



# **AWS 이론 5**

## **- Auto Scaling**

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

# 가용성/확장성

- Availability

- 시스템이나 서비스가 가동 및 실행되는 시간의 비율

- Scalability

- 서비스나 응용 프로그램이 증가하는 성능 요구에 맞게 향상될 수 있는 정도
- scale up, scale out

# What

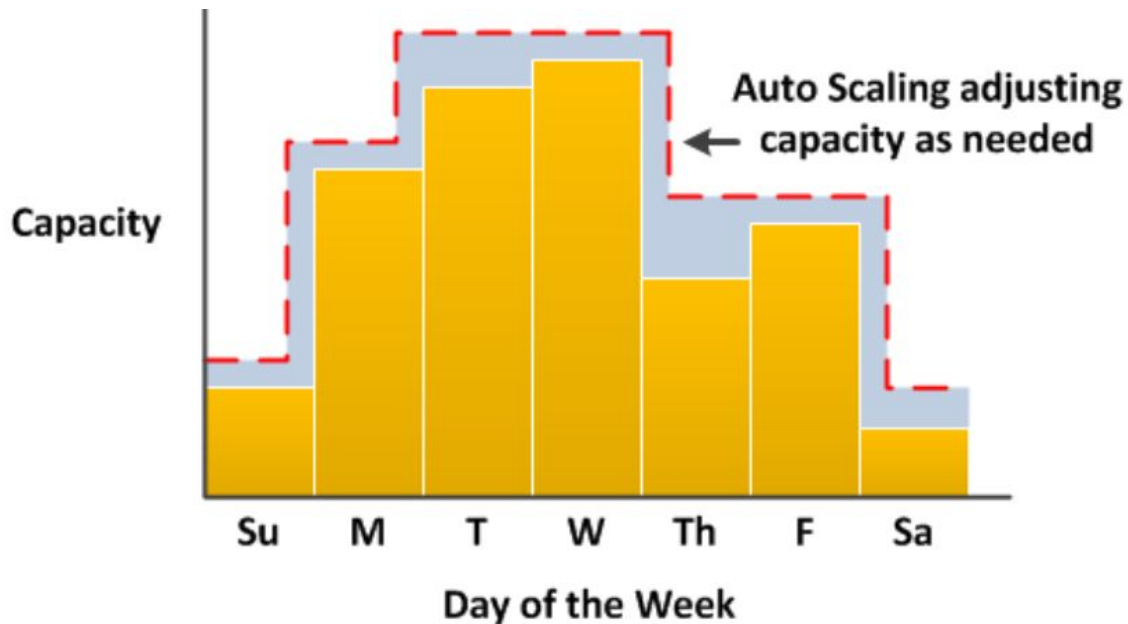
- Application의 로드를 처리할 수 있는 정확한 수의 EC2 instance를 보유하도록 보장
- **Auto scaling group**
  - EC2 instance 모음
  - 최대, 최소, 목표 instance 개수 지정
- Application의 가용성을 간편하게 관리: **조정 정책**을 지정하여 application 수요에 따라 instance 시작/종료

## 이점 (1)

- 내결함성 향상: 비정상적 instance를 탐지하여 정상적 instance로 대체
- 가용성 향상: 트래픽을 처리할 수 있는 적절한 용량
- 비용 관리 개선: 동적으로 instance 시작/종료

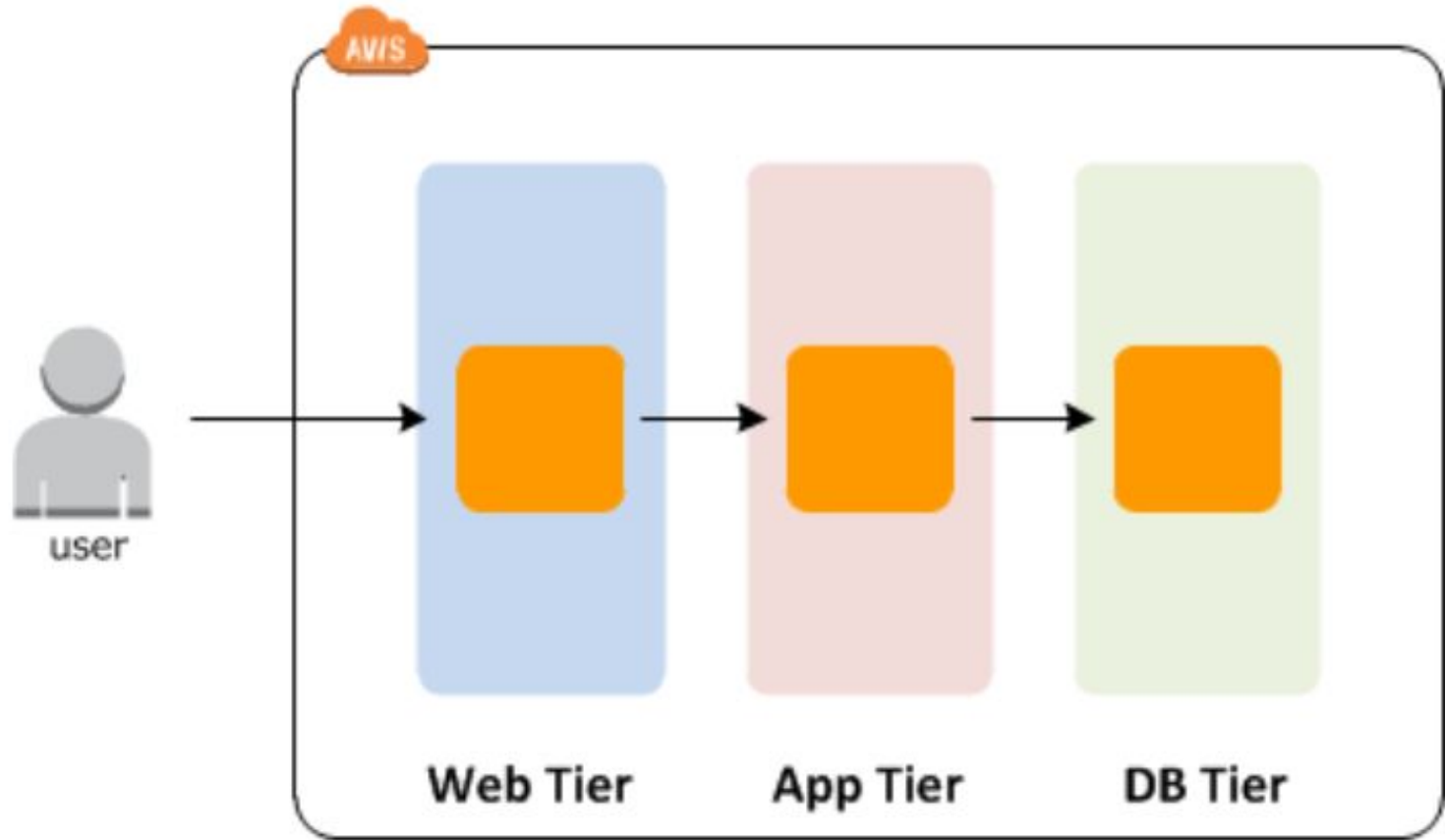
## 이점 (2)

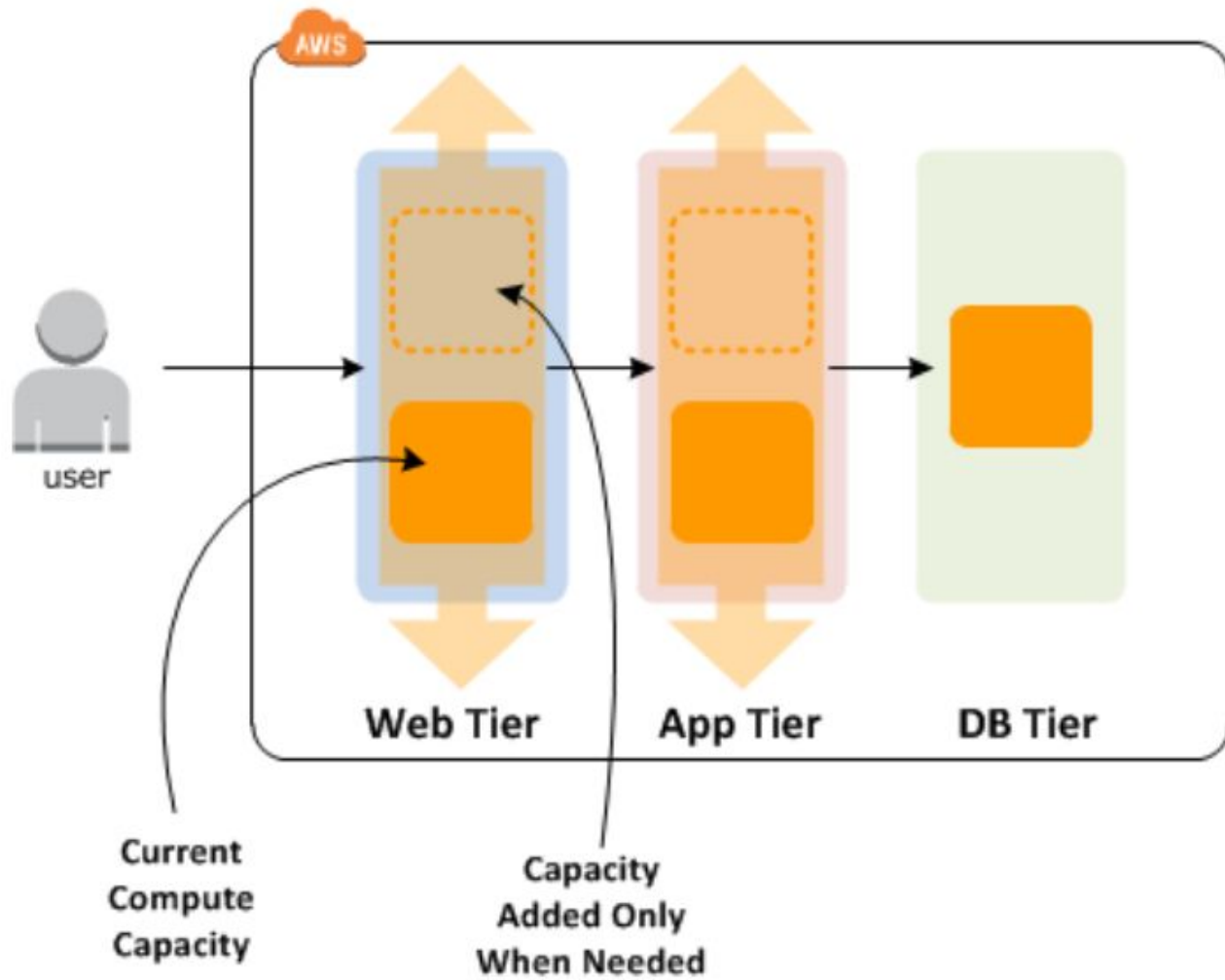
- 가변 수요에 대처



## 이점 (3)

- 웹 앱 아키텍처
  - 고객의 트래픽을 처리하기 위해 여러 개의 **app** 사본을 하나의 EC2 instance (cloud server)에서 호스팅, 각각에서 고객 요청이 처리됨
  - Auto scaling은 EC2 instane 시작/종료를 관리
  - 시작/종료 시기를 결정하는 조건(CloudWatch 경보 등) 집합 정의
  - application의 가용성, 내결함성 향상
  - ELB: 인스턴스 간 트래픽 분산





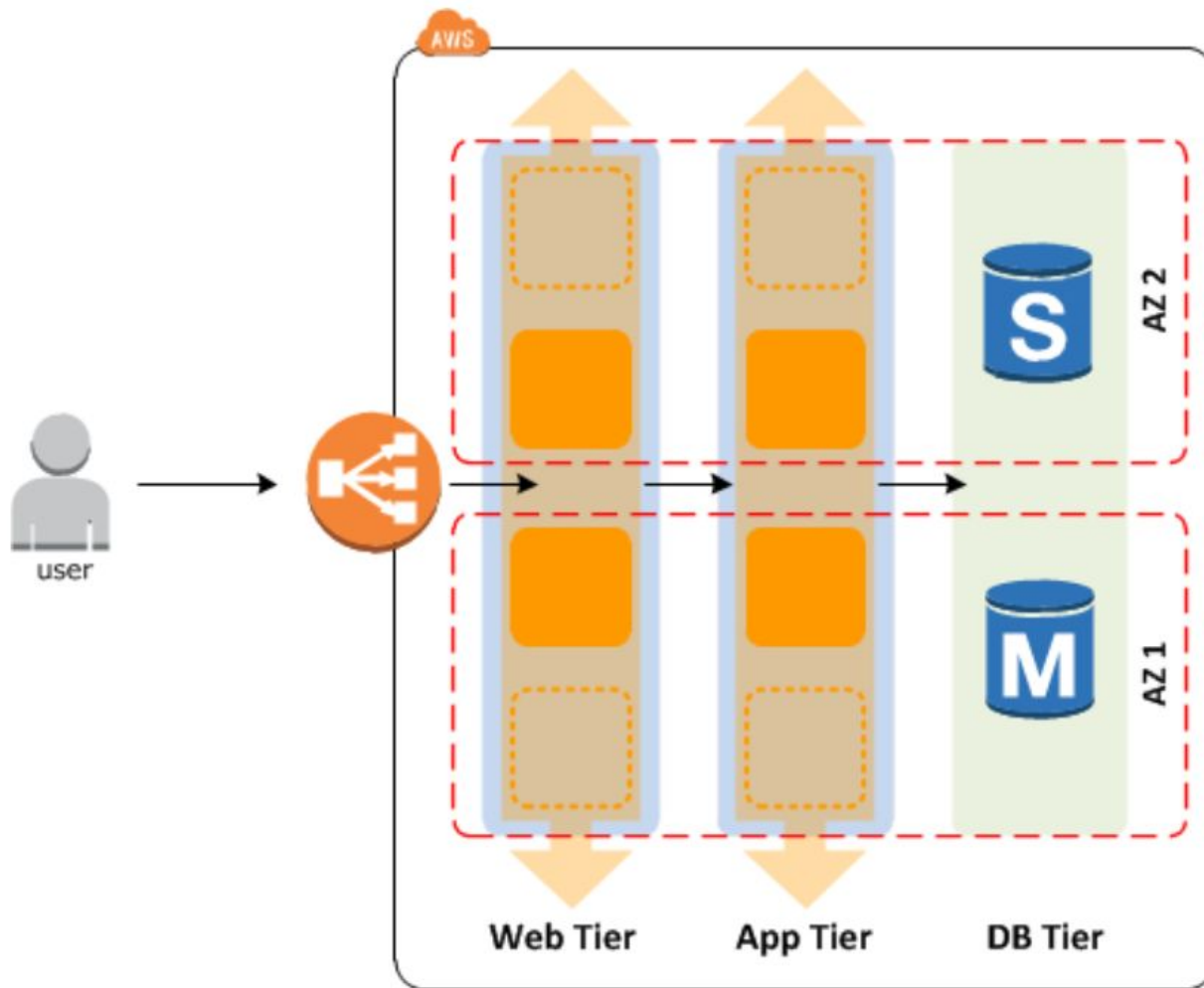


## 이점 (4)

- 가용 영역 전반에 instance 분산
  - region내 여러 AZ에 걸쳐 auto scaling 그룹 확장 → 지리적 이중화를 통한 보안 및 안정성 확보
  - instance를 AZ 간에 고르게 분산하려고 시도
  - VPC 내의 auto scaling group
    - subnet에서 EC2 instance가 시작
    - 한 AZ에 여러 subnet이 있으면 subnet을 무작위로 선택하여 시작

## 이점 (4)

- 가용 영역 전반에 instance 분산
  - 재분배 활동
    - AZ 간에 불균형시, 재분배 작업 수행
    - 이전 instance 종료 전, 새 instance 시작



# 구성 요소 (1)

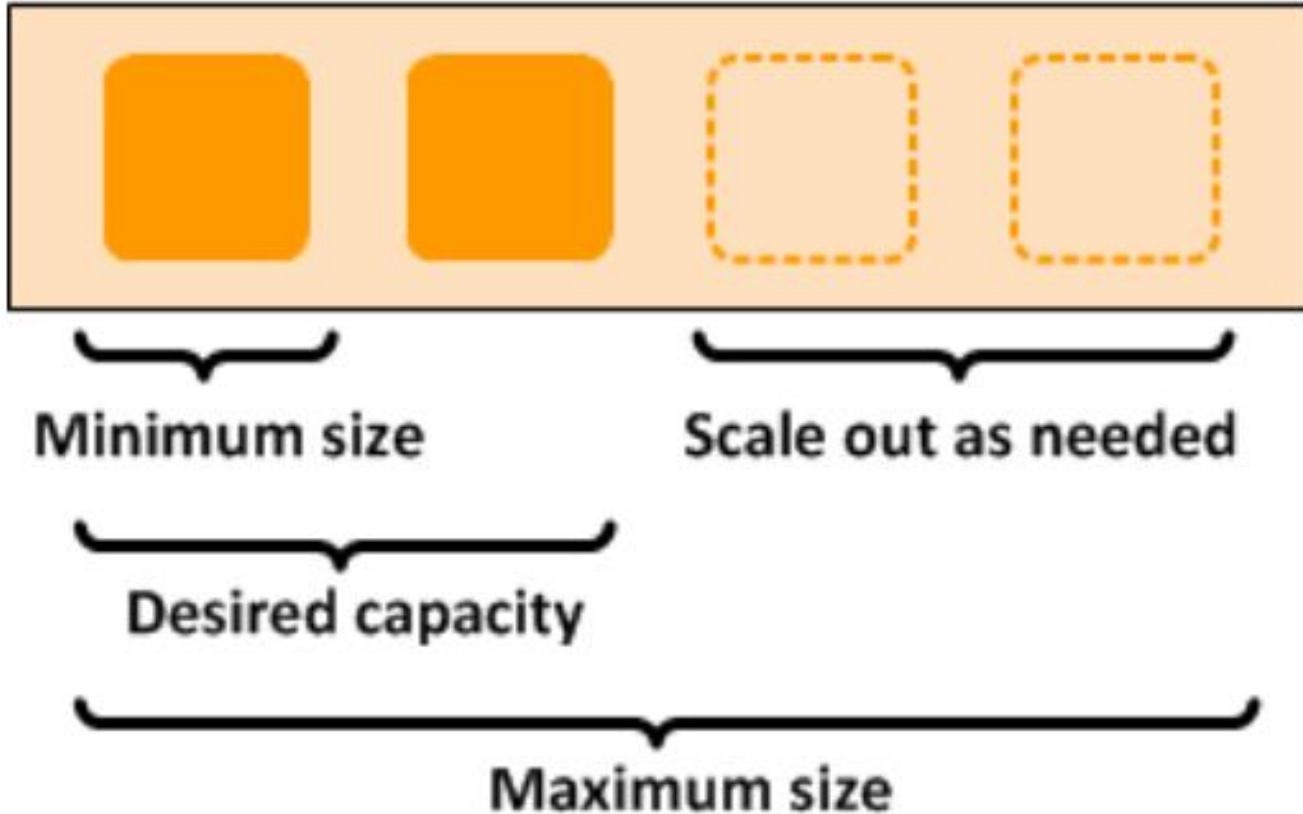
---

- Amazon Auto Scaling Group
- 구성 템플릿 (configuration template)
- 조정 옵션

## 구성 요소 (2)

- Amazon Auto Scaling Group
  - 논리적 단위로 처리되는 EC2 인스턴스의 모음
  - 인스턴스의 조정 및 관리 목적으로 구성된 논리적 그룹
  - 인스턴스의 수를 조건에 따라 자동 조정 및 관리하는 auto scaling의 핵심 기능
  - 그룹의 최소, 최대 및 원하는 용량 정의

## Auto Scaling group



## 구성 요소 (3)

- 구성 템플릿 - 시작 구성
  - 인스턴스를 시작하는 데 사용하는 템플릿
  - auto scaling group이 생성되기 위해서 EC2 instance를 어떻게 만들 것인가를 설정
  - AMI, instance type, key pair, 보안 그룹, EBS 등 인스턴스에 대한 정보를 지정

## 구성 요소 (4)

- 조정 옵션 - auto scaling group 조정
  - 인스턴스의 수를 늘리거나 줄이는 기능
  - 이벤트와 함께 시작 또는 auto scaling group의 조정 작업과 함께 시작
  - 조정 옵션: 그룹 조정 방법/정책
    - 현재 인스턴스 수준 유지
    - 수동 조정 / 일정 기반 조정
    - 온디맨드 기반 조정 - 사용률에 따른 조정



# 수명 주기 (1)

- auto scaling group이 instance를 시작하고 서비스에 들어갈 때, 수명 시작
- instance 종료, auto scaling group에서 instance를 제외시키고 종료할 때, 수명 종료

## 수명 주기 (2)

- 확장
  - EC2 instance를 시작하고, 그룹에 연결하라고 지시
  - 수동 조정, 조정 정책에 따른 동적 조정, 특정 시간에 조정하는 예약/일정 조정
- 축소
  - 수동 조정, 조정 정책에 따른 동적 조정, 특정 시간에 조정하는 예약/일정 조정

# 관련 서비스

- Amazon EC2 - 클라우드에서 가상 머신 생성/실행
- Amazon CloudWatch
  - 조정 정책 활성화
  - Auto Scaling 그룹과 EC2 인스턴스에 대한 지표 모니터링
- Elastic Load Balancing
  - 수신되는 application traffic을 AS group의 instance에 자동으로 분산시킴

# AWS auto scaling

---

- 인프라의 증설/축소를 손쉽게 구현, 확장성 및 탄력성 높은 시스템을 구축할 수 있음
- 서버나 애플리케이션을 모니터링하고 리소스를 자동으로 조정 (scale in/out)
- 최대한 저렴한 비용으로 안정적이고 예측 가능한 성능 유지

# EC2 auto scaling의 동적 조정

---

- 애플리케이션 수요 곡선에 따름
- 애플리케이션 로드 지표 선택
- 조건부 또는 일정 예약으로 설정
- CloudWatch로 사용 (선택 사양)

# EC2 auto scaling을 사용한 플릿 관리

---

- 중단 없이 손상된 EC2 instance 교체
- 실행 중인 인스턴스의 상태 모니터링
- 손상된 인스턴스 자동 교체
- 여러 AZ에서 용량 밸런싱

# Source



- <https://docs.aws.amazon.com/ko-kr/autoscaling/ec2/userguide/what-is-amazon-ec2-auto-scaling.html>

# Elastic Block Store - practice

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



# Contents

- ▶ EBS volume 생성
- ▶ 볼륨을 EC2 인스턴스에 연결하고 탑재
- ▶ 사용자 볼륨의 스냅샷 생성
- ▶ 스냅샷에서 새 볼륨 생성
- ▶ 새 볼륨을 EC2 인스턴스에 연결하고 탑재

## EBS Volume의 기능

- ▶ **영구 스토리지:** 볼륨 수명은 특정 Amazon EC2 인스턴스와 독립적
- ▶ **범용:** Amazon EBS 볼륨은 모든 운영 체제에서 사용할 수 있는 형식이 지정되지 않은 원시 블록 디바이스
- ▶ **고성능:** Amazon EBS 볼륨은 로컬 Amazon EC2 드라이브와 같거나 더 우수한 성능 제공
- ▶ **높은 안정성:** Amazon EBS 볼륨은 가용 영역 내에서 기본적으로 이중화 제공

## EBS Volume의 기능

- ▶ **뛰어난 복원력:** Amazon EBS의 AFR(연간 실패율)은 0.1%에서 1% 사이
- ▶ **가변 크기:** 볼륨 크기는 1GB에서 16TB
- ▶ **사용 편의성:** Amazon EBS 볼륨을 쉽게 생성하고 연결, 백업, 복원 및 삭제할 수 있음



# 1. EBS 볼륨 생성

- ▶ EC2 → instance (새로 만든 instance, **lab**)
- ▶ 가용 영역 확인
- ▶ Elastic Block Store → 볼륨
- ▶ 볼륨 생성
  - ▶ type: general purpose SSD (gp2)
  - ▶ size (GB): 1
  - ▶ 가용 영역
  - ▶ Tag: Name = **my volume**

## 2. 볼륨을 인스턴스에 연결

- ▶ 볼륨에서 **my volume** 선택
- ▶ 작업 → 볼륨 연결
  - ▶ lab 인스턴스 선택
  - ▶ 디바이스 이름 확인: `/dev/sdf`
- ▶ EC2 instance에서 [스토리지] 탭 확인

### 3. 파일 시스템 생성 및 구성

- ▶ EC2 접속 후,
- ▶ `df -h`
  - ▶ `/dev/xvda1`: 8GB 디스크 볼륨
  - ▶ 새 볼륨은 보이지 않음
- ▶ 새 볼륨에 ext3 파일 시스템 생성: `sudo mkfs -t ext3 /dev/sdf`
- ▶ 새 스토리지 볼륨을 탑재할 디렉토리 생성: `sudo mkdir /mnt/data-store`

### 3. 파일 시스템 생성 및 구성

- ▶ 새 볼륨 탑재: `sudo mount /dev/sdf /mnt/data-store`
- ▶ 인스턴스가 시작될 때마다 이 볼륨을 탑재하도록 Linux 인스턴스 구성
  - ▶ `echo “/dev/sdf /mnt/data-store ext3 defaults,noatime 1 2” | sudo tee -a /etc/fstab`
- ▶ 구성의 마지막 행 설정 확인: `cat /etc/fstab`
- ▶ `df -h`



### 3. 파일 시스템 생성 및 구성

- ▶ 탑재된 볼륨에 파일 생성

- ▶ `sudo sh -c "echo some text has been written > /mnt/data-store/file.txt"`

## 4. EBS 스냅샷 생성

- ▶ 볼륨 → **my volume** 선택
- ▶ 작업 → 스냅샷 생성
  - ▶ Tag: Name = **my snapshot**
- ▶ 스냅샷 → 확인
- ▶ file.txt 삭제: `sudo rm /mnt/data-store/file.txt`

## 5. EBS 스냅샷 복원

- ▶ 스냅샷을 이용한 볼륨 생성
- ▶ 스냅샷 → **my snapshot** 선택
- ▶ 작업 → 볼륨 생성
  - ▶ 동일 가용 영역
  - ▶ Tag: Name = **restored volume**
- ▶ 볼륨 선택 → 작업 → 볼륨 연결
  - ▶ lab 인스턴스
  - ▶ 디바이스 이름: `/dev/sdg`

## 5. EBS 스냅샷 복원

- ▶ 복원된 볼륨 탑재
  - ▶ 디렉토리 생성: `sudo mkdir /mnt/data-store2`
  - ▶ 볼륨 탑재: `sudo mount /dev/sdg /mnt/data-store2`
  - ▶ 확인: `ls /mnt/data-store2`

# VPC - practice

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

# Contents

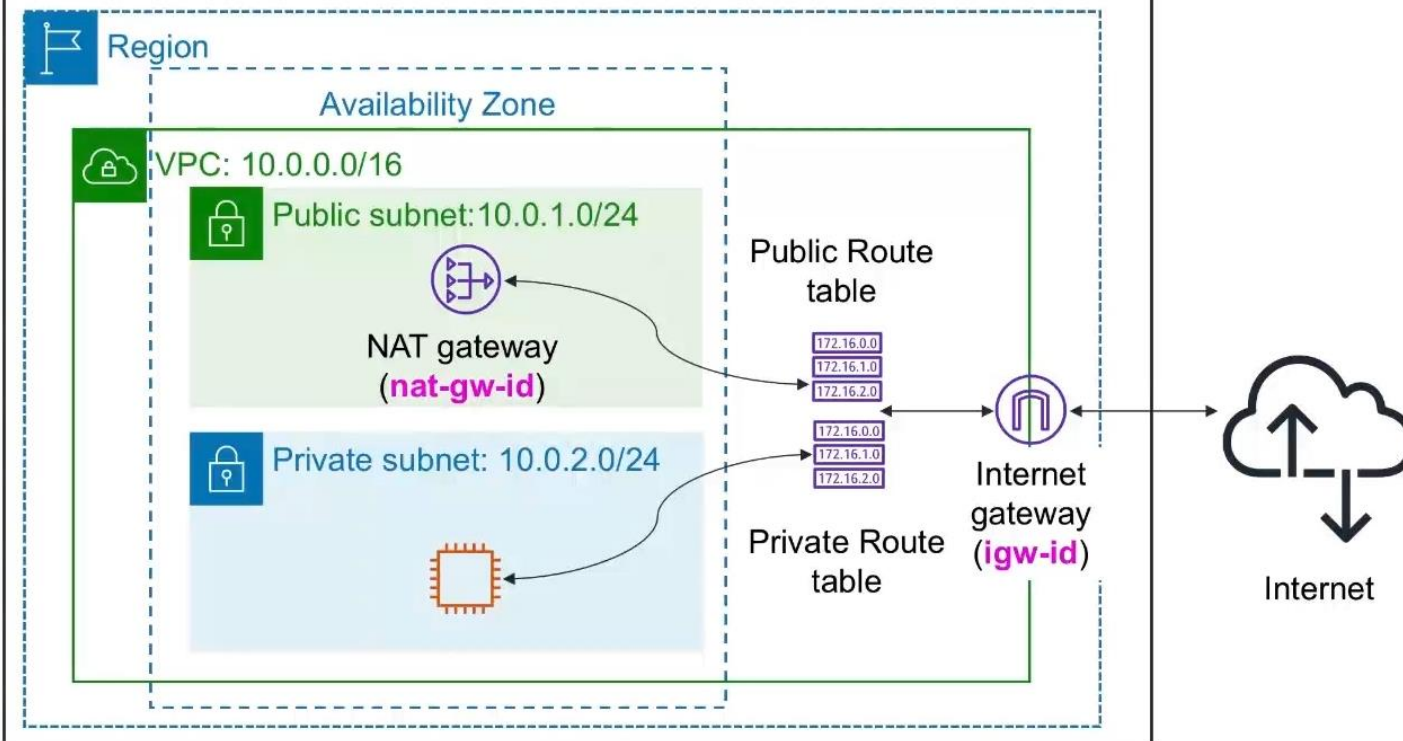
- ▶ VPC 및 VPC 관련 요소 검토
- ▶ VPC 생성 실습
  - ▶ Internet gateway 연결
  - ▶ Subnet 추가
  - ▶ Routing Table 정의: IG와 subnet간 traffic flow

# NAT Gateway

- ▶ VPC 마법사 → NAT gateway를 시작함
  - ▶ private subnet에 구축되는 private resource에 대한 internet access 제공
  - ▶ 공용 인터넷 (public internet)에 접속함
  - ▶ EIP가 할당됨

# Amazon VPC Demo

aws AWS Cloud



Public subnet route table

Destination	Target
10.0.0.0/16	local
0.0.0.0/0	igw-id

Private subnet route table

Destination	Target
10.0.0.0/16	local
0.0.0.0/0	nat-gw-id



## Step 2: VPC with Public and Private Subnets

**IPv4 CIDR block:**  (65531 IP addresses available)

**IPv6 CIDR block:** ☒ No IPv6 CIDR Block  
☐ Amazon provided IPv6 CIDR block  
☐ IPv6 CIDR block owned by me

**VPC name:**

**Public subnet's IPv4 CIDR:**  (251 IP addresses available)

**Availability Zone:**

**Public subnet name:**

**Private subnet's IPv4 CIDR:**  (251 IP addresses available)

**Availability Zone:**

**Private subnet name:**

You can add more subnets after AWS creates the VPC.

Specify the details of your NAT gateway ([NAT gateway rates apply](#)).

**Elastic IP Allocation ID:**

### Service endpoints

**Enable DNS hostnames:** ☒ Yes ☐ No

**Hardware tenancy:**

# Steps

- ▶ Step 1: EIP 생성 (고정 IP) → NAT gateway에 할당됨
- ▶ Step 2: VPC 생성 (VPC 마법사)
  - ▶ VPC 대시보드 → VPC 생성 → VPC등 = **my-vpc**

## 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 이름 태그를 자동으로 생성하는 데 사용됩니다.

☒ 자동 생성

my

### IPv4 CIDR 블록 정보

CIDR 표기법을 사용하여 VPC의 시작 IP와 크기를 결정합니다.

10.0.0.0/16

65,536 IPs

### 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

### 퍼블릭 서브넷 수 정보

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

0

1

### 프라이빗 서브넷 수 정보

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

0

1

2

### ▼ 서브넷 CIDR 블록 사용자 지정

#### ap-northeast-2a 퍼블릭 서브넷 CIDR 블록

10.0.1.0/24

256 IPs

#### ap-northeast-2a 프라이빗 서브넷 CIDR 블록

10.0.2.0/24

256 IPs

### 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

### VPC 세부 정보 표시

AWS 가상 네트워크

my-vpc

### 서브넷(2개)

이 VPC 내의 서브넷

#### ap-northeast-2a

my-subnet-public1-ap-northeast-2a

my-subnet-private1-ap-northeast-2a

### 라우팅 테이블(2개)








네트워크 트래픽을 리소스로 라우팅

my-rtb-public

my-rtb-private1-ap-northeast-2a

✔ 성공

▼ 세부 정보

- ✔ VPC 생성: [vpc-0c698debd965bb9f6](#) 
- ✔ Enable DNS hostnames
- ✔ Enable DNS resolution
- ✔ Verifying VPC creation: [vpc-0c698debd965bb9f6](#) 
- ✔ Create subnet: [subnet-0460d2104499d951b](#) 
- ✔ Create subnet: [subnet-06fcfba83a3a1377e](#) 
- ✔ Create internet gateway: [igw-07111c1173792a411](#) 
- ✔ Attach internet gateway to the VPC
- ✔ Create route table: [rtb-06676e4de059b6370](#) 
- ✔ Create route
- ✔ Associate route table
- ✔ Allocate elastic IP: [eipalloc-0459e52fb494705aa](#) 
- ✔ Create NAT gateway: [nat-010f32ed804996b53](#) 
- ✔ Wait NAT Gateways to activate
- ✔ Create route table: [rtb-0a5a3b052b606ac2f](#) 
- ✔ Create route
- ✔ Associate route table
- ✔ Verifying route table creation

# 확인

- ▶ 인터넷 게이트웨이 = my-igw
- ▶ public/private 서브넷

<input checked="" type="checkbox"/>	Name ▾	서브... ▾	상태 ▾	VPC ▾	IPv4 CI... ▾
<input checked="" type="checkbox"/>	my-subnet-public1-a...	subnet...	✔ Available	vpc-0c698debd965bb9f6   my-vpc	10.0.1.0/24
<input checked="" type="checkbox"/>	my-subnet-private1-...	subnet...	✔ Available	vpc-0c698debd965bb9f6   my-vpc	10.0.2.0/24

# 확인

- ▶ 사용 가능한 주소: 250
- ▶ 라우팅 테이블
  - ▶ local: VPC 내부 다른 위치로 향하는 traffic
  - ▶ 공용 인터넷: 0.0.0.0/0 → IG
- ▶ why public: IG 경로가 포함된 routing table에 연결되어 있음
  - ▶ IG를 통해 외부와 연결됨

## 라우팅 테이블: [rtb-06676e4de059b6370](#) / [my-rtb-public](#)

### 라우팅 (2)

🔍 라우팅 필터링

대상	대상
10.0.0.0/16	local
0.0.0.0/0	<a href="#">igw-07111c1173792a411</a>



# 확인

## ▶ Network ACLs

- ▶ subnet 안팎으로 전송되는 traffic을 제어하는 방화벽 역할을 하는 VPC에 대한 선택적 보안 계층
- ▶ 기본: 모든 트래픽 허용 (wide open)
- ▶ inbound / outbound rules
- ▶ 보안그룹: resource에 대한 추가적인 방화벽 역할

# 확인

- ▶ private subnet
  - ▶ 사용 가능한 주소: 251
  - ▶ routing table
    - ▶ local: VPC 내부에서의 트래픽을 처리하는 local 경로
    - ▶ NAT
      - ▶ private에서 public internet으로의 단방향 연결 정의
      - ▶ private subnet 내의 resource에 대한 patch, update traffic 제공

## 라우팅 테이블: [rtb-0a5a3b052b606ac2f / my-rtb-private1-ap-northeast-2a](#)

### 라우팅 (2)

🔍 라우팅 필터링

대상	대상
10.0.0.0/16	local
0.0.0.0/0	<a href="#">nat-010f32ed804996b53</a>

# 정리

- ▶ NAT 삭제
- ▶ EIP 릴리스
- ▶ 서브넷 삭제 (public, private)
- ▶ 인터넷게이트웨이 → 작업 → VPC에서 분리 → 삭제
- ▶ VPC 삭제
  - ▶ routing table 함께 자동 삭제됨

# AWS Practice - CloudFront

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

# Contents

- ▶ CloudFormation을 이용한 콘텐츠 전송
- ▶ CloudFront를 이용한 콘텐츠 전송

# CloudFormation을 이용한 콘텐츠 전송

# CloudFormation을 이용한 콘텐츠 전송

- ▶ AWS console → CloudFormation
  - ▶ **us-east-1 region** (버지니아 북부)
- ▶ [스택 생성] → 새 리소스 사용 (표준)
- ▶ 1 단계: 준비된 템플릿 -> Amazon S3 URL
  - ▶ [https://s3-eu-west-1.amazonaws.com/tomash-public/AWS/s3bucket\\_with\\_cloudfront.yml](https://s3-eu-west-1.amazonaws.com/tomash-public/AWS/s3bucket_with_cloudfront.yml)
- ▶ 스택 생성: 4단계까지 계속 default 적용



# CloudFormation을 이용한 콘텐츠 전송

- ▶ CloudFormation 스택 - 출력(output) 탭
  - ▶ S3 bucket 생성: 콘텐츠 upload → S3 bucket 이름 (S3BucketName) 확인
  - ▶ CloudFront 웹 배포 생성 → CloudFront 배포 도메인 이름 (CfDistributionDomainName) 확인

## 출력 (3)



 출력 검색


키



값



설명



내보내기 이름

CfDistributionDo  
mainNamed3e34j57fvsuq7.  
cloudfront.netDomain name  
for our  
cloudfront  
distribution

-

CfDistributionId

E2XGH5XXH9G0  
F5Id for our  
cloudfront  
distribution

-

S3BucketName

cf-simple-s3-  
origin-  
cloudfrontforksd  
s3-v1-  
352485268054

Bucket name

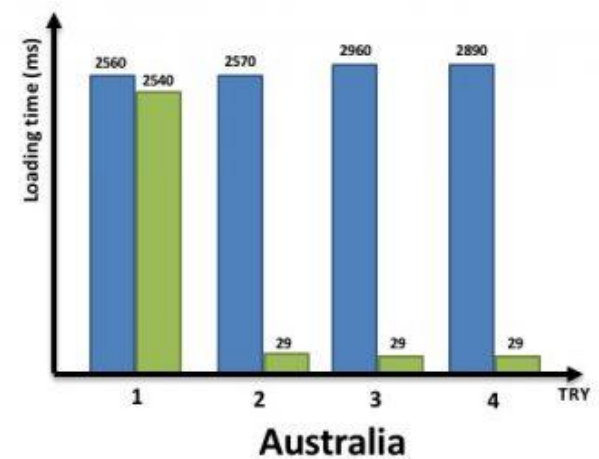
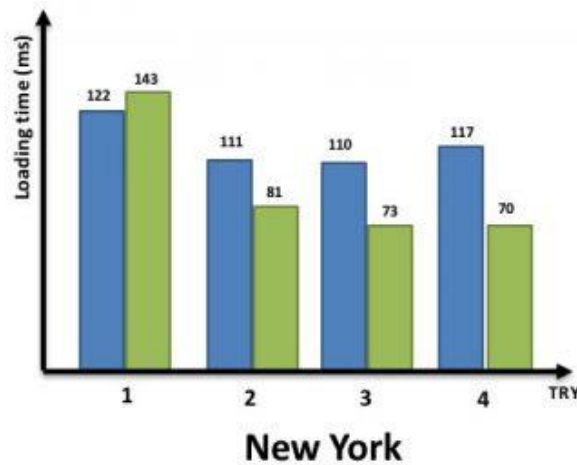
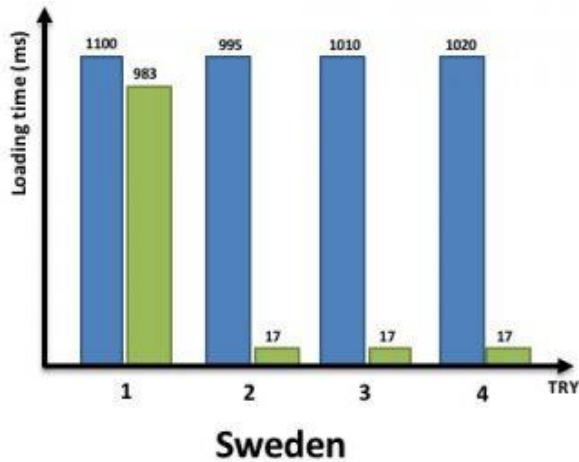
-

# CloudFormation을 이용한 콘텐츠 전송

- ▶ 정적 파일을 CloudFormation으로 배포하기
  - ▶ S3에 파일 upload: index.html, balloon.png, error.html
- ▶ S3에서 파일 url을 이용한 파일 접근 시도 → 실패
  - ▶ CloudFront, CloudFormation의 OAI 기능으로 S3 직접 접근이 안됨
- ▶ CloudFront 배포 도메인 이름으로 접근

# 배포 성능

- ▶ 로드 시간 테스트: 버킷은 버지니아 북부 리전에 존재



- Accessing image using standard S3 endpoint
- Accessing image using CloudFront edge global network

# CloudFront의 장점

- ▶ 콘텐츠를 빠르게 사용자에게 전송하여 애플리케이션의 성능 향상
- ▶ 애플리케이션의 보안 강화
- ▶ 비용 절감: CloudFront에서 전송하는 것이 S3에서 전송하는 것보다 저렴

## source

- ▶ <https://aws.amazon.com/ko/blogs/korea/amazon-s3-amazon-cloudfront-a-match-made-in-the-cloud/>

# CloudFront를 이용한 콘텐츠 전송

# CloudFront를 이용한 콘텐츠 전송 실습

- ▶ S3 bucket 생성
  - ▶ **public access** 가능하도록
  - ▶ 파일 upload (index.html + image file)
- ▶ CloudFront → [CloudFront 배포 생성]
  - ▶ 원본 도메인: 사용자가 요청할 때 콘텐츠를 가져올 오리진 세팅 (S3, http server, ...)
    - ▶ S3 bucket 선택 → create distribution 버튼
  - ▶ [배포 생성]



# CloudFront를 이용한 콘텐츠 전송 실습

## ▶ 링크 테스트

- ▶ 배포 선택 → [General] 탭
- ▶ Domain Name 확인 → url을 브라우저에 → 에러
- ▶ url/index.html 식으로 '/' 다음에 파일 이름 명시
- ▶ 오류 발생시,
  - ▶ 버킷의 권한 조정: 정책 생성

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Statement1",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::352485268054:root"
      },
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::ksd-cf-bk/*"
    }
  ]
}
```

# CloudFront를 이용한 콘텐츠 전송 실습

## ▶ 기타

- ▶ local에 있는 index.html에서 s3 bucket의 그림을 보여주도록 변경 후 index.html을 확인 → s3에 있는 contents를 access할 수 있음을 확인

## source

▶ <https://aws.amazon.com/ko/getting-started/tutorials/deliver-content-faster/>