



AWS Theory

- Elastic Beanstalk

Sung-Dong Kim,
School of Computer Engineering,
Hansung University



What (1)

- Easy-to-use service for deploying and scaling web applications and services
 - Java, PHP, Node.js, Python, ...
 - on servers such as Apache, Ngnix, ...
- 사용자는 upload -> EB가 capacity provisioning, load balancing, auto scaling, health monitoring 등의 모든 배포를 자동으로 관리

What (2)

- Elastic beanstalk은 애플리케이션을 쉽게 배포할 수 있고 운영 및 관리를 지원하는 **AWS** 서비스
- Elastic beanstalk은 프로비저닝(provisioning)의 결정체
- application을 저장/실행하는 데 필요한 자원에 대한 비용만 지불하면 됨

장점 (1)

- fast and simple to begin
 - AWS management console, Git repository, IDE (Visual Studio, Eclipse, ...)
 - 수분 내에 application이 준비됨 - infrastructure나 resource configuration을 하지 않아도 됨

장점 (2)

- Developer productivity
 - application을 실행하는 platform을 최신으로 유지시켜 줌
- Complete resource control
 - 최적의 resource를 user가 선택할 수 있음

특징 (1)

- EB는 auto-scaling, load-balancing 환경을 제공
- EB dashboard
 - EC2 최대 개수 지정
 - instance 추가/삭제를 위한 triggering rule 추가
 - static file 캐싱을 위한 proxy server 선택
 - application health monitoring 활성화
 - alarm 생성

기초 (1)

- Application

- project의 folder: EB application 내에서 실행되지 않음
- 배포 (deployment)되고 환경 (environment) 안에서 실행됨

- Environment

- EB application 내에 만듦
- application의 서로 다른 실행 버전을 관리

Elastic Beanstalk Application

Environment 1

Environment ...

Environment n

helloeb

helloeb-production

Environment tier: Web Server

Running versions: app-160126_143317

Last modified: 2016-01-27 14:21:53 UTC+0800

URL: helloeb-production.ap-southeast-1.elasticb...

helloeb-staging

Environment tier: Web Server

Running versions: app-160126_134819

Last modified: 2016-01-26 13:52:43 UTC+0800

URL: helloeb-staging.ap-southeast-1.elasticbean...

기초 (2)

- Environment tier
 - Web Server tier: HTTP(S) request를 다룸
 - Worker tier: background process를 다룸
- Environment health
 - green
 - gray
 - yellow
 - red

기초 (3)

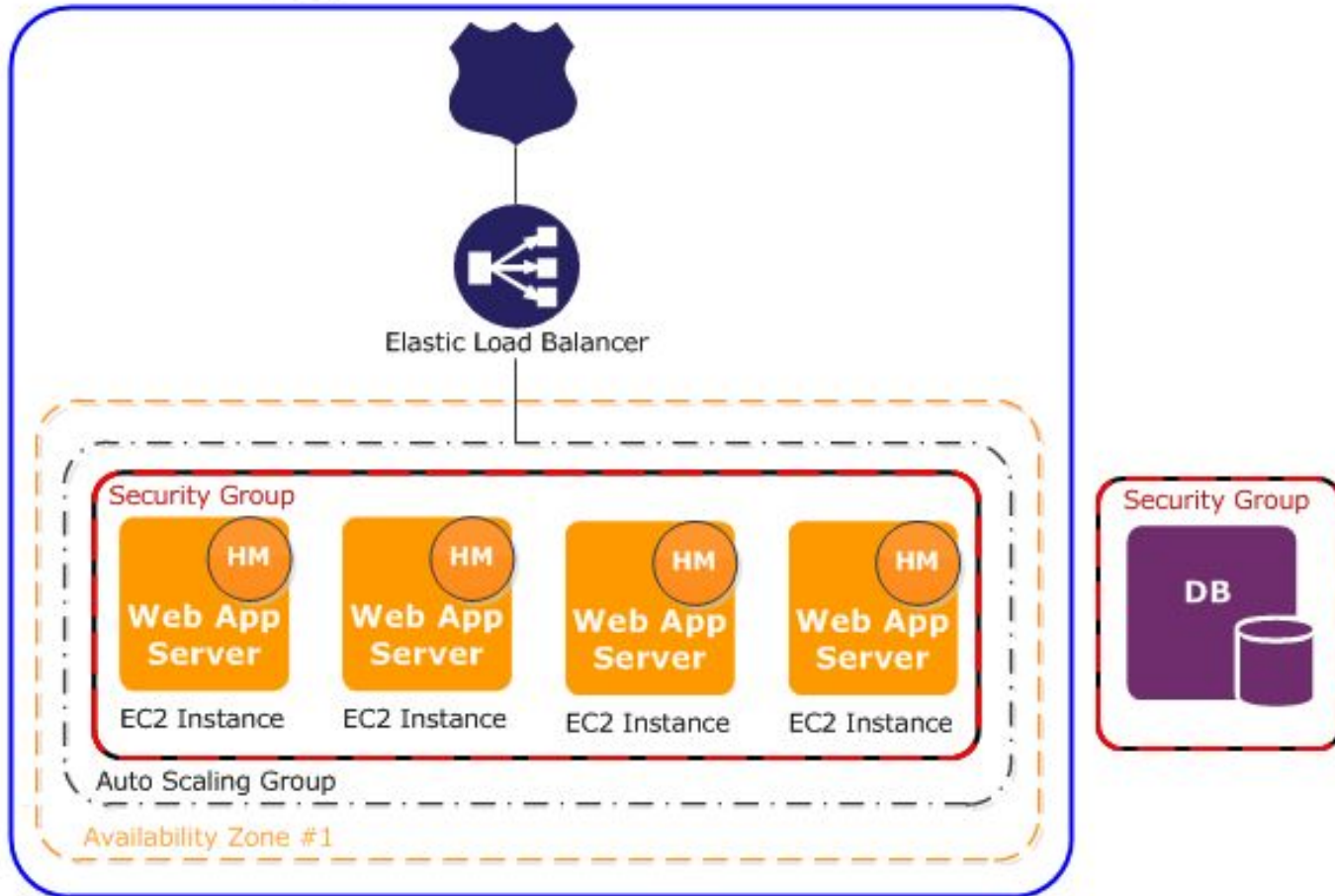
- EB 사용 도구
 - management console
 - eb - Elastic Beanstalk CLI
 - Web API

기초 (4)

- EB Architecture

- CNAME: environment에 대한 human-readable URL -> load balancer
- load balancer: traffic을 multiple instance로 보냄
- auto scaling group: traffic을 처리할 instance를 선정하고 instance 개수 증가/감소 시킴

MyApp.elasticbeanstalk.com



Source



- <https://aws.amazon.com/ko/elasticbeanstalk/>



AWS Theory

- Serverless: Introduction

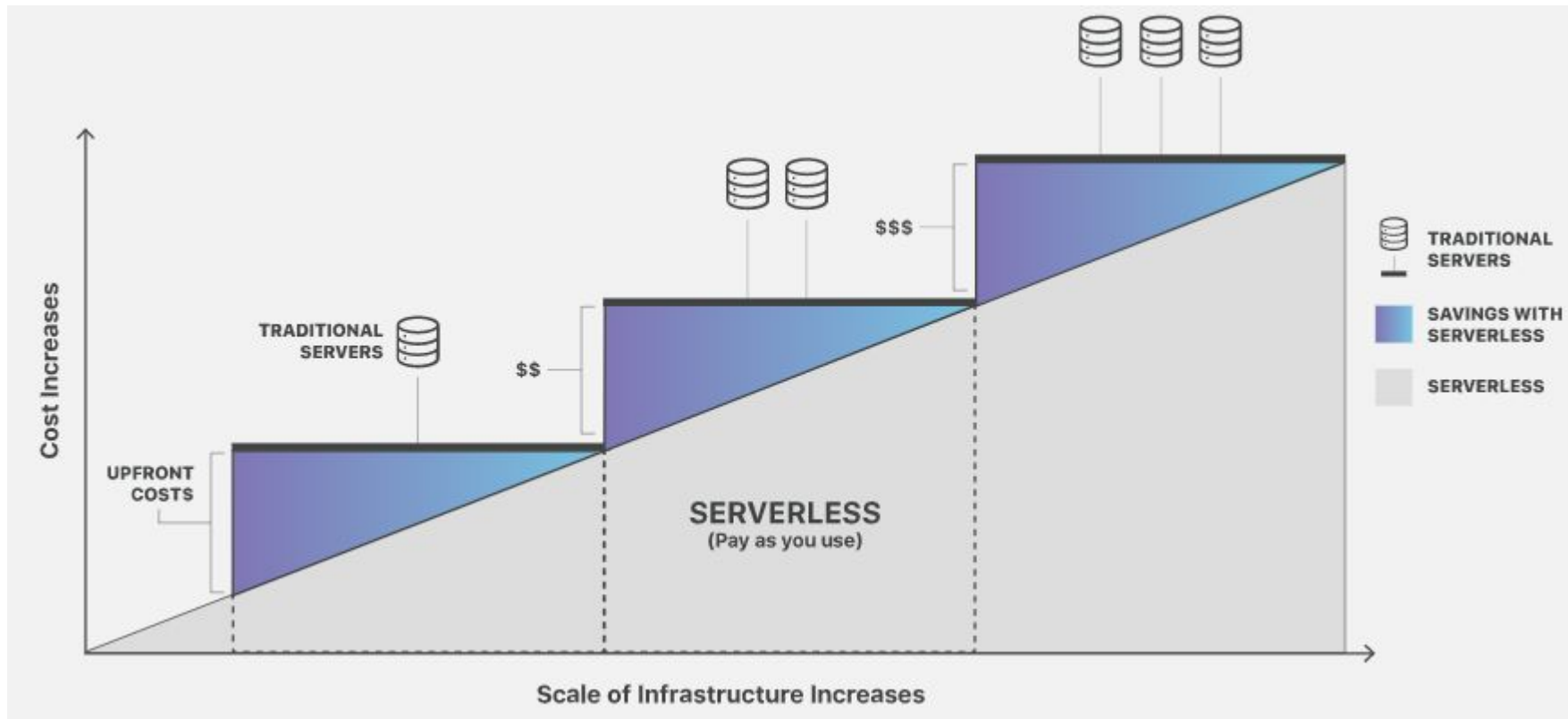
Sung-Dong Kim,
School of Computer Engineering,
Hansung University

Contents

- What - Why
- AWS Serverless Platform
- 이점
- 사례
- Application 구축 사례

What (1)

- 사용량에 따라 백엔드 서비스를 제공하는 방법
- 물리적 서버는 사용되며 개발자는 서버를 알 필요 없음
- 웹 애플리케이션 구축 역사
 - 서버 보유
 - 클라우드 시대 - 고정 개수 서버나 서버 공간 대여
 - 서버리스 - 고정 대역폭이나 서버 개수를 유지하지 않아도 됨 ->
사용량에 따라 백엔드 서비스 비용 지불



CLOUDFLARE: Serverless

What (2)

- 운영상의 책임을 클라우드로 전환 -> 민첩성과 혁신을 높이는 클라우드의 네이티브 아키텍처
- 서버를 고려하지 않고 application과 server를 구축하고 실행
- 거의 모든 application, 백엔드 서비스 구축 가능
- 고가용성 실행/확장에 필요한 사항이 자동 처리 됨

Why?

- 민첩성, 낮은 비용으로 최신 application 개발
- 개발자는 개발에 집중

AWS Serverless Platform (1)

- serverless application 구축/실행을 위한 완전 관리형 서비스 제공
- 백엔드 구성 요소 프로비저닝, 애플리케이션 내결함성/가용성에 대해 고려하지 않아도 됨

AWS Serverless Platform (2)

- Computing: Lambda, Lambda@Edge, Fargate
- Storage: S3, EFS
- Data Store: Dynamo DB, Aurora Serverless
- API Proxy: API Gateway
- Application 통합: SNS, SQS, AppSync, EventBridge
- Ochestration: Step Functions

AWS Serverless Platform (3)

- 분석: Kinesis, Athena
- 개발자 도구

이점

- 낮은 비용 - 종량제 요금
- 서버 관리 불필요
- 유연한 규모 조정: 간편한 확장성
- 자동화된 고가용성
- 빠른 개발: 복잡한 배포 과정, 버그 수정 대신 필요에 따라 코드 추가/수정

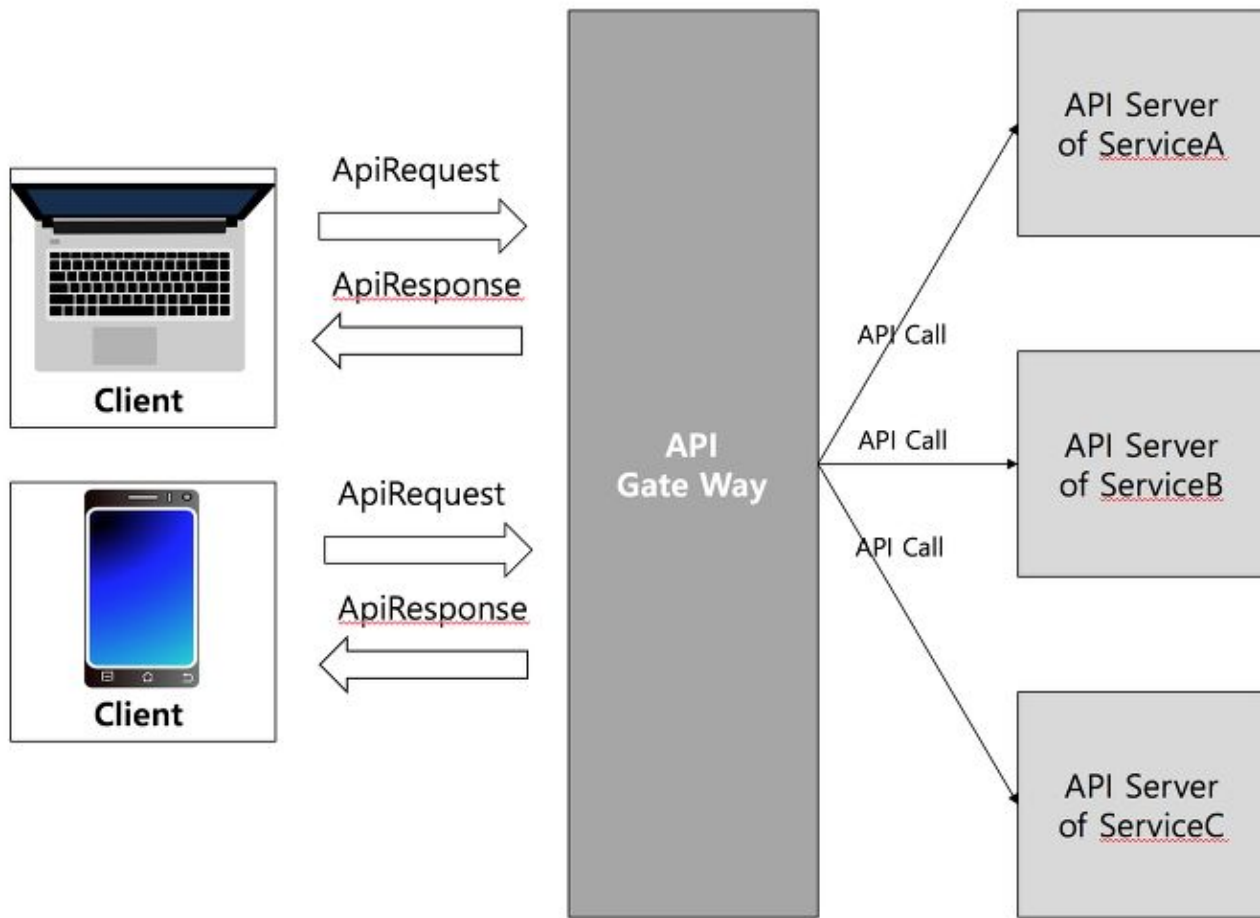
사례

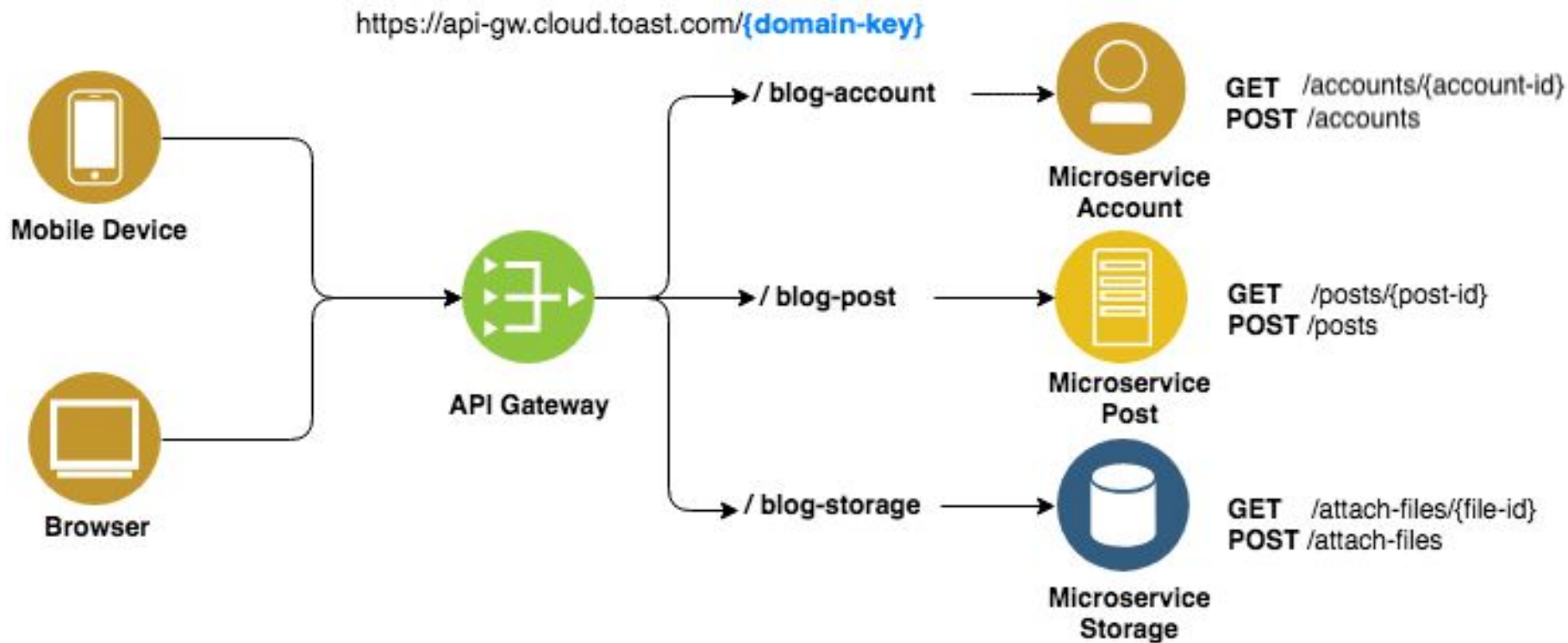
- 코카콜라: Lambda, Step Functions
- FINRA
- iRobot: Lambda, IoT -> Roomba 진공청소기에 연결하는 web application 실행
- Autodesk: Lambda -> IT 운영 자동화

Application 사용 사례 (1)

- 웹 애플리케이션과 백엔드 구축 (1)
 - Lambda, API Gateway, S3, Dynamo DB
 - 예: 날씨 애플리케이션



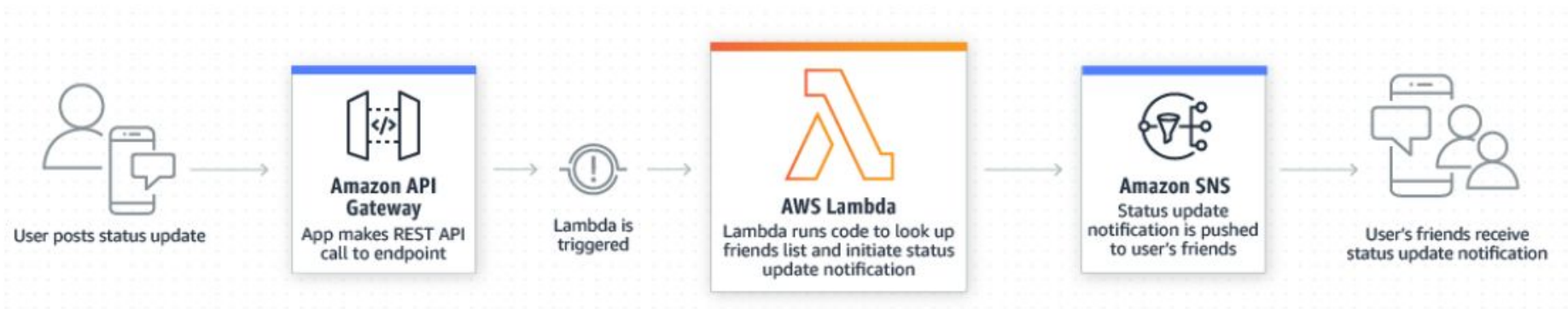




API Gateway 소개 (TOAST Service 들여다보기)

Application 사용 사례 (2)

- 웹 애플리케이션과 백엔드 구축 (2)
 - 예: 소셜 미디어 앱에 대한 모바일 백엔드



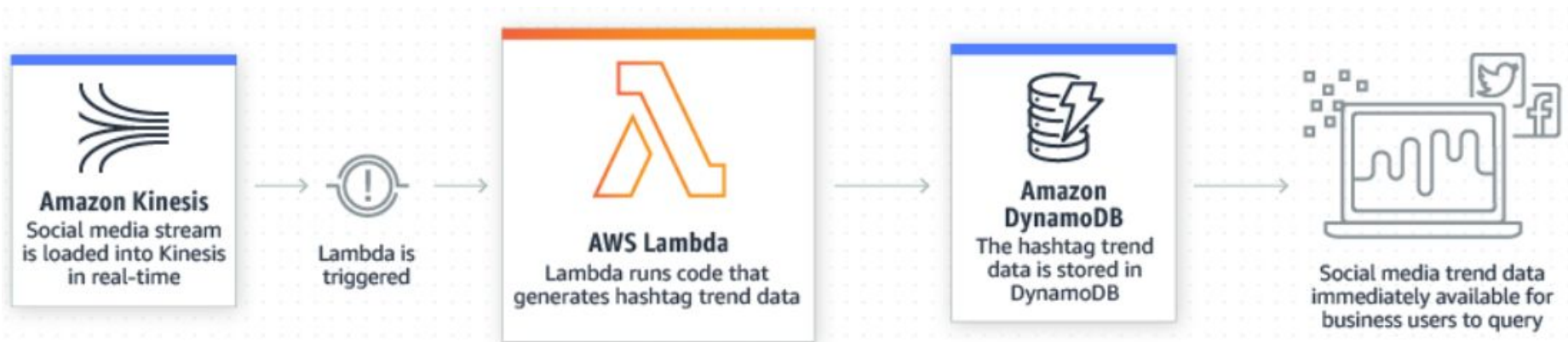
Application 사용 사례 (3)

- 실시간 데이터 처리 시스템 구축 (1)
 - Lambda, Kinesis, S3, Dynamo DB
 - 예: 이미지 썸네일 생성



Application 사용 사례 (4)

- 실시간 데이터 처리 시스템 구축 (2)
 - 예: 스트리밍 소셜 미디어 데이터 분석



Source



- <https://aws.amazon.com/ko/serverless/>



AWS Theory

- AWS Lambda

Sung-Dong Kim,
School of Computer Engineering,
Hansung University

What (1)

- 서버를 프로비저닝하거나 관리하지 않고도 코드를 실행할 수 있게 해주는 컴퓨팅 서비스
- 사용한 컴퓨팅 시간에 대해서만 과금
- 모든 유형의 애플리케이션이나 백엔드 서비스에 대한 코드를 별도의 관리 없이 실행할 수 있습니다.

What (2)

- 코드를 업로드하기만 하면, Lambda에서 높은 가용성으로 코드 실행 및 확장하는 데 필요한 모든 것을 처리
- 다른 AWS 서비스에서 코드를 자동으로 트리거하도록 설정하거나 웹 또는 모바일 앱에서 직접 코드를 호출

what to do (1)

- 고가용성 컴퓨팅 인프라에서 코드를 실행
- 서버 및 운영 체제 유지 관리
- 용량 프로비저닝 및 자동 조정

what to do (2)

- 코드 및 보안 패치 배포
- 코드 모니터링 및 로깅
- 모든 컴퓨팅 리소스 관리를 수행

이점

- 서버 관리 불필요
- 지속적 규모 조정
- 밀리초 단위 측정

Event 기반 실행 환경

- Event 예
 - S3 bucket나 DynamoDB table의 데이터 변화
 - Amazon API gateway를 이용한 HTTP 요청에 대한 응답
 - AWS SDK를 이용한 API 호출
 -

작동 방식



Upload your code to AWS Lambda or write code in Lambda's code editor



Set up your code to trigger from other AWS services, HTTP endpoints, or in-app activity



AWS Lambda

Lambda runs your code only when triggered, using only the compute resources needed



Just pay for the compute time you use

사용 사례 - 데이터 처리 (1)

- 실시간 파일 처리: Lambda, S3
 - S3에 파일을 업로드 하는 즉시 데이터를 처리하도록 lambda 트리거
 - 이미지 썸네일, 동영상 트랜스코딩, 파일 인덱싱, 로그 처리, 콘텐츠 검증, 데이터 수집/필터링,...
 - The Seattle Times: 데스크톱 컴퓨터, 태블릿, 스마트폰 등 다양한 디바이스에서 볼 수 있도록 이미지 크기를 조정 →

<https://aws.amazon.com/ko/solutions/case-studies/the-seattle-times/>



Photograph is taken



Amazon S3
Photo is uploaded
to an S3 Bucket



Lambda is
triggered



AWS Lambda
Lambda runs image
resizing code



Photo is resized into web,
mobile, and tablet sizes

사용 사례 - 데이터 처리 (2)

- 실시간 스트림 처리: Lambda, Kinesis
 - 애플리케이션 활동 추적
 - 트랜잭션 주문 처리
 - 클릭 스트림 분석
 - 데이터 정리
 - 지표 생성

사용 사례 - 데이터 처리 (3)

- 실시간 스트림 처리
 - 로그 필터링
 - 인덱싱, 소셜 미디어 분석, ...
 - **Localytics**: 실시간으로 수십억 개의 데이터 포인트를 처리하고 Lambda를 사용하여 S3에 저장되거나 Kinesis에서 스트리밍 된 기록 데이터 및 실시간 데이터를 처리 →

<https://aws.amazon.com/ko/solutions/case-studies/localytics/>



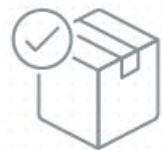
Lambda is
triggered



Social media trend data
immediately available for
business users to query

사용 사례 - 데이터 처리 (4)

- 추출, 변환, 로드
 - DynamoDB 테이블의 모든 데이터 변경에 대한 데이터 검증, 필터링, 정렬 또는 기타 변환 작업을 수행하고 변환된 데이터를 다른 데이터 스토어로 로드
 - Zillow: Lambda 및 Kinesis를 사용하여 실시간으로 모바일 지표의 하위 집합을 추적 ⇒ Lambda 및 Kinesis를 통해 2주 만에 비용 효과적인 솔루션을 개발 및 배포할 수 있었음



Online order is placed



**Amazon
DynamoDB**

Order data is stored
in an operational
database



Lambda is
triggered



AWS Lambda

Lambda runs data
transformation code



Amazon Redshift

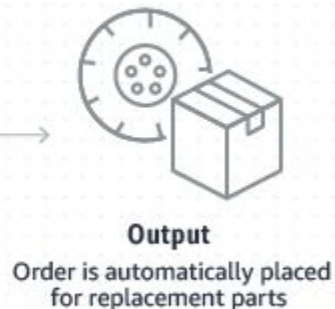
Lambda loads results
into data warehouse



Analytics generated
from data

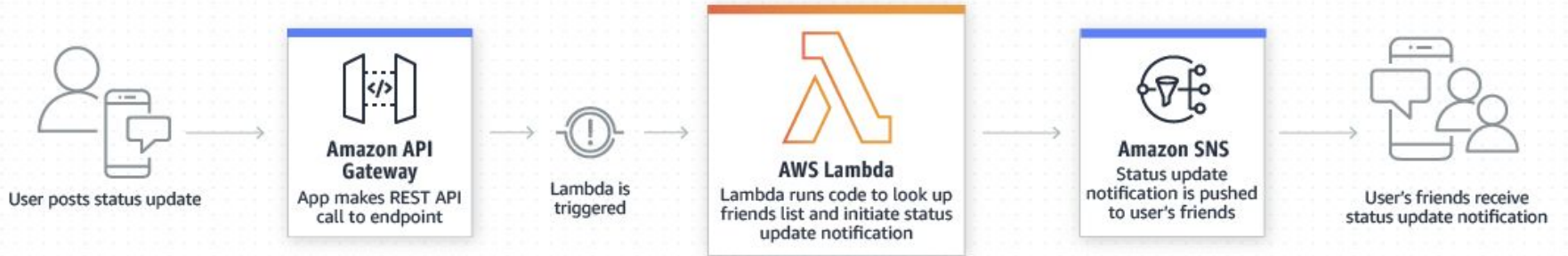
사용 사례 - 백엔드 (1)

- IOT 백엔드: 웹, 모바일, 사물 인터넷 및 타사 API 요청



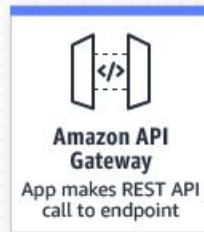
사용 사례 - 백엔드 (2)

- 모바일 백엔드: API 요청을 인증 및 처리하도록 백엔드 구성 → 개인화된 앱 환경을 손쉽게 생성할 수 있음
- Bustle: AWS Lambda와 Amazon API Gateway를 사용하여 Bustle iOS 앱 및 웹 사이트에 대해 서버리스 백엔드를 실행



사용 사례 - 백엔드 (3)

- 웹 애플리케이션
 - Lambda를 다른 AWS 서비스와 결합
 - 가용성 높은 구성에서 실행되는 강력한 웹 애플리케이션 구축 가능



source



- <https://aws.amazon.com/ko/lambda/>

Elastic Load Balancer - practice

Sung-Dong Kim,
School of Computer Engineering,
Hansung University

Contents

- ▶ 네트워크 구성
- ▶ EC2 instance 시작
- ▶ Storage 관리
- ▶ Custom EC2 instance 생성
- ▶고가용성 웹 서비스 - Elastic Load Balancing
- ▶ 리소스 삭제

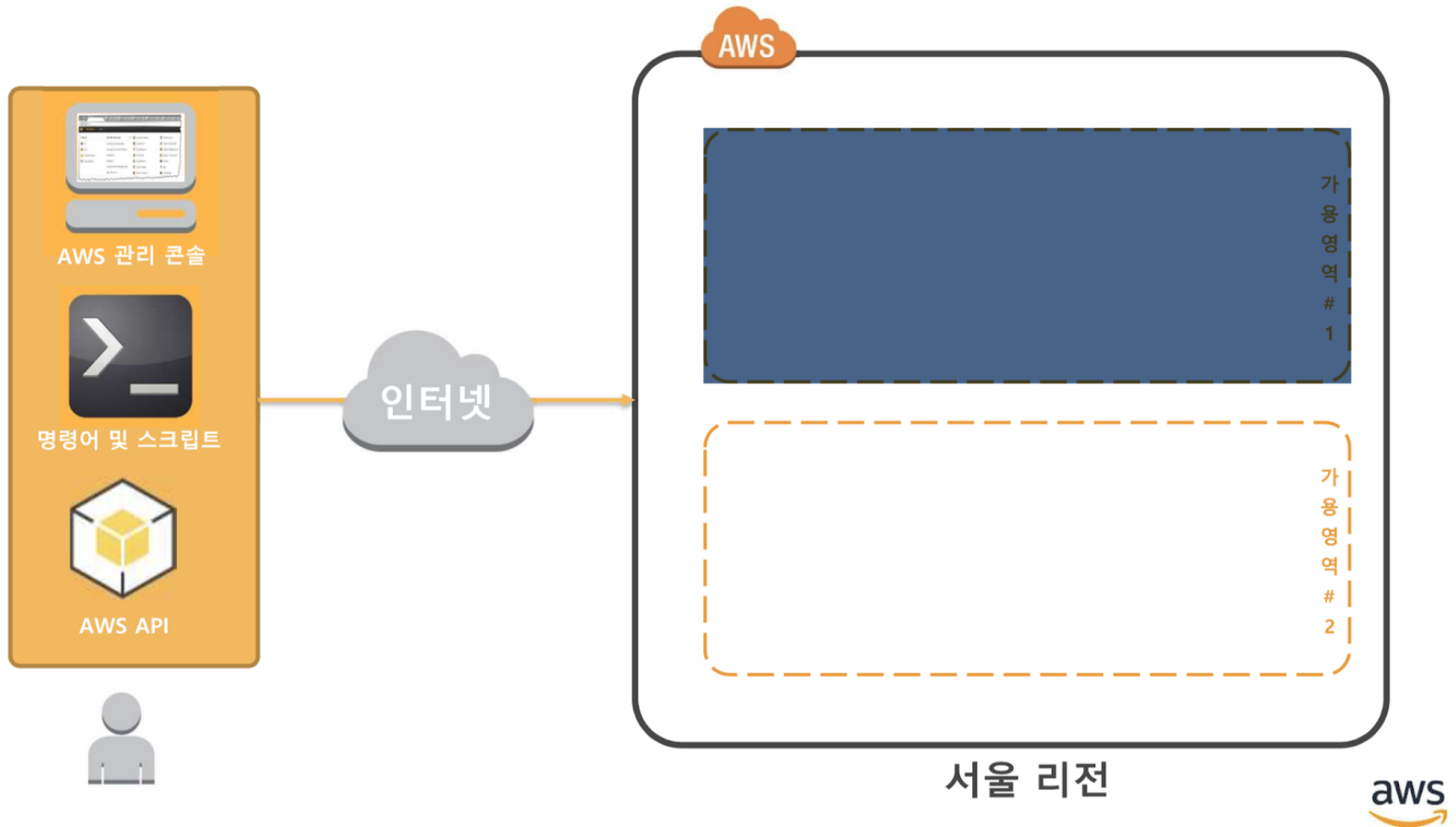
네트워크 구성

네트워크 구성

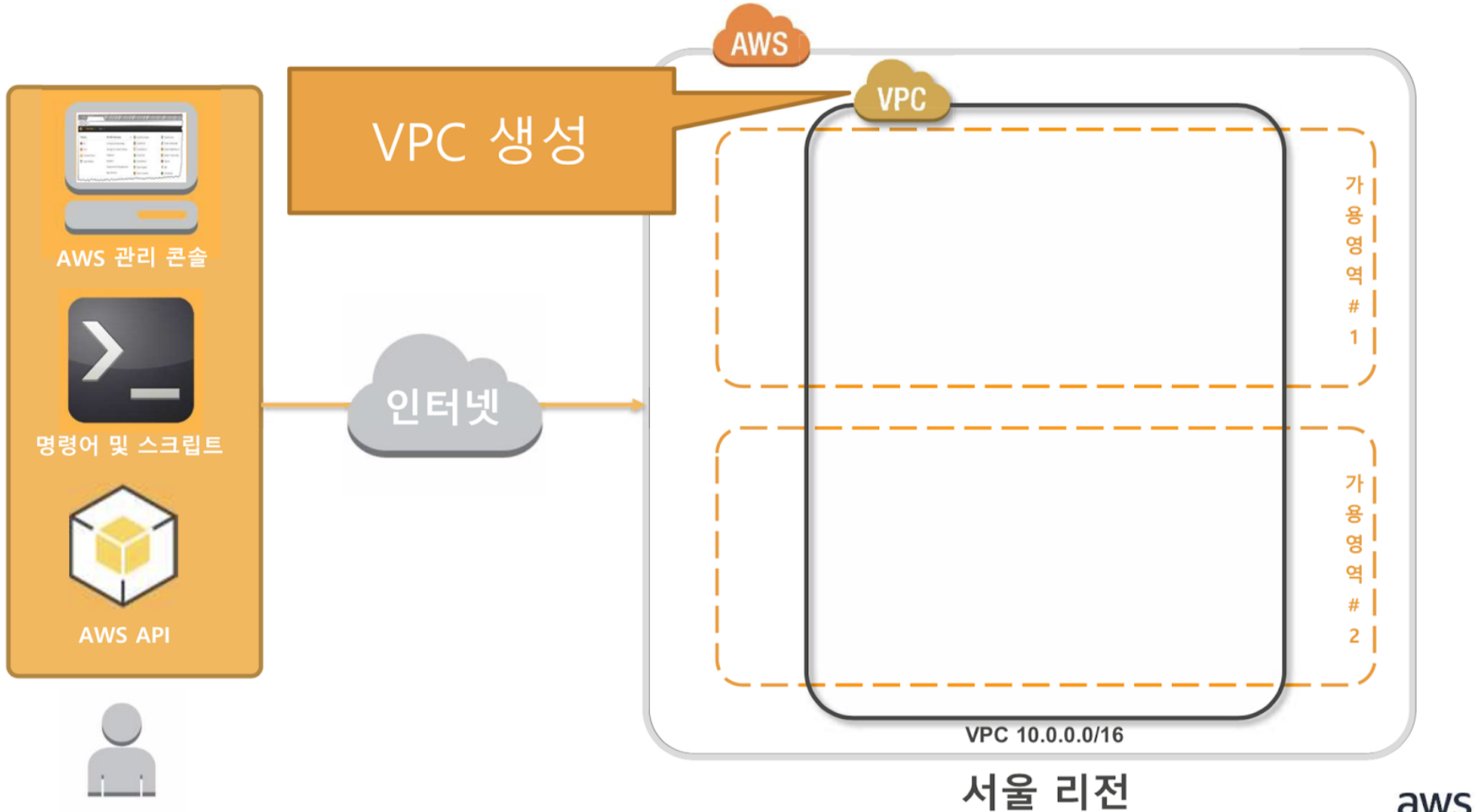
▶ VPC 생성

- ▶ IP 주소 범위 선택: 연결할 수 있는 다른 네트워크와 겹치는 범위는 회피
- ▶ 추천: /16 (64K 주소들)
- ▶ 2개의 가용 영역
- ▶ 2개의 퍼블릭 서브넷
- ▶ 인터넷 게이트웨이
- ▶ 라우팅 테이블
- ▶ 보안 그룹: SSH (22), HTTP (80)

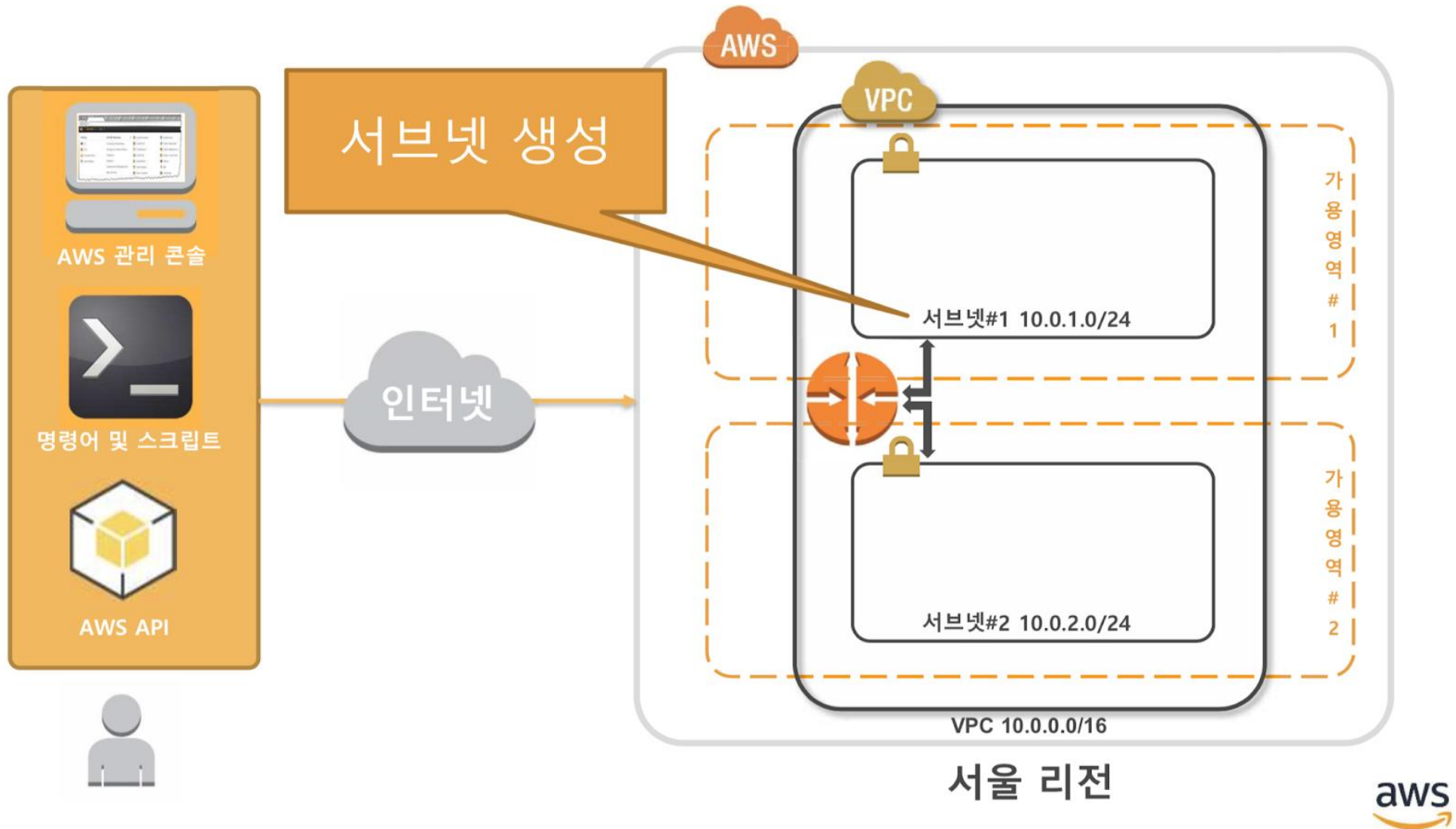
네트워크 구성



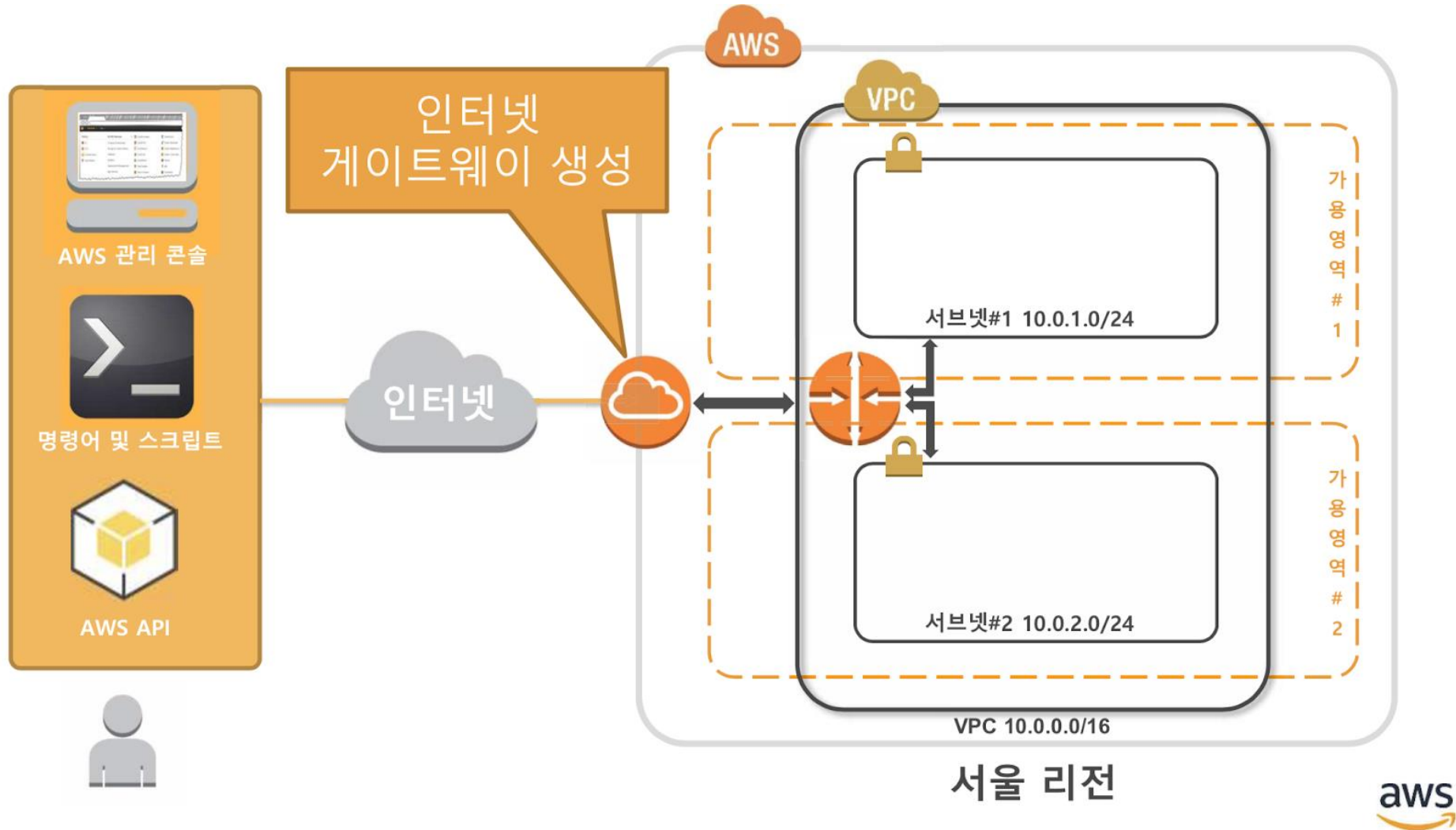
네트워크 구성



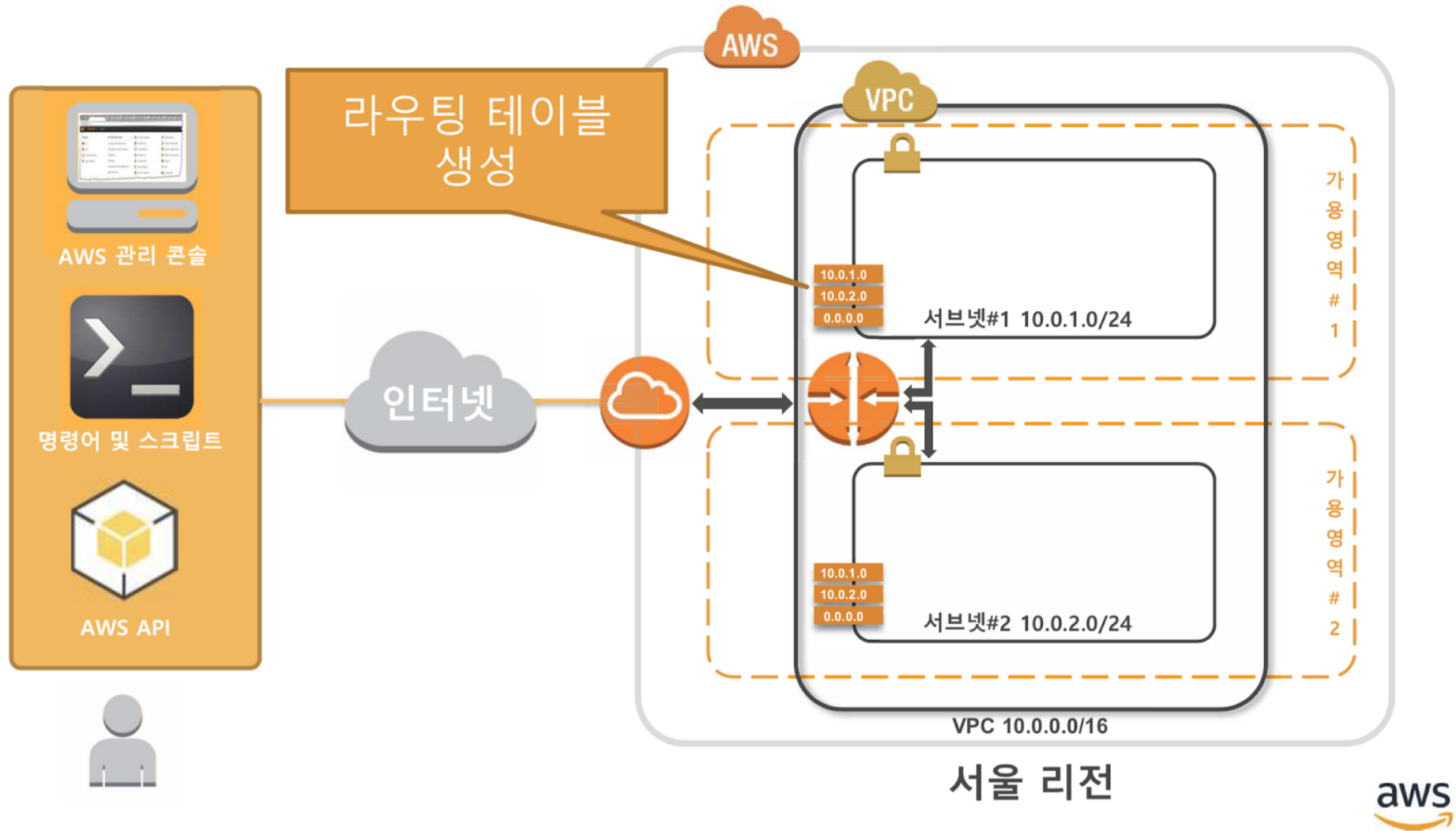
네트워크 구성



네트워크 구성



네트워크 구성



네트워크 구성

VPC 설정

생성할 리소스 정보

VPC 리소스 또는 VPC 및 기타 네트워킹 리소스만 생성합니다.

☐ VPC만

☒ VPC 등

이름 태그 자동 생성 정보

이름 태그의 값을 입력합니다. 이 값은 VPC의 모든 리소스에 대한 이름으로 생성하는 데 사용됩니다.

☒ 자동 생성

ksd-elb

IPv6 CIDR 블록 정보

☒ IPv6 CIDR 블록 없음

☐ Amazon 제공 IPv6 CIDR 블록

테넌시 정보

기본값

가용 영역(AZ) 수 정보

서브넷을 프로비저닝할 AZ 수를 선택합니다.고가용성을 위해서는 최소 2개 이상의 AZ를 사용하는 것이 좋습니다.

1

2

3

▼ AZ 사용자 지정

첫 번째 가용 영역

ap-northeast-2a

두 번째 가용 영역

ap-northeast-2c

네트워크 구성

퍼블릭 서브넷 수 **정보**

VPC에 추가할 퍼블릭 서브넷 수입니다. 인터넷을 통해 공개적으로 액세스할 수 있어야 하는 웹 애플리케이션에는 퍼블릭 서브넷을 사용합니다.

0	2
---	---

프라이빗 서브넷 수 **정보**

VPC에 추가할 프라이빗 서브넷 수입니다. 프라이빗 서브넷을 사용하여 퍼블릭 액세스가 필요 없는 백엔드 리소스를 보호합니다.

0	2	4
---	---	---

▼ 서브넷 CIDR 블록 사용자 지정

ap-northeast-2a 퍼블릭 서브넷 CIDR 블록

10.0.1.0/24	256 IPs
-------------	---------

ap-northeast-2c 퍼블릭 서브넷 CIDR 블록

10.0.2.0/24	256 IPs
-------------	---------

네트워크 구성

NAT 게이트웨이(\$) 정보

NAT 게이트웨이를 생성할 가용 영역(AZ) 수를 선택합니다. 각 NAT 게이트웨이마다 요금이 부과됩니다.

없음	1개의 AZ에서	AZ당 1개
----	----------	--------

VPC 엔드포인트 정보

엔드포인트는 VPC에서 S3에 직접 액세스하여 NAT 게이트웨이 요금을 줄이고 보안을 강화할 수 있습니다. 기본적으로 모든 액세스 정책이 사용됩니다. 언제든지 이 정책을 사용자 지정할 수 있습니다.

없음	S3 게이트웨이
----	----------

DNS 옵션 정보

- ☒ DNS 호스트 이름 활성화
- ☒ DNS 확인 활성화

네트워크 구성

VPC 세부 정보 표시

AWS 가상 네트워크

ksd-elb-vpc

서브넷(2개)

이 VPC 내의 서브넷

ap-northeast-2a

ksd-elb-subnet-public1-ap-northeast-2a

ap-northeast-2c

ksd-elb-subnet-public2-ap-northeast-2c

네트워크 구성

라우팅 테이블(1개)

네트워크 트래픽을 리소스로 라우팅

ksd-elb-rtb-public








네트워크 연결(2개)

다른 네트워크에 연결

ksd-elb-igw

ksd-elb-vpce-s3

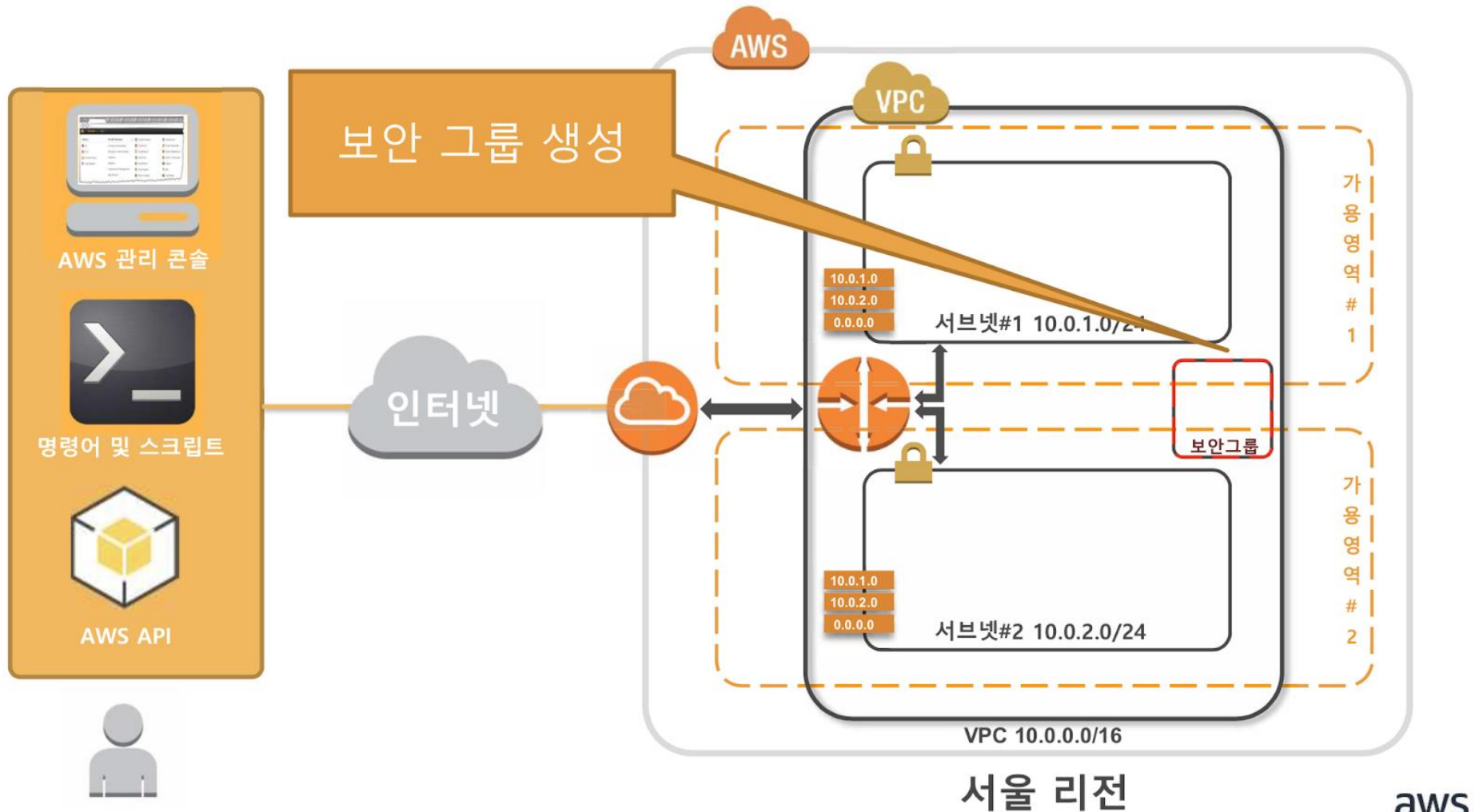
네트워크 구성

- ✔ VPC 생성: [vpc-0cfd6b570b37165dc](#) 
- ✔ DNS 호스트 이름 활성화
- ✔ DNS 확인 활성화
- ✔ VPC 생성 확인: [vpc-0cfd6b570b37165dc](#) 
- ✔ S3 엔드포인트 생성: [vpce-0e1b8a56f81a118bf](#) 
- ✔ 서브넷 생성: [subnet-099bba492df324dbc](#) 
- ✔ 서브넷 생성: [subnet-050ff6d2d86f399e1](#) 
- ✔ 인터넷 게이트웨이 생성: [igw-0d2bdd470fa5fd918](#) 
- ✔ VPC에 인터넷 게이트웨이 연결
- ✔ 라우팅 테이블 생성: [rtb-0f072fcc05028de93](#) 
- ✔ 경로 생성
- ✔ 라우팅 테이블 연결
- ✔ 라우팅 테이블 연결
- ✔ 라우팅 테이블 생성 확인

네트워크 구성

- ▶ VPC 확인: ksd-elb-vpc, CIDR
- ▶ 서브넷 확인: ksd-elb-subnet-public1, ksd-elb-subnet-public2
- ▶ 라우팅 테이블 확인: ksd-elb-rtb-public
 - ▶ 라우팅
 - ▶ 서브넷 연결
- ▶ 인터넷 게이트웨이 확인: ksd-elb-igw
 - ▶ 세부 정보: attached
- ▶ 엔드포인트 확인: ksd-elb-vpce-s3

네트워크 구성

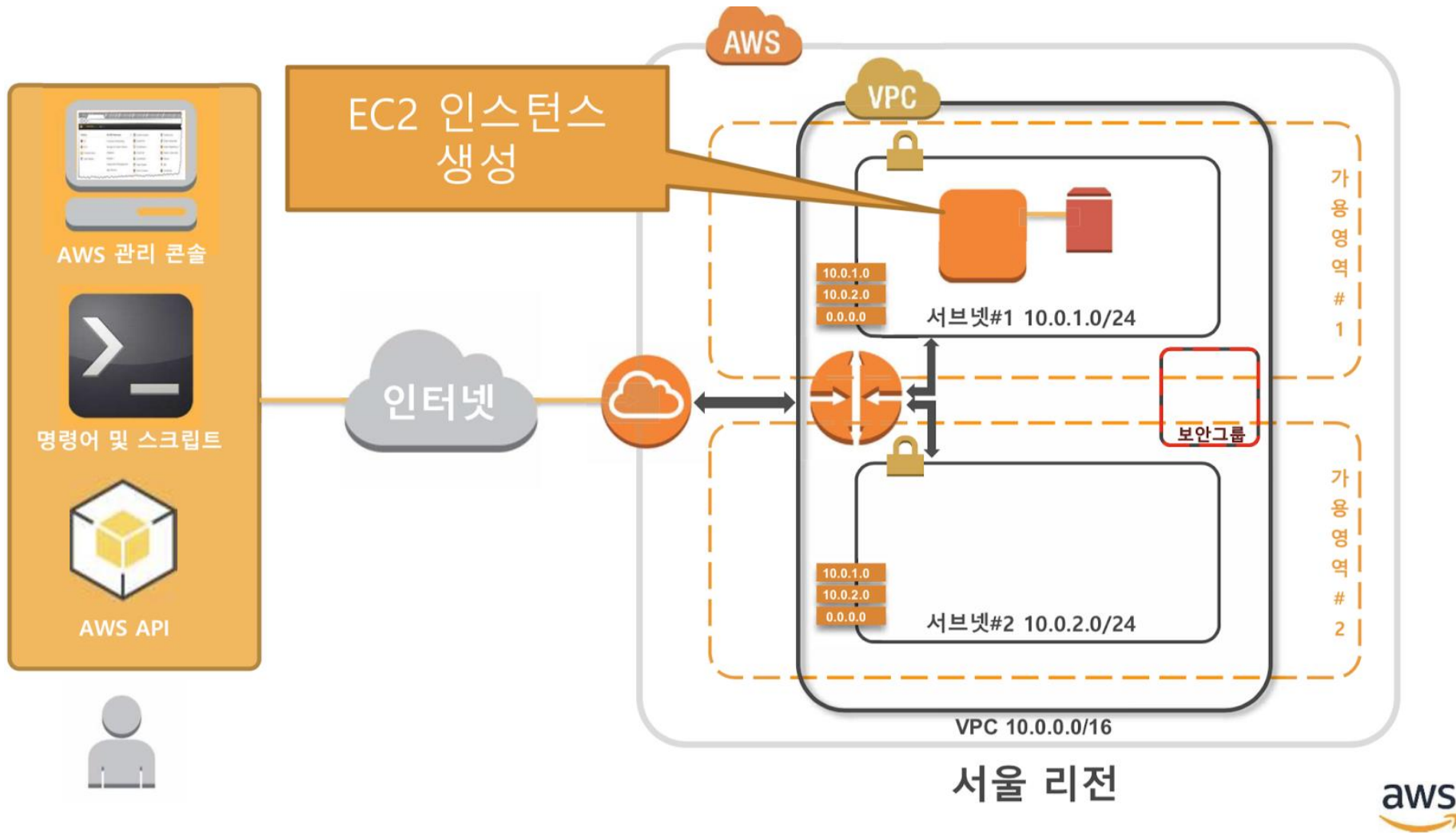


EC2 instance 시작

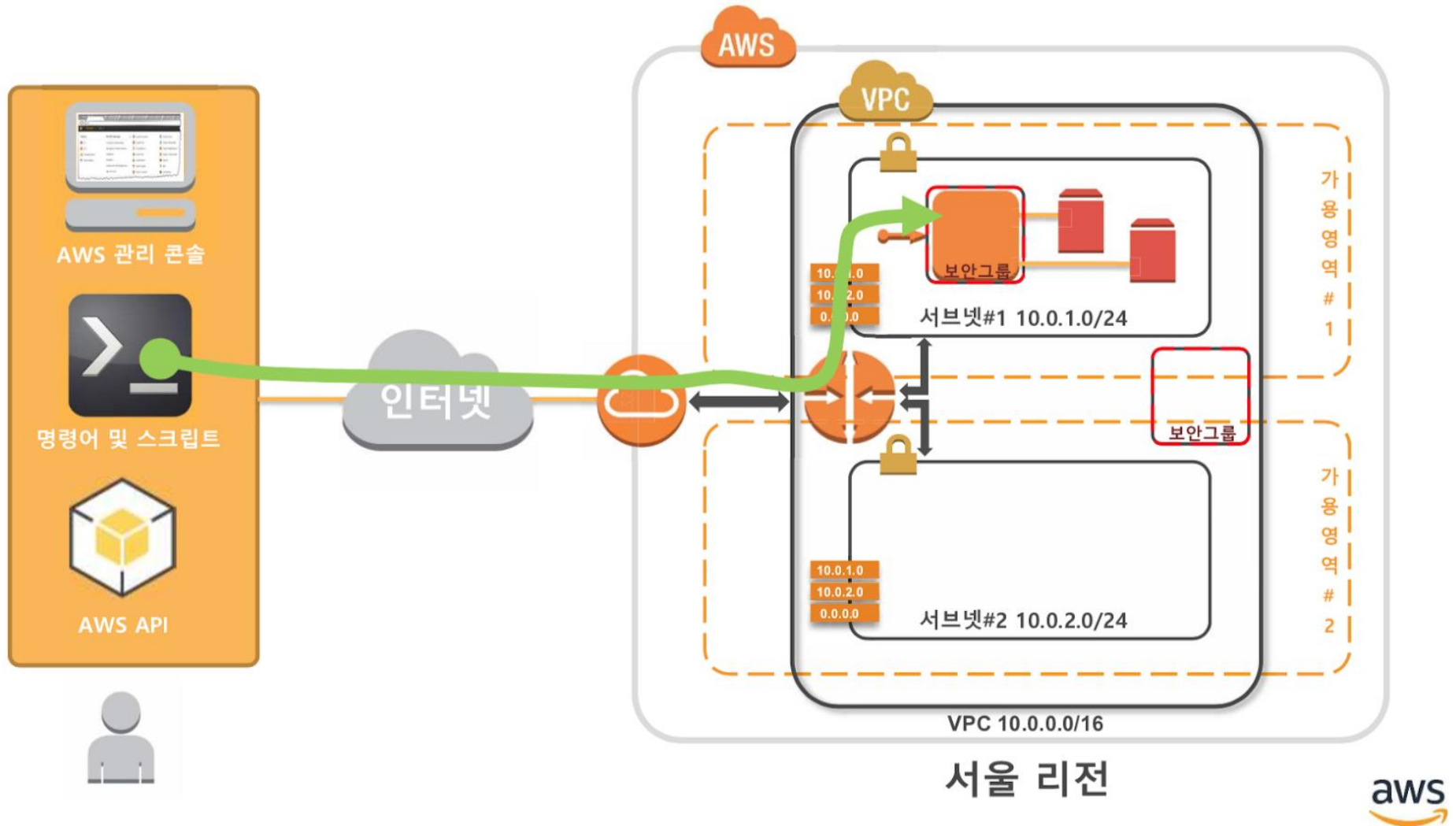
EC2 instance 시작

- ▶ 키페어: 생성 또는 기존 것 이용
- ▶ 인스턴스 생성: Amazon Linux 2 AMI
 - ▶ ksd-elb-vpc
 - ▶ ksd-elb-subnet-public1
 - ▶ ksd-elb-sg
- ▶ Elastic IP 생성 및 연결
- ▶ Web server, PHP server 설치

EC2 instance 시작



EC2 instance 시작



EC2 instance 시작

- ▶ PHP script: myip.php

```
<?php  
echo "Hello! My IP address is: ".$_SERVER['SERVER_ADDR']; ?>
```

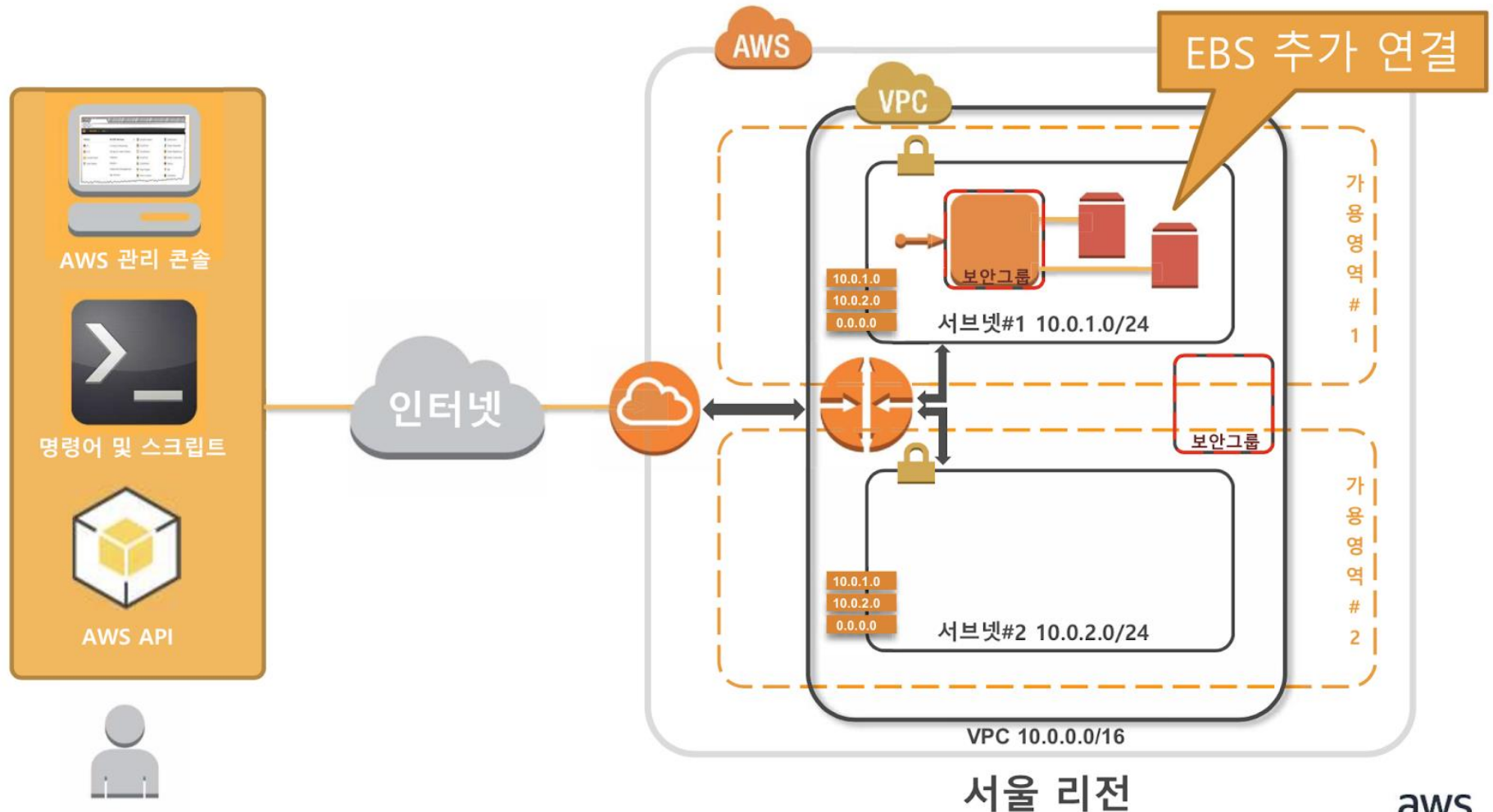
- ▶ test: EIP/myip.php

Storage 관리

Storage 관리

- ▶ EBS volume 생성: web-server-volume1
- ▶ EBS volume을 EC2에 연결

Storage 관리

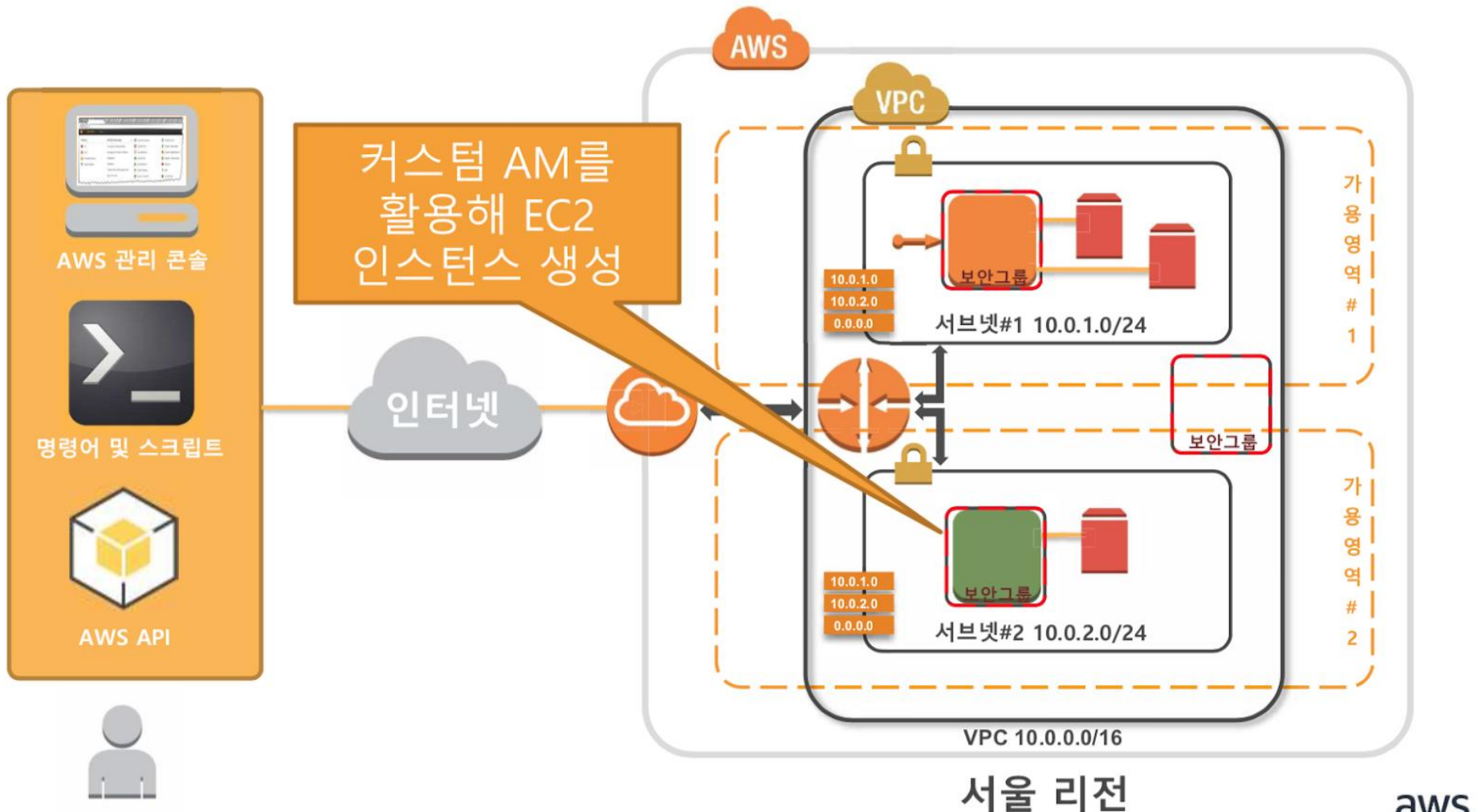


Custom EC2 instance 생성

Custom EC2 instance 생성

- ▶ Custom AMI 생성
- ▶ Custom AMI로 instance 생성
 - ▶ ksd-elb-vpc
 - ▶ ksd-elb-subnet-public2
- ▶ Elastic IP 연결: 새로 만든 instance에 연결
- ▶ Test

Custom EC2 instance 생성



고가용성 웹 서비스

고가용성 웹 서비스

- ▶ ELB 생성 및 ELB에 EC2 인스턴스 연결
 - ▶ web server 모두 시작 후
 - ▶ 로드밸런서 → 생성 → Application Load Balancer
 - ▶ 이름
 - ▶ 체계: 인터넷 경계
 - ▶ IP 주소유형: IPv4
 - ▶ VPC: ksd-elb-vpc
 - ▶ 매핑: 2개 서브넷 선택
 - ▶ 보안 그룹: ksd-elb-sg

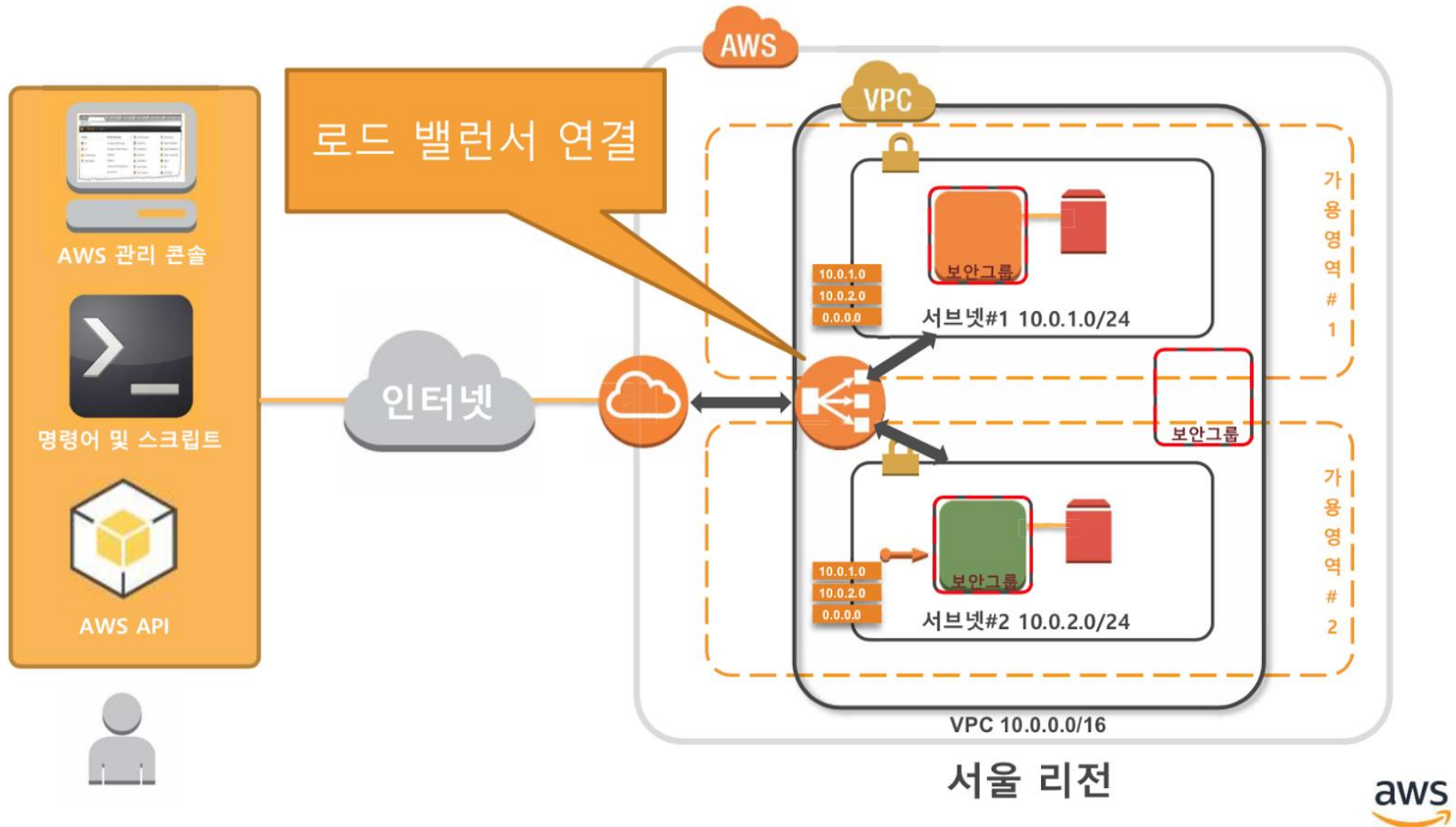
고가용성 웹 서비스

- ▶ 리스너 HTTP:80
- ▶ 대상 그룹 생성
 - ▶ 대상 유형: 인스턴스
 - ▶ 대상 그룹 이름: ksd-lb-group
 - ▶ VPC: ksd-elb-vpc
 - ▶ 프로토콜 버전: HTTP1
 - ▶ 다음 → 사용 가능한 인스턴스: 2개 web server 선택

▶ ELB 동작 확인

- ▶ DNS 확인 → dns/myip.php

고가용성 서비스



리소스 삭제

리소스 삭제

- ▶ ELB - 삭제
- ▶ Elastic IP - 주소 연결 해제 후, 릴리스
- ▶ EC2 인스턴스 - 종료
- ▶ custom AMI - 등록 취소 (삭제)
- ▶ EBS volume - 삭제
- ▶ VPC 삭제
 - ▶ default VPC를 삭제하면 안됨
 - ▶ AWS Practice - VPC 자료 참고

Sung-Dong Kim,
School of Computer Engineering,
Hansung University

Sung-Dong Kim,
School of Computer Engineering,
Hansung University

What to study

- ④ **PHP application 시작 및 실행**
- ④ 컴퓨팅 → Elastic Beanstalk



PHP application 생성

✓ Elastic Beanstalk → Create Application

✓ 웹 앱 생성

- 이름
- 태그
- 플랫폼 - php8.1
- 애플리케이션 코드 - 샘플 애플리케이션



Elastic Beanstalk > 환경 > Ksdphpapp-env



Ksdphpapp-env 생성
몇 분 정도 걸립니다..

5: 13오후 Created security group named:
awseb-e-yb3mcrpfnv-stack-AWSEBSecurityGroup-RL3FV7N9IEST

5: 13오후 Environment health has transitioned to Pending. Initialization in progress (running for 10 seconds). There are no instances.

5: 13오후 Created security group named:
sg-0a07b53b173eceaee

5: 13오후 Created target group named:
arn:aws:elasticloadbalancing:ap-northeast-2:352485268054:targetgroup/awseb-AWSEB-1EYT3A4MV8CR2/5c3b5ae5d3b2a55e

Ksdphpapp-env

Ksdphpapp-env.eba-2prewqkc.ap-northeast-2.elasticbeanstalk.com (e-yb3mcrpfnv)
애플리케이션 이름: **ksd-php-app**

↺ 새로 고침

작업 ▼

상태



확인

원인

실행 버전

Sample Application

업로드 및 배포

플랫폼



PHP 8.1 running on 64bit
Amazon Linux 2/3.5.0

변경 사항

- ④ S3
- ④ EC2 instance → Ksdphpapp-env
- ④ 로드 밸런싱
 - 로드 밸런서
 - 대상그룹
- ④ Auto Scaling
 - Auto Scaling 그룹



Beanstalk application 액세스

☑ Environment URL

- Ksdphpapp-env.eba-2prewqkc.ap-northeast-2.elasticbeanstalk.com

Application update

Application update

- ④ index.php 내용 변경 → php-v2.zip
- ④ [업로드 및 배포] → php-v2.zip 업로드
- ④ URL 재접속
- ④ 종료 후, [환경 종료]

- ✓ https://docs.aws.amazon.com/ko_kr/elasticbeanstalk/latest/dg/GettingStarted.html
- ✓ <https://aws.amazon.com/ko/getting-started/tutorials/launch-an-app/>