Introduction to DevOps on AWS

David Chapman

2014년 12월

목차

개요

혁신이 가속화되고 고객이 급속한 발전을 요구함에 따라 기업의 대응 능력도 그만큼 민첩해야 합니다. 따라서 시장 진입 시간은 매우 중요하며, 전반적인 비즈니스 목표를 실현하기 위해서는 IT 부서의 민첩한 대응 능력이 필수적으로 요구됩니다. 소프트웨어 개발 수명 주기는 수년간에 걸쳐 폭포형(Waterfall) 개발 모델에서 Agile 개발 모델로 바뀌었습니다. 이러한 개선이 DevOps 발전과 더불어 IT 운영으로 확산되고 있습니다.

Agile 비즈니스의 요구를 충족하기 위해서는 IT 운영팀에서 일관되고 반복 가능하며 신뢰할 수 있는 방식으로 애플리케이션을 배포해야 합니다. 이는 자동화를 통해서만 완전하게 실현할 수 있습니다.

Amazon Web Services(AWS)는 IT 부서에서 비즈니스의 민첩한 대응 능력을 향상시키기 위해 활용할 수 있는 다양한 DevOps 원칙과 사례를 지원합니다.

이 백서는 DevOps 원칙과 AWS 플랫폼에서 지원되는 사례에 대해 집중적으로 다룹니다. DevOps의 유래에 대한 간단한 소개를 통해 현재의 상황을 정의하고, DevOps가 어떻게 발전해왔는지 그리고 왜 발전해야 하는지에 대해 설명합니다.

소개

DevOps란 기본적으로 소프트웨어 개발자와 IT 운영팀 간의 향상된 협업, 커뮤니케이션 및 통합에 초점을 맞춘 새로운 용어입니다. 이 용어는 철학, 문화 변화 및 패러다임 전환으로 설명되는 포괄적인 용어입니다.

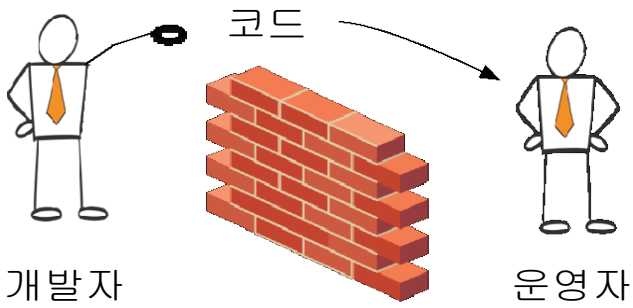

그림 1: "벽 너머로" 코드를 던지는 개발자

많은 조직은 개발, 인프라, 보안 및 지원 팀 간의 통합이 제대로 이루어지지 않는 수직적인 구조 형태를 보여 왔습니다. 흔히 그룹은 회사 목표와 철학이 서로 다른 다양한 조직 구조로 볼 수 있습니다.

소프트웨어 배포는 대체로 IT 운영 그룹의 역할이었습니다. 기본적으로 개발자는 신속하게 소프트웨어를 개발하고 변경하기를 좋아하는 반면, IT 운영팀에서는 안정성과 신뢰성에 중점을 둡니다. 이러한 목표의 불일치는 충돌로 이어지며, 결국 회사에 문제가 될 수 있습니다.

오늘날에는 IT와 개발자 역할이 일련의 조직적 원칙에 따라 통합되면서 이러한 과거이 분할 개념이 무너지고 있습니다.

- 코드형 인프라
- 연속 배포
- 자동화
- 모니터링
- 보안

이러한 각각의 원칙을 살펴보면 Amazon Web Services에서 제공하는 상품과의 밀접한 관계가 드러납니다.

Agile에서 DevOps로의 진화

DevOps 원칙을 제대로 인식하기 위해서는 DevOps가 발전해 온 정황을 이해하는 것이 도움이 됩니다. 이 이야기는 10년 전에 우수한 소프트웨어 빌드 방법으로 여겨져 큰 인기를 누렸던 Agile 소프트웨어 개발에서 시작됩니다. Agile 이전에 지배적이었던 폭포형(Waterfall) 개발 방법은 개발할 시스템을 100% 미리 정의해 놓고, 요구 사항 분석 단계에서 시작하는 순차적인 개발 프로세스입니다. 이 방법은 유연하지 못하고 획일적임이 자체적으로 입증되었습니다.

Agile 모델은 비즈니스 사용자와 개발자 간의 새롭고 향상된 협업의 개념을 가져왔습니다. 이 모델은 프로젝트 기간 동안 발전하면서 가치를 제공하는, 작동 소프트웨어의 반복적인 개발에 중점을 두기 시작했습니다. Agile은 경험적으로 제어되는 엔지니어링 프로세스로, 현재 다양한 도구가 이 모델을 지원합니다. 개발자를 위한 이러한 도구에는 IDE, 단위 테스트 프레임워크 및 코드 최적화 도구 등이 있습니다. 개발자의 생산성이 향상될수록 비즈니스의 대응 능력이 민첩해져서 고객 요청에 더 빠르고 효율적으로 대응할 수 있습니다.

그림 2: Agile 소프트웨어 개발과 DevOps의 공진화

지난 몇 년 동안 Agile 소프트웨어 개발의 발전이 *DevOps*라는 이름 아래 인프라 전체로 확산되었습니다. Agile 소프트웨어 개발은 기본적으로 회사와 개발자 간의 협업에 중점을 두는 반면, DevOps는 개발, IT 운영 및 보안 팀 간의 협업에 중점을 둡니다. IT 운영팀에는 시스템 관리자, 데이터베이스 관리자, 네트워크 엔지니어, 인프라 아키텍트 및 지원 담당자가 포함됩니다. Agile 소프트웨어 개발이 비즈니스 대응 능력을 제공하는 반면, DevOps는 IT 대응 능력을 제공하여 보다 안정되고 예측 가능하며 효율적인 애플리케이션을 개발할 수 있게 해줍니다.

DevOps 사례는 작업마다 다릅니다. 애플리케이션 개발의 경우, DevOps는 코드 빌드, 코드 범위, 단위 테스트, 패키지화 및 개발에 중점을 둡니다. 한편, 인프라의 경우에는 DevOps는 프로비저닝, 구성, 조직화 및 개발에 중점을 둡니다. 하지만 각 영역에서 버전 관리, 배포, 롤백, 롤포워드 및 테스트의 기본 원리는 모두 동일합니다.

인프라를 나타내는 코드

DevOps의 기본 원리는 개발자가 코드를 다루는 방식과 동일한 방식으로 인프라를 다루는 것입니다. 애플리케이션 코드에는 형식과 구문이 정의되어 있습니다. 코드가 프로그래밍 언어의 규칙에 따라 작성되지 않으면 애플리케이션을 만들 수 없습니다. 코드는 코드 개발, 변경 내용 및 버그 수정 사항 내역이 기록되는 버전 관리 시스템에 저장됩니다. 코드가 애플리케이션으로 컴파일(빌드)될 때 일관된 애플리케이션이 생성됩니다. 즉, 빌드가 반복적이고 안정적입니다.

"코드형 인프라"를 실현한다는 것은 애플리케이션 코드 개발과 똑같은 엄격성을 인프라 프로비저닝에 적용한다는 뜻입니다. 모든 구성을 선언 방식으로 정의하고 애플리케이션 코드와 똑같이 버전 관리 시스템에 저장해야 합니다. 인프라 프로비저닝, 조직화 및 개발에서 "인프라 코드" 사용을 지원해야 합니다.

최근까지는 애플리케이션 코드 개발에 적용되는 엄격성이 인프라에 반드시 적용될 필요가 없었습니다. 대체로 인프라는 수동 프로세스를 사용하여 프로비저닝됩니다. 프로비저닝 중에 개발된 스크립트는 버전 관리 시스템에 저장하지 못할 수도 있으며 환경 구축의 반복성, 안정성 또는 일관성이 반드시 유지되지는 않습니다.

이와 대조적으로, AWS는 인프라 구축 및 유지 관리를 위한 DevOps 중심의 방법을 제공합니다. 소프트웨어 개발자가 애플리케이션 코드를 작성할 때처럼, AWS는 프로그래밍, 설명 및 선언적인 방식으로 인프라의 구축, 배포 및 유지 관리를 지원하는 서비스를 제공합니다. 이러한 서비스는 엄격성, 명확성 및 안정성을 제공합니다. 이 문서에서 설명한 AWS 서비스는 DevOps 전략의 핵심 요소로, 다양한 상위 수준의 AWS DevOps 원칙 및 사례의 토대를 형성합니다.

AWS CloudFormation

AWS CloudFormation은 DevOps 원칙이 실제로 어떻게 사용되지를 보여 주는 좋은 예입니다.[1] AWS CloudFormation 템플릿을 사용하면 생성 및 업데이트할 수 있는 AWS 리소스를 정의하고 모델링할 수 있습니다. 이러한 템플릿은 JavaScript Object Notation(JSON)이라고 하는 형식으로 작성됩니다. 템플릿에는 생성 및 관리되는 리소스 유형에 따라 달라지는 특정 구문 및 구조가 필요합니다. 템플릿을 사용하면 반복 가능하고 안정적인 방식으로 인프라를 프로비저닝할 수 있습니다.

사용자 지정 AWS CloudFormation 템플릿을 만들거나 공개적으로 사용 가능한 샘플 템플릿을 사용할 수 있습니다. 템플릿을 AWS 환경으로 배포 또는 업데이트한 후, 관리되는 리소스 모음을 “스택”이라고 합니다. AWS Management Console, AWS 명령줄 인터페이스 또는 AWS CloudFormation API를 통해 스택을 관리할 수 있습니다. 일반적인 작업으로는 스택 생성, 스택 설명, 스택 나열 및 스택 업데이트가 있습니다.

콘솔에서 스택을 생성하거나 업데이트할 때 구성 상태를 보여 주는 이벤트가 표시됩니다. 오류가 발생하면 스택이 이전 상태로 롤백됩니다. Amazon Simple Notification Service(Amazon SNS)는 이러한 이벤트를 관리하는 데 도움이 됩니다. 예를 들어 Amazon SNS를 사용하여 이메일을 통해 스택 생성 및 삭제 진행 상황을 추적하고 다른 프로세스와 프로그래밍 방식으로 통합할 수 있습니다.

Amazon Simple Storage Service(Amazon S3), Auto Scaling, Amazon CloudFront, Amazon DynamoDB, Amazon Elastic Compute Cloud(EC2), Amazon ElastiCache, AWS Elastic Beanstalk, Elastic Load Balancing, AWS Identity and Access Management, AWS OpsWorks, Amazon Virtual Private Cloud 등을 비롯한 광범위한 AWS 상품에서 템플릿을 사용하여 작업할 수 있습니다.

[1] http://aws.amazon.com/cloudformation

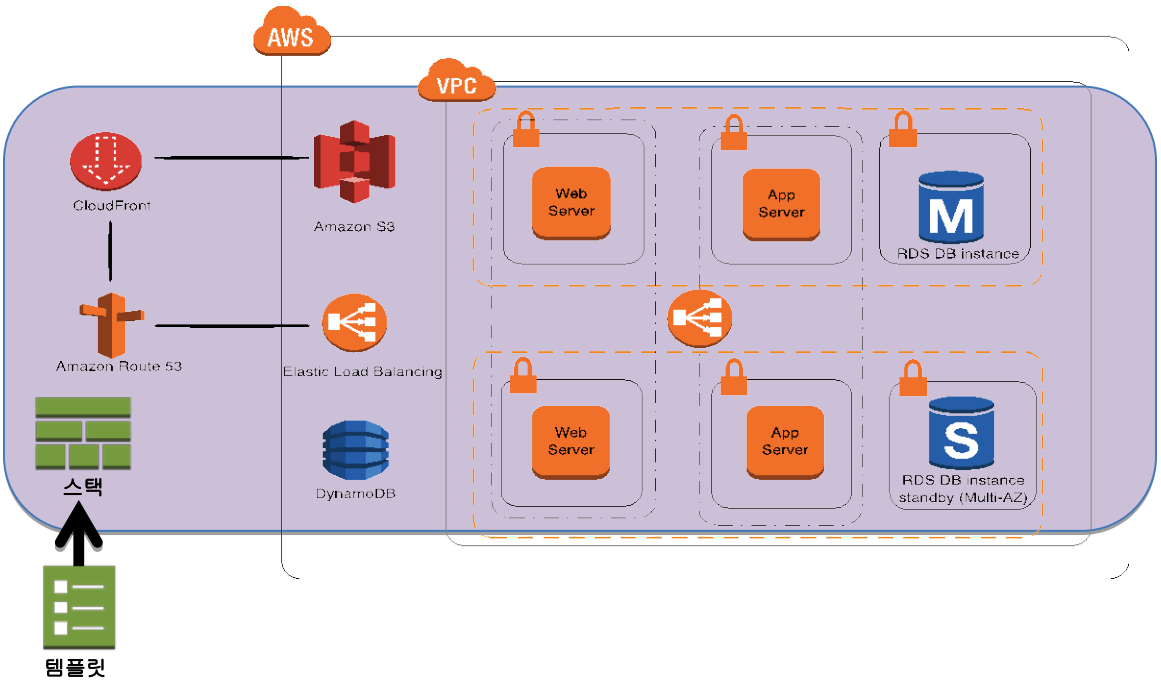

그림 3: AWS CloudFormation 예 1 — 템플릿 하나에서 전체 환경(스택) 구축

단일 템플릿을 사용하여 전체 환경을 구축 및 업데이트하거나 여러 개별 템플릿을 사용하여 환경 내 계층을 관리할 수 있습니다. 따라서 템플릿을 모듈화할 수 있으며 여러 조직에 중요한 거버넌스 계층도 제공되는 셈입니다.

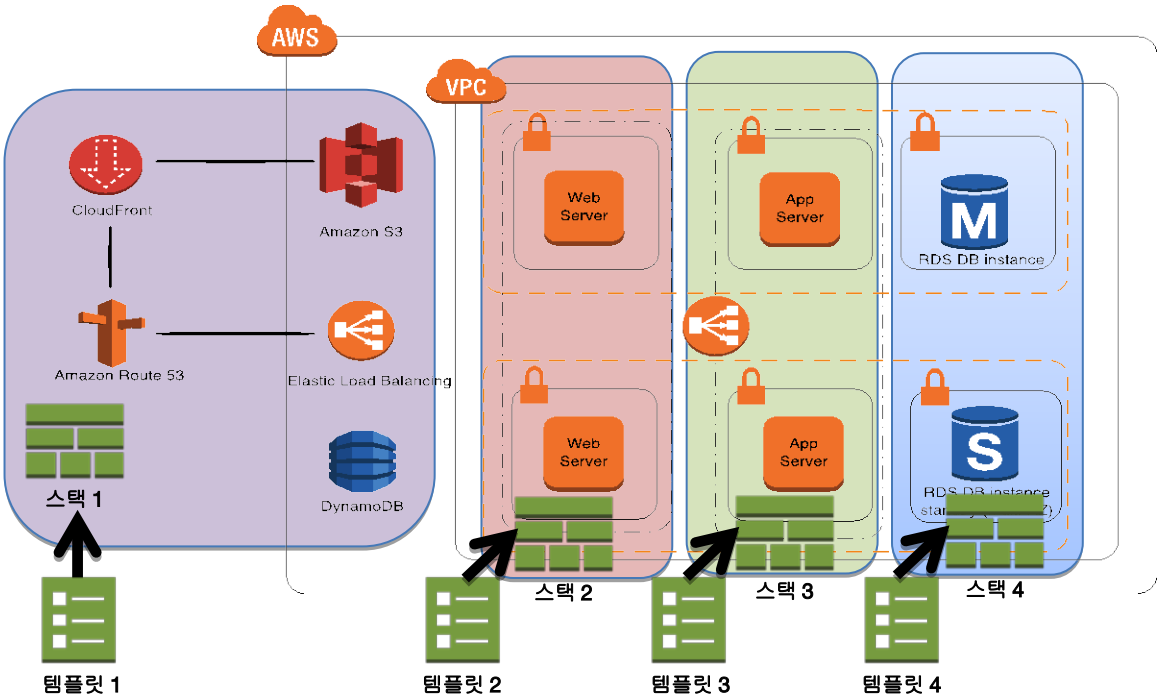

그림 4: AWS CloudFormation의 예 2 - 계층화된 방식으로 여러 템플릿에서 전체 환경(스택) 구축

AWS CloudFormation을 사용하면 AWS 리소스 모음을 쉽게 구성 및 배포할 수 있으며 스택이 구성될 때 특수 파라미터를 전달하거나 종속성을 설명할 수 있습니다.

AWS CloudFormation의 "코드형 정보(information as code)" 능력을 실현하려면 AWS에서 템플릿을 배포하거나 업데이트하기 전에 소스 코드 관리 시스템의 버전 관리 부분에 템플릿을 저장해야 합니다. Amazon S3은 템플릿을 저장하고 버전을 관리하기에 적합한 위치를 제공합니다. AWS CloudFormation을 원하는 배포 및 관리 도구와 통합할 수 있습니다.

AWS CloudFormation 서비스에서 "코드형 인프라(infrastructure as code)" 정의에는 별도의 요금이 청구되지 않습니다. AWS CloudFormation에서 생성되어 애플리케이션에서 사용된 AWS 리소스에 대한 일반 요금만 청구됩니다.

```
{
        "Description" : "Create an EC2 instance running the Amazon Linux 32 bit AMI.",
        "Parameters" : {
                "KeyPair" : {
                        "Description" : "The EC2 Key Pair to allow SSH access to the instance",
                        "Type" : "String"
                }
        },
        "Resources" : {
                "Ec2Instance" : {
                        "Type" : "AWS::EC2::Instance",
                        "Properties" : {
                                "KeyName" : { "Ref" : "KeyPair" },
                                "ImageId" : "ami-75g0061f",
                                "InstanceType" : "m1.medium"
                        }
                }
        },
        "Outputs" : {
                "InstanceId" : {
                        "Description" : "The InstanceId of the newly created EC2 instance",
                        "Value" : { "Ref" : "Ec2Instance" }
                }
        }
}
```

그림 5: EC2 인스턴스를 시작하기 위한 AWS CloudFormation 템플릿 예

위의 예에서 AWS CloudFormation 템플릿은 Amazon EC2 인스턴스를 생성하기 위해 JSON 형식으로 정의되었습니다. 이 사례에서는 m1.medium 유형의 EC2 인스턴스를 프로비저닝합니다. `Ref`라는 용어는 스택이 생성될 때 액세스되는 파라미터를 의미합니다. `KeyPair`라고 하는 파라미터는 스택을 생성할 때 제공할 수 있습니다. 이 이름은 SSH를 사용하여 인스턴스에 액세스할 때 사용되는 키 페어의 이름입니다.

AWS AMI

Amazon 머신 이미지(AMI)는 "코드형 인프라"를 보여 주는 또 하나의 예입니다. AWS 컴퓨팅의 이 핵심 구성 요소는 클라우드에서 기본 AWS 컴퓨팅 환경인 Amazon EC2 인스턴스를 시작(프로비저닝)할 수 있는 일종의 디지털 템플릿입니다. 이 이미지에는 웹 서버, 애플리케이션 서버 및 데이터베이스 등 소프트웨어 구성에 포함되어 있습니다.

세 가지 AMI 유형 중에서 선택할 수 있습니다.

- AWS에서 발행한 AMI
- 타사 AMI
- 사용자 지정하여 만든 AMI

AWS는 Linux나 Microsoft Windows 같은 인기 운영 체제에 기초한 일반적인 소프트웨어 구성을 포함하는 AMI를 게시합니다. AMI는 타사 공급업체에서 구할 수도 있으며, 일부 AMI는 AWS Marketplace에서 구입할 수 있습니다.[2] 조직에서 고유의 사용자 지정 AMI를 생성하여 게시할 수도 있습니다. 사용자 지정 조직 AMI에는 일반적으로 강화된 운영 체제, 안티바이러스 소프트웨어 및 Office 생산성 제품군 등 회사 차원에서 배포된 소프트웨어가 포함됩니다.

또한 AMI에는 AMI와 함께 번들화된 애플리케이션 소프트웨어가 포함되거나, 인스턴스에서 시작 시 애플리케이션 소프트웨어를 설치할 수 있도록 하는 "부팅" 스크립트 및 소프트웨어가 포함될 수 있습니다. AMI로 애플리케이션 수준의 소프트웨어를 미리 로드하는 것에는 장단점이 있습니다. 장점으로는, 부팅 시 추가 소프트웨어를 설치할 필요가 없으므로 인스턴스를 매우 빨리 시작할 수 있습니다. 하지만 애플리케이션 수준의 소프트웨어가 변경될 때마다 새 AMI를 생성해야 할 수도 있습니다.

지속적인 개발

연속 배포는 DevOps 전략의 또 한 가지 핵심 개념입니다. 기본 목표는 프로덕션 가능한 애플리케이션 코드의 자동화된 개발을 실현하는 것입니다.

경우에 따라서는 연속 배포가 연속 전송을 의미하기도 합니다. 단지 연속 배포가 일반적으로 프로덕션 배포를 의미한다는 점만 다릅니다.

연속 전송 사례 및 도구를 사용하면 소프트웨어를 반복적 및 안정적으로 빠르게 배포할 수 있습니다. 배포에 실패할 경우 이전 버전으로 자동 롤백할 수 있습니다.

[2] https://aws.amazon.com/marketplace

AWS CodeDeploy

AWS에서 이 원리를 보여 주는 가장 좋은 예는 코드 배포 서비스인 AWS CodeDeploy입니다.[3] 이 서비스의 핵심 기능은 관리를 중앙화하며 기존 소프트웨어 또는 연속 전송 프로세스와 통합하여 최소 중단 시간으로 Amazon E2C 집합 전체에 애플리케이션을 배포할 수 있는 기능을 제공합니다.

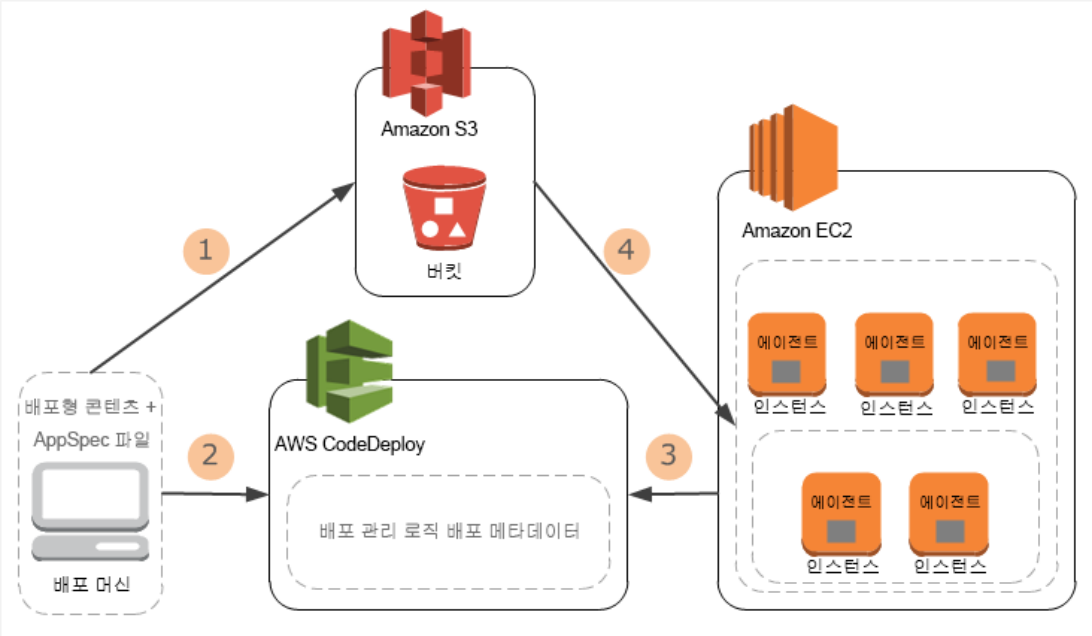

그림 6: AWS CodeDeploy 프로세스

운영 방식은 다음과 같습니다.

1. 애플리케이션 콘텐츠는 패키지로 압축되어 AWS CodeDeploy에서 실행해야 하는 일련의 배포 단계를 정의한 AppSpec(Application Specific) 파일과 함께 Amazon S3로 배포됩니다. 이 패키지를 CodeDeploy "개정"이라고 합니다.
2. AWS CodeDeploy에서 애플리케이션을 생성하고 애플리케이션을 배포해야 할 인스턴스를 정의할 수 있습니다(DeploymentGroup). 또한 애플리케이션은 배포 패키지가 상주하는 Amazon S3 버킷을 정의합니다.
3. AWS CodeDeploy 에이전트는 각각의 참여 Amazon EC2에 배포됩니다. 에이전트는 AWS CodeDeploy를 폴링하여 지정된 Amazon S3 버킷에서 개정을 가져와야 하는 시기와 가져올 개정을 결정합니다.
4. AWS CodeDeploy 에이전트는 패키지화된 애플리케이션 코드를 가져와서 인스턴스에 배포합니다. 배포 지침이 들어 있는 AppSec 파일도 다운로드됩니다.

[3] http://aws.amazon.com/codedeploy

이와 같이, AWS CodeDeploy는 DevOps의 중심이 되는 연속 자동화 배포를 보여 주는 전형적인 예입니다.

AWS CodePipeline

AWS CodeDeploy와 마찬가지로, AWS CodePipeline(2015년에 사용 가능)은 원활한 배포를 돕는 연속 전송 및 방출 자동화 서비스입니다.[4] 코드를 체크 인하여 구축하고 애플리케이션을 준비 단계로 배포하고 테스트하며 출시하는 개발 워크플로우를 설계할 수 있습니다. 출시 프로세스의 어느 단계에서든 타사 도구를 통합할 수 있으며 AWS CodePipeline을 엔드 투 엔드 솔루션으로도 사용할 수 있습니다. AWS CodePipeline을 사용하면 구축, 테스트, 출시 프로세스를 자동화하여 높은 품질의 기능 및 업데이트를 신속하게 전송할 수 있습니다.

AWS CodePipeline는 DevOps 연속 배포 원리에 맞는 여러 이점을 제공합니다.

- 빠른 전송
- 품질 향상
- 구성 가능한 워크플로우
- 손쉬운 통합

AWS CodeCommit

또한 2015년에는 프라이빗 Git 저장소를 호스팅하는 안전하고 확장 가능한 소스 관리형 서비스인 AWS CodeCommit 서비스도 실시될 예정입니다.[5] CodeCommit를 사용하면 자체 소스 제어 시스템을 운영하거나 인프라 조정을 염려할 필요가 없습니다. 코드부터 바이너리까지 모든 것을 저장할 수 있으며, Git의 표준 기능을 지원하여 기존 Git 기반 도구와도 원활하게 연동됩니다. 또한, 팀에서 CodeCommit의 온라인 코드 도구를 사용하여 프로젝트를 찾아보고 수정 및 공동 작업할 수 있습니다.

AWS CodeCommit은 여러 이점을 제공합니다.

- 종합 관리형
- 무엇이든 저장 가능
- 고가용성
- 보다 빠른 개발 수명 주기 제공
- 기존 도구와 작동
- 보안

---

[4] http://aws.amazon.com/codepipeline

[5] http://aws.amazon.com/codecommit

AWS Elastic Beanstalk 및 AWS OpsWorks

AWS Elastic Beanstalk[6]와 AWS OpsWorks[7] 모두 애플리케이션 코드 변경 사항 및 인프라 수정 사항의 연속 배포를 지원합니다. AWS Elastic Beanstalk에서 코드 변경 배포는 "애플리케이션 버전"으로 저장되며 인프라 변경은 "저장된 구성"으로 배포됩니다. AWS OpsWorks에는 고유한 애플리케이션 배포 프로세스가 있으며 추가 런타임 시작 명령과 Chef 레시피를 정의할 수 있습니다.

애플리케이션 버전 예는 `.zip` 또는 `.war` 파일로 업로드하는 새 Java 애플리케이션입니다. 저장된 구성의 예는 단일 인스턴스가 아닌 Elastic Load Balancing 및 Auto Scaling을 사용하는 AWS Elastic Beanstalk 구성입니다. 변경이 완료되면 새 구성을 저장할 수 있습니다.

AWS Elastic Beanstalk눈 "배포 롤링"이라고 하는 DevOps 사례를 지원합니다. 이 방법이 활성화된 경우 구성 배포가 Auto Scaling과 함께 작동하므로 구성이 변경될 때 정의된 사용 가능 인스턴스 수가 항상 유지됩니다. 따라서 Amazon EC2 인스턴스가 업데이트될 때 사용자에게 제어 기능이 제공됩니다. 예를 들어 EC2 인스턴스 유형을 변경되는 경우 AWS Elastic Beanstalk에서 모든 인스턴스를 동시에 업데이트할지 아니면 다른 인스턴스가 업데이트될 때 요청을 이행하기 위해 일부 인스턴스의 실행 상태를 유지할지 여부를 결정할 수 있습니다.

이와 마찬가지로, AWS OpsWorks는 배포될 때 업데이트해야 할 계층과 인스턴스를 정의할 수 있는 옵션도 제공합니다.

AWS Elastic Beanstalk 및 AWS OpsWorks의 추가 기능은 자동화 섹션에 설명되어 있습니다.

블루-그린(Blue-Green) 배포

Blue-green 배포는 도메인 이름 서비스(DNS)를 사용하여 애플리케이션을 배포하는 DevOps 배포 사례입니다. 이 전략에서는 기존(블루) 환경으로 시작하면서 동시에 새(그린) 환경을 테스트해야 합니다. 새 환경이 모든 필수 테스트를 통과했으며 가동 준비가 완료된 경우, 간단히 DNS를 통해 이전 환경에서 새 환경으로 트래픽을 리디렉션하기만 하면 됩니다.

AWS는 블루-그린 개발 전략을 구현하는 데 필요한 도구를 모두 제공합니다. AWS CloudFormation 또는 AWS Elastic Beanstalk 같은 서비스를 사용하여 이상적인 새 인프라 환경을 구성할 수 있습니다. AWS CloudFormation 템플릿을 사용하면 기존 프로덕션 환경과 동일한 새 환경을 쉽게 구축할 수 있습니다.

---

[6] http://aws.amazon.com/elasticbeanstalk

[7] http://aws.amazon.com/opsworks

AWS DNS 서비스인 Amazon Route 53을 사용할 경우 가중치를 할당한 리소스 레코드 세트를 사용하여 트래픽 흐름을 라우팅할 수 있습니다. 이러한 레코드 세트를 사용하면 DNS 확인을 통해 여러 서비스나 로드 밸런서를 정의할 수 있습니다.

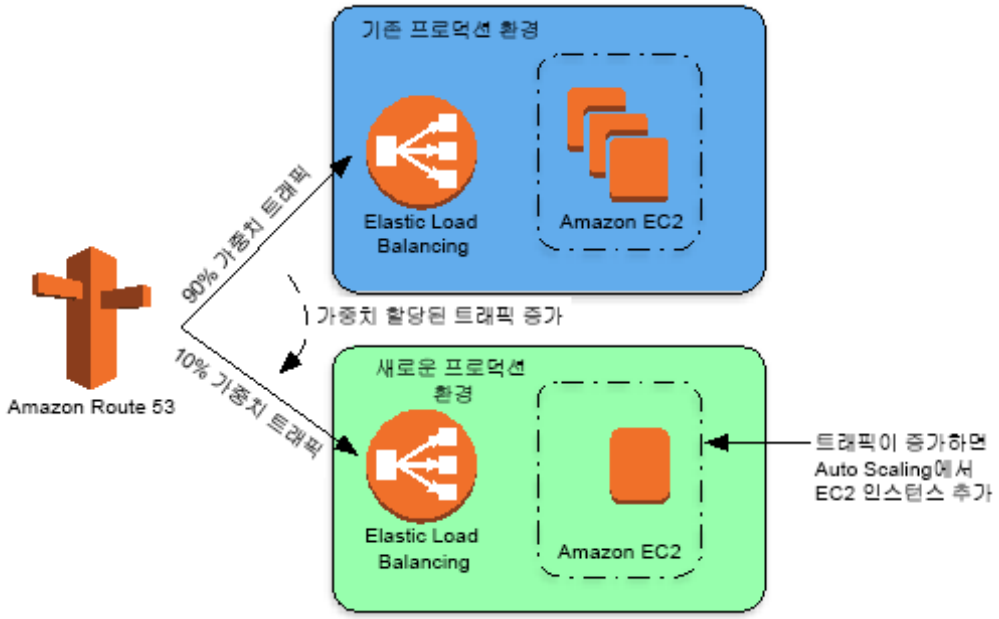

그림 7: Amazon Route 53 가중치 기반의 리소스 레코드 세트를 사용한 블루-그린 배포

DNS 서비스 리졸루션(도메인 이름을 IP 주소로 변환)은 가중치 기반으로, 새로 배포한 프로덕션 환경으로 라우팅되는 트래픽의 양을 정의할 수 있습니다. 이 기능을 사용하면 환경을 테스트한 후, 배포가 적절하다고 확신될 경우 가중치를 높일 수 있습니다. 이전 프로덕션 환경의 트래픽 수신율이 0%인 경우 이 환경을 백업 목적으로 유지하거나 폐기할 수 있습니다. 새 환경의 트래픽 양이 증가하면 Auto Scaling을 사용하여 추가 Amazon EC2 인스턴스를 확장할 수 있습니다.

AWS 클라우드에서 동일한 환경을 쉽게 구축 및 폐기할 수 있는 이 기능을 통해 블루-그린 배포 같은 DevOps 사례가 실현 가능합니다.

또한 데이터베이스 배포 및 장애 조치 같은 백엔드 서비스에도 블루-그린 배포를 사용할 수 있습니다.

자동화

DevOps의 또 한 가지 핵심 원칙과 사례는 자동화입니다. 자동화는 인프라와 인프라에서 실행되는 애플리케이션의 설정, 구성, 배포 및 지원에 중점을 둡니다. 자동화를 사용하여 표준화되고 반복 가능한 방식으로 환경을 매우 신속하게 설정할 수 있습니다. 성공적인 DevOps 전략의 핵심은 수동 프로세스를 제거하는 것입니다. 지금까지 서버 구성 및 애플리케이션 배포는 주로 수동 프로세스로 이루어져 왔습니다. 환경이 표준화되지 않으면 문제 발생 시 환경을 재현하기가 어렵습니다.

자동화의 사용은 클라우드의 모든 이점을 실현하는 데 있어서 매우 중요합니다. AWS는 내부적으로 자동화에 의존하여 탄력성과 확장성의 핵심 기능을 제공합니다. 수동 프로세스는 오류가 발생하기 쉽고, 신뢰성이 떨어지며, 민첩한 대응 능력을 필요로 하는 비즈니스를 지원하기에 적합하지 않습니다. 흔히 조직의 고급 기술 인력이 수동 구성을 제공하는 업무에 얽매여 있기도 합니다. 이러한 시간을 회사에서 보다 중요하고 더 높은 가치를 창출하는 다른 활동을 지원하는 데 쓰면 더 좋을 것입니다.

현대의 운영 환경은 일반적으로 전체 자동화에 의존하여 수동 개입을 제거하거나 프로덕션 환경에 액세스합니다. 여기에는 모든 소프트웨어 릴리스, 머신 구성, 운영 체제 패치, 문제 해결 또는 버그 수정이 포함됩니다. 자동화 사례의 다양한 수준을 함께 사용하면 보다 높은 수준의 완전 자동화 프로세스를 제공할 수 있습니다.

자동화는 많은 이점을 제공합니다.

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

자동화는 AWS 서비스의 기초로, 모든 서비스, 기능 및 상품에서 내부적으로 지원됩니다.

AWS Elastic Beanstalk

AWS의 자동화 예로, AWS Elastic Beanstalk 하나면 충분합니다. AWS Elastic Beanstalk는 개발자가 일반적으로 사용되는 기술 스택에 애플리케이션을 쉽고 생산적으로 배포할 수 있게 해주는 서비스입니다. 인터페이스가 간단하고 쉬워서 다계층 애플리케이션을 빠르고 쉽게 배포할 수 있습니다. AWS Elastic Beanstalk는 자동화는 물론, 자동화된 애플리케이션 배포, 모니터링, 인프라 구성 및 버전 관리 등 다양한 다른 DevOps 모범 사례를 지원합니다. 애플리케이션과 인프라의 변경을 쉽게 롤백하고 전달할 수 있습니다.

환경 구축은 AWS Elastic Beanstalk 자동화를 보여 주는 좋은 예입니다. 환경에 대한 세부 정보만 지정하면 AWS Beanstalk에서 자체적으로 모든 구성과 프로비저닝 작업을 수행합니다. 예를 들어 다음은 애플리케이션 생성 마법사에서 지정할 수 있는 몇 가지 옵션입니다.

- 웹 서비스 계층(웹 서버와 애플리케이션 서버 포함)을 사용할지 아니면 작업자 계층(Amazon Simple Queue Service 사용)을 사용할지 여부
- 애플리케이션에 사용할 플랫폼. IIS, Node.js, PHP, Python, Ruby, Tomcat 또는 Docker 중에서 선택할 수 있습니다.
- 단일 인스턴스를 시작할지 아니면 로드 밸런싱 자동 조정 환경을 구축할지 여부
- 환경에 자동으로 배정할 URL
- 환경에 Amazon Relational Database 인스턴스를 포함할지 여부
- Amazon Virtual Private Cloud 내부에서 환경을 구축할지 여부
- 애플리케이션에 대한 자동 상태 확인에 사용할 URL(있는 경우)
- 환경을 식별하기 위해 적용할 태그(있는 경우)

또한 AWS Elastic Beanstalk는 자동화를 사용하여 애플리케이션을 배포합니다. 플랫폼에 따라, `.war` 또는 `.zip` 파일 형태의 패키지를 컴퓨터나 Amazon S3에서 직접 업로드하는 것만으로도 애플리케이션을 배포할 수 있습니다.

AWS Elastic Beanstalk는 환경이 구축되는 동안 진행 상황에 대한 피드백과 시작 상태를 알려주는 이벤트를 관리 콘솔에 자동으로 기록합니다. 완료되고 나면 정의된 URL을 사용하여 애플리케이션에 액세스할 수 있습니다.

애플리케이션 및 기술 스택의 특정 측면을 제어하려는 경우 AWS Elastic Beanstalk를 그에 맞게 사용자 지정할 수 있습니다.

AWS OpsWorks

AWS OpsWorks는 AWS Elastic Beanstalk보다 더 세부적으로 DevOps의 원칙을 적용합니다. AWS OpsWorks는 구성 관리 소프트웨어(Chef) 및 애플리케이션 수명 주기 관리와의 통합 같은 추가 기능과 함께 훨씬 더 다양한 자동화 수준을 제공합니다. 애플리케이션 수명 주기 관리를 사용하여 리소스가 설정, 구성, 배포, 배포 해제 또는 종료되는 시점을 정의할 수 있습니다.

AWS OpsWorks는 구성 가능한 스택에서 애플리케이션을 정의할 수 있는 추가 유연성도 제공합니다. 또한 미리 정의된 애플리케이션 스택을 선택할 수도 있습니다. 애플리케이션 스택에는 애플리케이션 서버, 웹 서버, 데이터베이스 및 로드 밸런서 등 애플리케이션에 필요한 AWS 리소스에 대한 모든 프로비저닝이 포함되어 있습니다.

그림 8: DevOps 기능 및 아키텍처를 보여 주는 AWS OpsWorks

애플리케이션 스택은 스택을 독립적으로 유지 관리할 수 있도록 아키텍처 계층으로 구성됩니다. 계층의 예로는 웹 계층, 애플리케이션 계층 및 데이터베이스 계층 등을 들 수 있습니다. 즉시 사용 가능한 AWS OpsWorks는 Auto Scaling 그룹과 Elastic Load Balancing 로드 밸런서 설정을 간소화함으로써 DevOps 자동화 원칙을 확실하게 보여 줍니다. AWS Elastic Beanstalk와 마찬가지로, AWS OpsWorks도 애플리케이션 버전 관리, 연속 배포 및 인프라 구성 관리를 지원합니다.

AWS OpsWorks는 다음 섹션에서 설명하는 DevOps 모니터링 및 로깅 사례도 지원합니다. 모니터링 지원은 Amazon CloudWatch에서 제공됩니다. 모든 수명 주기 이벤트가 기록되며, 실행되는 모든 Chef 레시피가 예외 사항과 함께 개별 Chef 로그 문서에 문서화됩니다.[8]

[8] Chef 레시피는 패키지 및 패치 설치 등 시스템을 구성하기 위해 수행해야 하는 모든 작업을 정의하는 Ruby 애플리케이션입니다.

모니터링

커뮤니케이션과 협업은 DevOps 전략의 기본 요소입니다. 이를 지원하기 위해서는 피드백이 중요합니다. AWS에서는 피드백이 Amazon CloudWatch와 AWS CloudTrail이라는 두 가지 핵심 서비스를 통해 제공됩니다. 이들 두 서비스가 함께 작동하여 인프라에 대한 강력한 모니터링, 경고 및 감사 기능을 제공하므로 개발자와 운영 팀이 서로 긴밀하고 분명하게 협력할 수 있습니다.

Amazon CloudWatch

Amazon CloudWatch는 모든 AWS 리소스와 이러한 리소스에서 실행되는 애플리케이션을 실시간으로 모니터링합니다.[9] 리소스와 애플리케이션은 Amazon CloudWatch에서 수집하고 추적하는 매트릭스를 생성할 수 있습니다. 이벤트가 발생할 때마다 알림을 보내도록 경보를 구성할 수 있습니다. 이메일, Amazon SNS 및 Amazon Simple Queue Service 등 다양한 형식으로 알림을 구성할 수 있습니다. 알림을 개인, 팀 또는 다른 AWS 리소스로 전송할 수 있습니다.

Amazon CloudWatch는 피드백을 제공할 뿐만 아니라, DevOps 자동화 개념도 지원합니다. Auto Scaling 같은 AWS 서비스는 Amazon EC2 인스턴스의 확장/축소와 로드 증가/감소 등과 같이 해당 자동 작업을 트리거하는 알림을 위해 CloudWatch를 사용합니다.

그림 9: Amazon CloudWatch 및 Auto Scaling을 사용한 DevOps 자동화의 예

[9] http://aws.amazon.com/cloudwatch

위의 예에서 Amazon CloudWatch는 Elastic Load Balancing에서 지연 시간 측정치를 모니터링하고, 실행 중인 Amazon EC2 인스턴스에서 평균 CPU 측정치를 모니터링합니다. 지연 시간 측정치는 Amazon EC2 인스턴스에 대한 요청이 발생한 후 응답이 전송되는 데 걸리는 시간을 측정합니다. 정의된 임계값을 벗어나서 트리거되는 경보에 대해 조치를 취하는 조정 정책을 만들 수 있습니다. 이러한 정책을 통해 상황에 따라 Amazon EC2 인스턴스 수가 증가하거나 감소할 수 있습니다. 추가 알림을 정의하여 Amazon SNS를 통해 메시지를 전송할 수도 있습니다. 이 기능은 지원팀과 같은 관계자에게 이벤트가 발생했음을 알릴 때 유용합니다.

Auto Scaling의 예는 DevOps 전략을 수용하는 데 있어 핵심이 되는, 투명하고 자동화된 서비스를 제공하기 위해 AWS 서비스가 어떻게 함께 작동하는지를 보여 줍니다. 시스템 관리자와 지원 팀은 다른 부가 가치 비즈니스 요구에 초점을 맞출 수 있으며 AWS 인프라에서 애플리케이션 조정 요구 사항이 잘 처리되고 있음을 확신할 수 있습니다. 이 시나리오에서는 해당 애플리케이션이 클라우드에 최적화되었으며 수평 확장 가능한 방식으로 디자인되어 Auto Scaling의 이점을 활용할 수 있다고 가정합니다.

AWS CloudTrail

협력, 커뮤니케이션 및 투명성에 대한 DevOps 원칙을 적용하기 위해서는 누가 인프라를 변경하는지 알아야 합니다. AWS에서는 이러한 투명성이 AWS CloudTrail 서비스를 통해 제공됩니다.[10] 모든 AWS 상호 작용은 AWS CloudTrail에서 모니터링하고 기록하는 AWS API 호출을 통해 처리됩니다. 생성된 모든 로그 파일은 사용자가 정의한 Amazon S3 버킷에 저장됩니다. 로그 파일은 Amazon S3 서버 쪽 암호화(SSE)를 사용하여 암호화됩니다. 사용자로부터 직접 수신되든 사용자를 대신하여 AWS 서비스에 의해 수신되든 모든 API 호출이 기록됩니다. 지원 관련 운영팀, 거버넌스 관련 보안팀 및 결제 관련 회계팀 등과 같은 다양한 팀에서 CloudTrail 로그를 활용할 수 있습니다.

보안

DevOps 지원 환경에서도 여전히 보안을 최우선으로 생각해야 합니다. 인프라와 회사 자산을 보호해야 하며 문제 발생 시 반복적이고 효과적으로 문제를 해결해야 합니다.

[10] http://aws.amazon.com/cloudtrail

Identity and Access Management(IAM)

AWS Identity and Access Management(IAM) 서비스는 AWS 보안 인프라의 구성 요소 중 하나입니다. IAM을 사용하면 사용자가 어떤 AWS 서비스와 리소스에 액세스할 수 있는지를 제어하는 암호, 액세스 키 및 사용 권한 정책과 같은 보안 자격 증명과 사용자를 중앙에서 관리할 수 있습니다. IAM을 사용하여 DevOps 전략 내에서 광범위하게 사용되는 역할을 만들 수도 있습니다. IAM 역할을 사용하면 사용자나 서비스에서 필요로 하는 리소스에 액세스할 수 있는 권한 세트를 정의할 수 있습니다. 하지만 특정 사용자나 그룹에 권한을 연결하는 대신에, 이들 사용자나 그룹을 명명된 역할에 연결합니다. 그런 다음에 리소스를 역할과 연결하고 서비스를 프로그래밍 방식으로 정의하여 역할을 부여할 수 있습니다.

모든 자동화 프로세스 중에는 보안 요구 사항과 규제를 준수해야 하며 암호 및 키를 사용하여 작업할 때는 각별히 주의해야 합니다. 항상 보안 모범 사례를 따라야 합니다. AWS에서의 보안 중요성에 대한 자세한 내용은 AWS 보안 센터를 참조하십시오.[11]

결론

기술 업체에서는 DevOps 원칙과 사례를 수용하여 클라우드로 원활하고 효율적이며 효과적인 방법으로 전환할 수 있습니다. 이러한 원칙은 AWS 플랫폼에 포함되어 있습니다. 실제로 이러한 원칙은 다양한 AWS 서비스의 토대가 되며, 특히 배포 및 모니터링 상품에서는 더욱 그렇습니다.

AWS CloudFormation 서비스를 사용하여 코드형 인프라를 정의하는 것부터 시작하십시오. 그런 다음, 애플리케이션에서 AWS CodeDeploy, AWS CodePipeline 및 AWS CodeCommit 같은 서비스의 도움을 받아 연속 배포를 어떤 방식으로 사용할지 정의하십시오. 애플리케이션 수준에서는 AWS Elastic Beanstalk 및 AWS OpsWorks 같은 서비스를 사용하여 일반 아키텍처의 구성을 간소화하십시오. 이러한 서비스를 사용하면 Auto Scaling 및 Elastic Load Balancing 같은 다른 중요 서비스도 쉽게 포함할 수 있습니다. 마지막으로 DevOps 모니터링 전략(AWS CloudWatch)과 강력한 보안 사례(AWS IAM)를 사용하십시오.

AWS를 파트너로 삼음으로써, 귀사의 DevOps 원칙이 비즈니스와 IT 조직에 민첩성을 제공하고 클라우드로의 전환을 가속화할 것입니다.

[11] http://aws.amazon.com/security/