



AWS DevOps

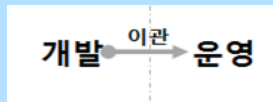
1. 데브옵스(DevOps)란?

개발자와 운영자의 소통, 협업 및 통합을 강조하는 문화, 방법론, 프로세스, 도구 모두를 의미

1. 기존 개발 체계 문제점

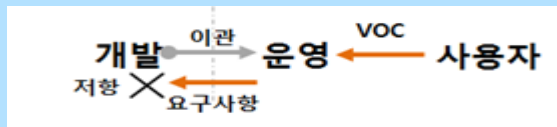
가. 개발팀과 운영팀 분리

- 개발팀: 시스템 개발, 개발 완료 후 운영팀에 이관
- 운영팀: 해당 시스템 배포 및 관리, 운영



나. 개발팀과 운영팀의 업무 분리로 인해 발생하는 문제

- 서비스 요구 사항의 신속한 반영의 문제
- 고객의 요구 사항에 민감한 소프트웨어 개발



개발자(Dev)는 고객의 요구사항을 빠르게 수용해서 서비스를 개발하고, 개발된 내용을 빠르게 적용하고 확인하길 바람.

운영자(Ops)는 제공될 서비스가 정확하게 동작하며, 테스트 되었고, 성능적 문제가 없으며, 다른 시스템에 영향을 주지 않고 안정적으로 동작하기를 바람.

이러한 고민에서 데브옵스(DevOps)라는 개념이 출현

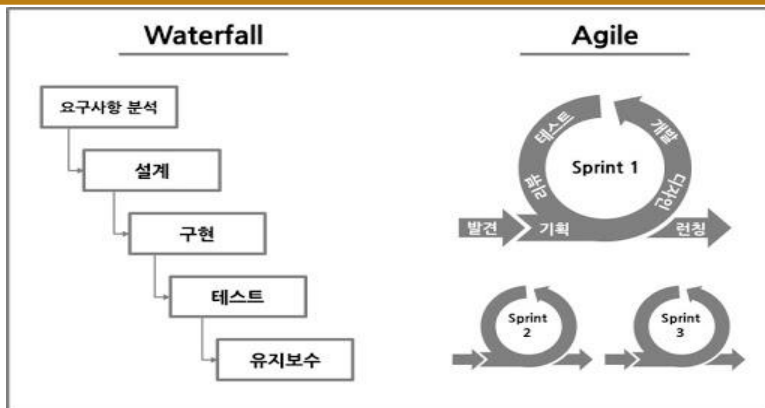
1. 데브옵스(DevOps)란?

개발자와 운영자의 소통, 협업 및 통합을 강조하는 문화, 방법론, 프로세스, 도구 모두를 의미

2. 애자일 방법론과 데브옵스의 유래

가. 애자일 방법론

- 협업(기획과 개발 통합)
- 기획팀과 개발팀을 하나의 팀으로 합쳐서 요구 사항 변화에 빠르게 반응할 수 있는 구조 변경
- Iterative 개발(반복적)과, Short Release를 이용한 비즈니스의 요구 사항 신속 반영, 변화에 대응



단점
운영팀에게 많은 부하와 운영적인 문제점을
촉발.

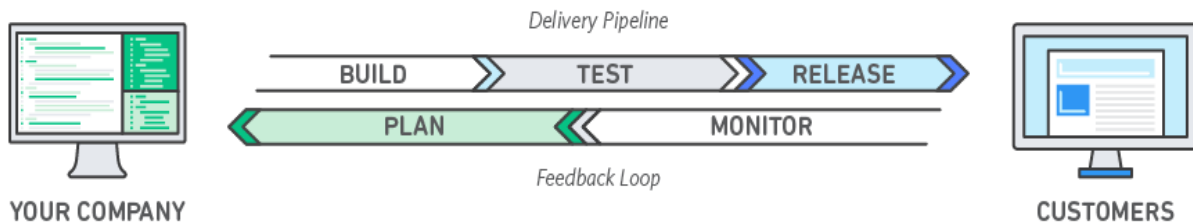
1. 데브옵스(DevOps)란?

개발자와 운영자의 소통, 협업 및 통합을 강조하는 문화, 방법론, 프로세스, 도구 모두를 의미

2. 애자일 방법론과 데브옵스의 유래

나. 데브옵스

- 협업(개발과 운영의 통합)
- 개발과 운영의 관계 해결
- 개발과 운영이 분리되면서 오는 문제점을 해결하기 위해 개발과 운영을 하나의 조직으로 합쳐서 팀을 운영하는 문화이자 방법론
- 개발과 운영을 합친 팀 구성 및 운영, 조금 더 정확하게 이야기 하면, 개발 운영 뿐만 아니라 테스트까지 하나의 팀으로 통합
- 엔지니어가 프로그래밍, 빌드, 직접 시스템에 배포 및 서비스를 실행, 사용자와 끊임 없이 상호작용하면서 서비스를 개선해 나가는 일련의 과정이자 문화

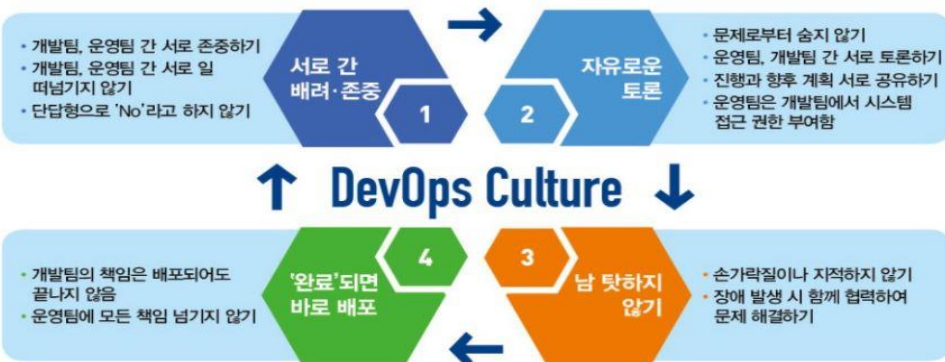
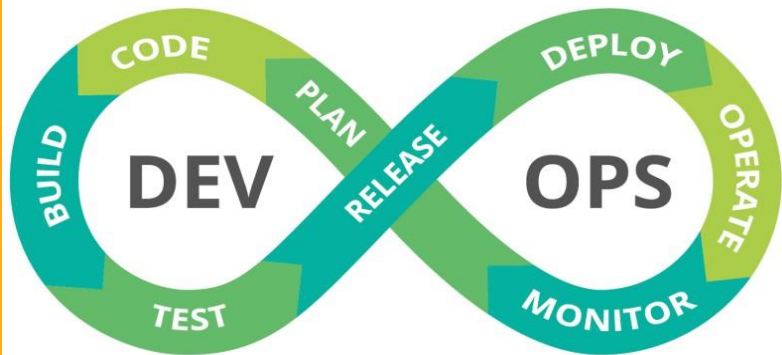


1. 데브옵스(DevOps)란?

개발자와 운영자의 소통, 협업 및 통합을 강조하는 문화, 방법론, 프로세스, 도구 모두를 의미

3. DevOps 효과

1. 빠르고 정확하게 운영 환경에 코드를 배포할 수 있음.
2. 배포 프로세스 단계별로 빠른 피드백 루프를 생성하여 빠르게 피드백 받을 수 있음.
3. 자동화를 통해 보다 빠르고 효과적으로 업무 진행이 가능.
4. 원하는 기능과 고객의 요구사항을 독립적으로 전달하고 배포할 수 있음.
5. 신규 제품이나 기능의 출시와 이에 대한 피드백을 보다 효과적으로 진행할 수 있음.



1. 데브옵스(DevOps)란?

개발자와 운영자의 소통, 협업 및 통합을 강조하는 문화, 방법론, 프로세스, 도구 모두를 의미

4. DevOps 기술적 구성 요소

1. 코드 기반 인프라 관리(IaC)

Code 기반으로 시스템과 운영/배포 환경을 구축 및 관리

2. 버전 관리

소스 및 빌드 관리를 위한 단일 시스템 구성
빌드 검증 테스트를 자동으로 수행

3. One Step 빌드/배포

한번 클릭으로 빌드/배포
예약작업을 통한 빌드/배포
검증 실패 시 배포 중지 및 알림
CI/CD(Continuous Integration/Continuous Delivery)를 사용

4. 장애 시 빠른 인프라 배포

문제 발생 시 기존 인프라를 수정하지 않고 Image 기반의 인프라를 활용하여 빠르게 재배포
환경 관리 도구를 이용하여 쉽고 빠르게 신규 환경 구성

2. AWS와 DevOps를 위한 Tool - Infra as a Code

1. Infra as a Code의 정의

Infrastructure as a Code의 약자로 IT 서비스를 위한 시스템을 구성할 때 수동으로 구성하는 대신 Shell 기반의 스크립트를 사용하여 컴퓨팅 인프라를 구성하는 기술.

보통 '프로그래밍형 인프라'라고도 하는 Infrastructure as a Code는 인프라 구성을 프로그램 처리하는 방식을 말함.

현재 IaC는 클라우드 컴퓨팅의 핵심적인 구성 요소로 자리잡고 있으며, 성공적인 DevOps 도입을 위해 반드시 필요한 요소로 각광받고 있는 기술 중 하나가 되었음.

2. Infra as a Code의 기대 효과

첫 번째, 비용 절감

두 번째, 빠른 실행

세 번째, 리스ٹ 관리

2. AWS와 DevOps를 위한 Tool - Infra as a Code









3. Infra as a Code의 도구

구성 조정 도구(Configuration Orchestration Tool)

서버 및 기타 인프라의 구축을 자동화하도록 설계
예) AWS Cloud Formation, Terraform

구성 관리 도구(Configuration Management Tool)

이미 프로비저닝된 인프라의 소프트웨어와 시스템을
구성하고 관리하도록 설계
예) Chef, ANSIBLE

Tool	Tool type	Infrastructure	Architecture	Approach	Language
 CHEF	Config management	Mutable	Pull	Declarative & Imperative	Ruby
 puppet	Config management	Mutable	Pull	Declarative	DSL & ERB
 SALTSTACK	Config management	Mutable	Push & pull	Declarative & Imperative	YAML
 AWS CloudFormation	Provisioning	Immutable	Push	Declarative	JSON & YAML
 AWS Cloud Development Kit	Provisioning	Immutable	Push	Declarative	TS, JS, Python, Java, C#/ .Net
 ANSIBLE	Config management	Mutable	Push	Declarative & Imperative	YAML
 Terraform	Provisioning	Immutable	Push	Declarative	HashiCorp Configuration Language
 pulumi	Provisioning	Immutable	Push	Declarative	JS, TS, Python, Go, NET language, incl C#, F#, and VB

2. AWS와 DevOps를 위한 Tool – AWS Cloudformation

1. 템플릿 기반의 구성 조정 도구 – AWS Cloudformation

AWS Cloudformation 란?

Amazon Web Services 리소스를 모델링하고 설정하여 리소스 관리 시간을 줄이고, AWS에서 실행되는 애플리케이션에 더 많은 시간을 사용하도록 해주는 대표적인 IaC기반의 구성 조정 도구.

구분	내용
서비스명	AWS CloudFormation
설명	모든 클라우드 인프라 리소스를 모델링 및 프로비저닝
주요 특징	<ul style="list-style-type: none"> - 사용자 인프라용 템플릿 생성 - JSON(JavaScript Object Notation), YAML(Yet Another Markup Language)로 텍스트 모델링 작성 - 인프라 변경 사항에 대해 미리보기 기능 제공 - 코드처럼 버전 관리/코드 검토/템플릿 업데이트 기능 제공 - 종속성 요구에 기반하여 AWS 리소스 제공 - 개발, CI/CD 및 관리 도구와 통합 - 추가적인 비용 없이 사용할 수 있으며, 리소스 사용 비용만 지불

2. AWS와 DevOps를 위한 Tool – AWS Cloudformation

1. 템플릿 기반의 구성 조정 도구 – AWS Cloudformation

AWS Cloudformation의 특징

AWS Cloudformation의 특징

첫 번째, 인프라 관리 간소화

일반적인 클라우드 인프라 구성은 여러 리소스를 생성하고 서로 연계되어 작동하도록 구성하므로 시간이 많이 소요되며 관리가 어려움
Cloudformation 템플릿을 사용하여 스택을 생성하면 리소스를 스택 단위로 손쉽게 구성 및 관리할 수 있음.

두 번째, 신속한 인프라 복제

애플리케이션의 가용성을 높이기 위해 AWS Cloudformation을 이용하여 여러 리전에 복제할 수 있음.

세 번째, 인프라 변경 사항을 쉽게 제어 및 추적

인프라의 프로비저닝을 위한 리소스와 설정이 템플릿에 저장되어 버전 관리 기능을 통해 이전과의 차이점을 추적하여 인프라 변경 사항을 추적할 수 있음.

이를 통해 인프라의 프로비저닝 중 문제가 발생되면 언제든지 인프라에 대한 변경 사항을 되돌리기 위해 이전 버전의 템플릿을 사용할 수 있음.

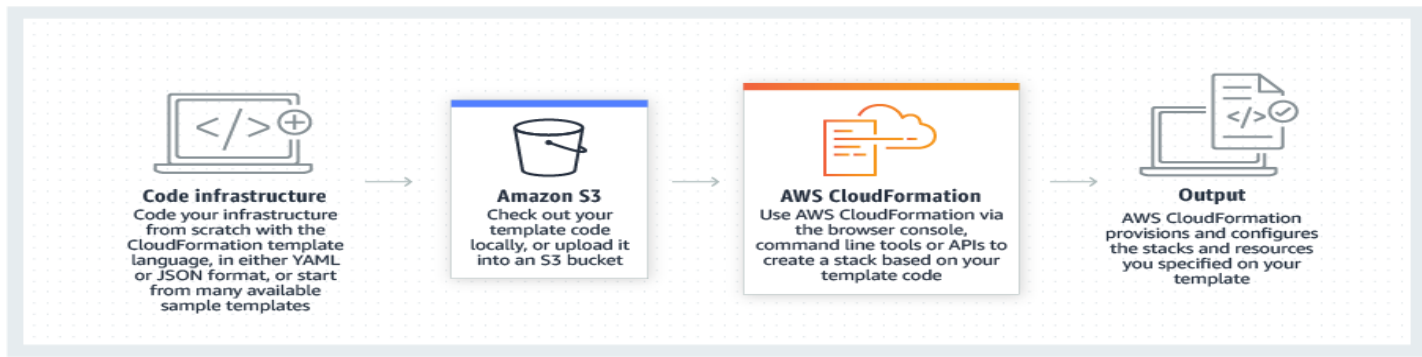
2. AWS와 DevOps를 위한 Tool – AWS Cloudformation

1. 템플릿 기반의 구성 조정 도구 – AWS Cloudformation

AWS Cloudformation의 작동 방식

AWS Cloudformation의 작동 방식

1. YAML 또는 JSON 포맷으로 Cloudformation 템플릿을 작성하여, 수행하기를 원하는 인프라에 대한 정보를 코드로 작성.
2. Cloudformation 템플릿을 로컬에 저장하거나, S3 Bucket에 저장.
3. AWS 콘솔, 명령줄도구(CLI) 또는 API Call을 통해 AWS Cloudformation을 실행하여 템플릿 코드를 기반으로 스택을 생성.
4. 템플릿에 지정된 스택을 통해 AWS 리소스에 대한 프로비저닝 및 구성 작업을 진행.



2. AWS와 DevOps를 위한 Tool – AWS Cloudformation

1. 템플릿 기반의 구성 조정 도구 – AWS Cloudformation

AWS Cloudformation의 구성

구분	내용
Template	<ul style="list-style-type: none"> - 템플릿(Template) : 스택 리소스 프로비저닝 및 구성을 위해 필요한 파일 - JSON 또는 YAML 형식 텍스트 파일로 작성 - 템플릿은 CloudFormation 스택에서 프로비저닝 할 리소스를 설명함. - CloudFormation Designer 또는 텍스트 편집기를 사용하여 템플릿 생성 가능
Cloudformation	<ul style="list-style-type: none"> - Stack을 생성하고 Stack에 대한 변경 사항을 확인 및 업데이트 - Stack 생성 및 변경 중 에러 감지를 통한 롤백 지원
Stack	<ul style="list-style-type: none"> - 스택이란 하나의 단위로 관리할 수 있는 AWS 리소스 모음 - 스택의 생성, 수정, 삭제를 통해 리소스 모음의 생성, 수정, 삭제 가능 - 스택의 모든 리소스는 CloudFormation 템플릿을 통해 정의됨 - 스택을 삭제하면 관련 리소스가 모두 삭제됨



2. AWS와 DevOps를 위한 Tool – AWS Cloudformation

1. 템플릿 기반의 구성 조정 도구 – AWS Cloudformation

AWS Cloudformation Template 구성사항

```
{
  "AWSTemplateFormatVersion" : " version date ",
  "Description" : " JSON string ",
  "Metadata" : {
    template metadata : 템플릿에 대한 추가 정보
  },
  "Parameters" : {
    set of parameters : 템플릿 실행 시 전달할 파라미터 값
  },
  "Mappings" : {
    set of mappings : 템플릿 실행 시 선택하게 되는 값(특정 리전, 인스턴스)
  },
  "Conditions" : {
    set of conditions : 특정 자원에 대한 생성 여부를 판단하는 조건
  },
  "Transform" : {
    set of transform : Serverless 애플리케이션용
  },
  "Resources" : {
    set of resources : 생성될 AWS 자원 나열 (필수)
  },
  "Outputs" : {
    set of outputs : 템플릿 실행 후 만들어진 자원 결과값(자원 ID, IP 등)
  }
}
```

필수 섹션은 **Resources** 이며,
그 외 나머지 섹션은 옵션 사항

2. AWS와 DevOps를 위한 Tool – AWS Cloudformation

1. 템플릿 기반의 구성 조정 도구 – AWS Cloudformation

AWS Cloudformation Template 구성사항

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "Create an EC2 instance running the latest Amazon Linux AMI.",
  "Parameters": {
    "KeyPair": {
      "Description": "The EC2 Key Pair to allow SSH access to the instance",
      "Type": "String"
    }
  },
  "Resources": {
    "Ec2Instance": {
      "Properties": {
        "ImageId": "ami-9d23aeea",
        "InstanceType": "m3.medium",
        "KeyName": {
          "Ref": "KeyPair"
        }
      },
      "Type": "AWS::EC2::Instance"
    }
  },
  "Outputs": {
    "InstanceId": {
      "Description": "The InstanceId of the newly created EC2 instance",
      "Value": {
        "Ref": "Ec2Instance"
      }
    }
  }
}
```

Parameter 설정을 위해 사용자가 직접 타이핑 하거나, 선택옵션을 통해 선택 (생성 리소스 수량, 크기 설정 등)

2. AWS와 DevOps를 위한 Tool – AWS Cloudformation

1. 템플릿 기반의 구성 조정 도구 – AWS Cloudformation

AWS Cloudformation Template 구성사항

```
{
  "AWSTemplateFormatVersion": "2010-09-09",

  "Description": "Create an EC2 instance running the latest Amazon Linux AML.",
  "Parameters": {
    "KeyPair": {
      "Description": "The EC2 Key Pair to allow SSH access to the instance",
      "Type": "String"
    }
  },

  "Resources": {
    "Ec2Instance": {
      "Properties": {
        "ImageId": "ami-9d23aeaa",
        "InstanceType": "m3.medium",
        "KeyName": {
          "Ref": "KeyPair"
        }
      },
      "Type": "AWS::EC2::Instance"
    }
  },

  "Outputs": {
    "InstanceId": {
      "Description": "The InstanceId of the newly created EC2 instance",
      "Value": {
        "Ref": "Ec2Instance"
      }
    }
  }
}
```

지정된 속성값 (ImageID & Instance Type)
 및 KeyPair 파라미터 값에대한 참조를 포함

2. AWS와 DevOps를 위한 Tool – AWS Cloudformation

1. 템플릿 기반의 구성 조정 도구 – AWS Cloudformation

AWS Cloudformation Template 구성사항

```
{
  "AWSTemplateFormatVersion": "2010-09-09",

  "Description": "Create an EC2 instance running the latest Amazon Linux AML.",
  "Parameters": {
    "KeyPair": {
      "Description": "The EC2 Key Pair to allow SSH access to the instance",
      "Type": "String"
    }
  },
  "Resources": {
    "Ec2Instance": {
      "Properties": {
        "ImageId": "ami-9d23aeea",
        "InstanceType": "m3.medium",
        "KeyName": {
          "Ref": "KeyPair"
        }
      },
      "Type": "AWS::EC2::Instance"
    }
  },
  "Outputs": {
    "InstanceId": {
      "Description": "The InstanceId of the newly created EC2 instance",
      "Value": {
        "Ref": "Ec2Instance"
      }
    }
  }
}
```

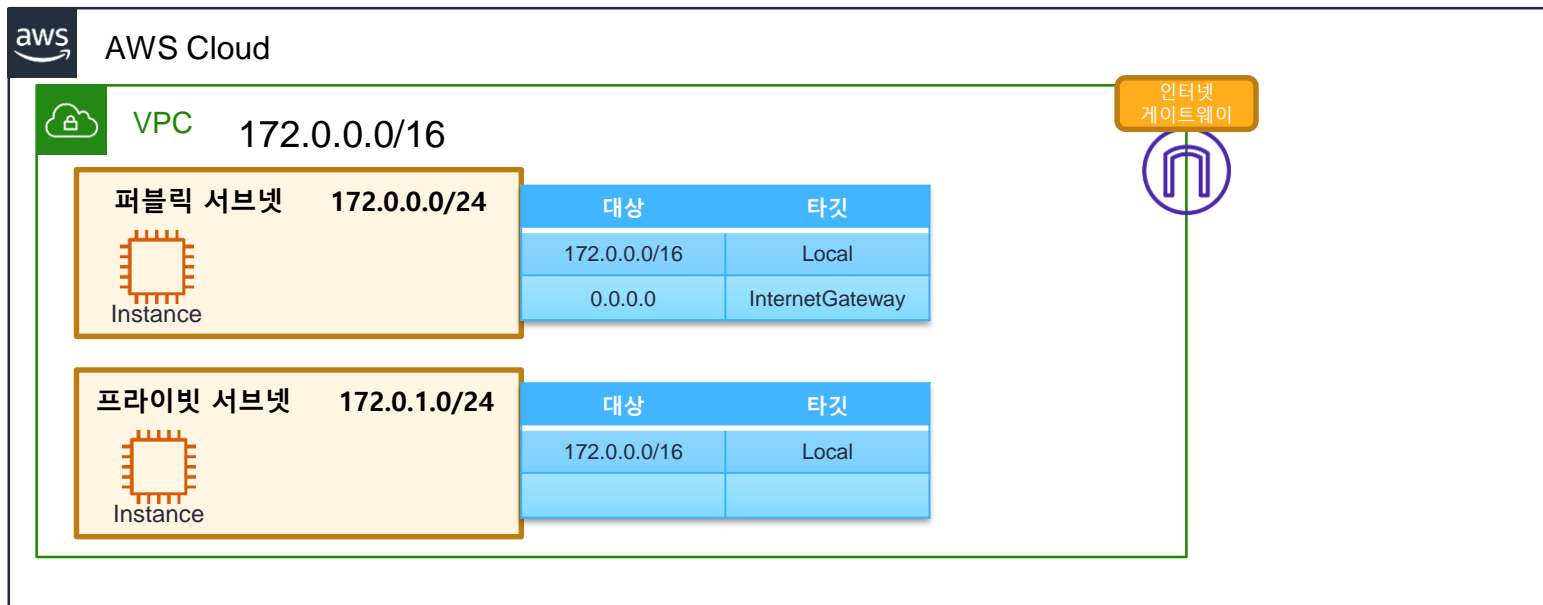
템플릿 실행 후 결과 값 표시 (IP 주소 등)



2. AWS와 DevOps를 위한 Tool – AWS Cloudformation

1. 템플릿 기반의 구성 조정 도구 – AWS Cloudformation

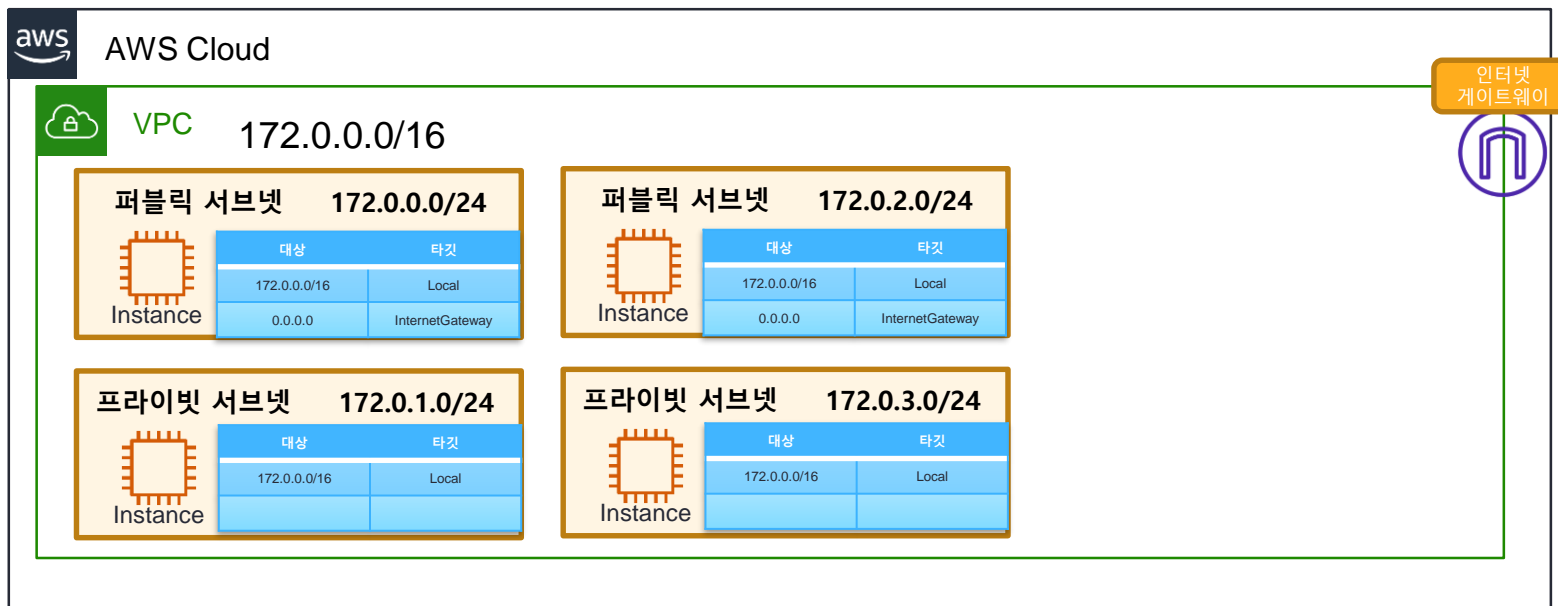
AWS Cloudformation 실습 1



2. AWS와 DevOps를 위한 Tool – AWS Cloudformation

1. 템플릿 기반의 구성 조정 도구 – AWS Cloudformation

AWS Cloudformation 실습 2



2. AWS와 DevOps를 위한 Tool – AWS OpsWorks

2. Chef 기반의 구성 관리 도구 – AWS OpsWorks

엔터프라이즈 기업 및 많은 클라우드 인프라를 사용하는 환경에서 애플리케이션 및 인프라를 구성하고 운영하도록 지원하는 대표적인 IaC(Infrastructure as a Code).

세 가지 서비스로 구분

첫 번째, AWS OpsWorks for Chef Automate

두 번째, AWS OpsWorks for Puppet Enterprise

세 번째, AWS OpsWorks Stacks

2. AWS와 DevOps를 위한 Tool – AWS OpsWorks

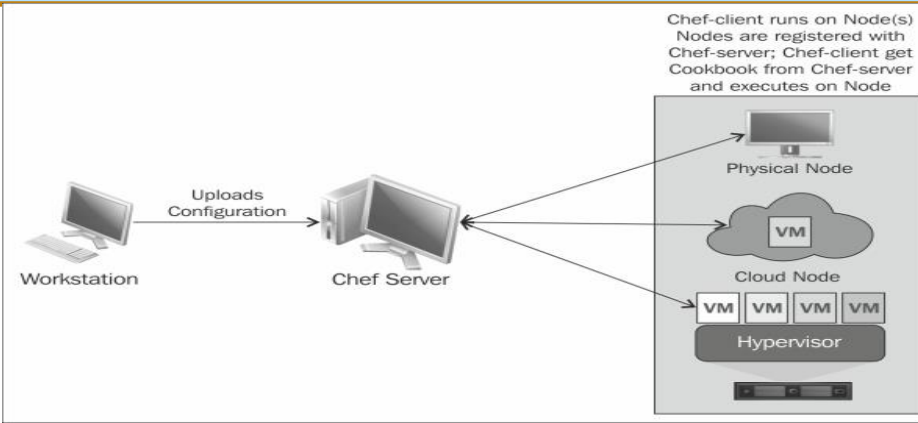
2. Chef 기반의 구성 관리 도구 – AWS OpsWorks

AWS OpsWorks for Chef Automate

Chef 레시피 기반으로 서버와 애플리케이션에 대한 자동화 관리, 규정 준수, 보안 자동화 테스트 등을 지원하는 구성 관리 플랫폼으로 호스트 기반의 Chef Automate를 완전 관리형 서비스로 제공하는 AWS 기반의 구성 관리 서비스.

Chef Automate 플랫폼은 소프트웨어 및 운영 체제 구성, 지속적 규정 준수, 패키지 설치, 데이터베이스 설정 등의 운영 작업을 처리할 수 있음.

쿡북을 사용하여 호스트 및 애플리케이션 구성, 패키지 설치, 인스턴스 종료 등과 같은 운영 작업을 자동화할 수 있으며, 커스텀 쿡북을 만들거나, Chef 커뮤니티에서 공개되어 있는 3,000개 이상의 쿡북을 활용할 수 있음.



2. AWS와 DevOps를 위한 Tool – AWS OpsWorks

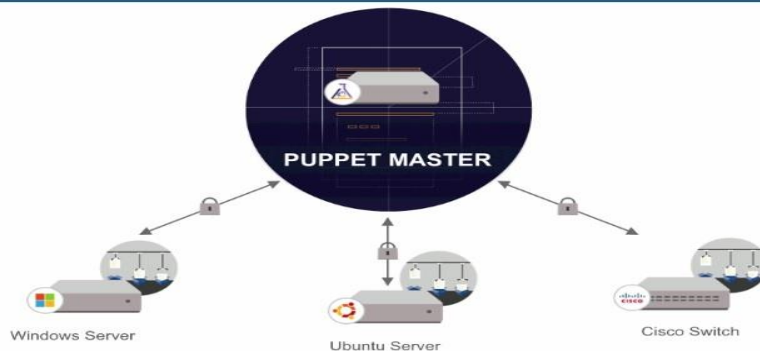
2. Chef 기반의 구성 관리 도구 – AWS OpsWorks

AWS OpsWorks for Puppet Enterprise

인프라 및 애플리케이션의 관리를 위한 Puppet의 자동화 툴 세트인 Puppet Enterprise를 AWS 기반으로 제공하는 완전 관리형 구성 관리 서비스.

Puppet Master 서버를 통해 EC2 인스턴스나 On-Premise 서버를 자동으로 패치/업데이트 및 백업을 수행함으로써, 인프라 운영/관리를 위한 별도의 구성 관리 시스템을 운영하거나 인프라 유지 관리에 대해 걱정할 필요가 없음.

Web기반의 Puppet 콘솔을 통해 Puppet Enterprise의 강력한 인프라 운영/관리 기능인 모든 기능을 사용할 수 있으며, 기존 Puppet코드와도 호환.



2. AWS와 DevOps를 위한 Tool – AWS OpsWorks

2. Chef 기반의 구성 관리 도구 – AWS OpsWorks

AWS OpsWorks Stacks

EC2 인스턴스와 On-Premise에서 애플리케이션과 서버를 모두 관리할 수 있도록 해주는 구성 관리 서비스를 제공. AWS OpsWorks for Chef Automate 나 AWS OpsWorks for Puppet Enterprise와는 다르게 별도의 관리용 서버를 필요로 하지 않으며, Chef Solo를 통한 Chef 레시피를 통해 구성 관리 서비스를 제공.

OpsWorks Stacks을 사용하면 로드 밸런싱, 데이터베이스, 애플리케이션 서버와 같이 다양한 인프라 계층이 포함된 스택을 통해 애플리케이션을 모델링할 수 있음.

프리티어를 제공하지 않음.

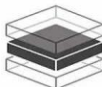


AWS OpsWorks

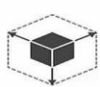
An integrated DevOps application management solution



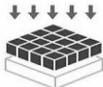
A **stack** represents the compute infrastructure and applications that you want to manage together.



A **layer** defines how to set up and configure a set of instances and related resources.



Decide how to scale: manually, with **24/7** instances, or automatically, with **load-based** or **time-based** instances.



Then deploy your app to specific instances and customize the deployment with Chef recipes.

1. Stacks

OpsWorks로 관리할 인프라 및 애플리케이션의 집합

2. Layer

인스턴스 및 리소스 집합을 설정하고 구성하는 방법을 정의

3. Scale

수동, 스케줄 기반, 부하기반으로 확장에 대한 조건을 정의

4. Chef 레시피

인스턴스의 특성에 따른 의존성과 커스텀요건을 고려한 구성 반영

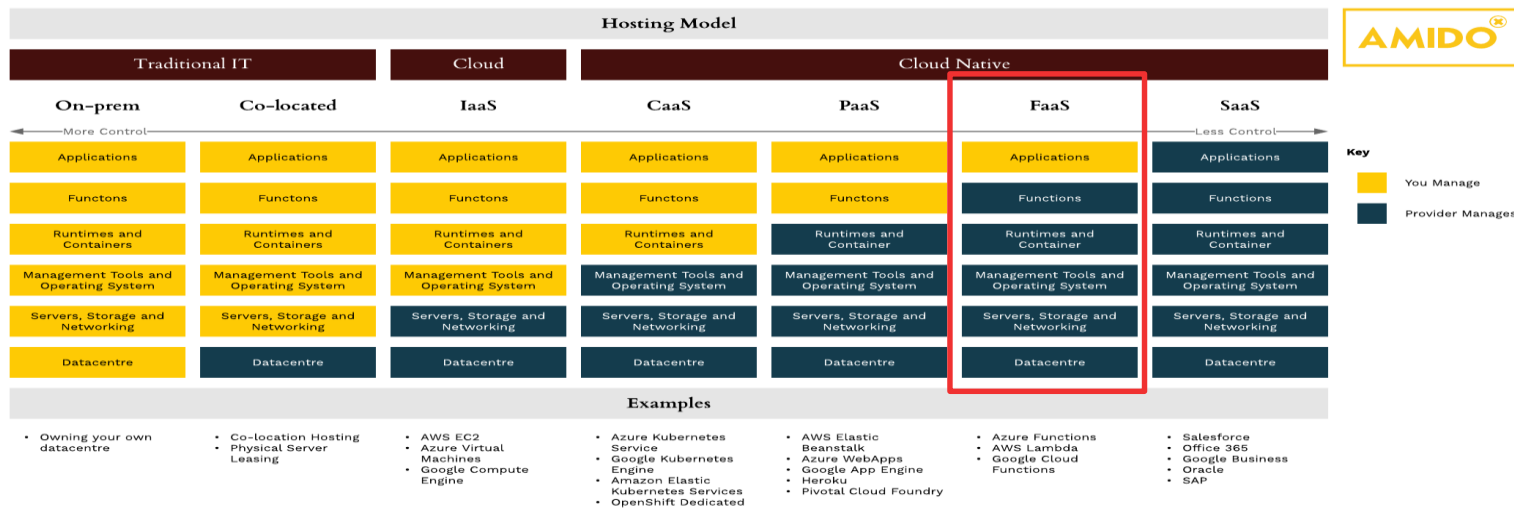


2. AWS와 DevOps를 위한 Tool – AWS Lambda

3. 서버리스 컴퓨팅 서비스 – AWS Lambda

Cloud Native Patterns, XaaS(Everything as a Service)

On-Premise 환경에서 서비스를 직접 제공해주는 인프라부터 소프트웨어까지 서비스 제공의 형태가 바뀌는 것을 의미.





2. AWS와 DevOps를 위한 Tool – AWS Lambda

3. 서버리스 컴퓨팅 서비스 – AWS Lambda

FaaS(Function as a Service)

함수를 서비스로 이용하는 것.

개발자가 환경을 구성하고 서버코드를 작성하는 것이 아닌, Faas를 이용함으로써 함수만 구현하면 됨.

서버코드를 실행하기 위해 서버를 구성하고 코드를 배포하던 형식을 줄이고, 원하는 로직만 함수를 기반으로 구현하여 각 클라우드 서비스 제공사의 조건에 충족되면 실행됨.

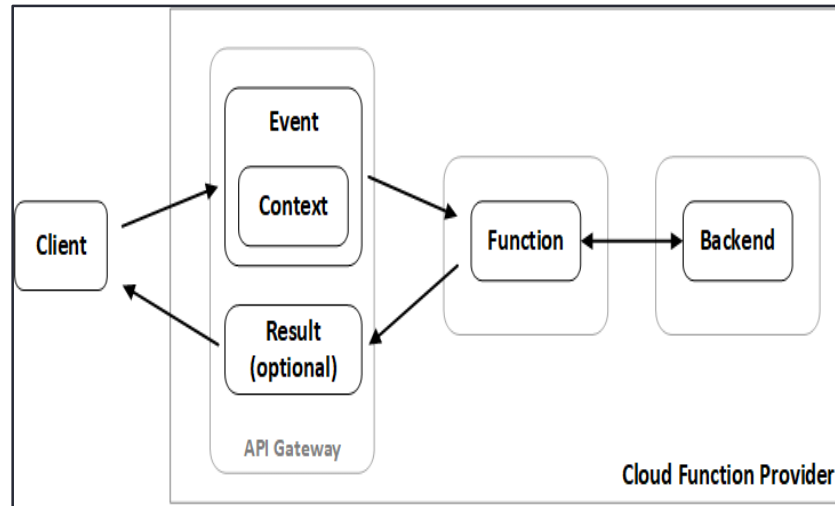
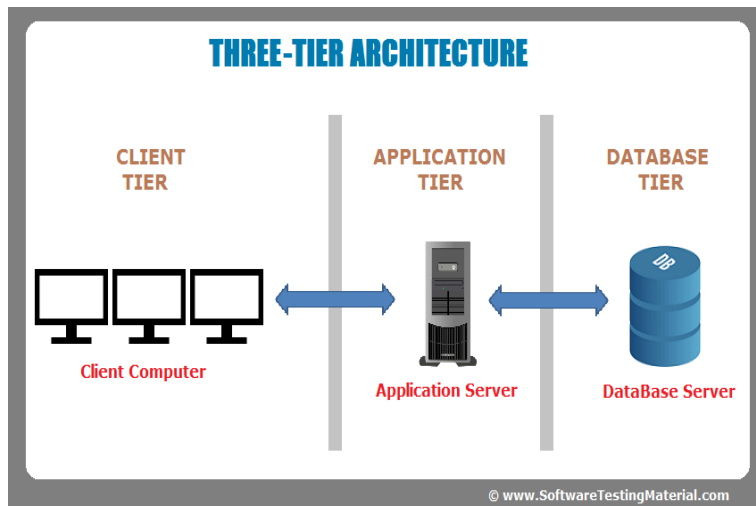
함수가 호출되면 컨테이너(혹은 VM)이 실행되며 정의한 함수가 런타임(실행 환경)내에서 실행된 다음 컨테이너(혹은 VM)이 종료. 이벤트 기반으로 동작하며 단일 함수로 간단한 작업만이 아니라 여러 함수, 다른 서비스들을 연결하여 복잡한 로직 구현도 가능.

AWS에서 제공하는 AWS Lambda가 대표적인 Faas.



2. AWS와 DevOps를 위한 Tool – AWS Lambda

3. 서버리스 컴퓨팅 서비스 – AWS Lambda





2. AWS와 DevOps를 위한 Tool – AWS Lambda

3. 서버리스 컴퓨팅 서비스 – AWS Lambda

FaaS(Function as a Service) 단점

1. 상태유지가 되지 않는다.
 - 컨테이너는 잠시 실행되는 환경으로 상태 비저장(Stateless).
 - 서버코드로 상태 값을 유지하고, 그 값을 이용하여 로직을 구현하던 방식은 사용할 수 없음.
2. 함수가 실행되기 위해 항상 준비된 상태가 아니다.
 - 실행 시 약간의 지연시간이 발생함. 이를 '콜드 스타트(Cold Start)'라고 함.
 - 함수를 실행했다면 함수를 실행한 컨테이너는 잠시 대기상태가 되는데 이때 다시 실행하는 것을 '웜 스타트(Warm Start)'라고 함.
 - 이는 준비되어 있어 처음 호출할 때 발생하는 지연시간이 발생하지 않는다.
3. 서비스 제공사(클라우드 회사)에 의존적이다.
 - AWS기반으로 설계되어 있는 것을 Microsoft Azure의 환경이나 Google Cloud Platform의 환경으로 변경하기 쉽지 않다.



2. AWS와 DevOps를 위한 Tool – AWS Lambda

3. 서버리스 컴퓨팅 서비스 – AWS Lambda

AWS Lambda는 서버를 프로비저닝하거나 관리하지 않아도 코드를 실행할 수 있도록 해주는 컴퓨팅 서비스. 필요 시에만 코드를 실행하며, 하루 몇 개의 요청에서 초당 수천 개의 요청까지 자동으로 확장이 가능. 사실상 모든 유형의 애플리케이션이나 백엔드 서비스에 대해 코드를 별도로 관리 없이 실행할 수 있음.

구분	내용
서비스명	AWS Lambda
설명	<ul style="list-style-type: none">- 서버에 대한 걱정 없이 코드 실행- 사용한 컴퓨팅 시간에 대해서만 비용 지불
주요 특징	<ul style="list-style-type: none">- 기존 코드 활용 가능(Node.js, Java, Python, Go, C#)- 단순한 지원 모델을 가지며, 실행되는 메모리에 따라 CPU, N/W 자원 할당- 여러 AWS 서비스들과 통합되어 있으며, Event, Request 기반으로 실행 가능- 자체 Editor, Zip 배포, Cloud9을 통해 개발 및 배포 가능- Cloudwatch, X-ray를 통해 요청 수, 에러 수, 처리 시간, 처리량 모니터링 기능- AWS IAM Role을 사용한 권한 관리와 AWS 이벤트 소스의 자원 정책 적용
프리티어	<ul style="list-style-type: none">- 월 1백만 건 무료 요청과 월 44,000초 컴퓨팅 시간(메모리에 따라 변경됨)- 프리티어는 12개월 이후에도 종료되지 않으며, AWS 고객에게 무기한 제공



2. AWS와 DevOps를 위한 Tool – AWS Lambda

3. 서버리스 컴퓨팅 서비스 – AWS Lambda

AWS Lambda의 서비스 동작 방식

4단계의 절차를 걸쳐 동작하도록 구현할 수 있음.

Lambda 함수의 호출은 이벤트 소스 또는 사전에 정의된 일정 또는 스케줄러를 통해 이벤트가 발생됨.

COMPONENTS OF A SERVERLESS APP



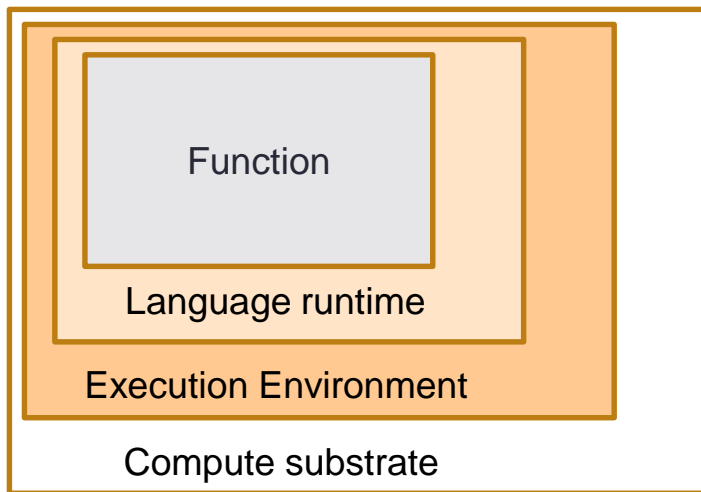


2. AWS와 DevOps를 위한 Tool – AWS Lambda

3. 서버리스 컴퓨팅 서비스 – AWS Lambda

AWS Lambda의 내부 정보

Lambda가 실행되면 아마존 리눅스 OS 기반의 Micro VM이 실행된다.
환경 변수 등 실행환경을 맞추고, 지정한 언어별 런타임 환경을 준비한다.
그리고 마지막에 작성한 함수를 실행한다. 이때 Lambda는 큰 단점이 있다.



Function	함수
Language runtime	언어별 런타임
Execution Environment	환경 변수 등의 실행 환경
Compute substrate	함수가 실행될 Micro VM



2. AWS와 DevOps를 위한 Tool – AWS Lambda

3. 서버리스 컴퓨팅 서비스 – AWS Lambda

AWS Lambda의 Cold Start / Warm Start

AWS Lambda가 실행되면 작성한 코드를 다운로드 하고 실행환경을 구성한다. 이때를 Full cold start 라고 함.

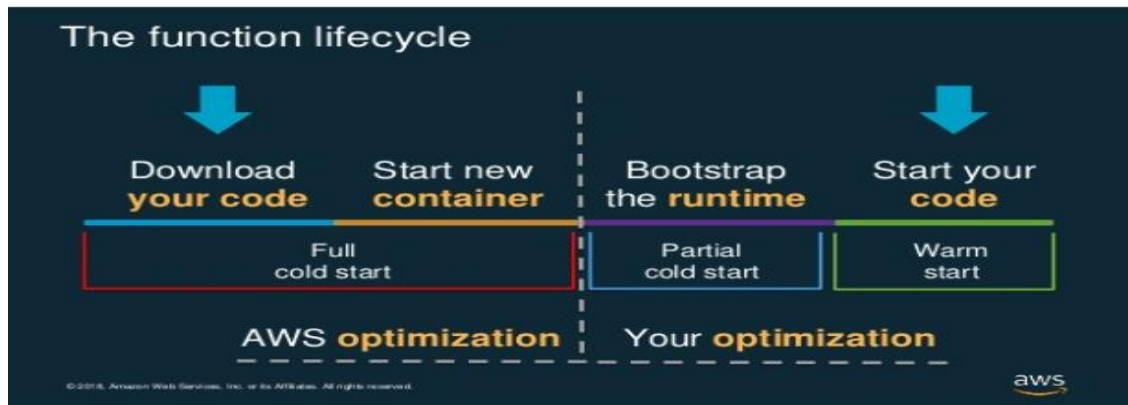
런타임을 준비하는 과정을 Partial cold start 라고 함.

마지막으로 함수가 실행될 때를 Warm start 라고 함.

이처럼 함수가 실행될 때 순서가 정해져 있고 이에 따라 지연시간이 발생함.

Micro VM이 올라 갔다가 내려간 뒤 다시 실행하면 Full cold start 부터 시작함.

만약, Micro VM이 유지되는 시간 이전에 재요청하면 Partial cold start 가 바로 진행되어 지연시간을 줄일 수 있음.





2. AWS와 DevOps를 위한 Tool – AWS Lambda

3. 서버리스 컴퓨팅 서비스 – AWS Lambda

AWS Lambda의 기대 효과

구분	내용
완전 관리형 서비스	<ul style="list-style-type: none">- 가용성이 뛰어나며, 내결함성을 갖춘 인프라에서 코드 실행- OS의 패치나, 사용량 증가에 따른 서버 증설 및 관리 불필요- 코드를 원활하게 배포하고 내장된 로깅 및 모니터링 기능 제공
기존 보유 코드 재활용	<ul style="list-style-type: none">- 새로운 언어 및 도구, 프레임워크를 배울 필요 없음- 기존 라이브러리 및 타사 도구 활용 가능- 모든 코드나 라이브러리를 Layer로 만들어 손쉽게 공유 활용 가능
통합된 보안 모델	<ul style="list-style-type: none">- 내장 SDK, IAM과 통합하여 코드가 안정하게 다른 서비스의 액세스 제공- 기본적으로 VPC 내부에서 코드가 실행되며, 리소스의 액세스 제어 가능
사용량 기반 지불	<ul style="list-style-type: none">- 코드 실행에 필요한 컴퓨팅 시간 및 지원된 요청에 대해서만 비용 지불- 100밀리 단위로 청구 금액이 정산되기 때문에 비용 효율적인 서비스
높은 내결함성	<ul style="list-style-type: none">- 각 리전의 여러 가용 영역의 컴퓨팅 파워를 활용하여 높은 내결함성- 데이터 센터나 시설에 장애가 발생되어도 코드에 대한 보호 가능- 유지 관리 시간이나 예약된 가동 중단 시간 없음
자동 규모 조정	<ul style="list-style-type: none">- 사용량 변화에 따라 고객이 관리할 필요 없으며, 자동으로 확장- 하루 몇 개의 요청부터 1초당 수천 개의 요청까지 자동으로 쉽게 확장- 이벤트 발생 빈도의 증가에 상관없이 일관되게 높은 성능 유지



2. AWS와 DevOps를 위한 Tool – AWS CloudWatch

4. 최적의 모니터링 도구 – AWS CloudWatch

CI/CD(Continuous Integration / Continuous Delivery)

애플리케이션 개발에 필요한 여러 단계에 대한 자동화를 통해 애플리케이션을 보다 빠르고 짧은 주기로 고객에게 제공하는 방법. 지속적인 통합(Continuous Integration), 지속적인 서비스 제공(Continuous Delivery) 및 지속적인 배포(Continuous Deployment)를 통해 새로운 코드의 통합, 테스트, 릴리스, 배포 등의 애플리케이션 라이프 사이클 전체에 대한 자동화 과정을 모니터링 가능하도록 하는 것을 의미.

AWS CloudWatch

개발자 및 DevOps 엔지니어, IT 관리자가 이용할 수 있는 모니터링 서비스로 애플리케이션 및 시스템의 전반적인 성능을 모니터링하고, 상태 변화에 따라 서비스의 Scale Out/Up, 서비스 재시작 등을 수행할 수 있는 CI/CD 구현에 필수적인 모니터링 도구.

사전에 정의된 모니터링 지표에 대해 미리 임계치를 설정하고, 지표가 임계치를 넘어서게 되거나 기계 학습 알고리즘을 기반으로 지표를 분석함. 이상 동작을 식별하여 이를 기반으로 알림을 설정하거나, Scale-Out/Up을 수행하는 등의 인프라에 대한 작업 등을 자동화할 수 있으며, 이를 통해 운영 성능 및 리소스에 대한 최적화를 수행할 수 있음.

2. AWS와 DevOps를 위한 Tool – AWS CloudWatch

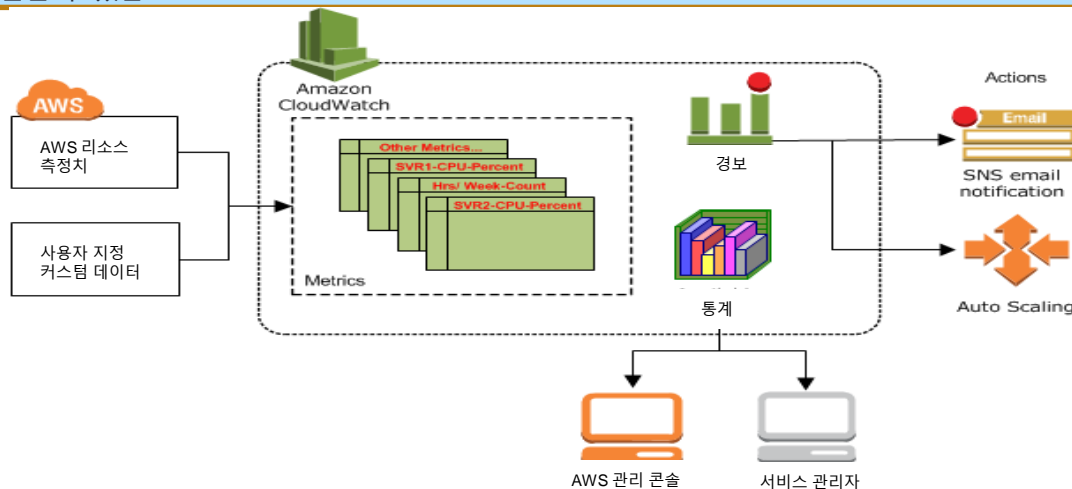
4. 최적의 모니터링 도구 – AWS CloudWatch

Amazon CloudWatch 동작 방식

기본적으로 리소스 및 사용자 커스텀 데이터의 저장소 역할, 저장된 데이터의 지표를 기반으로 통계를 작성하고 이에 대한 검색을 수행. 또한 사용자 지정 커스텀 데이터에 대해서도 지표를 작성하면 검색을 수행할 수 있음.

특정 기준을 충족하는 경우 EC2 인스턴스를 중지, 시작, 또는 종료하도록 경보 설정을 구성할 수 있음.

또한 Amazon EC2 Auto Scaling, SNS 작업을 수행하는 경보를 만들어 인프라 환경 변화에 따라 자동으로 대응할 수 있는 인프라 운영의 자동화를 구현할 수 있음.



2. AWS와 DevOps를 위한 Tool – AWS CloudWatch

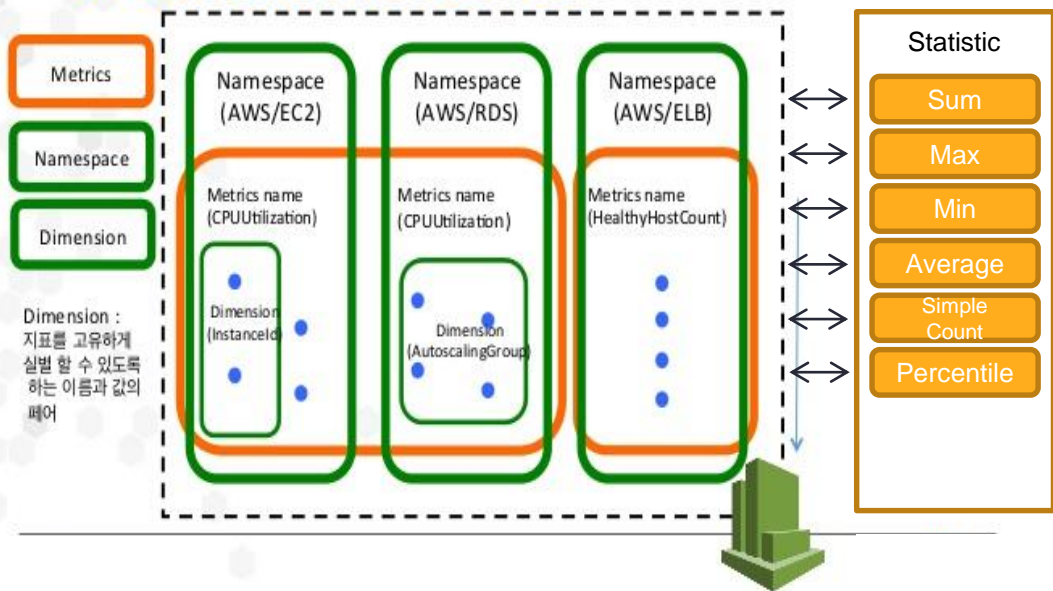
4. 최적의 모니터링 도구 – AWS CloudWatch

구분	내용
네임스페이스(Namespace)	<ul style="list-style-type: none"> - 서로 다른 애플리케이션에 대한 지표 저장을 위한 컨테이너 - 각 애플리케이션의 측정 지표를 서로 격리 - 모든 AWS 서비스는 고유한 네임스페이스 사용 - 사용자 지정 지표 생성 시 고유 네임스페이스 생성 필요
지표(Metrics)	<ul style="list-style-type: none"> - 애플리케이션 또는 서비스에 의해 생성되는 데이터 요소의 세트 - 정량적 평가, 측정, 비교를 위한 특정 시간 간격의 데이터 집합 - 기본적인 지표 데이터의 저장 간격은 5분이며, 고급 옵션은 1분(유료)
차원(Dimensions)	<ul style="list-style-type: none"> - 지표를 고유하게 식별할 수 있게 하는 Name/Value로 구분된 정보 - 지표 검색 시 필터링을 통해 원하는 데이터 검색 가능
통계(Statics)	<ul style="list-style-type: none"> - 지정한 기간에 대한 지표 데이터의 집계에 대한 결과 값 - 사용자 지정 데이터 또는 서비스 지표 데이터 요소 기반 통계 제공 - Min, Max, Sum, Average, SampleCount, Percentile 등 사용 가능
백분위수(Percentile)	<ul style="list-style-type: none"> - 데이터 세스에서 값의 상대적 위치를 나타내는 수치 - 지표 데이터의 분포 및 상대적 위치를 정확하게 이해하게 해줌 - 95백분위: 데이터의 95%가 이 값보다 아래, 5%가 위에 있음을 의미
경보(Alarms)	<ul style="list-style-type: none"> - 경보를 사용하여 작업을 자동으로 시작 - 지정한 기간에 지표를 감시하고, 임계값 기준으로 작업 수행 - 경보를 통해 Amazon SNS 주제, Auto Scaling에 대한 작업 수행

2. AWS와 DevOps를 위한 Tool – AWS CloudWatch

4. 최적의 모니터링 도구 – AWS CloudWatch

CloudWatch 기본 관리 구조



Amazon CloudWatch는 네임스페이스->지표->차원의 구조로 구성되며, AWS 리소스와 자원으로부터 수집된 데이터를 시계열로 분류하여 저장하게 됨.

이에 대한 통계 값을 사전에 정의된 Min, Max, Sum, Average, SimpleCount, Percentile와 함께 적합한 통계 지표 값을 선택하여 수치를 확인할 수 있음.

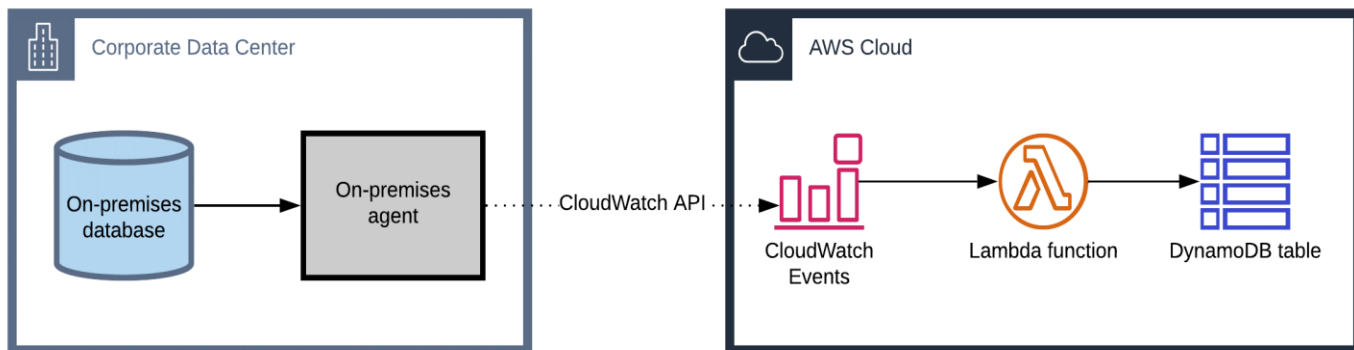
2. AWS와 DevOps를 위한 Tool – AWS CloudWatch

4. 최적의 모니터링 도구 – AWS CloudWatch

Amazon CloudWatch Event

AWS의 주요 리소스에 대한 변경 사항에 대해 실시간 스트림을 활용하여 EC2 인프라 관리의 자동화, Amazon 인공지능 기반 보안 서비스인 GuardDuty와 연계를 통해 보안 관제 및 이슈에 대한 처리 및 대응을 자동화할 수 있음.

CloudWatch Event 리소스의 변화를 실시간으로 인식할 수 있으며, 빠르게 사용할 수 있는 간단한 규칙을 통해 이벤트를 분류하고, 처리할 수 있는 대상으로 전달하여 다양한 방식으로 리소스 변화에 대해 빠르게 대응 가능한 체계를 만들 수 있음.



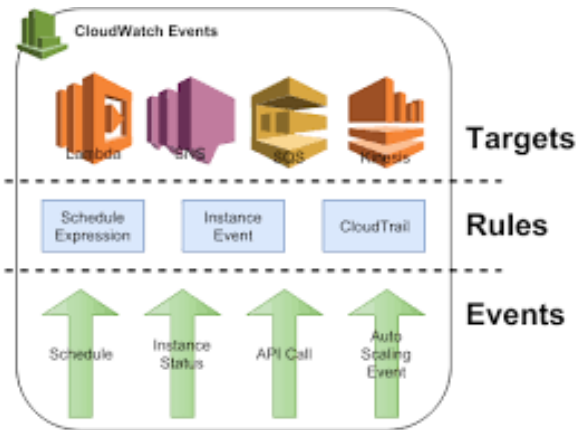
2. AWS와 DevOps를 위한 Tool – AWS CloudWatch

4. 최적의 모니터링 도구 – AWS CloudWatch

Amazon CloudWatch Event

AWS의 주요 리소스에 대한 변경 사항에 대해 실시간 스트림을 활용하여 EC2 인프라 관리의 자동화, Amazon 인공지능 기반 보안 서비스인 GuardDuty와 연계를 통해 보안 관제 및 이슈에 대한 처리 및 대응을 자동화할 수 있음.

CloudWatch Event 리소스의 변화를 실시간으로 인식할 수 있으며, 빠르게 사용할 수 있는 간단한 규칙을 통해 이벤트를 분류하고, 처리할 수 있는 대상으로 전달하여 다양한 방식으로 리소스 변화에 대해 빠르게 대응 가능한 체계를 만들 수 있음.



구분	내용
Event	<ul style="list-style-type: none"> - AWS 내부의 리소스에 대한 상태가 변경될 경우 발생 - AWS 리소스의 상태 변경 시 이벤트를 발생시켜 원하는 다른 작업 수행 예) EC2의 인스턴스 상태가 보류 상태에서 실행으로 변경 시 이벤트 생성 또는 EC2 인스턴스의 시작 또는 종료 발생 시 이벤트 생성
Rule	<ul style="list-style-type: none"> - CloudWatch로 들어오는 이벤트 중 처리가 필요한 이벤트 분류 작업 수행 - 이벤트를 처리할 Target으로 Event 전달 - 단일 규칙에서 여러 Target으로 이벤트 전달이 가능하며, 병렬 처리됨
Target	<ul style="list-style-type: none"> - Rule로 부터 전달받은 Event에 대한 처리 작업 수행 - EC2, Lambda, Kinesis 스트림, ECS, SNS, SQS 사용 가능

3. AWS와 DevOps를 위한 개발 Toolkits

1. CI/CD 란?

CI/CD(Continuous Integration / Continuous Delivery)

애플리케이션 개발에 필요한 여러 단계에 대한 자동화를 통해 애플리케이션을 보다 빠르고 짧은 주기로 고객에게 제공하는 방법. 지속적인 통합(Continuous Integration), 지속적인 서비스 제공(Continuous Delivery) 및 지속적인 배포(Continuous Deployment)를 통해 새로운 코드의 통합, 테스트, 릴리스, 배포 등의 애플리케이션 라이프 사이클 전체에 대한 자동화 과정을 모니터링 가능하도록 하는 것을 의미.

CI(Continuous Integration)

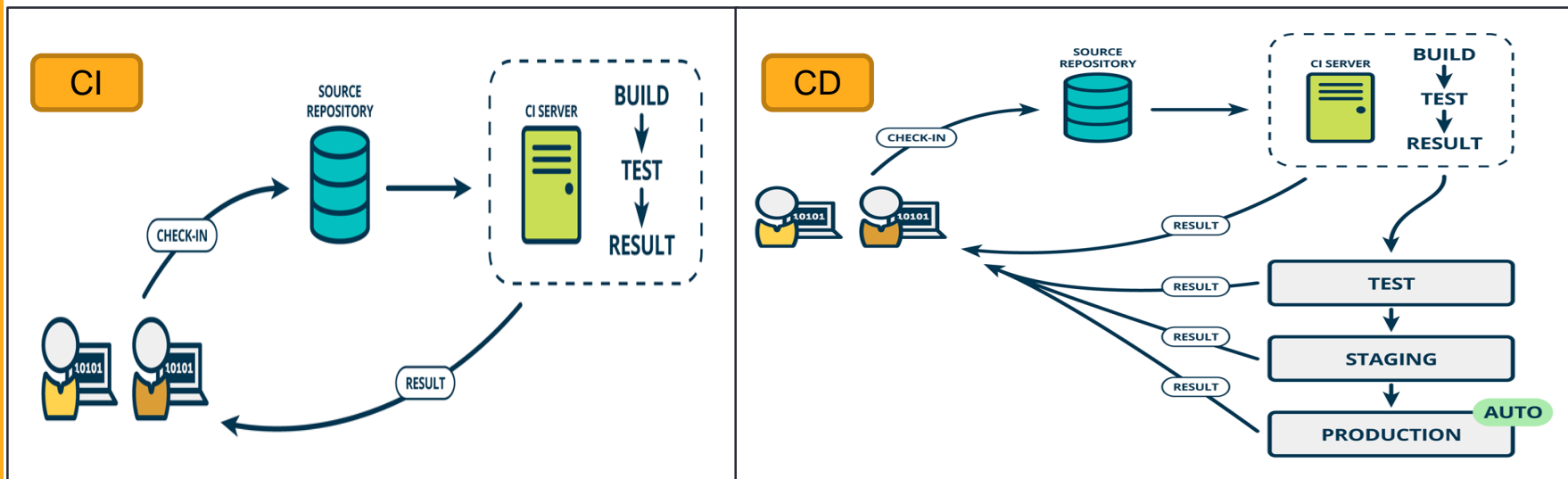
개발자를 위한 자동화 프로세스를 통해 새로운 코드 개발과, 코드의 변경 사항이 정기적으로 빌드 및 테스트되고 공유 리포지토리에 병합되어 여러 명의 개발자가 동시에 애플리케이션 개발과 관련된 코드를 작업할 경우에도 서로 충돌 없이 원하는 개발 작업을 수행하고 문제를 해결해 나가는 것.

CD(Continuous Delivery & Continuous Deployment)

개발자가 지속적인 서비스 전달과 배포를 통해 최소한의 노력으로 새로운 코드에 대한 배포를 자동화할 수 있으며, 이로써 신속한 전달과 배포를 통해 최소한의 노력으로 새로운 코드에 대한 배포를 자동화할 수 있으며, 이로써 신속한 애플리케이션 제공의 속도 저하를 유발하는 수동 프로세스로 인한 운영팀의 프로세스에 대한 과부하 문제를 해결할 수 있도록 도움을 주는 역할을 함.

3. AWS와 DevOps를 위한 개발 Toolkits

1. CI/CD 란?



3. AWS와 DevOps를 위한 개발 Toolkits

2. CI 구성요소

구 분	내 용
리포지토리 관리	<ul style="list-style-type: none">소스코드 및 마스터 리포지토리 관리를 위한 버전 관리 솔루션 필요다양한 파일의 버전 관리를 통해 소스코드의 개발 내역 관리체크아웃 기능을 통해 빌드 가능한 코드 확인 가능현재 사용 중인 소스의 버전 관리 및 모든 변경 사항을 리포지토리에 저장
빌드 자동화	<ul style="list-style-type: none">한 번의 빌드 실행으로 전체 시스템에 대한 빌드 작업 수행팀은 빌드 스크립트 또는 표준화된 빌드 도구를 활용하여 빌드 수행빌드 자동화에는 소스코드 통합, 코드 컴파일, 유닛 및 통합 테스트, 프로덕션과 같은 환경 배포 포함.버전 제어 시스템에서도 이전 빌드의 백업본 유지빌드 중 오류 발생 시 이전 버전으로 복구 수행 및 소스코드 비교를 통해 원인 분석 작업 수행
셀프 테스트	<ul style="list-style-type: none">오류/버그의 신속하고 효과적인 빌드를 위해 테스트 자동화 포함 필요자동화된 테스트 시스템을 통해 실패한 테스트 정보 확인 및 대응 수행자체 테스트 실패 시 전체 테스트도 실패로 처리
반영(commit)을 통한 빌드	<ul style="list-style-type: none">소스 반영(commit)의 정기적인 수행 시 개발자 간 이슈의 최소화 가능1일 1회 이상의 소스 변경 반영(commit) 권장테스트 상황과 애플리케이션의 상태 확인 가능팀은 빈번한 반영(commit)을 통해 더 많은 테스트된 빌드 취득

3. AWS와 DevOps를 위한 개발 Toolkits

3. CI를 통한 기대효과

구 분	내 용
개발자 생산성 향상	<ul style="list-style-type: none">• 개발자의 수동 작업에 대한 부담 감소• 고객에게 제공되는 오류 및 버그 수를 줄이는 데 도움이 되는 도구를 활용 가능• 팀의 생산성 향상 기대
버그를 더 빠르게 발견하고 해결	<ul style="list-style-type: none">• 테스트를 보다 빈번하게 수행할 수 있음• 버그가 더 큰 문제로 발전하기 전 조기 해결 가능
업데이트를 빠르게 제공	<ul style="list-style-type: none">• 팀이 좀더 빠르고 빈번하게 고객에게 업데이트 제공 가능

3. AWS와 DevOps를 위한 개발 Toolkits

4. Continuous Delivery/Continuous Deployment 프로세스

구 분	구성 요소	내 용
Continuous Delivery	빌드 자동화	<ul style="list-style-type: none"> 운영 환경 배포에 필요한 빌드는 자동화 도구 활용
	Continuous Integration	<ul style="list-style-type: none"> 개발자가 공유 리포지토리에 코드 체크인 빌드 자동화 도구가 체크인을 확인하고, 오류, 버그, 발급자 정보를 확인하며, 팀에게 진행 결과에 대한 Report 수행 반복적이며, 지속적인 S/W의 Delivery 수행
	테스트 자동화	<ul style="list-style-type: none"> 새로운 버전의 응용 프로그램 품질과 기능에 대해 점검 파이프라인을 통한 보안, 성능, 컴프라이언스 검증 다양한 자동화도니 활동 수행
	배포 자동화	<ul style="list-style-type: none"> 고객에게 새로운 기능을 수분 이내에 신뢰성 있게 제공
Continuous Deployment	리포지토리 체크인	<ul style="list-style-type: none"> 개발 작업 완료 후 리포지토리에 체크인 수행
	통합 테스트	<ul style="list-style-type: none"> CI 서버에서 변경사항을 선택하여 소스 병합 Unit 테스트 결과를 통해 스테이징 환경에 반영 후 QA 테스트 수행
	운영 환경 배포	<ul style="list-style-type: none"> QA 테스트 통과 시 빌드가 운영 환경으로 전달 CI 서버에서 운영 환경에 빌드 병합 여부 최종 확정 및 배포

3. AWS와 DevOps를 위한 개발 Toolkits

5. Continuous Delivery/Continuous Deployment 기대 효과

구 분	구성 요소	내 용
Continuous Delivery	배포 위험 최소화	<ul style="list-style-type: none"> • 변경 사항을 작은 조각으로 배포함으로써 배포 위험의 최소화 • 문제 발생 시 빠르고 손쉽게 복원 가능
	프로세스 모니터링	<ul style="list-style-type: none"> • 진행 절차나 프로세스 모니터링 가능 • 운영 환경과 동일 환경에서 전체 프로세스 사전 점검 가능
	사용자 피드백	<ul style="list-style-type: none"> • 작게 구성하여 빠르게 사용자 피드백 확보 • 실사용자로부터 애플리케이션의 유용성에 대한 빠른 피드백
Continuous Deployment	생산성 및 품질 향상	<ul style="list-style-type: none"> • 제품에 대한 집중도가 높아지며, 업무 효율 및 생산성 향상 • 테스트에 집중하고, 반복 작업 자동화를 통한 품질 향상
	업무 프로세스 개선	<ul style="list-style-type: none"> • 통합 파이프라인을 통해 팀과 프로세스 간 통합 • 개발, 테스트 및 운영 환경 전반에서 워크플로우 생성 • 기존 톨과의 연동을 통해 간단하게 작업 수행 가능

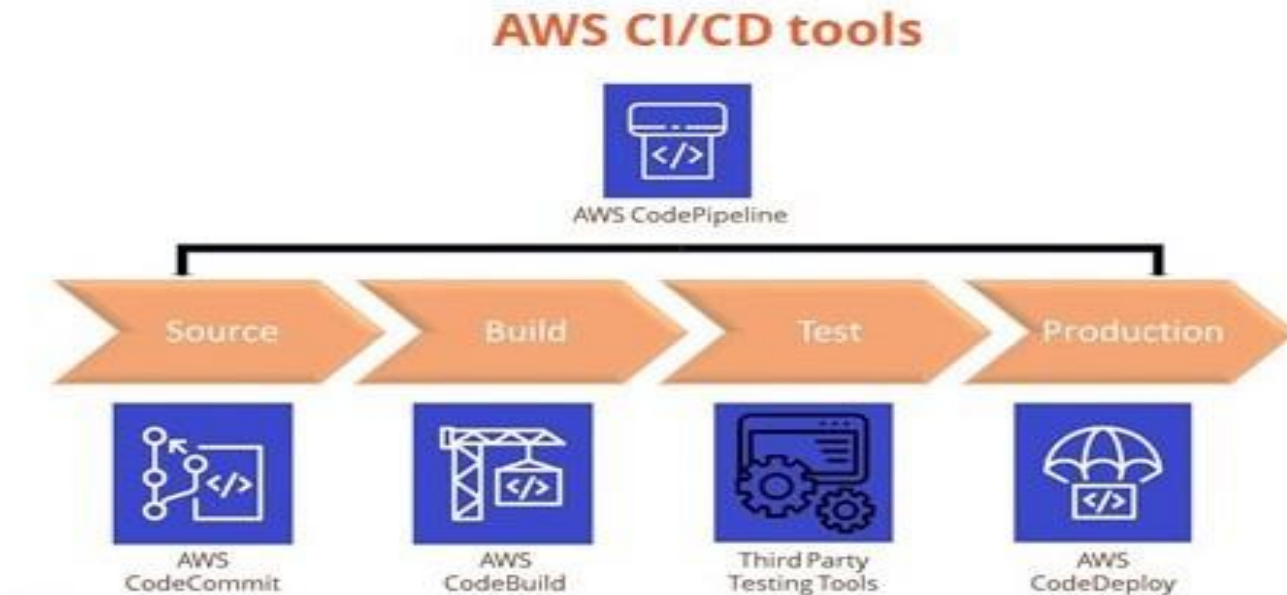
3. AWS와 DevOps를 위한 개발 Toolkits

6.CI/CD를 위한 도구들

구분	내용
소스코드 관리	GIT, Bitbucket, Subversion
빌드 자동화 도구	Maven, Ant, Gradle
테스트 자동화 도구	Selenium, JUnit, Cucumber
CI 도구	Jenkins, Bamboo, Hudson
구성 관리 도구	Puppet, Chef, Ansible
모니터링 도구	Nagios, Ganglia, Sensu

3. AWS와 DevOps를 위한 개발 Toolkits

7. 성공적인 DevOps를 위한 AWS의 도구들



3. AWS와 DevOps를 위한 개발 Toolkits-AWS Cloud9

1. AWS Cloud9이란?

웹 브라우저만으로 코드의 개발 및 실행 디버깅을 수행할 수 있는 클라우드 기반 IDE(Integrated Development Environment).

코드 개발을 위한 웹 기반의 에디터와 디버거, 터미널을 자체 내장하고 있으며, JavaScript, Python, PHP, C, C++, 루비, 펄, Go를 포함한 40여 개의 다양한 프로그래밍언어를 지원.

이런 프로그래밍 언어를 위한 필수 도구가 사전에 패키징되어 제공되므로, 새로운 프로젝트를 위해 프로그램을 설치하거나 환경 설정을 위한 시간을 투자할 필요가 없음.

서버리스 애플리케이션을 개발할 수 있는 원활한 환경을 제공하므로 손쉽게 서버리스 애플리케이션의 리소스를 정의하고, 디버깅하고, 로컬 실행과 원격 실행 간의 전환을 할 수 있으며, 서버리스 개발에 필요한 모든 SDK, 라이브러리 및 플러그인으로 개발 환경을 사전에 구성하여, 서버리스 개발을 보다 손쉽게 수행할 수 있는 개발 환경을 제공.

AWS Cloud9은 추가적인 비용이 없음. 다만, AWS Cloud9을 구동하기 위해 필요한 EC2 인스턴스와 EBS에 대한 비용만 지불.

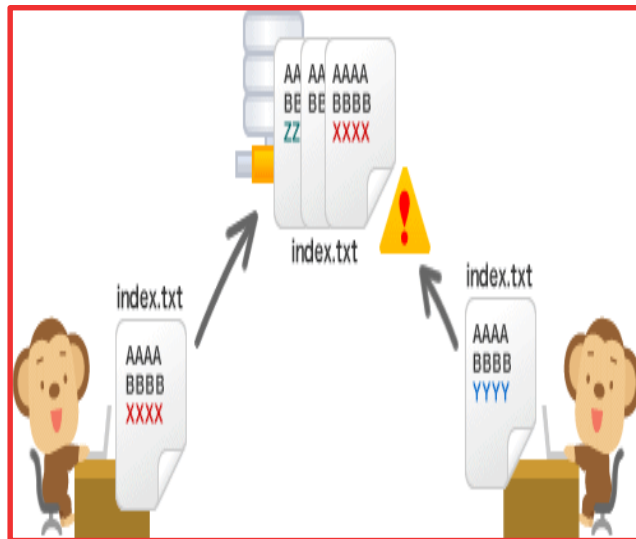
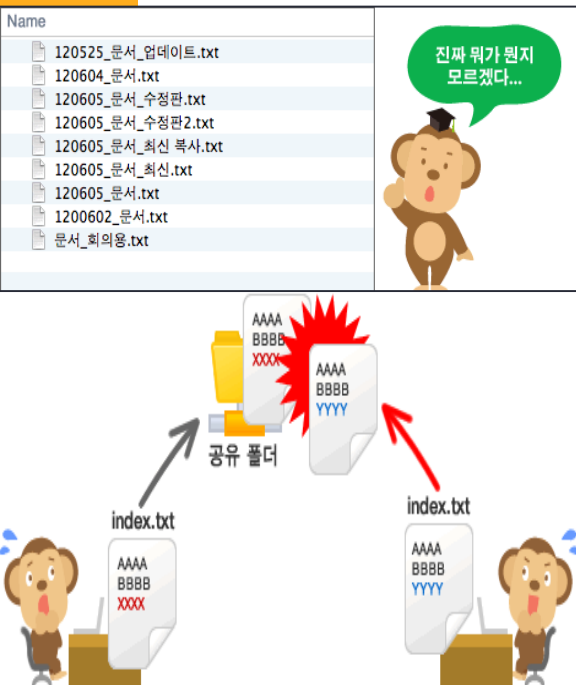
3. AWS와 DevOps를 위한 개발 Toolkits-AWS CodeCommit

1. AWS CodeCommit이란?

- 안전한 Git 기반의 리포지토리를 클라우드 기반으로 제공하는 완전 관리형 소스 제어 서비스.
- 일반적인 애플리케이션 개발을 진행하면서 개발된 소스를 저장하고 제어할 수 있는 기능을 제공.
- 관리형 클라우드 서비스이므로 소스코드를 저장하기 위한 리포지토리를 생성만 하면 되며, 프로비저닝 및 확장할 하드웨어나 설치, 구성, 운영할 소프트웨어가 없음.
- AWS CodeCommit의 가져오기 요청, 분기 및 병합 기능을 활용하면 팀에서 다른 개발자나 구성원과 협업을 할 수 있으며, 이를 통해 보다 효율적으로 소스코드를 관리할 수 있음.
- HTTPS, SSH를 활용하여, 파일을 송수신할 수 있으며, 소스 저장에 사용되는 리포지토리는 Key Management Service(KMS)를 통해 저장 중 자동으로 데이터를 암호화함.
- AWS IAM과 CloudTrail, CloudWatch와 연동을 통해 누가 데이터에 액세스할 수 있는지, 액세스 방법, 시기 및 위치까지 제어하고 모니터링할 수 있어서 데이터에 대한 보안과 접근을 제어할 수 있음.

3. AWS와 DevOps를 위한 개발 Toolkits-AWS CodeCommit

소스 관리 시스템 Git 란?



- 리눅스 토발즈가 2005년 개발한 오픈소스 기반의 소스 관리 툴
- 작업 폴더에서 일어나는 모든 기록들과 각각의 기록을 추적할 수 있는 정보를 포함하고 있는 저장소 역할을 수행하는 프로그램.

3. AWS와 DevOps를 위한 개발 Toolkits-AWS CodeCommit

CodeCommit 실습순서

1.

- CodeCommit 실행을 위한 계정 생성 및 Git HTTP 접속을 위한 자격 증명 생성

2.

- Cloud9 과 CodeCommit의 리포지토리에 접속 및 서비스 이용에 대한 방법 진행
- 간단한 Git 명령을 활용하여 소스를 CodeCommit으로 동기화하는 방법 진행

3.

- GitHub에서 관리하던 Source를 CodeCommit으로 Migration하는 방법 진행



3. AWS와 DevOps를 위한 개발 Toolkits-AWS CodeBuild

1. AWS CodeBuild이란?

- 소프트웨어 개발에 필요한 소스코드를 컴파일하는 단계에서부터 테스트 후 소프트웨어 배포까지의 단계를 지원하는 완전 관리형 지속적 통합(CI) 서비스.
- 자체 빌드 서버가 필요하지 않으며, 빌드 서버를 프로비저닝하거나, 운영/관리 및 확장을 수행할 필요가 없음.
- 빌드를 수행할 볼륨에 따라 인프라가 자동으로 확장 및 축소를 지원하여, 제출되는 빌드에 대해 즉각적으로 처리되므로, 여러 빌드를 동시에 처리할 수 있기 때문에 빌드를 위해 대기하지 않고 빠르고 효율적으로 빌드 작업을 수행할 수 있음.



3. AWS와 DevOps를 위한 개발 Toolkits-AWS CodeBuild

CodeBuild 실습순서

1.

- Cloud9, CodeCommit를 활용하여 Vue.js 설치 및 프로젝트 생성

2.

- Vue.js의 소스를 S3 정적 호스팅에 필요한 설정으로 수행
- S3 버킷 생성과 정적 호스팅 설정
- 외부 액세스 설정을 위한 권한 설정

3.

- Vue.js의 소스로 S3 정적 호스팅을 위해 CodeBuild 설정 진행
- Cloud9에서 작성된 Vue.js 소스가 S3 정적 웹 페이지에 빌드되고 배포되는지 확인

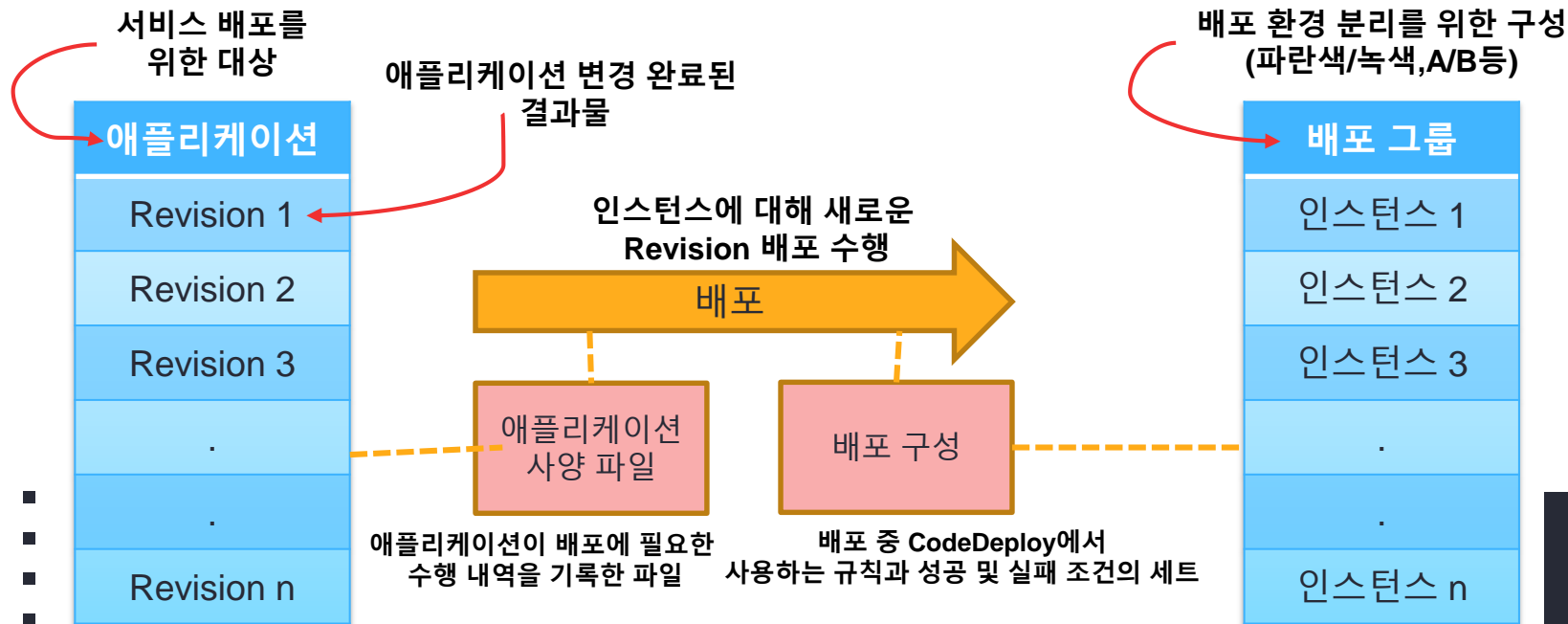
3. AWS와 DevOps를 위한 개발 Toolkits-AWS CodeDeploy

1. AWS CodeDeploy란?

- Amazon EC2, Amazon ECS, AWS Lambda 및 On-Premise 서버와 같은 다양한 컴퓨팅 서비스에 대해 소프트웨어 배포를 자동화하여 제공하는 완전 관리형 배포 서비스다.
- 지속적인 배포(CD)를 지원하는 대표적인 CD 도구로써, 새로운 기능 및 문제가 되는 코드에 대해 빠르고 신속하게 배포할 수 있다.
- AWS Console 또는 AWS CLI를 통해 배포를 시작하고, 배포 상태에 대해 추적을 수행할 수 있으며, 상세한 보고서를 통해 애플리케이션의 수정 버전이 언제 어디에 배포되었는지 확인할 수 있다.
- 또한, 진행 과정에서 오류가 발생하게 되면 손쉽게 소프트웨어의 배포를 중단하고 롤백을 수행할 수 있다.

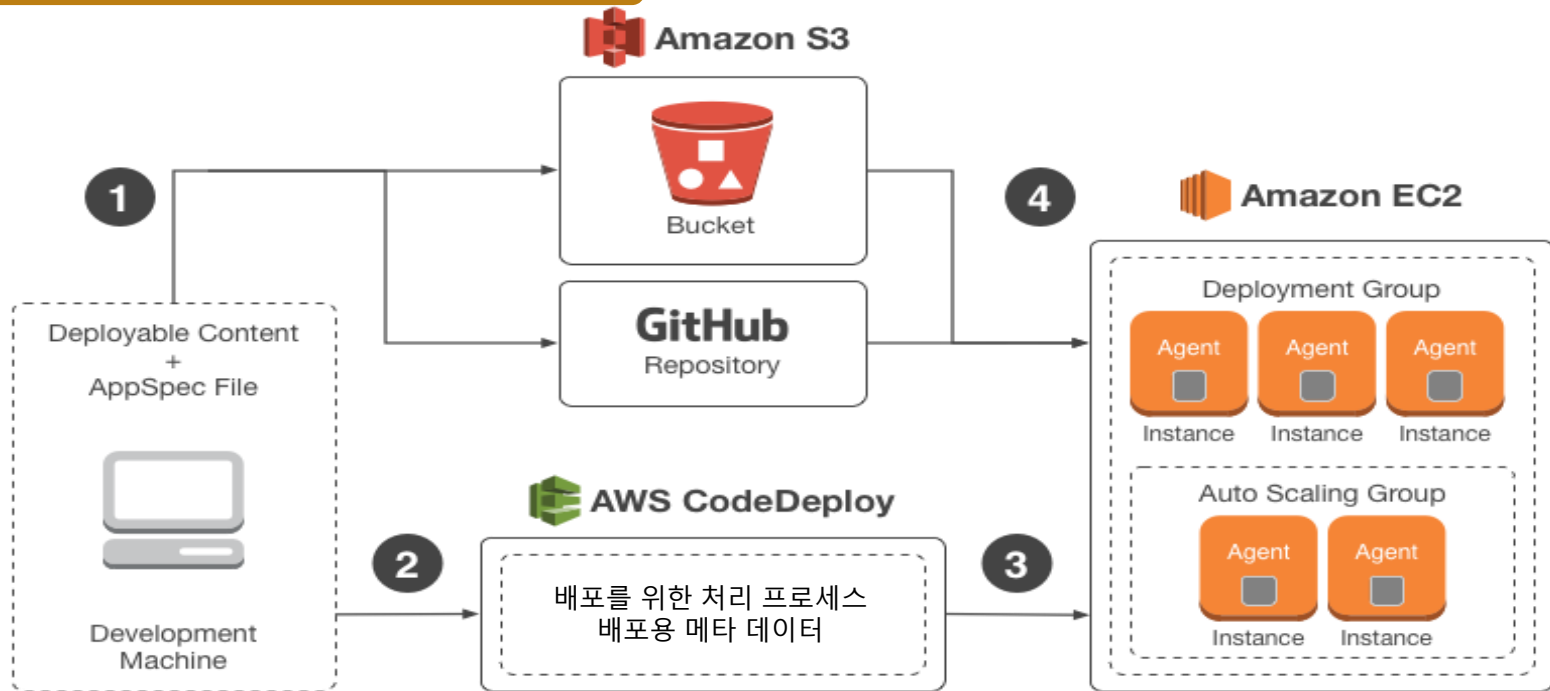
3. AWS와 DevOps를 위한 개발 Toolkits-AWS CodeDeploy

2. AWS CodeDeploy의 구성 요소



3. AWS와 DevOps를 위한 개발 Toolkits-AWS CodeDeploy

3. AWS CodeDeploy의 동작 방식



3. AWS와 DevOps를 위한 개발 Toolkits-AWS CodeDeploy

AWS CodeDeploy 실습 순서

1.

•EC2 Auto Scaling 그룹에 소스를 배포하기 위해 IAM에 CodeDeploy 서비스 역할과 EC2에 부여할 IAM 인스턴스 프로파일 생성

2.

•EC2 인스턴스 IAM 역할을 활용하여 Auto Scaling 시작 구성을 진행

3.

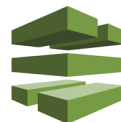
•Auto Scaling으로 구성된 EC2에 배포할 웹 페이지와 CodeDeploy 배포를 위해 필요한 AppSpec.yml 작성

4.

•EC2 Auto Scaling 으로 소스를 배포하기 위해 CodeDeploy 애플리케이션을 생성 후 배포

5.

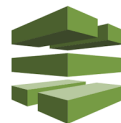
•변경된 소스에 대해 EC2에 소스 배포 수행과 Auto Scaling에서 서버 추가로 인해 신규로 생성된 인스턴스에 대해 소스 배포



3. AWS와 DevOps를 위한 개발 Toolkits-AWS CodePipeline

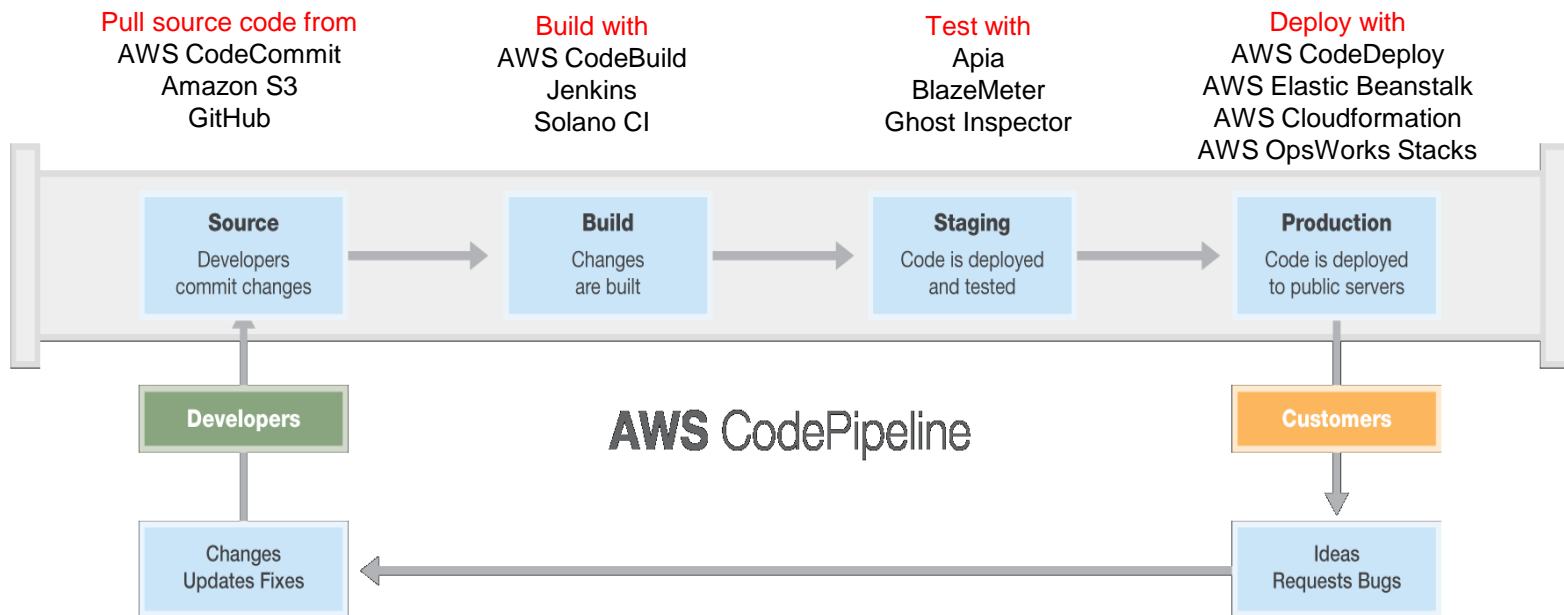
1. AWS CodePipeline 이란?

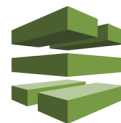
- 빠르고 안정적으로 애플리케이션과 인프라의 업데이트를 위한 릴리스 파이프라인을 자동화할 수 있도록 서비스 형태로 제공되는 완전 관리형 지속적인 전달(Continuous Delivery)을 제공하는 서비스이다.
- 코드에 변경이 있을 때마다 사용자가 사전에 정의한 릴리스 모델을 기반으로 애플리케이션의 릴리스 프로세스의 빌드, 테스트, 배포 단계에 걸친 프로세스에 대해 자동화를 수행한다. 이를 통해 애플리케이션의 변경 사항이나 새로운 기능을 신속하고 안정적으로 제공할 수 있다.
- AWS CodeCommit, GitHub, Amazon ECR(Elastic Container Registry) 또는 Amazon S3에서 바로 파이프라인에 대한 소스코드를 가져올 수 있으며, AWS CodeBuild에서 빌드와 단위 테스트를 실행할 수 있다.



3. AWS와 DevOps를 위한 개발 Toolkits-AWS CodePipeline

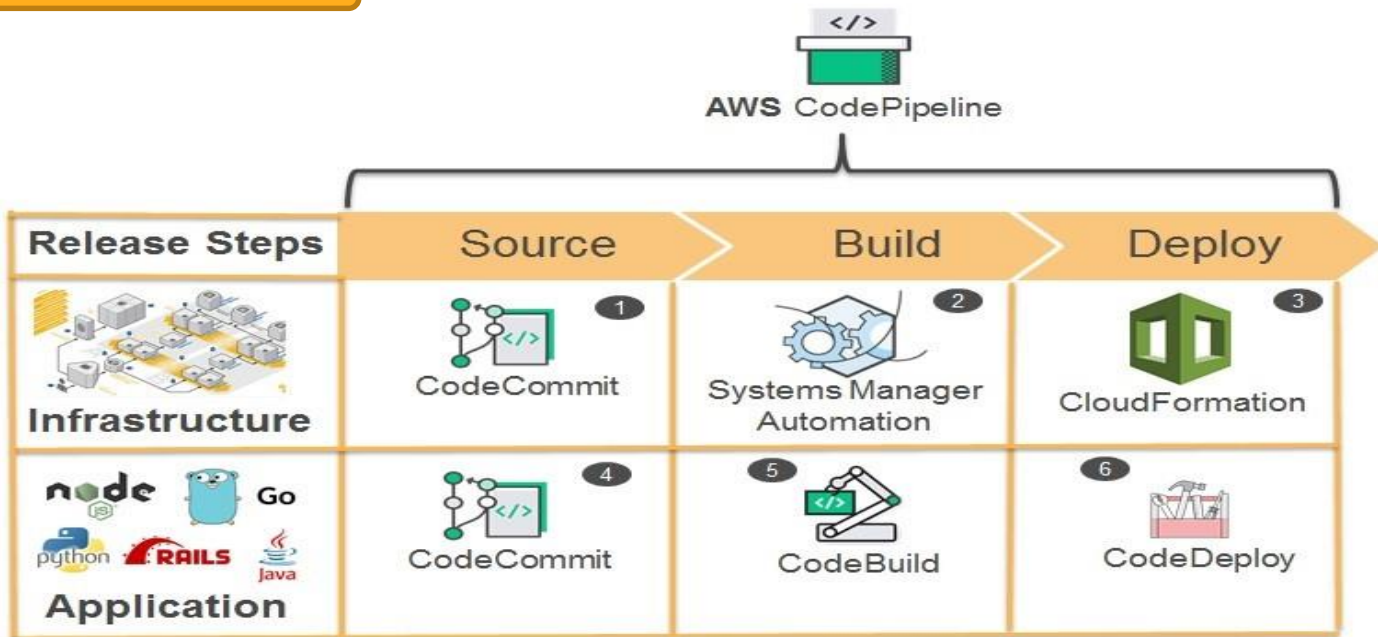
1. AWS CodePipeline이란?





3. AWS와 DevOps를 위한 개발 Toolkits-AWS CodePipeline

1. AWS CodePipeline이란?





3. AWS와 DevOps를 위한 개발 Toolkits-AWS CodePipeline

AWS CodePipeline 실습 순서

1.

•CodeCommit, CodeDeploy, CodePipeline에서 사용할 사용자 계정과 역할, 정책 생성

2.

•IAM 계정과 CodeCommit 인증을 활용하여 Cloud9과 CodeCommit에 Sample 코드를 등록하고 CodeDeploy를 이용하여 배포할 수 있도록 사전 준비 작업 진행

3.

•CodeCommit으로 등록한 소스코드를 업로드할 EC2 생성과 CodeDeploy의 배포 그룹 생성

4.

•생성한 EC2와 CodeDeploy를 활용하여 CodePipeline을 생성하고 CodeCommit과 CodeDeploy를 활용하여 소스 배포 파이프라인을 구현
•CodeCommit으로 새롭게 'git push'가 요청되었을 때 CodePipeline의 처리과정 확인