

Cloud DevOps Engineering

Module3 - AWS 개발 도구 활용하기

David Yoon | CEO
david@2miles.co.kr



INDEX

01 DevOps on AWS

02 AWS의 다양한 개발 도구

03 AWS 개발 도구 활용하기

04 모범 사례

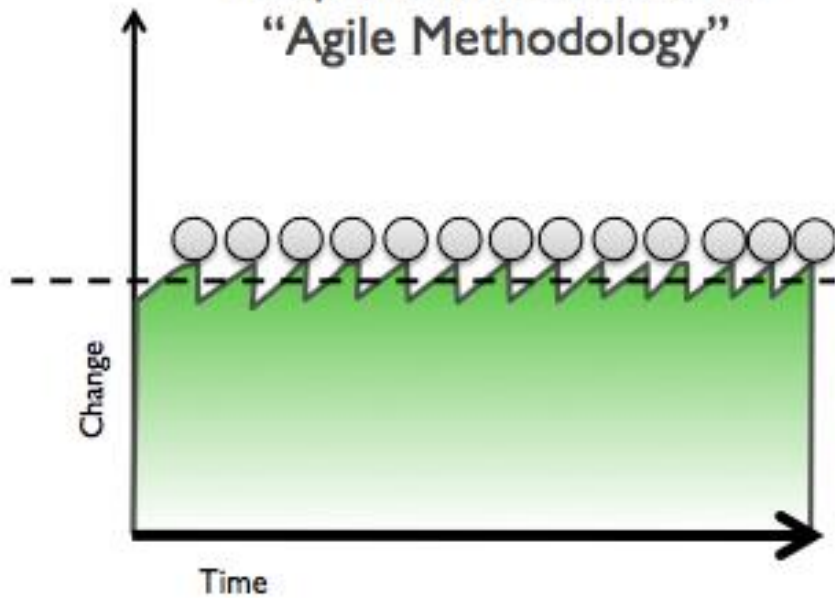
최근 소프트웨어 개발에서 발견되고 있는 큰 특성은...

소프트웨어는 **빠르게 움직인다는 것!**

소프트웨어 제작과 배포가 그 어느때보다 쉬워짐

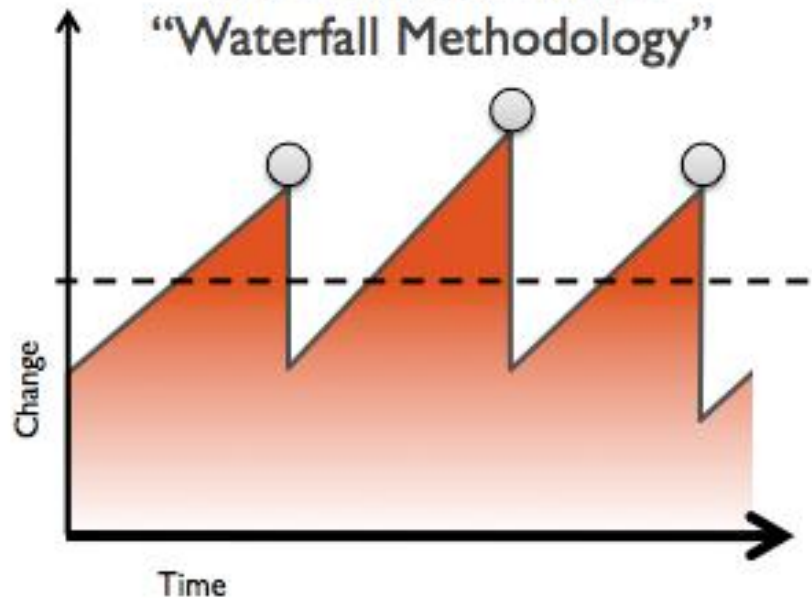
- 스타트업들은 초기 자금 없이도 커질 수 있음
- 수백만 명의 사용자가 소프트웨어를 다운로드 할 수 있음
- 빠른 배포는 장애 복구에 가장 중요함

Frequent Release Events
"Agile Methodology"



Smoother Effort
Less Risk

Rare Release Events
"Waterfall Methodology"



Effort Peaks
High Risk

빠르게 움직이기 위해 필요한 도구는?

다양한 도구를 활용하여 소프트웨어를 빠르게 배포

- 소프트웨어 개발 배포 프로세스의 흐름을 관리할 도구
- 코드에서 결함 및 잠재적 문제를 테스트하는 도구
- 애플리케이션을 배포하는 도구

Develop



Test



Deploy



Monitor

Nagios



Log



Configuration Management



Security



Collaboration Platform

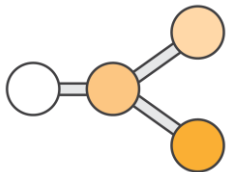


들어가기에 앞서
잠시 릴리즈 프로세스를 확인해봅시다

릴리즈 프로세스 4 단계

Source

- 소스 코드
체크인
- 코드 상호
검토 (peer
review)



Build

- 코드 컴파일
- 유닛 테스트
- 스타일 검증
- 코드 메트릭
- 컨테이너
이미지 제작



Test

- 타 시스템과
통합 테스트
- 부하 테스트
- UI 테스트
- 침투 테스트



Production

- 프로덕션
환경으로
배포

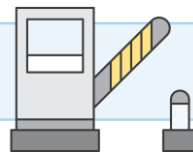


릴리즈 프로세스 수준



지속적 통합: Continuous integration

지속적 전달: Continuous delivery



지속적 배포: Continuous deployment

DEPLOYMENTS AT AMAZON.COM

(2014년 기준)

~11.6s

~1,079

~10,000

~30,000

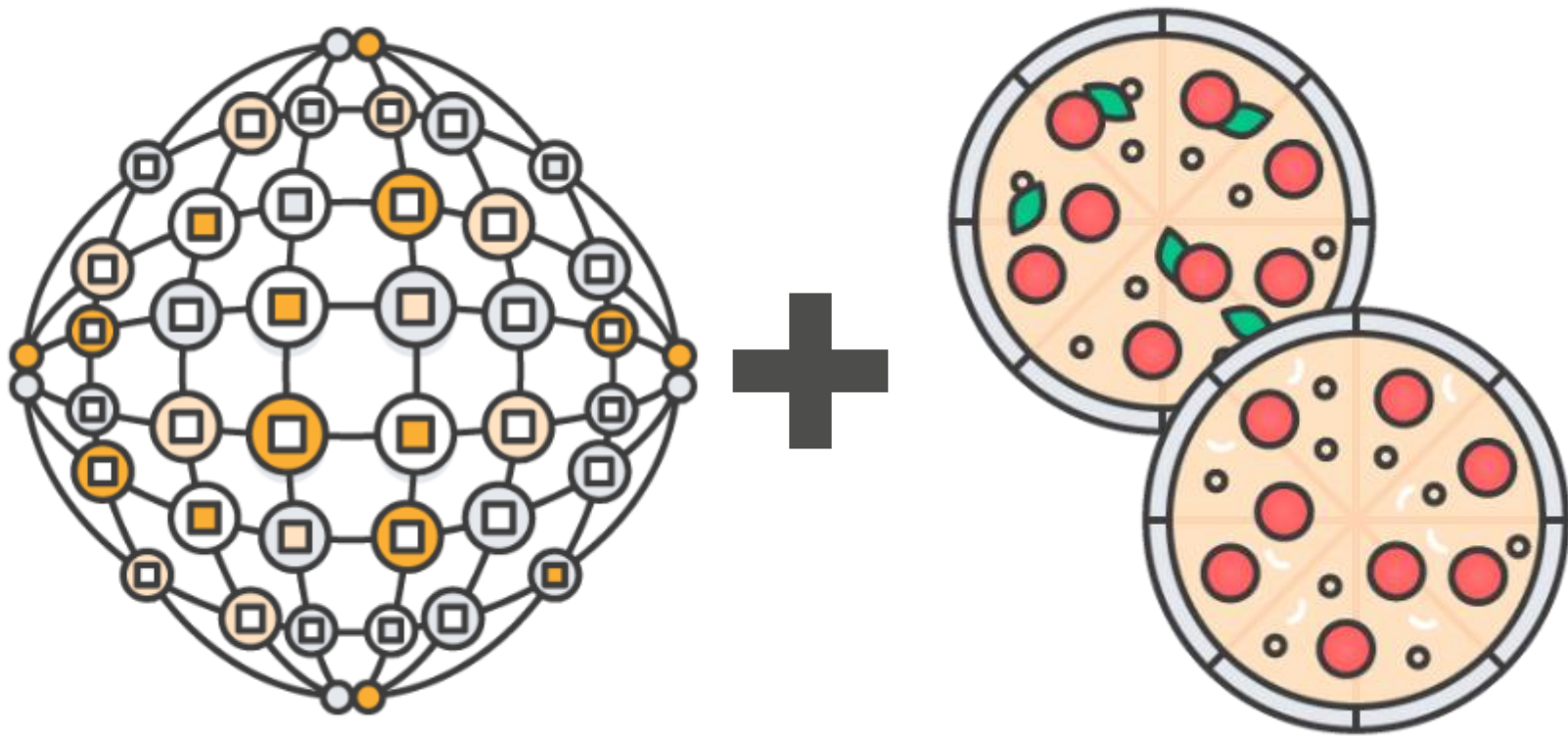
(주간) 배포간 간격

한 시간내의 최대
배포 횟수

동시에 배포 명령을
받는 평균 호스트 수

동시에 배포 명령을
받는 최대 호스트 수

5천만번의 배포



마이크로서비스 + 2-피자 팀

데브옵스 개발 문화

2-피자팀은 각자의 제품을 '소유':

- 제품 제작 (소프트웨어)
- 제품 관련 Q/A 처리
- 문제 발생시 대응 (항시)
 - “you build it, you run it”
- 비즈니스와 기술 메트릭에 대한 서비스 및 트랙/목표 지원



데브옵스 개발 방식

2-피자팀의 방식은 표준을 기반으로 **'최대한 오픈'**:

- Agile? Scrum? Daily standups? Weekly? None? 뭐든 팀에 맞다면!
- 중앙 집중식 변경 관리
게시판/팀/승인 없음, 다만
도구들은 어느정도의
승인이나 프로세스 리뷰가
필요



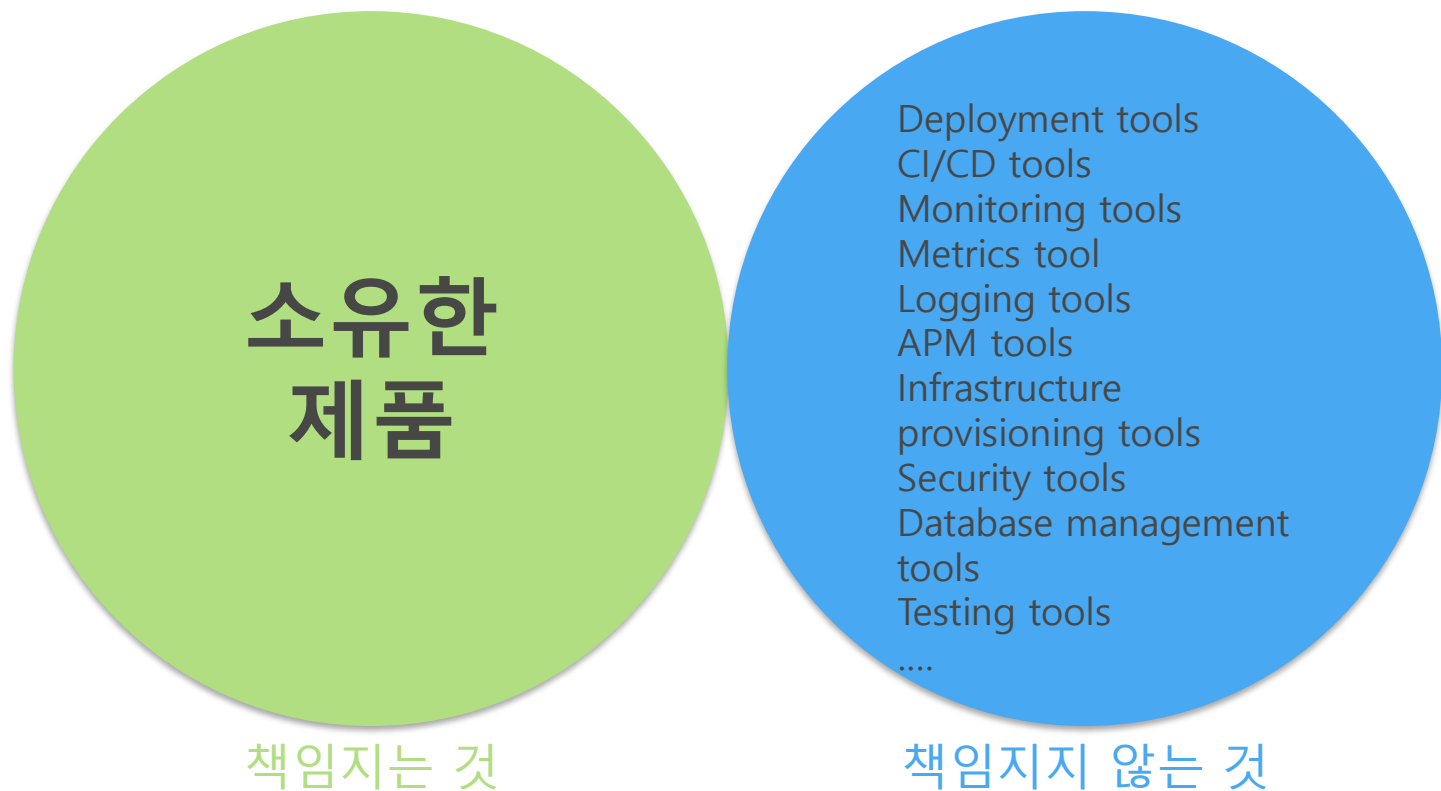
데브옵스 개발 도구

2-피자팀 개발자들의 '도구박스':

- 사용하거나 직접 운영:
 - 운영에 소용되는 시간은 개발에 소요되는 시간 보다 적음
 - 개발에 적은 시간을 사용한다면 목표를 놓칠 위험성이 증가함
- 기존에 성공적으로 동작하는 최고의/최대의 결과를 유도
- 도구는 다른 2-피자팀에 의해 관리됨



2-피자팀의 책임 벤다이어그램



*해당 제품이 파란색에 속하지 않는 한

이런 배경과, 결과를 위해 도구를
만들었습니다



배포 서비스

중단 없는 배포

헬스 체크

버저닝된 아티팩트와
롤백



Pipelines

체크인 부터
프로덕션까지
자동화된 액션과 전환

개발 이점:

- 더 빠른
- 더 안전한
- 일관성 & 표준화
- 프로세스의 시각화

매년 소프트웨어 개발자들을
대상으로 설문조사를 실시한
결과 2014년에 **행복한**
개발자들과 통계적으로 관련있는
단 하나의 개발 도구/서비스를
발견함



매년 소프트웨어 개발자들을
대상으로 설문조사를 실시한
결과 2014년에 **행복한**
개발자들과 통계적으로 관련있는
단 하나의 개발 도구/서비스를
발견함



Pipelines 서비스!

INDEX

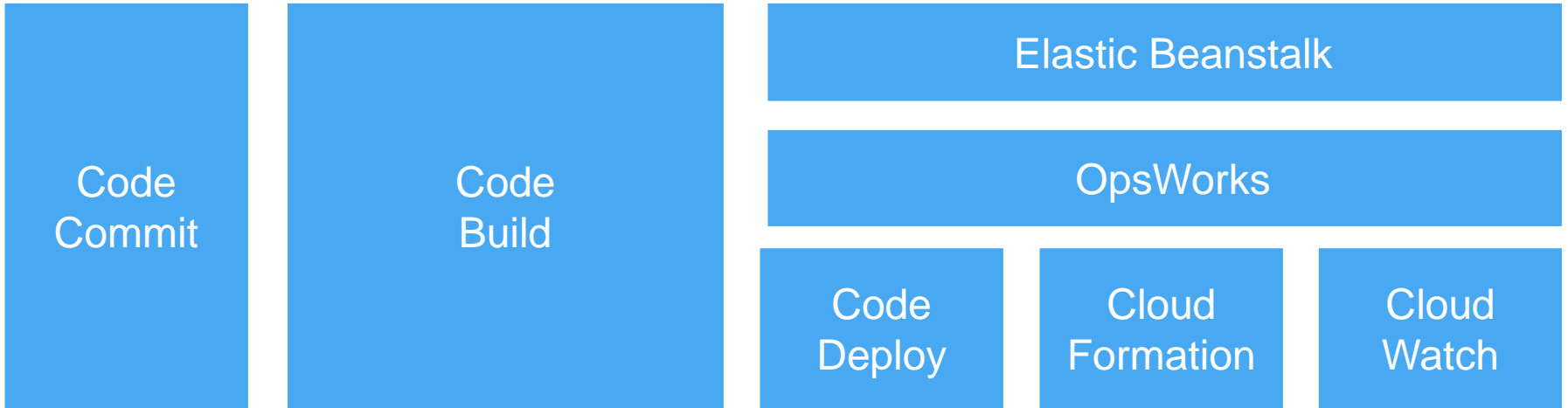
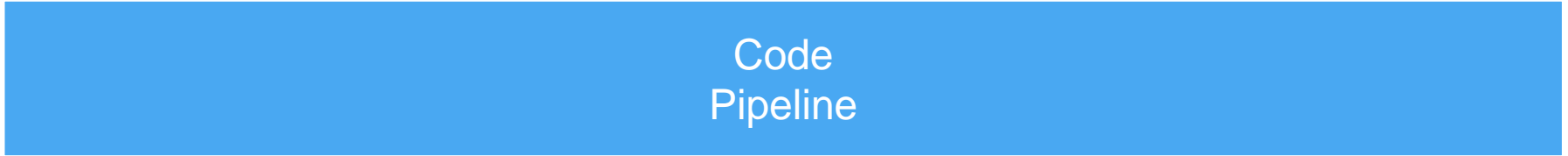
01 DevOps on AWS

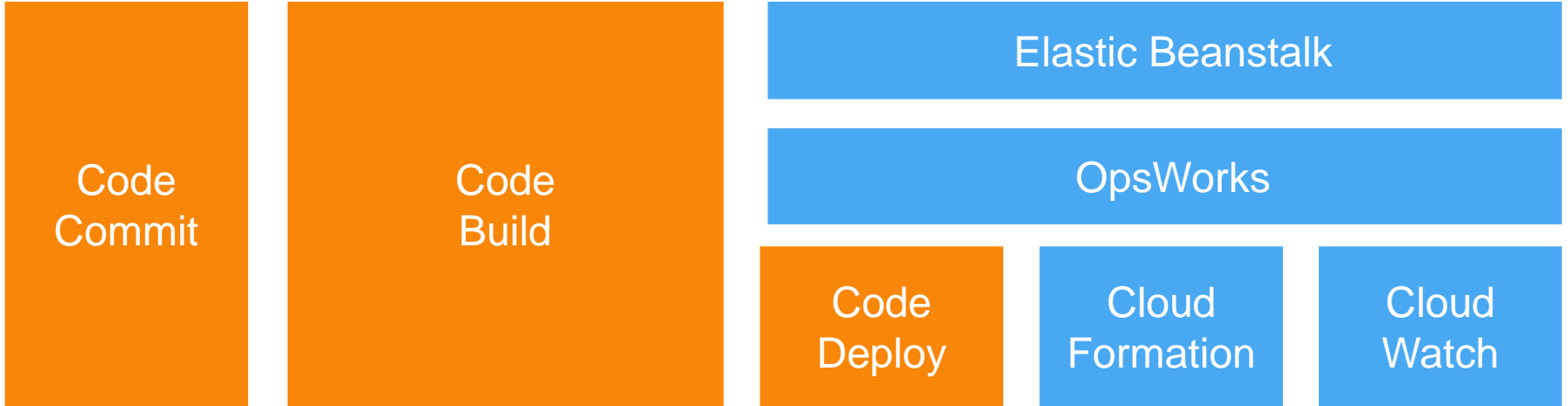
02 AWS의 다양한 개발 도구

03 AWS 개발 도구 활용하기

04 모범 사례

지속적인 배포 = 개발자들의 행복





AWS Code* 서비스



AWS CodePipeline



AWS CodeDeploy



AWS CodeCommit



AWS CodeBuild

AWS Code* 서비스

소프트웨어 릴리즈 단계:



Commit

Build

Test

Production

AWS Code* 서비스

소프트웨어 릴리즈 단계:

Commit

Build

Test

Production



AWS CodeCommit

AWS Code* 서비스

소프트웨어 릴리즈 단계:

Commit

Build

Test

Production



AWS CodeBuild

또는



Third-Party
Tooling

AWS Code* 서비스

소프트웨어 릴리즈 단계:

Commit

Build

Test

Production



AWS CodeDeploy

AWS Code* 서비스

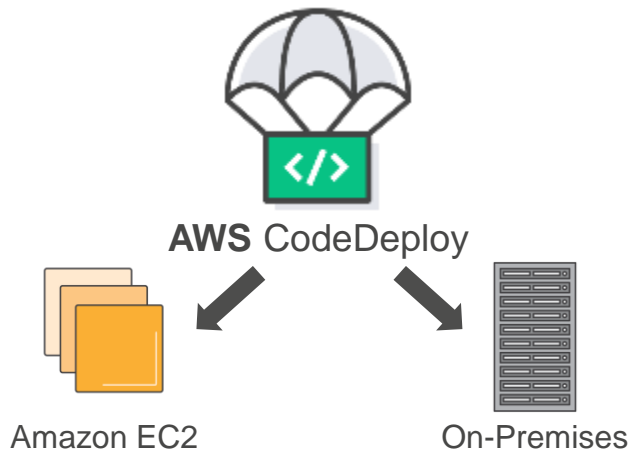
소프트웨어 릴리즈 단계:

Commit

Build

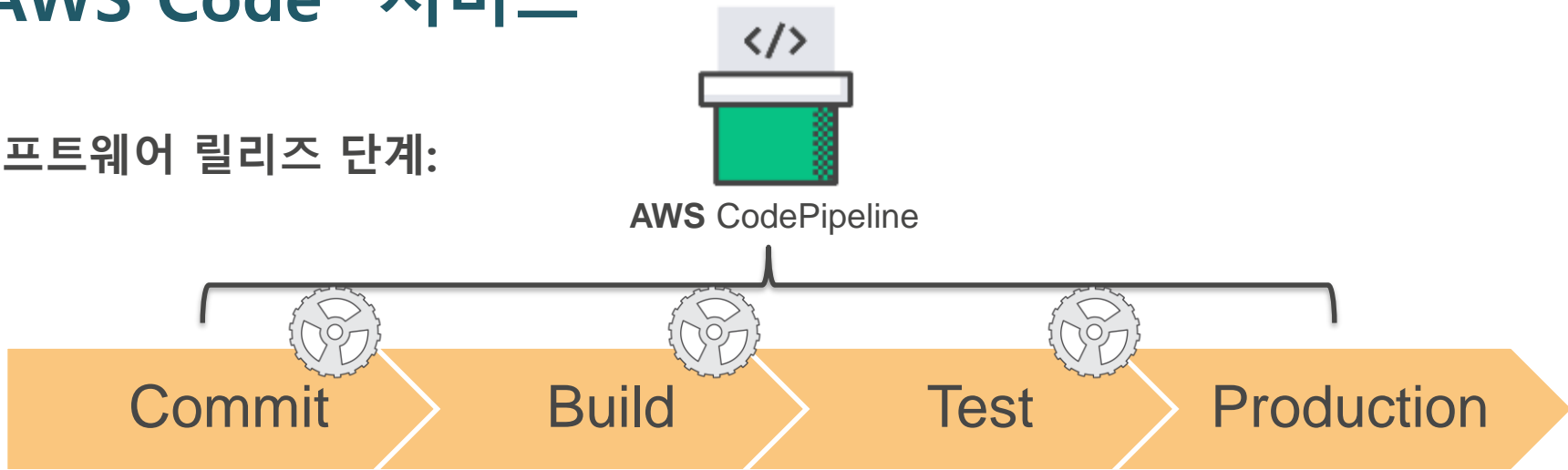
Test

Production



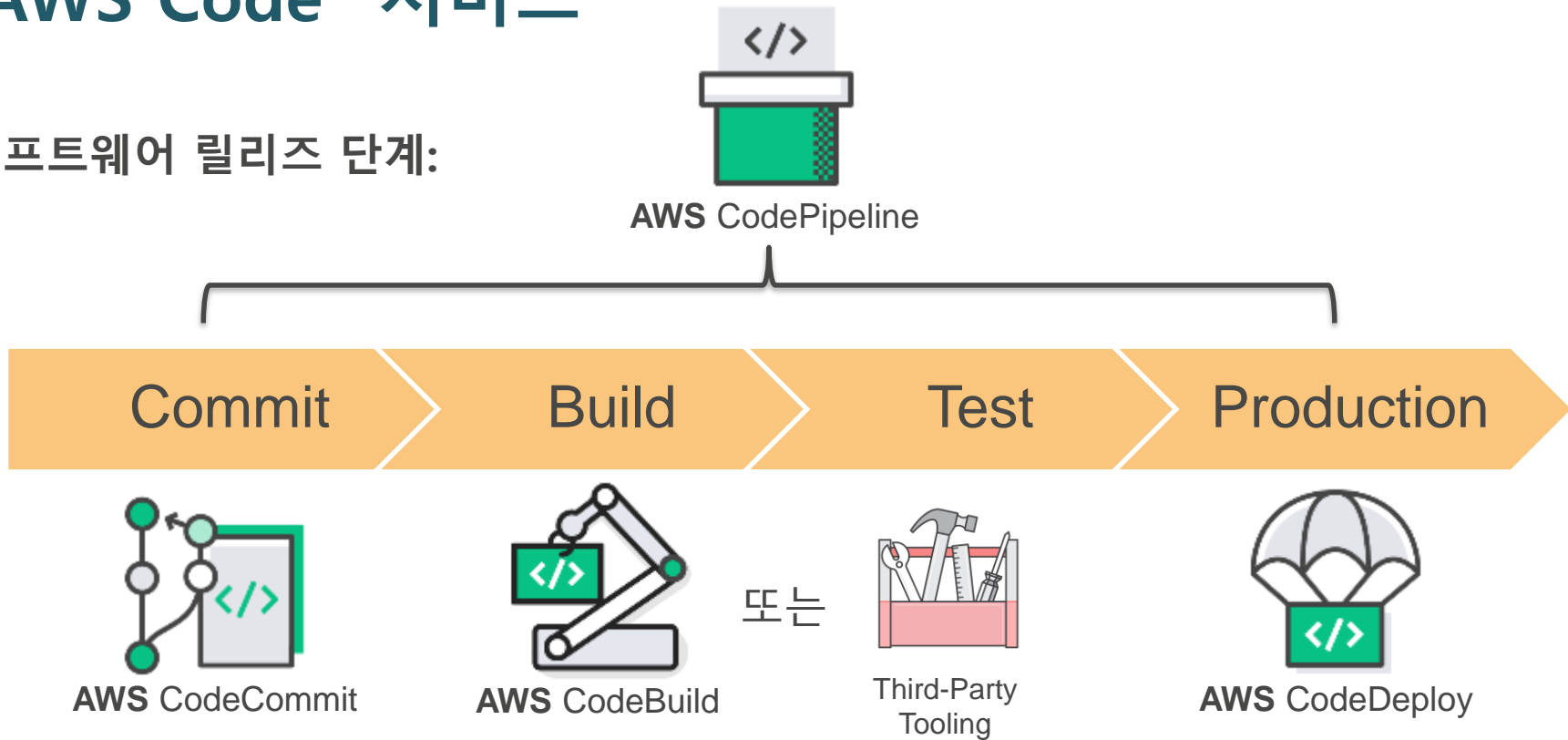
AWS Code* 서비스

소프트웨어 릴리즈 단계:

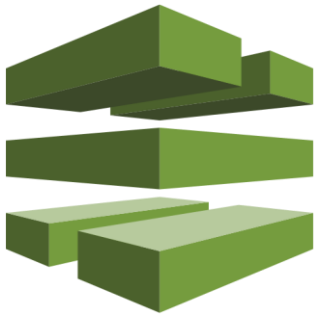


AWS Code* 서비스

소프트웨어 릴리즈 단계:



AWS CodePipeline



- 신속하고 신뢰할 수 있는 애플리케이션 업데이트를 위한 지속적인 전달 서비스
- 소프트웨어 릴리즈 프로세스 모델링 및 시각화
- 코드가 변경될 때마다 빌드, 테스트, 배포
- AWS 와 다양한 도구들과의 통합



AWS CodePipeline



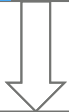
Source

Source
CodeCommit



Build

JenkinsOnEC2
Jenkins



Deploy

App
CodeDeploy





AWS CodePipeline

Source

Source
CodeCommit



Build

JenkinsOnEC2
Jenkins



Deploy

App
CodeDeploy



Pipeline



AWS CodePipeline

Source

Source
CodeCommit



Build

JenkinsOnEC2
Jenkins



Deploy

App
CodeDeploy



Stage





AWS CodePipeline



Source

Source
CodeCommit

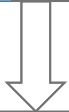


Build

JenkinsOnEC2
Jenkins



Action



Deploy

App
CodeDeploy





AWS CodePipeline

Source

Source
CodeCommit



Build

JenkinsOnEC2
Jenkins



Deploy

App
CodeDeploy



Transition





AWS CodePipeline

Source

Source
CodeCommit



Build

JenkinsOnEC2
Jenkins



Deploy

App
CodeDeploy



AWS CodePipeline

Source

Source
CodeCommit



Build

JenkinsOnEC2
Jenkins



NotifyDevelopers
Lambda



Deploy

App
CodeDeploy





AWS CodePipeline

Source

Source
CodeCommit



Build

JenkinsOnEC2
Jenkins



Deploy

App
CodeDeploy



AWS CodePipeline

Source

Source
CodeCommit



Build

JenkinsOnEC2
Jenkins



NotifyDevelopers
Lambda



Deploy

App
CodeDeploy



Parallel actions



AWS CodePipeline



Source

Source
CodeCommit



Build

JenkinsOnEC2
Jenkins



NotifyDevelopers
Lambda



Deploy

App
CodeDeploy



AWS CodePipeline

Source

Source
CodeCommit



Build

JenkinsOnEC2
Jenkins



NotifyDevelopers
Lambda



TestAPI
Runscope



Deploy

App
CodeDeploy





AWS CodePipeline



Source

Source
CodeCommit



Build

JenkinsOnEC2
Jenkins



NotifyDevelopers
Lambda



Deploy

App
CodeDeploy



AWS CodePipeline

Source

Source
CodeCommit



Build

JenkinsOnEC2
Jenkins



NotifyDevelopers
Lambda



TestAPI
Runscope



Sequential
actions

Deploy

App
CodeDeploy





MyApplication

Build

JenkinsOnEC2
[Jenkins](#)



Staging-Deploy

JavaApp
[Elastic Beanstalk](#)



QATeamReview
[Manual Approval](#)



Review

Manual approvals

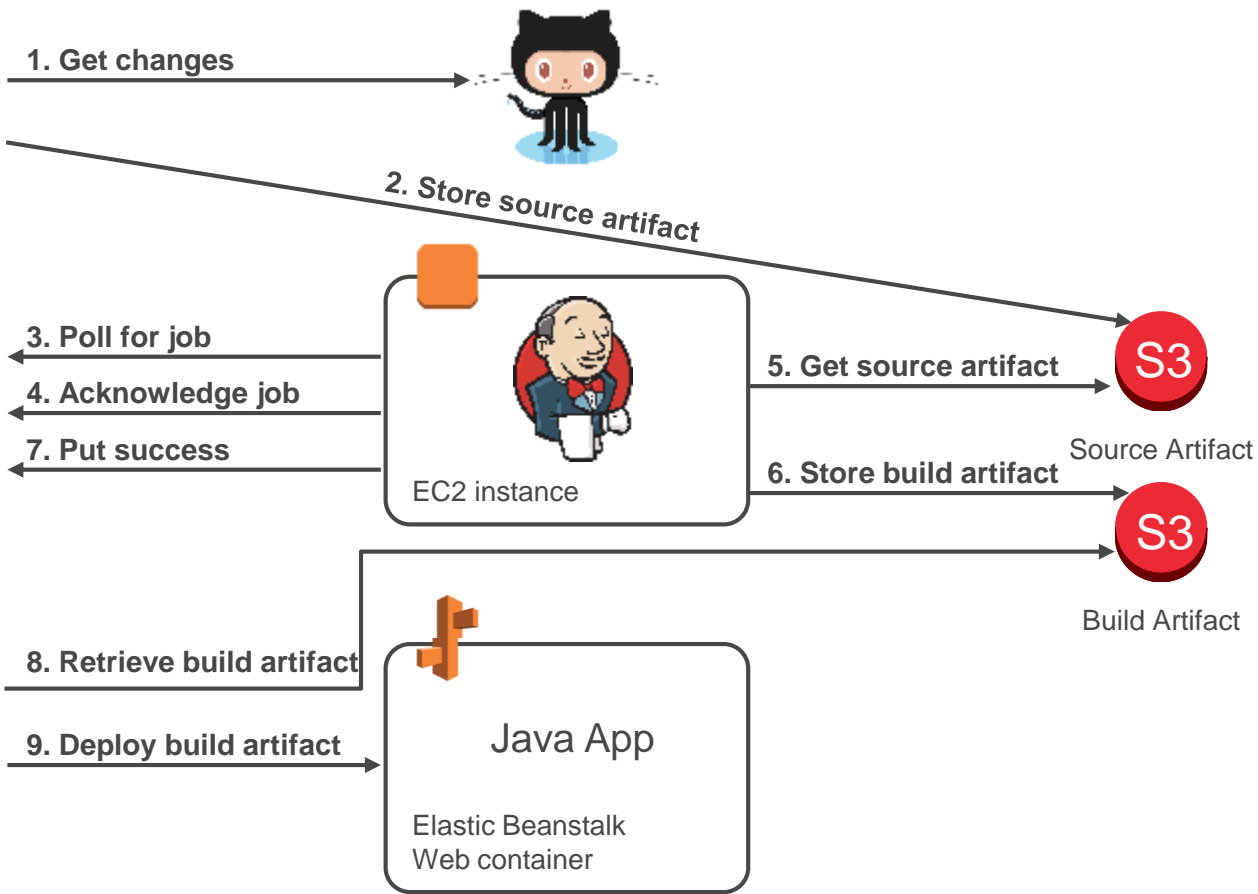
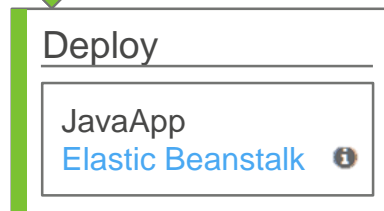
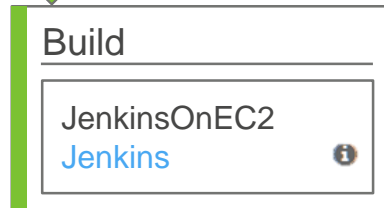
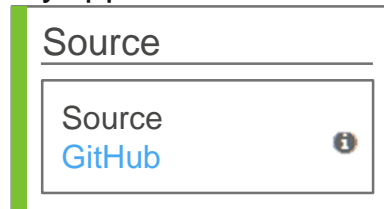
Prod-Deploy

JavaApp
[Elastic Beanstalk](#)



CodePipeline

MyApplication



AWS 서비스들과 통합

Source

Build

Invoke Logic

Deploy

Amazon S3

AWS CodeBuild

AWS Lambda

AWS CodeDeploy

AWS CodeCommit

AWS CloudFormation

AWS Elastic Beanstalk

AWS OpsWorks

지속적으로 증가하는 파트너들

Source

Build

Test

Deploy

GitHub


CloudBees

 **Apica**

XebiaLabs
Deliver Faster



Jenkins

 **BlazeMeter**




Ghost Inspector

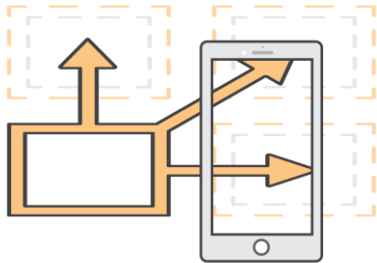

Solano Labs

 **HPE StormRunner**

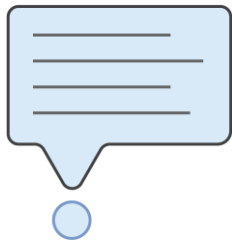
 **TeamCity**

 **Runscope**

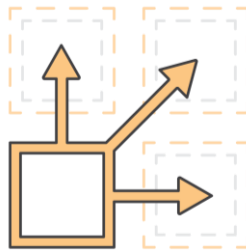
사용자 정의 동작으로 AWS CodePipeline 확장



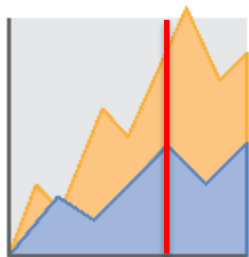
모바일 테스트



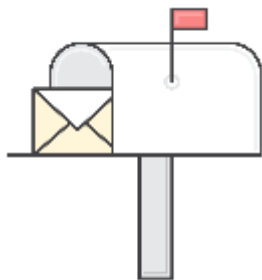
티켓 업데이트



리소스 프로비저닝



대시보드 업데이트



알림 전송



보안 검사

확장 방법 선택

Lambda	사용자 정의 동작
단기 실행 작업은 쉽게 구축 가능	모든 워크로드 수행가능
장기 실행 작업은 더 많은 작업이 필요	콘솔에 표시된 링크를 제어
Node.js, Python, Java, C# 지원	모든 언어 지원
AWS 에서 실행	온프레미스에서 실행 가능
프로비저닝하거나 관리 할 서버 없음	컴퓨팅 자원 필요

AWS CodeBuild



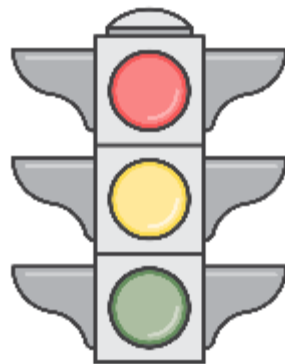
- 소스 코드를 컴파일 하고 테스트를 실행하며, 소프트웨어 패키지를 생성하는 완전 관리형 빌드 서비스
- 지속적으로 스케일을 조정하고 여러 빌드를 동시에 처리
- Docker 이미지를 통해 사용자 요구에 맞는 사용자 지정 빌드 환경 제공 가능
- 사용하는 컴퓨팅 리소스에 대해 분 단위로 과금 발생
- CodePipeline 과 Jenkins 와 통합

AWS CodeBuild 작동 방식

1. 소스 코드 다운로드
2. 임시 컴퓨팅 컨테이너(매 빌드마다 새로 생성)에서 buildspec 에 정의된 명령을 실행
3. 서비스 콘솔과 CloudWatch logs 에 빌드 출력을 스트리밍
4. 생성된 아티팩트를 S3 버킷에 업로드

CodeBuild 로 릴리즈 프로세스 자동화

- CI/CD 를 위해 AWS CodePipeline 과 통합
- API/CLI 을 활용하여 쉽게 탈부착 가능
- 자신만의 빌드 환경을 활용
 - 필요로 하는 도구를 포함한 도커 이미지 생성
- 오픈소스 Jenkins 플러그인
 - Jenkins 마스터의 워커로 CodeBuild 를 활용



buildspec.yml 예제

```
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
```

buildspec.yml 예제

```
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 단계에서 사용할 변수

- 빌드 단계에서 수행할 수 있는 작업의 예:

- "install" 에서 환경을 준비하기 위해 패키지를 설치하거나 명령을 실행 가능
- "pre_build" 에서 구문 확인 및 명령들을 수행
- "build" 에서 빌드 도구/명령을 실행
- "post_build" 에서 추가적인 앱 테스트 또는 저장소로 컨테이너 이미지를 저장

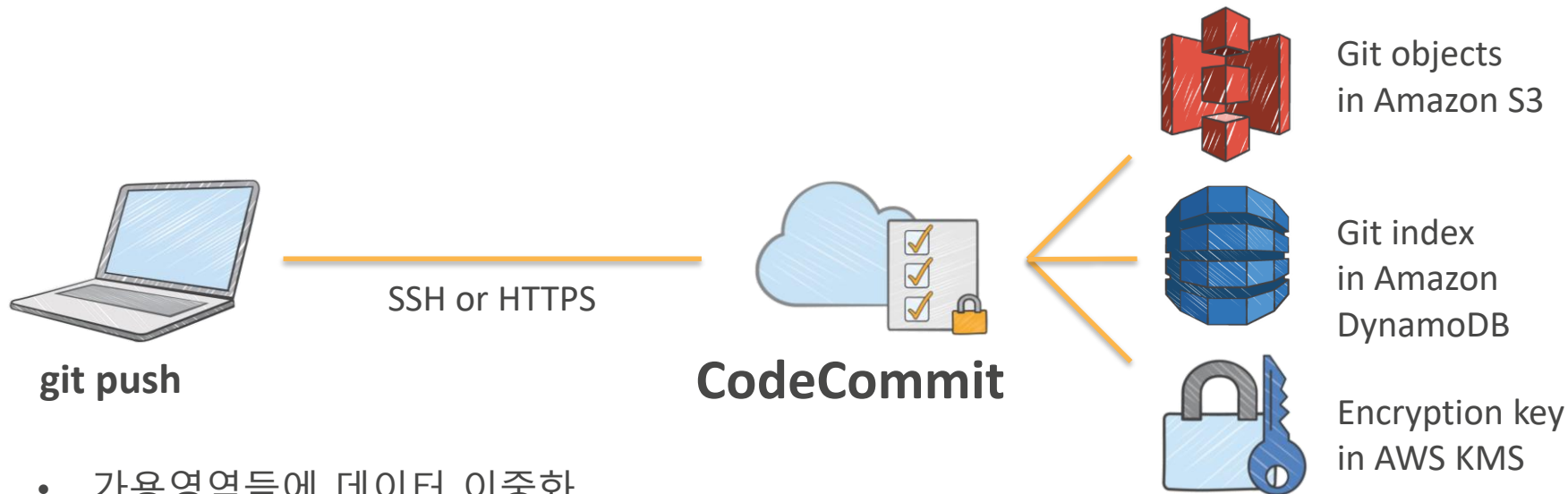
- S3 에 아티팩트를 생성 및 저장

AWS CodeCommit



- 완전 관리형 코드 저장소
- 전송 및 저장시 암호화를 통해 코드를 안전하게 관리
- 많은 파일이나 브랜치, 로그를 유지할 수 있도록 쉽게 확장
- 저장할 수 있는 파일의 크기나 종류에 제한 없이 어떤 파일도 저장 가능
- 기존 Git 도구를 그대로 활용
- 코드/커밋/시각화/비교/트리거

AWS의 코드 리비전 컨트롤 서비스



- 가용영역들에 데이터 이중화
- 데이터 암호화
- AWS IAM과 통합
- 무한 확장 (Repo 크기제한 없음)
- Git 명령을 지원하는 기존 도구와의 호환
 - (Visual Studio, Jenkins, Asana, ZenDesk, Jira, Eclipse, etc.)

대시보드

코드

커밋

시각화 도우미

비교 **New**

트리거

설정

◀ 코드: devopsdemo



The repository for the Summit Demo Code

브랜치: master ▼

URL 복제 ▼

i 연결

devopsdemo

cloudformation

codedeploy

config

appspec.yml

buildspec.yml

LICENSE

main.go

README.md

README.md

AWS DevOps using AWS Code Services!!

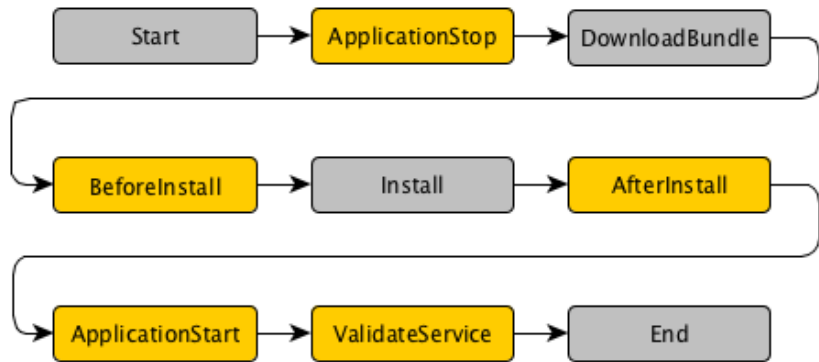
AWS CodeDeploy



- 모든 인스턴스에 코드 배포 자동화
- 애플리케이션 업데이트의 복잡성을 처리
- 애플리케이션 배포 중 다운 타임 회피
- 오류가 감지되면 자동으로 롤백
- 언어 및 운영체제 관계 없이 Amazon EC2 또는 온프레미스 서버에 배포
- AWS 와 다양한 도구들과의 통합

Application Specification File

YAML 포맷의 Application Specification File (AppSpec file)



Hooks

```
version: 0.0
os: linux
files:
  - source: /
    destination: /var/www/html/WordPress
hooks:
  BeforeInstall:
    - location: scripts/install_dependencies.sh
      timeout: 300
      runas: root
  AfterInstall:
    - location: scripts/change_permissions.sh
      timeout: 300
      runas: root
  ApplicationStart:
    - location: scripts/start_server.sh
      timeout: 300
      runas: root
  ApplicationStop:
    - location: scripts/stop_server.sh
      timeout: 300
      runas: root
```

appspec.yml 샘플

appspec.yml 예제

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

appspec.yml 예제

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:

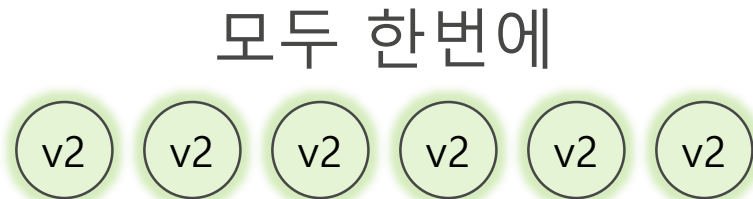
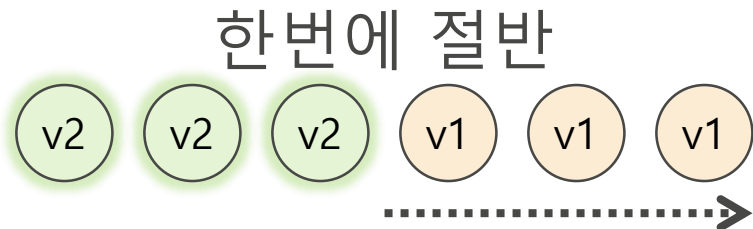
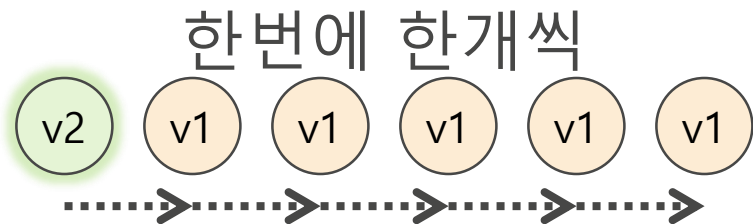
- location: scripts/start_httpd.sh

ValidateService:

- location: scripts/test_site.sh
- location: scripts/register_with_elb.sh

- 애플리케이션 파일과 설정 파일을 디렉토리에 위치
- 특정 디렉토리와 파일에 특정 퍼미션을 설정
- **ELB** 에 인스턴스 추가/제거
- 종속 패키지 설치
- 아파치 실행
- 배포 성공 유무 확인
- 기타 등등

배포 속도와 그룹 지정



배포 자동화



배포 자동화

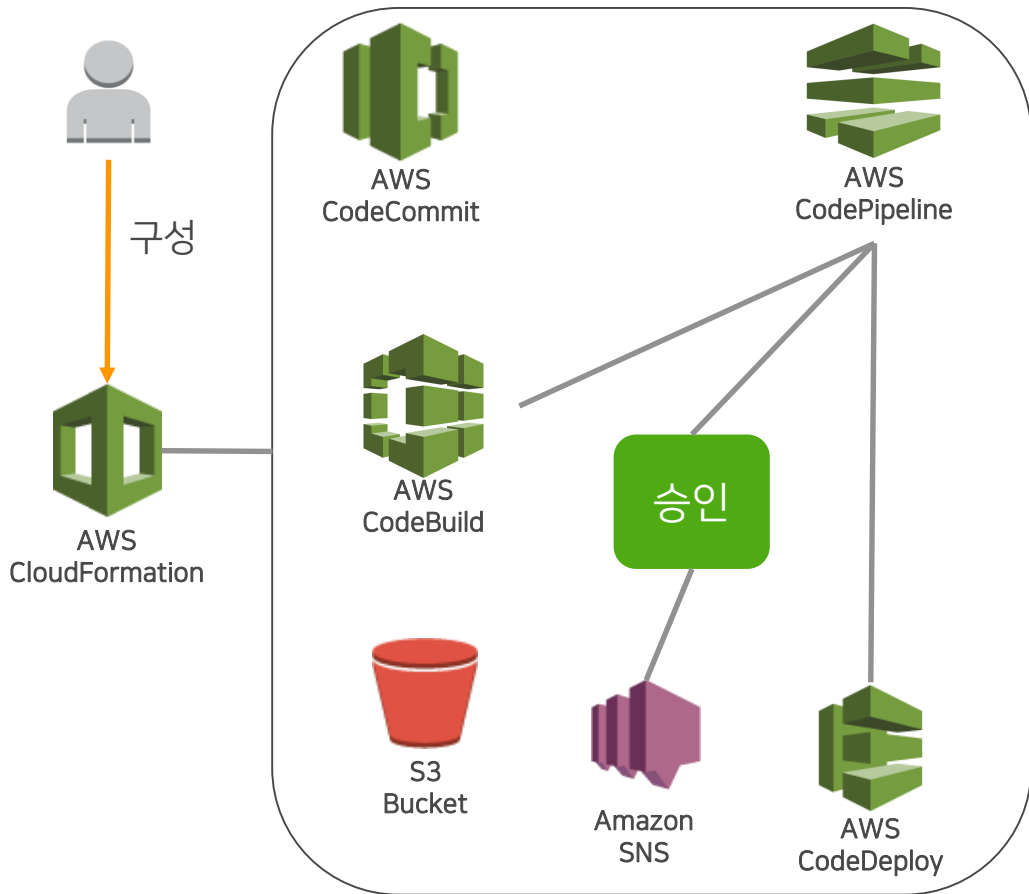


구성

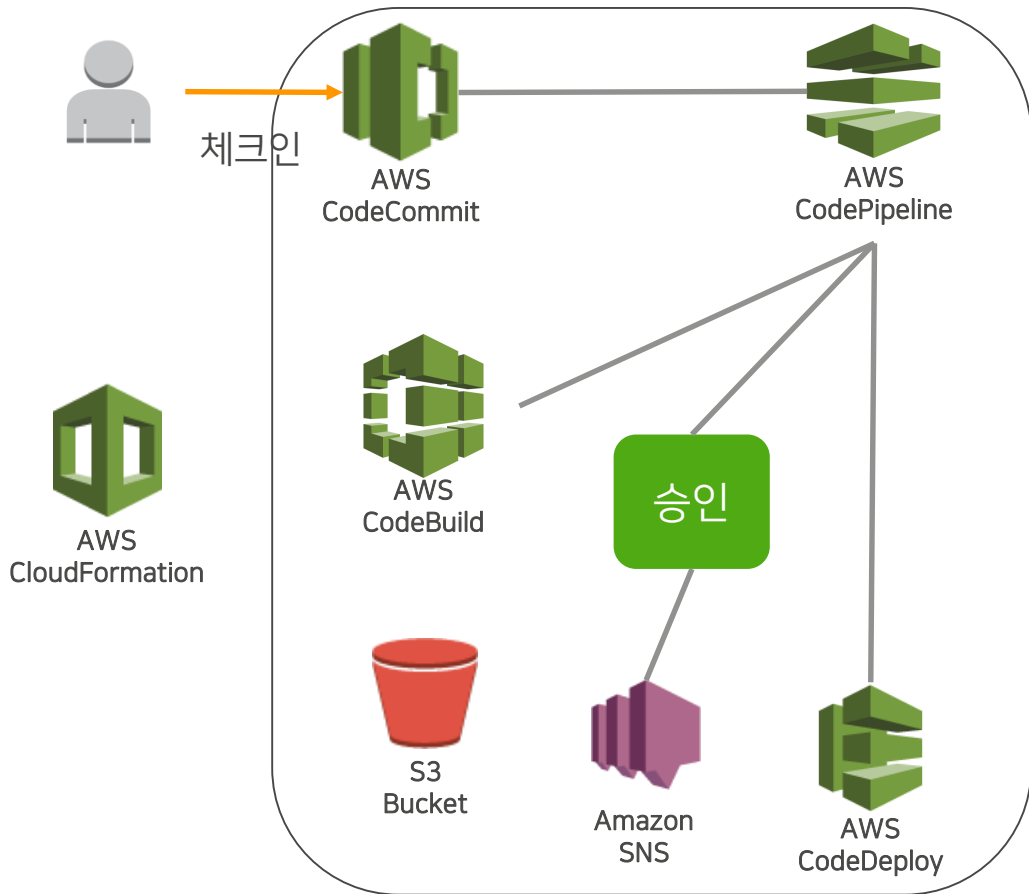


AWS
CloudFormation

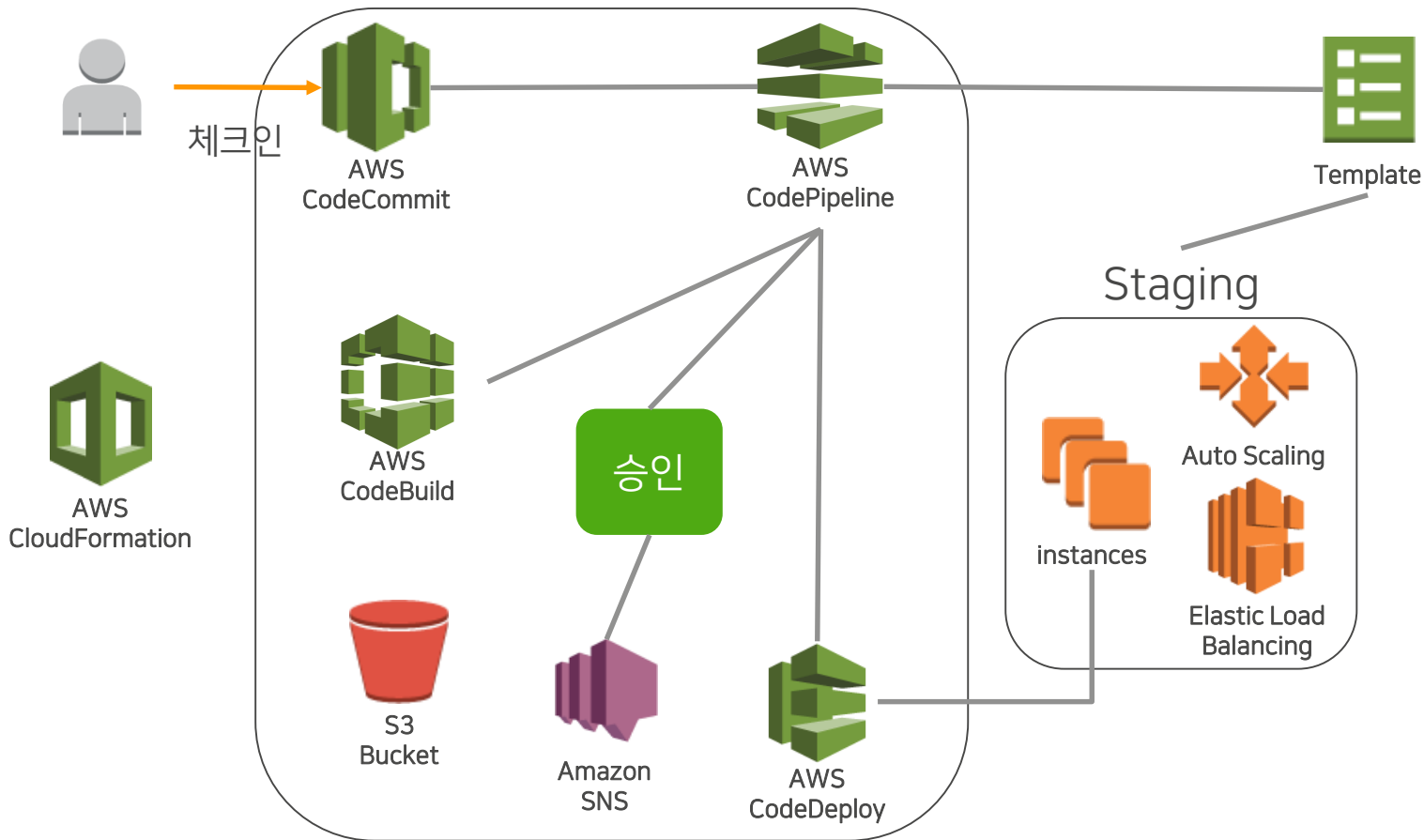
배포 자동화



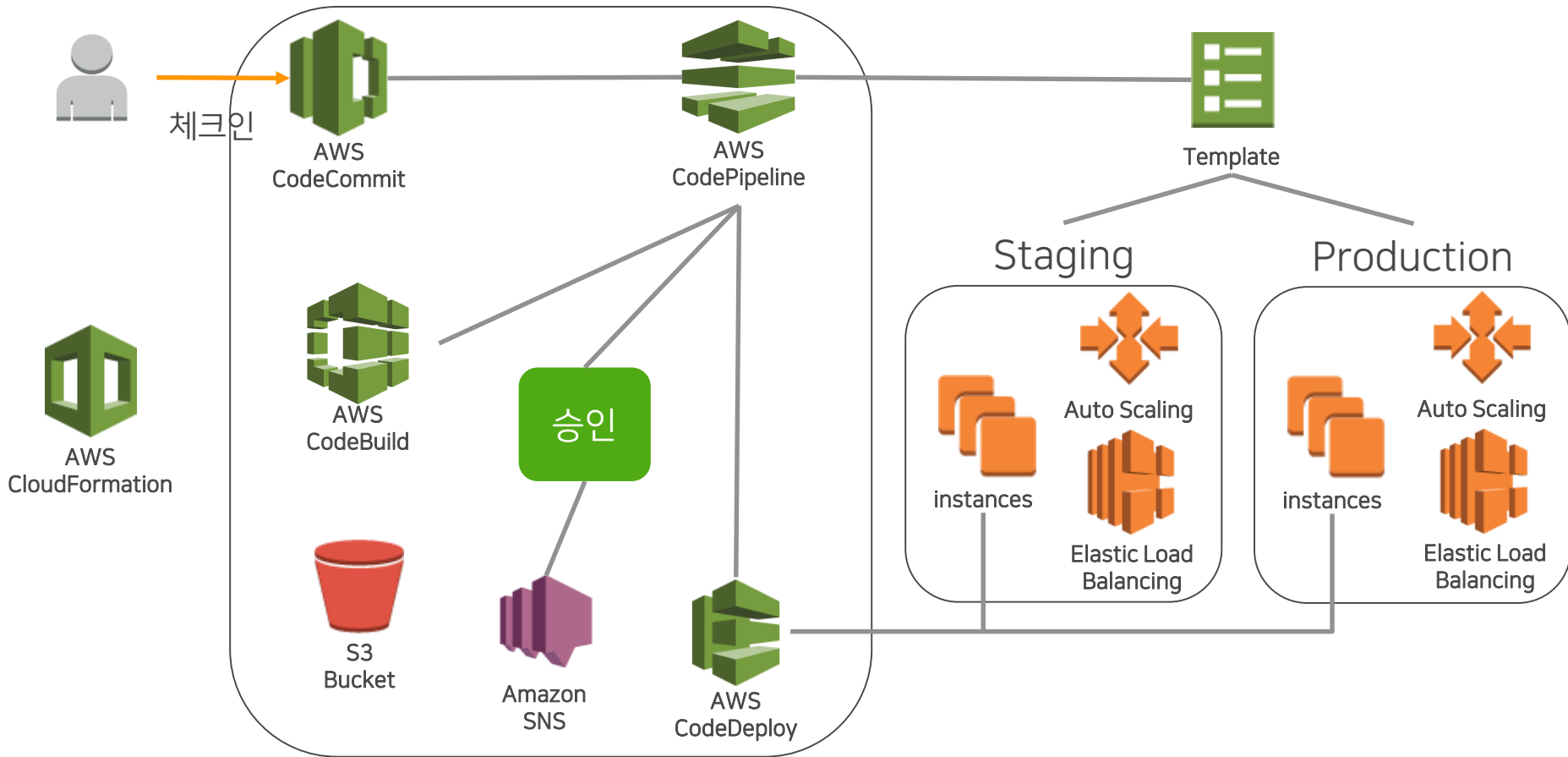
배포 자동화



배포 자동화



배포 자동화



INDEX

01 DevOps on AWS

02 AWS의 다양한 개발 도구

03 다양한 배포방법

Rolling Deployment

- 애플리케이션의 새 버전을 한 번에 하나의 배치로 배포
- 기존 버전을 실행하는 인스턴스는 점차 제거
- 트래픽은 구 버전에서 새 버전으로 점진적으로 옮겨가게 됨

Deployment—Rolling

Scenario

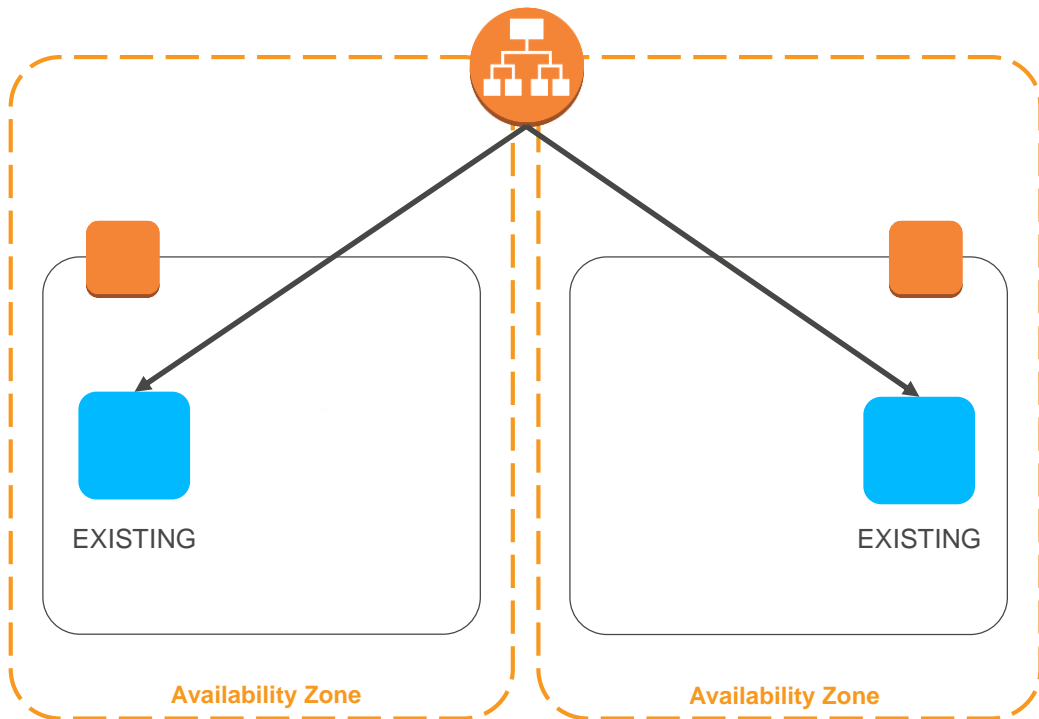
새로운 버전의 애플리케이션을
다음의 구성에 따라 배포합니다.

Desired Count = **2**

Minimum Healthy Percent = **50%**

Maximum Percent = **100%**

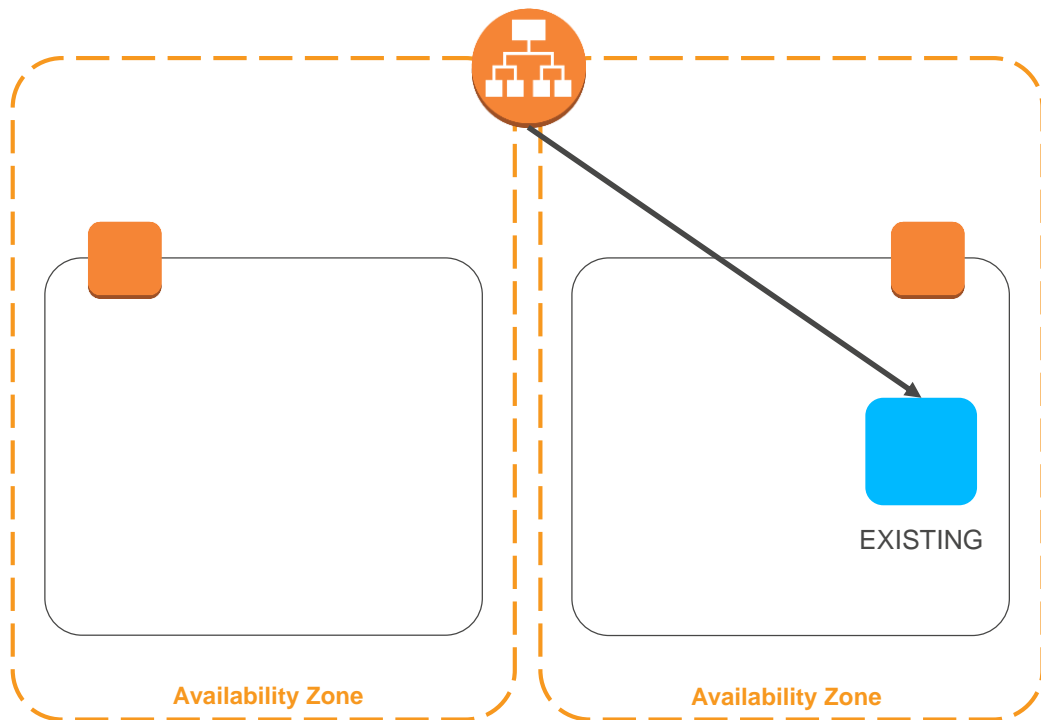
이러한 설정은 서비스가 원하는
크기를 초과하지 않도록 하지만
배포 중 인스턴스의 수는 절반으로
줄어들게 됩니다.



Deployment—Rolling

우선, 기존의 인스턴스
한대가 종료되어 서비스의
비율이 50% 가 됩니다.

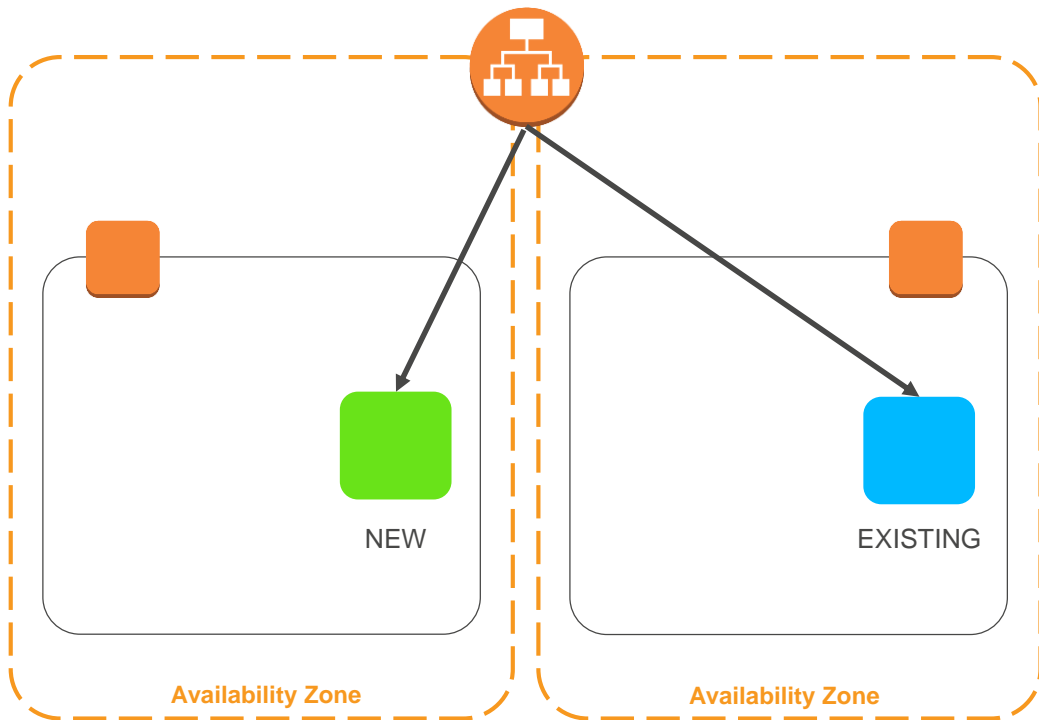
Desired Count = 2
Minimum Healthy Percent = **50%**
Maximum Percent = **100%**



Deployment—Rolling

새로운 버전의 애플리케이션
인스턴스가 배포된 뒤
서비스는 100% 정상 상태가
됩니다.

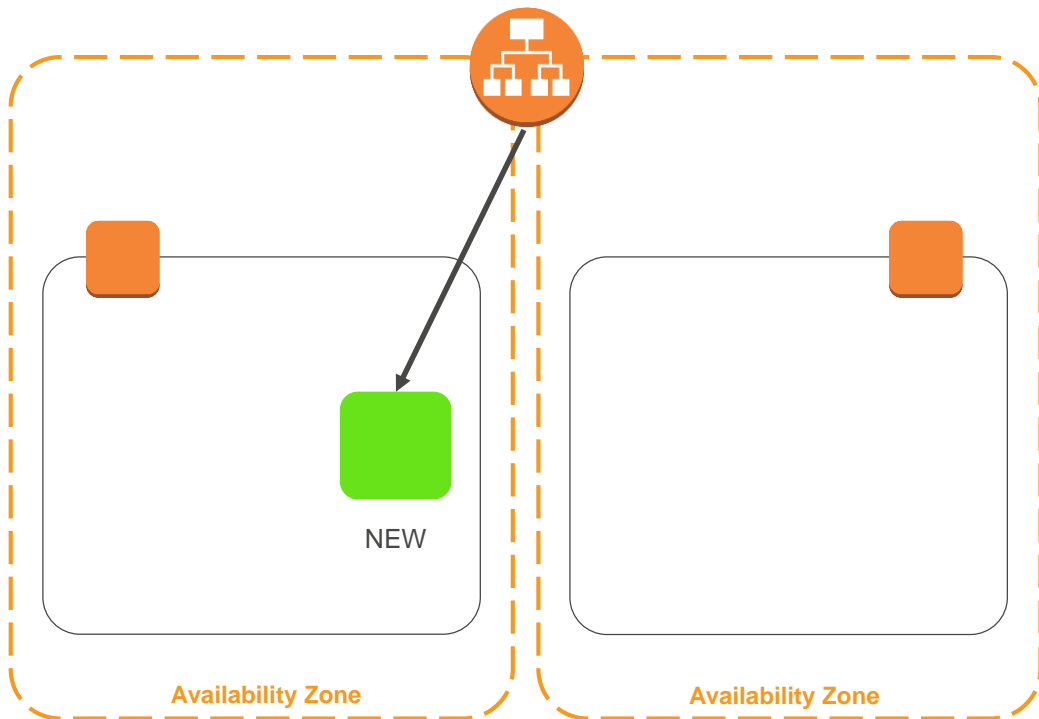
Desired Count = 2
Minimum Healthy Percent = **50%**
Maximum Percent = **100%**



Deployment—Rolling

ELB가 새로운 인스턴스가 정상인 것을 확인한 뒤에 남아 있는 구 버전의 인스턴스를 삭제합니다.

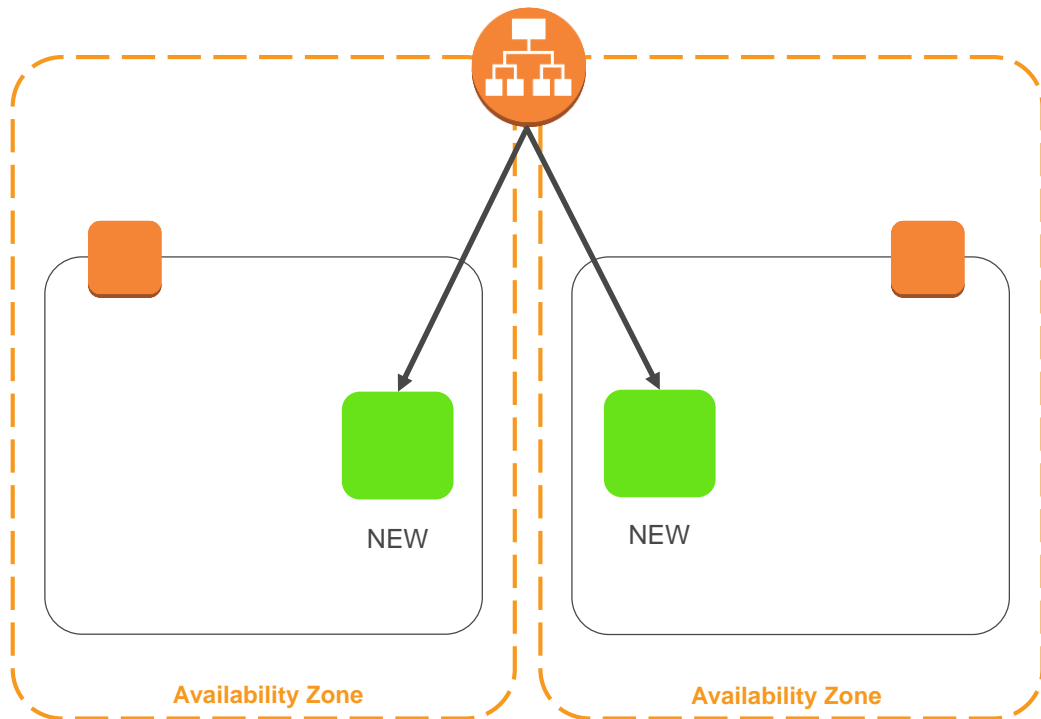
Desired Count = 2
Minimum Healthy Percent = **50%**
Maximum Percent = **100%**



Deployment—Rolling

새로운 버전의 인스턴스가
ELB 에 할당되고 배포가 완료
됩니다. 100% 정상 상태가
됩니다.

Desired Count = 2
Minimum Healthy Percent = **50%**
Maximum Percent = **100%**



Blue/Green Deployments

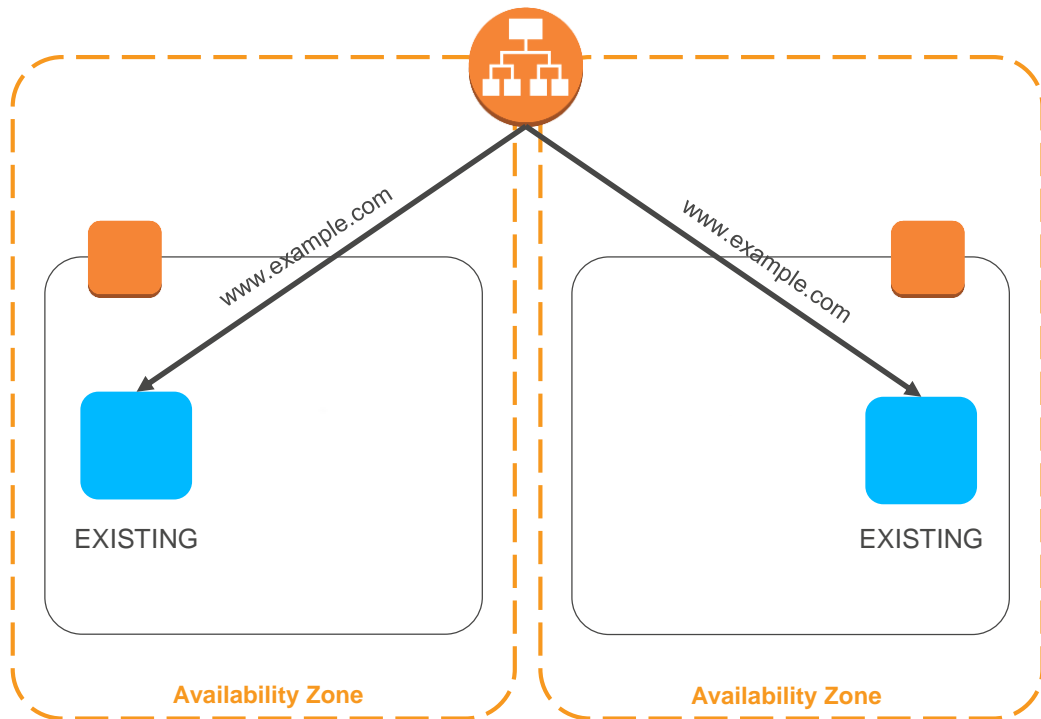
- 구버전과 새버전을 동시에 나란히 구성
- 하나의 버전만 프로덕션 트래픽을 처리
- 빠른 롤백 가능
- 운영환경에 영향을 주지 않고 새 버전 테스트 가능
 - Ex) 새 버전은 다른 포트에서 수신하거나 내부 리소스로만 액세스 가능

Deployment – Blue/Green – Target Group Swap

Scenario

호스트 기반 라우팅을 사용하여 동일한 ALB에 다른 버전의 애플리케이션이 서로 다른 타겟 그룹으로 나란히 구성됩니다.

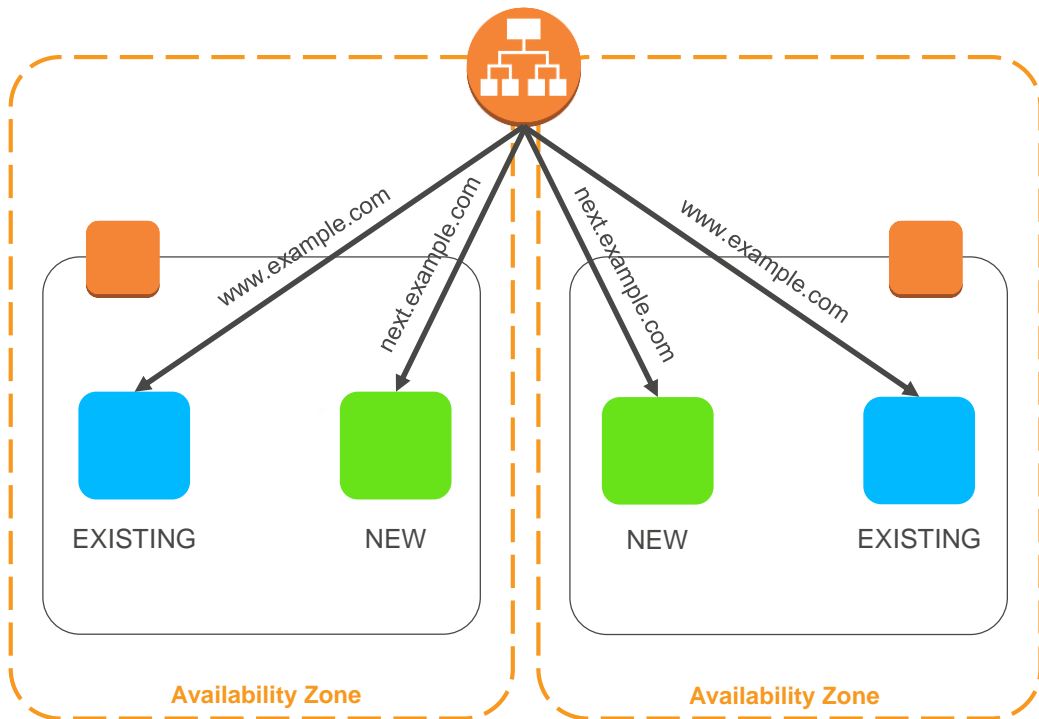
배포는 두 버전의 리스너 규칙이 Swap되며 완료됩니다.



Deployment – Blue/Green – Target Group Swap

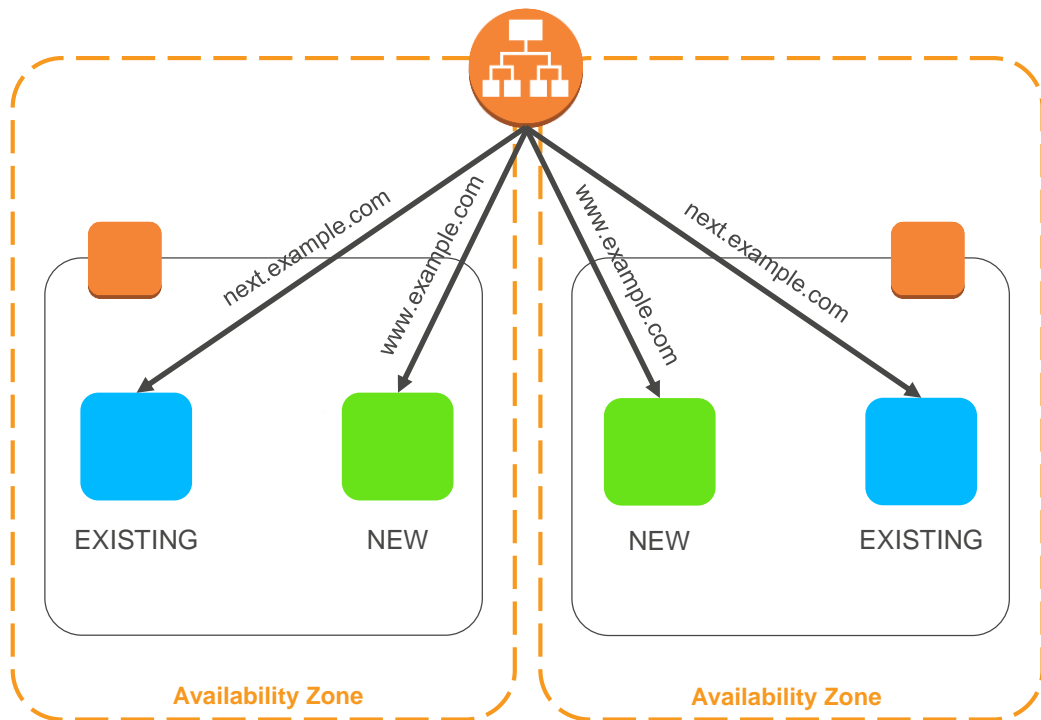
새로운 서비스가 동일한 ALB의
새로운 타겟 그룹으로
등록됩니다.

호스트 기반 라우팅을 사용하여
www.example.com에 대한
요청은 Blue 서비스(기존)로
전달되고, next.example.com에
대한 요청은 Green
서비스(신규)로 연결됩니다.



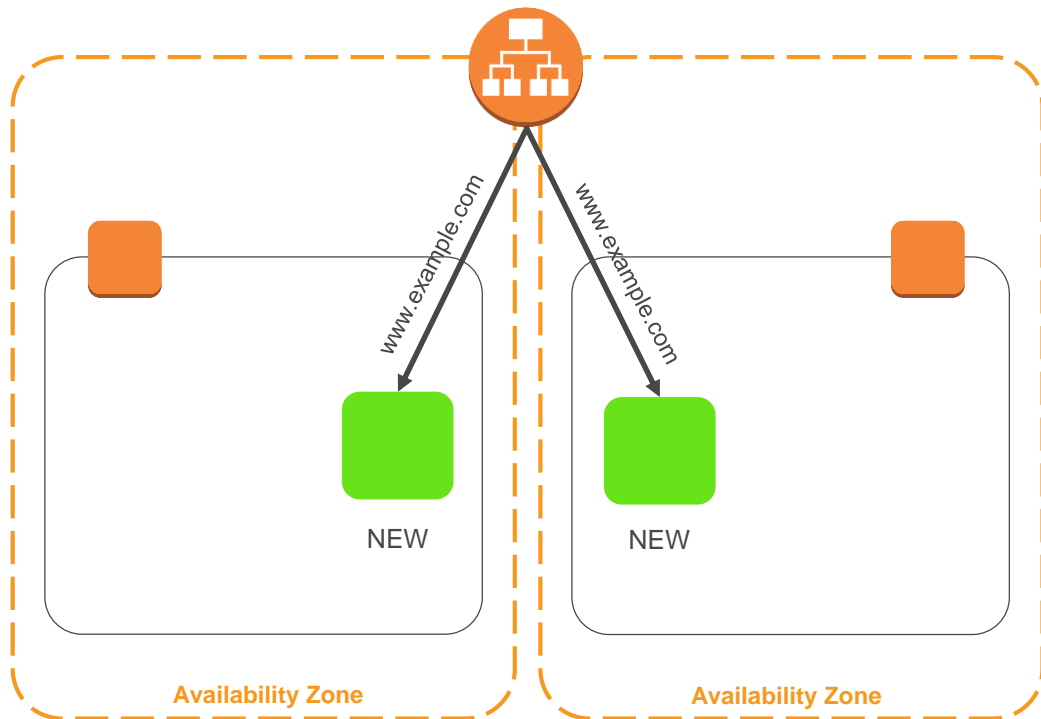
Deployment – Blue/Green – Target Group Swap

자동 또는 수동 테스트 후
ALB에서 리스너 규칙을
Swap하고 프로덕션 트래픽을
Green 서비스로 전달하여
배포를 완료하게 됩니다.



Deployment – Blue/Green – Target Group Swap

이전 서비스 및 해당 대상
그룹을 파괴할 수
있습니다.



Canary-Style Deployments

- 새 버전의 애플리케이션으로 프로덕션 트래픽의 일부를 분산
- A/B 테스트 사용 가능
- 라우딩은 임의적이거나 사용자 프로필을 기반으로 할 수 있음
- 결과에 따라 Canary 버전이 운영 환경을 대체하거나 혹은 중지할 수 있음

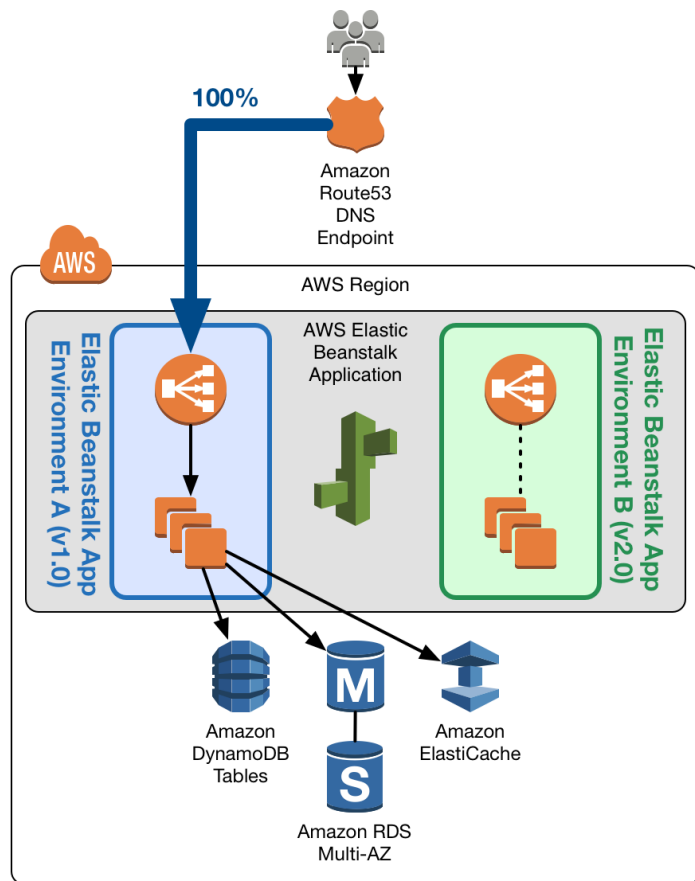
DNS를 이용한 방식

Scenario

새로운 버전의 애플리케이션을 Elastic Beanstalk 를 이용하여 배포합니다.

Route53의 가중치 기반 라우팅 방식으로 새로운 애플리케이션에 조금씩 트래픽을 이동합니다.

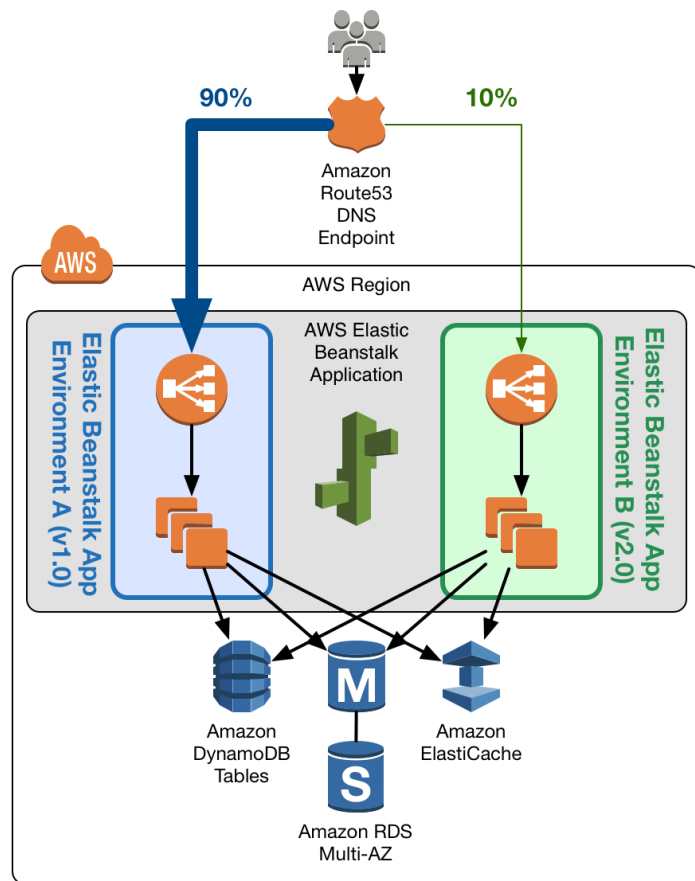
이후 테스트 결과에 따라 이전 버전으로의 롤백도 수월하게 할 수 있습니다.



DNS를 이용한 방식

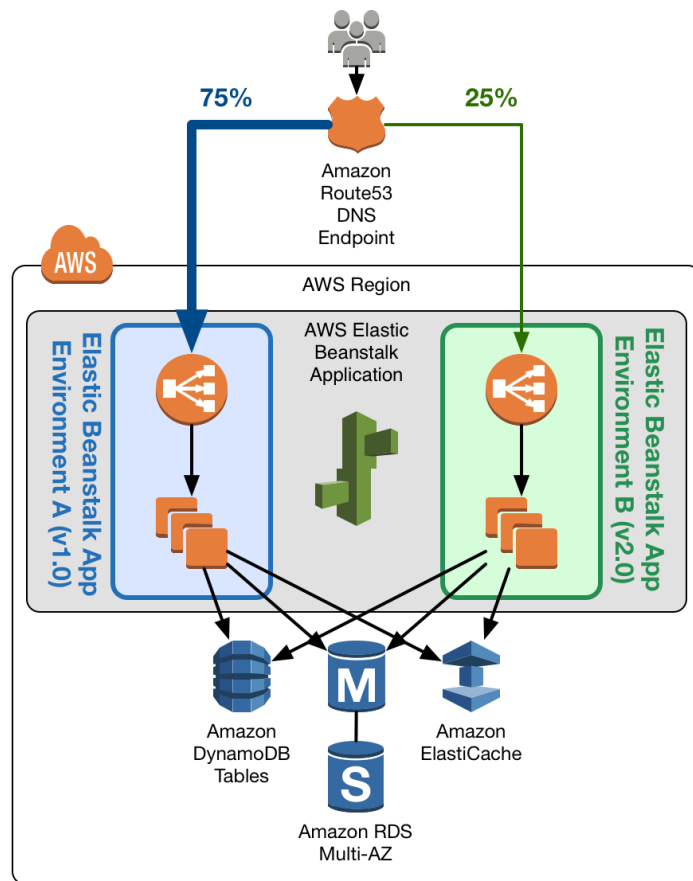
우선 조금의 트래픽을 이동한 뒤
테스트를 진행합니다.

내부 테스트를 진행할 수도 있고
A/B 테스트 방식으로 사용자
반응을 볼 수도 있습니다.



DNS를 이용한 방식

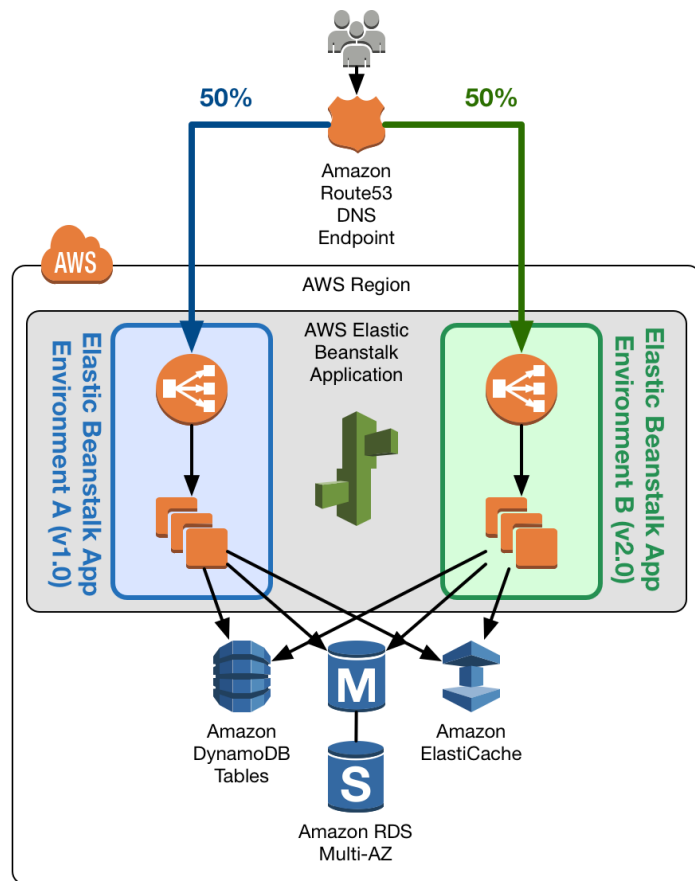
10%의 트래픽을 라우팅한 뒤
문제가 발견되지 않는다면
점진적으로 트래픽을 새로운
버전으로 옮겨갑니다.



DNS를 이용한 방식

10%의 트래픽을 라우팅한 뒤
문제가 발견되지 않는다면
점진적으로 트래픽을 새로운
버전으로 옮겨갑니다.

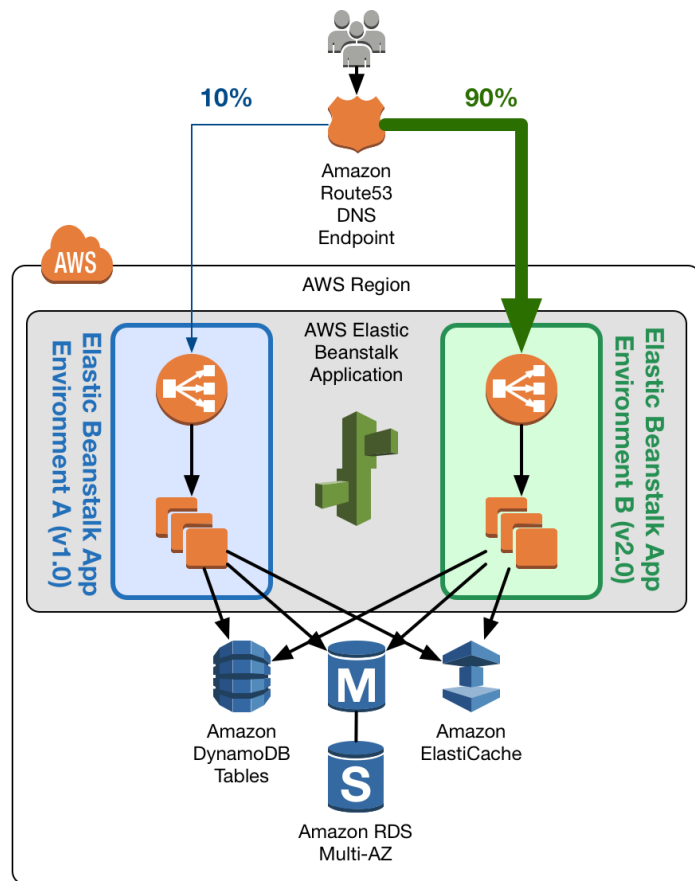
모니터링을 통해 결과가 좋지
않다면 이전 버전으로 롤백할 수
있습니다.



DNS를 이용한 방식

트래픽을 새로운 버전으로
라우팅한 뒤 문제가 발견되지
않는다면 점진적으로 트래픽을
새로운 버전으로 옮겨갑니다.

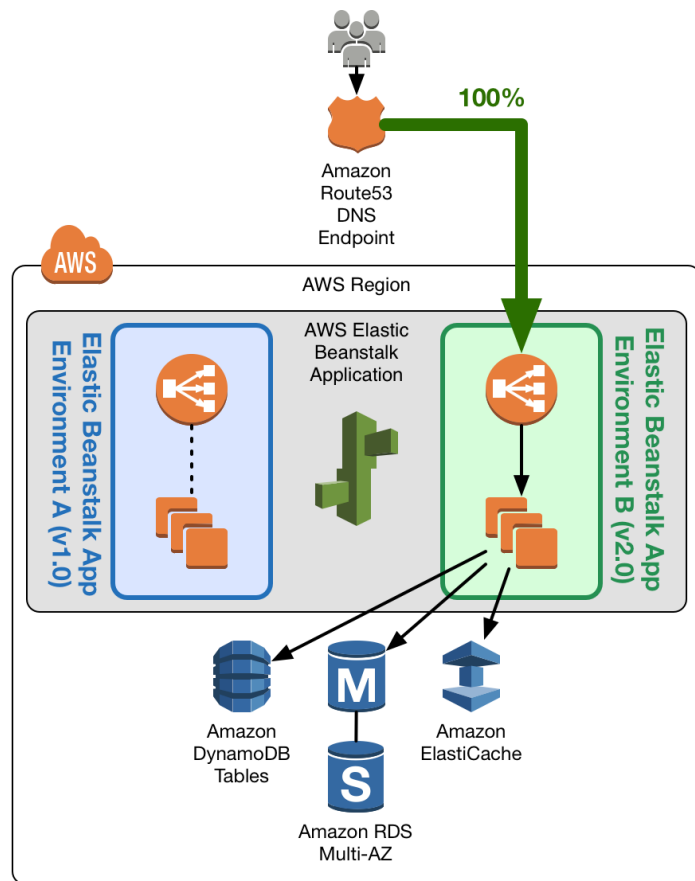
모니터링을 통해 결과가 좋지
않다면 이전 버전으로 롤백할 수
있습니다.



DNS를 이용한 방식

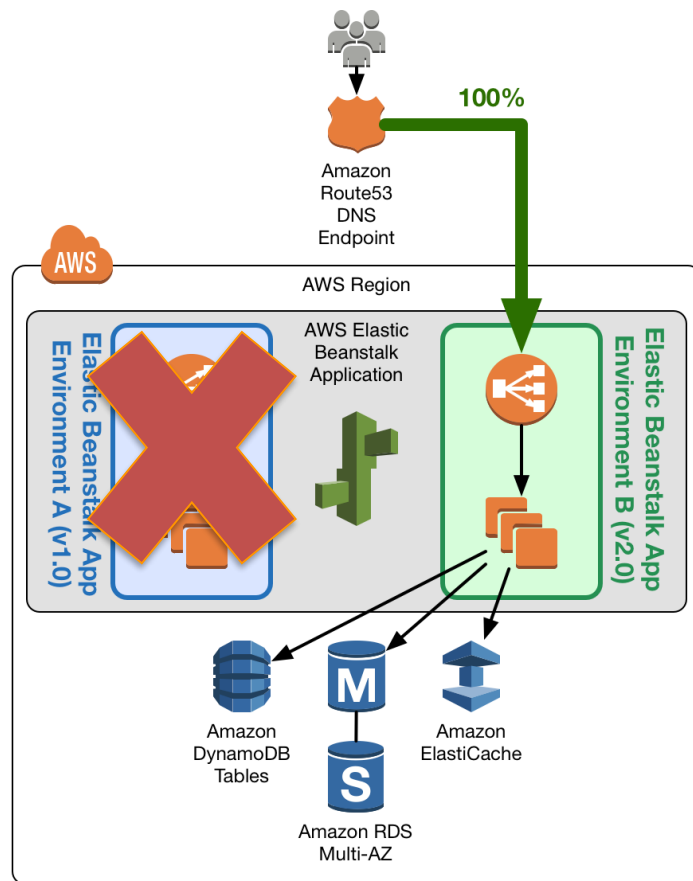
새로운 버전으로 트래픽을 100% 이동합니다.

구 버전의 애플리케이션 환경은 여전히 남아있기 때문에 여전히 롤백이 가능합니다.

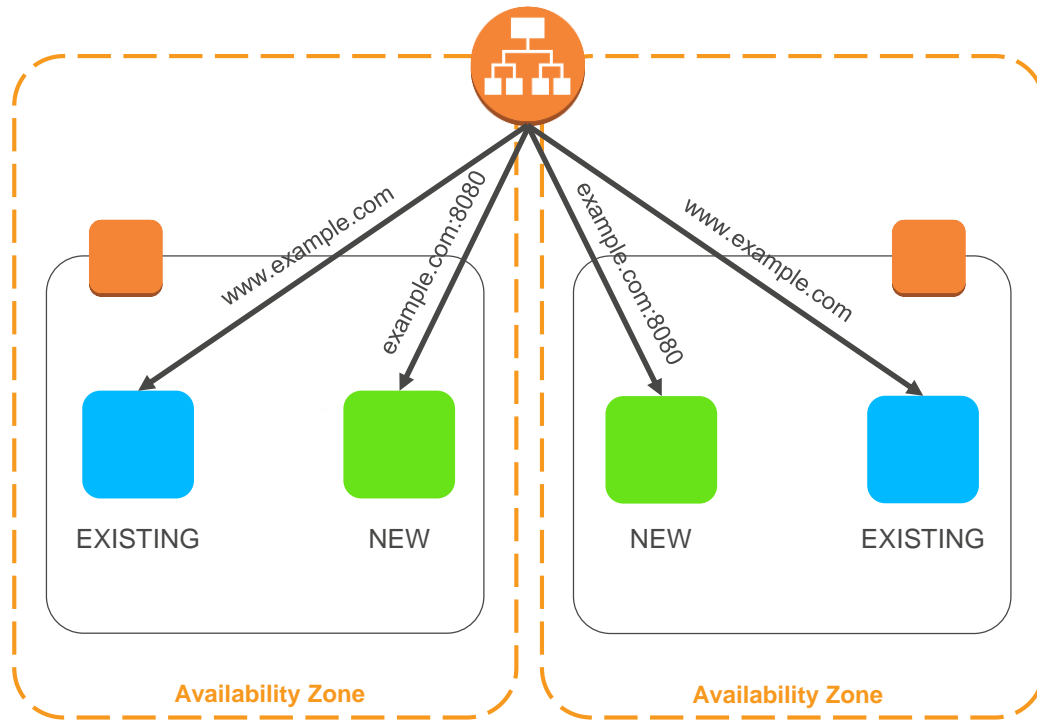


DNS를 이용한 방식

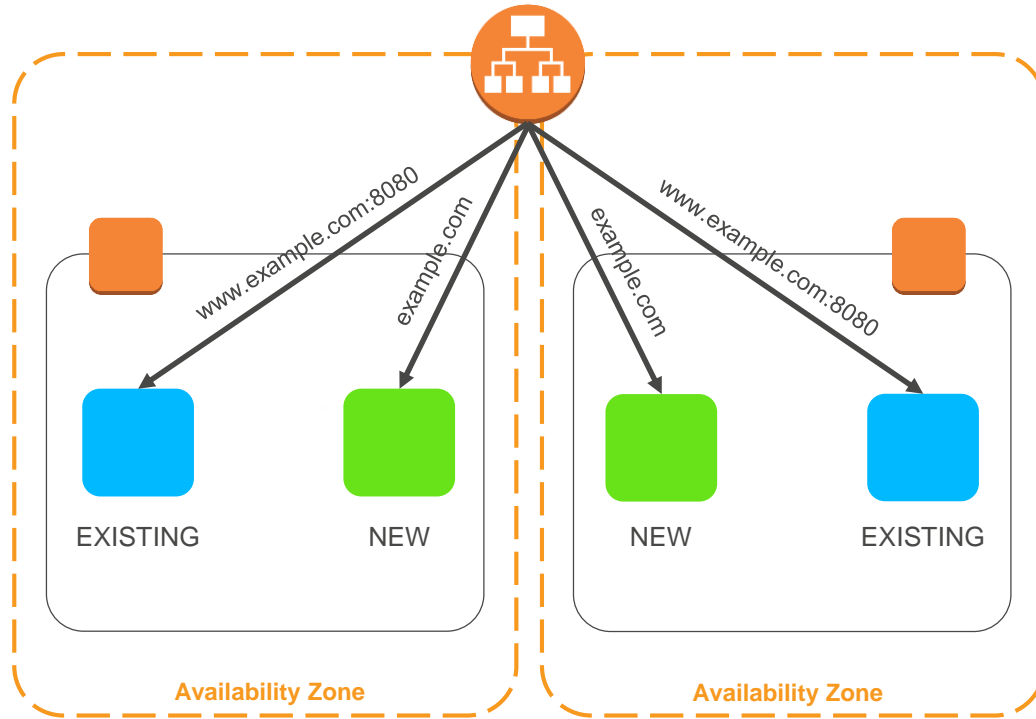
이 후 이전 환경을 제거하면 모든
배포 프로세스가 완료됩니다.



Demo Architecture



Demo Architecture



Events

Tags

Reports

Limits

INSTANCES

Instances

Launch Templates

Spot Requests

Reserved Instances

Dedicated Hosts

Scheduled Instances

IMAGES

AMIs

Bundle Tasks

ELASTIC BLOCK STORE

Volumes

Snapshots

NETWORK & SECURITY

Security Groups

Elastic IPs

Placement Groups

Key Pairs

Network Interfaces

LOAD BALANCING

Load Balancers

Target Groups

AUTO SCALING

Launch Configurations

Auto Scaling Groups

SYSTEMS MANAGER SERVICES

Run Command

State Manager

Configuration Compliance

Automations

Patch Compliance

Patch Baselines

SYSTEMS MANAGER SHARED RESOURCES

Managed Instances

Activations

Documents

Maintenance Windows

Create Load Balancer Actions

Filter by tags and attributes or search by keyword

	Name	DNS name	State	VPC ID	Availability Zones	Type	Created At	Monitoring
<input checked="" type="checkbox"/>	ECSDeployLB	ECSDeployLB-2061670082....	active	vpc-3daba345	us-east-1a, us-east-1b	application	February 2, 2018 at 12:22:0...	<input checked="" type="checkbox"/>
<input type="checkbox"/>	testAPI	testAPI-a9a2c99e72f4e56a....	active	vpc-4c734e2a	us-east-1c, us-east-1b, ...	network	January 8, 2018 at 4:42:54 P...	<input type="checkbox"/>

Load balancer: ECSDeployLB

DescriptionListenersMonitoringTags

Basic Configuration

Name:

ECSDeployLB

ARN:

arn:aws:elasticloadbalancing:us-east-1:684778767920:loadbalancer/app/ECSDeployLB/3354f83462352c01

DNS name:

ECSDeployLB-2061670082.us-east-1.elb.amazonaws.com (A Record)

Scheme:

internet-facing

Type:

application

Availability Zones:

subnet-73aacb2e - us-east-1a, subnet-ec451f88 - us-east-1b

Edit availability zones

Creation time:

February 2, 2018 at 12:22:07 AM UTC+9

Hosted zone:

Z35SXDOTRQ7X7K

State:

active

VPC:

vpc-3daba345

IP address type:

ipv4

AWS WAF Web ACL:

Security

Security groups:

sg-ad8c8ad9, ECSDeployLB-SG

ECSDeploy LoadBalancer SG

Canary 스타일 배포

- 새 버전의 애플리케이션으로 프로덕션 트래픽의 일부를 분산
- A/B 테스트 사용 가능
- 라우팅은 임의적이거나 사용자 프로필을 기반으로 할 수 있음
- 결과에 따라 Canary 버전이 운영 환경을 대체하거나 혹은 중지할 수 있음

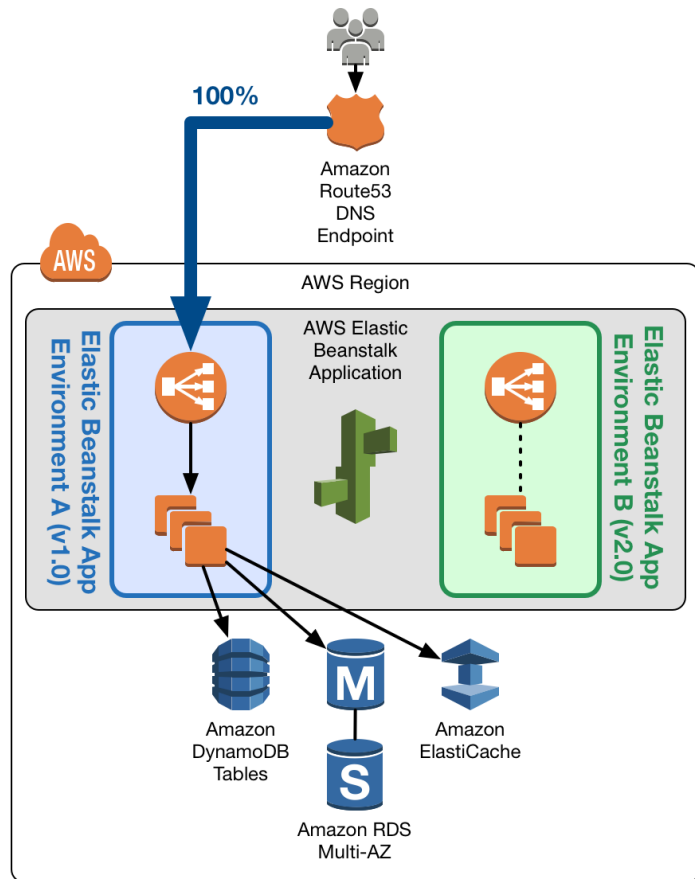
Canary 스타일 배포

Scenario

새로운 버전의 애플리케이션을 Elastic Beanstalk 를 이용하여 배포합니다.

Route53의 가중치 기반 라우팅 방식으로 새로운 애플리케이션에 조금씩 트래픽을 이동합니다.

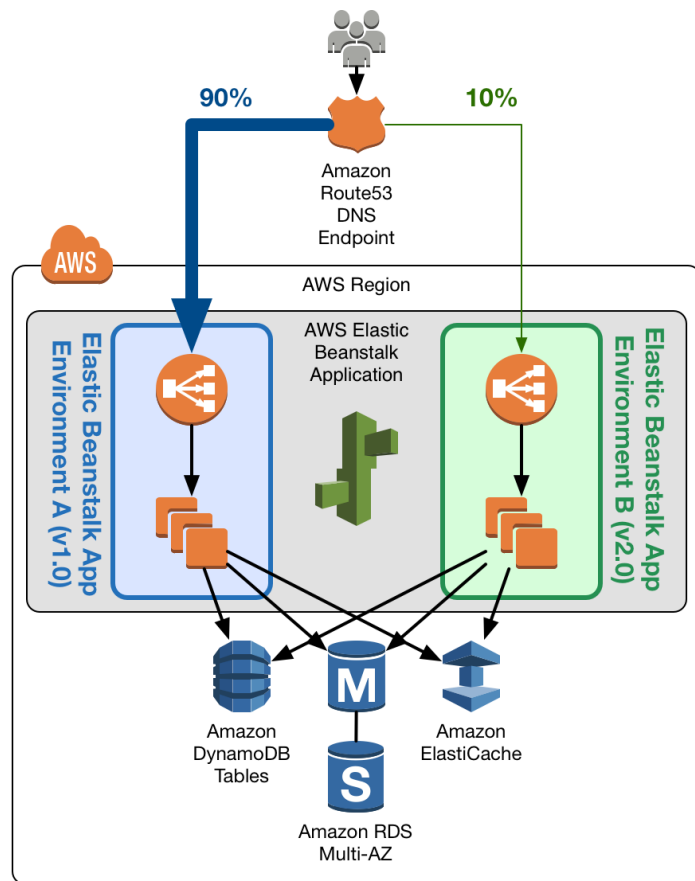
이후 테스트 결과에 따라 이전 버전으로의 롤백도 수월하게 할 수 있습니다.



Canary 스타일 배포

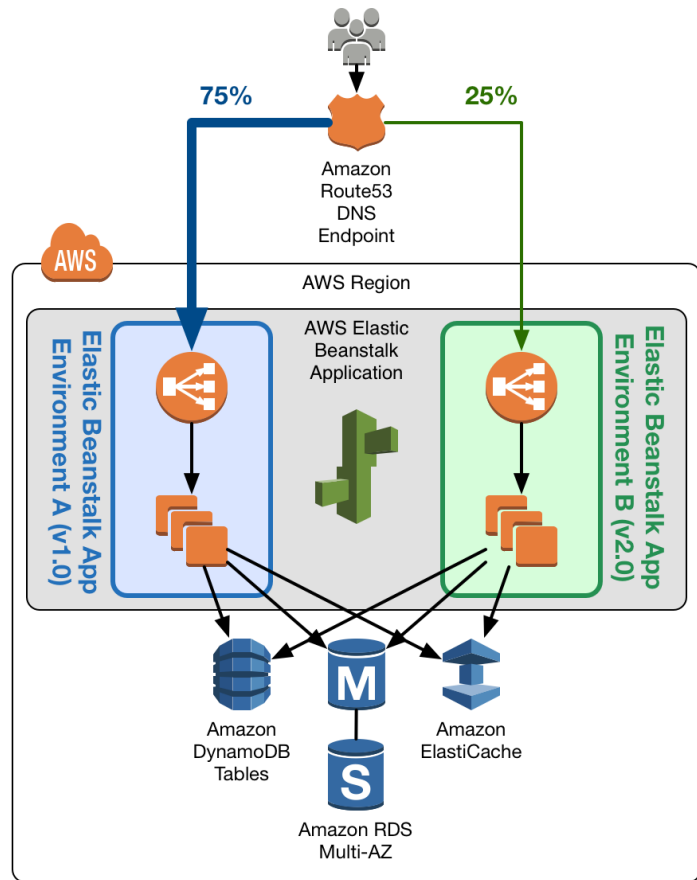
우선 조금의 트래픽을 이동한 뒤
테스트를 진행합니다.

내부 테스트를 진행할 수도 있고
A/B 테스트 방식으로 사용자
반응을 볼 수도 있습니다.



Canary 스타일 배포

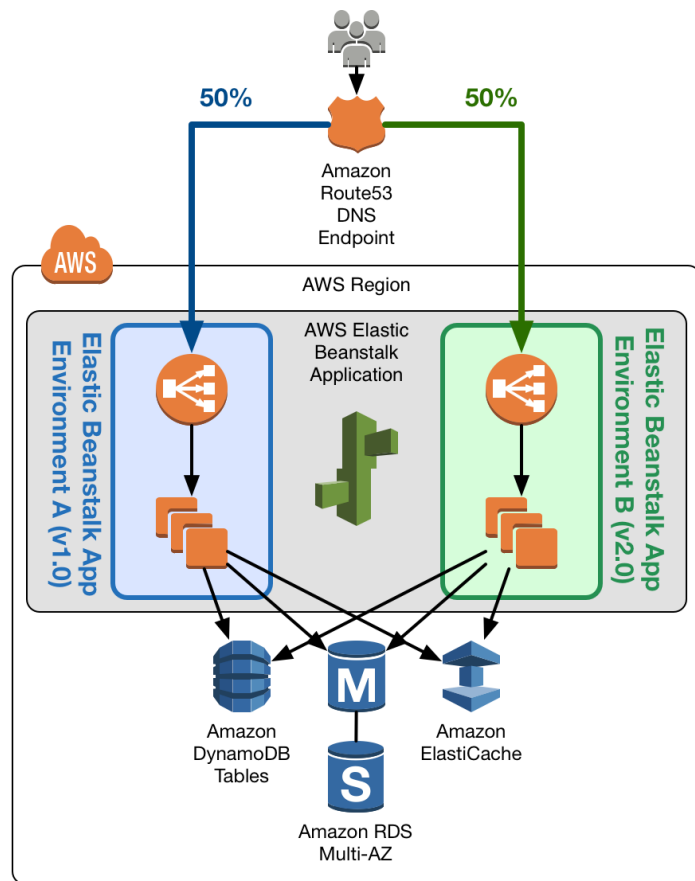
10%의 트래픽을 라우팅한 뒤
문제가 발견되지 않는다면
점진적으로 트래픽을 새로운
버전으로 옮겨갑니다.



Canary 스타일 배포

10%의 트래픽을 라우팅한 뒤
문제가 발견되지 않는다면
점진적으로 트래픽을 새로운
버전으로 옮겨갑니다.

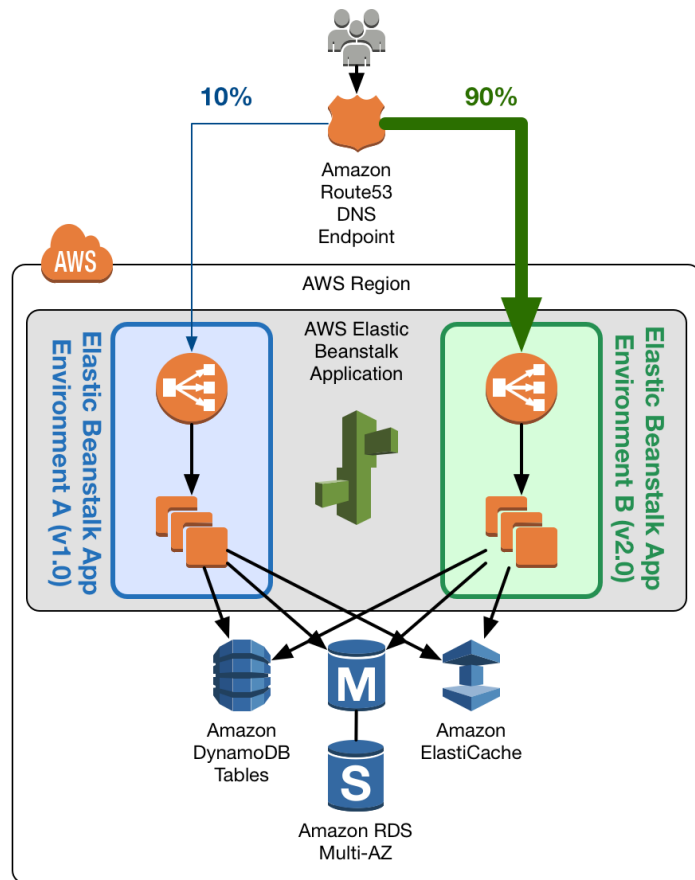
모니터링을 통해 결과가 좋지
않다면 이전 버전으로 롤백할 수
있습니다.



Canary 스타일 배포

트래픽을 새로운 버전으로
라우팅한 뒤 문제가 발견되지
않는다면 점진적으로 트래픽을
새로운 버전으로 옮겨갑니다.

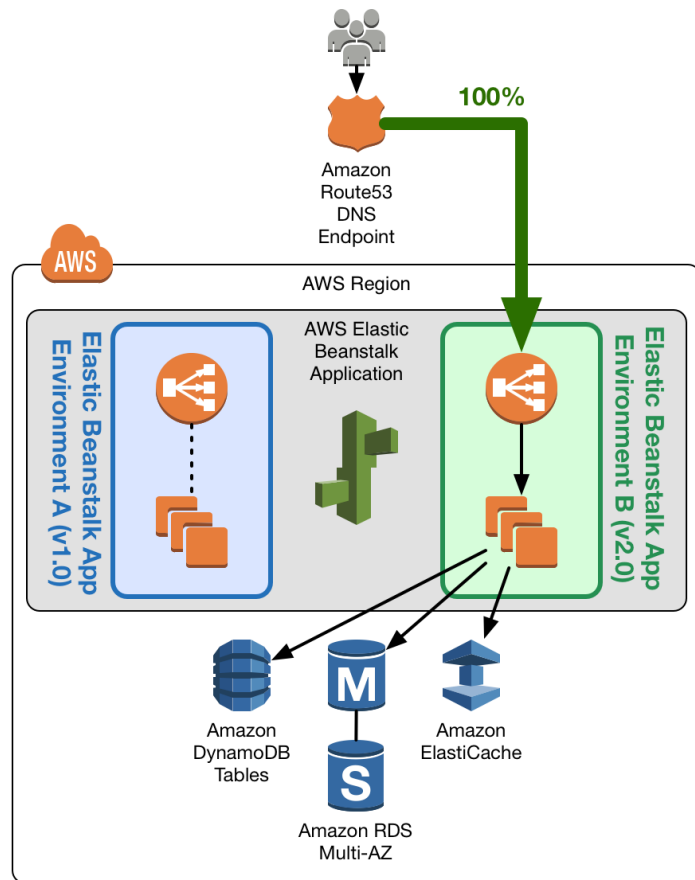
모니터링을 통해 결과가 좋지
않다면 이전 버전으로 롤백할 수
있습니다.



Canary 스타일 배포

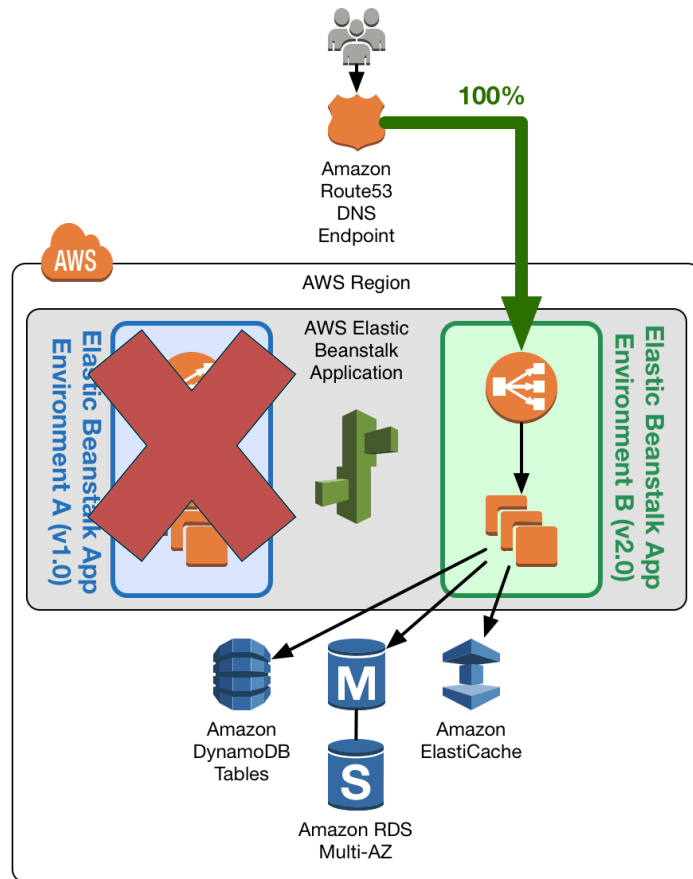
새로운 버전으로 트래픽을 100% 이동합니다.

구 버전의 애플리케이션 환경은 여전히 남아있기 때문에 여전히 롤백이 가능합니다.



Canary 스타일 배포

이 후 이전 환경을 제거하면 모든 배포 프로세스가 완료됩니다.



INDEX

01 DevOps on AWS

02 AWS의 다양한 개발 도구

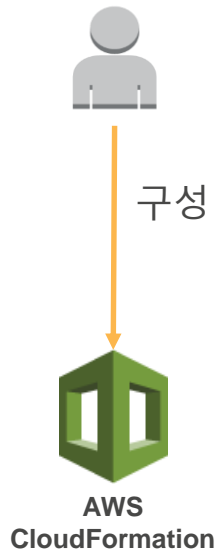
03 AWS 개발 도구 활용하기

04 모범 사례

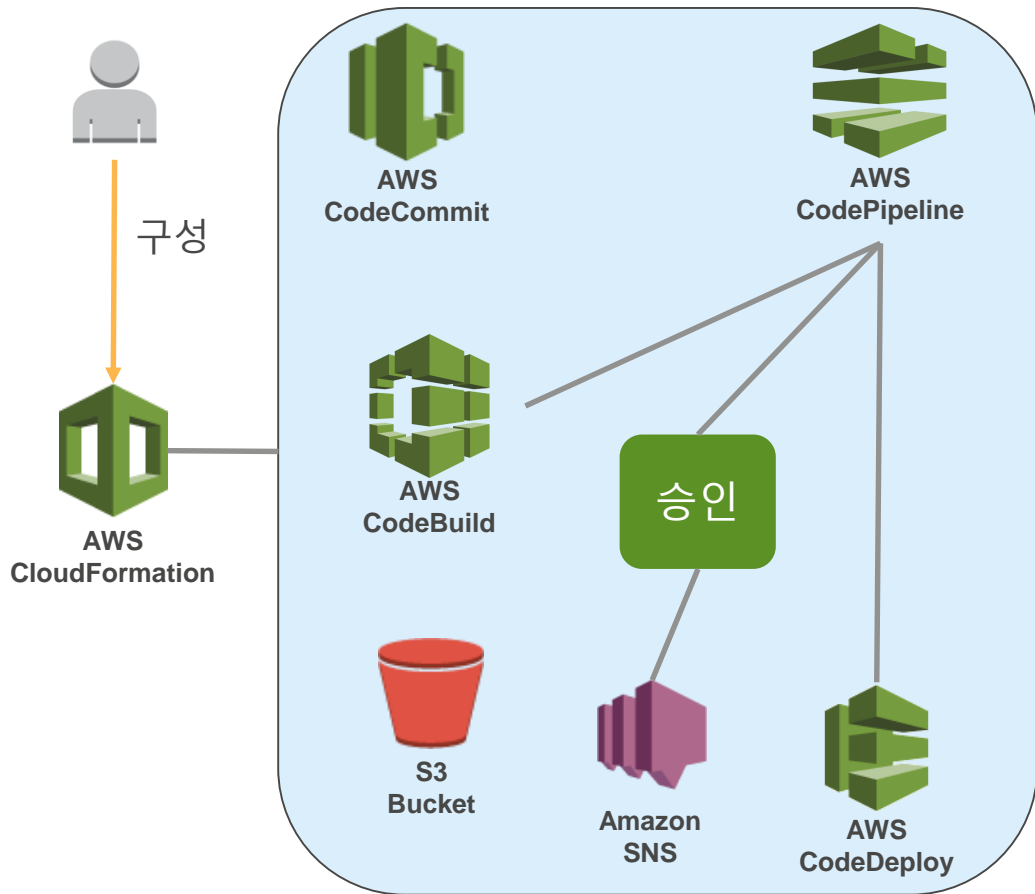
데모 아키텍처



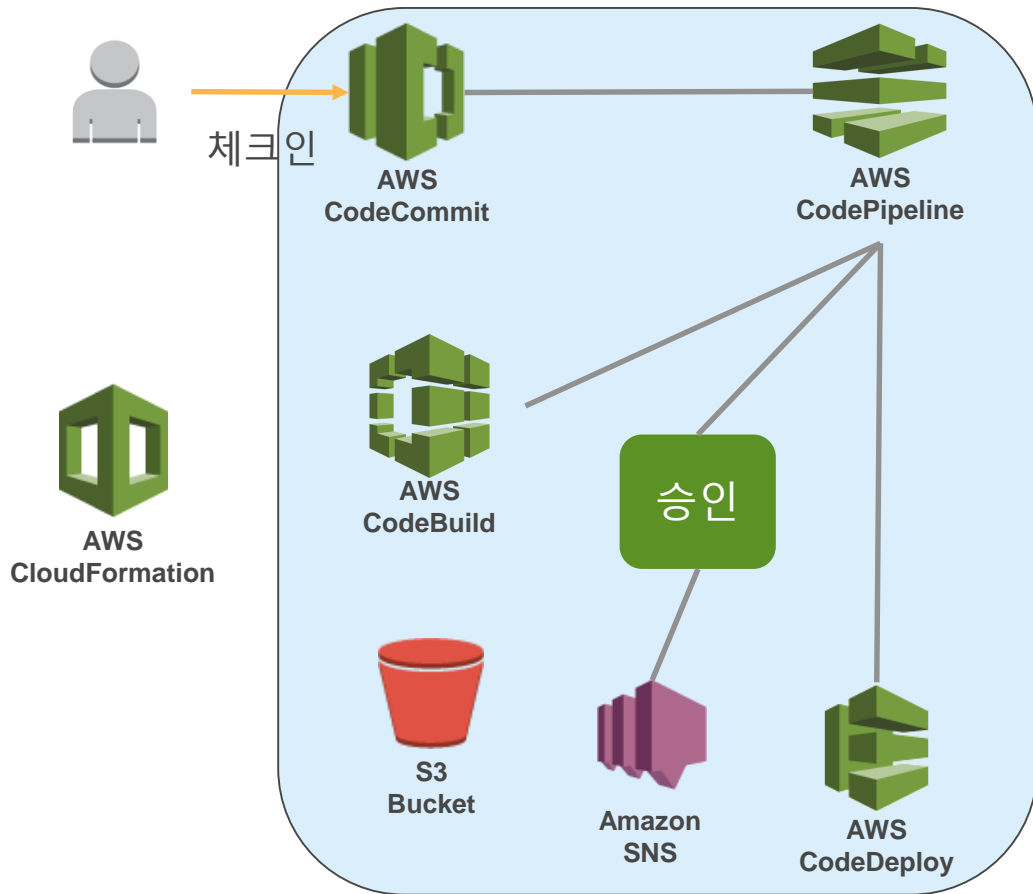
데모 아키텍처



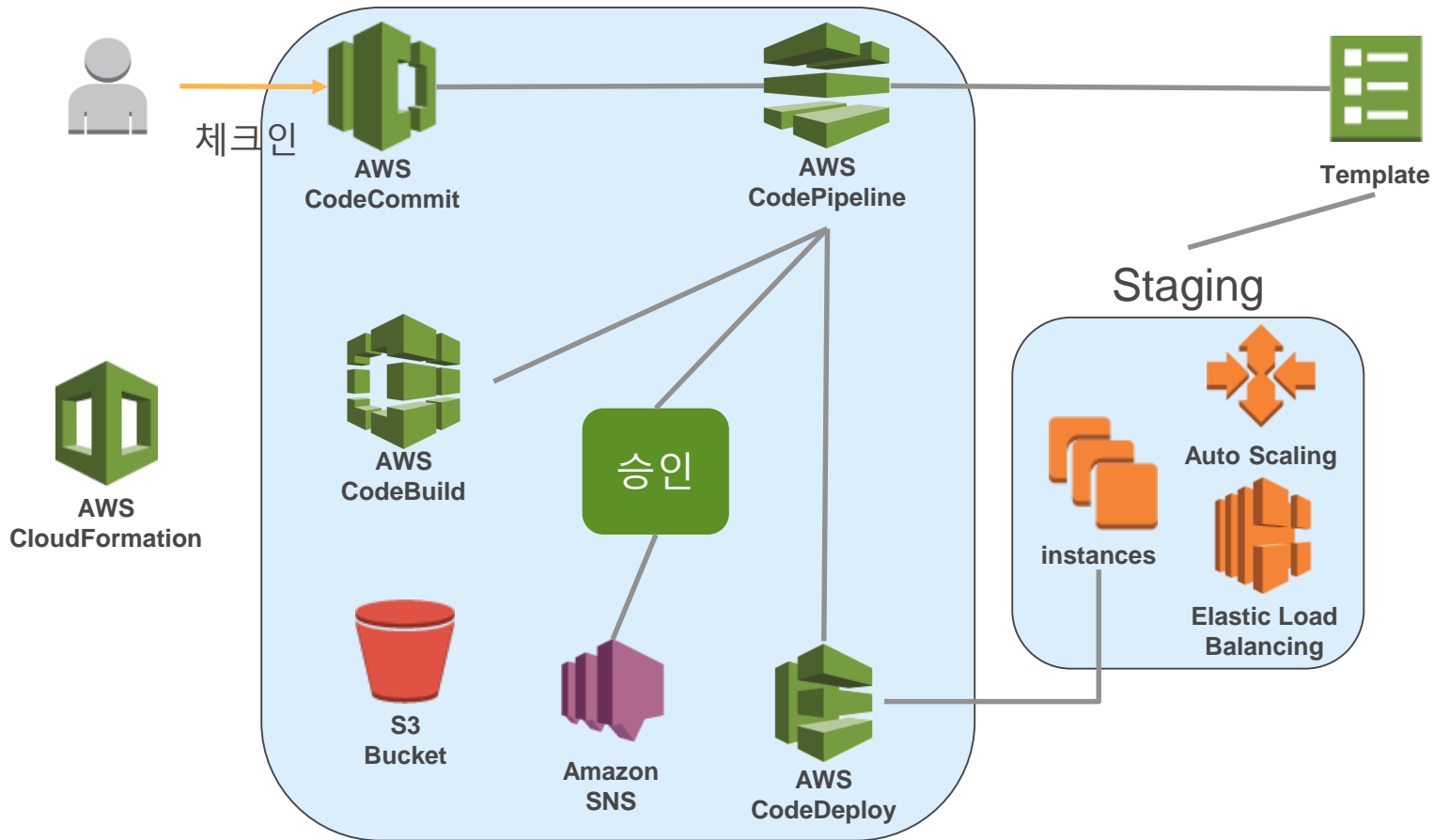
데모 아키텍처



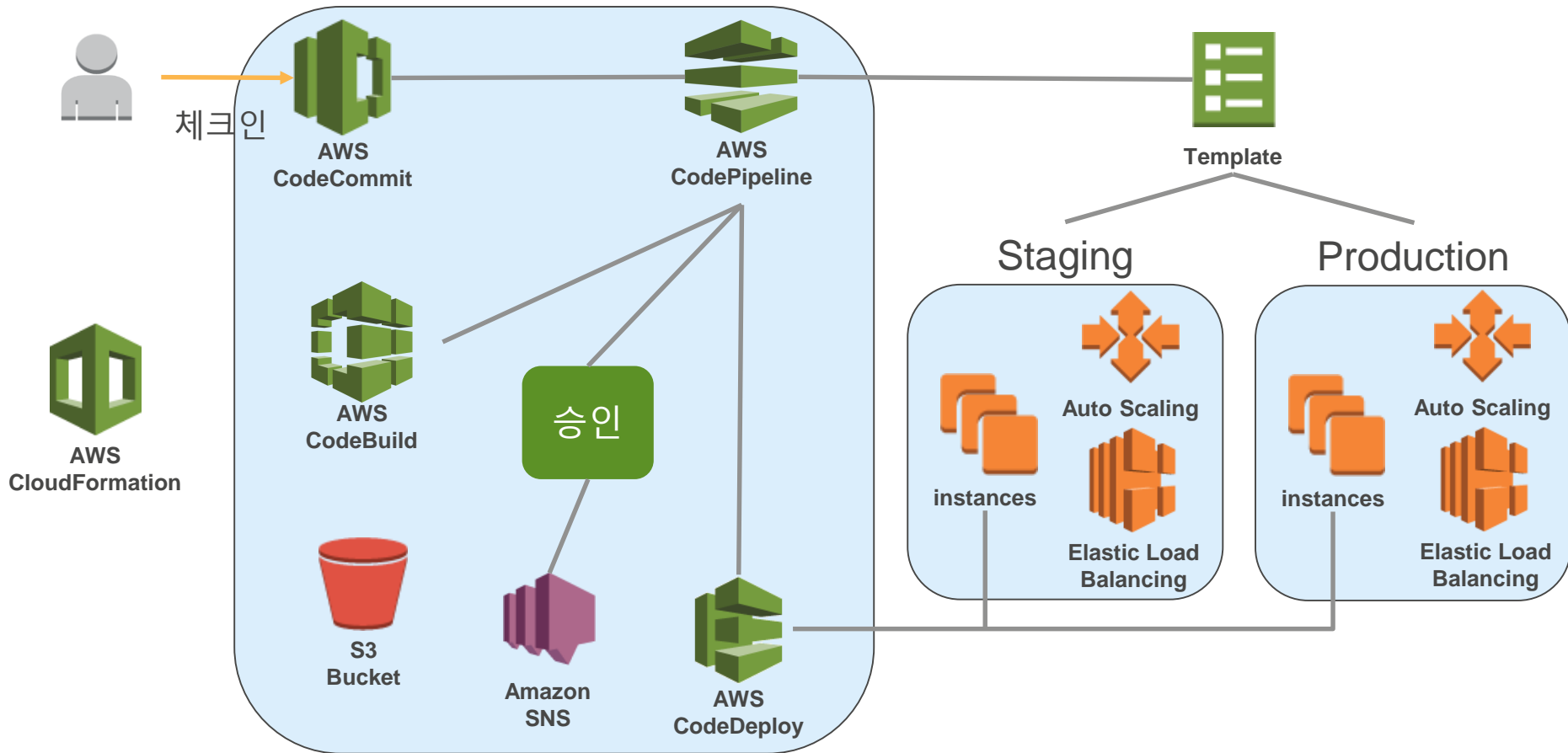
데모 아키텍처



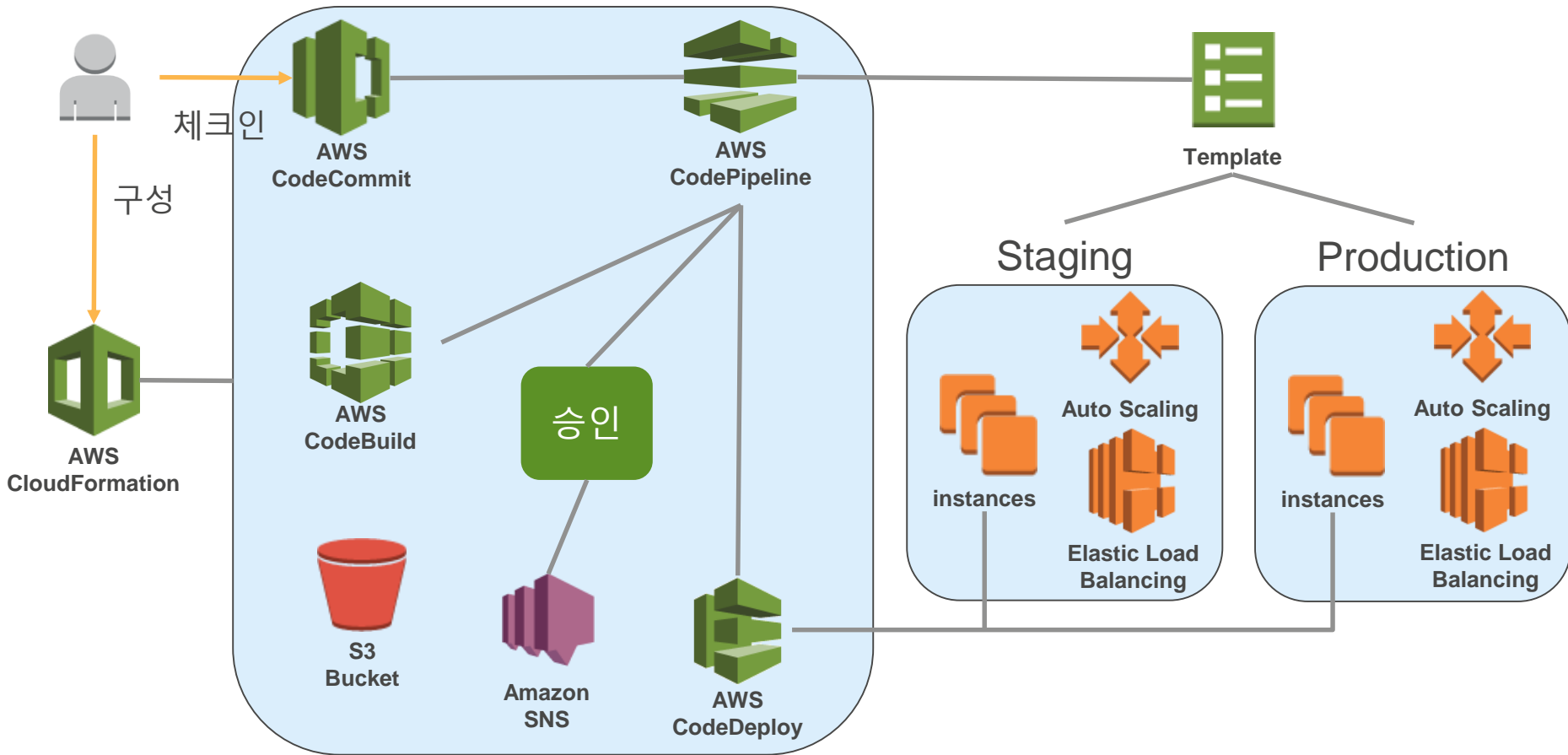
데모 아키텍처



데모 아키텍처



데모 아키텍처



Code* 관련 팁

- 모든 Code* 제품은 AWS CloudFormation 으로 프로비저닝하고 관리 가능
 - Code* 리소스를 프로비저닝하는 CloudFormation 템플릿을 CodeCommit 에 저장, CodePipeline 을 통해 업데이트 가능
- IAM 과의 통합하여 누가 코드를 커밋할 수 있고, 직접 승인이 필요할 때 누가 승인할 수 있으며, 특정 배포 그룹으로만 배포하는 등의 권한 관리가 가능
- AWS Lambda 와 통합하여 거의 모든것이 가능
 - CodeCommit 저장소 트리거 기능
 - CodeDeploy 이벤트 알림
 - CodePipeline 직접 Lambda 를 호출

INDEX

01 DevOps on AWS

02 AWS의 다양한 개발 도구

03 AWS 개발 도구 활용하기

04 모범 사례



Amazon 개발자들의 모범 사례

- CI/CD 는 필수!
 - 자주 커밋
 - 매 커밋마다 빌드
 - 동작하는 환경에 배포하여 추가 테스트 수행
- 코드화 가능한 모든 것은 저장소에 저장 (애플리케이션, 인프라, 문서)
 - 저장소에 없다면 프로덕션에 배포되지 않음
- 지속적 전달(Continuous delivery)로 시작하여 높은 수준의 테스트 우수성에 대해 확신이 선다면
지속적 배포 (Continuous deployment) 로 진행





Amazon 개발자들의 모범 사례 (계속)

- 코드 리뷰는 “좋은” 코드를 위한 최고의 매커니즘 중 하나:
 - 이 코드는 깔끔해보이며 다른 사람이 이해할 수 있는지?
 - 요구사항에 대한 기대치를 충족시키는 디자인인지?
 - 같은 일을 더 잘/더 쉽게 할 수 있는 다른 방법이 있는지?
- 스타일 검사
 - 회사의 다른 누군가가 이 코드를 업데이트/수정/유지 관리 할 수 있는지?
- 자동 롤백은 실패 후 가장 빠른 복구 매커니즘일 수 있음
 - 롤백한 뒤 로그/그래프/기타 등을 활용하여 디버깅





Amazon 개발자들의 모범 사례 (계속)

- 철저한 대시보드
 - 지금 무슨 일이 일어나고 있는지?
 - 일정 기간 동안 "정상적인" 모습은 무엇이었는지?
 - 그래프가 이상하게 보이거나 경보가 발생하면 어떻게 해야 하는지?
 - 어떤 이벤트를 그래프의 이동과 관련지을 수 있는지?

