AWS 백서

AWS의 DevOps 소개

AWS의 DevOps 소개: AWS 백서

Table of Contents

요약 1
요약 1
소개 2
지속적 통합 3
AWS CodeCommit 3
AWS CodeBuild 4
AWS CodeArtifact 4
지속적 전달 6
AWS CodeDeploy 6
AWS CodePipeline 7
배포 전략 9
현재 위치 배포 9
블루 및 그린 배포 9
카나리 배포 10
선형 배포 10
한 번에 모두 배포 10
배포 전략 매트릭스 11
AWS Elastic Beanstalk 배포 전략 11
코드형 인프라 13
AWS CloudFormation 13
AWS Cloud Development Kit 15
AWS Cloud Development Kit for Kubernetes 15
자동화 16
AWS OpsWorks 17
AWS Elastic Beanstalk 18
모니터링 및 로깅 19
Amazon CloudWatch 19
Amazon CloudWatch 경보 19
Amazon CloudWatch Logs 19
Amazon CloudWatch Logs Insights 20
Amazon CloudWatch Events 20
Amazon EventBridge 21
AWS CloudTrail 21
커뮤니케이션 및 협업 22

피자 두 판 팀 22
보안 23
AWS 공동 책임 모델 23
ID 및 액세스 관리 24
결론 25
문서 개정 26
기여자 27
고지 사항 28

AWS의 DevOps 소개

게시 날짜: 2020년 10월 16일(문서 개정)

요약

오늘날 기업들은 그 어느 때보다도 지속 가능하고 지속적인 비즈니스 가치를 달성하기 위해 고객과 더 깊은 관계를 구축하고자 디지털 트랜스포메이션 여정을 시작하고 있습니다. 모든 형태와 규모의 조직은 보다 빠르고 효율적인 혁신을 통해 경쟁업체들을 물리치고 새로운 시장에 진입하고 있습니다. 이러한 조직의 경우 혁신과 소프트웨어의 파괴적 혁신에 집중하는 것이 중요하므로 소프트웨어 제공을 간소화하는 것이 반드시 필요합니다. 아이디어에서 생산까지의 시간을 단축하여 속도와 민첩성을 최우선으로 생각하는 조직은 내일의 혁신 기업이 될 수 있습니다.

차세대 디지털 혁신 기업이 되기 위해 고려해야 할 몇 가지 요소가 있지만, 이 백서에서는 DevOps와 애플리케이션 및 서비스를 빠른 속도로 제공하는 조직의 능력을 향상시키는 데 도움이 되는 AWS 플랫폼의 서비스 및 기능에 초점을 맞추고 있습니다.

소개

DevOps는 높은 속도와 더 나은 품질로 애플리케이션과 서비스를 제공할 수 있는 조직의 능력을 향상시키는 문화, 엔지니어링 사례와 패턴, 그리고 도구의 조합입니다. 시간이 지남에 따라 DevOps를 도입할 때 지속적 통합, 지속적 전달, 코드형 인프라, 모니터링 및 로깅과 같은 몇 가지 필수 사례가 등장했습니다.

이 백서에서는 DevOps 여정을 가속화하는 데 도움이 되는 AWS 기능과 AWS 서비스가 DevOps 적응과 관련된 획일적인 부담을 제거하는 데 어떻게 도움이 되는지 설명합니다. 또한 서버를 관리하거나 노드를 구축하지 않고 지속적 통합 및 전달 기능을 구축하는 방법과 코드형 인프라를 활용하여 일관되고 반복 가능한 방식으로 클라우드 리소스를 프로비저닝하고 관리하는 방법을 중점적으로 살펴봅니다.

- 지속적 통합: 자동화된 구축 및 테스트가 수행된 후, 개발자가 코드 변경 사항을 중앙 리포지토리에 정기적으로 병합하는 소프트웨어 개발 방식입니다.
- 지속적 전달: 프로덕션에 릴리스할 수 있도록 코드 변경이 자동으로 구축, 테스트 및 준비되는 소프트웨어 개발 방식입니다.
- 코드형 인프라: 코드와 버전 관리 및 지속적 통합과 같은 소프트웨어 개발 기술을 사용하여 인프라를 프로비저닝하고 관리하는 방식입니다.
- 모니터링 및 로깅: 조직은 모니터링 및 로깅을 통해 애플리케이션 및 인프라 성능이 제품 최종 사용자의 경험에 어떤 영향을 미치는지 확인할 수 있습니다.
- 커뮤니케이션 및 협업: 워크플로를 구축하고 DevOps에 대한 책임을 분배하여 팀을 더 긴밀하게 만들기 위한 방식입니다.
- 보안: 공통 관심 사항이어야 합니다. 지속적 통합 및 지속적 전달(CI/CD) 파이프라인 및 관련 서비스를 보호하고 적절한 액세스 제어 권한을 설정해야 합니다.

이러한 각 원칙을 살펴보면 Amazon Web Services(AWS)에서 제공하는 제품과 밀접한 관련이 있음을 알 수 있습니다.

지속적 통합

지속적 통합(CI)은 개발자가 코드 변경 사항을 정기적으로 중앙 코드 리포지토리에 병합하면, 나중에 자동 구축 및 테스트가 실행되는 소프트웨어 개발 방식입니다. CI는 버그를 신속하게 찾아 해결하고, 소프트웨어 품질을 개선하고, 새로운 소프트웨어 업데이트를 검증 및 릴리스하는 데 걸리는 시간을 단축할 수 있도록 돕습니다.

AWS는 지속적 통합을 위해 다음과 같은 서비스를 제공합니다.

주제

- AWS CodeCommit
- AWS CodeBuild
- AWS CodeArtifact

AWS CodeCommit

AWS CodeCommit은 프라이빗 Git 리포지토리를 호스팅하는 안전하고 확장 가능한 관리형 소스 제어 서비스입니다. CodeCommit을 사용하면 자체 소스 제어 시스템을 운영할 필요가 없으며 하드웨어를 프로비저닝하고 확장하거나 소프트웨어를 설치, 구성 및 운영할 필요가 없습니다. CodeCommit을 통해 무엇이든 코드에서 바이너리로 저장할 수 있으며, Git의 표준 기능을 지원하여 기존 Git 기반 도구와 원활하게 작동합니다. 또한, 팀에서 CodeCommit의 온라인 코드 도구를 사용하여 프로젝트를 찾아보고 수정 및 공동 작업할 수 있습니다. AWS CodeCommit에는 몇 가지 이점이 있습니다.

협업 - AWS CodeCommit은 협업 소프트웨어 개발을 위해 설계되었습니다. 손쉽게 코드를 커밋, 브랜치 및 병합할 수 있으므로 팀 프로젝트에 대한 제어를 간편하게 유지 관리할 수 있습니다. 또한 CodeCommit은 협업자에게 코드 검토를 요청하고 코드를 논의할 수 있는 메커니즘을 제공하는 pull 요청을 지원합니다.

암호화 - 원하는 대로 HTTPS나 SSH를 사용해 AWS CodeCommit에서 파일을 송수신할 수 있습니다. 또한 리포지토리는 고객별 키를 사용하여 AWS Key Management Service(AWS KMS)를 통해 저장 시 자동으로 암호화됩니다.

액세스 제어 - AWS CodeCommit은 AWS Identity and Access Management(IAM)를 사용하여 데이터에 액세스할 수 있는 방법, 시기 및 위치 외에도 데이터에 액세스할 수 있는 사용자를 제어하고 모니터링합니다. 또한 CodeCommit은 AWS CloudTrail 및 Amazon CloudWatch를 통해 리포지토리를 모니터링하는 데 도움이 됩니다.

고가용성 및 내구성 - AWS CodeCommit은 리포지토리를 Amazon Simple Storage Service(Amazon S3) 및 Amazon DynamoDB에 저장합니다. 암호화된 데이터는 여러 시설에 걸쳐 중복으로 저장됩니다. 이러한 아키텍처는 리포지토리 데이터의 가용성과 내구성을 높입니다.

알림 및 사용자 지정 스크립트 - 이제 리포지토리에 영향을 주는 이벤트에 대한 알림을 받을 수 있습니다. 알림은 Amazon Simple Notification Service(Amazon SNS) 알림으로 수신됩니다. 각 알림에는 상태 메시지와 함께 해당 알림이 생성되도록 한 이벤트가 발생한 리소스의 링크가 포함됩니다. 뿐만 아니라, AWS CodeCommit 리포지토리 트리거를 사용하면 Amazon SNS를 사용해 알림을 보내고 HTTP 웹후크를 생성하거나 선택한 리포지토리 이벤트에 대한 응답으로 AWS Lambda 함수를 호출할 수 있습니다.

AWS CodeBuild

AWS CodeBuild는 소스 코드를 컴파일하고 테스트를 실행하며 배포 준비가 완료된 소프트웨어 패키지를 생성하는 완전관리형 지속적 통합 서비스입니다. 자체 구축 서버를 프로비저닝, 관리 및 조정할 필요가 없습니다. CodeBuild는 GitHub, GitHub Enterprise, BitBucket, AWS CodeCommit 또는 Amazon S3 중 하나를 소스 공급자로 사용할 수 있습니다.

CodeBuild는 지속적으로 확장되며 여러 구축을 동시에 처리할 수 있습니다. CodeBuild는 다양한 버전의 Microsoft Windows 및 Linux를 위해 사전 구성된 다양한 환경을 제공합니다. 고객은 맞춤형 구축 환경을 Docker 컨테이너로 가져올 수도 있습니다. 또한 CodeBuild는 Jenkins 및 Spinnaker와 같은 오픈 소스 도구와도 통합됩니다.

CodeBuild는 유닛, 기능 또는 통합 테스트에 대한 보고서를 생성할 수 있습니다. 이러한 보고서는 실행된 테스트 사례 수와 통과 또는 실패 횟수를 시각적으로 보여 줍니다. 구축 프로세스는 Amazon Virtual Private Cloud(Amazon VPC) 내에서도 실행할 수 있으며, 이는 통합 서비스 또는 데이터베이스가 VPC 내에 배포된 경우 유용할 수 있습니다.

AWS CodeArtifact

AWS CodeArtifact는 조직이 소프트웨어 개발 프로세스에서 사용되는 소프트웨어 패키지를 안전하게 저장, 게시 및 공유할 수 있도록 지원하는 완전관리형 아티팩트 리포지토리 서비스입니다. 개발자가 최신 버전에 액세스할 수 있도록 퍼블릭 아티팩트 리포지토리에서 소프트웨어 패키지와 종속성을 자동으로 가져오도록 CodeArtifact를 구성할 수 있습니다.

소프트웨어 개발 팀은 애플리케이션 패키지에서 일반적인 작업을 수행하기 위해 오픈 소스 패키지에 점점 더 의존하고 있습니다. 이제 소프트웨어 개발 팀이 취약점이 없는 특정 버전의 오픈 소스 소프트

웨어에 대한 제어를 유지하는 것이 중요해졌습니다. CodeArtifact를 사용하면 위의 사항을 적용하는 제어 기능을 설정할 수 있습니다.

CodeArtifact는 일반적으로 사용되는 패키지 관리자와 구축 도구(예: Maven, Gradle, npm, yarn, twine 및 pip)에서 작동하므로 기존 개발 워크플로에 손쉽게 통합할 수 있습니다.

지속적 전달

지속적 전달은 프로덕션에 릴리스할 수 있도록 코드 변경이 자동으로 준비되는 소프트웨어 개발 방식입니다. 현대적 애플리케이션 개발의 기반인 지속적 전달은 구축 단계 이후의 모든 코드 변경을 테스트 환경 및/또는 프로덕션 환경에 배포함으로써 지속적 통합을 확장합니다. 적절하게 구현할 경우, 개발자는 언제나 즉시 배포할 수 있고 표준화된 테스트 프로세스를 통과한 구축 아티팩트를 보유할 수 있습니다.

지속적 전달에서는 개발자가 단순한 유닛 테스트 외에도 다양한 테스트를 자동화할 수 있으므로, 고객에게 배포하기 전에 여러 차원에서 애플리케이션 업데이트를 확인할 수 있습니다. 이러한 테스트에는 UI 테스트, 로드 테스트, 통합 테스트, API 안정성 테스트 등이 포함될 수 있습니다. 이를 통해 개발자는 업데이트를 좀 더 철저히 검증하고 문제를 사전에 발견할 수 있습니다. 온프레미스에서는 힘들었지만, 클라우드에서는 테스트용으로 여러 개의 환경을 생성하고 복제하는 작업을 비용 효율적으로 손쉽게 자동화할 수 있습니다.

AWS는 지속적 전달을 위해 다음과 같은 서비스를 제공합니다.

- AWS CodeBuild
- AWS CodeDeploy
- AWS CodePipeline

주제

- AWS CodeDeploy
- AWS CodePipeline

AWS CodeDeploy

AWS CodeDeploy는 Amazon Elastic Compute Cloud(Amazon EC2), AWS Fargate, AWS Lambda 등의 다양한 컴퓨팅 서비스와 사용자의 온프레미스 서버에 대한 소프트웨어 배포를 자동화하는 완전관리형 배포 서비스입니다. AWS CodeDeploy를 사용하면 새로운 기능을 보다 쉽고 빠르게 릴리스할 수 있고, 애플리케이션 배포 중 가동 중지 시간을 피할 수 있으며, 애플리케이션 업데이트의 복잡성을 처리할 수 있습니다. CodeDeploy를 사용하여 소프트웨어 배포를 자동화하면 오류가 발생하기 쉬운 수동 작업을 할 필요가 없습니다. 이 서비스는 배포 요구 사항에 따라 확장됩니다.

CodeDeploy에는 지속적 배포라는 DevOps 원칙에 부합하는 몇 가지 이점이 있습니다.

자동화된 배포: CodeDeploy에서 소프트웨어 배포를 완전히 자동화하므로 신속하고 안정적으로 배포할 수 있습니다.

중앙 집중식 제어: CodeDeploy를 사용하면 AWS 관리 콘솔 또는 AWS CLI를 통해 손쉽게 애플리케이션 배포를 시작하고 해당 배포 상태를 추적할 수 있습니다. CodeDeploy에서는 상세한 보고서를 제공하므로 각 애플리케이션 수정 버전이 언제 어디에 배포되었는지 확인할 수 있습니다. 또한, 배포에 대한 라이브 업데이트를 받도록 푸시 알림을 생성할 수 있습니다.

가동 중지 시간 최소화: CodeDeploy는 소프트웨어 배포 프로세스 동안 애플리케이션의 가용성을 최대화하도록 지원합니다. 변경 사항을 점진적으로 도입하고 구성 가능한 규칙에 따라 애플리케이션 상태를 추적합니다. 오류가 발생하는 경우 손쉽게 소프트웨어 배포를 중단하고 롤백할 수 있습니다.

간편한 도입: CodeDeploy는 모든 애플리케이션에서 작동하며 다양한 플랫폼과 언어에서 동일한 환경을 제공합니다. 또한 기존 설정 코드를 쉽게 재사용할 수 있습니다. CodeDeploy는 기존 소프트웨어 릴리스 프로세스나 지속적 전달 도구 체인(예: AWS CodePipeline, GitHub, Jenkins)과도 통합할 수 있습니다.

AWS CodeDeploy는 여러 배포 옵션을 지원합니다. 자세한 내용은 배포 전략을 참조하세요.

AWS CodePipeline

AWS CodePipeline은 소프트웨어를 릴리스하는 데 필요한 단계를 모델링, 시각화 및 자동화하도록 해주는 지속적 전달 서비스입니다. AWS CodePipeline을 사용하면 코드를 구축하고, 사전 프로덕션 환경에 배포하고, 애플리케이션을 테스트하고, 프로덕션으로 릴리스하기 위한 전체 릴리스 프로세스를 모델링할 수 있습니다. 그런 다음 AWS CodePipeline은 코드가 변경될 때마다 정의된 워크플로에 따라 애플리케이션을 구축, 테스트 및 배포합니다. APN 파트너 도구 및 자체 사용자 지정 도구를 릴리스 프로세스 중 원하는 단계에 통합하여 포괄적인 지속적 전달 솔루션을 구축할 수 있습니다.

AWS CodePipeline에는 지속적 배포라는 DevOps 원칙에 부합하는 몇 가지 이점이 있습니다.

신속한 전달: AWS CodePipeline은 소프트웨어 출시 프로세스를 자동화하므로 새로운 기능을 신속하게 릴리스할 수 있습니다. CodePipeline을 사용하면 피드백에 따라 신속하게 반복하고 사용자에게 새로운 기능을 더 빠르게 제공할 수 있습니다.

품질 향상: 구축, 테스트, 릴리스 프로세스를 자동화하는 AWS CodePipeline을 사용하면 모든 새로운 변경 사항을 일관된 품질 검사 세트를 통해 실행하여 소프트웨어 업데이트의 속도와 품질을 높일 수 있습니다.

간편한 통합: AWS CodePipeline은 특정한 요구에 맞게 손쉽게 확장할 수 있습니다. 사전 구축된 플러그 인 또는 자체 사용자 지정 플러그 인을 릴리스 프로세스 중 원하는 단계에 사용할 수 있습니다. 예를

들면 GitHub에서 소스 코드를 가져오거나, 온프레미스 Jenkins 구축 서버를 사용하거나, 서드 파티 서비스를 사용하여 로드 테스트를 실행하거나, 사용자 지정 운영 대시보드로 배포 정보를 전달할 수 있습니다.

구성 가능한 워크플로: AWS CodePipeline은 콘솔 인터페이스, AWS CLI, AWS CloudFormation 또는 AWS SDK를 사용하여 소프트웨어 릴리스 프로세스의 여러 단계를 모델링할 수 있습니다. 손쉽게 실행할 테스트를 지정하고 애플리케이션과 관련 종속성을 배포할 단계를 사용자 지정할 수 있습니다.

배포 전략

배포 전략은 소프트웨어를 제공하는 방법을 정의합니다. 조직은 비즈니스 모델에 따라 다양한 배포 전략을 따릅니다. 일부는 완전히 테스트된 소프트웨어를 제공하도록 선택할 수도 있고, 다른 일부는 사용자가 피드백을 제공하고 개발 중인 기능(예: 베타 릴리스)을 사용자가 평가하기를 원할 수도 있습니다. 다음 단원에서는 다양한 배포 전략에 대해 설명합니다.

주제

- 현재 위치 배포
- 블루 및 그린 배포
- 카나리 배포
- 선형 배포
- 한 번에 모두 배포

현재 위치 배포

이 전략에서는 배포 그룹의 각 인스턴스에 있는 애플리케이션이 중지되고 최신 애플리케이션 수정 버전이 설치되며 애플리케이션의 새 버전이 시작되고 유효성이 검사되는 방식으로 배포가 수행됩니다. 로드 밸런서를 사용하면 배포가 진행될 때 각 인스턴스를 등록 취소한 후 배포가 완료되면 서비스로 복원할 수 있습니다. 현재 위치 배포는 서비스 중단을 가정하고 한 번에 모두 수행하거나 롤링 업데이트로 수행할 수 있습니다. AWS CodeDeploy 및 AWS Elastic Beanstalk은 한 번에 하나씩, 한 번에 절반씩 그리고 한 번에 모두 배포 구성을 제공합니다. 현재 위치 배포를 위한 이러한 동일한 배포 전략을 블루/그린 배포에도 사용할 수 있습니다.

블루 및 그린 배포

블루/그린 배포는 서로 다른 버전의 애플리케이션을 실행하는 2개의 동일한 환경 간에 트래픽을 이동함으로써 애플리케이션을 릴리스하는 기술입니다. 블루/그린 배포는 애플리케이션 업데이트 중에 가동 중지 시간을 최소화하여 가동 중지 및 롤백 기능과 관련된 위험을 완화하는 데 도움이 됩니다. 블루/그린 배포를 사용하면 이전 버전(블루)과 함께 애플리케이션의 새 버전(그린)을 시작하고, 새 버전으로 트래픽을 다시 라우팅하기 전에 새 버전을 모니터링 및 테스트하여 문제가 탐지되면 롤백할 수 있습니다.

카나리 배포

트래픽이 2증분씩 이동합니다. 카나리 배포는 위험을 회피하는 블루/그린 전략으로, 단계적 접근 방식이 사용됩니다. 카나리 배포는 2단계 또는 선형일 수 있으며(새로운 애플리케이션 코드가 배포되고 평가를 위해 공개됨) 수락 시 환경의 나머지 부분에 배포되거나 선형 방식으로 배포됩니다.

선형 배포

선형 배포는 트래픽이 동일한 증분으로 이동하며 각 증분 간에 시간(분)이 동일합니다. 각 증분에서 이동되는 트래픽 비율(%)과 각 증분 간의 시간(분)을 지정하는 사전 정의된 선형 옵션 중에서 선택할 수 있습니다.

한 번에 모두 배포

한 번에 모두 배포는 모든 트래픽이 원래 환경에서 대체 환경으로 한 번에 이동한다는 것을 의미합니다.

배포 전략 매트릭스

다음 표에는 Amazon Elastic Container Service(Amazon ECS), AWS Lambda 및 Amazon EC2/온프레미스에 지원되는 배포 전략이 나와 있습니다.

- Amazon ECS는 완전관리형 오케스트레이션 서비스입니다.
- AWS Lambda를 사용하면 서버를 프로비저닝하거나 관리할 필요 없이 코드를 실행할 수 있습니다.
- Amazon EC2를 사용하면 클라우드에서 안전하고 크기 조정이 가능한 컴퓨팅 용량을 실행할 수 있습니다.

	A	B	C	D
1	배포 전략 매트릭스	Amazon ECS	AWS Lambda	Amazon EC2/온프레미스
2	현재 위치	✓	✓	✓
3	블루/그린	✓	✓	✓*
4	카나리	✓	✓	X
5	선형	✓	✓	X
6	한 번에 모두	✓	✓	X

> **Note**
>
> EC2/온프레미스를 사용한 블루/그린 배포는 EC2 인스턴스에서만 작동합니다.

AWS Elastic Beanstalk 배포 전략

AWS Elastic Beanstalk은 다음과 같은 유형의 배포 전략을 지원합니다.

- 한 번에 모두: 모든 인스턴스에서 현재 위치 배포를 수행합니다.
- 롤링: 인스턴스를 배치로 분할하고 한 번에 하나의 배치에 배포합니다.

- 추가 배치를 사용한 롤링: 배포를 배치로 분할하지만 첫 번째 배치의 경우 기존 EC2 인스턴스에 배포하는 대신 새 EC2 인스턴스를 생성합니다.
- 변경 불가능: 기존 인스턴스를 사용하는 대신 새 인스턴스로 배포해야 하는 경우 사용합니다.
- 트래픽 분할: 변경 불가능한 배포를 수행한 다음 사전 결정된 기간 동안 트래픽의 일정 비율을 새 인스턴스로 전달합니다. 인스턴스가 정상 상태를 유지하면 모든 트래픽을 새 인스턴스로 전달하고 이전 인스턴스를 종료합니다.

코드형 인프라

DevOps의 기본 원칙은 개발자가 코드를 처리하는 것과 동일한 방식으로 인프라를 처리하는 것입니다. 애플리케이션 코드에는 정의된 형식과 구문이 있습니다. 프로그래밍 언어의 규칙에 따라 코드를 작성하지 않으면 애플리케이션을 만들 수 없습니다. 코드는 코드 개발, 변경 및 버그 수정 기록을 기록하는 버전 관리 또는 소스 제어 시스템에 저장됩니다. 코드를 컴파일하거나 애플리케이션에 구축하면 일관된 애플리케이션 생성과 반복 가능하고 신뢰할 수 있는 구축을 보장할 수 있습니다.

코드형 인프라를 실행한다는 것은 이와 같은 엄격한 애플리케이션 코드 개발을 인프라 프로비저닝에도 적용한다는 것을 의미합니다. 모든 구성은 선언적 방식으로 정의되어야 하며 애플리케이션 코드와 마찬가지로 AWS CodeCommit과 같은 소스 제어 시스템에 저장되어야 합니다. 인프라 프로비저닝, 오케스트레이션 및 배포도 코드형 인프라 사용을 지원해야 합니다.

인프라는 전통적으로 스크립트와 수동 프로세스의 조합을 사용하여 프로비저닝되었습니다. 때때로 이러한 스크립트는 버전 관리 시스템에 저장되거나 텍스트 파일이나 런북에 단계별로 문서화되었습니다. 런북을 작성하는 사람과 이러한 스크립트를 실행하거나 런북의 단계를 수행하는 사람이 동일인이 아닌 경우가 많습니다. 이러한 스크립트 또는 런북을 자주 업데이트하지 않으면 배포에 중대한 영향을 주는 문제로 발전할 수 있습니다. 이로 인해 새로운 환경의 생성이 항상 반복 가능하거나 신뢰할 수 있거나 일관성이 있는 것이 아니게 됩니다.

앞서 설명한 것과 달리 AWS는 DevOps 중심의 인프라 생성 및 유지 관리 방법을 제공합니다. 소프트웨어 개발자가 애플리케이션 코드를 작성하는 방식과 유사하게 AWS는 프로그래밍 방식, 설명 및 선언 방식으로 인프라를 생성, 배포 및 유지 관리할 수 있는 서비스를 제공합니다. 이러한 서비스는 엄격하고 명확하며 신뢰성을 제공합니다. 이 백서에서 설명하는 AWS 서비스는 DevOps 방법론의 핵심이며, 여러 상위 수준의 AWS DevOps 원칙 및 사례의 토대가 됩니다.

AWS는 코드형 인프라를 정의하기 위해 다음과 같은 서비스를 제공합니다.

- AWS CloudFormation
- AWS Cloud Development Kit (AWS CDK)
- AWS Cloud Development Kit for Kubernetes

AWS CloudFormation

AWS CloudFormation은 개발자가 정돈되고 예측 가능한 방식으로 AWS 리소스를 생성할 수 있게 해주는 서비스입니다. 리소스는 JavaScript Object Notation(JSON) 또는 Yet Another Markup Language(YAML) 형식을 사용하여 텍스트 파일로 작성됩니다. 템플릿에는 생성 및 관리되는 리소스

유형에 따라 달라지는 특정 구문 및 구조가 필요합니다. AWS Cloud9와 같은 코드 편집기를 사용하여 JSON 또는 YAML로 리소스를 작성하고 버전 관리 시스템에 체크인하면 CloudFormation이 지정된 서비스를 안전하고 반복 가능한 방식으로 구축합니다.

CloudFormation 템플릿은 AWS 환경에 스택으로 배포됩니다. AWS 관리 콘솔, AWS Command Line Interface 또는 AWS CloudFormation API를 통해 스택을 관리할 수 있습니다. 스택에서 실행 중인 리소스를 변경해야 하는 경우 스택을 업데이트합니다. 리소스를 변경하기 전에 제안된 변경 사항이 요약된 변경 세트를 생성할 수 있습니다. 변경 세트를 사용하면 변경 사항을 구현하기 이전에 해당 변경이 실행 중인 리소스 특히, 중요 리소스에 미치는 영향을 확인할 수 있습니다.

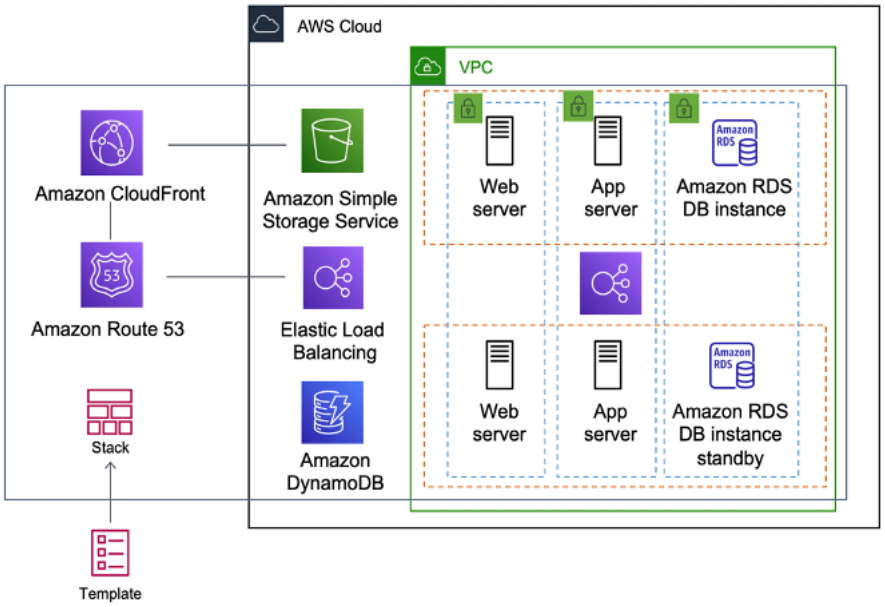

그림 1 - 하나의 템플릿 워크플로에서 전체 환경(스택)을 생성하는 AWS CloudFormation

단일 템플릿을 사용하여 전체 환경을 생성 및 업데이트하거나 개별 템플릿을 사용하여 환경 내의 여러 계층을 관리할 수 있습니다. 이를 통해 템플릿을 모듈화할 수 있으며 많은 조직에 중요한 거버넌스 계층도 제공할 수 있습니다.

콘솔에서 스택을 생성하거나 업데이트하면 구성 상태를 보여 주는 이벤트가 표시됩니다. 오류가 발생하면 기본적으로 스택이 이전 상태로 롤백됩니다. Amazon Simple Notification Service(Amazon SNS)는 이벤트에 대한 알림을 제공합니다. 예를 들어 Amazon SNS를 사용하면 이메일을 통해 스택 생성 및 삭제 진행 상황을 추적하고 프로그래밍 방식으로 다른 프로세스와 통합할 수 있습니다.

AWS CloudFormation을 사용하여 간편하게 AWS 리소스 모음을 정리하고 배포할 수 있으며 스택을 구성할 때 전달할 특수 파라미터와 모든 종속성을 설명할 수 있습니다.

CloudFormation 템플릿을 사용하면 Amazon S3, Auto Scaling, Amazon CloudFront, Amazon DynamoDB, Amazon EC2, Amazon ElastiCache, AWS Elastic Beanstalk, Elastic Load Balancing,

IAM, AWS OpsWorks 및 Amazon VPC와 같은 다양한 AWS 서비스로 작업할 수 있습니다. 지원되는 리소스의 최신 목록은 AWS 리소스 및 속성 유형 참조를 참조하세요.

AWS Cloud Development Kit

AWS Cloud Development Kit (AWS CDK)는 익숙한 프로그래밍 언어를 사용하여 클라우드 애플리케이션 리소스를 모델링 및 프로비저닝할 수 있는 오픈 소스 소프트웨어 개발 프레임워크입니다. AWS CDK를 사용하면 TypeScript, Python, Java 및 .NET을 사용하여 애플리케이션 인프라를 모델링할 수 있습니다. 개발자는 기존 통합 개발 환경(IDE)과 자동 완성 및 인라인 문서와 같은 도구를 활용하여 인프라 개발을 가속화할 수 있습니다.

AWS CDK는 백그라운드에서 AWS CloudFormation을 활용하여 안전하고 반복 가능한 방식으로 리소스를 프로비저닝합니다. 구문은 CDK 코드의 기본 구성 요소입니다. 구문은 클라우드 구성 요소를 나타내며 AWS CloudFormation이 구성 요소를 생성하는 데 필요한 모든 것을 캡슐화합니다. AWS CDK에는 많은 AWS 서비스를 나타내는 AWS Construct Library가 포함되어 있습니다. 구문을 함께 결합하면 AWS에서 배포할 복잡한 아키텍처를 빠르고 쉽게 생성할 수 있습니다.

AWS Cloud Development Kit for Kubernetes

AWS Cloud Development Kit for Kubernetes(cdk8s)는 범용 프로그래밍 언어를 사용하여 Kubernetes 애플리케이션을 정의하기 위한 오픈 소스 소프트웨어 개발 프레임워크입니다.

프로그래밍 언어로 애플리케이션을 정의하면(문서 게시 날짜 기준으로 Python과 TypeScript만 지원됨) cdk8s가 애플리케이션 설명을 Kubernetes 이전의 YML로 변환합니다. 이 YML 파일은 어디서나 실행되는 모든 Kubernetes 클러스터에서 사용할 수 있습니다. 구조는 프로그래밍 언어로 정의되므로 프로그래밍 언어에서 제공하는 다양한 기능을 사용할 수 있습니다. 프로그래밍 언어의 추상화 기능을 사용하여 고유한 상용구 코드를 만들고 모든 배포에서 재사용할 수 있습니다.

자동화

DevOps의 또 다른 핵심 철학과 사례는 자동화입니다. 자동화는 인프라와 인프라에서 실행되는 애플리케이션의 설정, 구성, 배포 및 지원에 중점을 둡니다. 자동화를 사용하면 표준화되고 반복 가능한 방식으로 환경을 보다 신속하게 설정할 수 있습니다. 수동 프로세스를 제거하는 것은 성공적인 DevOps 전략의 핵심입니다. 지금까지 서버 구성 및 애플리케이션 배포는 주로 수동 프로세스였습니다. 환경은 표준화되지 않고 따라서 문제가 발생했을 때 환경을 재현하는 것이 어렵습니다.

자동화의 사용은 클라우드의 모든 이점을 실현하는 데 매우 중요합니다. 내부적으로 AWS는 탄력성과 확장성의 핵심 기능을 제공하기 위해 자동화에 크게 의존합니다. 수동 프로세스는 오류가 발생하기 쉽고 신뢰할 수 없으며 민첩한 비즈니스를 지원하기에 부적합합니다. 종종 조직은 비즈니스 내에서 더 중요하고 가치가 높은 다른 활동을 지원하는 데 시간을 잘 활용할 수 있는 경우 고도로 숙련된 리소스를 수동 구성을 제공하는 데 할애할 수 있습니다.

현대적 운영 환경은 일반적으로 수동 개입이나 프로덕션 환경에 대한 액세스를 제거하기 위해 완전 자동화에 의존합니다. 여기에는 모든 소프트웨어 릴리스, 시스템 구성, 운영 체제 패치 적용, 문제 해결 또는 버그 수정이 포함됩니다. 여러 수준의 자동화 사례를 함께 사용하여 더 높은 수준의 엔드 투 엔드 자동화 프로세스를 제공할 수 있습니다.

자동화의 주요 이점은 다음과 같습니다.

- 신속한 변경
- 생산성 향상
- 반복 가능한 구성
- 재현 가능한 환경
- 탄력성 활용
- 자동 크기 조정 활용
- 테스트 자동화

자동화는 AWS 서비스의 초석이며 모든 서비스, 기능 및 제품에서 내부적으로 지원됩니다.

주제

- AWS OpsWorks
- AWS Elastic Beanstalk

AWS OpsWorks

AWS OpsWorks는 DevOps의 원칙을 AWS Elastic Beanstalk보다 훨씬 더 효율적으로 사용합니다. 이는 단순한 애플리케이션 컨테이너가 아닌 애플리케이션 관리 서비스로 간주할 수 있습니다. AWS OpsWorks는 구성 관리 소프트웨어(Chef)와의 통합 및 애플리케이션 수명 주기 관리와 같은 추가 기능을 통해 훨씬 더 높은 수준의 자동화를 제공합니다. 애플리케이션 수명 주기 관리를 사용하여 리소스가 설정, 구성, 배포, 배포 취소, 종료되는 시기를 정의할 수 있습니다.

유연성을 높이기 위해 AWS OpsWorks에서는 구성 가능한 스택에 애플리케이션을 정의해야 합니다. 사전 정의된 애플리케이션 스택을 선택할 수도 있습니다. 애플리케이션 스택에는 애플리케이션 서버, 웹 서버, 데이터베이스 및 로드 밸런서를 비롯하여 애플리케이션에 필요한 AWS 리소스에 대한 모든 프로비저닝이 포함됩니다.

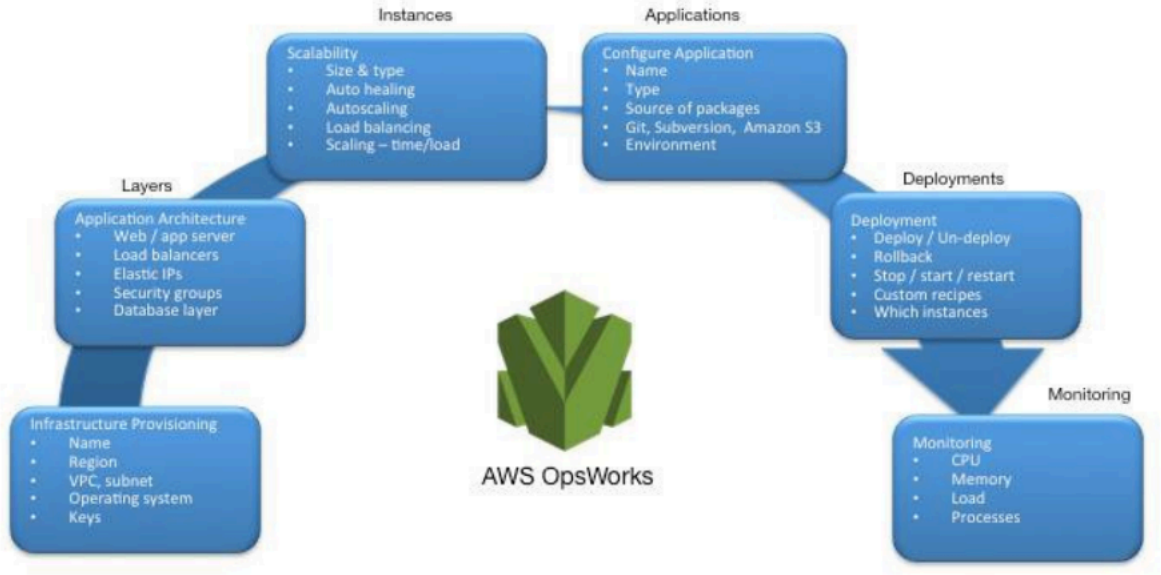

그림 2 - DevOps 기능 및 아키텍처를 보여 주는 AWS OpsWorks

애플리케이션 스택은 스택을 독립적으로 유지 관리할 수 있도록 아키텍처 계층으로 구성됩니다. 예제로 나온 계층에는 웹 계층, 애플리케이션 계층 및 데이터베이스 계층이 포함될 수 있습니다. 또한 AWS OpsWorks는 기본적으로 Auto Scaling 그룹 및 Elastic Load Balancing 로드 밸런서 설정을 간소화하여 DevOps의 자동화 원칙을 구체적으로 보여 줍니다. AWS Elastic Beanstalk과 마찬가지로 AWS OpsWorks는 애플리케이션 버전 관리, 지속적인 배포 및 인프라 구성 관리를 지원합니다.

AWS OpsWorks는 DevOps의 모니터링 및 로깅 사례도 지원합니다(다음 단원에서 설명). 모니터링 지원은 Amazon CloudWatch에서 제공합니다. 모든 수명 주기 이벤트가 기록되고 별도의 Chef 로그에는 실행되는 모든 Chef 레시피가 예외와 함께 문서화됩니다.

AWS Elastic Beanstalk

AWS Elastic Beanstalk은 Java, .NET, PHP, Node.js, Python, Ruby, Go, Docker를 사용하여 개발된 웹 애플리케이션을 Apache, NGINX, Passenger, IIS와 같은 친숙한 서버에 신속하게 배포하고 확장하기 위한 서비스입니다.

Elastic Beanstalk은 Amazon EC2 Auto Scaling을 기반으로 한 추상화이며 복제, 블루/그린 배포, Elastic Beanstalk Command Line Interface(EB CLI)와 개발자 생산성 향상을 위한 AWS Toolkit for Visual Studio, Visual Studio Code, Eclipse 및 IntelliJ와의 통합 같은 추가 기능을 제공하여 배포를 간소화합니다.

모니터링 및 로깅

커뮤니케이션과 협업은 DevOps 철학의 기본입니다. 이를 용이하게 하기 위해서는 피드백이 중요합니다. AWS에서는 두 가지 핵심 서비스인 Amazon CloudWatch와 AWS CloudTrail에서 피드백을 제공합니다. 이 두 서비스는 개발자와 운영 팀이 긴밀하고 투명하게 협력할 수 있도록 강력한 모니터링, 알림 및 감사 인프라를 제공합니다.

AWS는 모니터링 및 로깅을 위해 다음과 같은 서비스를 제공합니다.

주제

- Amazon CloudWatch
- Amazon CloudWatch 경보
- Amazon CloudWatch Logs
- Amazon CloudWatch Logs Insights
- Amazon CloudWatch Events
- Amazon EventBridge
- AWS CloudTrail

Amazon CloudWatch

Amazon CloudWatch 지표는 Amazon EC2 인스턴스, Amazon EBS 볼륨 및 Amazon RDS DB 인스턴스와 같은 AWS 서비스에서 데이터를 자동으로 수집합니다. 그런 다음 이러한 지표를 대시보드로 구성하고 경보 또는 이벤트를 생성하여 이벤트를 트리거하거나 Auto Scaling 작업을 수행할 수 있습니다.

Amazon CloudWatch 경보

Amazon CloudWatch 지표로 수집된 지표를 기반으로 경보를 설정할 수 있습니다. 그러면 경보가 Amazon Simple Notification Service(Amazon SNS) 주제에 알림을 보내거나 Auto Scaling 작업을 시작할 수 있습니다. 경보에는 기간(지표를 평가하는 데 걸리는 시간), 평가 기간(가장 최근 데이터 포인트 수) 및 경보를 생성할 데이터 포인트(평가 기간 내의 데이터 포인트 수)가 필요합니다.

Amazon CloudWatch Logs

Amazon CloudWatch Logs는 로그 집계 및 모니터링 서비스입니다. AWS CodeBuild, CodeCommit, CodeDeploy 및 CodePipeline은 CloudWatch 로그와의 통합을 제공하므로 모든 로그를 중앙에서 모니

터링할 수 있습니다. 또한 앞서 언급한 다양한 AWS 서비스는 CloudWatch와의 직접적인 통합을 제공합니다.

CloudWatch Logs를 사용하면 다음을 수행할 수 있습니다.

- 로그 데이터 쿼리
- Amazon EC2 인스턴스의 로그 모니터링
- AWS CloudTrail에서 기록한 이벤트 모니터링
- 로그 보존 정책 정의

Amazon CloudWatch Logs Insights

Amazon CloudWatch Logs Insights는 로그를 스캔하여 대화형 쿼리 및 시각화를 수행할 수 있도록 지원합니다. Amazon CloudWatch Logs Insights는 다양한 로그 형식을 이해하고 JSON 로그에서 필드를 자동으로 검색합니다.

Amazon CloudWatch Events

Amazon CloudWatch Events는 AWS 리소스의 변경 사항을 설명하는 시스템 이벤트의 스트림을 거의 실시간으로 제공합니다. 신속하게 설정할 수 있는 단순 규칙을 사용하여 일치하는 이벤트를 검색하고 하나 이상의 대상 함수 또는 스트림으로 이를 라우팅할 수 있습니다. CloudWatch Events는 운영 변경 사항이 발생하면 이를 인식합니다. CloudWatch Events는 이러한 운영 변경에 응답하고, 환경에 응답하기 위한 메시지를 전송하고, 함수를 활성화하며, 변경을 수행하고, 상태 정보를 캡처하는 등 필요에 따라 교정 조치를 취합니다.

CloudWatch Events에서 규칙을 구성하여 AWS 서비스의 변경 사항을 알리고 Amazon EventBridge를 사용하여 이러한 이벤트를 다른 서드 파티 시스템과 통합할 수 있습니다. 다음은 CloudWatch Events와 통합되는 AWS DevOps 관련 서비스입니다.

- Application Auto Scaling 이벤트
- CodeBuild 이벤트
- CodeCommit 이벤트
- CodeDeploy 이벤트
- CodePipeline 이벤트

Amazon EventBridge

Amazon CloudWatch Events와 EventBridge는 기본 서비스 및 API가 동일하지만 EventBridge가 더 많은 기능을 제공합니다.

Amazon EventBridge는 AWS 서비스, 서비스형 소프트웨어(SaaS) 및 애플리케이션을 통합할 수 있는 서버리스 이벤트 버스입니다. 이벤트 기반 애플리케이션을 구축하는 것 외에도 EventBridge를 사용하면 CodeBuild, CodeDeploy, CodePipeline 및 CodeCommit과 같은 서비스의 이벤트에 대해 알릴 수 있습니다.

AWS CloudTrail

협업, 커뮤니케이션 및 투명성이라는 DevOps 원칙을 수용하려면 누가 인프라를 수정하는지 이해하는 것이 중요합니다. AWS에서는 이러한 투명성이 AWS CloudTrail 서비스에 의해 제공됩니다. 모든 AWS 상호 작용은 AWS API 호출을 통해 처리되며 AWS CloudTrail은 이러한 호출을 모니터링하고 기록합니다. 생성된 모든 로그 파일은 사용자가 정의한 Amazon S3 버킷에 저장됩니다. 로그 파일은 Amazon S3 서버 측 암호화(SSE)를 사용하여 암호화됩니다. 모든 API 호출은 사용자가 직접 호출했는지 또는 AWS 서비스에서 사용자를 대신하여 호출되었는지에 관계없이 기록됩니다. 지원을 위한 운영 팀, 거버넌스를 위한 보안 팀, 결제를 위한 재무 팀 등 수많은 그룹이 CloudTrail 로그를 활용할 수 있습니다.

커뮤니케이션 및 협업

조직에서 DevOps 문화를 도입하든 DevOps 문화 혁신 커뮤니케이션을 진행하든 협업은 접근 방식에서 중요한 부분입니다. Amazon은 팀의 사고방식을 바꿔야 할 필요가 있다는 점을 깨달았고, 이에 따라 피자 두 판 팀이라는 개념을 도입했습니다.

주제

- 피자 두 판 팀

피자 두 판 팀

"우리는 피자 두 판으로 해결할 수 있는 규모의 팀을 만들려고 합니다."라고 Bezos는 말했습니다. "우리는 이것을 피자 두 판 팀 규칙이라고 부릅니다."

팀의 규모가 작을수록 협업의 효율성이 높습니다. 소프트웨어 릴리스가 그 어느 때보다 빠르게 진행되고 있기 때문에 협업도 매우 중요합니다. 또한 소프트웨어를 제공하는 팀의 능력은 경쟁업체와 차별화되는 요소가 될 수 있습니다. 새로운 제품 기능을 출시해야 하거나 버그를 수정해야 하며 출시 기간을 단축하기 위해 이를 최대한 빨리 해결해야 할 상황에 놓여 있다고 상상해 보세요. 변화의 물결이 곧바로 영향을 미치기 시작하는 애자일 접근 방식이 아니라 혁신이 느리게 진행되는 프로세스를 원하지 않기 때문에도 협업은 중요합니다.

팀 간의 커뮤니케이션도 중요합니다. AWS는 공동 책임 모델로 나아가고 사일로화된 개발 접근 방식에서 벗어나기 시작했습니다. 이는 팀에 주인 의식을 부여하고 이를 전체적으로 바라보도록 팀의 관점을 바꿉니다. 팀은 프로덕션 환경을 가시성이 없는 블랙박스로 생각해서는 안 됩니다.

공통 DevOps 팀을 구성하거나 팀에 DevOps에 주력하는 구성원을 한 명 이상 배치하는 다른 접근 방식도 사용할 수 있으므로 문화 혁신도 중요합니다. 이 두 가지 접근 방식 모두 팀에 공동 책임의 개념을 가져옵니다.

보안

DevOps 혁신을 진행 중이든 DevOps 원칙을 처음 구현하든 보안은 DevOps 프로세스에 통합해야 하는 요소로 생각해야 합니다. 보안은 구축, 테스트 배포 단계 전반에서 공통 관심 사항이어야 합니다.

AWS 기반 DevOps의 보안에 대해 이야기하기 전에 AWS 공동 책임 모델을 살펴보겠습니다.

주제

- AWS 공동 책임 모델
- ID 및 액세스 관리

AWS 공동 책임 모델

보안은 AWS와 고객의 공동 책임입니다. 다음은 공동 책임 모델의 여러 부분에 대한 설명입니다.

- AWS 책임 '클라우드의 보안' - AWS는 AWS 클라우드에서 제공되는 모든 서비스를 실행하는 인프라를 보호할 책임이 있습니다. 이 인프라는 AWS 클라우드 서비스를 실행하는 하드웨어, 소프트웨어, 네트워킹 및 시설로 구성됩니다.
- 고객 책임 '클라우드에서의 보안' – 고객 책임은 고객이 선택하는 AWS 클라우드 서비스에 따라 달라집니다. 서비스에 따라 보안 책임의 일환으로서 고객이 수행해야 할 구성 작업의 양이 달라집니다.

이 공유 모델은 고객의 운영 부담을 더는 데 도움이 될 수 있습니다. 호스트 운영 체제 및 가상화 계층부터 서비스 운영 시설의 물리적 보안에 이르는 구성 요소를 AWS에서 운영, 관리 및 통제하기 때문입니다. 이는 고객이 구축 환경의 보안을 이해하고자 하는 경우에 매우 중요합니다.

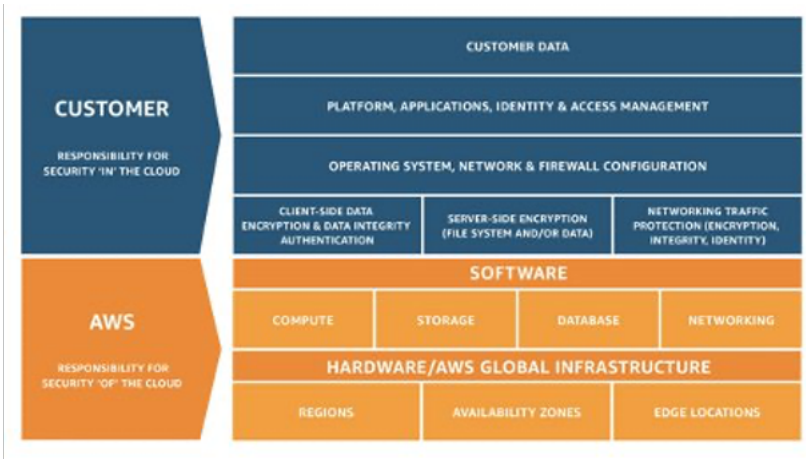

그림 3 - AWS 공동 책임 모델

ID 및 액세스 관리

AWS Identity and Access Management(IAM)는 AWS 리소스에 대한 액세스를 관리하는 데 사용되는 제어 및 정책을 정의합니다. IAM을 사용하면 사용자 및 그룹을 생성하고 다양한 DevOps 서비스에 대한 권한을 정의할 수 있습니다.

사용자 외에도 다양한 서비스에서 AWS 리소스에 액세스해야 할 수도 있습니다. 예를 들어 CodeBuild 프로젝트는 Amazon Elastic Container Registry(Amazon ECR)에 Docker 이미지를 저장하기 위한 권한과 Amazon ECR에 쓰기 위한 권한이 필요할 수 있습니다. 이러한 유형의 권한은 서비스 역할이라고 하는 특수한 유형의 역할에 의해 정의됩니다.

IAM은 AWS 보안 인프라의 구성 요소 중 하나입니다. IAM을 사용하면 그룹, 사용자 및 서비스 역할은 물론 사용자가 어떤 AWS 서비스와 리소스에 액세스할 수 있는지를 제어하는 암호, 액세스 키 및 사용 권한 정책과 같은 보안 자격 증명을 한 곳에서 관리할 수 있습니다. IAM 정책을 사용하면 권한 집합을 정의할 수 있습니다. 그런 다음 이 정책을 역할, 사용자 또는 서비스에 연결하여 권한을 정의할 수 있습니다. IAM을 사용하여 원하는 DevOps 전략 내에서 광범위하게 사용되는 역할을 생성할 수도 있습니다. 경우에 따라 권한을 직접 가져오는 대신 프로그래밍 방식으로 AssumeRole을 사용하는 것이 적절할 수 있습니다. 서비스 또는 사용자가 역할을 수임하면 일반적으로 액세스 권한이 없는 서비스에 액세스할 수 있는 임시 자격 증명이 제공됩니다.

결론

클라우드로의 여정을 원활하고 효율적이며 효과적으로 만들기 위해 기술 기업은 DevOps 원칙과 사례를 수용해야 합니다. 이러한 원칙은 AWS 플랫폼에 기본적으로 스며들어 있습니다. 실제로 이러한 원칙은 수많은 AWS 서비스, 특히 배포 및 모니터링 제품에 포함된 서비스의 초석이 됩니다.

먼저 AWS CloudFormation 또는 AWS Cloud Development Kit (AWS CDK) 서비스를 사용하여 코드형 인프라를 정의합니다. 그런 다음 AWS CodeBuild, AWS CodeDeploy, AWS CodePipeline 및 AWS CodeCommit과 같은 서비스를 사용하여 애플리케이션에서 지속적 배포를 사용하는 방식을 정의합니다. 애플리케이션 수준에서 AWS Elastic Beanstalk, Amazon Elastic Container Service(Amazon ECS) 또는 Amazon Elastic Kubernetes Service(Amazon EKS)와 같은 컨테이너 서비스와 AWS OpsWorks를 사용하여 공통 아키텍처의 구성을 간소화합니다. 이러한 서비스를 사용하면 Auto Scaling 및 Elastic Load Balancing과 같은 다른 중요한 서비스를 쉽게 포함할 수 있습니다. 마지막으로 Amazon CloudWatch와 같은 DevOps 모니터링 전략과 AWS IAM과 같은 견고한 보안 사례를 사용합니다.

AWS를 파트너로 하는 DevOps 원칙으로 비즈니스와 IT 조직에 민첩성을 가져오고 클라우드로의 여정을 가속화합니다.

문서 개정

이 백서의 업데이트에 대한 알림을 받으려면 RSS 피드를 구독하세요.

업데이트 기록-변경	update-history-description	update-history-date
누락된 기여자 단원을 복원했습니다.	누락된 기여자 단원 및 사소한 텍스트 변경 사항을 복원했습니다.	2020년 11월 21일
새 서비스를 포함하도록 단원을 업데이트했습니다.	새 서비스를 포함하도록 단원을 업데이트했습니다.	2020년 10월 16
최초 게시	처음 게시된 백서	2014년 12월 1일

기여자

이 문서를 작성하는 데 도움을 주신 분들입니다.

- Muhammad Mansoor, 솔루션스 아키텍트
- Ajit Zadgaonkar, 현대화 부문 World Wide 기술 책임자
- Juan Lamadrid - 솔루션스 아키텍트
- Darren Ball - 솔루션스 아키텍트
- Rajeswari Malladi - 솔루션스 아키텍트
- Pallavi Nargund - 솔루션스 아키텍트
- Bert Zahniser - 솔루션스 아키텍트
- Abdullahi Olaoye - 클라우드 솔루션스 아키텍트
- Mohamed Kiswani - 소프트웨어 개발 관리자
- Tara McCann – 관리자 솔루션스 아키텍트

고지 사항

고객은 본 문서에 포함된 정보를 독자적으로 평가할 책임이 있습니다. 본 문서는 (a) 정보 제공만을 위한 것이며, (b) 사전 고지 없이 변경될 수 있는 현재의 AWS 제품 제공 서비스 및 사례를 보여 주며, (c) AWS 및 자회사, 공급업체 또는 라이선스 제공자로부터 어떠한 약정 또는 보증도 하지 않습니다. AWS 제품 또는 서비스는 명시적이든 묵시적이든 어떠한 종류의 보증, 진술 또는 조건 없이 '있는 그대로' 제공됩니다. 고객에 대한 AWS의 책임과 법적 책임은 AWS 계약서에 준하며 본 문서는 AWS와 고객 간의 계약에 포함되지 않고 계약을 변경하지도 않습니다.