AWS 200: DevOps 서비스

AWS 솔루션즈 아키텍트 이민우

2020.07.15

Agenda

- **DevOps** 개념
- **DevOps** 를 위한 도구
  - 통합개발환경 **(IDE)**
  - **Code Repository**
  - 지속적 통합 **(CI)**
  - 코드로서의 인프라 **(IaC)**
  - **Deployment**
  - **Debugging**
  - 파이프라인 관리
- 데모(실습)

DevOps 개념

DevOps란 무엇인가?

문화적 철학

관행

도구들(**Tools**)

책임 공유

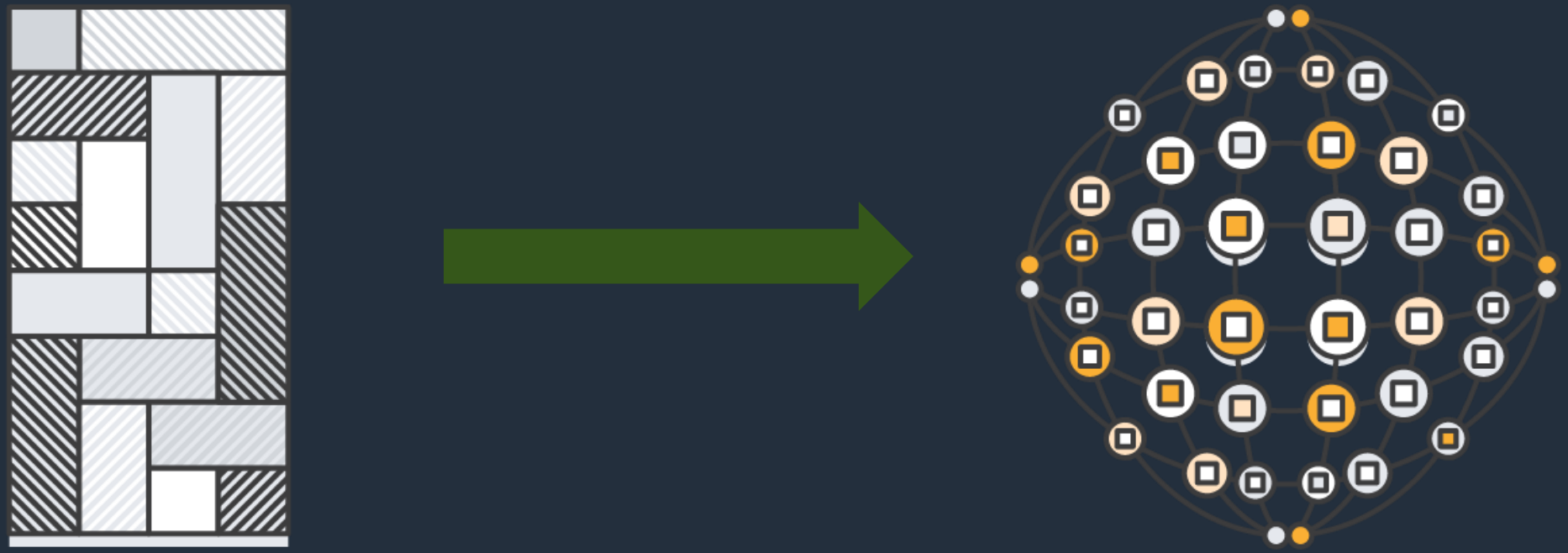
Ownership

가시성 및 의사소통

DevOps 관행

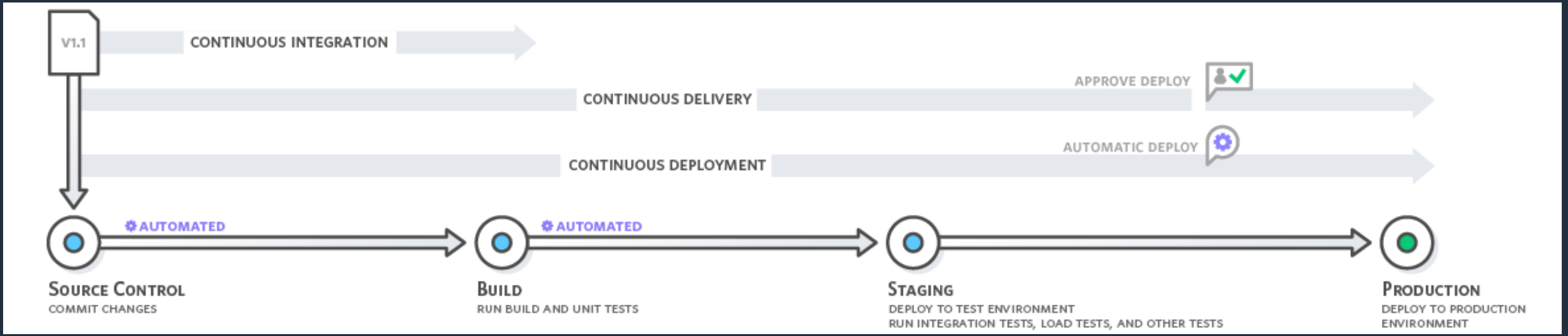
마이크로서비스 (Microservices)

- **Monolithic** 어플리케이션 아키텍처에서 여러개의 개별 서비스로 전환



DevOps 관행

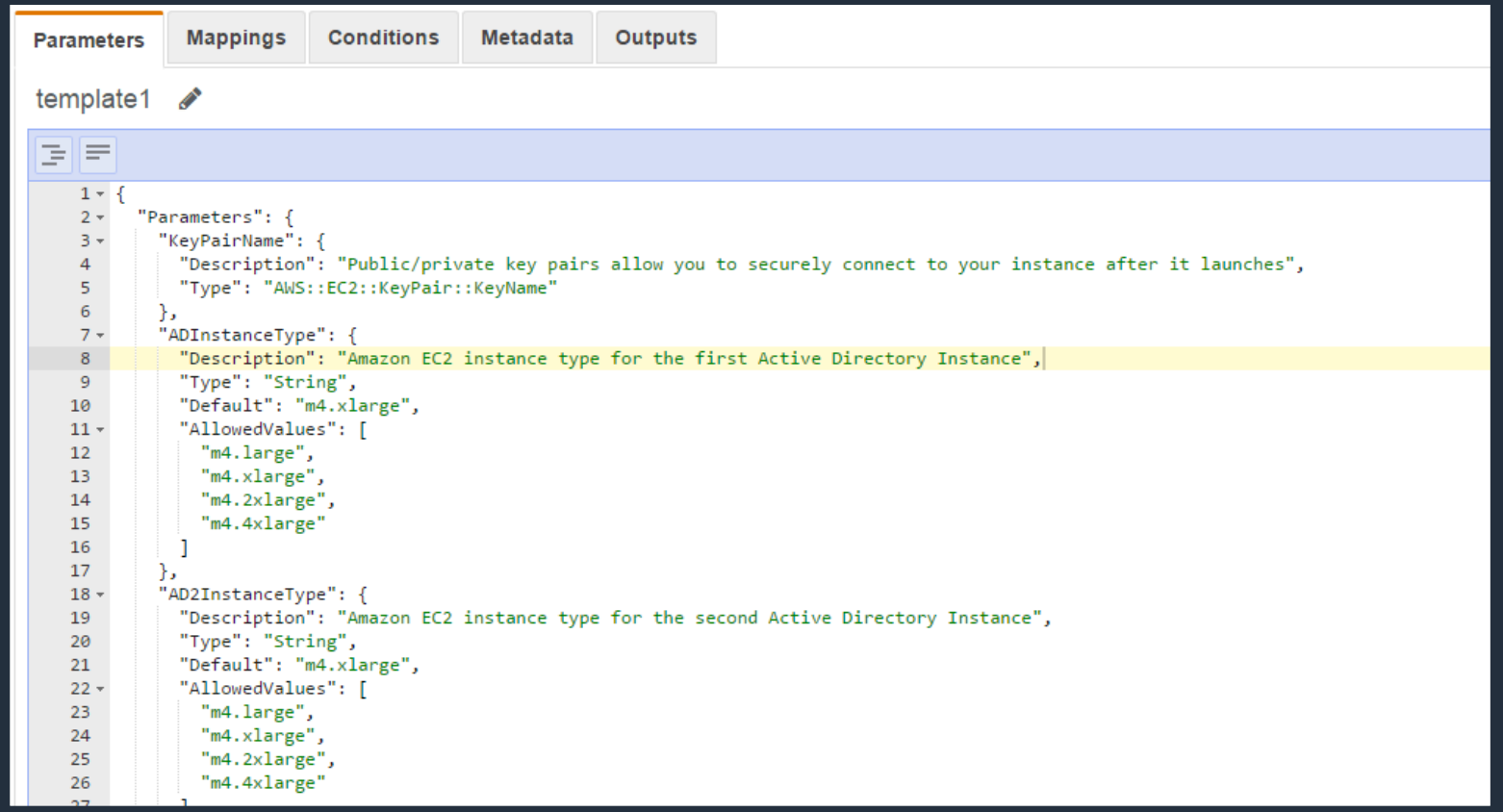
- 지속적 통합 (**Continuous integration**)
- 지속적 전달/배포 (**Continuous delivery & deployment**)



DevOps 관행

Infrastructure as code

- 코드를 사용하여 **AWS** 리소스를 모델링



DevOps 관행

- 모니터링과 로깅
  - 지표(Metric) 와 로그들을 추적하고 분석
  - 인프라스트럭처와 어플리케이션의 실시간 성능을 이해

DevOps 의 이점

협업 개선

빠른
전달/배포

신뢰성

보안

확장성

속도

숫자로 보는 DevOps 이점

7x
더 적은 변경 실패

106x
더 빠른 **Commit to Deploy**

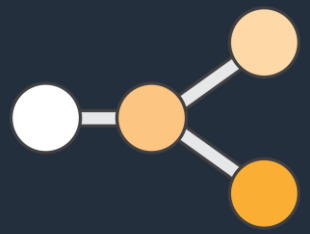
208x 더
빈번한 배포

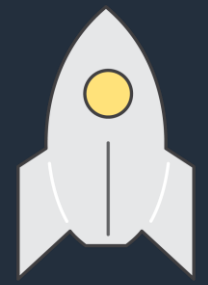
2,604x
사고로 부터 더 빠른 복구

Source: The State of DevOps Report 2019

릴리스 및 모니터의 5 가지 주요 단계

소스	빌드	테스트	배포	모니터
• .java 파일과 같은 소스 코드 체크인 • 신규 코드의 Peer review	• 코드 컴파일 • 단위 테스트 • Style checkers • Code 지표 • 컨테이너 이미지 생성	• 다른 시스템과의 통합 테스트 • 부하 테스트 • UI 테스트 • 침투 테스트	• 프로덕션 환경에 배포	• 비정상적인 활동이나 오류를 신속하게 감지하기 위해 프로덕션 환경에서 코드 모니터링



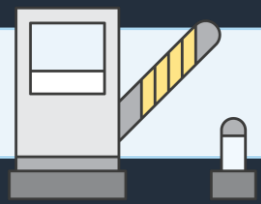




릴리스 프로세스 수준

소스 → 빌드 → 테스트 → 프로덕션

지속적 통합: **Continuous integration**

지속적 전달: **Continuous delivery**



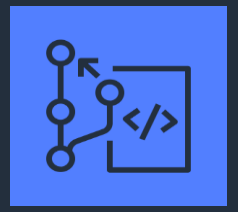
지속적 배포: **Continuous deployment**

AWS 에서의 CI/CD 파이프라인

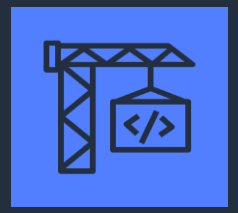
AWS CodePipeline

소스 | 빌드 | 테스트 | 배포 | 모니터

AWS Cloud9



AWS CodeCommit

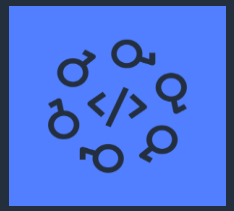


AWS CodeBuild

Third-party tooling

AWS CodeDeploy

AWS X-Ray



AWS CodeStar

프로젝트 관리

Amazon CloudWatch

AWS DevOps 포트폴리오

소프트웨어 개발과 지속적 전달
Toolchain


AWS CodeCommit


AWS CodeStar


AWS CodeBuild


AWS CodeDeploy


AWS CodePipeline


AWS Cloud9

코드로서의 인프라
(IaC)


AWS CloudFormation


AWS OpsWorks


AWS Cloud Development Kit


AWS OpsWorks for Chef Automate

모니터링 및 로깅


AWS X-Ray


Amazon CloudWatch


AWS CloudTrail


AWS Config

통합개발환경 (IDE)

통합개발환경(IDE) 선택

AWS Toolkit for PyCharm

Python

AWS Toolkit for IntelliJ

Java, Python

AWS Toolkit for Visual Studio Code

.NET, Node

AWS Toolkit for Visual Studio

.NET

New

AWS Toolkit for Webstorm

Node.js

AWS Toolkit for Rider

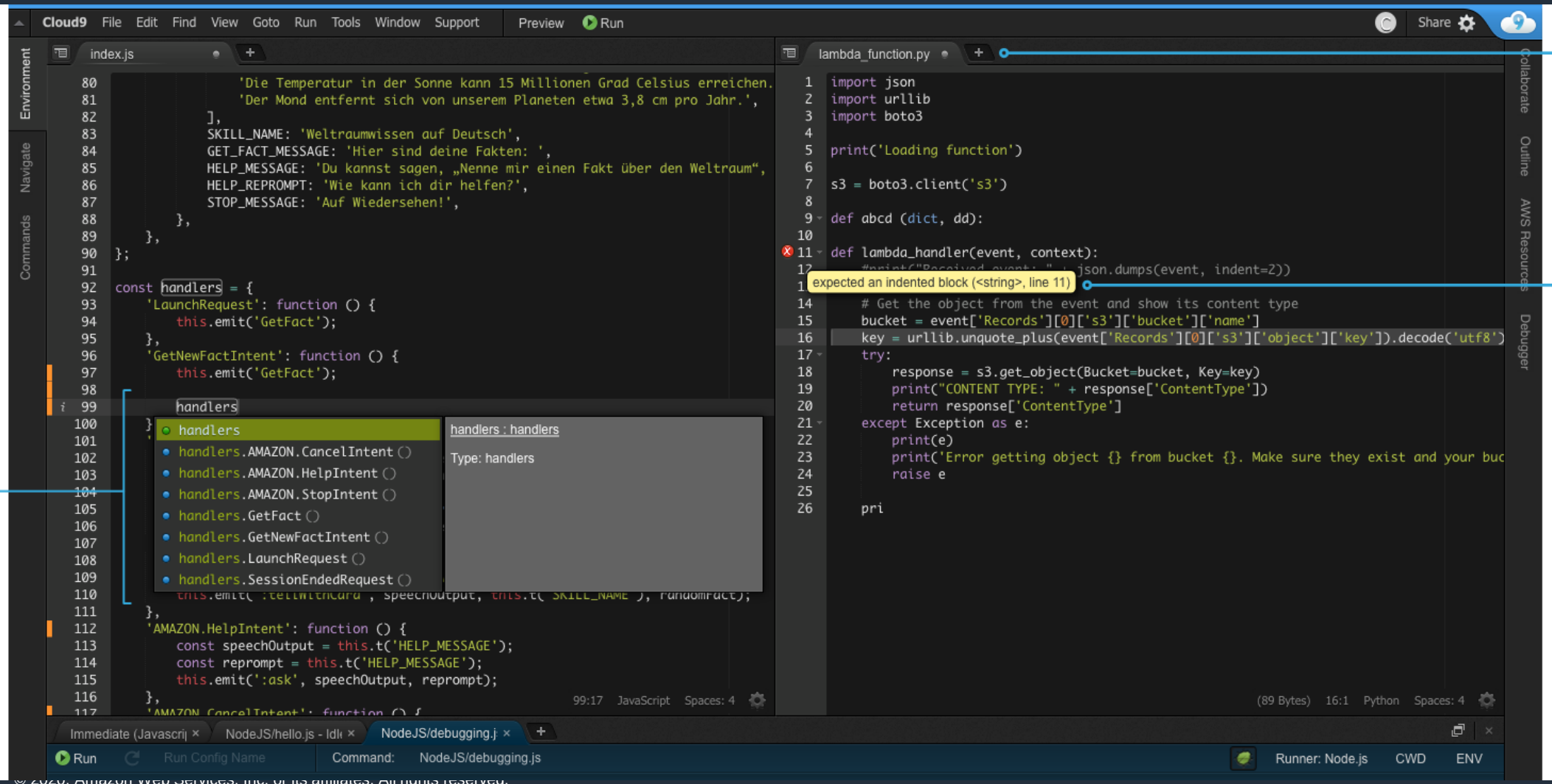
.NET

AWS Cloud9

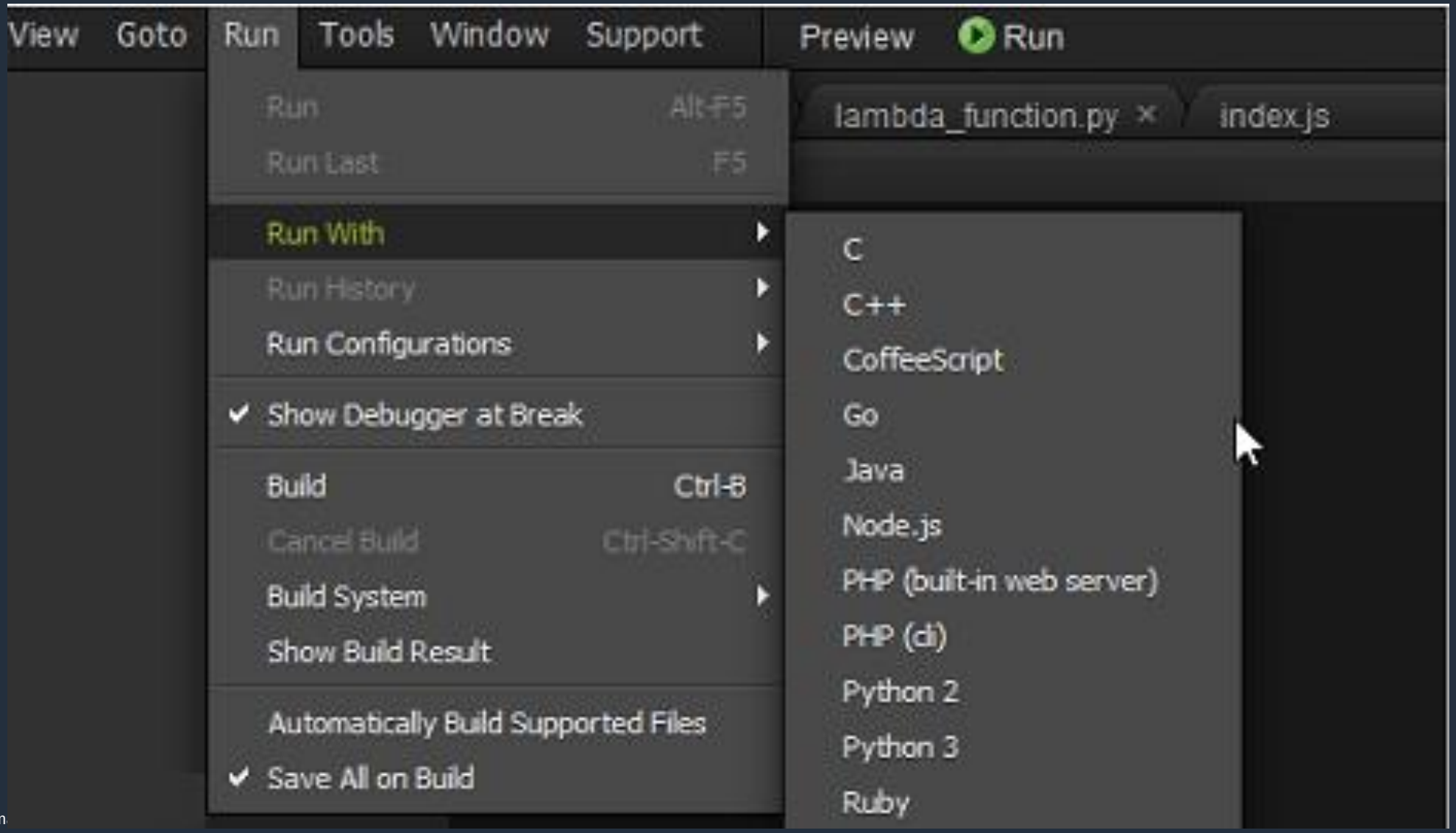
코드 작성, 실행 및 디버깅을위한 클라우드 **IDE**

- 브라우저만으로 코딩
- 실시간으로 코딩
- **Serverless** 어플리케이션을 쉽게 구축
- **AWS**에 직접 터미널 엑세스
- 신규 프로젝트를 빠르게 시작

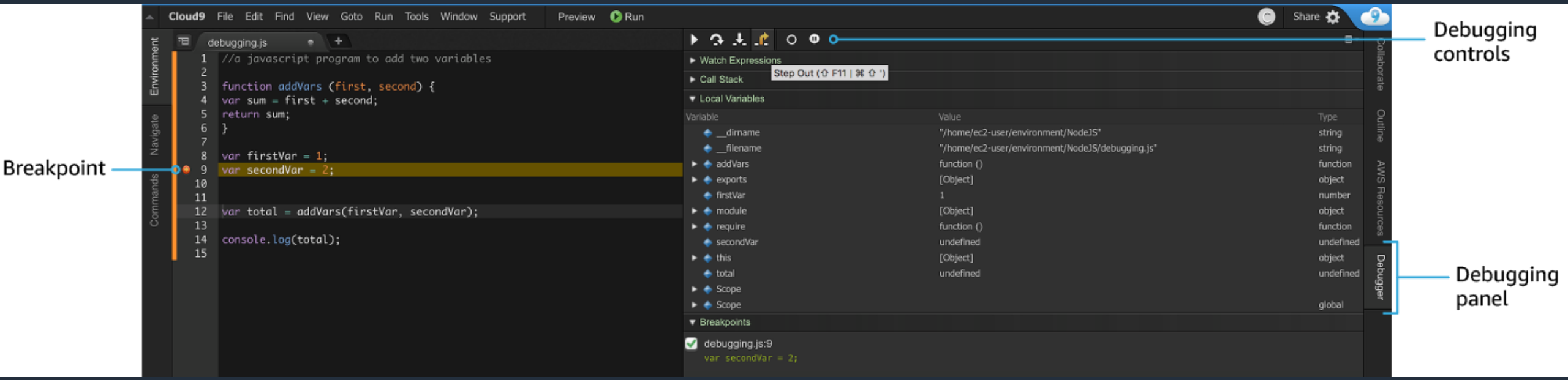
완전 기능화된 편집기



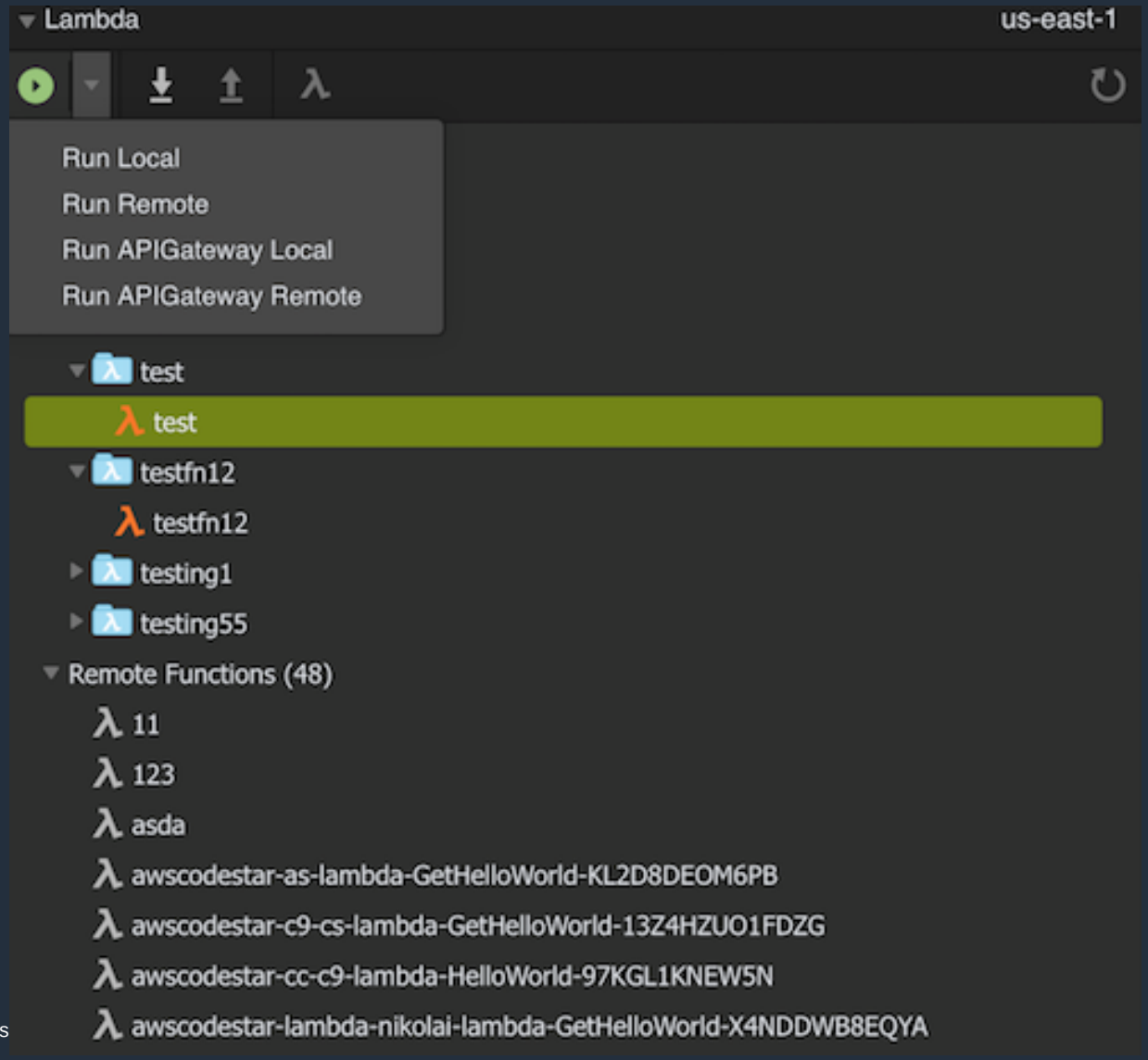
광범위한 Run Time 선택



완전 기능화된 Debugger



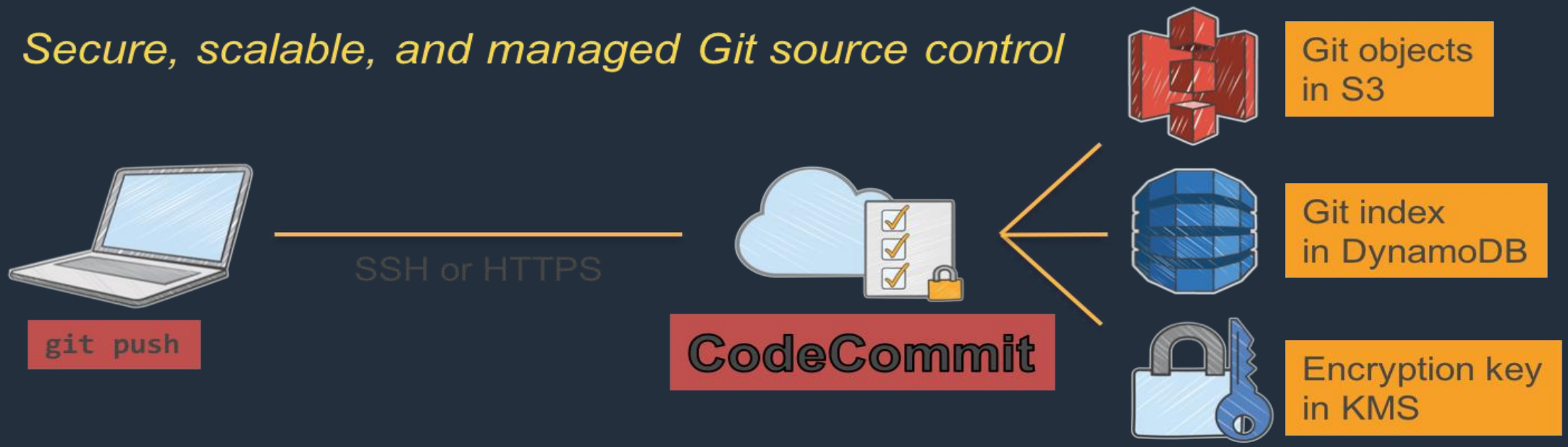
Serverless 개발을 위한 통합 도구



Code Repository

Code Repository 로 코드 이동

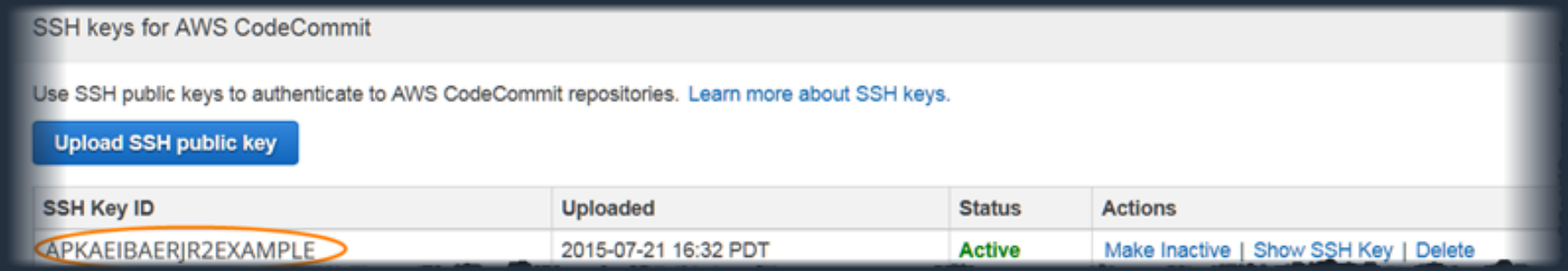
AWS CodeCommit



- **AZ** 간 데이터 이중화
- 저장시 데이터 암호화
- **AWS Identity and Access Management (IAM)** 와 통합
- **Repo** 사이즈 제한 없음

CodeCommit & ssh 시작하기

```
$ ssh-keygen
```

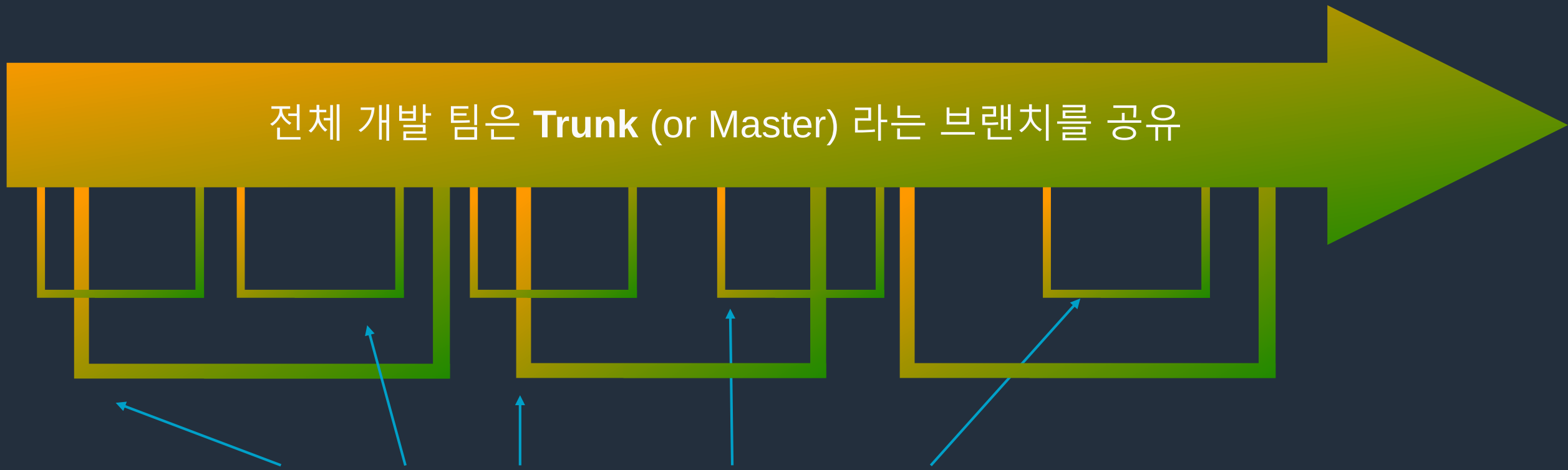



```
$ vi ~/.ssh
        Host git-codecommit.*.amazonaws.com
        User APKAEiBAERJR2EXAMPLE
        identityFile ~/.ssh/codecommit_rsa
```

Git 과 동일한 경험

```
$ git clone https://git-codecommit.us-east-1.amazonaws.com/v1/repos/aws-cli
Cloning into 'aws-cli'...
Receiving objects: 100% (16032/16032), 5.55 MiB | 1.25 MiB/s, done.
Resolving deltas: 100% (9900/9900), done.
Checking connectivity... done.
$ nano README.rst
$ git commit -am 'updated README'
[master 4fa0318] updated README
 1 file changed, 1 insertion(+)
$ git push
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 297 bytes | 0 bytes/s, done.
Total 3 (delta 2), reused 0 (delta 0)
remote:
To https://git-codecommit.us-east-1.amazonaws.com/v1/repos/aws-cli
   4dacd6d..4fa0318  master -> master
```

Branching 전략

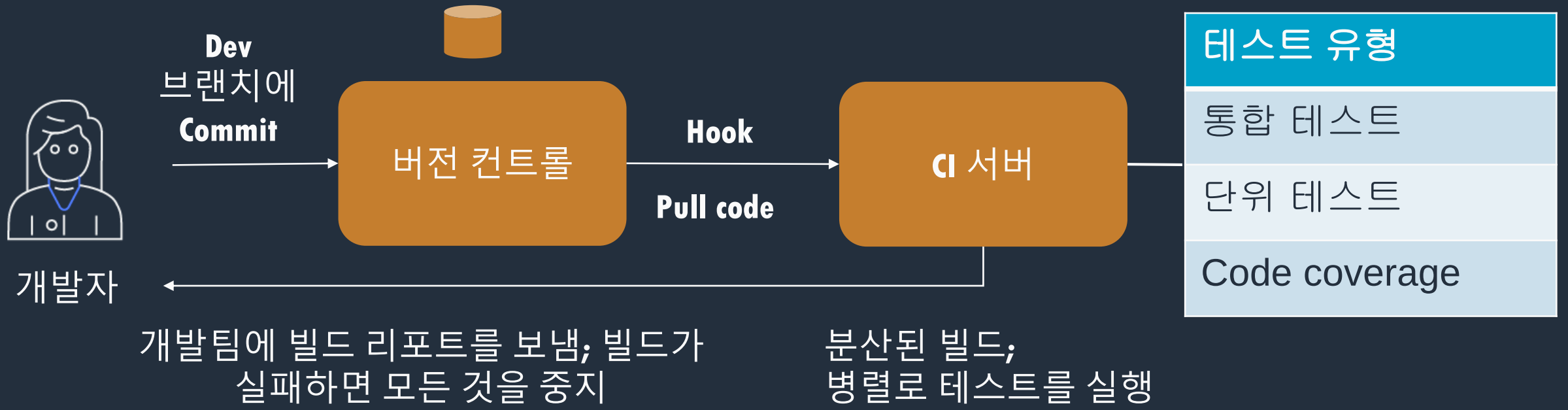


개발자들은 수명이 짧은 기능의 브랜치를 생성하고,

Pull 요청을 통해 병합 (Merge)

지속적 통합 (CI)

지속적 통합 workflow



AWS CodeBuild

지속적인 확장으로 코드 작성 및 테스트

- 소스 코드를 컴파일하고, 테스트를 실행하며, 소프트웨어 패키지를 생성하는 완전 관리형 빌드 서비스
- 지속적으로 확장하고 여러 빌드를 동시에 처리
- **Docker** 이미지를 통해 필요에 맞는 맞춤형(**custom**) 빌드 환경을 제공 할 수 있음
- 사용하는 컴퓨팅 리소스에 대해 분 단위로만 지불
- **AWS CodePipeline** 및 **Jenkins** 통합으로 시작

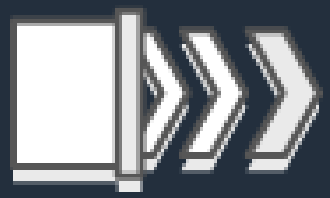
AWS CodeBuild 동작 방법

소스 코드를 컴파일하고, 테스트를 실행하며, 배포 준비가 된 소프트웨어 패키지를 생성하는 완전 관리형 빌드 서비스

소스 코드 다운로드

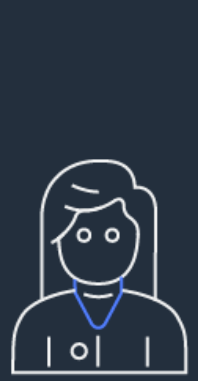
빌드 명령어 실행

빌드 진행상황 모니터링



S3로 아티팩트 업로드

지속적 통합 workflow on AWS



개발자

Dev
브랜치에
Commit

AWS CodeCommit

Hook

Pull code

AWS CodeBuild

테스트 유형
통합 테스트
단위 테스트
Code coverage

개발팀에 빌드 리포트를 보냄; 빌드가 실패하면 모든 것을 중지

분산된 빌드;
병렬로 테스트를 실행

buildspec.yml 예시

```
version: 0.1

environment_variables:
  plaintext:
    JAVA_HOME: "/usr/lib/jvm/java-8-openjdk-amd64"

phases:
  install:
    commands:
      - apt-get update -y
      - apt-get install -y maven
  pre_build:
    commands:
      - echo Nothing to do in the pre_build phase...
  build:
    commands:
      - echo Build started on `date`
      - mvn install
  post_build:
    commands:
      - echo Build completed on `date`
artifacts:
  type: zip
  files:
    - target/messageUtil-1.0.jar
  discard-paths: yes
```

빌드 단계에 사용될 변수

빌드 단계를 명시

- **Dependencies** 설치
- **Pre-build/Post-build** 단계

아티팩트를 생성하고 **S3**에 저장

빌드를 요구하는 것

코드의 “**Building**"은 일반적으로 컴파일 된 바이너리가 필요한 언어를 나타냄:

- **.NET languages: C#, F#, VB.net, etc.**
- **Java and JVM languages: Java, Scala, JRuby**
- **Go**
- **iOS languages: Swift, Objective-C**

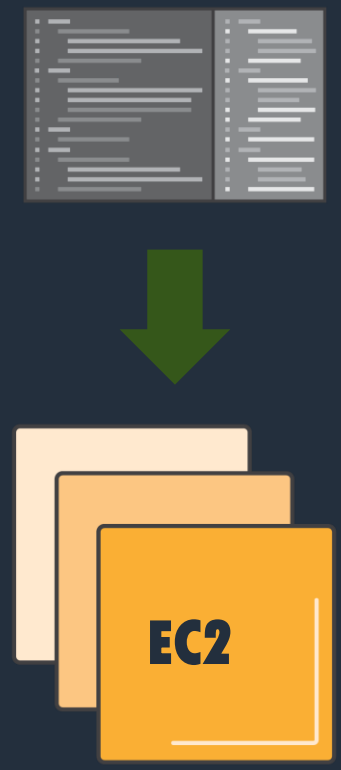
또한 **Docker** 컨테이너 이미지를 만드는 과정을 **Image “Building“** 이라고 함

빌드를 요구하지 않는 것

많은 언어가 빌드를 요구하지 않음. 이들은 해석 된 언어 (**Interpreted**)로 간주됨:

- **PHP**
- **Ruby**
- **Python**
- **Node.js**

단지 코드를 배포하면 됨!



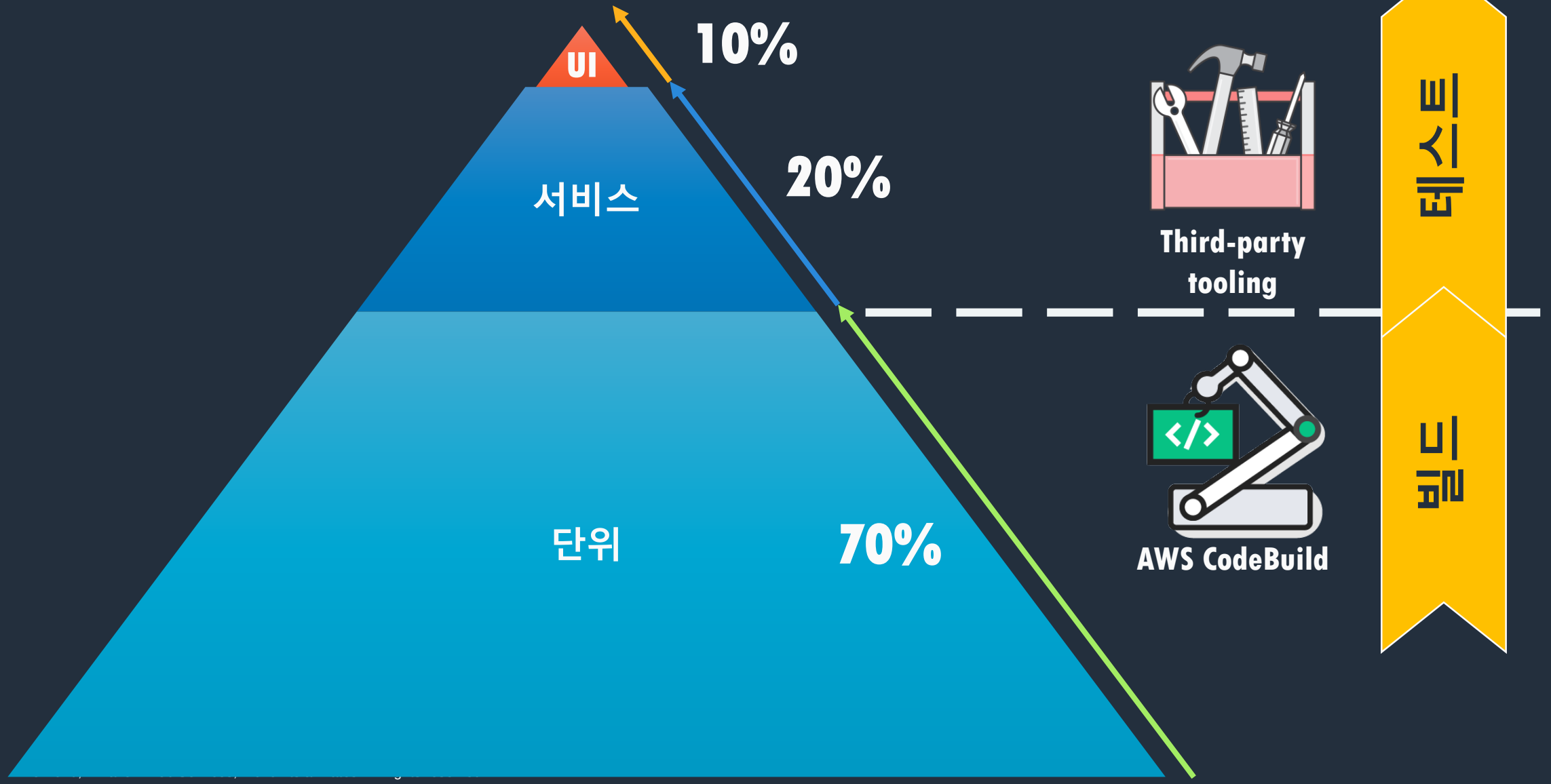
코드 테스트

테스트는 과학과 예술의 형태!

코드 테스트 목표:

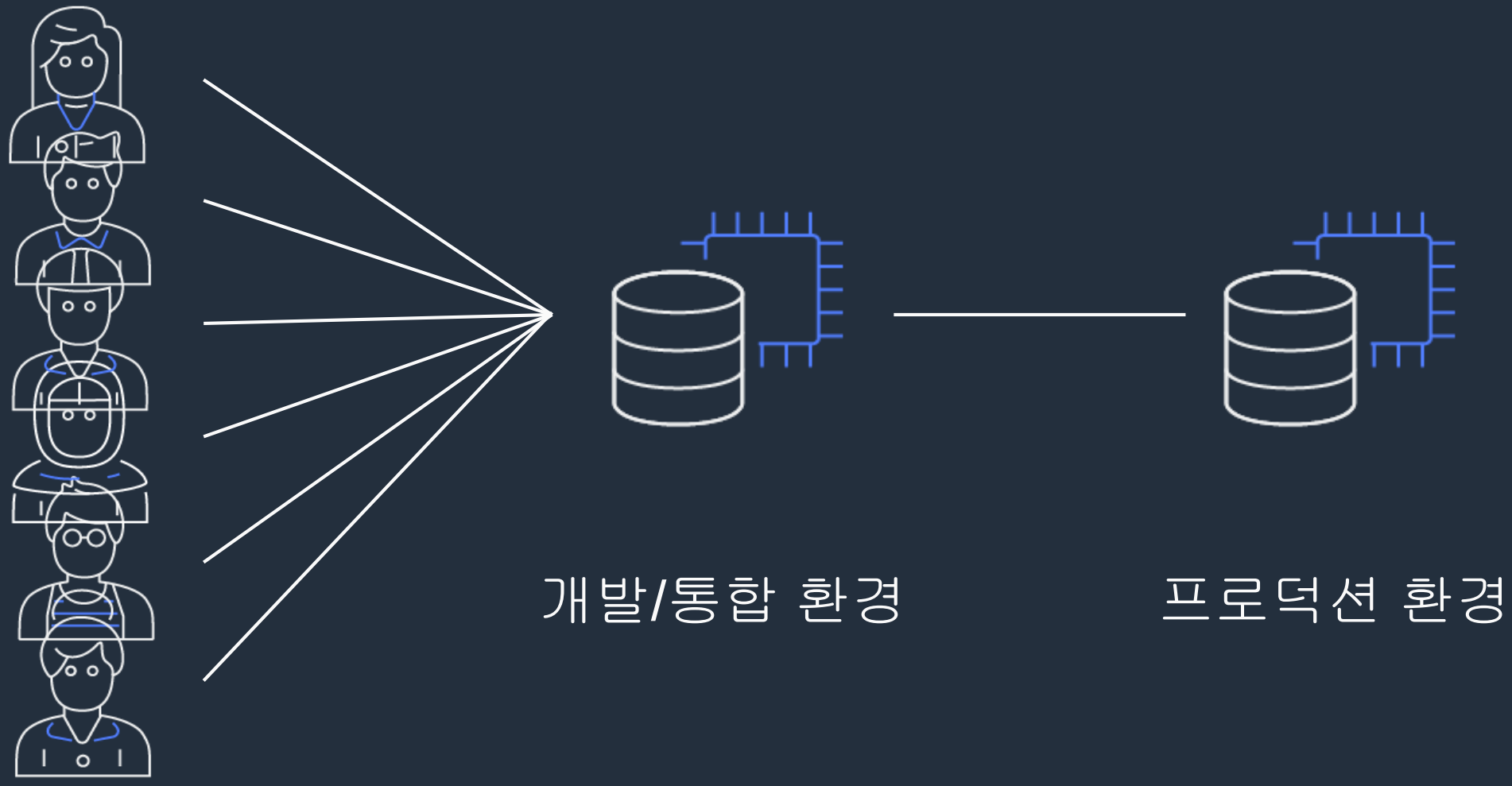
- 의도한 기능을 확인하고 싶음
- 프로그래밍 구문 오류 포착
- 코드 패턴 및 형식 표준화
- 원치 않는 어플리케이션 사용률 및 로직 에러로 인한 **Bug** 감소
- 보다 안전한 어플리케이션 만들기

서비스 및 릴리스 단계 별 해당하는 테스트

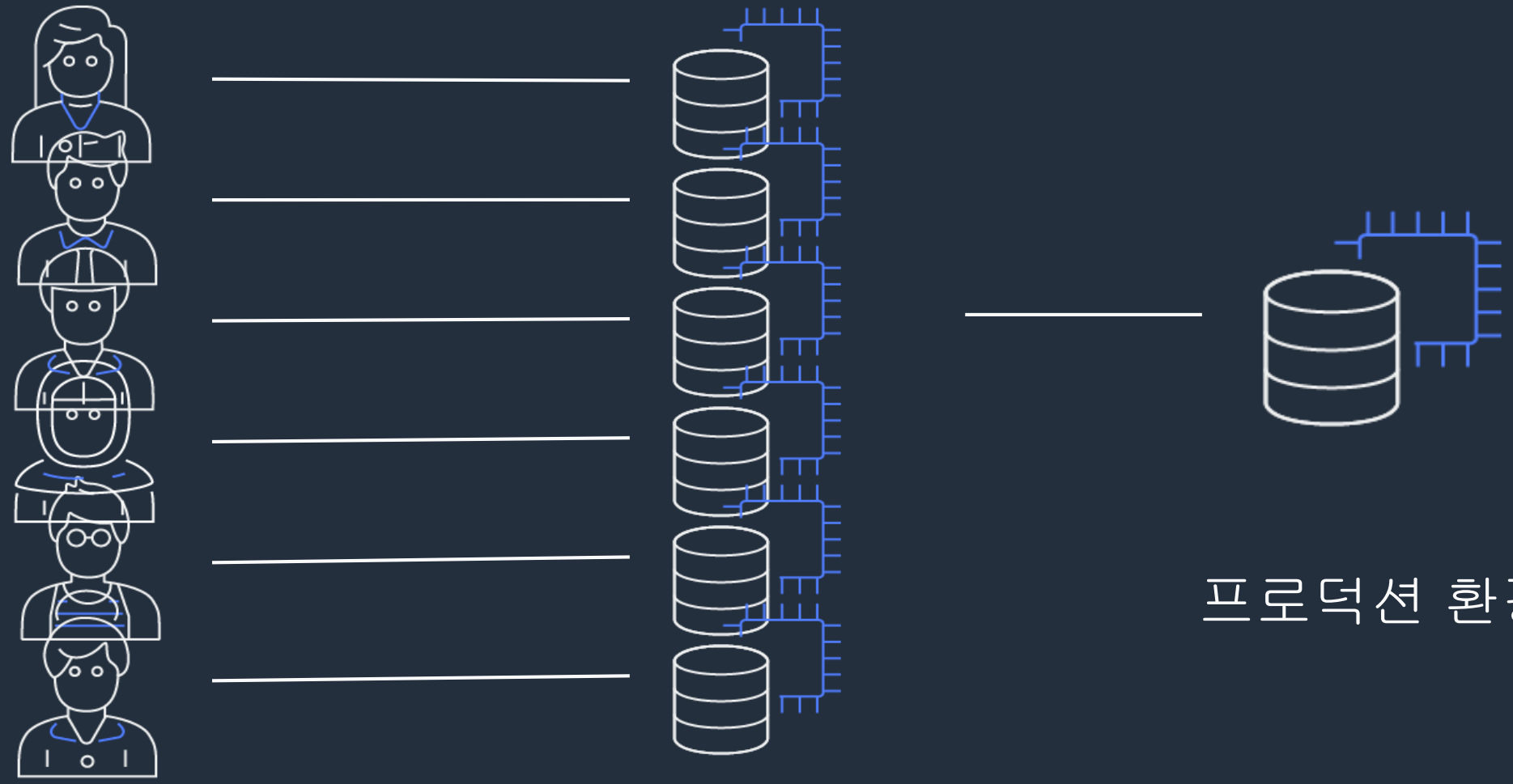


코드로써의 인프라 **(IaC)**

하나의 개발 환경은 확장되지 않음



개발자 당 하나의 개발 / 통합 환경

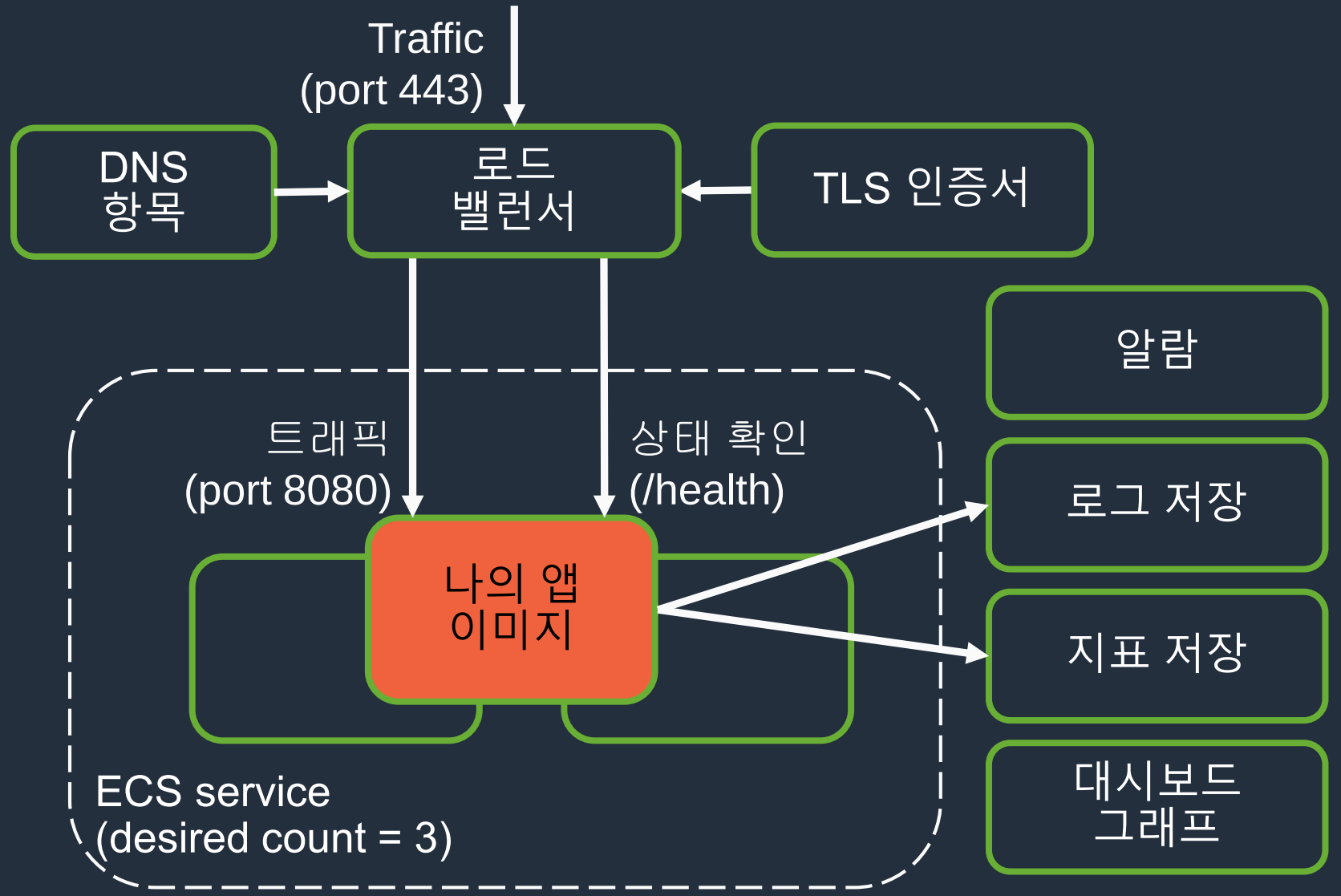


개발 팀

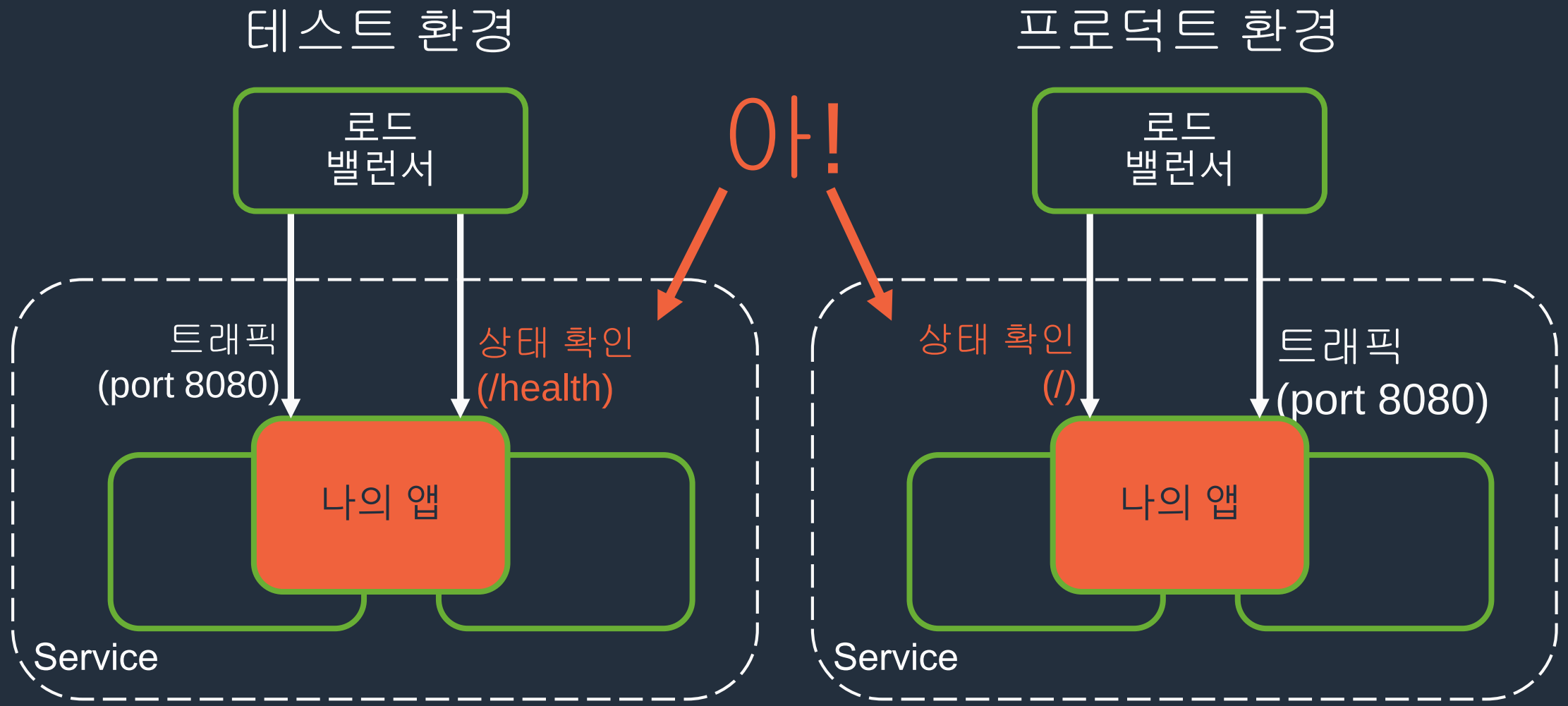
개발/통합 환경

프로덕션 환경

어플리케이션에 영향을 주는 변수



어플리케이션 릴리스 자동화의 한계



AWS CloudFormation

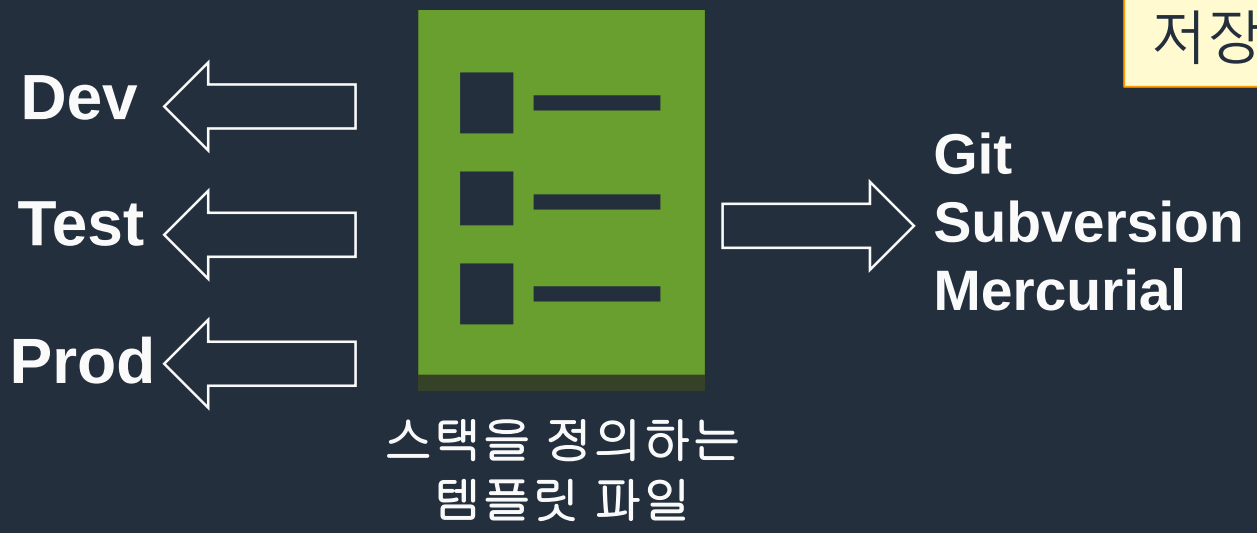
AWS CloudFormation

- 코드로서의 인프라 (IaC)
- 버전 관리와 통합
- JSON, YAML 형식
- 템플릿
- 스택
- 광범위한 AWS 리소스 지원
- AWS CloudFormation Designer – 비주얼 도구

AWS CloudFormation: 어플리케이션 스택 예시

템플릿을 사용하여 개발, 테스트 및 프로덕션과 같은 여러 환경을 구축

선택한 버전 관리 시스템을 사용하여 이 템플릿의 변경 사항을 저장하고 추적.



전체 애플리케이션은 AWS CloudFormation 템플릿 내에 나타낼 수 있음

템플릿 구조

```
{
  "Description" : "Create an EC2 instance.",
  "Resources" : {
    "Ec2Instance" : {
      "Type" : "AWS::EC2::Instance",
      "Properties" : {
        "KeyName" : "my-key-pair",
        "ImageId" : "ami-75g0061f",
        "InstanceType"   : "m1.medium"
      }
    }
  }
}
```

템플릿 구조

```
{
  "Description" : "Create an EC2 instance.”,
  "Parameters" : {
    "UserKeyName" : {
      "Description" : "The EC2 Key Pair to allow SSH access to the instance",
      "Type" : "String"
    }
  },
  "Resources" : {
    "Ec2Instance" : {
      "Type" : "AWS::EC2::Instance",
      "Properties" : {
        "KeyName" : { “Ref” : “UserKeyName”},
        "ImageId" : "ami-75g0061f”,
        “InstanceType” : “m1.medium”
      }
    }
  }
}
```

템플릿 구조

```
{
  "Description" : "Create an EC2 instance.",
  "Parameters" : {
    "UserKeyName" : {
       "Description" : "The EC2 Key Pair to allow SSH access to the instance",
       "Type" : "String"
    },
    "InstanceType" : {
      "Description" : "The EC2 Instance Type to launch.",
      "Type" : "String",
      "AllowedValues" : ["t1.micro", "m1.small", "m1.medium"]
    }
  },
  "Resources" : {
    "Ec2Instance" : {
       "Type" : "AWS::EC2::Instance",
       "Properties" : {
          "KeyName" : { "Ref" : "UserKeyName"},
          "ImageId" : "ami-75g0061f",
          "InstanceType" : { "Ref" : "InstanceType" }
       }
    }
  },
  "Outputs" : {
    "InstancePublicDnsName" : {
       "Description" : "The public DNS name of the newly created EC2 instance",
       "Value" : { "Fn::GetAtt" : [ "Ec2Instance", "PublicDnsName" ] }
    }
  }
}
```

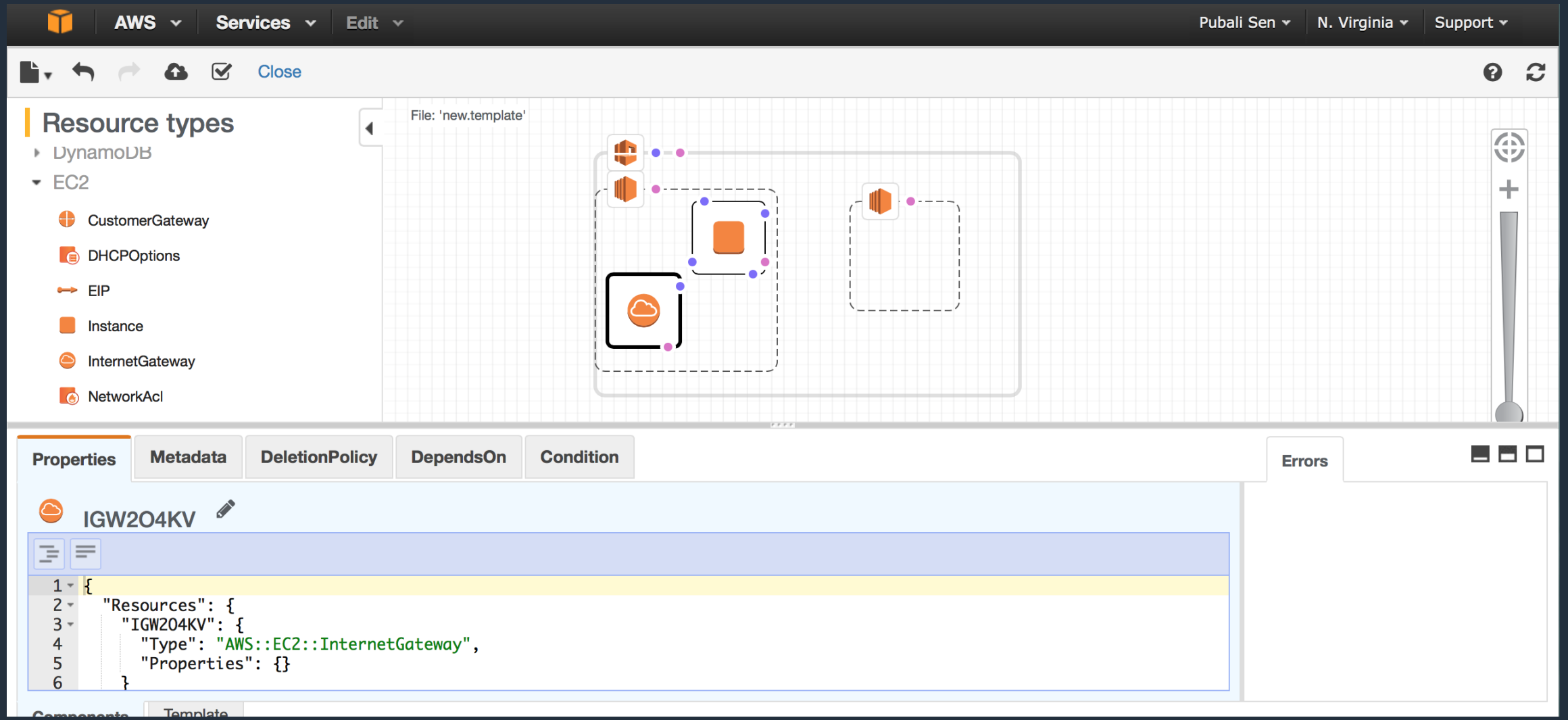
어플리케이션 배포 - User Data

```
"UserData": {
          "Fn::Base64": {
            "Fn::Join": [
              "",
              [
                "#!/bin/bash -ex\n",
                "yum -y install git-core\n",
                "yum -y install php-pear\n",
                "pear install Crypt_HMAC2-1.0.0\n",
                "pear install HTTP_Request-1.4.4\n",
                "pear install aws/sdk\n",
```

어플리케이션 배포 - cfn-init

```
"Ec2Instance": {
  "Metadata": {
    "AWS::CloudFormation::Init": {
      "config": {
        "sources" : {
          "/usr/local/bin/s3cmd" : "https://github.com/s3tools/s3cmd”
        },
        "packages": {
          "yum": { "git": [] }
        }
    }
  }
}
```

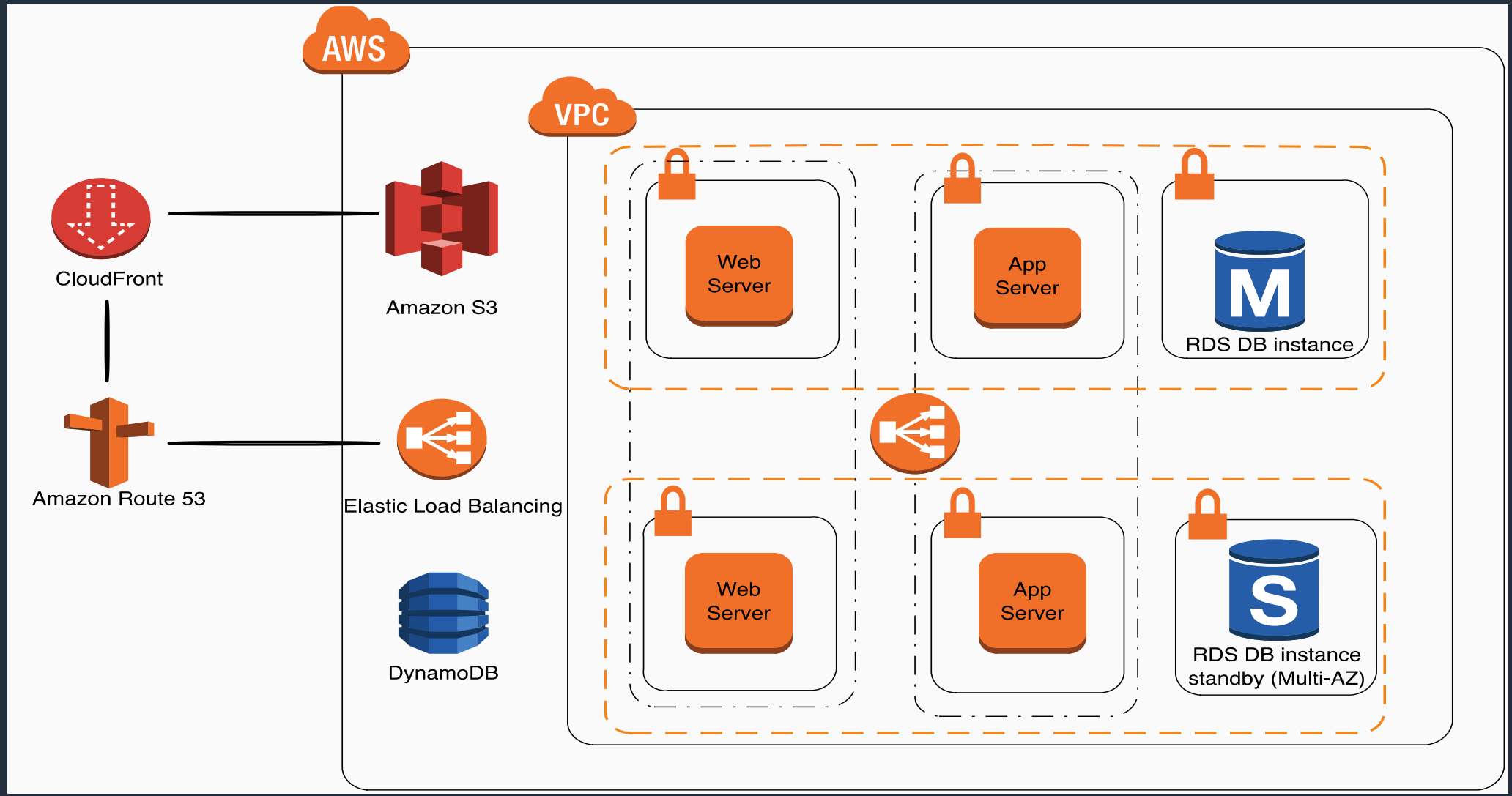
AWS CloudFormation Designer – 시각화 도구



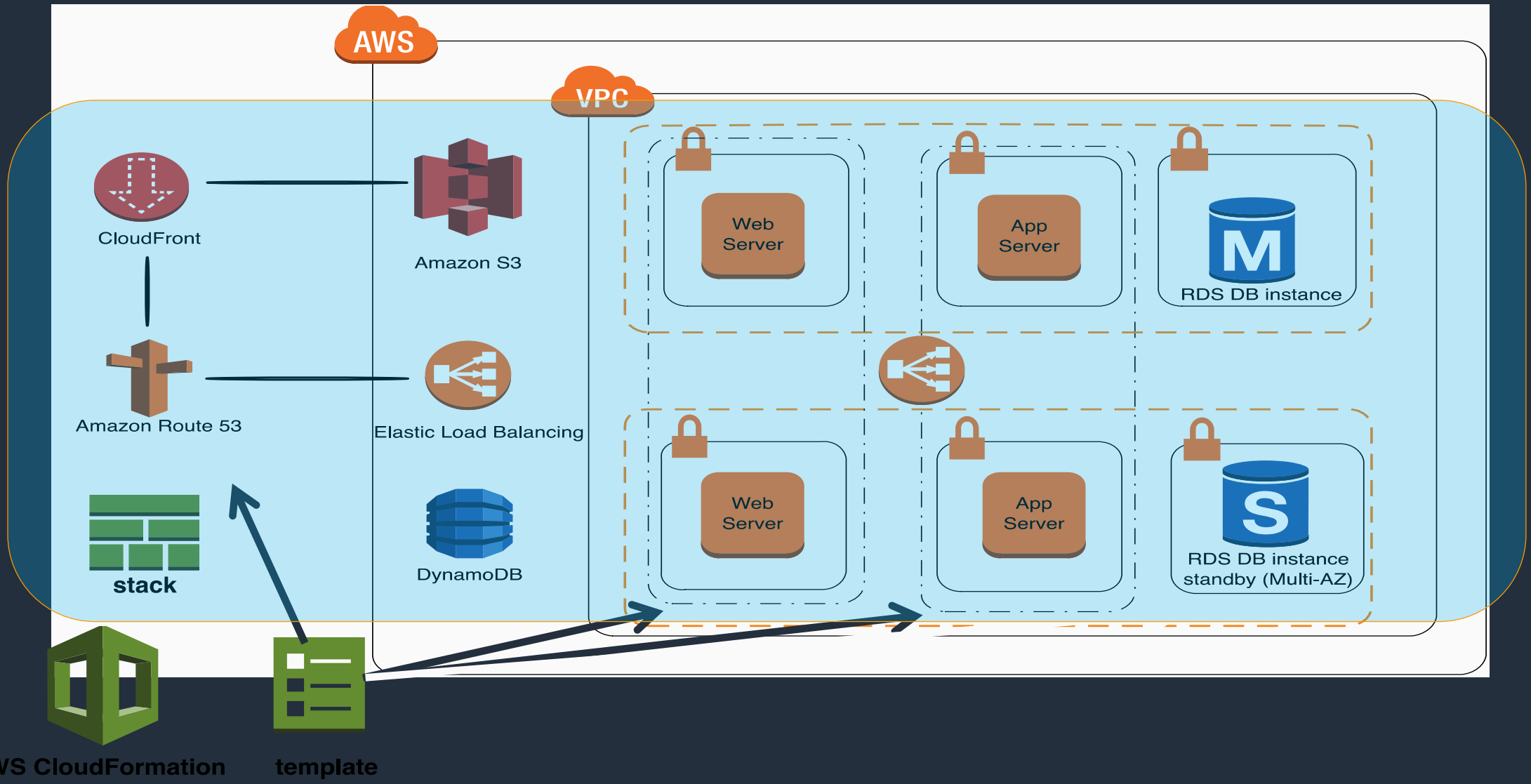
3rd Party 도구

- 기존 구성 관리 도구 (Configuration Management Tolls) 와 쉽게 통합
- User-Data 또는 cfn-init를 사용하여 쉽게 에이전트를 구성

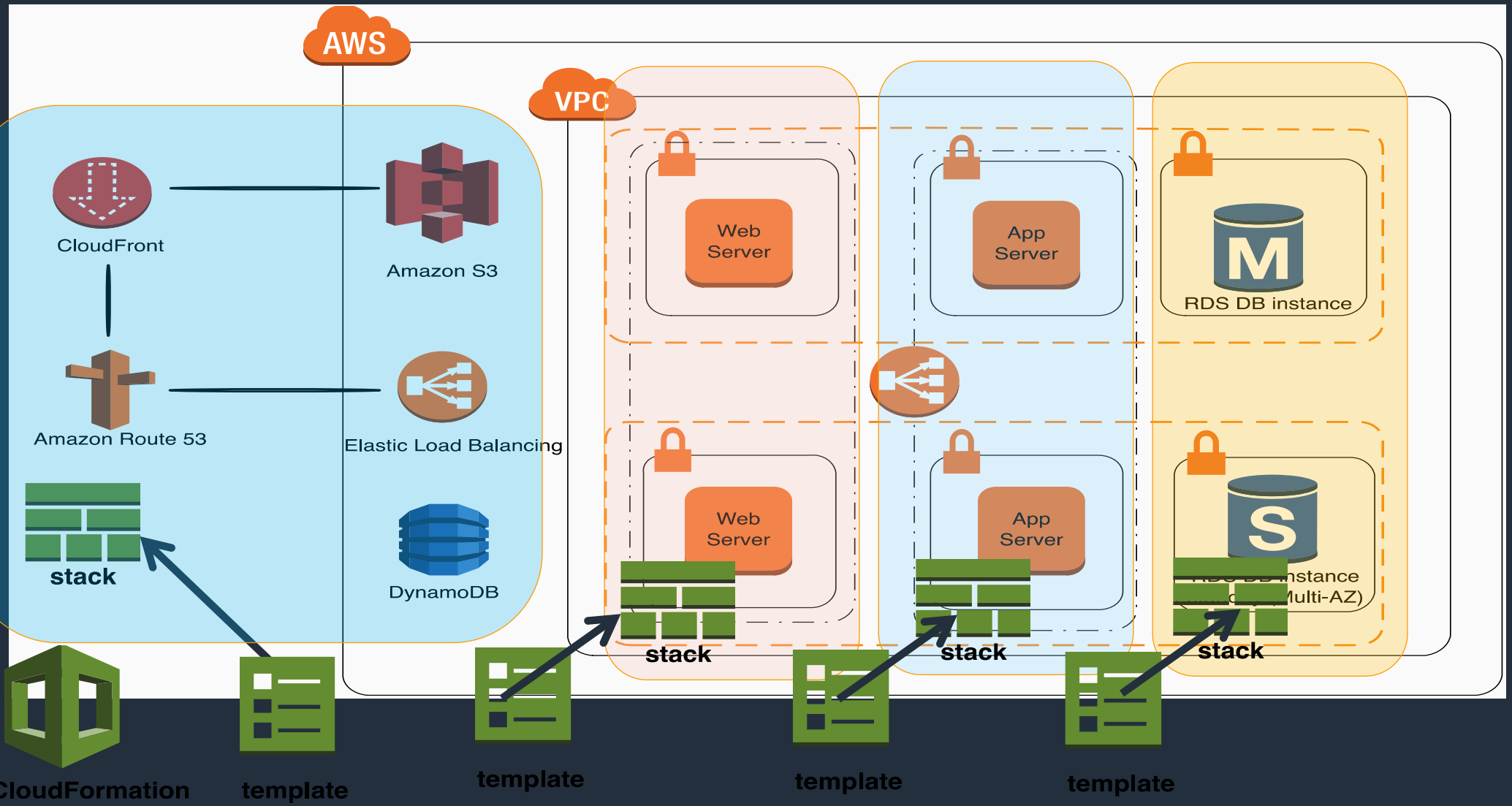
샘플 아키텍처



CloudFormation을 사용한 샘플 아키텍처



CloudFormation을 사용한 샘플 아키텍처

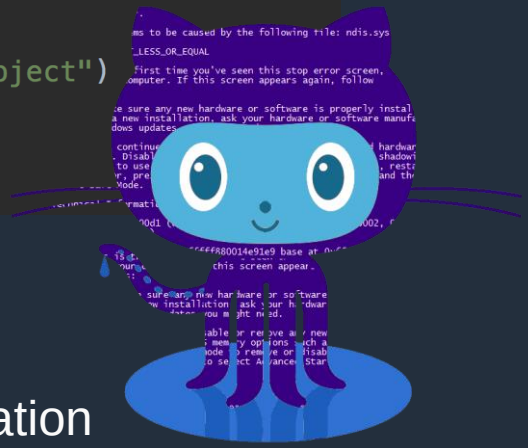


AWS Cloud Development Kit (AWS CDK)

재사용 가능한 구성 요소로 클라우드 인프라를 모델링하기위한 multi-language 소프트웨어 개발 프레임 워크


```
#!/usr/bin/env node
import ...

export class MyStack extends cdk.Stack {
  constructor(scope: cdk.App, id: string, props?: cdk.StackProps) {
    super(scope, id, props);

    let myVpc = new ec2.Vpc(this, id: "Vpc", props: {
      maxAZs: 2,
    });

    let myCluster = new ecs.Cluster(this, id: 'Cluster', props: {
      vpc: myVpc,
    })

    // Define an ECS Fargate Service that runs two instances
    // of a Docker image created from a local Dockerfile
    new ecs_patterns.LoadBalancedFargateService(this, id: 'Service', props: {
      cluster: myCluster,
      desiredCount: 2,
      image: ecs.AssetImage.fromAsset( directory: "my-docker-project")
    });
  }
}
```

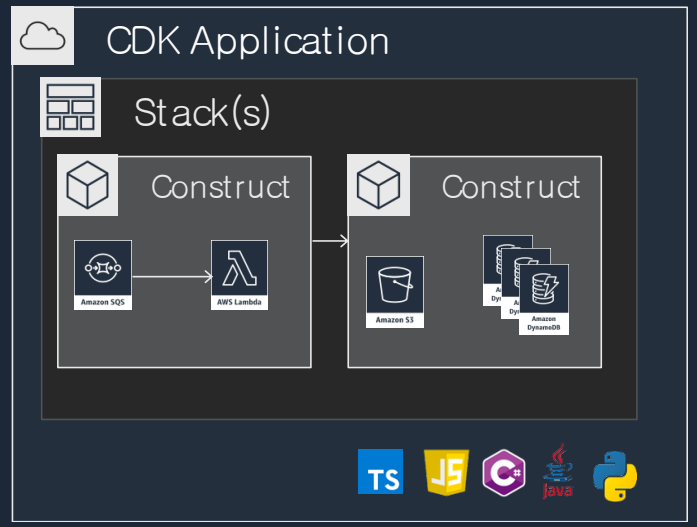
AWS CloudFormation



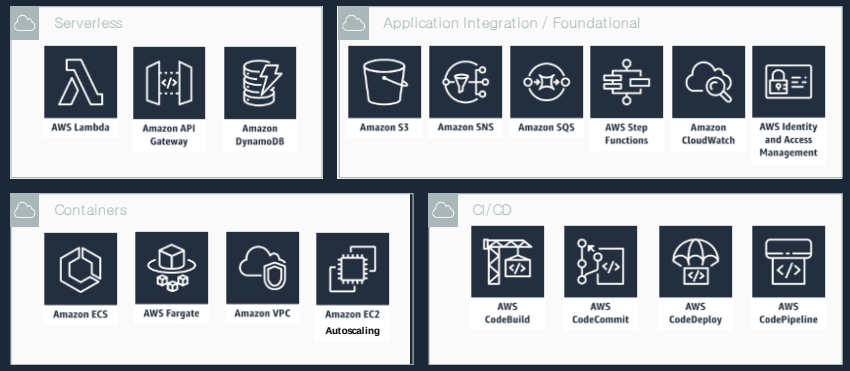
AWS Cloud Development Kit (AWS CDK)

주요 구성 요소

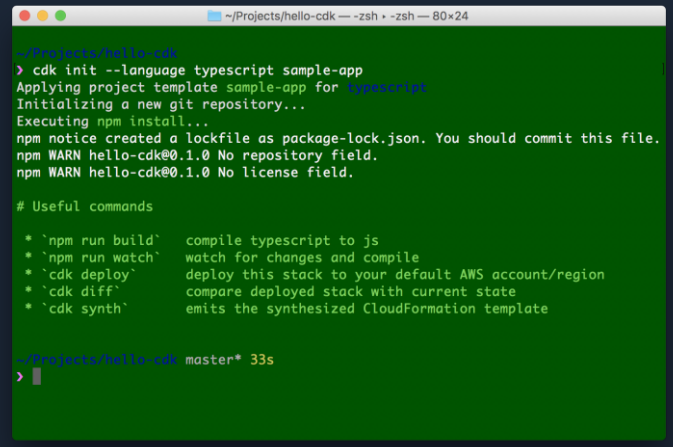
Core Framework



AWS Construct Library



AWS CDK CLI



CDK: VPC 생성

```
//
// create VPC w/ public and private subnets in 2 AZ
// this also creates a NAT Gateway
//
const vpc = new ec2.Vpc(this, 'NewsBlogVPC', {
  maxAzs : 2
});
```

CDK: Application 패키징

```
    //
    // create static web site as S3 assets
    //
    var path = require('path');
    const asset = new assets.Asset(this, 'YourSampleApp', {
      path: path.join(__dirname, '../html')
    });
```

CDK: 서버 Bootstrapping

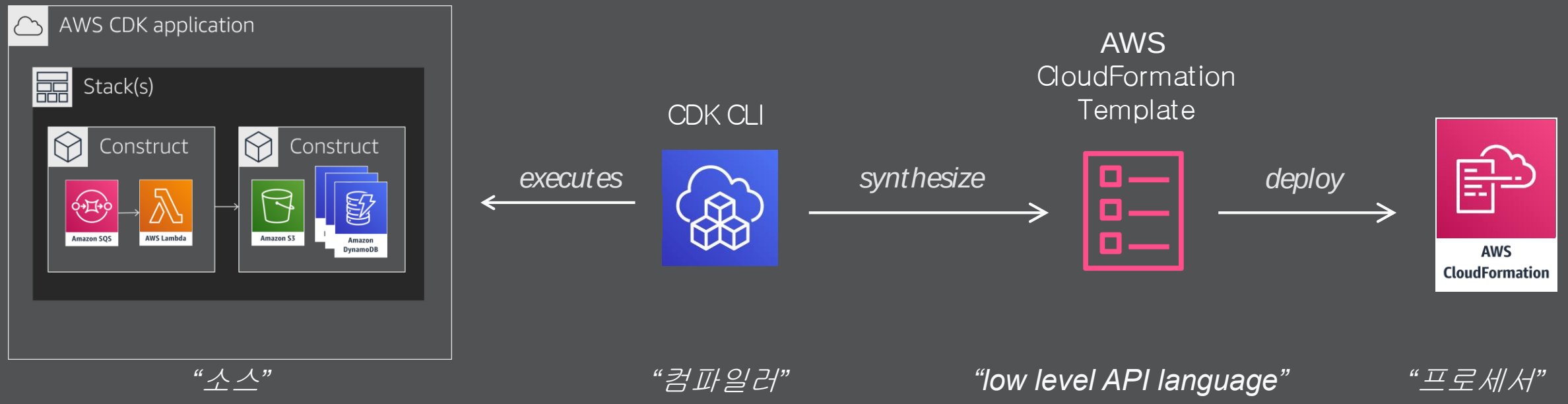
```
// define a user data script to install & launch our app
const userData = UserData.forLinux();

userData.addCommands('yum install -y nginx',
                     'chkconfig nginx on', 'service nginx start');

userData.addCommands(`aws s3 cp s3://${asset.s3BucketName}/${asset.s3ObjectKey} .`,
                     `unzip *.zip`,
                     `/bin/cp -r -n ${env}/* /usr/share/nginx/html/`);
```

CDK: Auto Scaling Group 생성

```
 // create an auto scaling group for each environment
   const asg = new autoscaling.AutoScalingGroup(this, 'YourAppgAutoScalingG
roup' , {


     vpc,
     instanceType: ec2.instanceType.of(ec2.instanceClass.BURSTABLE3,
                                       ec2.instanceSize.MiCRO),
     machineimage: new ec2.AmazonLinuximage(),
     desiredCapacity: 2,
     role: role,
     userData: userData
  });
```

CDK: 개발(dev) 환경 배포

```
$ cdk deploy
```



Deployment

AWS CodeDeploy

자동화 된 배포 조정

Application revisions

v1, v2, v3

배포 그룹

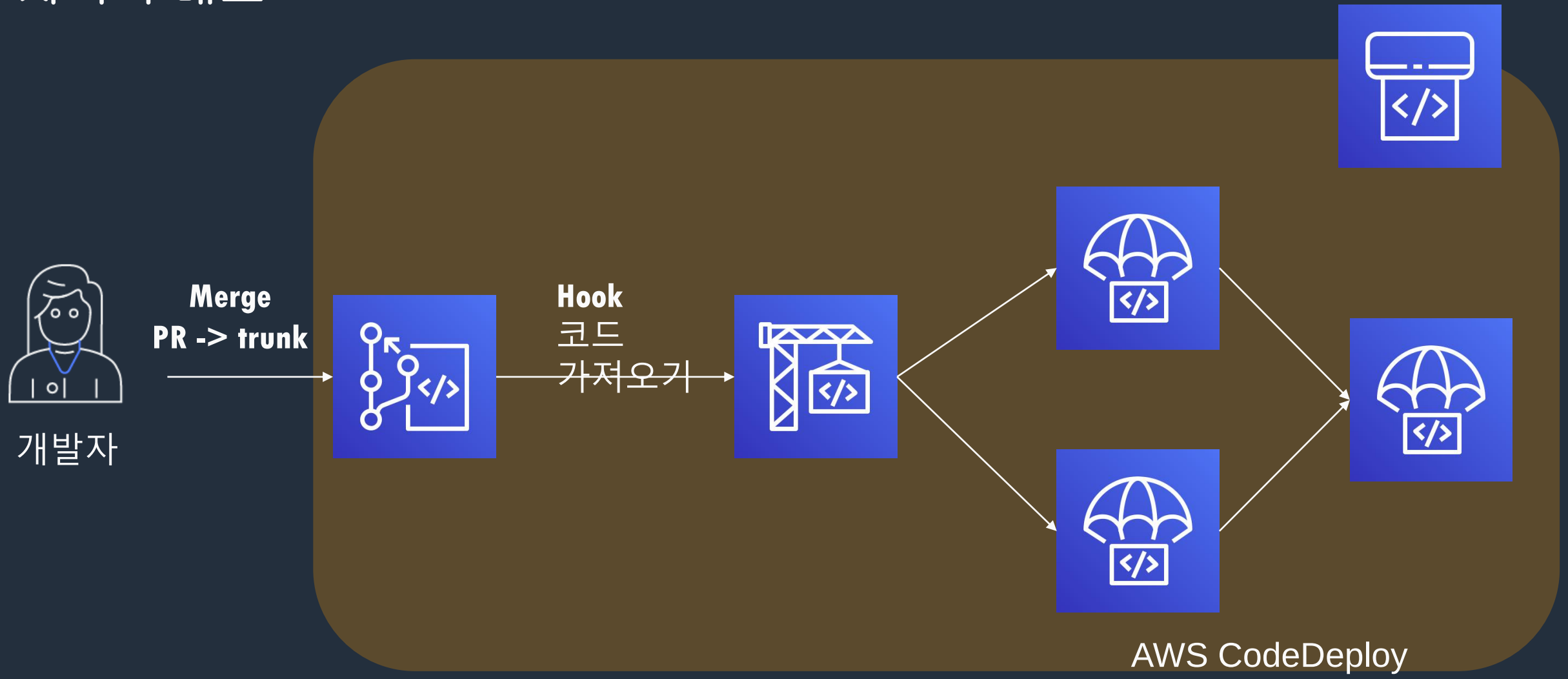
- 단일 인스턴스에서 수천으로 확장
- **Zero-downtime** 배포를 위한 다중 구성
- 배포 제어 및 모니터링 중앙 집중화
- 구성 가능한 배치 후크 **(hooks)**
- **EC2,** 온 프레미스 및 **Lambda**에 배포

appspec.yml 예시

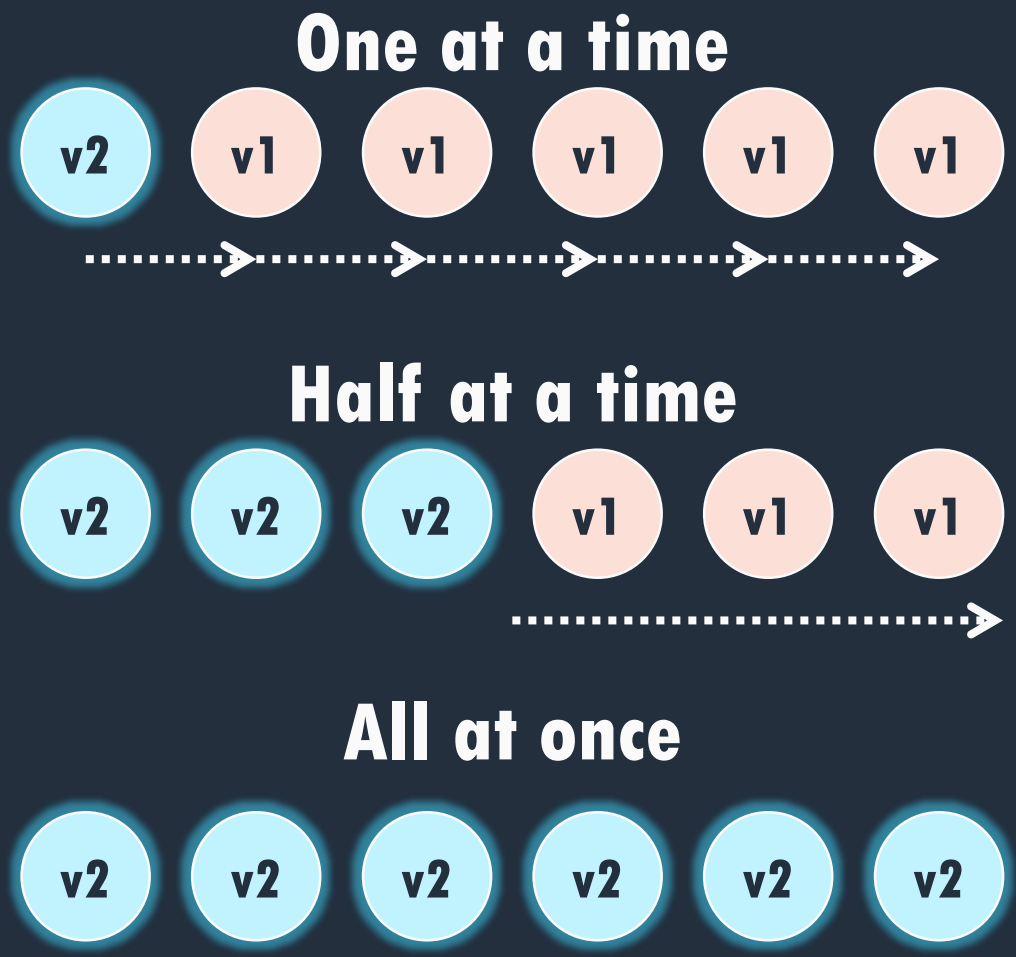
```
version: 0.0
os: linux
files:
  - source: /
    destination: /var/www/html
permissions:
  - object: /var/www/html
    pattern: “*.html”
    owner: root
    group: root
    mode: 755
hooks:
  ApplicationStop:
    - location: scripts/deregister_from_elb.sh
  BeforeInstall:
    - location: scripts/install_dependencies.sh
  ApplicationStart:
    - location: scripts/start_httpd.sh
  ValidateService:
    - location: scripts/test_site.sh
    - location: scripts/register_with_elb.sh
```

- 한 디렉토리로 응용 프로그램 파일을 보내고, 다른 디렉토리로 구성 파일을 보냄
- 특정 디렉토리 및 파일에 대한 특정 권한 설정
- **ELB**에 인스턴스 제거 / 추가
- **Dependancy** 패키지 설치
- **Apache** 시작
- 성공적인 배포를 확인
- 추가 작업!

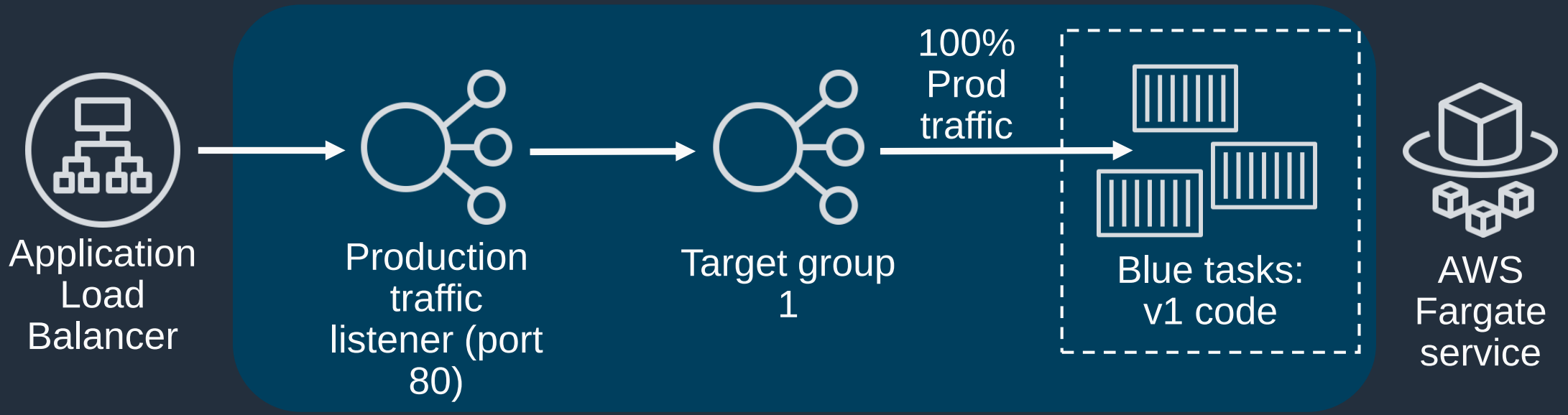
지속적 배포



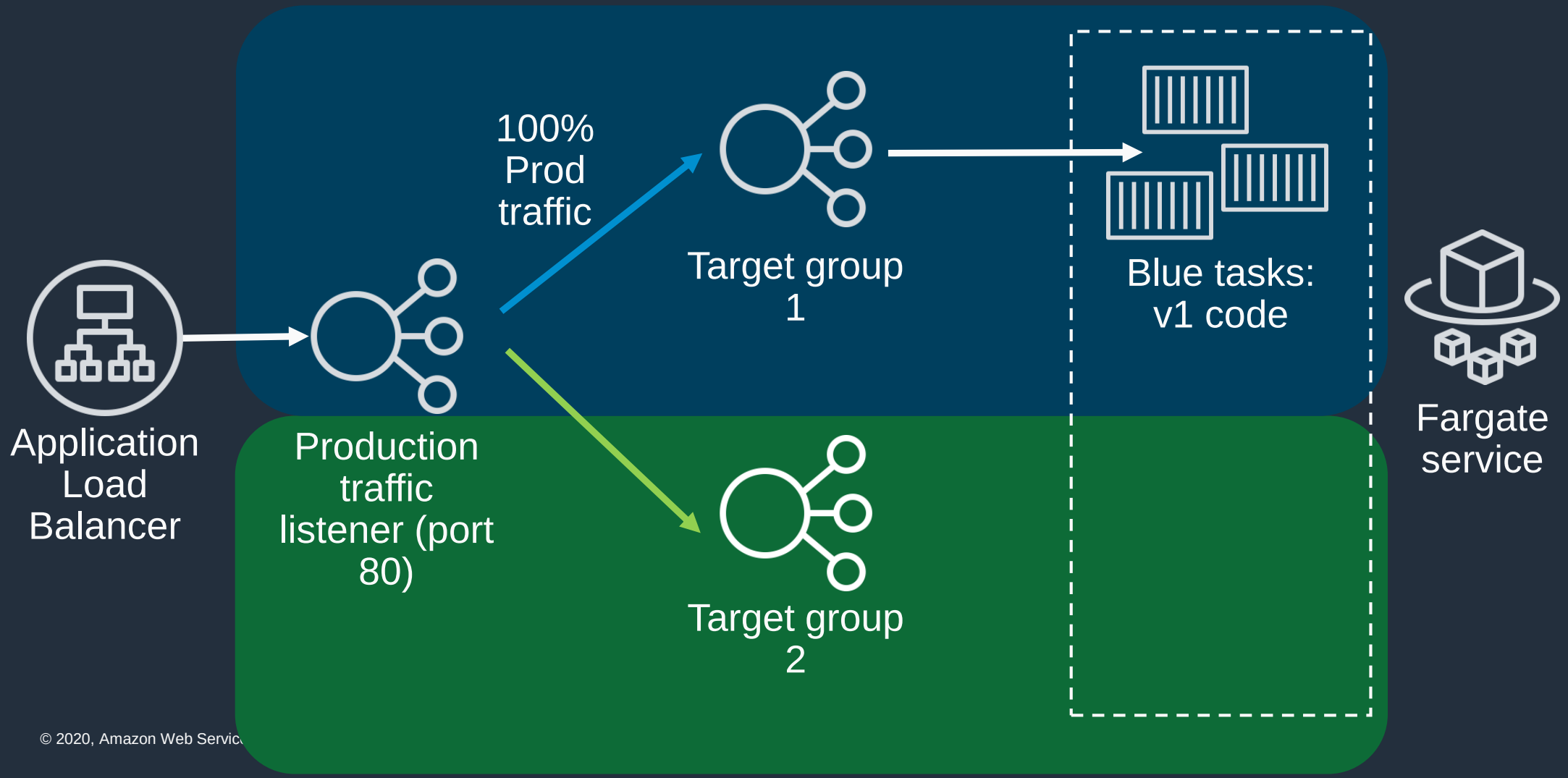
배포 속도와 그룹 선택



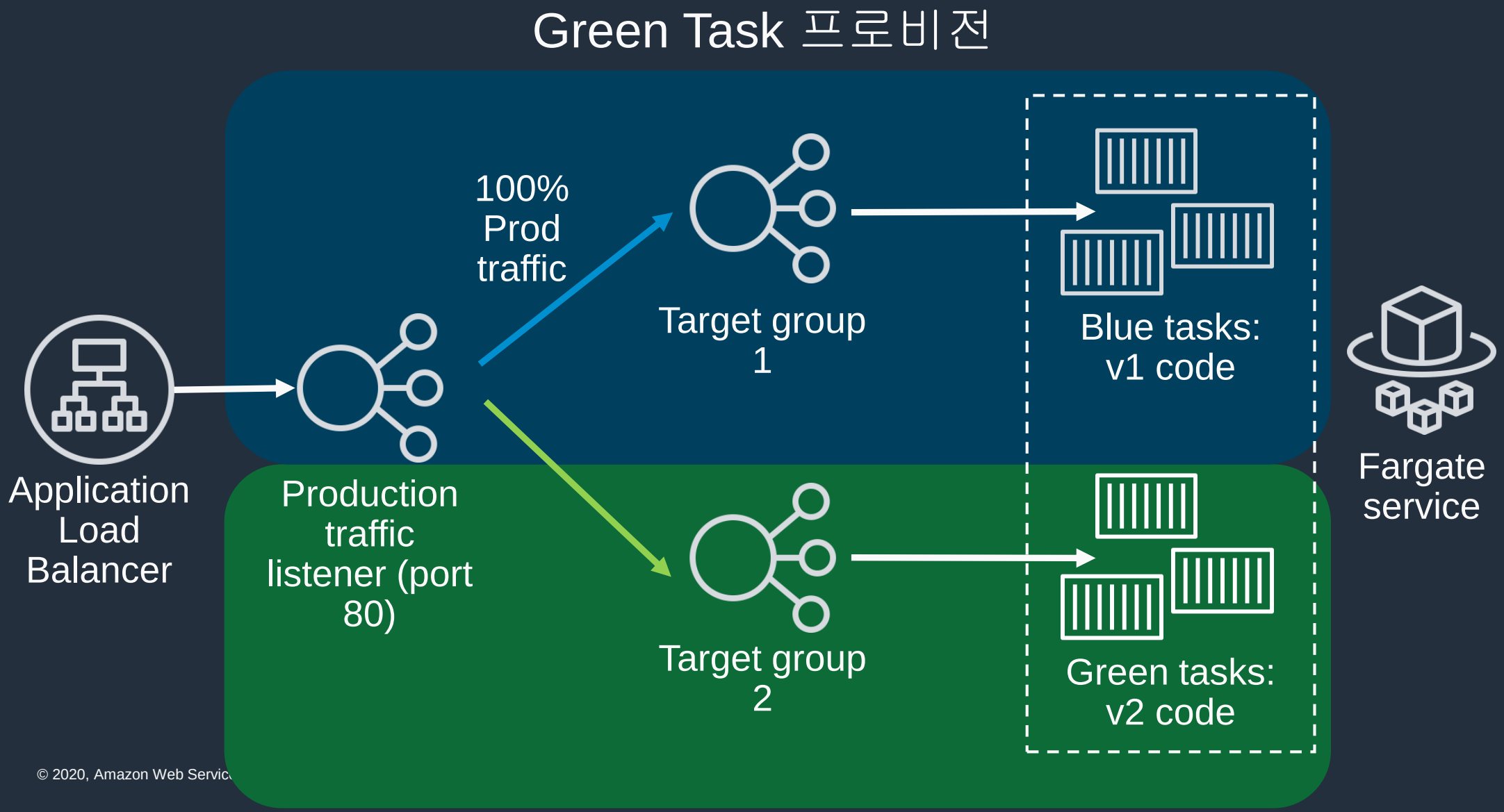
블루-그린 배포



블루-그린 배포

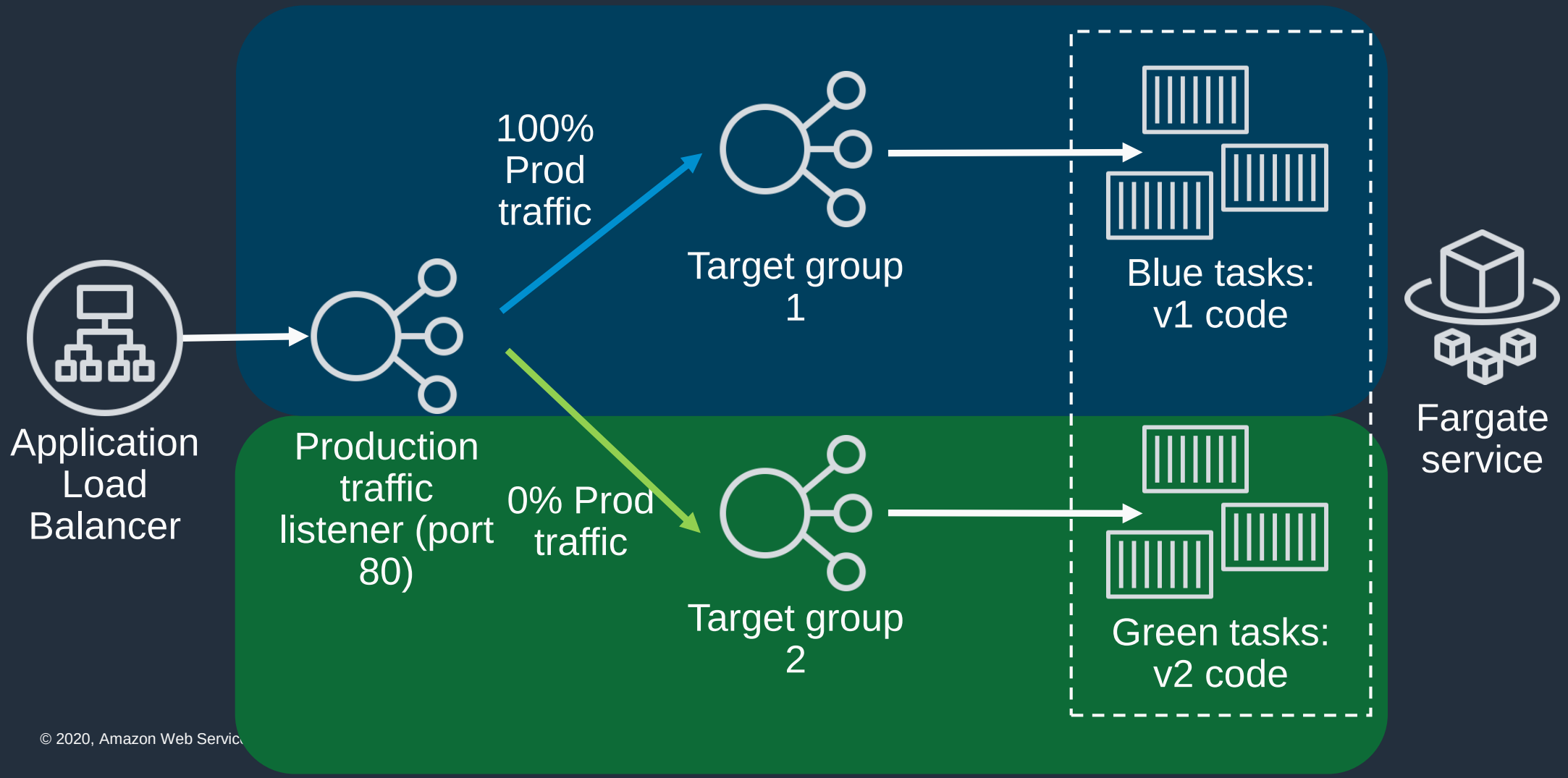


블루-그린 배포



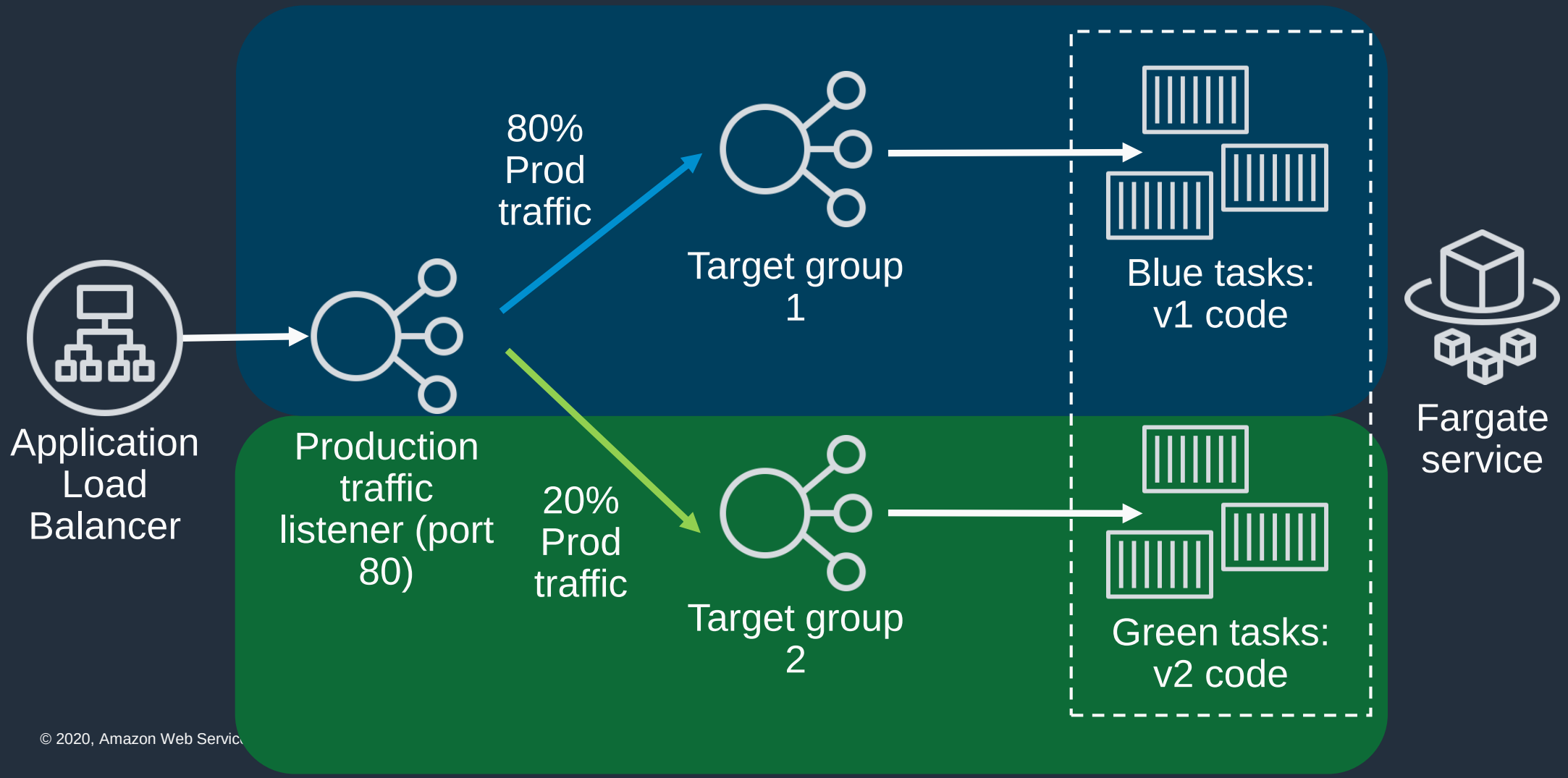
블루-그린 배포

Green Task가 프로덕션 트래픽을 받기 전에 테스트 엔드 포인트에 Hook 실행

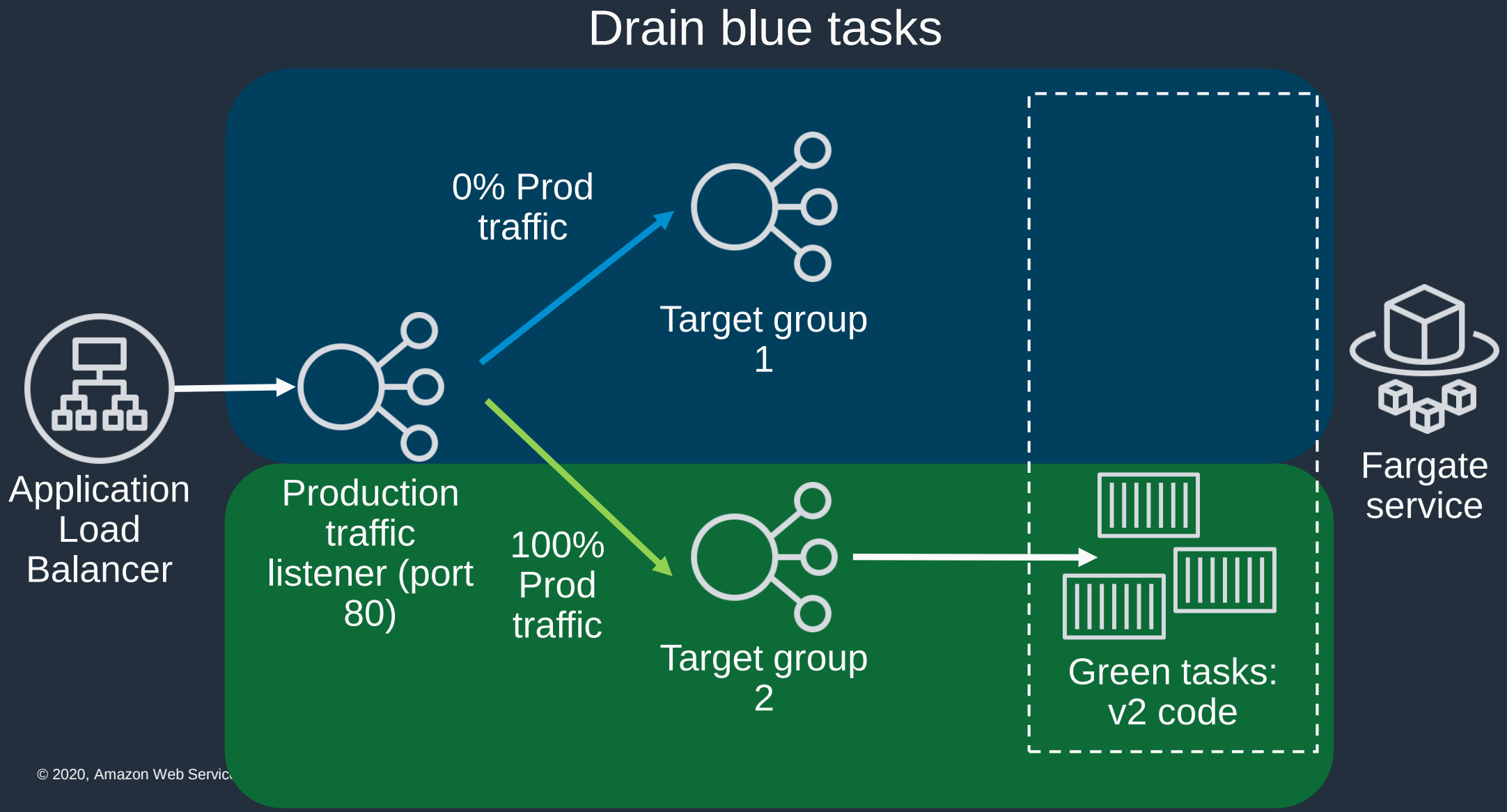


블루-그린 배포

경보(alarm)가 발생하면, Green Task 의 트래픽을 전환하고 롤백

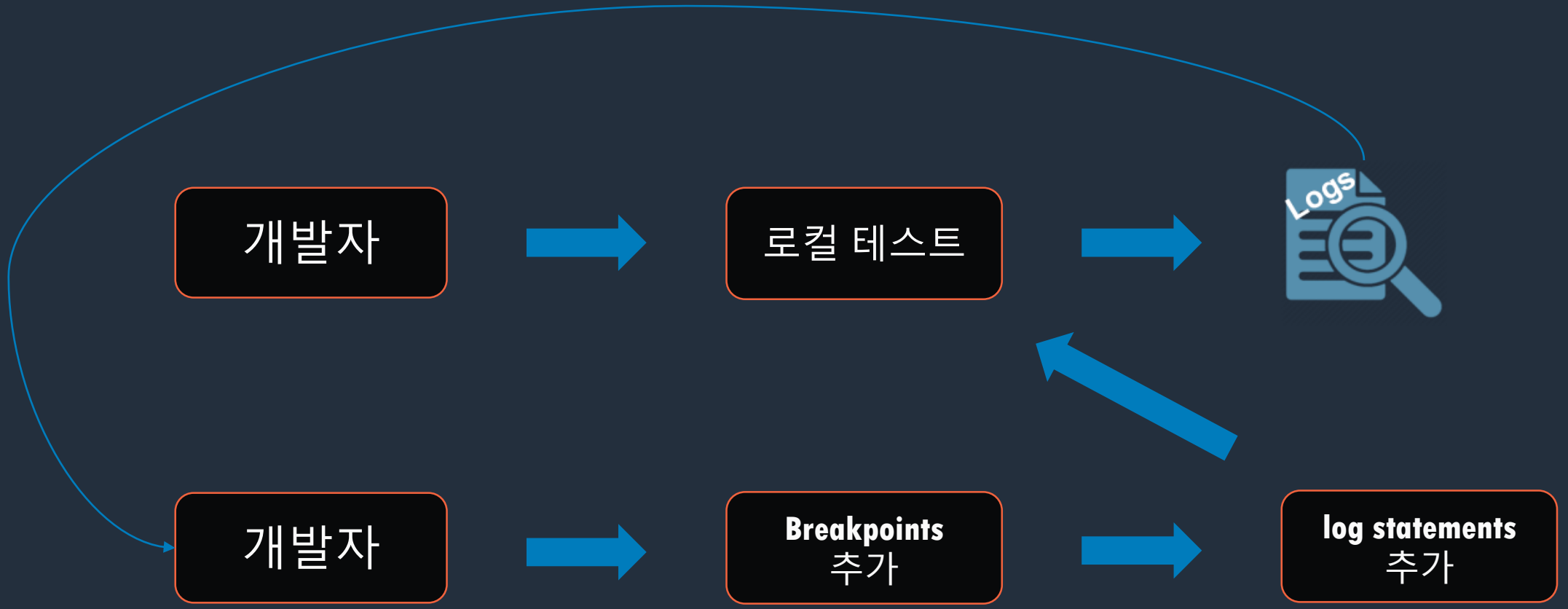


블루-그린 배포



Debugging

전통적인 디버깅



현대적(Modern) 애플리케이션은 로컬에서 디버깅하기 어려움

어플리케이션이

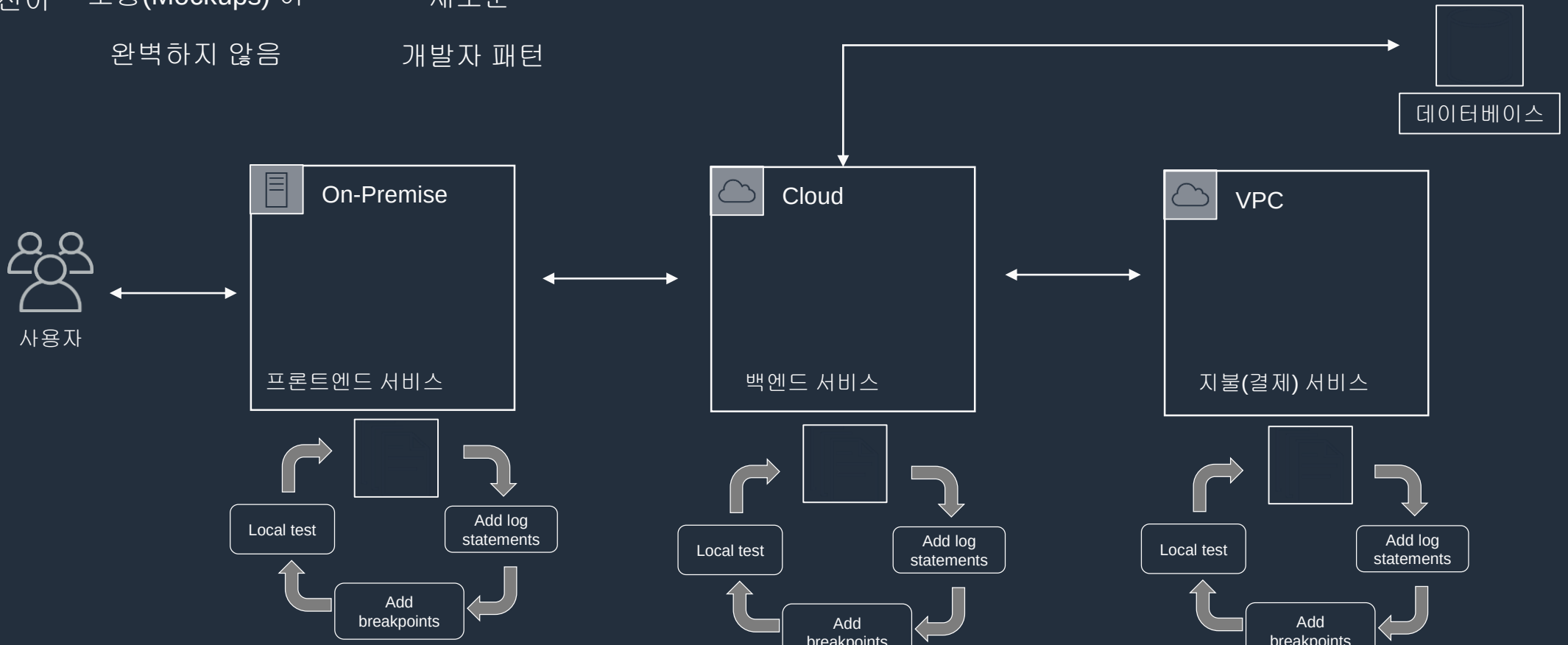
크다

모형(Mockups) 이

완벽하지 않음

새로운

개발자 패턴



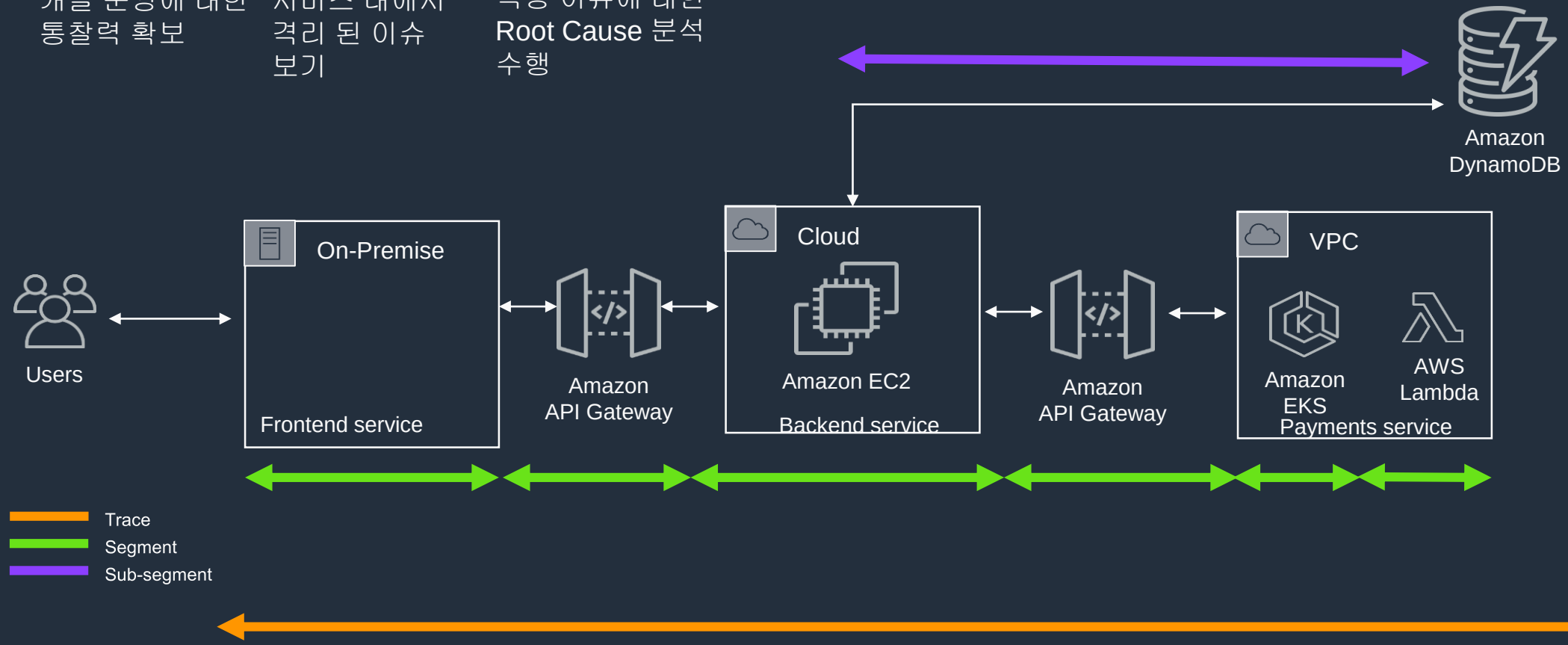
현대적(Modern) 애플리케이션을 위한 추적

여러 서비스를 발견

개별 운영에 대한 통찰력 확보

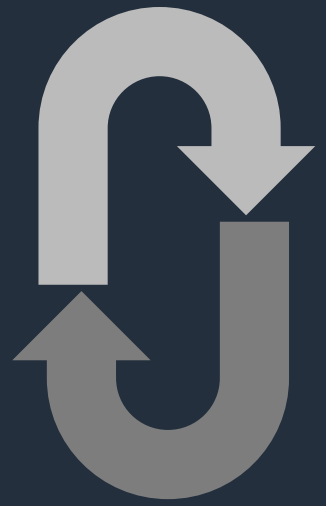
서비스 내에서 격리 된 이슈 보기

특정 이슈에 대한 Root Cause 분석 수행



현대적(Modern) 애플리케이션을 위해 제작된 AWS X-Ray

문제를 신속하게 분석 및 디버깅



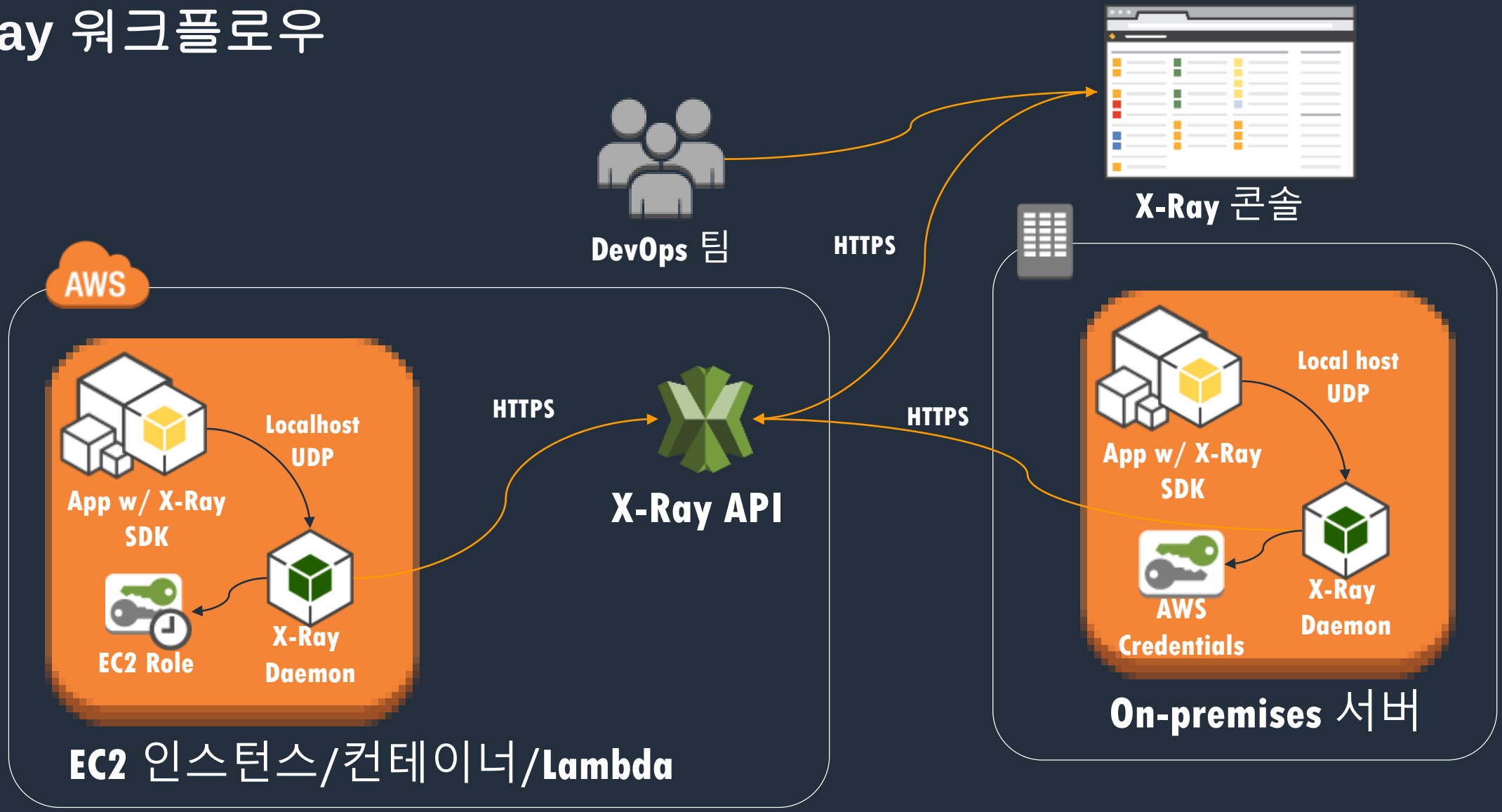
개별 서비스들의 End-to-End 보기

고객 영향도 파악

Cloud agnostic (Serverless 지원)

X-Ray 서비스 동작 방법

X-Ray 워크플로우



X-Ray SDK

Java, .NET, .NET Core, Python, Ruby, Go 및 **Node.js**에 사용 가능

Metadata 를 자동으로 캡처하는 필터를 추가:

- **AWS SDK**를 사용하는 **AWS** 서비스
- **HTTP** 및 **HTTPS**를 통한 **Non-AWS** 서비스
- 데이터베이스 (**MySQL, PostgreSQL** 및 **Amazon DynamoDB**)
- **Queues** (**Amazon SQS**)

Request에 대한 메타 데이터를 **Logging** 하기 위해 애플리케이션 코드를 수동으로 변조하지 않고도 빠르게 시작할 수 있음

Source on GitHub under https://github.com/aws?q=xray-sdk

X-Ray Daemon

UDP를 통해 SDK에서 데이터를 수신하고 로컬 버퍼 역할을 수행. 매 초마다 또는 로컬 버퍼가 채워질 때 데이터가 백엔드로 전송 및 비워짐

Amazon Linux AMI, RHEL, Ubuntu, macOS 및 Windows에서 사용 가능

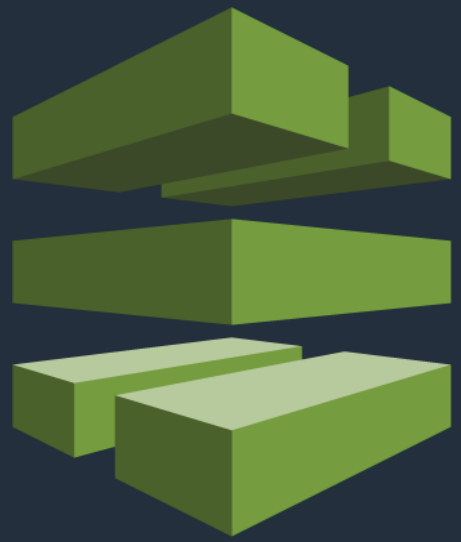
AWS Lambda에 사전 설치되어 있음

AWS 자격 증명이 제공되는 한 어디에서나 실행 가능 (예 : EC2, Amazon ECS, 온프레미스, 개발자 컴퓨터 등)

Source on GitHub under https://github.com/aws/aws-xray-daemon

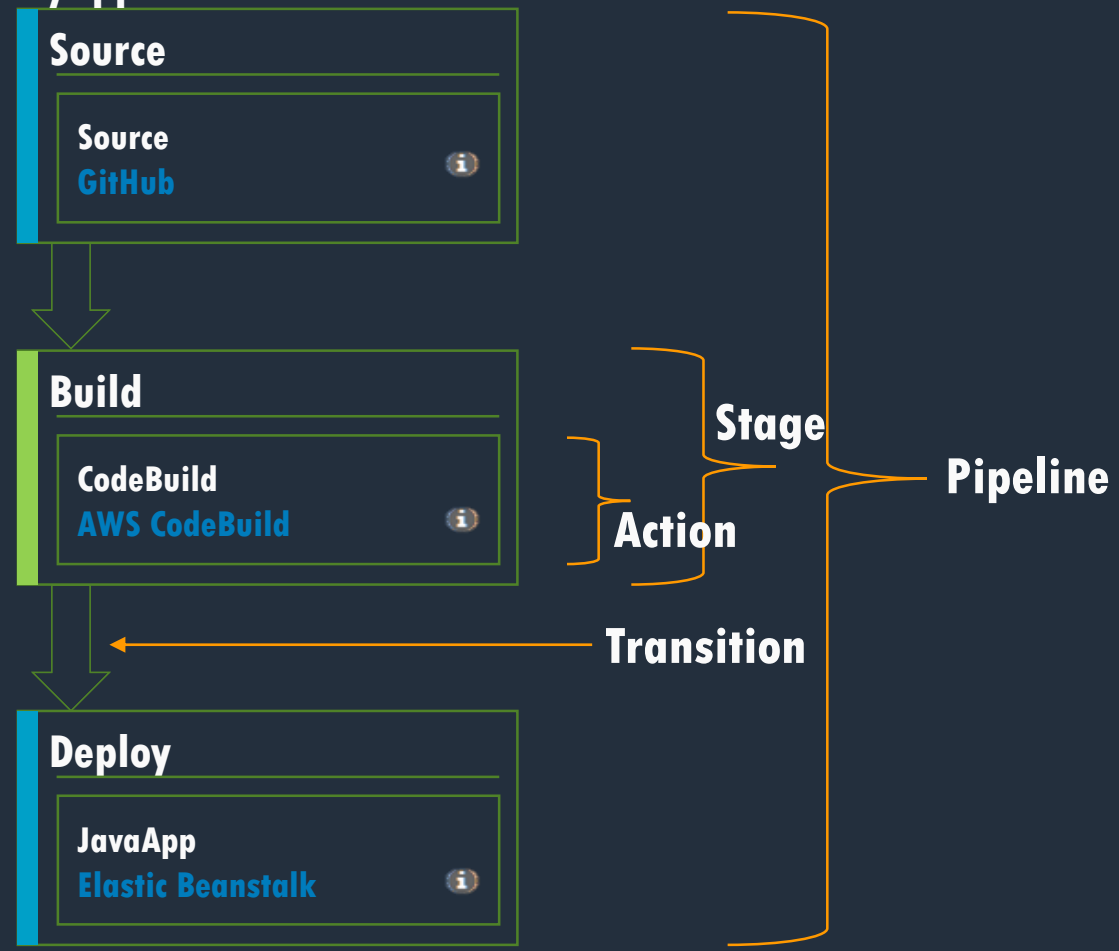
파이프라인 관리

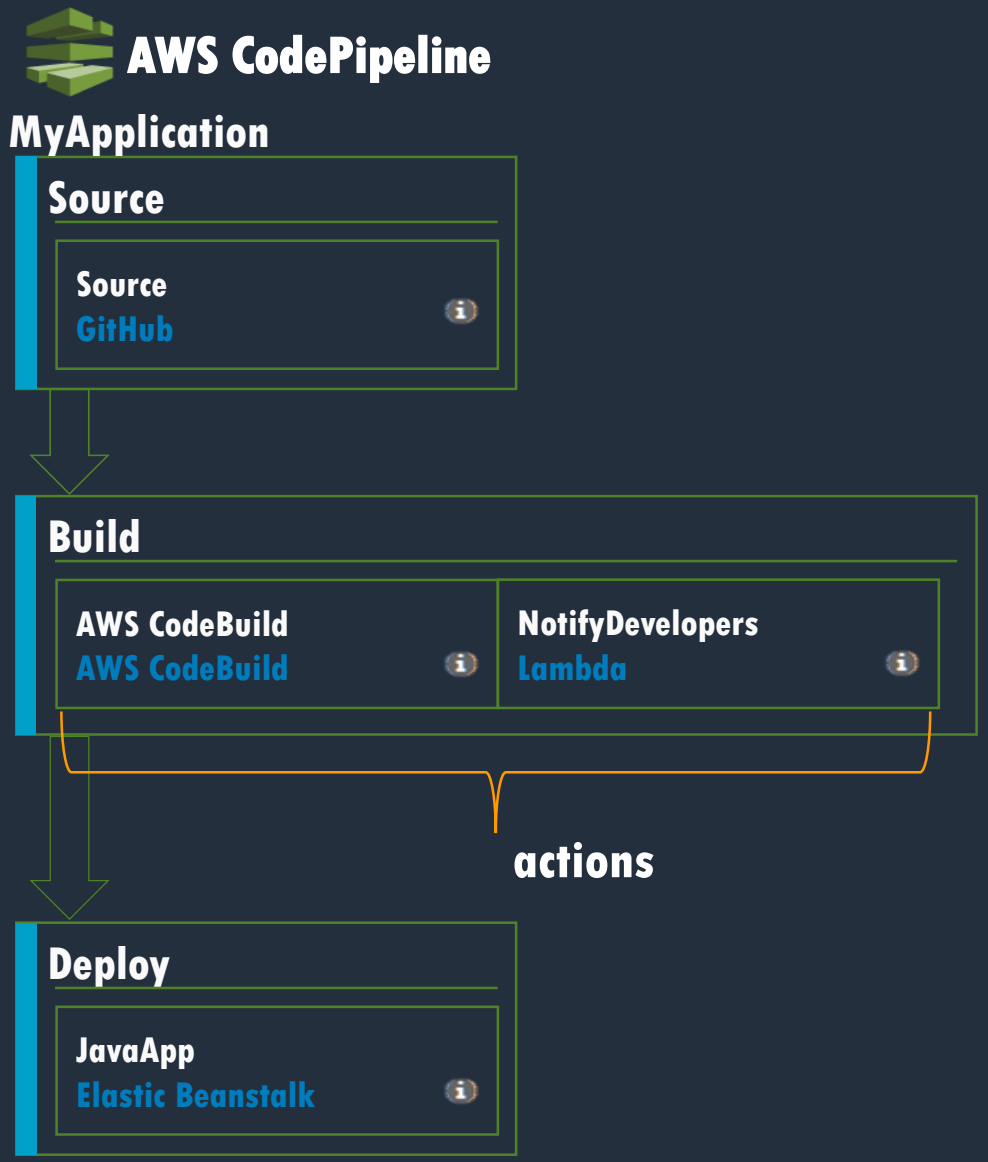
AWS CodePipeline



- 빠르고 신뢰성 있는 어플리케이션 업데이트를위한 지속적인 전달 (**Continuous Delivery**) 서비스
- 소프트웨어 릴리스 프로세스 모델링 및 시각화
- 코드가 변경 될 때마다 코드를 빌드, 테스트 및 배포
- 타사 도구 (**3rd-party**) 및 **AWS** 와 통합

AWS CodePipeline





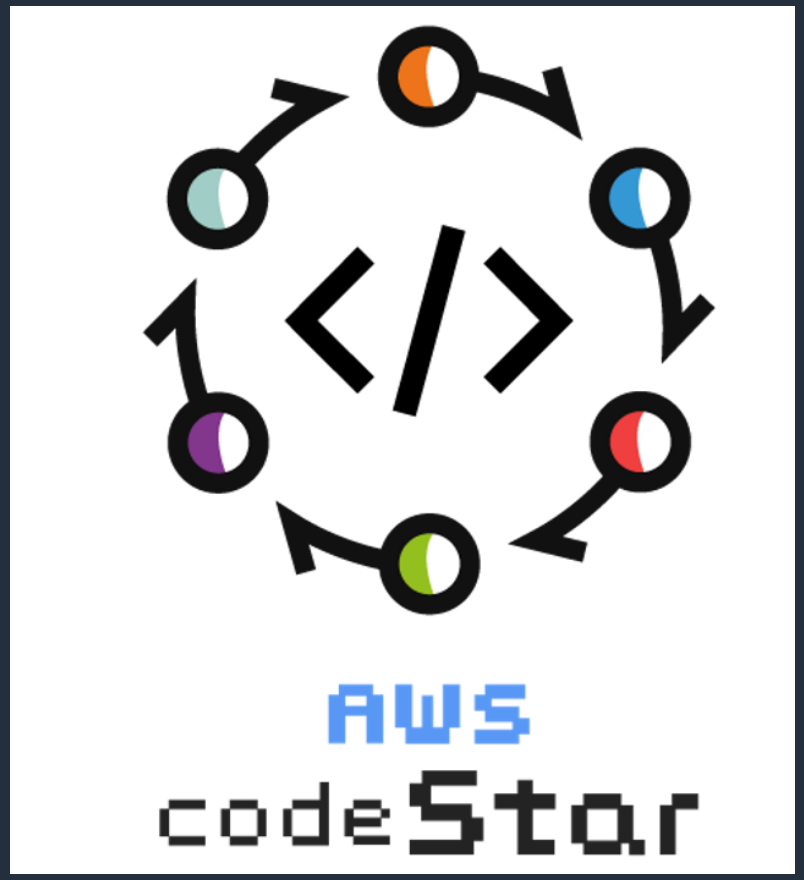
AWS CodePipeline

데모(실습)

https://bit.ly/38HfL80

Appendix.
- 프로젝트 관리

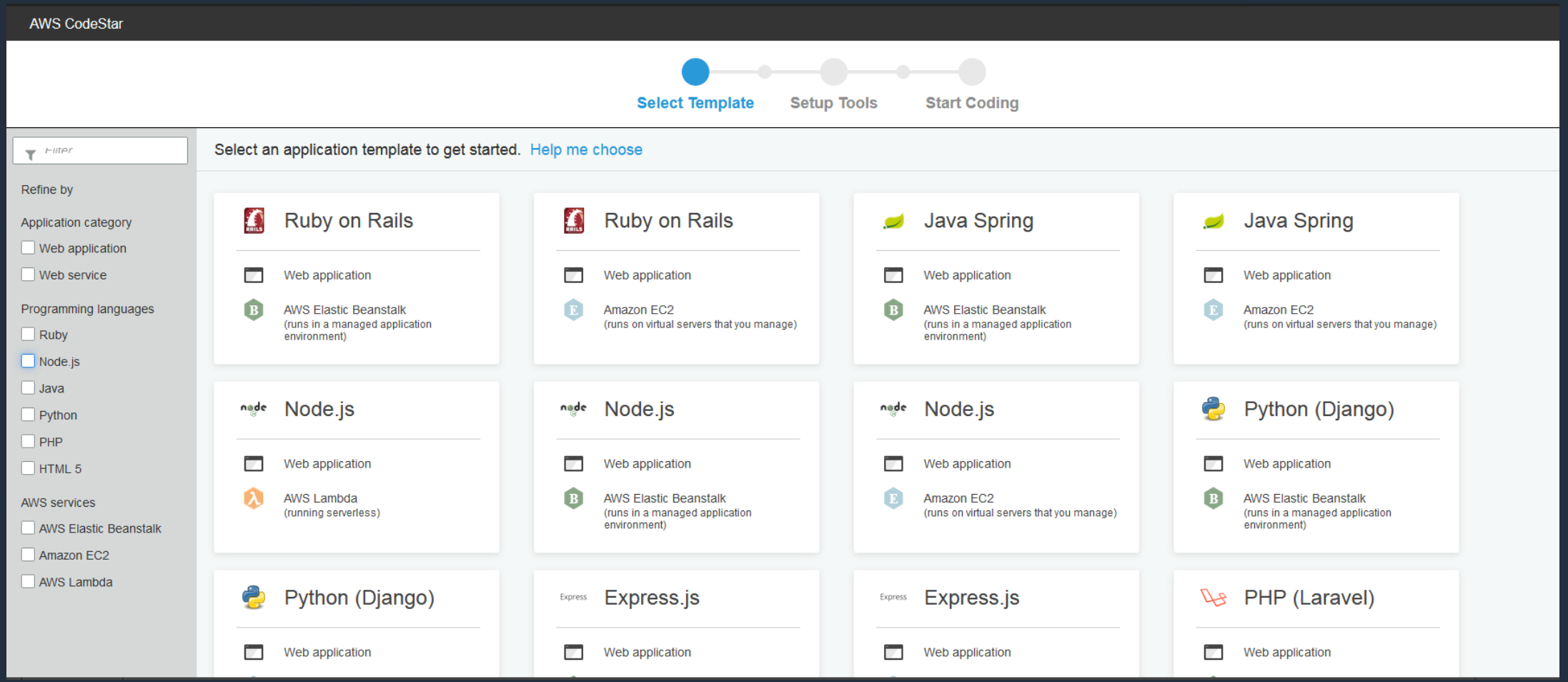
AWS CodeStar



- AWS에서 애플리케이션을 신속하게 개발, 빌드 및 배포
- 몇 분 안에 AWS에서 개발 시작
- 팀 전체에서 안전하게 작업
- 소프트웨어 제공을 쉽게 관리
- 다양한 프로젝트 템플릿 중에서 선택

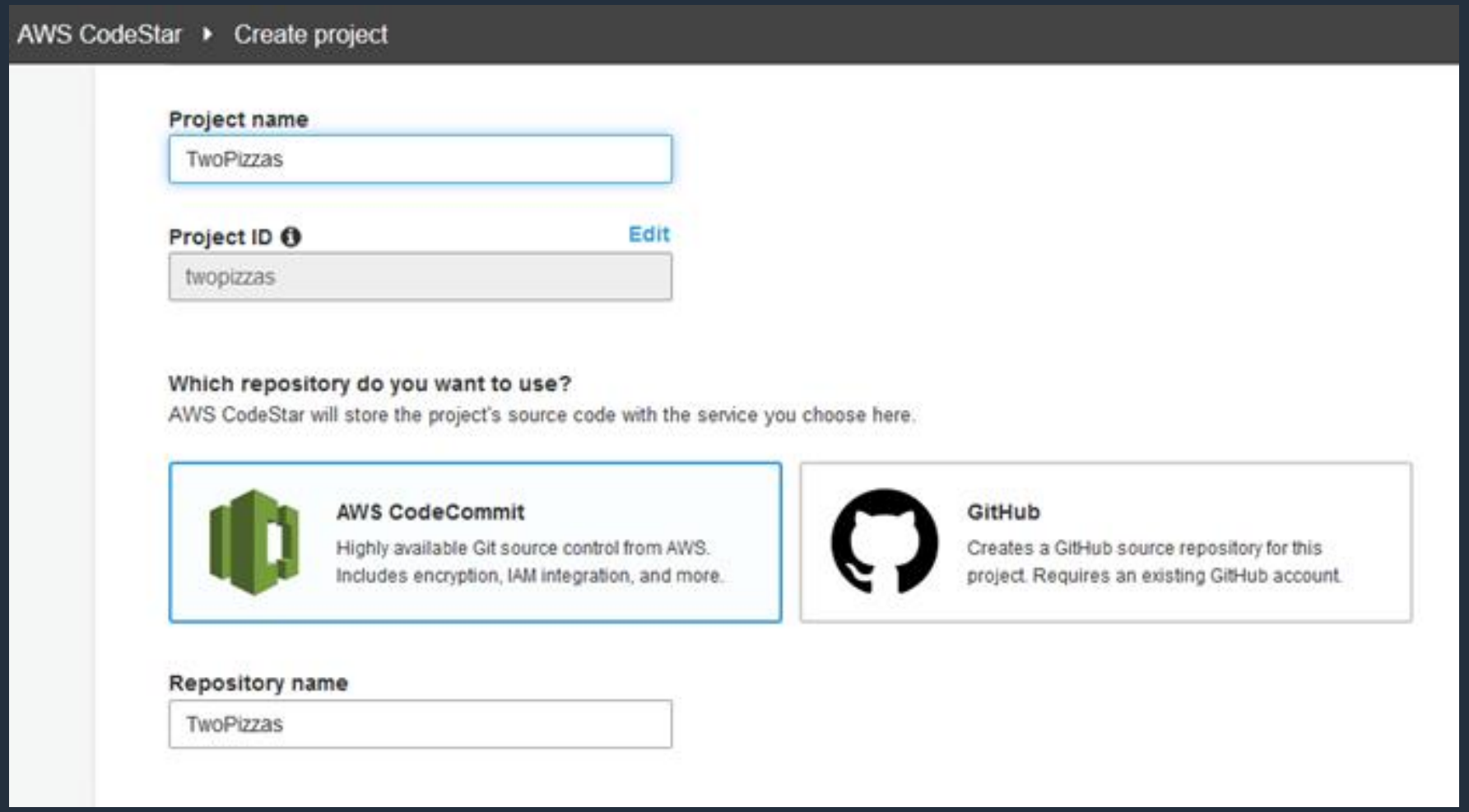
AWS CodeStar

EC2, AWS Lambda 및 Elastic Beanstalk 용 프로젝트 템플릿



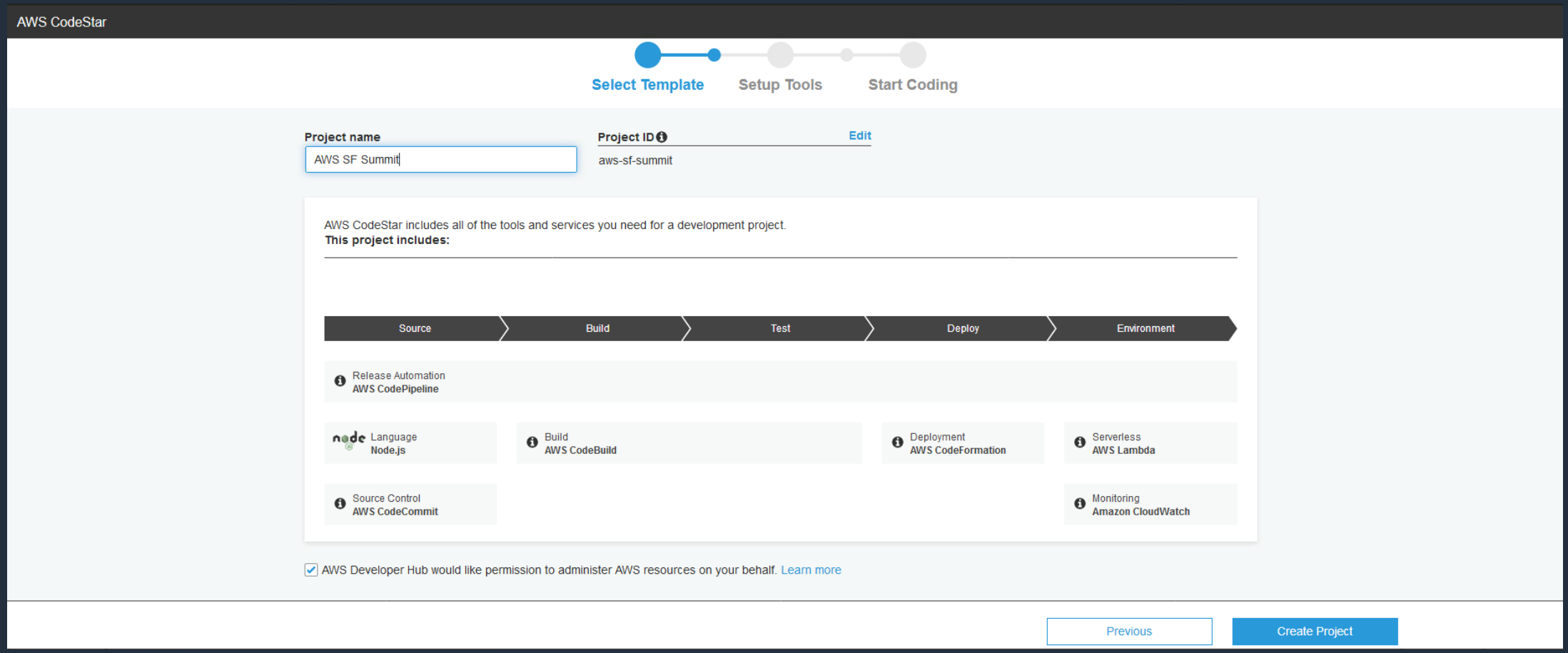
AWS CodeStar

소스 제어 제공자 (Source Control Provider) 선택



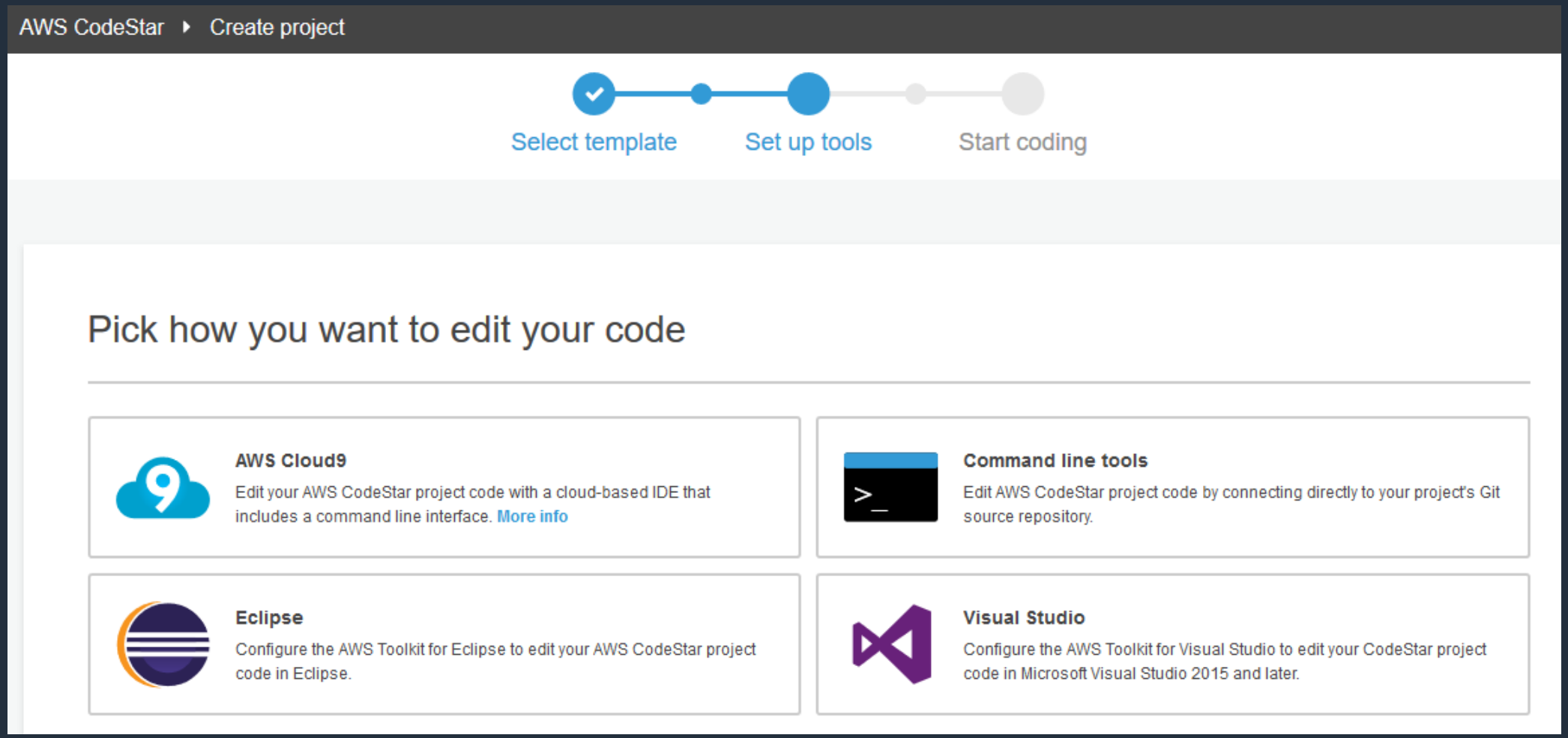
AWS CodeStar

사전 구성된 지속적 전달(Continuous Delivery) 툴체인



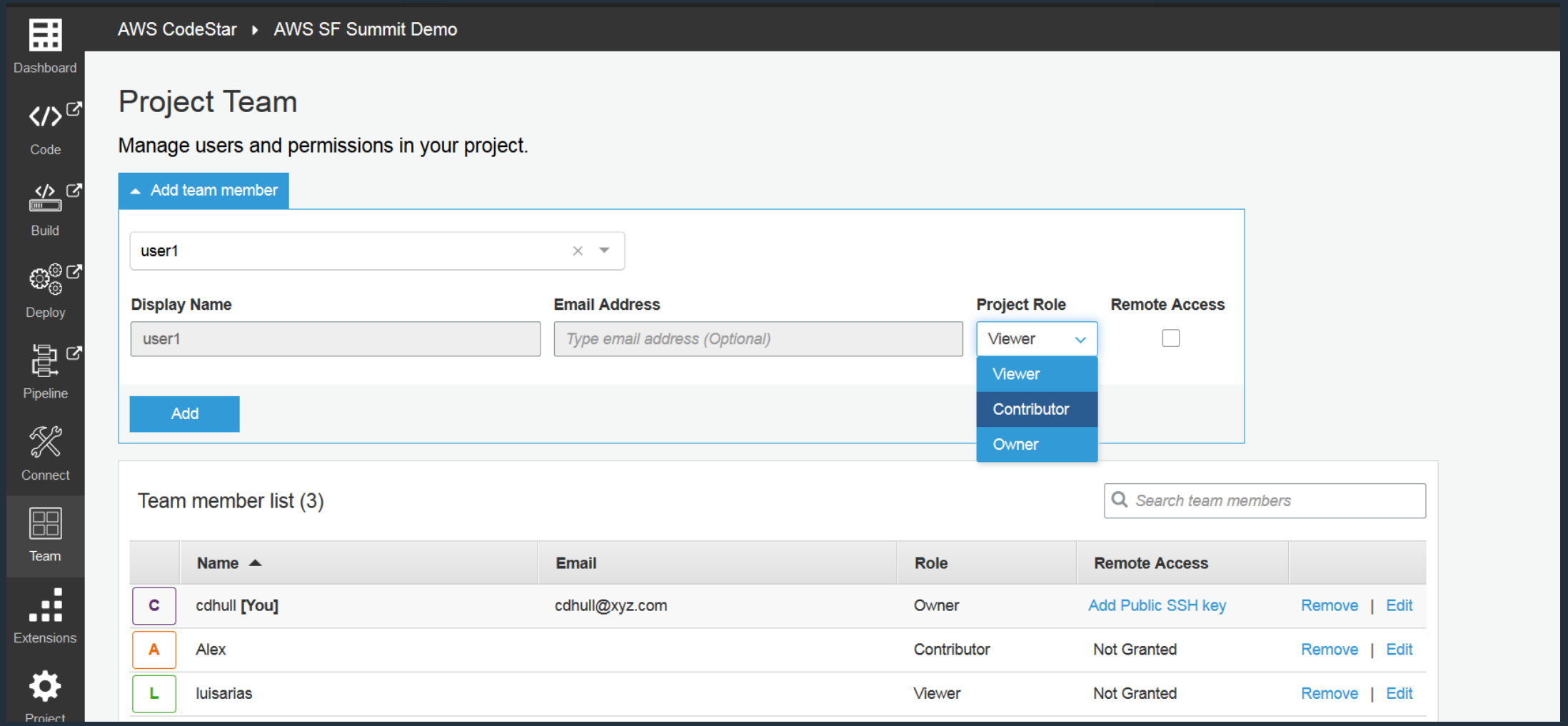
AWS CodeStar

선호하는 IDE 에 쉽게 연결



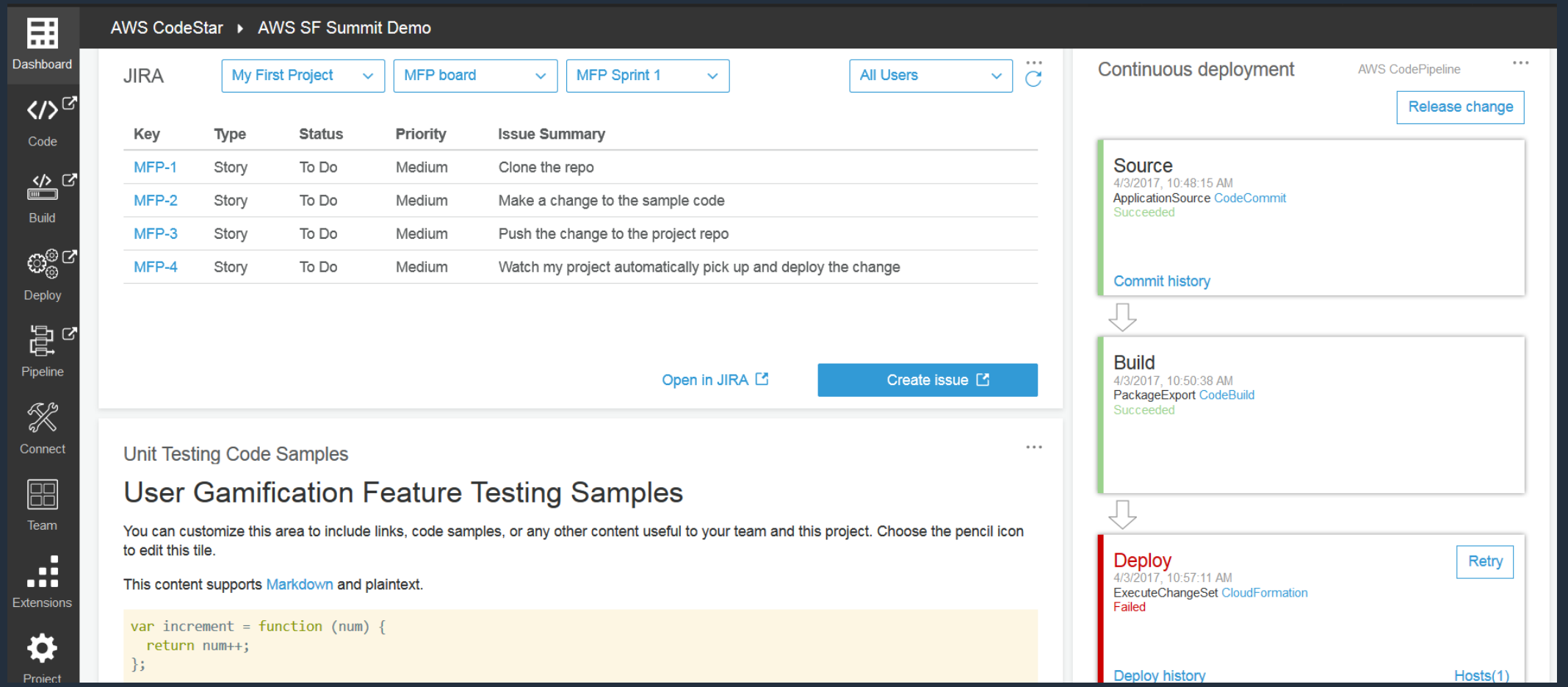
AWS CodeStar

몇 번의 클릭만으로 안전한 팀 액세스 설정



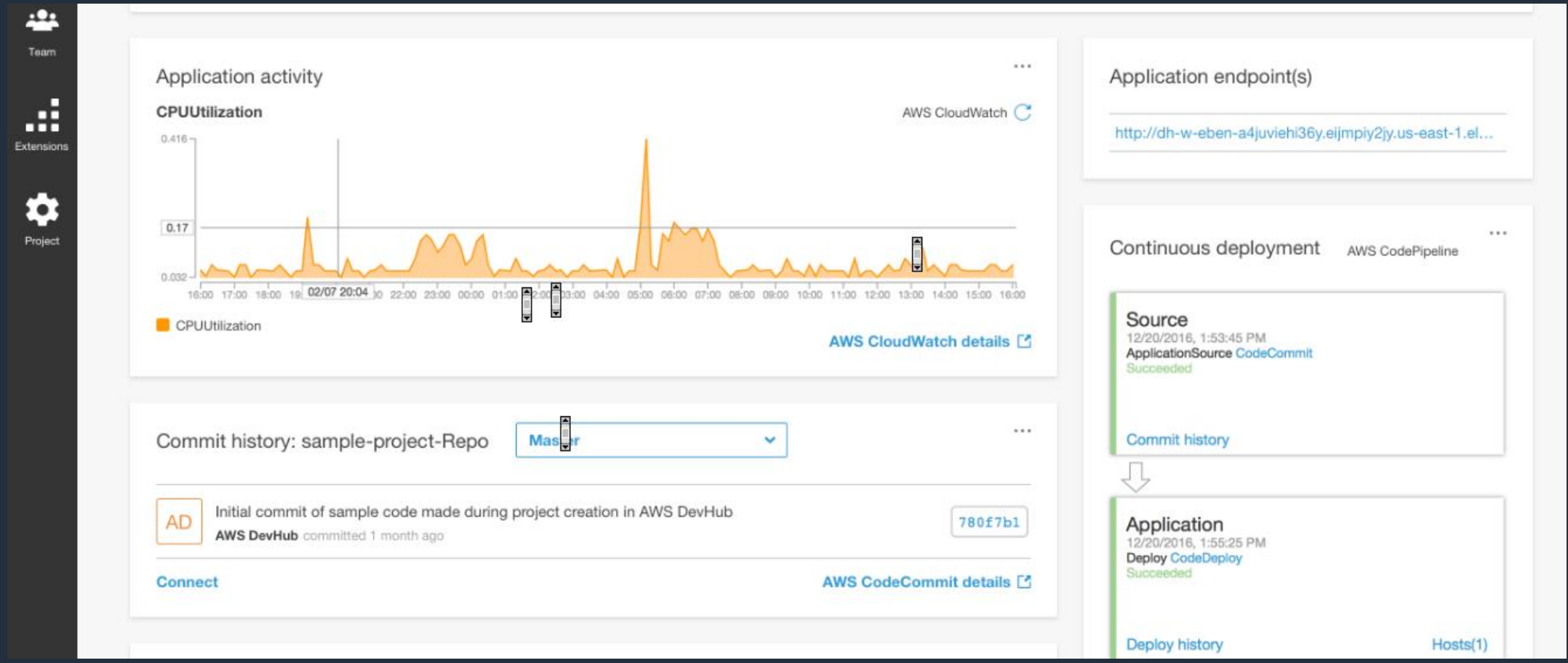
AWS CodeStar

통합 대시 보드 – 전달 파이프 라인 및 문제 추적 관리



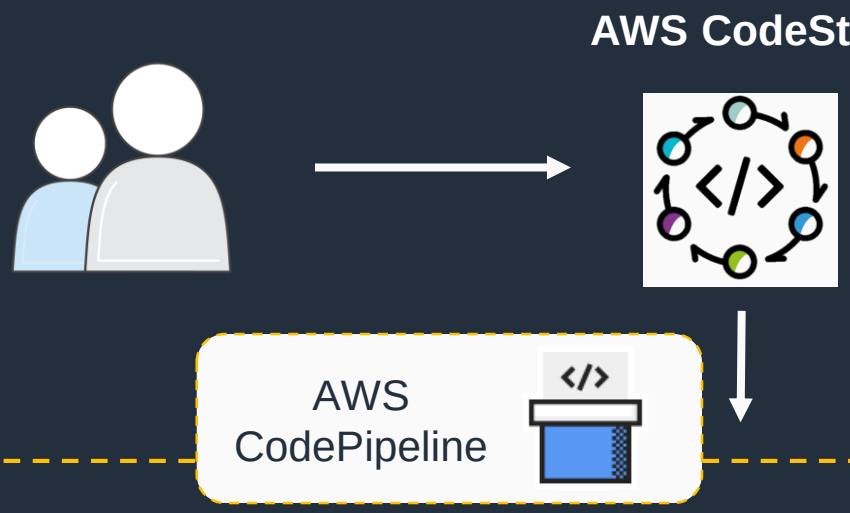
AWS CodeStar

통합 대시 보드 – 어플리케이션 활동 및 **Commit history**

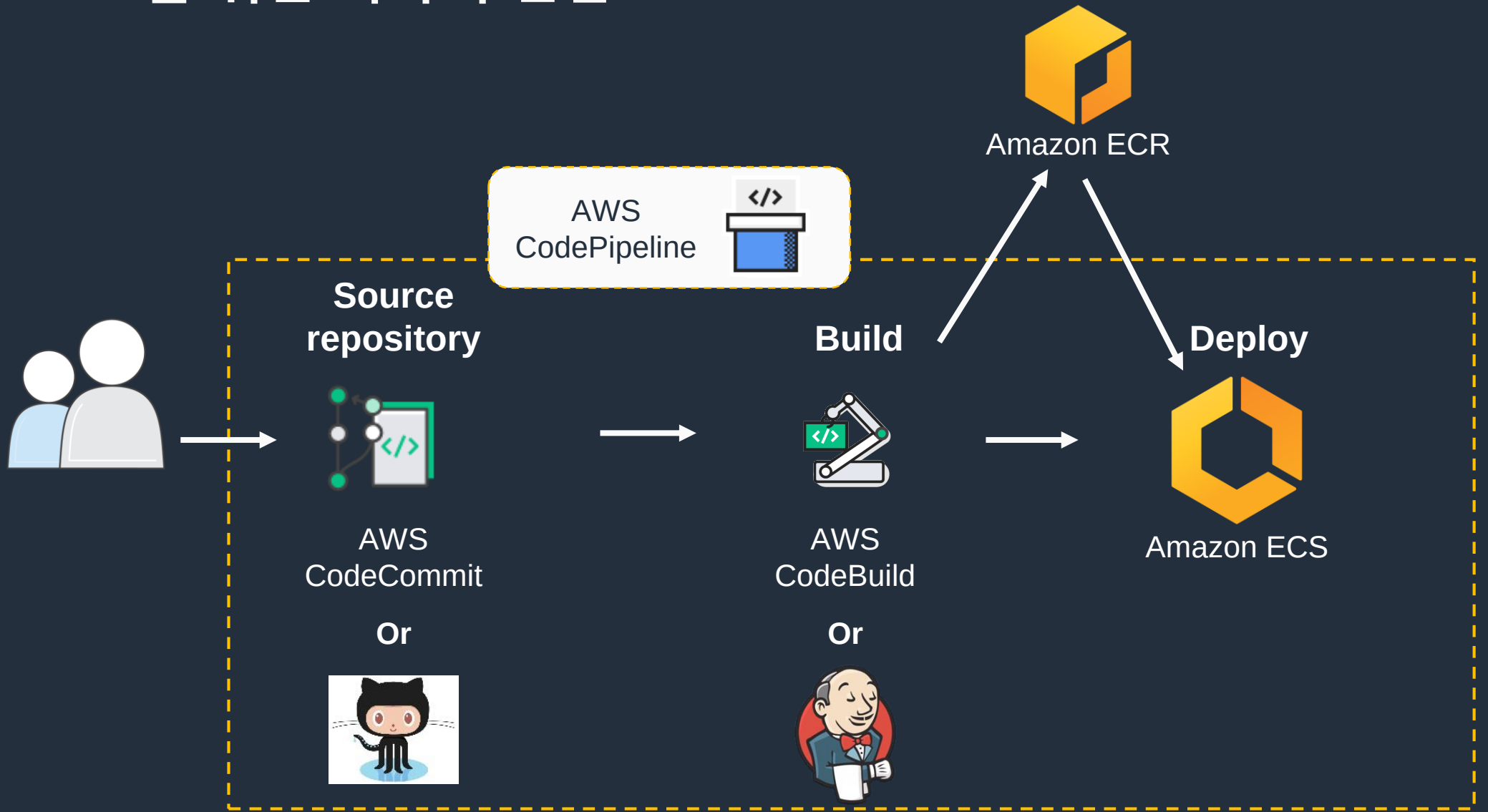


Appendix.
- CI/CD Use Case

Serverless 어플리케이션을 위한 지속적 전달



Container를 위한 지속적 전달



감사합니다!