 서비스, 백엔드 웹 서비스 및 웹 사이트가 포함됩니다. App Runner는 컨테이너 이미지와 Node.js 및 Python을 포함한 런타임 및 웹 프레임워크를 지원합니다.

Amazon Lightsail

Amazon Lightsail는 소기업, 스타트업 및 개인이 클라우드에서 애플리케이션을 쉽게 배포하고 관리할 수 있는 간단하고 비용 효율적인 클라우드 서비스입니다. 기본 인프라 관리의 대부분을 추상화하고 클라우드에서 애플리케이션을 쉽게 시작하고 실행할 수 있는 사용자 친화적 인터페이스를 제공합니다.

를 사용하면 Virtual Private Server(VPS), 데이터베이스 및 스토리지 인스턴스Lightsail를 빠르게 배포하고 관리할 수 있습니다. 이 서비스는 WordPress, Drupal, Joomla 등 다양한 워크로드에 최적화된 사전 구성된 인스턴스를 제공합니다. 이를 통해 환경을 설정하고 구성하는 데 필요한 시간과 노력을 줄일 수 있습니다. Lightsail 또한는 통합 로드 밸런서와 자동 조정을 제공하므로 수동 개입 없이 트래픽 수요의 변화를 처리할 수 있습니다. 또한이 서비스는 모니터링 및 알림을 제공하므로 애플리케이션의 상태와 성능을 파악할 수 있습니다.

의 주요 이점 중 하나는 단순성과 사용 편의성Lightsail입니다. 이 서비스는 클라우드 컴퓨팅 경험을 최소화한 사용자가 액세스할 수 있도록 설계되었으므로 클라우드에서 빠르게 시작하려는 소규모 기업 또는 개인에게 적합합니다. 또한 Lightsail는 컴퓨팅, 스토리지 및 데이터 전송을 포함하는 예측 가능한 요금으로 비용 효율적입니다.

Amazon Lightsail 컨테이너

Amazon Lightsail Containers는 클라우드에서 컨테이너화된 애플리케이션을 쉽게 배포하고 관리할 수 있는 완전 관리형 컨테이너 서비스 AWS입니다. Docker 및 Kubernetes와 같은 인기 있는 컨테이너 관리 도구를 사용하여 컨테이너를 시작하고 실행하는 간단하고 비용 효율적인 방법을 제공합니다.

Lightsail 컨테이너는 컨테이너화된 애플리케이션을 구축, 테스트 및 배포하기 위한 통합 환경을 제공합니다. 기본 인프라 관리의 대부분을 추상화하는 사용자 친화적 인터페이스를 제공하여 컨테이너 배포 및 관리 프로세스를 간소화합니다.

Lightsail 컨테이너를 사용하면 몇 번의 클릭만으로 컨테이너화된 애플리케이션을 VPC에 배포할 수 있습니다. 이 서비스는 Node.js, Python, Ruby, Java 등 널리 사용되는 프로그래밍 언어에 맞게 사전 구성된 컨테이너 이미지를 제공합니다. 이렇게 하면 컨테이너 환경을 설정하고 구성하는 데 필요한 시간과 노력을 줄일 수 있습니다.

Lightsail 또한 컨테이너는 컨테이너 인스턴스 간에 트래픽을 자동으로 분산하여 애플리케이션 가용성과 확장성을 개선할 수 있는 통합 로드 밸런서를 제공합니다. 또한이 서비스는 컨테이너 인스턴스의 자동 조정을 제공하므로 수동 개입 없이 트래픽 수요의 변화를 처리할 수 있습니다.

Lightsail 컨테이너를 사용하면 기본 제공 지표 및 로그를 사용하여 컨테이너화된 애플리케이션의 성능을 모니터링할 수 있습니다. 또한를 Amazon S3, Amazon RDS 및 같은 다른 AWS 서비스와 통합하여 컨테이너화된 애플리케이션을 위한 완전 자동화된 통합 CI/CD 파이프라인 AWS CodePipeline을 생성할 수 있습니다.

Red Hat OpenShift Service on AWS

Red Hat OpenShift Service on AWS (ROSA)는 AWS Management Console을 통해 사용할 수 있는 관리형 서비스입니다. ROSA를 사용하면 Red Hat OpenShift 사용자로서 AWS에서 컨테이너화된 애플

리케이션을 구축, 확장 및 관리할 수 있습니다. ROSA를 사용하여 Red Hat OpenShift APIs 및 도구를 사용하여 Kubernetes 클러스터를 생성하고 AWS 서비스의 전체 폭과 깊이에 액세스할 수 있습니다. ROSA는 온프레미스 Red Hat OpenShift 워크로드를 AWS로 이동하는 작업을 간소화하고 다른 AWS 서비스와 긴밀하게 통합합니다. AWS를 통해 Red Hat OpenShift 라이선스, 결제 및 지원에 직접 액세스할 수도 있습니다.

각 ROSA 클러스터에는 완전 관리형 컨트롤 플레인 및 컴퓨팅 노드가 함께 제공됩니다. 설치, 관리, 유지 관리 및 업그레이드는 Red Hat SRE에서 공동 Red Hat 및 Amazon 지원을 통해 수행됩니다. 클러스터 서비스(예: 로깅, 지표 및 모니터링)도 사용할 수 있습니다. ROSA에서는 Red Hat Enterprise Linux CoreOS(R™S) 작업자만 지원합니다.

ROSA는 다양한 AWS 컴퓨팅, 스토리지, 데이터베이스, 분석, 기계 학습, 네트워킹, 모바일 및 다양한 애플리케이션 서비스와 통합되므로 고객은 전 세계에서 온디맨드 방식으로 확장되는 강력한 AWS 서비스 포트폴리오를 활용할 수 있습니다. 이러한 AWS 네이티브 서비스에 직접 액세스하여 동일한 관리 인터페이스를 통해 서비스를 빠르게 배포하고 확장할 수 있습니다.

AWS 로컬 영역

AWS Local Zone은 사용자 AWS 리전 와 지리적으로 가까운의 확장입니다. 로컬 영역은 인터넷에 대한 자체 연결을 가지고 있으며 AWS Direct Connect를 지원합니다. Local Zones에서 생성된 리소스는 대기 시간이 매우 짧은 통신으로 로컬 사용자에게 제공될 수 있습니다. 로컬 영역은 리전 코드와 위치를 나타내는 식별자(예: us-west-2-lax-1a)로 표시됩니다.

Amazon ECS는 짧은 지연 시간 또는 로컬 데이터 처리가 필요한 경우 로컬 영역을 사용하는 워크로드를 지원합니다. Amazon ECS 컨트롤 플레인은 항상에서 실행됩니다 AWS 리전.

Amazon EKS는 Local Zones의 특정 리소스를 지원합니다. 여기에는 자체 관리형 Amazon EC2 노드, Amazon EBS 볼륨 및 Application Load Balancer가 포함됩니다. Amazon EKS 관리형 Kubernetes 제어 플레인은 항상 AWS 리전에서 실행됩니다. Amazon EKS 관리형 Kubernetes 제어 영역은 로컬 영역에서 실행될 수 없습니다. Local Zones는 VPC 내에 서브넷으로 표시되므로 Kubernetes는 로컬 영역 리소스를 해당 서브넷의 일부로 인식합니다.

AWS Wavelength

AWS Wavelength는 5G-connected 사용자 및 디바이스에 더 가깝게 워크로드를 배포할 수 있는 AWS 인프라입니다. Wavelength를 사용하여 AWS Marketplace에서 사용할 수 있는 Amazon EC2 인스턴스, Amazon EKS 클러스터 및 지원되는 파트너 솔루션 제품군을 배포할 수 있습니다. Wavelength Zone은

중복되고 지연 시간이 짧으며 처리량이 많은 연결을 통해 AWS 리전에 다시 연결되는 통신 공급자의 네트워크 내에서 논리적으로 격리된 데이터 센터입니다.

Wavelength의 주요 기능에는 Wavelength Zone에서 Amazon EC2 인스턴스, Amazon EBS 볼륨, Amazon VPC 서브넷 및 통신 사업자 게이트웨이를 생성하는 기능이 포함됩니다. Amazon EC2 Auto Scaling, Amazon EKS 클러스터, Amazon ECS 클러스터, Amazon EC2 Systems Manager, Amazon CloudWatch 및 Application Load Balancer와 같이 Amazon EC2 AWS CloudTrail AWS CloudFormation, Amazon EBS 및 Amazon VPC를 오케스트레이션하거나 사용하는 서비스를 사용할 수도 있습니다. Application Load Balancer Wavelength 서비스는 Amazon DynamoDB 및 Amazon Relational Database Service(RDS)를 포함한 서비스에 쉽게 액세스할 수 있도록 AWS 리전에 대한 안정적인 고대역폭 연결을 통해 연결된 VPC의 일부입니다.

추가 배포 서비스

Amazon Simple Storage Service(Amazon S3)는 정적 콘텐츠 및 단일 페이지 애플리케이션(SPA)의 웹 서버로 사용할 수 있습니다. 정적 콘텐츠 전송의 성능을 높이기 위해 Amazon CloudFront와 결합된 Amazon S3를 사용하면 정적 콘텐츠를 배포하고 업데이트하는 간단하고 강력한 방법이 될 수 있습니다. 이 접근 방식에 대한 자세한 내용은 백서의 정적 웹 사이트 호스팅에서 AWS 확인할 수 있습니다.

AWS Proton

AWS Proton은 마이크로서비스 및 컨테이너 기반 애플리케이션을 배포하고 관리하는 프로세스를 간소화하고 자동화하는 완전관리형 서비스입니다. 널리 사용되는 DevOps 도구 및 서비스와 통합되는 통합되고 일관된 배포 환경을 제공하므로 애플리케이션 개발을 더 쉽게 관리하고 간소화할 수 있습니다. Proton을 사용하면 개발자가 인프라, 코드 및 파이프라인과 같은 애플리케이션 구성 요소를 재사용 가능한 템플릿으로 정의하고 생성할 수 있습니다. 이러한 템플릿은 개발, 테스트 및 프로덕션과 같은 여러 환경을 생성하는 데 사용할 수 있으며 팀 또는 조직 간에 공유할 수 있습니다. 이 접근 방식은 시간이 많이 걸리고 오류가 발생하기 쉬운 마이크로서비스 및 컨테이너 기반 애플리케이션을 배포하고 관리하는 복잡성을 줄이는 데 도움이 됩니다.

AWS Proton은 특정 요구 사항에 맞게 사용자 지정할 수 있는 웹 애플리케이션, APIs 및 데이터베이스와 같은 일반적인 유형의 마이크로서비스를 위한 사전 구축된 템플릿을 제공합니다. 또한 AWS CodePipeline, AWS CodeCommit 및 AWS CodeBuild와 같은 널리 사용되는 DevOps 도구와 통합되어 지속적 통합 및 배포(CI/CD) 워크플로를 지원합니다.

개발자는 AWS Proton을 사용하여 마이크로서비스 및 컨테이너 기반 애플리케이션을 배포하고 관리하는 데 필요한 시간과 노력을 줄일 수 있습니다. 이 접근 방식을 통해 팀은 배포 및 관리 프로세스에 시간을 소비하는 대신 애플리케이션을 개발하고 개선하는 데 집중할 수 있습니다.

AWS App2Container

AWS App2Container는 Java 및 .NET 웹 애플리케이션을 컨테이너 형식으로 마이그레이션하고 현대화하기 위한 명령줄 도구입니다. App2Container는 베어 메탈, 가상 머신, Amazon EC2 인스턴스 또는 클라우드에서 실행되는 애플리케이션의 인벤토리를 분석하고 빌드합니다. 컨테이너화하려는 애플리케이션을 선택하면 App2Container가 애플리케이션 아티팩트와 식별된 종속성을 컨테이너 이미지로 패키징하고, 네트워크 포트를 구성하고, ECS 작업 및 Kubernetes 포드 정의를 생성합니다. App2Container는 가상 머신에서 실행되는 지원되는 ASP.NET 및 Java 애플리케이션을 식별하여 환경의 모든 애플리케이션에 대한 포괄적인 인벤토리를 구축합니다. App2Container는 Linux, 독립형 또는 JBoss, Apache Tomcat, Springboot, IBM Websphere 및 Oracle Weblogic과 같은 애플리케이션 서버에서 실행되는 Windows 또는 Java 애플리케이션의 IIS에서 실행되는 ASP.NET 웹 애플리케이션을 컨테이너화할 수 있습니다.

AWS Copilot

AWS Copilot은 AWS에서 컨테이너화된 애플리케이션을 빠르게 시작하고 관리하는 데 사용할 수 있는 명령줄 인터페이스(CLI)입니다. Amazon ECS, Fargate 및 App Runner에서 애플리케이션 실행을 간소화합니다. AWS Copilot은 현재 Linux, macOS 및 Windows 시스템을 지원합니다. Copilot을 사용하면 로드 밸런싱된 웹 서비스와 같은 서비스 패턴을 사용하여 인프라를 프로비저닝하고, 테스트 또는 프로덕션과 같은 여러 환경에 배포하며, 자동 배포에 AWS CodePipeline 릴리스 파이프라인을 사용할 수도 있습니다.

AWS Serverless Application Model

AWS Serverless Application Model (AWS SAM)는 서버리스 애플리케이션을 빌드하기 위한 오픈 소스 프레임워크입니다. 함수, API, 데이터베이스 및 이벤트 소스 매핑을 표현하는 속기 구문을 제공합니다. 리소스당 몇 줄만 있으면 원하는 애플리케이션을 정의하고 YAML을 사용하여 모델링할 수 있습니다. 배포 중에 SAM은 SAM 구문을 AWS CloudFormation 구문으로 변환하고 확장하므로 서버리스 애플리케이션을 더 빠르게 구축할 수 있습니다.

AWS SAM CLI는 AWS에서 서버리스 애플리케이션을 쉽게 개발, 테스트 및 배포할 수 있는 오픈 소스 명령줄 도구입니다. AWS CloudFormation의 확장인 AWS SAM 사양을 사용하여 서버리스 애플리케이션을 빌드하기 위한 명령줄 인터페이스입니다.

AWS SAM CLI를 사용하면 개발자가 AWS에 배포하기 전에 서버리스 애플리케이션을 로컬에서 정의하고 테스트할 수 있습니다. AWS Lambda 및 API Gateway를 시뮬레이션하는 로컬 테스트 환경을 제공하므로 개발자는 코드와 구성을 클라우드에 배포하기 전에 테스트할 수 있습니다.

AWS SAM CLI에는 자동 코드 배포, 로깅 및 디버깅 기능과 같은 다양한 유용한 기능도 포함되어 있습니다. 이를 통해 개발자는 단일 명령으로 애플리케이션을 빌드, 패키징 및 배포할 수 있으므로 서버리스 애플리케이션을 배포하고 관리하는 데 필요한 시간과 노력을 줄일 수 있습니다.

또한 AWS SAM CLI는 Node.js, Python, Java 및 .NET Core를 비롯한 다양한 프로그래밍 언어를 지원합니다. 이를 통해 개발자는 선호하는 프로그래밍 언어와 도구를 사용하여 서버리스 애플리케이션을 구축하고 배포할 수 있습니다.

AWS SAM CLI는 AWS CodePipeline 및 AWS CodeBuild와 같은 다른 AWS 서비스와 통합되어 서버리스 애플리케이션을 위한 완전 자동화된 통합 CI/CD 파이프라인을 제공합니다. 또한 개발자는 서버리스 애플리케이션의 일부로 Amazon S3, Amazon DynamoDB 및 Amazon SNS와 같은 다른 AWS 서비스를 사용할 수 있습니다.

AWS Cloud Development Kit (AWS CDK)

AWS Cloud Development Kit (AWS CDK) (AWS CDK)는 클라우드 인프라를 최신 프로그래밍 언어로 코드로 정의하고 AWS CloudFormation을 통해 배포하기 위한 오픈 소스 소프트웨어 개발 프레임워크입니다. AWS Cloud Development Kit(AWS CDK)는 애플리케이션을 모델링하기 위해 공통 프로그래밍 언어를 사용하여 클라우드 개발을 가속화합니다. AWS CDK를 사용하면 프로그래밍 언어의 상당한 표현력으로 클라우드에서 안정적이고 확장 가능하며 비용 효율적인 애플리케이션을 구축할 수 있습니다.

AWS CDK를 최신 프로그래밍 언어의 모든 기능을 활용하여 AWS 인프라를 코드로 정의하는 개발자 중심 툴킷이라고 생각하십시오. AWS CDK 애플리케이션이 실행되면 완전히 구성된 CloudFormation JSON/YAML 템플릿으로 컴파일된 다음 프로비저닝을 위해 CloudFormation 서비스에 제출됩니다. AWS CDK는 CloudFormation을 활용하므로 안전한 배포, 자동 롤백 및 드리프트 감지와 같은 CloudFormation의 모든 이점을 계속 누릴 수 있습니다.

이 접근 방식은 다음과 같은 많은 이점을 제공합니다.

- AWS 리소스에 합리적이고 안전한 기본값을 자동으로 제공하는 상위 수준 구성으로 빌드하여 더 적은 코드로 더 많은 인프라를 정의합니다.
- 파라미터, 조건부, 루프, 구성 및 상속과 같은 프로그래밍 관용어를 사용하여 AWS 등이 제공하는 빌딩 블록에서 시스템 설계를 모델링합니다.
- 인프라, 애플리케이션 코드 및 구성을 모두 한 곳에 배치하여 모든 마일스톤에서 클라우드 배포가 가능한 완전한 시스템을 확보할 수 있습니다.
- 코드 검토, 단위 테스트 및 소스 제어와 같은 소프트웨어 엔지니어링 사례를 사용하여 인프라를 더욱 견고하게 만듭니다.

- AWS Solutions Constructs는 AWS CDK의 오픈 소스 라이브러리 확장입니다. AWS Solutions Constructs는 AWS Well-Architected Framework에서 설정한 모범 사례를 사용하여 구축된 검증된 다중 서비스 아키텍처 패턴 모음을 제공합니다.

AWS Serverless Application Model과 AWS CDK는 모두 AWS 인프라를 코드로 추상화하므로 클라우드 인프라를 더 쉽게 정의할 수 있습니다. AWS SAM은 서버리스 사용 사례 및 아키텍처에 특별히 중점을 두고 있으며, 이를 통해 인프라를 작고 선언적인 JSON/YAML 템플릿으로 정의할 수 있습니다. AWS CDK는 모든 AWS 서비스에서 광범위한 범위를 제공하며 최신 프로그래밍 언어로 클라우드 인프라를 정의할 수 있습니다.

Amazon EC2 Image Builder

EC2 Image Builder는 AWS 또는 온프레미스에서 사용할 VM 및 컨테이너 이미지의 구축, 테스트 및 배포를 간소화합니다. VM 및 컨테이너 이미지를 up-to-date 유지하려면 시간이 많이 걸리고 리소스 집약적이며 오류가 발생하기 쉽습니다. 현재 고객은 VM은 수동으로 업데이트하고 스냅샷을 생성하거나 이미지를 유지하기 위해 자동화 스크립트를 구축하는 팀을 두고 있습니다. Image Builder는 간단한 그래픽 인터페이스, 내장 자동화 및 AWS 제공 보안 설정을 제공하여 이미지를 up-to-date 안전하게 유지하는 노력을 크게 줄입니다. Image Builder를 사용하면 이미지를 업데이트하는 수동 단계가 없으며 자체 자동화 파이프라인을 구축할 필요가 없습니다. Image Builder는 이미지를 생성, 저장 및 공유하는 데 사용되는 기본 AWS 리소스 비용을 제외하고 무료로 제공됩니다.

EC2 Image Builder는 Amazon EC2, 컨테이너 및 온프레미스 서버에 사용할 사용자 지정 이미지를 생성하고 관리하는 프로세스를 간소화하여 AWS에서 배포를 더 쉽게 만들 수 있습니다. 이 서비스는 이미지 생성 및 관리 프로세스를 간소화할 수 있는 자동화된 빌드 파이프라인을 통해 사용자 지정 이미지를 생성하고 관리하는 간단하고 유연한 방법을 제공합니다.

EC2 Image Builder는 대부분의 기본 인프라 관리를 추상화하는 사용자 친화적 인터페이스를 제공하므로 개발자가 사용자 지정 이미지를 더 쉽게 생성하고 관리할 수 있습니다. EC2 Image Builder를 사용하면 개발자가 이미지에 포함할 운영 체제, 애플리케이션 및 패키지를 지정할 수 있으며, 서비스는 업데이트, 패치 및 보안 수정을 포함하여 이미지를 구축하고 테스트하는 프로세스를 자동화합니다. 자동화된 빌드 파이프라인을 통해 개발자는 이미지 생성 및 관리 프로세스를 간소화하여 수동 이미지 생성 및 테스트에 필요한 시간과 노력을 줄일 수 있습니다. 이를 통해 일관성을 개선하고 오류를 줄이며 이미지가 up-to-date이고 안전하며 규정을 준수하는지 확인할 수 있습니다.

다음은 EC2 Image Builder의 몇 가지 이점입니다.

- 간소화된 이미지 생성: EC2 Image Builder는 Amazon EC2, 컨테이너 및 온프레미스 서버에 사용할 사용자 지정 이미지를 생성하는 간단하고 유연한 방법을 제공합니다. 이를 통해 사용자 지정 이미지

를 생성하고 유지 관리하는 데 필요한 시간과 노력을 줄이고 애플리케이션 개발 및 테스트와 같은 배포의 다른 측면에 집중할 수 있습니다.

- 자동화된 이미지 빌드 파이프라인: EC2 Image Builder는 사용자 지정 이미지를 빌드, 테스트 및 배포하기 위한 자동화된 파이프라인을 제공하므로 이미지 생성 및 관리 프로세스를 간소화하는 데 도움이 될 수 있습니다. 이렇게 하면 이미지가 up-to-date 상태이고 안전하며 규정을 준수하는지 확인하고 수동 이미지 생성 및 테스트에 필요한 시간과 노력을 줄일 수 있습니다.
- AWS 서비스와의 통합: EC2 Image Builder는 Amazon Elastic Container Registry(ECR) 및 Amazon Elastic Kubernetes Service(EKS)와 같은 다른 AWS 서비스와 통합되어 컨테이너에 사용할 사용자 지정 이미지를 구축할 수 있습니다. 이렇게 하면 컨테이너 빌드 및 배포 프로세스를 간소화하여 애플리케이션, 라이브러리 및 구성을 포함하는 사용자 지정 이미지를 빌드할 수 있습니다.
- 유연한 이미지 생성: EC2 Image Builder는 사용자 지정 이미지를 생성하는 유연한 방법을 제공하므로 이미지에 포함할 운영 체제, 애플리케이션 및 패키지를 지정할 수 있습니다. 이를 통해 이미지를 특정 사용 사례 및 요구 사항에 맞게 조정하고 배포 중에 오류 또는 비호환성의 위험을 줄일 수 있습니다.
- 이미지 보안 및 규정 준수 개선: EC2 Image Builder를 사용하면 취약성 및 규정 준수 스캔을 포함한 이미지 테스트를 자동화하여 이미지가 안전하고 규정을 준수하는지 확인할 수 있습니다. 이를 통해 보안 침해 위험을 줄이고 규정 준수를 개선하고 애플리케이션을 자신 있게 배포할 수 있습니다.

배포 전략

애플리케이션 코드를 업데이트하는 데 적합한 도구를 선택하고 인프라를 지원하는 것 외에도 올바른 배포 프로세스를 구현하는 것은 완전하고 잘 작동하는 배포 솔루션의 중요한 부분입니다. 애플리케이션을 업데이트하도록 선택한 배포 프로세스는 원하는 제어 균형, 속도, 비용, 위험 허용 범위 및 기타 요인에 따라 달라질 수 있습니다.

각 AWS 배포 서비스는 다양한 배포 전략을 지원합니다. 이 섹션에서는 배포 솔루션과 함께 사용할 수 있는 범용 배포 전략에 대한 개요를 제공합니다.

프리베이킹 및 부트스트래핑 AMIs

애플리케이션이 Amazon EC2 인스턴스에 애플리케이션을 사용자 지정하거나 배포하는 데 크게 의존하는 경우 부트스트래핑 및 프리베이킹 사례를 통해 배포를 최적화할 수 있습니다.

Amazon EC2 인스턴스가 시작될 때마다 애플리케이션, 종속성 또는 사용자 지정을 설치하는 것을 인스턴스 부트스트래핑이라고 합니다. 복잡한 애플리케이션이나 대규모 다운로드가 필요한 경우 배포 및 조정 이벤트 속도가 느려질 수 있습니다.

Amazon Machine Image(AMI)는 인스턴스를 시작하는 데 필요한 정보(운영 체제, 스토리지 볼륨, 권한, 소프트웨어 패키지 등)를 제공합니다. 단일 AMI에서 동일한 인스턴스를 여러 개 시작할 수 있습니다. EC2 인스턴스가 시작될 때마다 템플릿으로 사용할 AMI를 선택합니다. 프리베이킹은 애플리케이션 아티팩트의 상당 부분을 AMI 내에 임베딩하는 프로세스입니다.

애플리케이션 구성 요소를 AMI로 사전 베이킹하면 Amazon EC2 인스턴스를 시작하고 운영하는 시간을 단축할 수 있습니다. 배포 프로세스 중에 프리베이킹 및 부트스트래핑 사례를 결합하여 현재 환경에 맞게 사용자 지정된 새 인스턴스를 빠르게 생성할 수 있습니다.

블루/그린 배포

블루/그린 배포는 별개의 동일한 두 환경을 생성하는 배포 전략입니다. 한 환경(파란색)은 현재 애플리케이션 버전을 실행하고 다른 환경(녹색)은 새 애플리케이션 버전을 실행하고 있습니다. 블루/그린 배포 전략을 사용하면 배포에 실패할 경우 롤백 프로세스를 간소화하여 애플리케이션 가용성을 높이고 배포 위험을 줄일 수 있습니다. 그린 환경에서 테스트가 완료되면 라이브 애플리케이션 트래픽이 그린 환경으로 전달되고 블루 환경은 더 이상 사용되지 않습니다.

여러 AWS 배포 서비스는 Elastic Beanstalk, OpsWorks, CloudFormation, CodeDeploy 및 Amazon ECS를 포함한 블루/그린 배포 전략을 지원합니다. 애플리케이션에 블루/그린 배포 프로세스를 구현하기 위한 자세한 내용과 전략은 AWS의 블루/그린 배포를 참조하세요.

롤링 배포

롤링 배포는 애플리케이션이 실행 중인 인프라를 완전히 대체하여 애플리케이션의 이전 버전을 애플리케이션의 새 버전으로 천천히 대체하는 배포 전략입니다. 예를 들어 Amazon ECS의 롤링 배포에서 애플리케이션의 이전 버전을 실행하는 컨테이너는 애플리케이션의 새 버전을 실행하는 컨테이너로 one-by-one 교체됩니다.

롤링 배포는 일반적으로 블루/그린 배포보다 빠르지만 블루/그린 배포와 달리 롤링 배포에서는 이전 애플리케이션 버전과 새 애플리케이션 버전 간에 환경 격리가 없습니다. 이렇게 하면 롤링 배포가 더 빠르게 완료될 수 있지만 배포가 실패할 경우 위험을 높이고 롤백 프로세스를 복잡하게 만듭니다.

롤링 배포 전략은 대부분의 배포 솔루션에서 사용할 수 있습니다. CloudFormation을 사용한 롤링 배포에 대한 자세한 내용은 CloudFormation 업데이트 정책을 참조하고, Amazon ECS를 사용한 롤링 배포에 대한 자세한 내용은 Amazon ECS를 사용한 롤링 업데이트, Elastic Beanstalk를 사용한 롤링 배포에 대한 자세한 내용은 Elastic Beanstalk 롤링 환경 구성 업데이트를 참조하고, OpsWorks를 사용한 롤링 배포에 대한 자세한 내용은 에서 롤링 배포 사용을 AWS OpsWorks 참조하세요. CloudFormation

카나리 배포

카나리아 배포는 위험이 더 큰 블루/그린 배포 전략의 한 유형입니다. 이 전략에는 트래픽이 두 증분으로 새 버전의 애플리케이션으로 이동하는 단계별 접근 방식이 포함됩니다. 첫 번째 증분은 카나리아 그룹이라고 하는 작은 비율의 트래픽입니다. 이 그룹은 새 버전을 테스트하는 데 사용되며, 성공하면 트래픽이 두 번째 증분으로 새 버전으로 이동합니다.

Canary 배포는 두 단계로 구현하거나 선형으로 구현할 수 있습니다. 2단계 접근 방식에서는 새 애플리케이션 코드가 배포되고 평가판에 노출됩니다. 수락 시 환경의 나머지 부분으로 또는 선형 방식으로 롤아웃됩니다. 선형 접근 방식에는 모든 트래픽이 새 릴리스로 흐를 때까지 애플리케이션의 새 버전으로 트래픽을 점진적으로 늘리는 작업이 포함됩니다.

인 플레이스(in-place) 배포

인플레이스 배포는 인프라 구성 요소를 교체하지 않고 애플리케이션 버전을 업데이트하는 배포 전략입니다. 현재 위치 배포에서는 각 컴퓨팅 리소스의 이전 애플리케이션 버전이 중지되고, 최신 애플리케

이션이 설치되며, 애플리케이션의 새 버전이 시작되고 검증됩니다. 이렇게 하면 기본 인프라에 대한 장애를 최소화하면서 애플리케이션 배포를 진행할 수 있습니다.

현재 위치 배포를 사용하면 새 인프라를 생성하지 않고도 애플리케이션을 배포할 수 있지만 이러한 배포 중에 애플리케이션의 가용성에 영향을 미칠 수 있습니다. 또한이 접근 방식은 새 리소스 생성과 관련된 인프라 비용과 관리 오버헤드를 최소화합니다.

CodeDeploy에서 인플레이스 배포 전략을 사용하는 방법에 대한 자세한 내용은 인플레이스 배포 개요를 참조하세요.

배포 서비스 결합

AWS에는 '모든 상황에 맞는' 배포 솔루션이 없습니다. 배포 솔루션을 설계할 때는 애플리케이션 유형을 고려하는 것이 중요합니다. 어떤 AWS 서비스가 가장 적합한지를 결정할 수 있기 때문입니다. 애플리케이션을 프로비저닝, 구성, 배포, 확장 및 모니터링하는 완전한 기능을 제공하려면 여러 배포 서비스를 결합해야 하는 경우가 많습니다.

AWS에서 애플리케이션의 일반적인 패턴은 CloudFormation(및 확장)을 사용하여 범용 인프라를 관리하고 애플리케이션 업데이트를 관리하기 위한 보다 특수한 배포 솔루션을 사용하는 것입니다. 컨테이너화된 애플리케이션의 경우 CloudFormation을 사용하여 애플리케이션 인프라를 생성하고 Amazon ECS 및 Amazon EKS를 사용하여 컨테이너를 프로비저닝, 배포 및 모니터링할 수 있습니다.

AWS 배포 서비스는 타사 배포 서비스와 결합할 수도 있습니다. 이를 통해 조직은 AWS 배포 서비스를 기존 CI/CD 파이프라인 또는 인프라 관리 솔루션에 쉽게 통합할 수 있습니다. 예를 들어 OpsWorks를 사용하여 온프레미스 노드와 AWS 노드 간에 구성을 동기화할 수 있으며 CodeDeploy를 여러 타사 CI/CD 서비스와 함께 전체 파이프라인의 일부로 사용할 수 있습니다.

결론

AWS는 인프라 프로비저닝 및 애플리케이션 배포를 간소화하고 자동화하는 다양한 도구를 제공하며, 각 배포 서비스는 애플리케이션 관리를 위한 다양한 기능을 제공합니다. 성공적인 배포 아키텍처를 구축하려면 애플리케이션 및 조직의 요구 사항에 따라 각 서비스의 사용 가능한 기능을 평가합니다.

기여자

다음은 이 문서의 기여자입니다.

- Manikandan Chandrasekaran, 책임 기술자
- Anil Nadiminti, Senior Solutions Architect
- Bryant Bost, AWS ProServe 컨설턴트

참고 문헌

자세한 내용은 섹션을 참조하세요.

- AWS 백서 페이지
- AWS의 DevOps 소개 - 배포 전략

문서 수정

이 백서에 대한 업데이트 알림을 받으려면 RSS 피드를 구독하면 됩니다.

변경 사항	설명	날짜
백서 업데이트	최신 배포 서비스 및 전략을 위해 전체적으로 업데이트됨	2024년 5월 31일
마이너 업데이트	명확성을 위해 블루/그린 배포 섹션이 개정되었습니다.	2021년 4월 8일
백서 업데이트	최신 서비스 및 기능으로 업데이트되었습니다.	2020년 6월 3일
최초 게시	백서가 처음 게시되었습니다.	2015년 3월 1일

Notices

고객은 본 문서의 정보를 독립적으로 평가할 책임이 있습니다. 이 문서: (a) 정보 제공만을 목적으로 하고, (b) 현행 AWS 제품 제공 및 관행을 나타내며, (c) AWS와 그 계열사, 공급업체 또는 라이선스 제공자로부터 어떠한 약정이나 보증도 하지 않습니다. AWS 제품 또는 서비스는 명시적이든 묵시적이든 어떠한 종류의 보증, 진술 또는 조건 없이 “있는 그대로” 제공됩니다. 고객에 대한 AWS의 책임 및 채무는 AWS 계약에 준거합니다. 본 문서는 AWS와 고객 간의 어떠한 계약도 구성하지 않으며 이를 변경하지도 않습니다.

기계 번역으로 제공되는 번역입니다. 제공된 번역과 원본 영어의 내용이 상충하는 경우에는 영어 버전이 우선합니다.