

DevOps 구축 BOOTCAMP



2주차 목표

Web Server에 웹 프로젝트(HTML, CSS, JS) 배포하기

AWS ELB(Load balancer), Auto scaling group을 통한 다중 웹서버 환경 구축

운영 서버 아키텍처 미리보기

Git을 활용한 코드 배포

```
1. ec2-user@ip-172-31-25-131:~ (bash)
a1@1ui-MacBook-Air:~$

sudo ssh -i ~/Desktop/ec2_test.pem ec2-user@{접속할 인스턴스 DNS 주소} ↵

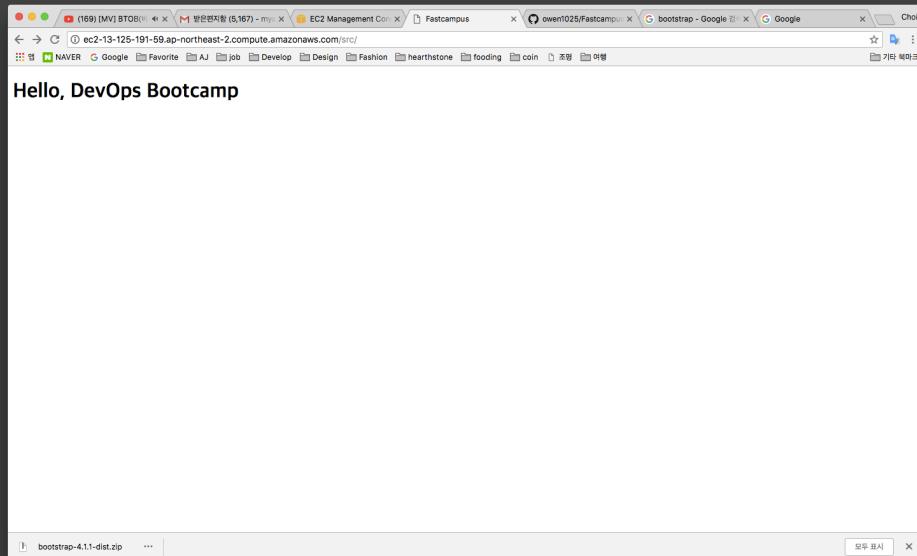
sudo su ↵

yum install -y git ↵

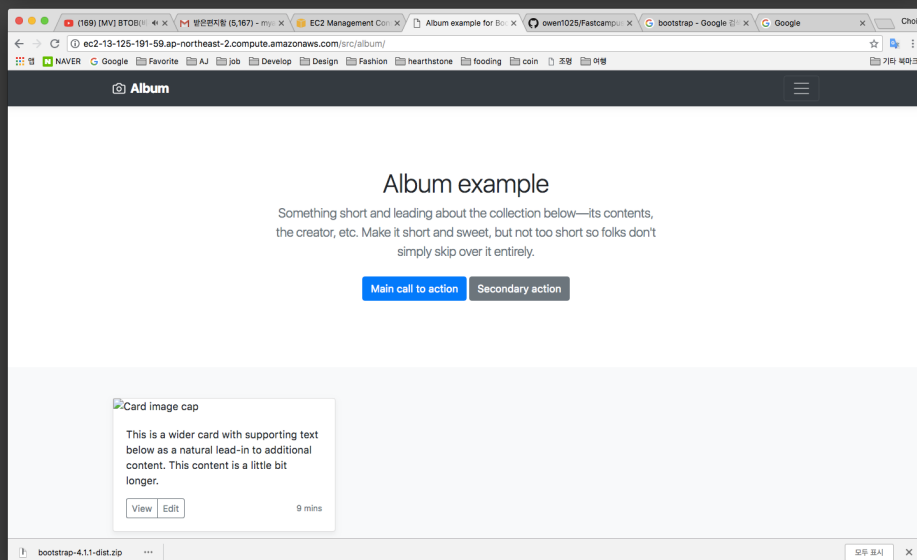
cd /usr/share/nginx/html ↵

git clone https://github.com/owen1025/Fastcampus-web-deploy.git ↵
```

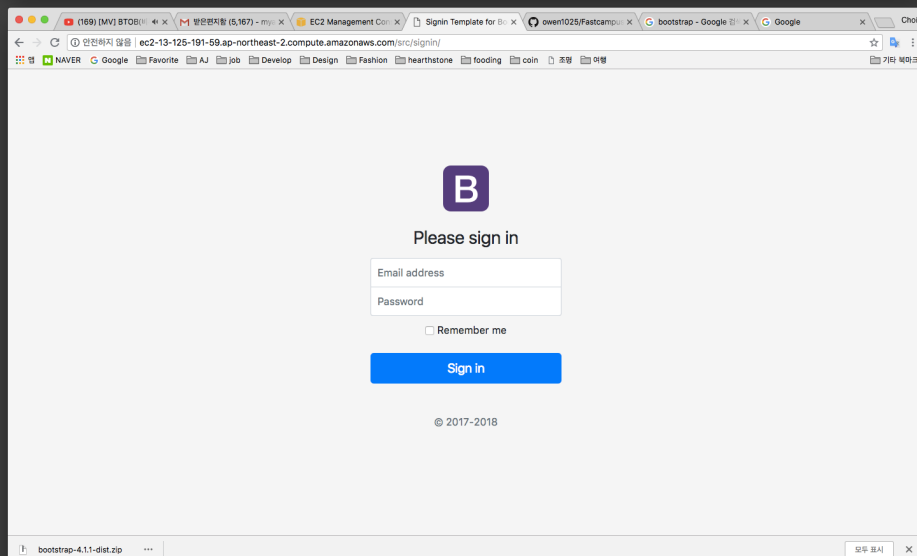
Git을 활용한 코드 배포



<http://ec2-domain/Fastcampus-web-deploy/>
- 해당 EC2 퍼블릭 DNS



<http://ec2-domain/Fastcampus-web-deploy/page/album>



<http://ec2-domain/Fastcampus-web-deploy/page/signin>

Git을 활용한 코드 배포

1. ec2-user@ip-172-31-25-131:~ (bash)

a1@1ui-MacBook-Air:~\$

1) `sudo ssh -i ~/Desktop/ec2_test.pem ec2-user@{접속할 인스턴스 DNS 주소}` ↵

2) `sudo su` ↵

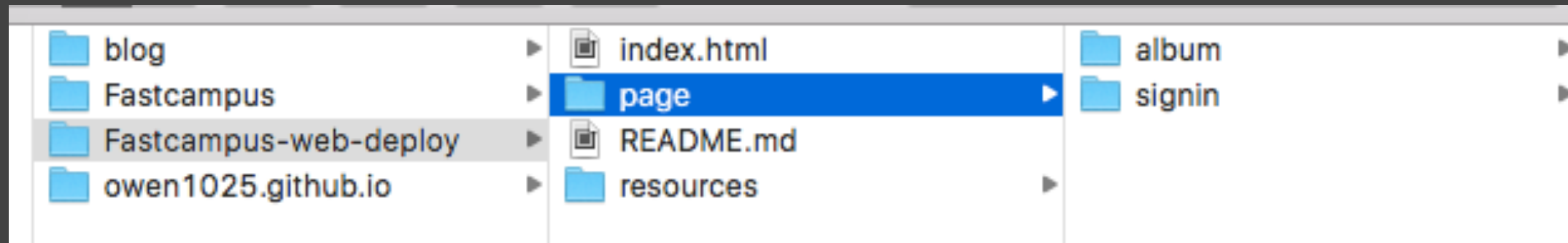
3) `yum install -y git` ↵

4) `cd /usr/share/nginx/html` ↵

5) `git clone https://github.com/owen1025/Fastcampus-web-deploy.git` ↵

1. ec2 인스턴스에 ssh를 통해 접속
2. 관리자 권한으로 실행
3. yum 레포지토리를 통해 git 클라이언트 설치
4. Nginx root directory로 이동
5. git을 통해 Fastcampus-web-deploy 레포지토리 내에 파일 받아오기

Web UI 프로젝트 구조



Fastcampus-web-deploy

- index.html
- page
 - album
 - index.html
 - signin
 - index.html
- resources
 - css
 - js

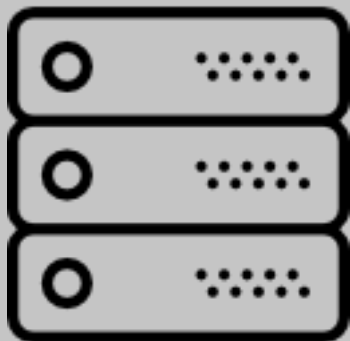
프로젝트 이름

- 메인 페이지 HTML 소스 코드
- 소스코드 디렉토리
 - album 페이지
 - album 페이지 HTML 소스 코드
 - signin 페이지
 - signin 페이지 HTML 소스 코드
- HTML을 제외한 페이지 렌더링을 위한 기타 파일(CSS, JS, image 등)

Nginx HTTP 요청/응답



웹 브라우저, 모바일...

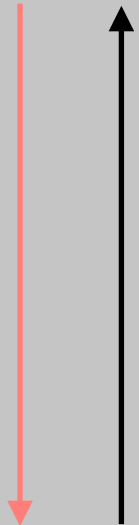


웹 서버(NginX)

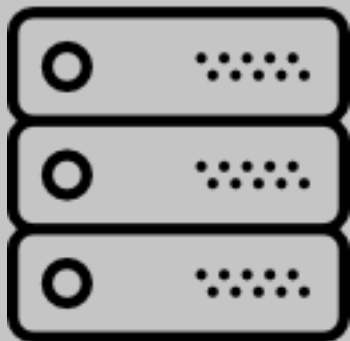
Nginx HTTP 요청/응답



웹 브라우저, 모바일...



<http://ec2-domain/Fastcampus-web-deploy/>

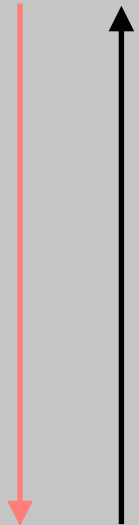


웹 서버(NginX)

Nginx HTTP 요청/응답

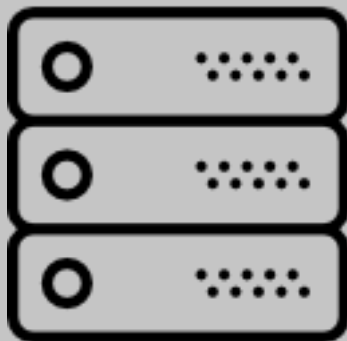


웹 브라우저, 모바일...



`http://ec2-domain:80/Fastcampus-web-deploy/index.html`

- 생략 가능



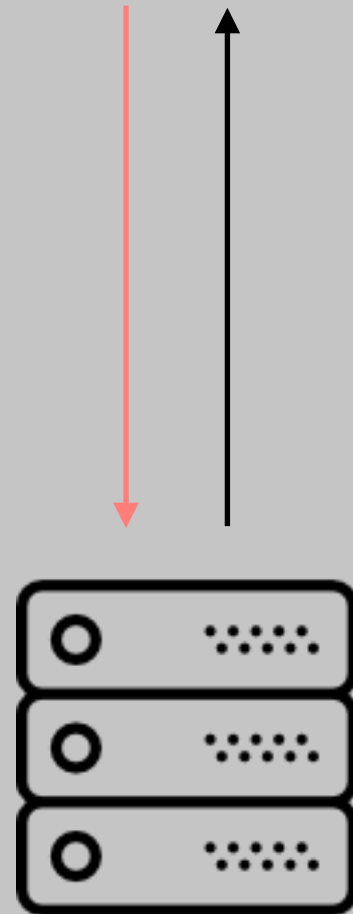
웹 서버(NginX)

Nginx HTTP 요청/응답



웹 브라우저, 모바일...

<http://ec2-domain/Fastcampus-web-deploy>



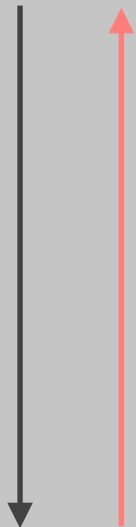
웹 서버(NginX)

Nginx root 디렉토리(/usr/share/nginx/html)에
Fastcampus-web-deploy/index.html이 있는 지 확인

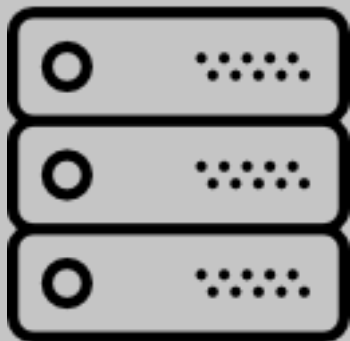
Nginx HTTP 요청/응답



웹 브라우저, 모바일...



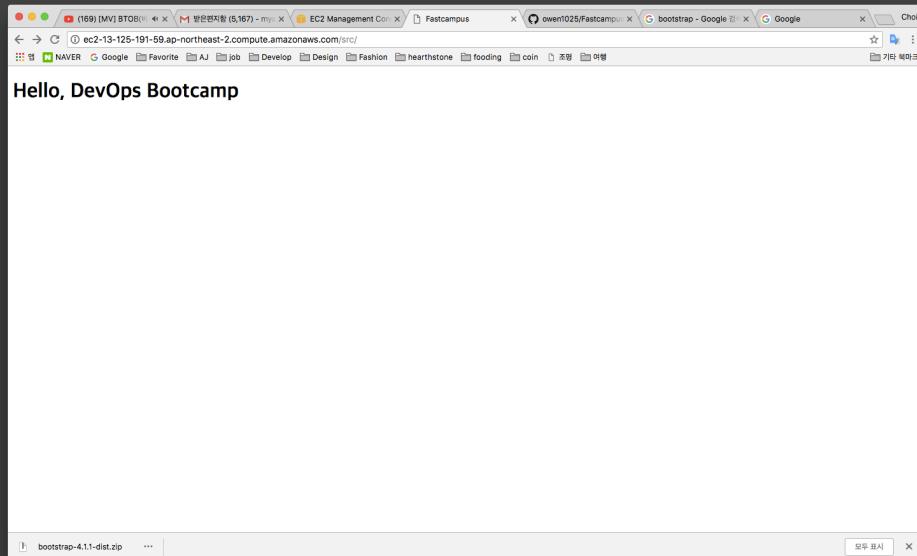
있다면 해당 파일(index.html) 제공
없다면 404 Not found



웹 서버(NginX)

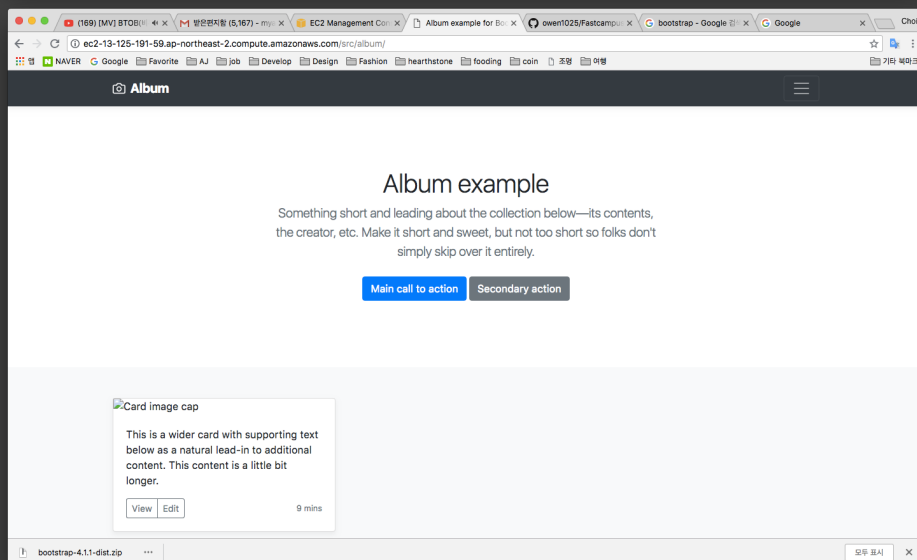
Nginx root 디렉토리(/usr/share/nginx/html)에
Fastcampus-web-deploy/index.html이 있는지 확인

변경해야 할 점



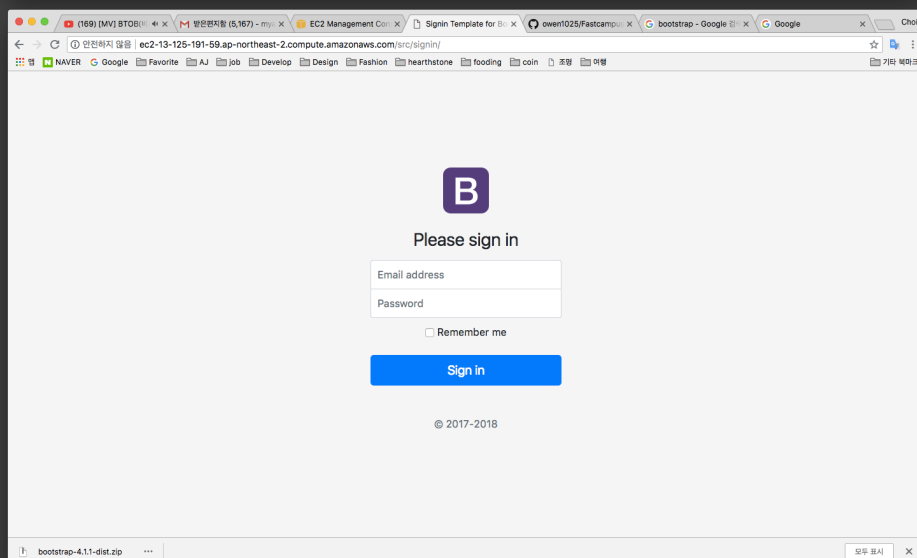
<http://ec2-domain/Fastcampus-web-deploy>

- 해당 EC2 퍼블릭 DNS



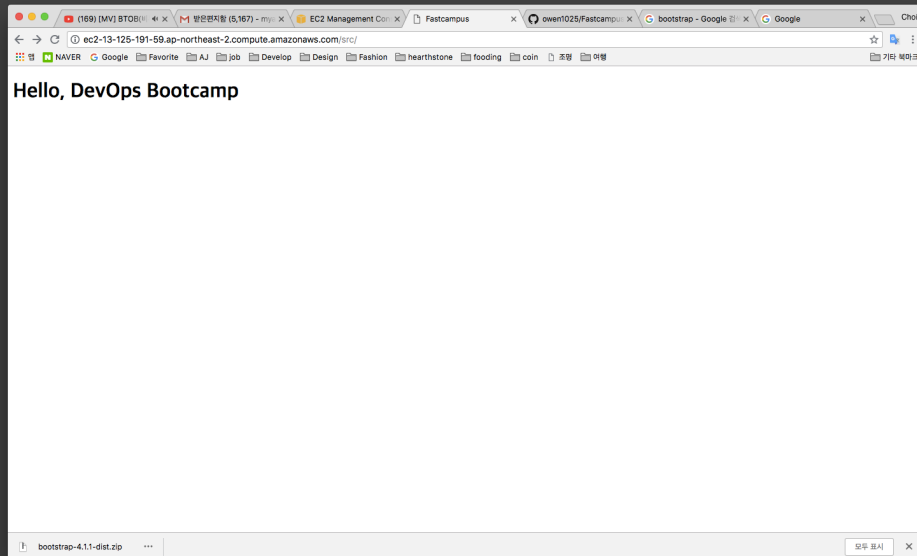
<http://ec2-domain/Fastcampus-web-deploy/page/album>

- 외부에 보여주면 안되는 디렉토리

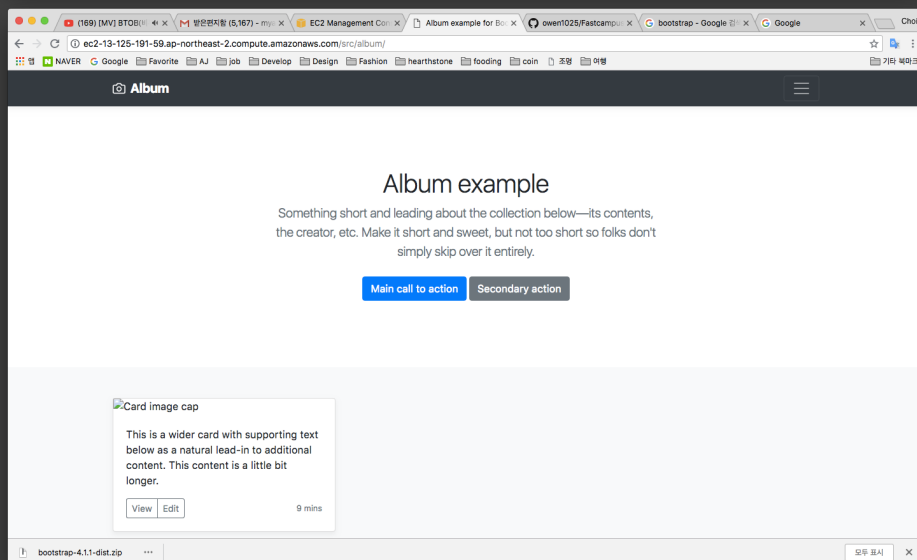


<http://ec2-domain/Fastcampus-web-deploy/page/signin>

변경해야 할 점

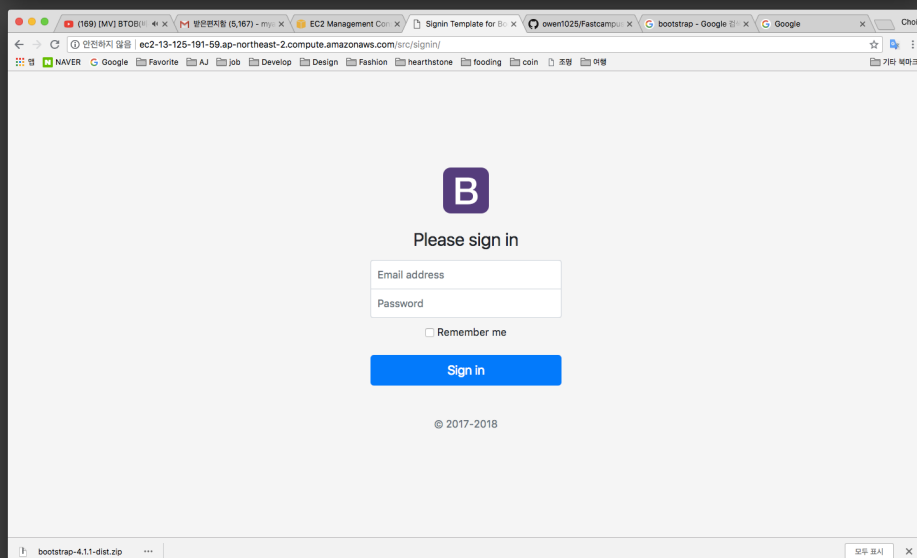


<http://ec2-domain>



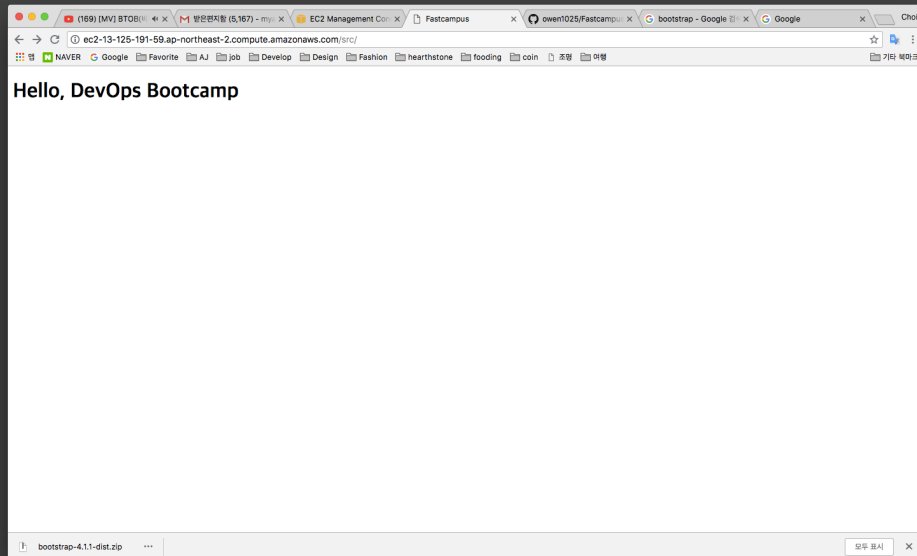
<http://ec2-domain/album>

- Nginx 설정 파일을 수정하여 /Fastcampus-web-deploy/
page 디렉토리 감추기



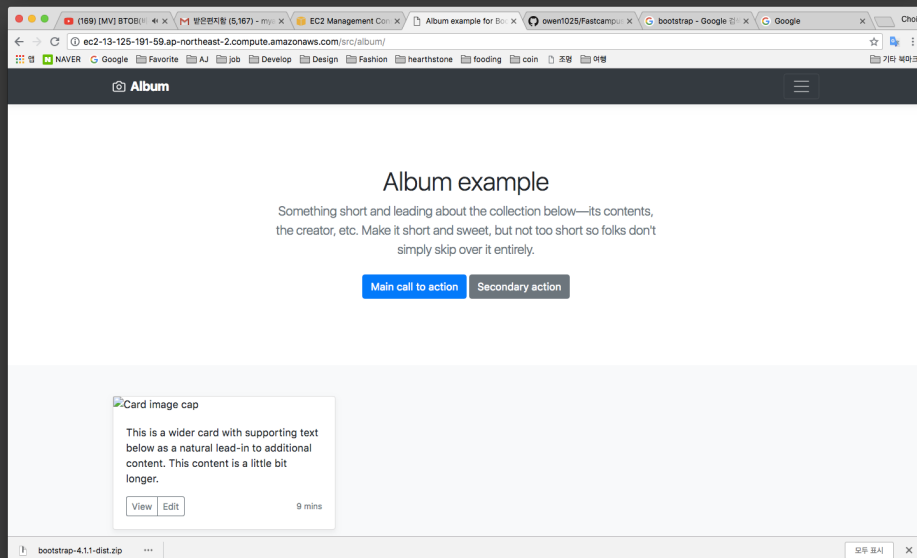
<http://ec2-domain/signin>

변경해야 할 점

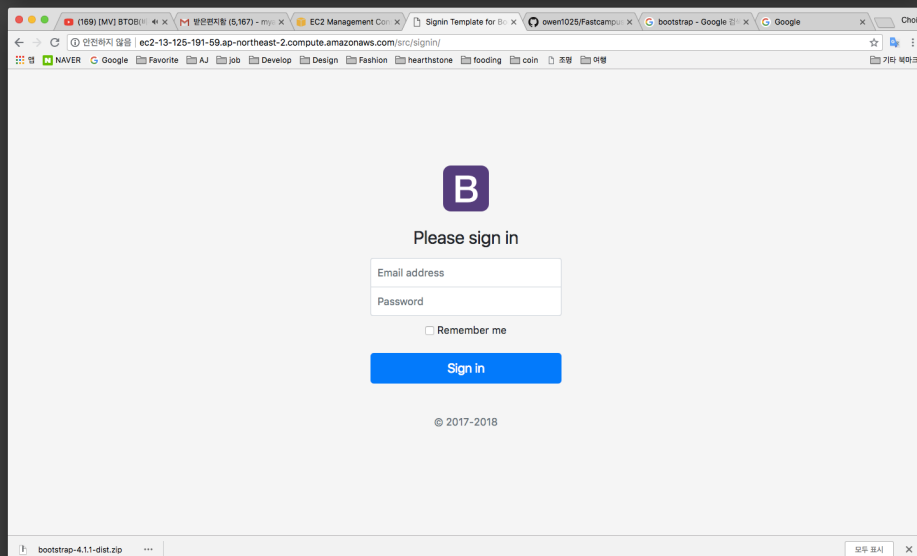


<http://custom-URL>

- **AWS Route53**을 이용하여 서비스 도메인과 EC2 퍼블릭 DNS 매핑



<http://custom-URL/album>

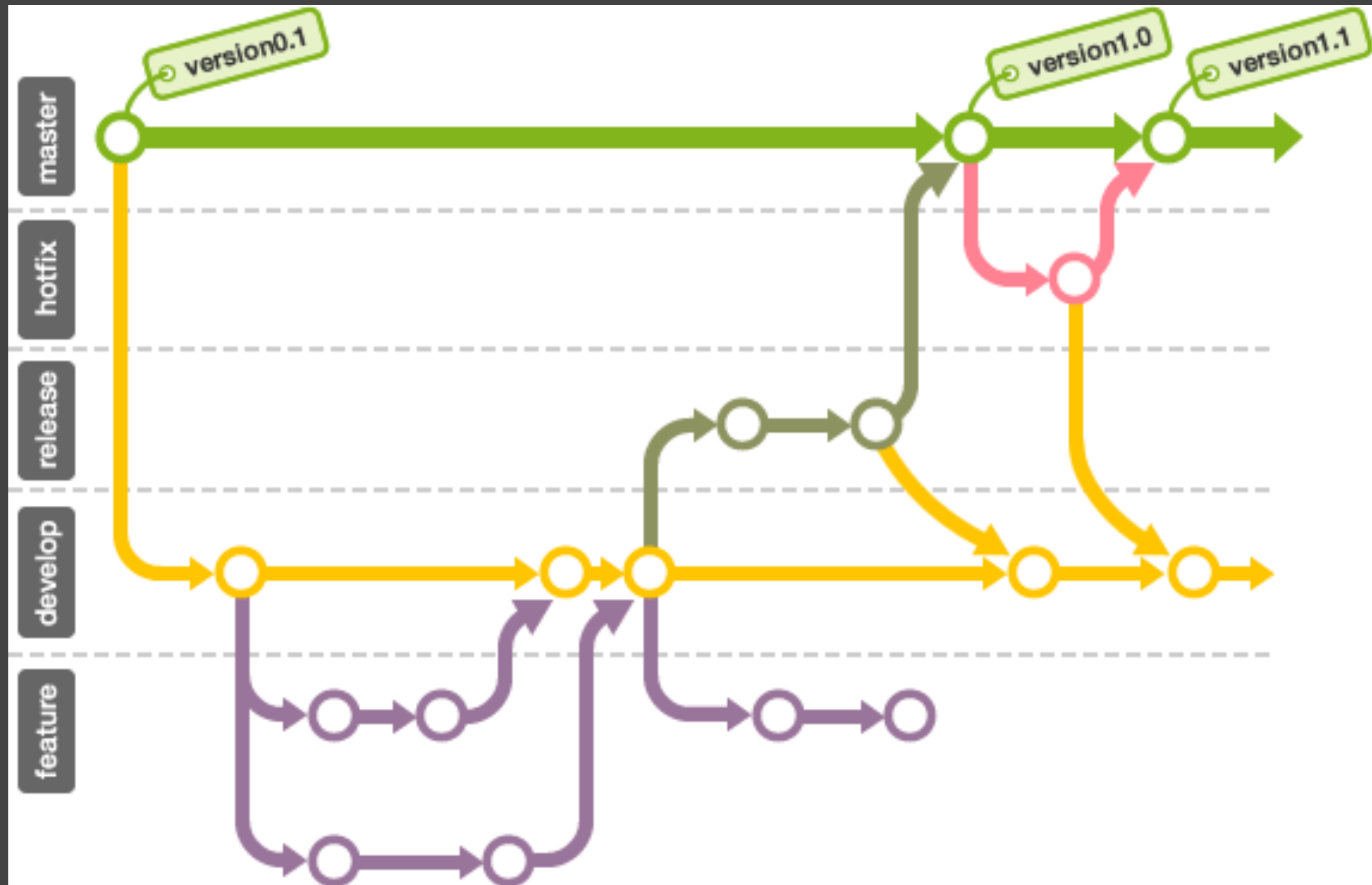


<http://custm-URL/signin>

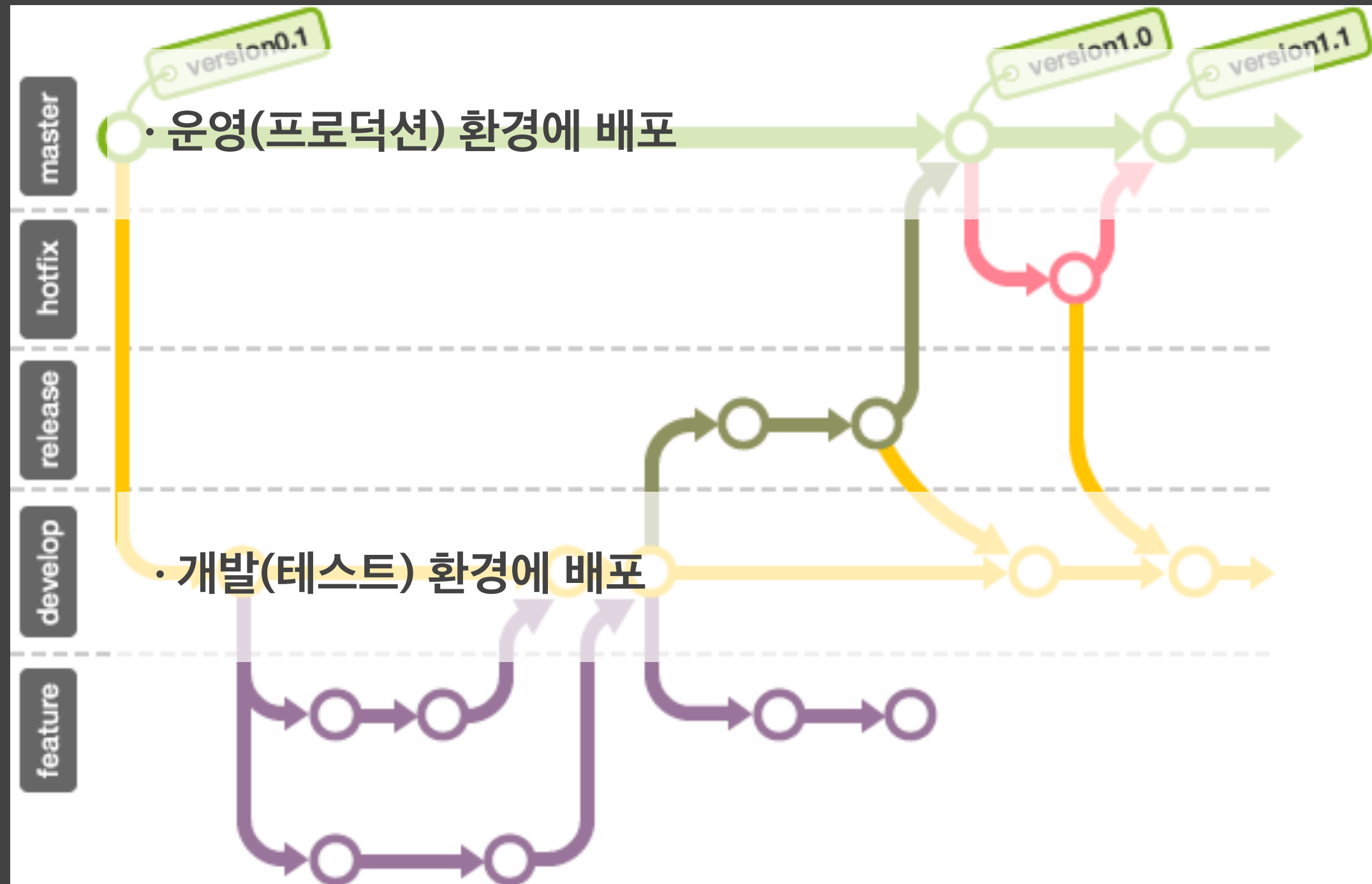
Git

- 소스 코드 관리를 위한 분산형 **버전 관리** 시스템
 - 소스 코드를 버전 별로 관리하며 예전 버전으로 rollback이나 협업을 하며 코드를 합치기 쉬워진다.
 - 브랜치 별로 커밋/머지된 코드를 분리하여 **운영/개발 환경에 각각 배포**할 수 있다.

Git



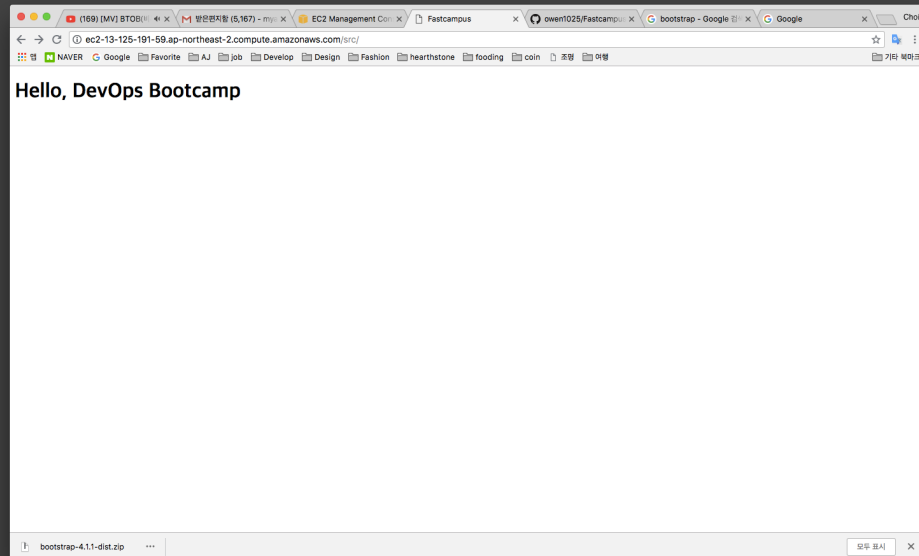
Git



Git

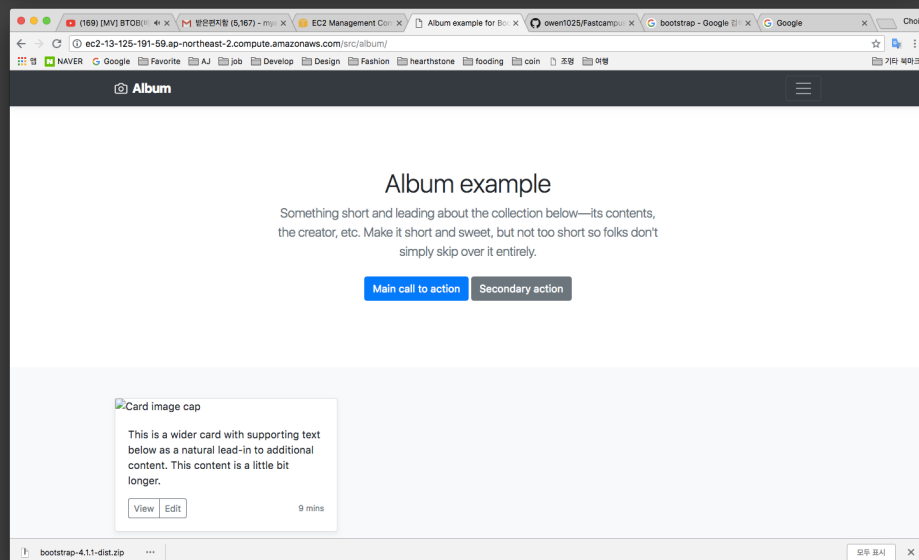
- 저는 이렇게 사용했습니다.
 - develop 브랜치 : 각각 개인 브랜치(ex. module1, 2)의 코드를 합치기 위한 테스트 코드 버전 관리 브랜치. 로컬 환경에서 개인 테스트가 마친 후 커밋 후 관리자가 머지하는 걸 원칙
 - hotfix 브랜치 : 긴급 수정 코드 머지를 위한 브랜치. 관리자가 부재시 해당 모듈/서비스의 담당자가 머지하는 것을 허용
 - stage 브랜치 : 개발(테스트) 환경에 배포를 위한 브랜치. develop/hotfix 브랜치에 머지된 코드가 있을 경우 테스트 환경에 자동 배포
 - production 브랜치 : stage 브랜치에 머지된 코드를 확인, 테스트 환경에서 미리 정의된 테스트 시나리오를 전부 통과하면 관리자에게 Email / Slack으로 알림. 만약 stage 브랜치에 머지된 코드의 출처가 hotfix 브랜치일 경우 알림 없이 운영 환경에 자동 배포. 그 외면 관리자의 허가에 의해 운영 환경에 배포

NginX 설정파일 수정하기



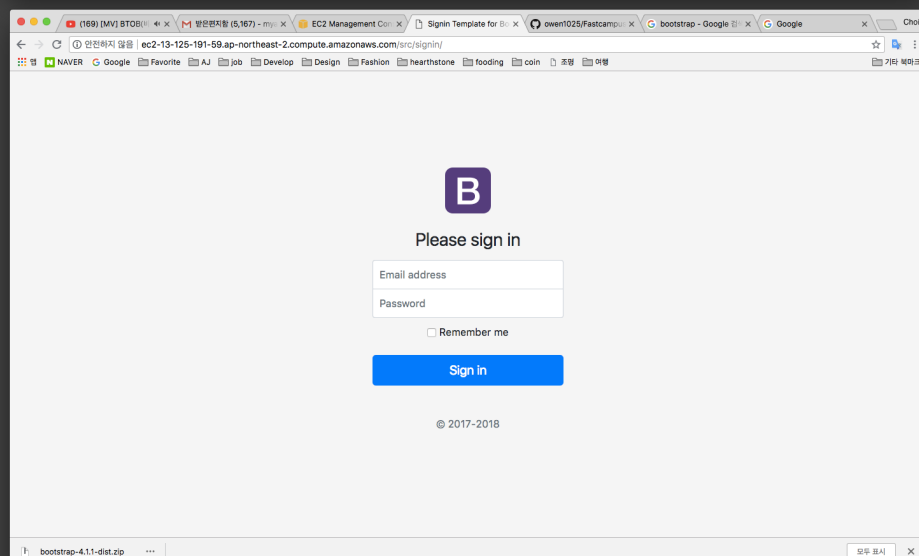
<http://ec2-domain/Fastcampus-web-deploy/>

- 해당 EC2 퍼블릭 DNS



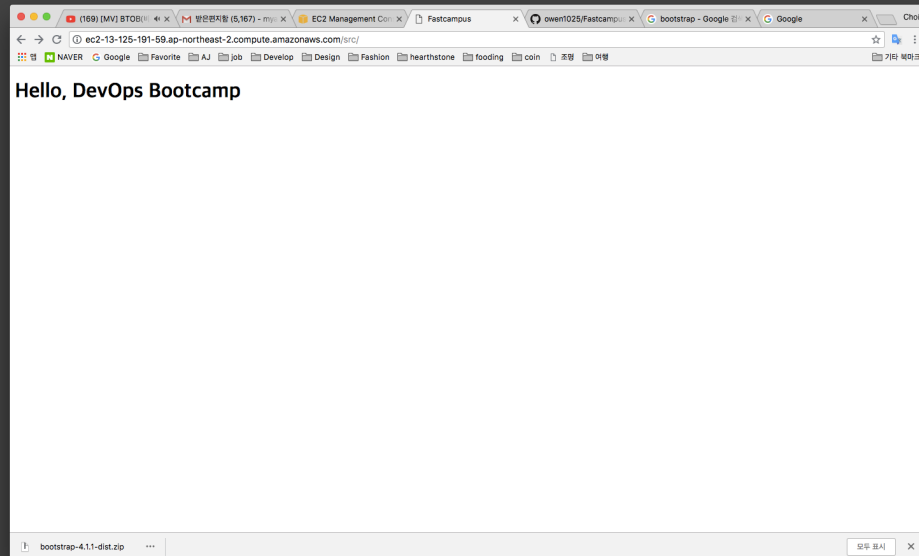
<http://ec2-domain/Fastcampus-web-deploy/page/album>

- 외부에 보여주면 안되는 디렉토리
- 서비스 운영 시 URL이 길어지는 불편함

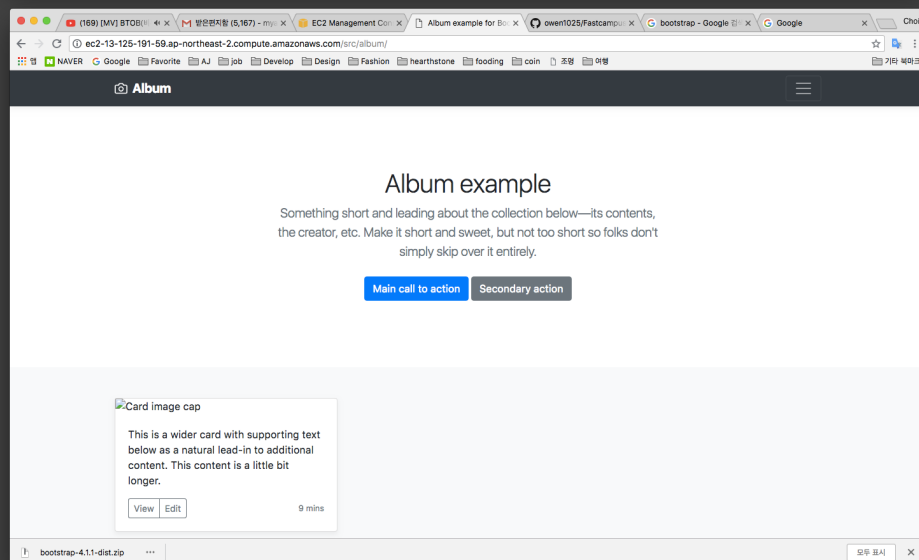


<http://ec2-domain/Fastcampus-web-deploy/page/signin>

NginX 설정파일 수정하기

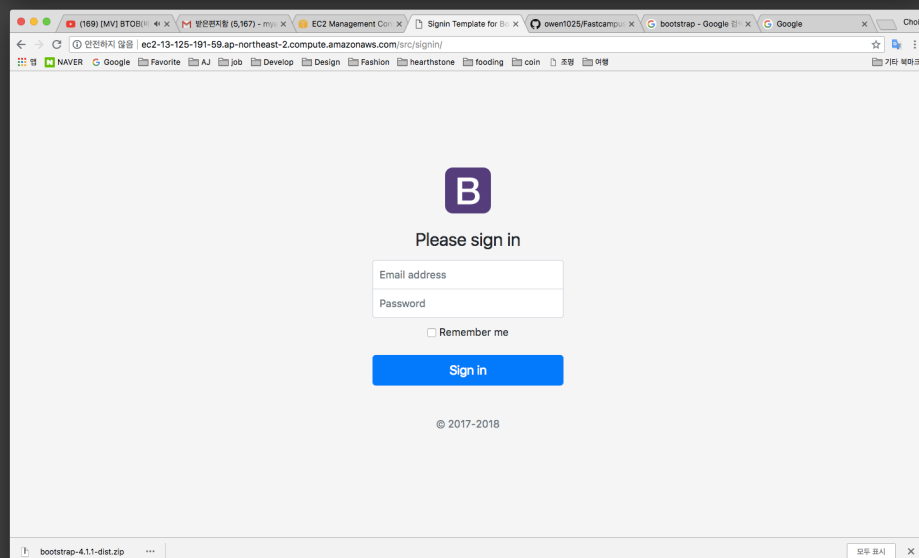


<http://ec2-domain>



<http://ec2-domain/album>

- Nginx 설정 파일을 수정하여 /Fastcampus-web-deploy/
page 디렉토리 감추기



<http://ec2-domain/signin>

NginX 설정파일 수정하기

```
1. ec2-user@ip-172-31-25-131:~ (bash)
a1@1ui-MacBook-Air:~$

find / -name nginx.conf ↵
cd /etc/nginx ↵
cp nginx.conf nginx-copy.conf ↵
vi nginx.conf ↵
nginx -s reload ↵
```

NginX 설정파일 수정하기

1. ec2-user@ip-172-31-25-131:~ (bash)

a1@1ui-MacBook-Air:~\$

- 1) `find / -name nginx.conf` ↵
- 2) `cd /etc/nginx` ↵
- 3) `cp nginx.conf nginx-copy.conf` ↵
- 4) `vi nginx.conf` ↵
- 5) `nginx -s reload` ↵

1. nginx.conf(Nginx 설정 파일) 위치 찾기
2. nginx.conf가 위치한 디렉토리로 이동
3. 잘못된 수정을 하면 원래 상태로 백업을 위해 기존 설정 파일 복사
4. vi 에디터로 nginx.conf 열고 파일 수정하기
5. 바뀐 설정을 적용하기 위해 Nginx 재기동

NginX 설정파일 수정하기

```
server {  
    listen    80 default_server;  
    listen    [::]:80 default_server;  
    server_name _;  
    root      /usr/share/nginx/html/Fastcampus-web-deploy;  
  
    # Load configuration files for the default server block.  
    include /etc/nginx/default.d/*.conf;  
  
    location / {  
    }  
  
    error_page 404 /404.html;  
        location = /40x.html {  
    }  
  
    error_page 500 502 503 504 /50x.html;  
        location = /50x.html {  
    }  
}
```

해당 내용 추가하기

NginX 설정파일 수정하기

```
server {  
    listen      80 default_server;  
    listen      [::]:80 default_server;  
    server_name _;  
    root        /usr/share/nginx/html/Fastcampus-web-deploy;  
  
    # Load configuration files for the default server block.  
    include /etc/nginx/default.d/*.conf;  
    root  
    location / {  
        Nginx로 들어오는 HTTP 요청에 응답하기 위해 맨 처음으로 찾는 디렉토리  
        - /usr/share/nginx/html -> /usr/share/nginx/html/Fastcampus-web-deploy로 변경  
    }  
  
    error_page 404 /404.html;  
    location = /40x.html {  
    }  
  
    error_page 500 502 503 504 /50x.html;  
    location = /50x.html {  
    }  
}
```


NginX의 시점

http://ec2-dns/page/album/

HTTP로 들어온 요청

NginX의 시점

<http://ec2-dns/page/album/>

웹 서버가 설치된 서버 컴퓨터 or VM or 컨테이너의 위치
(IP -> DNS)

NginX의 시점

`http://ec2-dns/page/album/`

여기서부터 NginX가 찾을 디렉토리

NginX의 시점

`/usr/share/nginx/html/Fastcampus-web-deploy/page/album/`

방금 설정된 NginX root 디렉토리

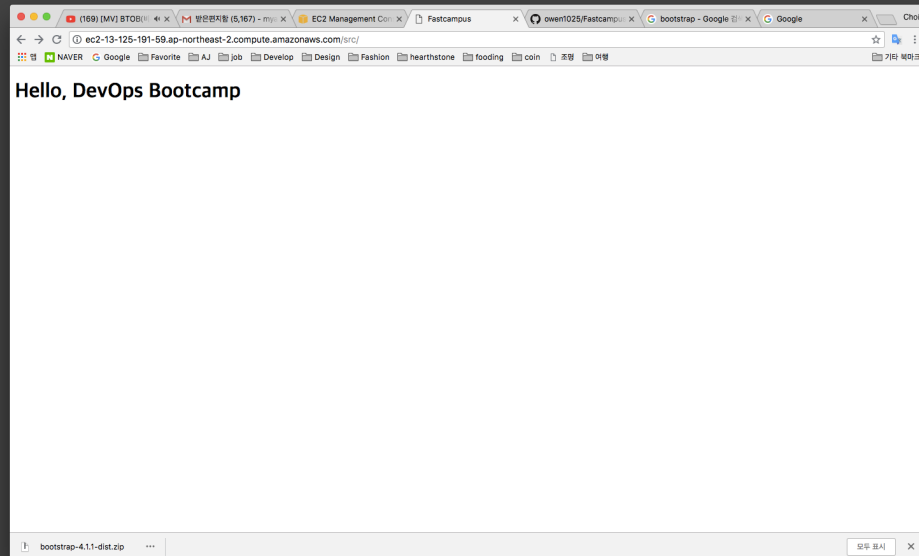
NginX의 시점

<http://ec2-dns/page/album/>

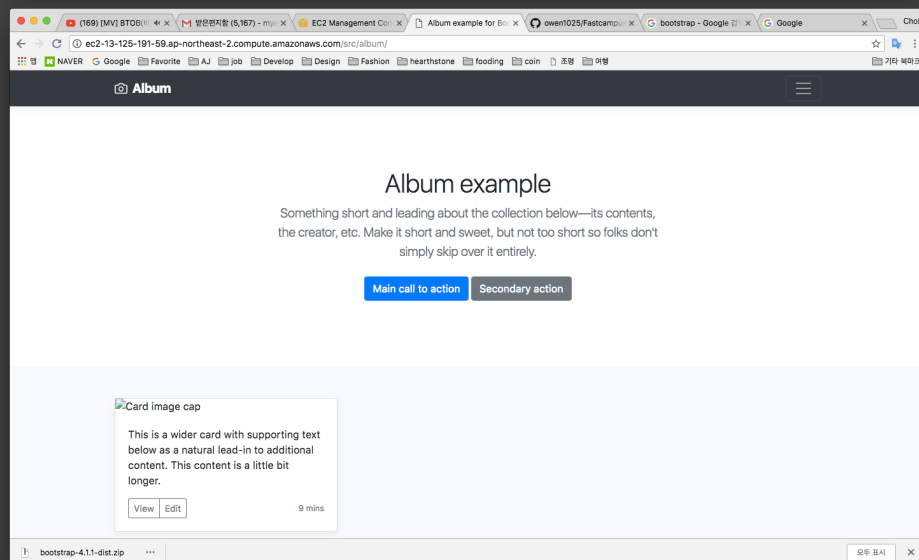


<http://ec2-dns/usr/share/nginx/html/Fastcampus-web-deploy/page/album/>

NginX 설정파일 수정하기

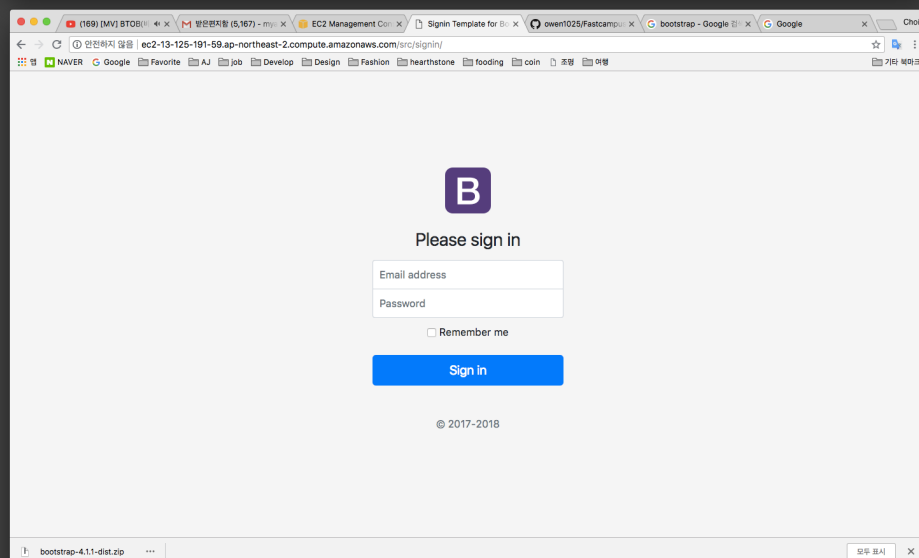


http://ec2-domain



http://ec2-domain/page/album

- 외부에 보여주면 안되는 디렉토리
- 서비스 운영 시 URL이 길어지는 불편함



http://ec2-domain/page/signin

NginX의 시점

http://ec2-dns/album/



http://ec2-dns/page/album/

Nginx 설정 파일 중 location의 내용을 바꿔 해당 요청처럼 변환

NginX 설정파일 수정하기

```
server {  
    listen      80 default_server;  
    listen      [::]:80 default_server;  
    server_name _;  
    root        /usr/share/nginx/html/Fastcampus-web-deploy;  
  
    # Load configuration files for the default server block.  
    include /etc/nginx/default.d/*.conf;
```

```
location / {  
}
```

```
error_page 404 /404.html;
```

```
location = /40x.html {
```

```
}location
```

- Nginx에 요청된 URL 별 특정 위치에 적용할 설정 그룹에 대한 정의 내용

```
error_page 500 503 502 504 /50x.html;  
# 여러 URL을 처리하기 위해 정규표현식을 사용하는 경우가 일반적임
```

```
location = /50x.html {
```

```
}
```

```
}
```


NginX 설정파일 수정하기

```
server {
```

```
...
```

```
location / {  
}
```

해당 내용 추가하기

```
location ~* /(album|signin) {  
    root /usr/share/nginx/html/Fastcampus-web-deploy/page;  
}
```

```
...
```

```
}
```

NginX 설정파일 수정하기

```
server {
```

```
...
```

```
location / {  
}
```

```
location ~* /(album|signin) {  
    root /usr/share/nginx/html/Fastcampus-web-deploy/page;  
}
```

```
...
```

```
~ / ~*
```

```
}
```

- 들어오는 URL을 정규표현식으로 처리를 위해 사용하는 옵션
- ~ : 대소문자 구분 하여 적용
- ~* : 대소문자 구분 없이 적용

NginX 설정파일 수정하기

```
server {
```

```
...
```

```
location / {  
}
```

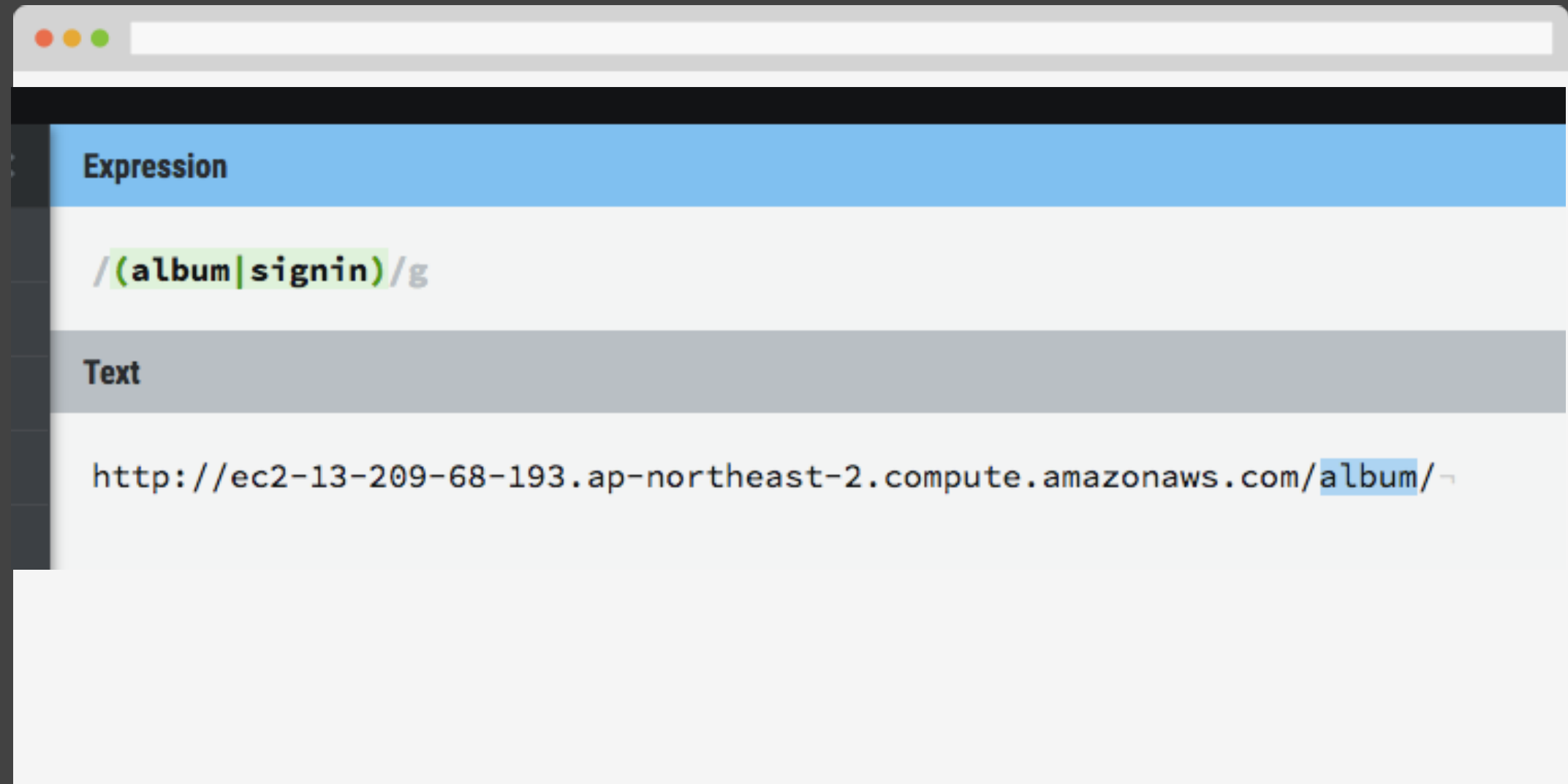
```
location ~* /(album|signin) {  
    root /usr/share/nginx/html/Fastcampus-web-deploy/page;  
}
```

```
...
```

```
} (album|signin)
```

- Nginx가 처리하는 URL이 album 혹은 signin에 매칭되는 지 확인
- 해당 URL이 매칭된다면 Nginx root 디렉토리를 해당 블록에 적힌 디렉토리로 변경

NginX 설정파일 수정하기



The screenshot shows a web browser window with a light gray header bar containing three colored dots (red, yellow, green) and a search bar. Below the header, there are two main sections. The first section has a blue header labeled 'Expression' and contains the text `/(album|signin)/g`. The second section has a gray header labeled 'Text' and contains the text `http://ec2-13-209-68-193.ap-northeast-2.compute.amazonaws.com/album/`. The word 'album' in the URL is highlighted in blue.

<https://regexr.com/>

- 정규표현식 확인, 검사, 연습

NginX 설정파일 수정하기

```
server {
```

```
...
```

```
location /api {  
    proxy_pass {ec2_private_dns/ip}:8080  
}
```

```
location ~* /(album|signin) {  
    root /usr/share/nginx/html/Fastcampus-web-deploy/page;  
    proxy_pass  
} 들어온 요청을 다른 웹 서버 or WAS로 전달 (reverse proxy)  
- DB서버와 연결된 WAS의 ip와 포트 번호를 감출 수 있음(보안)  
}
```

NginX 설정파일 수정하기

그 외에도

- worker_processes auto(or number);
 - 몇 개의 thread를 사용할 것인지 정의, CPU core 수를 넣어주면 된다.
 - auto로 설정하면 해당 머신의 CPU core 수 만큼 자동으로 설정됨
- worker_connections 1024;
 - worker thread 당 최대 몇 개의 connection을 처리할 것인지 정의
 - 최대 처리량 = worker_processes * worker_connections
- reverse proxy, **Load balancer**, web cache 서버 등 설정에 따라 다양하게 사용 가능

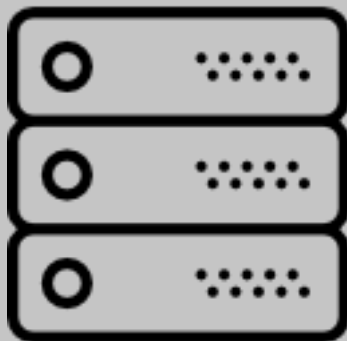
Load balancer



Client



동시 접속 최대 사용자 수 : 2000명



웹 서버(NginX - 2 Cpu, worker_processes 2)

Load balancer



Client



동시 접속 최대 사용자 수 : 2000명
-> Nginx 프로세스가 **마비**된다면?



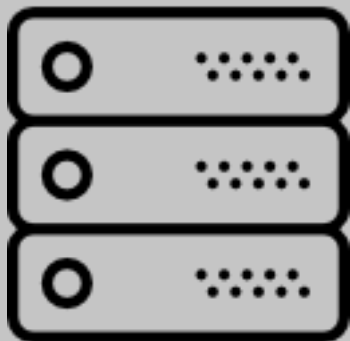
웹 서버(Nginx - 2 Cpu, worker_processes 2)

Error

Load balancer



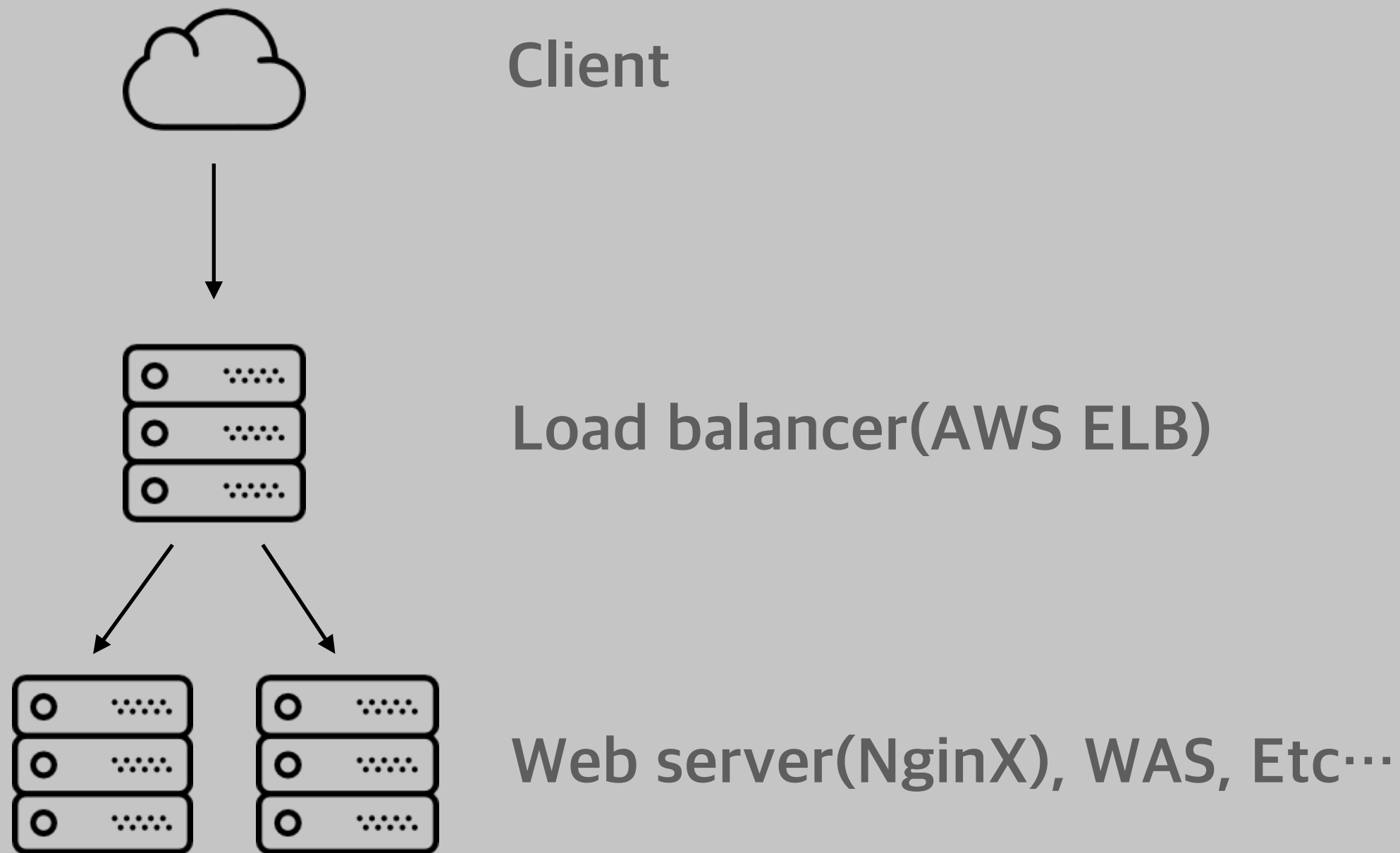
Client



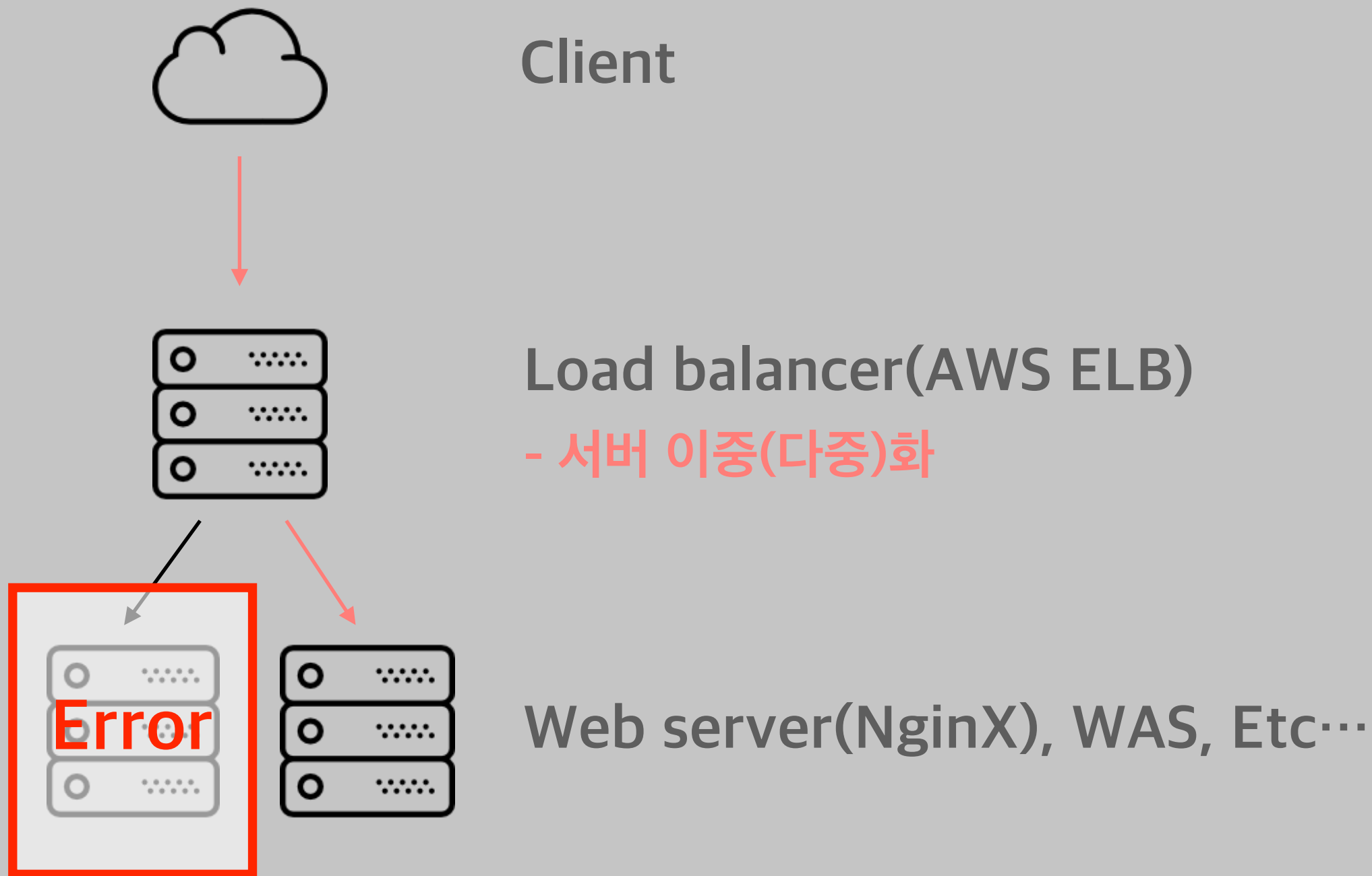
동시 접속 최대 사용자 수 : 2000명
-> 3000명으로 늘게 된다면?

웹 서버(NginX - 2 Cpu, worker_processes 2)

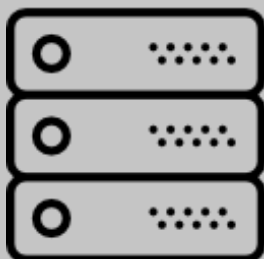
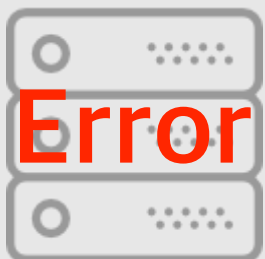
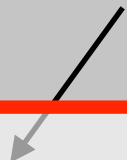
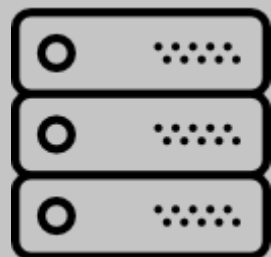
Load balancer



Load balancer



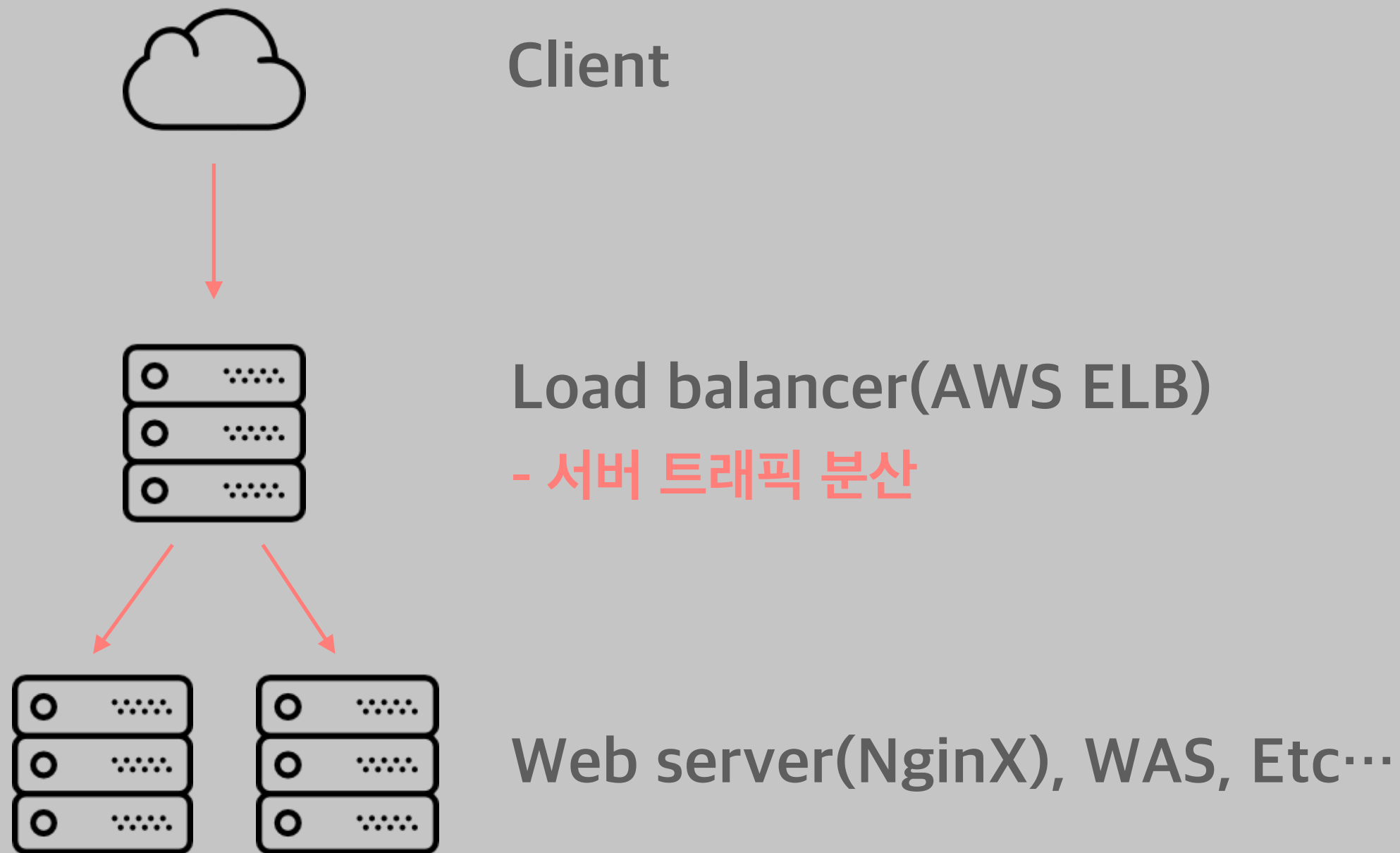
Load balancer



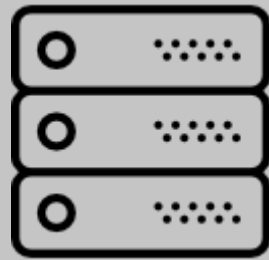
Load balancer(Health-check)

- 각 인스턴스들에게 주기적으로 정상적으로 작동하고 있는 지 물어본다.
 - ex) /GET index.html
- 확인 요청을 보냈을 때 200번대 HTTP status code를 주면 정상
- 그 외에 HTTP status code(ex. **500**)를 받으면 장애 상태라고 판단
 - 정상적인 인스턴스에게만 요청을 전달한다.

Load balancer

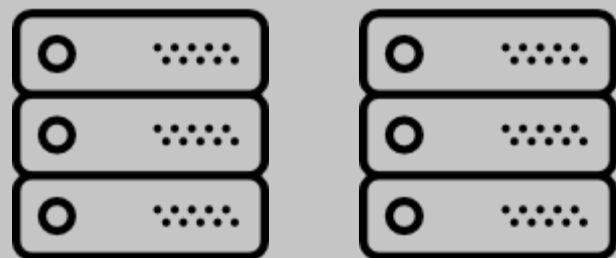


Load balancer



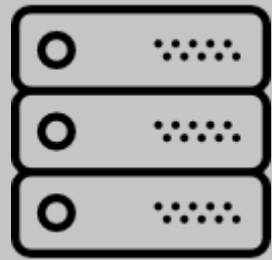
Load balancer(AWS ELB)

- 서버 트래픽 분산
- 동시 접속 최대 사용자 수가 가늠이 안될 때
 - ex. 마케팅 진행 시



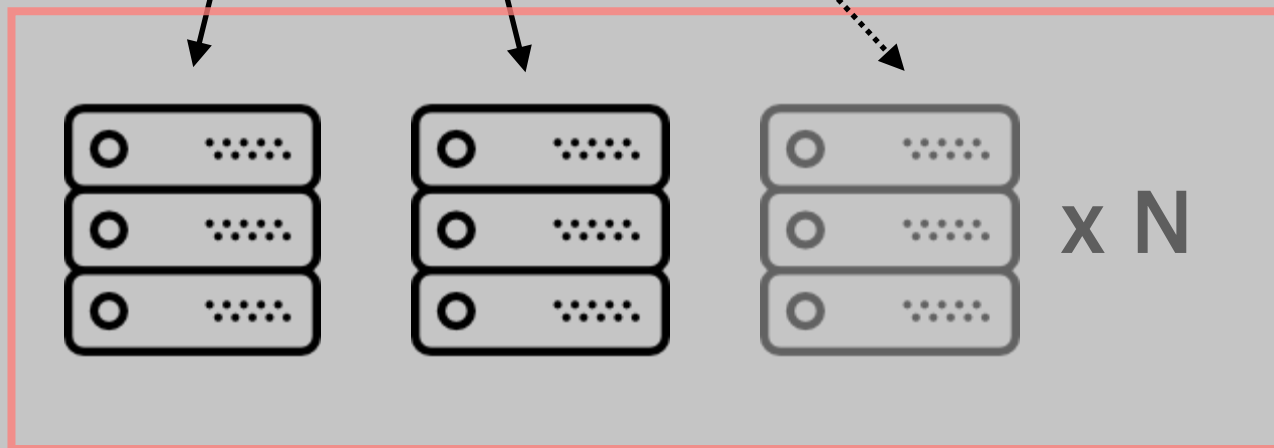
Web server(NginX), WAS, Etc...

Load balancer



Load balancer(AWS ELB)

- 서버 트래픽 분산
- 동시 접속 최대 사용자 수가 가늠이 안될 때
 - ex. 마케팅 진행 시



Auto scaling group

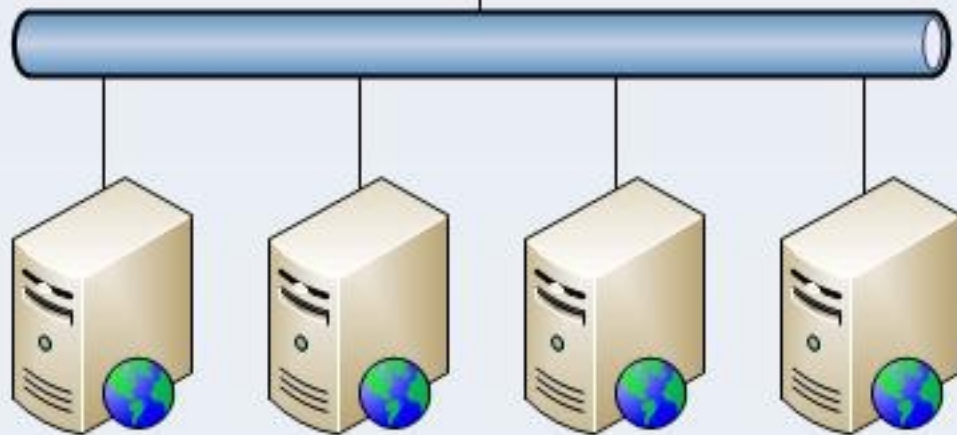
- 특정 조건(cpu 사용량, 시간)에 해당되면 자동으로 EC2 생성, 운영

Web server(NginX), WAS, Etc...

Scale-up VS Scale-out



Scale Out



Small Instance

Scale Up



Large Instance

Scale-up VS Scale-out

Scale-up (사양 추가)

- 물리 머신(베어메탈), VM, Container 등 OS가 작동하는 물리적/논리적 머신의 CPU, RAM, Disk 등 서버의 자원을 올린다.
- AI, Big Data 등 고성능 머신이 필요한 경우

Scale-out (서버 추가)

- 데이터 처리가 많은 환경에서 하나의 서버가 처리하는 일을 같은 서버가 나누어서(분산) 처리
- 일반적인 웹 서비스에 적합

Load balancer

**AWS EC2 생성, Nginx 설치/운영.pdf를 참고하셔서
새로운 EC2를 만들어주세요.**

AWS ELB 생성, 설정

The screenshot shows the AWS Management Console interface for selecting a load balancer type. The header includes the AWS logo and navigation links. The main content area is titled '로드 밸런서 유형 선택' (Select Load Balancer Type). Below the title, there is a brief introduction to Elastic Load Balancing and its three types: Application Load Balancer, Network Load Balancer (new), and Classic Load Balancer. The page is divided into three columns, each representing a different load balancer type. Each column has a header, a central icon, a '생성' (Create) button, and a '자세히 알아보기 >' (Learn more >) link.

Application Load Balancer	Network Load Balancer	Classic Load Balancer
생성	생성	생성
<p>HTTP 및 HTTPS 트래픽을 사용하는 웹 애플리케이션에 대해 유연한 기능을 설정해야 할 경우 Application Load Balancer를 선택합니다. 요청 수준에서 작동하는 Application Load Balancer는 마이크로서비스 및 컨테이너를 비롯해 애플리케이션 아키텍처를 목표로 한 고급 라우팅, TLS 종료 및 표시 기능을 제공합니다.</p> <p>자세히 알아보기 ></p>	<p>애플리케이션에 초고성과 정적 IP 주소가 필요한 경우 Network Load Balancer를 선택합니다. 연결 수준에서 작동하는 Network Load Balancer는 초당 수백만 개의 요청을 처리하면서도 극히 낮은 지연 시간을 유지할 수 있습니다.</p> <p>자세히 알아보기 ></p>	<p>EC2-Classic 네트워크에서 구축된 기존 애플리케이션이 있는 경우 Classic Load Balancer를 선택합니다.</p> <p>자세히 알아보기 ></p>

1. Classic Load Balancer 생성

AWS ELB 생성, 설정

aws

서비스 ▾ 리소스 그룹 ▾

owen ▾ 서울 ▾ 지원 ▾

1. 로드 밸런서 정의 2. 보안 그룹 할당 3. 보안 설정 구성 4. 상태 검사 구성 5. EC2 인스턴스 추가 6. 태그 추가 7. 검토

단계 1: 로드 밸런서 정의

기본 구성

이 마법사는 새 로드 밸런서를 설정하는 방법을 안내합니다. 먼저 새 로드 밸런서를 다른 로드 밸런서와 구별할 수 있도록 고유한 이름을 지정하는 것부터 시작합니다. 또한 로드 밸런서에 포트 및 프로토콜도 구성해야 합니다. 클라이언트의 트래픽은 로드 밸런서 포트부터 EC2 인스턴스의 포트까지 라우팅됩니다. 기본적으로 로드 밸런서는 포트 80에서 표준 웹 서버로 구성되어 있습니다.

로드 밸런서 이름:

nginx-lb

LB 생성할 VPC:

내 기본 VPC (172.31.0.0/16)

내부 로드 밸런서 생성:

☐ (자세히 알아보기)

고급 VPC 구성 활성화:

☐

리스너 구성:

로드 밸런서 프로토콜	로드 밸런서 포트	인스턴스 프로토콜	인스턴스 포트
HTTP	80	HTTP	80

추가

- 로드 밸런서 이름 : 로드 밸런서 이름입니다. nginx-lb를 입력합니다.
- LB 생성할 VPC : 로드 밸런서가 생성될 VPC입니다. 기본값 그대로 사용합니다.
- 내부 로드 밸런서 생성 : 인터넷에 연결되지 않은 내부 로드 밸런서로 생성하는 옵션입니다. 기본값 그대로 체크를 해제합니다.
- 고급 VPC 구성 활성화 : VPC에 속한 서브넷을 선택하는 옵션입니다. 이 부분을 체크하면 뒤에서 서브넷을 선택할 수 있습니다. 기본값 그대로 체크를 해제합니다.
- 리스너 구성 : 로드 밸런서가 처리할 프로토콜과 포트 번호입니다. 기본값 그대로 사용합니다.

52

 fast campus

AWS ELB 생성, 설정

 서비스 ▾ 리소스 그룹 ▾ ★

🔔 owen ▾ 서울 ▾ 지원 ▾

1. 로드 밸런서 정의 2. 보안 그룹 할당 3. 보안 설정 구성 4. 상태 검사 구성 5. EC2 인스턴스 추가 6. 태그 추가 7. 검토

단계 2: 보안 그룹 할당

VPC에서 탄력적 로드 밸런서를 사용하는 옵션을 선택하셨습니다. 그러므로 로드 밸런서에 보안 그룹을 할당할 수 있습니다. 이 로드 밸런서에 할당할 보안 그룹을 선택하십시오. 이 선택은 언제든지 변경할 수 있습니다.

보안 그룹 할당:

☒ 새 보안 그룹 생성

☐ 기존 보안 그룹 선택

보안 그룹 이름:

nginx-lb-secure

설명:

nginx lb security group

유형 ⓘ	프로토콜 ⓘ	포트 범위 ⓘ	소스 ⓘ
사용자 지정 TC ⚡	TCP	80	사용자 지정 ⚡ 0.0.0.0/0 ✕

규칙 추가

- 보안 그룹 할당 : 로드 밸런서의 Security Group입니다. ‘새 보안 그룹 생성’을 선택합니다.
- 보안 그룹 이름 : 새로 생성될 Security Group 이름입니다. 기본값 그대로 사용합니다.
- 설명 : 새로 생성될 Security Group의 설명입니다. 기본값 그대로 사용합니다.
- 앞에서 Load Balancer Protocol을 HTTP에 80번 포트로 설정했으므로 동일하게 Type을 HTTP로 설정합니다.

AWS ELB 생성, 설정



서비스 ▾

리소스 그룹 ▾



owen ▾

서울 ▾

지원 ▾

1. 로드 밸런서 정의

2. 보안 그룹 할당

3. 보안 설정 구성

4. 상태 검사 구성

5. EC2 인스턴스 추가

6. 태그 추가

7. 검토

단계 4: 상태 검사 구성

로드 밸런서는 자동으로 EC2 인스턴스에서 상태 검사를 수행하며 상태 검사를 통과하는 인스턴스로만 트래픽을 라우팅합니다. 상태 검사에 실패하는 인스턴스는 자동으로 로드 밸런서에서 제거됩니다. 요구 사항에 맞게 상태 검사를 사용자 지정하십시오.

Ping 프로토콜	<input type="text" value="HTTP"/>
Ping 포트	<input type="text" value="80"/>
Ping 경로	<input type="text" value="/index.html"/>

고급 세부 정보

응답 시간 초과 ⓘ	<input type="text" value="5"/>	초
간격 ⓘ	<input type="text" value="30"/>	초
비정상 임계값 ⓘ	<input type="text" value="2"/>	
정상 임계 값 ⓘ	<input type="text" value="10"/>	

- Ping 프로토콜 : 헬스 체크를 할 때 사용할 프로토콜입니다. HTTP, HTTPS, TCP, SSL을 선택할 수 있습니다.
- Ping 포트 : 헬스 체크를 할 때 사용할 포트 번호입니다.
- Ping 경로 : 헬스 체크를 할 때 접속할 경로입니다. HTTP, HTTPS에서만 설정할 수 있습니다.
- 응답 시간 초과 : 헬스 체크 응답 시간입니다. 이 시간이 지나도 응답이 없으면 EC2 인스턴스 가동 확인에 실패한 것으로 판단합니다.
- 간격 : 헬스 체크 주기입니다.
- 비정상 임계값 : 연속으로 설정한 값만큼 가동 확인에 실패했을 때 가동이 중단된 것으로 판단합니다. 기본값 그대로 사용합니다.
- 정상 임계 값: 가동이 중단되어 트래픽 분산에서 제외되었을 때 연속으로 설정된 값만큼 가동 확인에 성공하면 다시 포함됩니다.

AWS ELB 생성, 설정



서비스 ▾

리소스 그룹 ▾



owen ▾

서울 ▾

지원 ▾

1. 로드 밸런서 정의 2. 보안 그룹 할당 3. 보안 설정 구성 4. 상태 검사 구성 5. EC2 인스턴스 추가 6. 태그 추가 7. 검토

단계 5: EC2 인스턴스 추가

아래 표에는 모든 실행 중인 EC2 인스턴스 목록이 있습니다. 현재 로드 밸런서에 인스턴스를 추가하려면 선택 열에서 확인란을 선택하십시오.

VPC vpc-ec151b84 (172.31.0.0/16)

<input type="checkbox"/>	인스턴스 ▾	이름 ▾	상태 ▾	보안 그룹 ▾	영역 ▾	서브넷 ID ▾	서브넷 CIDR ▾
<input type="checkbox"/>	i-01591b7d512f3e180		● running	aws-ec2-secure	ap-northea...	subnet-ab9cf7e7	172.31.16.0/20
<input type="checkbox"/>	i-0b1e48d67b88e64f9		● running	aws-ec2-secure	ap-northea...	subnet-ab9cf7e7	172.31.16.0/20

가용 영역 배포

ap-northeast-2c 내 인스턴스 2개

- ☒ 교차 영역 로드 밸런싱 활성화 ⓘ
- ☒ 연결 드레인 활성화 ⓘ 300 초

취소

이전

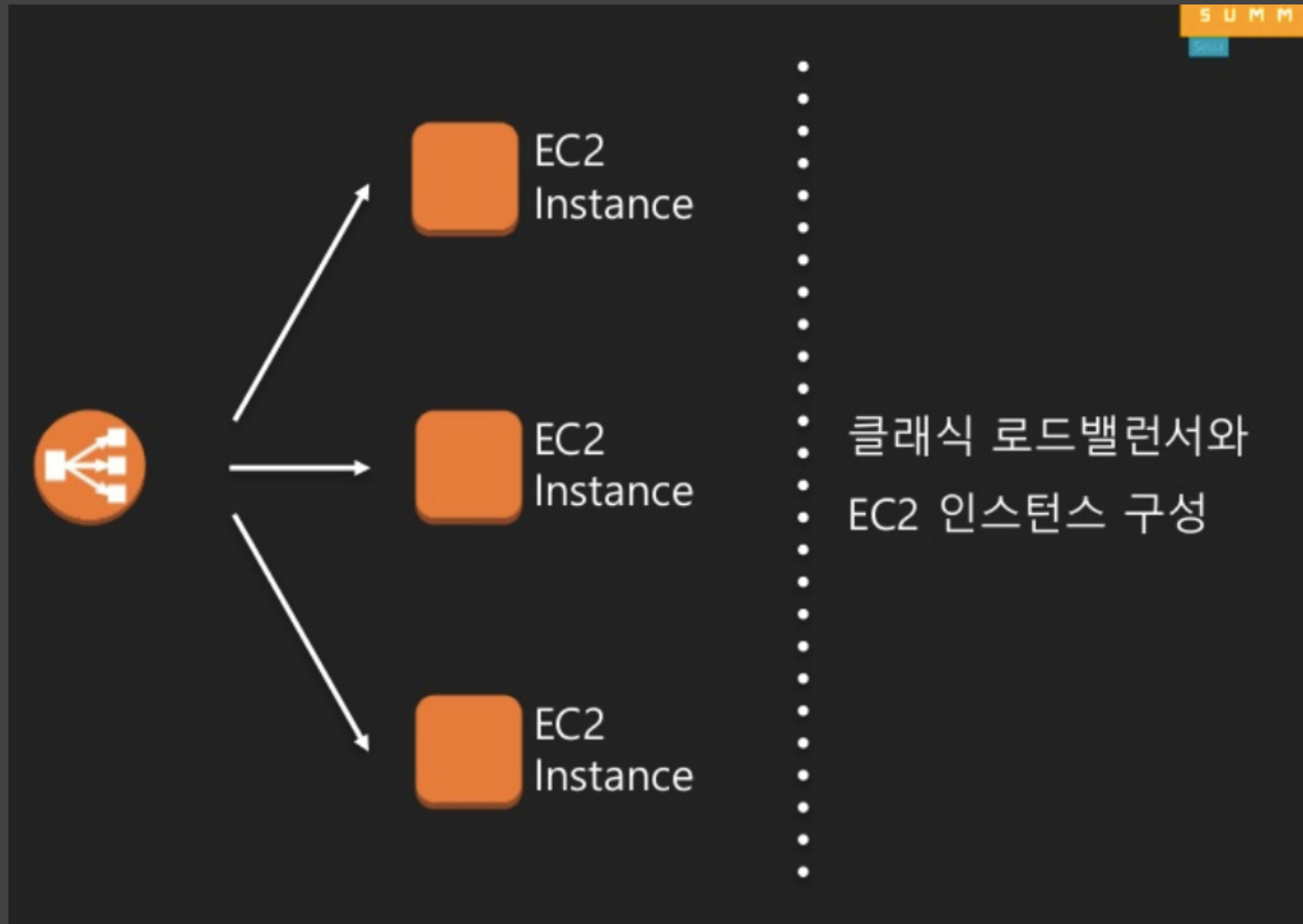
다음: 태그 추가

- 로드 밸런싱에 연결할 EC2 인스턴스를 체크합니다.
- 교차 영역 로드 밸런싱 활성화 : 여러 가용 영역에 생성된 EC2 인스턴스에 부하를 분산하는 옵션입니다.
- 연결 드레인 활성화 : Connection Draining 사용 옵션입니다. 1초부터 3600초(1시간)까지 설정할 수 있습니다.

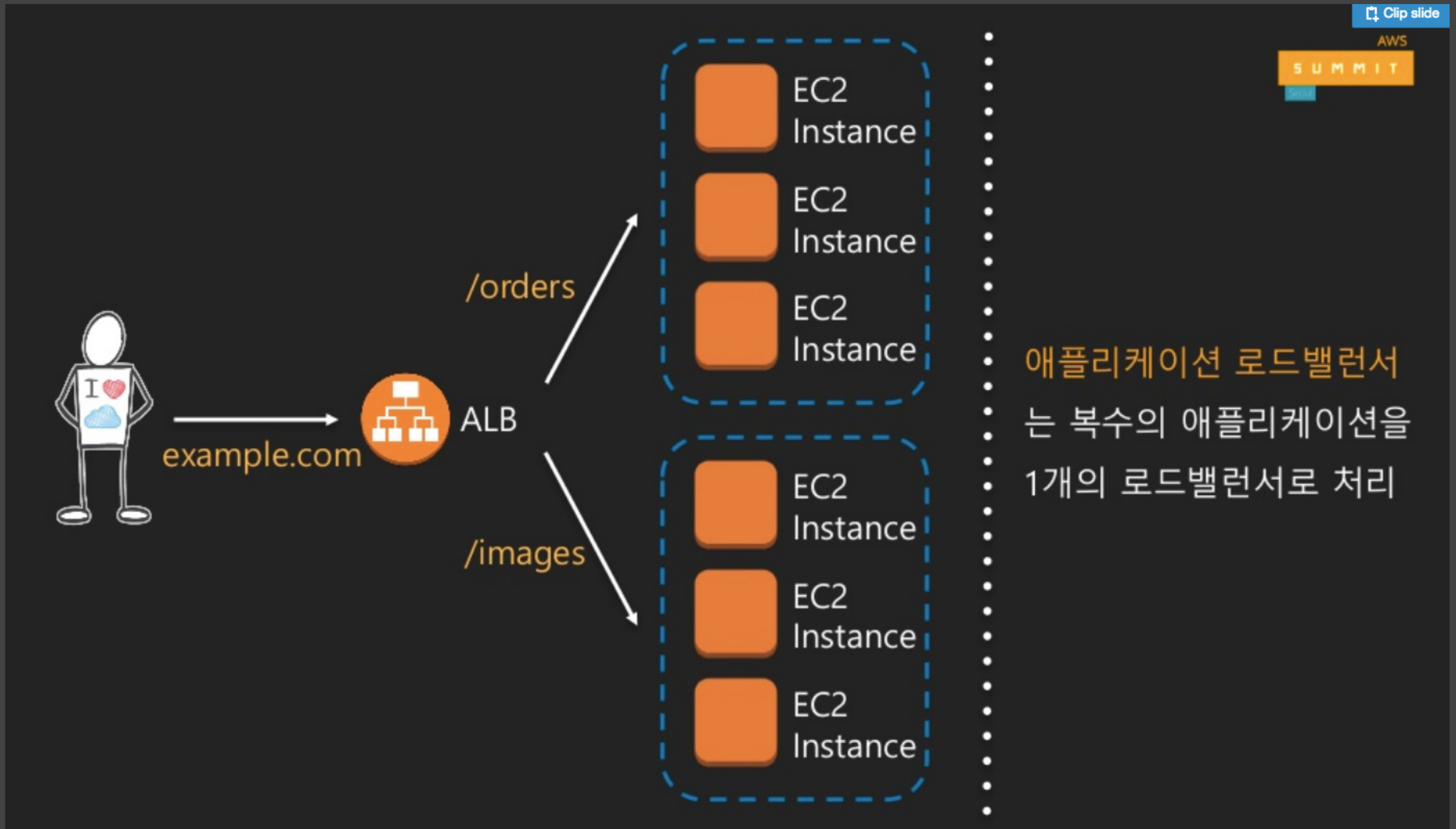
AWS ELB(Elastic Load balancer)

- AWS에서 제공하는 L4 스위치 기능을 제공하는 로드 밸런싱 서비스
- ELB가 받은 요청을 해당하는 EC2 인스턴스, 서비스 그룹 등 다양하게 전달할 수 있다.
- 너무 많은 요청을 처리하고 있거나, 정상적으로 작동하지 않는(health-check) 서버에는 요청을 보내지 않는다(고가용성 보장).
- 미니멀 한 기능을 제공하는 Classic LB, 좀 더 안정적으로 발전한 Network LB, URL/domain/service(Layer 7) 단위로 분산할 수 있는 Application LB가 있음

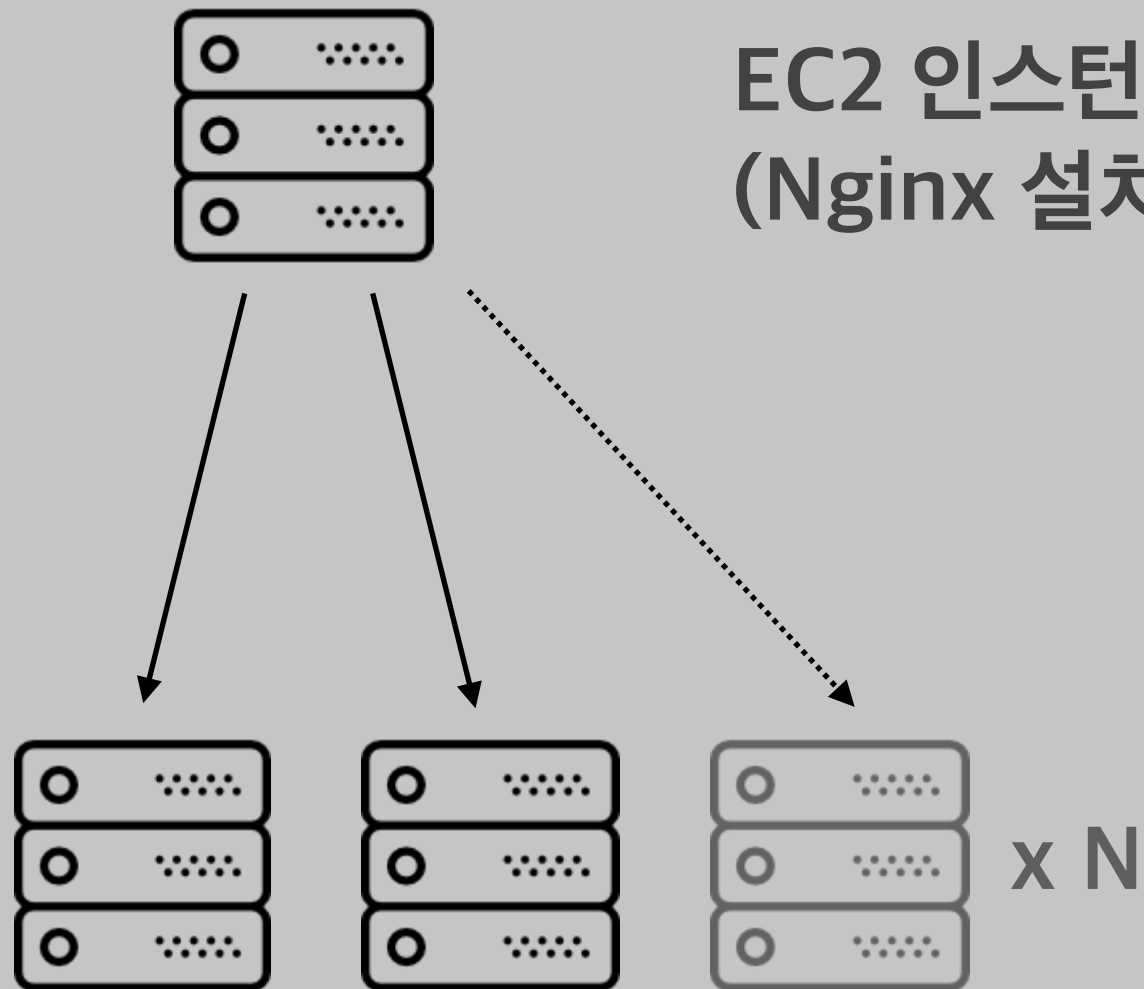
AWS ELB(Elastic Load balancer)



AWS ELB(Elastic Load balancer)



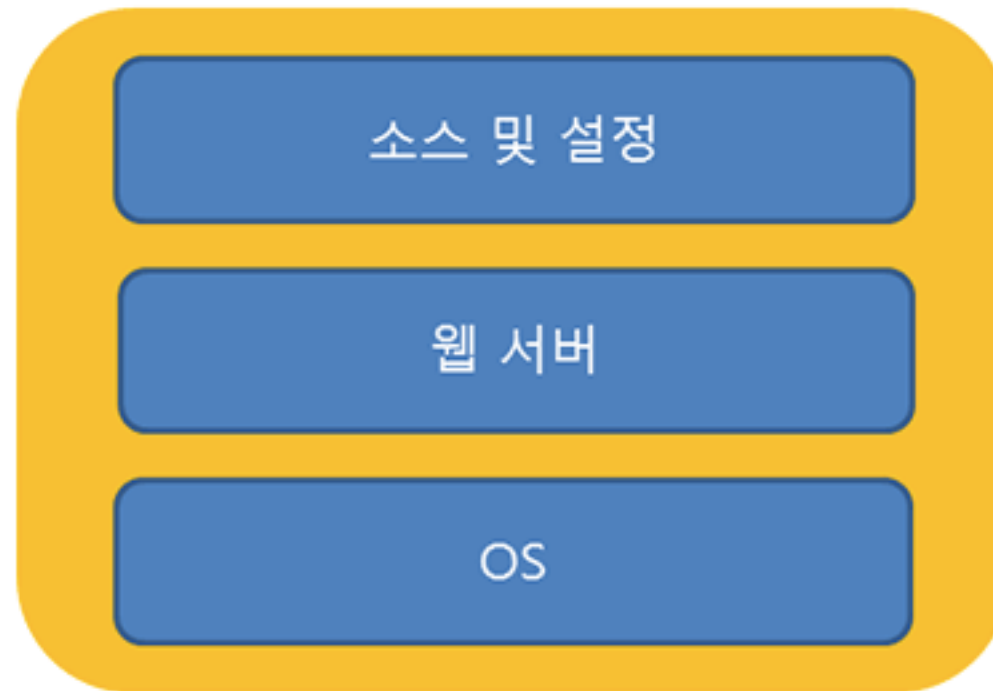
AWS AMI 생성



EC2 인스턴스를 추가할 때 마다 **새롭게 설정**
(Nginx 설치, 설정파일 변경, 실행)해야할까?

AWS AMI 생성

모든 설치와 설정이 완료된 AMI



Auto Scaling

AMI로 생성한 EC2 인스턴스들

AWS AMI 생성

1. ec2-user@ip-172-31-25-131:~ (bash)

a1@1ui-MacBook-Air:~\$

AMI 빌드를 위해 스냅샷을 찍을 EC2 인스턴스에 접속

`sudo su` ↵

`systemctl enable nginx` ↵

AMI를 기준으로 자동으로 EC2 인스턴스가 생성될 때 NginX 프로세스를 자동으로 실행하기 위해 systemctl에 nginx 실행 등록

AWS AMI 생성

The screenshot shows the AWS Management Console interface. On the left, the navigation pane includes 'EC2 대시보드', '이벤트', '태그', '보고서', '제한', '인스턴스', '인스턴스', 'Launch Templates', '스팟 요청', '예약 인스턴스', '전용 호스트', '이미지', and 'AMI'. The main content area shows the '인스턴스 시작' (Start Instance) button, a search bar for '태그 및 속성별 필터 또는 키워드', and a table of EC2 instances. A context menu is open over the first instance (Name: i-015, ID: i-015), showing options like '연결', '기존 인스턴스를 기반으로 시작', '인스턴스 상태', '인스턴스 설정', '이미지', '네트워킹', and 'CloudWatch 모니터링'. The '이미지' option is highlighted, and a sub-menu is open showing '이미지 생성' (Create Image) and '번들 인스턴스(인스턴스 스토어 AMI)' (Bundle Instance (Instance Store AMI)). The table of instances has columns: '가용 영역' (Availability Zone), '인스턴스 상태' (Instance State), '상태 검사' (Status Checks), '경보 상태' (Alarm State), and '퍼블릭 DNS(IPv4)' (Public DNS (IPv4)). The first instance is in the 'ap-northeast-2c' availability zone, has a 'running' state, and has passed '2/2 checks'.

Name	인스턴스 ID	가용 영역	인스턴스 상태	상태 검사	경보 상태	퍼블릭 DNS(IPv4)
i-015	i-015	ap-northeast-2c	running	2/2 검사 통과	없음	ec2-52-79-227-242.ap-northeast-2.compute.amazonaws.com
i-0b1	i-0b1		terminated		없음	

- 기존 생성된 EC2 인스턴스의 설정을 기반으로 AMI 생성

AWS AMI 생성

이미지 생성

인스턴스 ID

i-01591b7d512f3e180

이미지 이름

ec2-nginx-ami

이미지 설명

amazon linux2 + nginx1.12 + fastcampus-web-dep

재부팅 안 함

☐

인스턴스 볼륨

볼륨 유형	디바이스	스냅샷	크기 (GiB)	볼륨 유형	IOPS	처리량(MB/초)	종료 시 삭제	암호화
루트	/dev/xvda	snap-0d59c1d0590f2723d	8	범용 SSD(GP2)	100/3000	해당 사항 없음	<input checked="" type="checkbox"/>	암호화되지 않음

새 볼륨 추가

EBS 볼륨의 전체 크기: 8 GiB



EBS 이미지를 생성할 때 위의 각 볼륨에 대해 EBS 스냅샷이 생성됩니다.

취소

이미지 생성

- 선택한 EC2의 스냅샷(현재 상태 - 설정 등)을 가지고 AMI 빌드

AWS AMI 생성

 서비스 ▾ 리소스 그룹 ▾ 

owen ▾ 서울 ▾ 지원 ▾

1. AMI 선택 2. 인스턴스 유형 선택 3. 인스턴스 구성 4. 스토리지 추가 5. 태그 추가 6. 보안 그룹 구성 7. 검토

단계 1: Amazon Machine Image(AMI) 선택

AMI는 인스턴스를 시작하는 데 필요한 소프트웨어 구성(운영 체제, 애플리케이션 서버, 애플리케이션)이 포함된 템플릿입니다. AWS, 사용자 커뮤니티 또는 AWS Marketplace에서 제공하는 AMI를 선택하거나, 자체 AMI 중 하나를 선택할 수도 있습니다.

빠른 시작


나의 AMI

AWS Marketplace

커뮤니티 AMI

▼ 소유권

Q 나의 AMI 검색




ec2-nginx-ami - ami-0120fdd8943f2549f
amazon linux2 + nginx1.12 + Fastcampus-web-deploy
루트 디바이스 유형: ebs 가상화 유형: hvm 소유자: 322749112518

선택

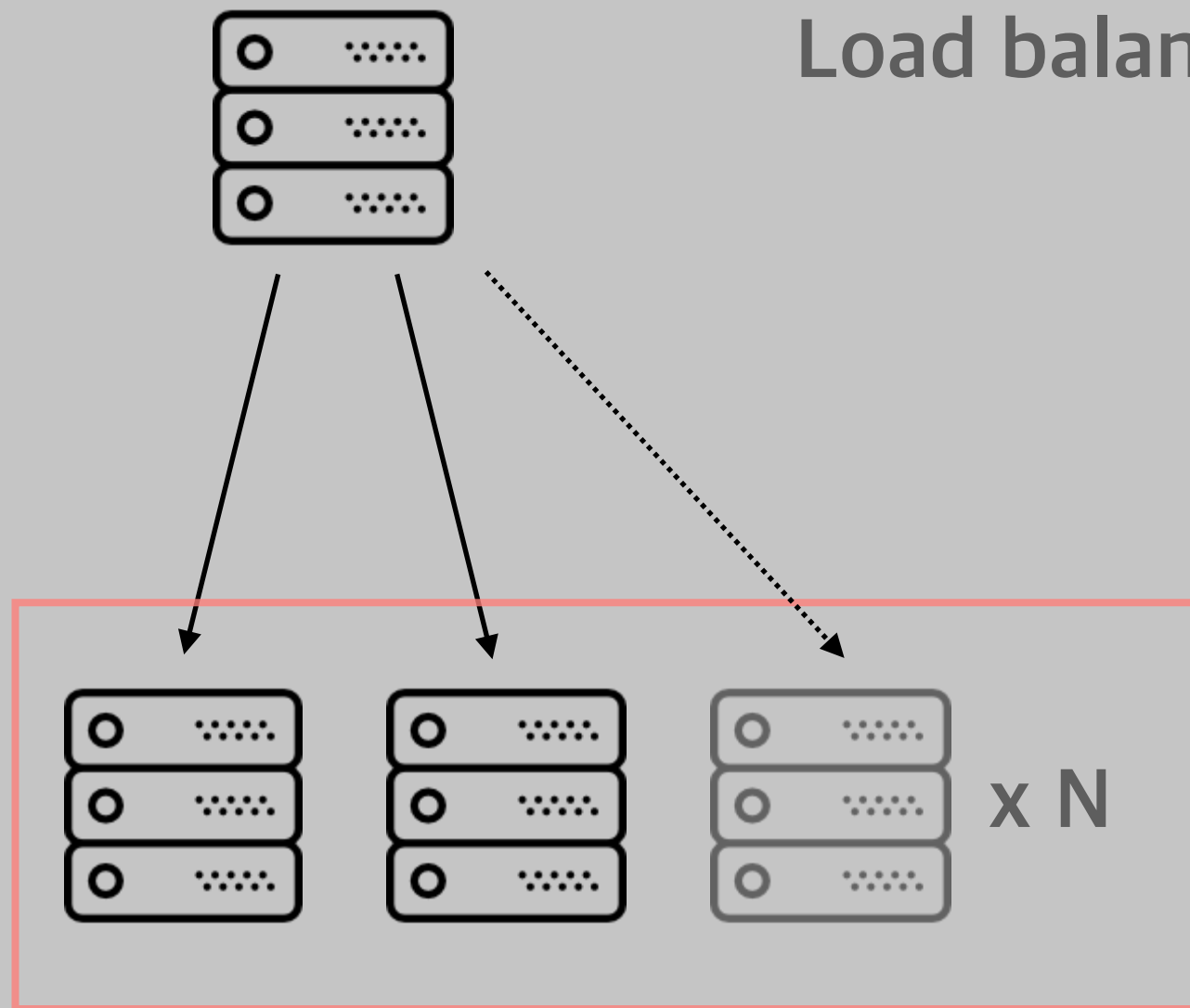
64비트

- 빌드한 AMI를 기반으로 EC2 인스턴스 생성(5~10분 정도 대기)

64

 fast campus

Auto scaling group



Load balancer(AWS ELB)

Auto scaling group

- 특정 조건(cpu 사용량, 시간)에 해당되면
자동으로 EC2 생성, 운영

Web server(NginX), WAS, Etc...

Auto scaling group

The screenshot displays the AWS Management Console interface for Auto Scaling Groups. The top navigation bar includes the AWS logo, service and resource group dropdowns, and user information (owen, 서울, 자원). The left sidebar lists various AWS services, with 'Auto Scaling' currently selected. The main content area features a notification banner about new features, a set of action buttons (시작 구성 생성, Auto Scaling 그룹 생성, 시작 템플릿으로 복사, 작업), and a table for listing Auto Scaling Groups. The table has columns for Name, AMI ID, Instance Type, Spot Price, and Creation Time, but it is currently empty, displaying the message '시작 구성이 없습니다.' (No initial configurations).

aws 서비스 ▾ 리소스 그룹 ▾

ELASTIC BLOCK STORE
볼륨
스냅샷

네트워크 및 보안
보안 그룹
탄력적 IP
배치 그룹
키 페어
네트워크 인터페이스

로드 밸런싱
로드밸런서
대상 그룹

AUTO SCALING
시작 구성
Auto Scaling 그룹

SYSTEMS MANAGER 서비스
명령 실행
상태 관리자
구성 규정 준수
자동화
패치 규정 준수
패치 기준

SYSTEMS MANAGER 공유 리소스

신규 기능 시작 템플릿 출시!
이제 EC2 Auto Scaling 콘솔은 EC2 시작 템플릿 전체를 지원합니다. 새 Auto Scaling 그룹에는 시작 템플릿을 사용하는 것이 좋습니다. 시작 템플릿을 통해서 Amazon EC2의 최신 기능을 이용할 수 있습니다. Auto Scaling 그룹을 생성하여 시작하거나 [자세히 알아보기](#).

시작 구성 생성 Auto Scaling 그룹 생성 시작 템플릿으로 복사 작업 ▾

필터: 🔍 시작 구성 필터링... ✕

이름 ▲ AMI ID ▼ 인스턴스 유형 ▼ 스팟 가격 ▼ 생성 시간 ▼

시작 구성이 없습니다.

위에서 시작 구성을 선택하십시오

• Auto scaling 시작 구성 생성

Auto scaling group

1. AMI 선택 2. 인스턴스 유형 선택 3. 세부 정보 구성 4. 스토리지 추가 5. 보안 그룹 구성 6. 검토

시작 구성 생성

[취소 및 종료](#)

AMI는 인스턴스를 시작하는 데 필요한 소프트웨어 구성(운영 체제, 애플리케이션 서버, 애플리케이션)이 포함된 템플릿입니다. AWS, 사용자 커뮤니티 또는 AWS Marketplace에서 제공하는 AMI를 선택하거나, 자체 AMI 중 하나를 선택할 수도 있습니다.

빠른 시작

|< < 1~1/1 AMI > >|

🔍 나의 AMI 검색

내 AMI

AWS Marketplace

커뮤니티 AMI



ec2-nginx-ami - ami-0120fdd8943f2549f

amazon linux2 + nginx1.12 + Fastcampus-web-deploy

루트 디바이스 유형: ebs 가상화 유형: hvm 소유자: 322749112518

선택

64비트

▼ 소유권

☒ 내 소유

☐ 나와 공유 상태

- 미리 빌드한 AMI(Amazon linux2 + nginx1.12 + Fooding-web-deploy)를 기반으로 자동으로 인스턴스 생성

Auto scaling group

1. AMI 선택 2. 인스턴스 유형 선택 3. 세부 정보 구성 4. 스토리지 추가 5. 보안 그룹 구성 6. 검토

시작 구성 생성

보안 그룹은 인스턴스에 대한 트래픽을 제어하는 방화벽 규칙 세트입니다. 이 페이지에서는 특정 트래픽을 인스턴스에 도달하도록 허용할 규칙을 추가할 수 있습니다. 예를 들면 웹 서버를 설정하여 인터넷 트래픽을 인스턴스에 도달하도록 허용하려는 경우 HTTP 및 HTTPS 트래픽에 대한 무제한 액세스를 허용하는 규칙을 추가합니다. 새 보안 그룹을 생성하거나 아래에 나와 있는 기존 보안 그룹 중에서 선택할 수 있습니다. Amazon EC2 보안 그룹에 대해 [자세히 알아보기](#).

보안 그룹 할당: ☐ 새 보안 그룹 생성
☒ 기존 보안 그룹 선택

보안 그룹 ID	이름	VPC ID	설명	작업
<input checked="" type="checkbox"/> sg-06d43ed8fbd192964	aws-ec2-secure	vpc-ec151b84	secure test interface	새로 복사
<input type="checkbox"/> sg-2c8efe46	default	vpc-ec151b84	default VPC security group	새로 복사
<input type="checkbox"/> sg-0ca09d09c7badcb91	nginx-lb-secure	vpc-ec151b84	elb nginx secure group	새로 복사

- Auto scaling group 시작 구성 - 보안 그룹 구성
 - 기존 보안 그룹 선택
 - 기존에 만든 aws-ec2-secure(22, 80 포트가 열려있는 보안 그룹) 선택
 - 기존에 만든 ELB로 들어오는 HTTP 요청이 80포트를 통해 들어오기에 시작 구성을 통해 생성되는 EC2 인스턴스들의 보안 그룹은 80포트가 열려있어야합니다.

Auto scaling group

1. Auto Scaling 그룹 세부 정보 구성 2. 조정 정책 구성 3. 알림 구성 4. 태그 구성 5. 검토

Auto Scaling 그룹 생성

시작 구성 ⓘ nginx-asg

그룹 이름 ⓘ

그룹 크기 ⓘ 시작 개수: 인스턴스

네트워크 ⓘ [새 VPC](#)

서브넷 ⓘ

subnet-ab9cf7e7(172.31.16.0/20) | 다음에서 기본값 ap-northeast-2c ×
subnet-2d1a1d45(172.31.0.0/20) | 다음에서 기본값 ap-northeast-2a ×

[새 서브넷 생성](#)

현재 Auto Scaling 그룹의 각 인스턴스에 퍼블릭 IP 주소가 할당됩니다. ⓘ

▼ 고급 세부 정보

로드 밸런싱 ⓘ ☒ 하나 이상의 로드 밸런서에서 트래픽 수신 [탄력적 로드 밸런싱](#)

클래식 로드 밸런서 ⓘ ×

대상 그룹 ⓘ

상태 검사 유형 ⓘ ☐ ELB ☒ EC2

상태 검사 유예 기간 ⓘ 초

모니터링 ⓘ ☒ CloudWatch 세부 모니터링 활성화 [자세히 알아보기](#)

인스턴스 보호 ⓘ

• Auto Scaling group 생성

- 그룹 이름 : Auto Scaling 그룹의 이름입니다. nginx-asg을 입력합니다.
- 그룹 크기 : 최초에 EC2 인스턴스를 생성할 개수입니다.
- 네트워크 : Auto Scaling 그룹이 생성될 VPC입니다.
- 서브넷 : EC2 인스턴스가 생성될 서브넷입니다. 모든 서브넷을 체크합니다.
- 로드 밸런싱 : ELB 로드 밸런서를 사용하는 옵션입니다. 앞에서 생성한 ELB 로드 밸런서(nginx-lb)를 선택합니다.
- 상태 검사 유형 : 생성 된 / 앞으로 생성 할 각 EC2 인스턴스의 헬스 체크 기능을 활성화를 위해 EC2를 체크합니다.
 - ELB : ELB 로드 밸런서에서 확인한 헬스 체크 값을 사용합니다.
 - EC2 : Auto Scaling 그룹이 개별적으로 EC2 인스턴스의 헬스 체크를 합니다.
- 상태 검사 유예 기간 : EC2 인스턴스가 부팅 되었을 때(InService) 설정한 시간 만큼 헬스 체크를 미룹니다. 기본값은 300초(5분)이지만 실습을 위해 5초로 설정합니다.
- 모니터링 : CloudWatch 세부 모니터링을 사용하는 옵션입니다. 이 부분에 체크합니다.

Auto scaling group

1. Auto Scaling 그룹 세부 정보 구성 2. 조정 정책 구성 3. 알림 구성 4. 태그 구성 5. 검토

Auto Scaling 그룹 생성

그룹의 크기(인스턴스 수)를 자동으로 조정하려는 경우 선택적으로 조정 정책을 추가할 수 있습니다. 조정 정책은 할당된 Amazon CloudWatch 경보에 대응하여 이러한 조정을 수행하기 위한 명령 세트입니다. 거하도록 선택하거나, 그룹을 정확한 크기로 설정할 수 있습니다. 경보가 트리거되면 정책이 실행되고 그룹의 크기가 적절히 조정됩니다. 조정 정책에 대해 [자세히 알아보기](#).

☐ 이 그룹을 초기 크기로 유지

☒ 조정 정책을 사용하여 이 그룹의 용량 조정

조정 범위: 및 개 사이의 인스턴스 - 이 값은 그룹의 최소 및 최대 크기입니다.

그룹 크기 증가

그룹 크기 증가를 위한 정책 추가

그룹 크기 감소

그룹 크기 감소를 위한 정책 추가

[대상 추적 조정 정책을 사용하여 Auto Scaling 그룹 조정](#) ⓘ

- Auto scaling group 생성 - 조정 정책
 - 특정 상황일 때 scale-out을 위해 조정 정책을 설정합니다.
 - 조정 범위
 - EC2 인스턴스를 최대 몇 개까지 추가하고, 삭제하더라도 최소 몇 개까지 남겨둘지 설정합니다. 최소 1개에서 최대 3개까지 늘려보겠습니다. 1과 3을 입력합니다.

Auto scaling group

경보 생성

지표 데이터가 정의한 수준에 도달할 때는 항상 자동으로 통지되는 CloudWatch 경보를 사용할 수 있습니다.
경보를 편집하려면 먼저 알림을 받을 사람을 선택한 다음 알림을 보낼 시간을 정의하십시오.

☒ 알림 보낼 대상: 취소

수신자:

다음 경우 항상: /

이(가): %

최소 다음의 경우: 연속 기간

경보 이름:

CPU 사용률 %

80
60
40
20
0

7/12 04:00 7/12 06:00 7/12 08:00

nginx-asg

취소 **경보 생성**

• Auto scaling group 생성 - 조정 정책(그룹 크기 증가)

- 알림 보낼 대상 : 측정치에 도달했을 때 알림을 받습니다. 알람 이름과 알람을 받을 이메일 주소를 입력합니다.
- 다음 경우 항상 : CloudWatch 측정치 종류와 측정 기준입니다. CPU 값의 모니터링을 위해 Average와 CPU 사용률을 체크합니다.
- 이(가) : 측정치입니다. CPU 사용률 80% 이상으로 설정할 것이므로 >=에 80을 입력합니다.
- 최소 다음의 경우 : 특정 시간 동안 설정한 값만큼 연속으로 측정치에 도달했을 때 알람을 발생시킵니다. 5분 동안 1번 도달했을 때 알람을 발생시킬 것이므로 1을 입력하고, 5 Minute를 선택합니다.

Auto scaling group

그룹 크기 증가

이름:

정책 실행 요건: **awsec2-nginx-asg-High-CPU-** [편집](#) [제거](#)
경보 임계값 위반: 1회 연속 기간(300초) 동안 CPUUtilization >= 80
지표 차원: AutoScalingGroupName = nginx-asg

작업 수행: 1 조건 <= CPUUtilization < +무제한

[단계 추가](#) ⓘ

인스턴스 필요 시간: 각 단계 후 워밍업 시간(초)

[단순 조정 정책 생성](#) ⓘ

• Auto scaling group 생성 - 조정 정책(그룹 크기 증가)

- 작업 수행 : 앞서 설정한 Cloudwatch에서 설정 값에 도달한 경우 수행하는 값입니다.
 - 추가(1) : 오른쪽 조건($80 \leq \text{CPUUtilization}$)에 도달하면 수행합니다. 1개의 인스턴스를 늘립니다.
 - 인스턴스 : EC2 인스턴스 단위로 추가합니다.
 - 그룹 백분율 : 현재 Auto Scaling 그룹 안에 생성된 EC2 인스턴스 개수를 기준으로 추가합니다.
- 인스턴스 필요 시간 : EC2 인스턴스를 추가한 뒤 설정한 시간 동안 기다립니다. 짧은 시간 안에 연속으로 EC2 인스턴스가 추가되는 것을 방지합니다.

Auto scaling group

Auto Scaling 그룹 생성

지정된 이벤트(인스턴스 시작 성공, 인스턴스 시작 실패, 인스턴스 종료 및 인스턴스 종료 실패 포함)가 발생할 때마다 알림을 이메일 주소와 같은 지정된 엔드포인트로 전송하도록 Auto Scaling 그룹을 구성합니다.

새 주제를 생성한 경우 확인 메시지가 도착되었는지 이메일을 확인하고 포함된 링크를 클릭하여 구독 정보를 확인합니다. 알림은 확인된 주소로만 전송할 수 있습니다.

알림 보낼 대상:

cpu-high (myartame@gmail.com) 주제 생성

인스턴스 상태:

☒ 시작
☒ 종료
☒ 시작 실패
☒ 종료 실패

알림 보낼 대상:

cpu-low (myartame@gmail.com) 주제 생성

인스턴스 상태:

☒ 시작
☒ 종료
☒ 시작 실패
☒ 종료 실패

알림 추가

- Auto Scaling group 알림 설정

- cpu-high일 때, cpu-low일 때 설정한 email로 알림을 받을 수 있도록 추가합니다.
- 인스턴스 상태 : 추가 / 제거될 때 각 상태에 맞게 알림을 받도록 체크합니다.

Auto scaling group

필터:

<input type="checkbox"/>	이름	시작 구성 / 템플릿	인스턴스	목표 용량	최소	최대	가용 영역	기본 휴지	상태 검사 유예 기간
<input checked="" type="checkbox"/>	nginx-asg	nginx-asg	1	1	1	3	ap-northeast-2a, ap-northea...	300	5

Auto Scaling 그룹: nginx-asg

세부 정보 | 활동 기록 | 조정 정책 | **인스턴스** | 모니터링 | 알림 | 태그 | 예약된 작업 | 수명 주기 후크

작업 ▾

Filter: 모든 상태 ▾ 모든 수명 주기 상태 ▾

<input type="checkbox"/>	인스턴스 ID	수명 주기	시작 구성 이름	가용 영역	상태	다음에서 보호
	i-0952b55c565eb3ec9	보류 중	nginx-asg	ap-northeast-2a	Healthy	

- Auto scaling group 설정을 통해 처음(최소)로 생성된 인스턴스에 SSH로 접속합니다.

Auto scaling group

1. ec2-user@ip-172-31-25-131:~ (bash)

a1@1ui-MacBook-Air:~\$

방금 Auto scaling group을 통해 생성된 EC2 인스턴스에 SSH로 접속 후

yes > /dev/null ↵

서비스에 사용자가 늘어나서 트래픽이 증가했다고 가정하고, CPU 사용률을 강제로 올리기 위해 yes > /dev/null 명령을 입력합니다.

- 약 5~7분 정도 기다리면 알림과 함께 새로운 EC2 인스턴스가 생성됩니다.
 - 로드밸런서 탭에서 로드밸런서와 새로 생성된 EC2 인스턴스가 물려있는 지 확인

Auto scaling group 응용 방법

- CPU, 메모리, 네트워크 응답 속도 등의 지표를 통하여 갑작스럽게 몰려오는 요청이나, 특정 인스턴스에 장애가 나더라도 사람이 직접 나서기 전에 시스템이 자동으로 대처하게 할 수 있다.
- Auto Scaling Group에서 어떤 경위로 인스턴스 수가 자동으로 변경됐는지 이메일로 알림을 받아 서비스가 이상이 생겼는지 파악할 수도 있다.
- 서비스 특성 상 사용자가 몰리는 시간, 몰리지 않는 시간에 최소/최대 값을 지정하여 **서버 비용**을 효율적으로 관리할 수 있다

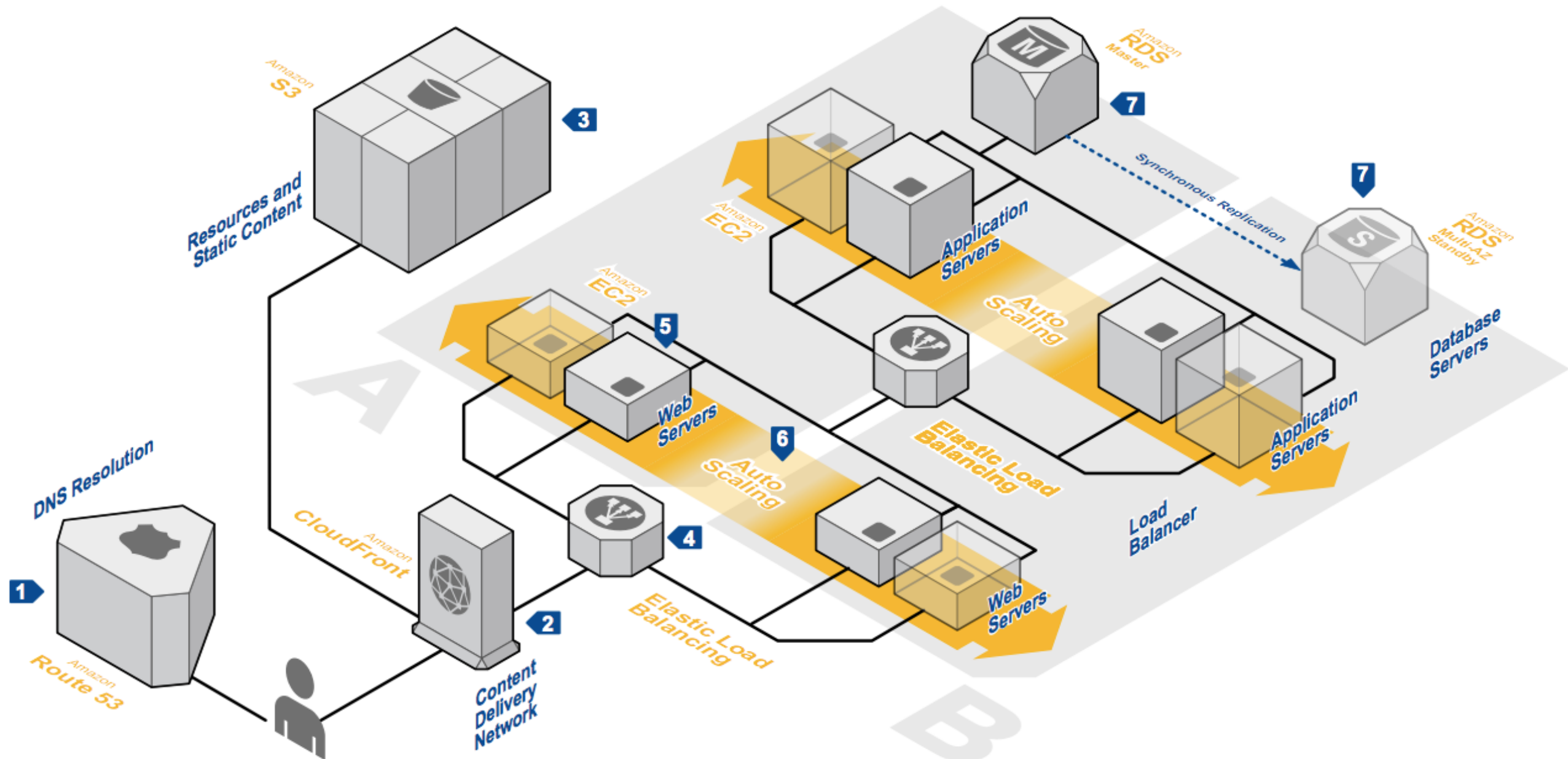
서버 아키텍처 별 비교

자주 사용하는 운영 서버 아키텍처

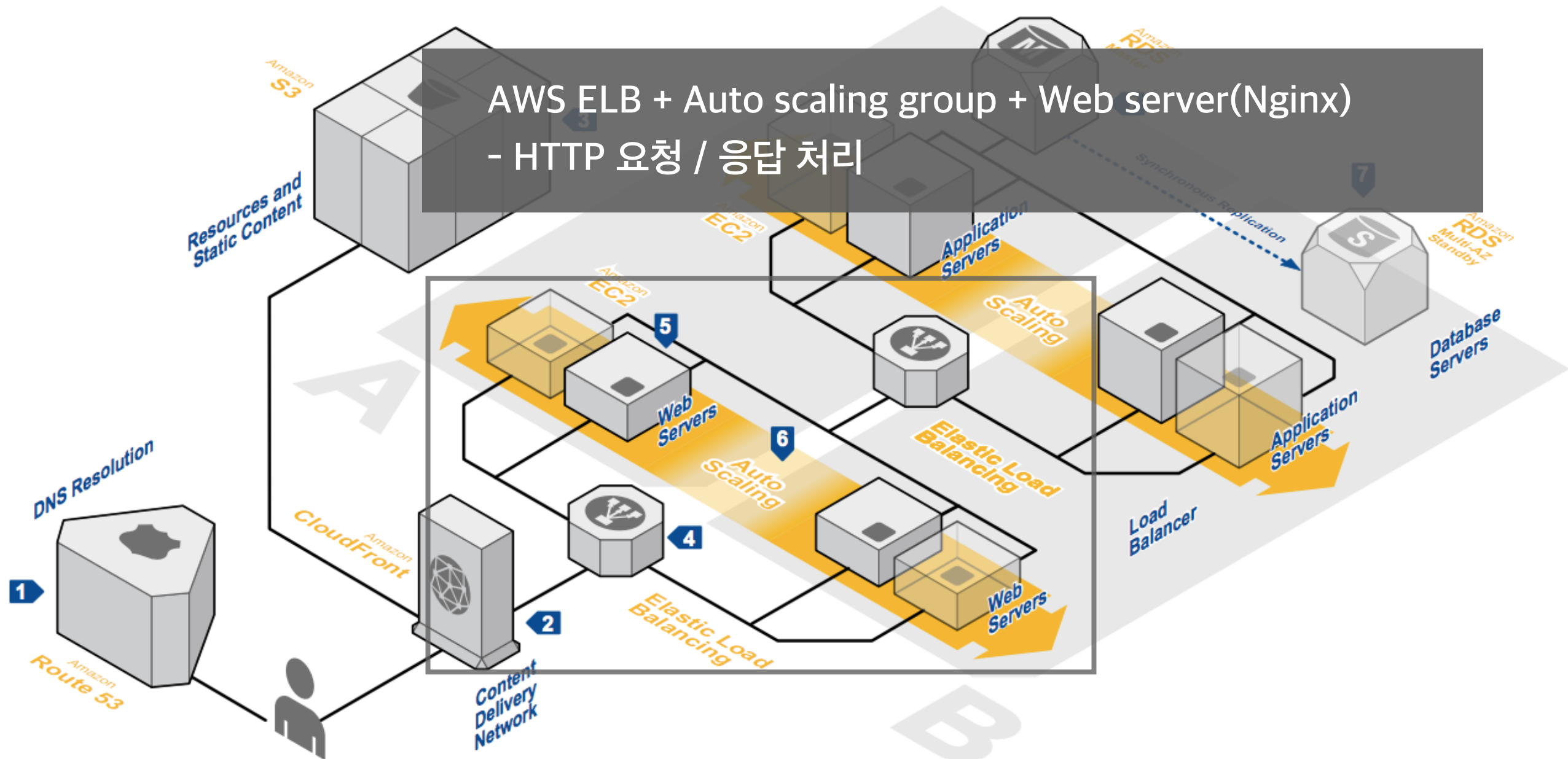
모노로틱, 서버리스, 마이크로 서비스 아키텍처

운영 서버에서 주로 사용하는 솔루션, 툴

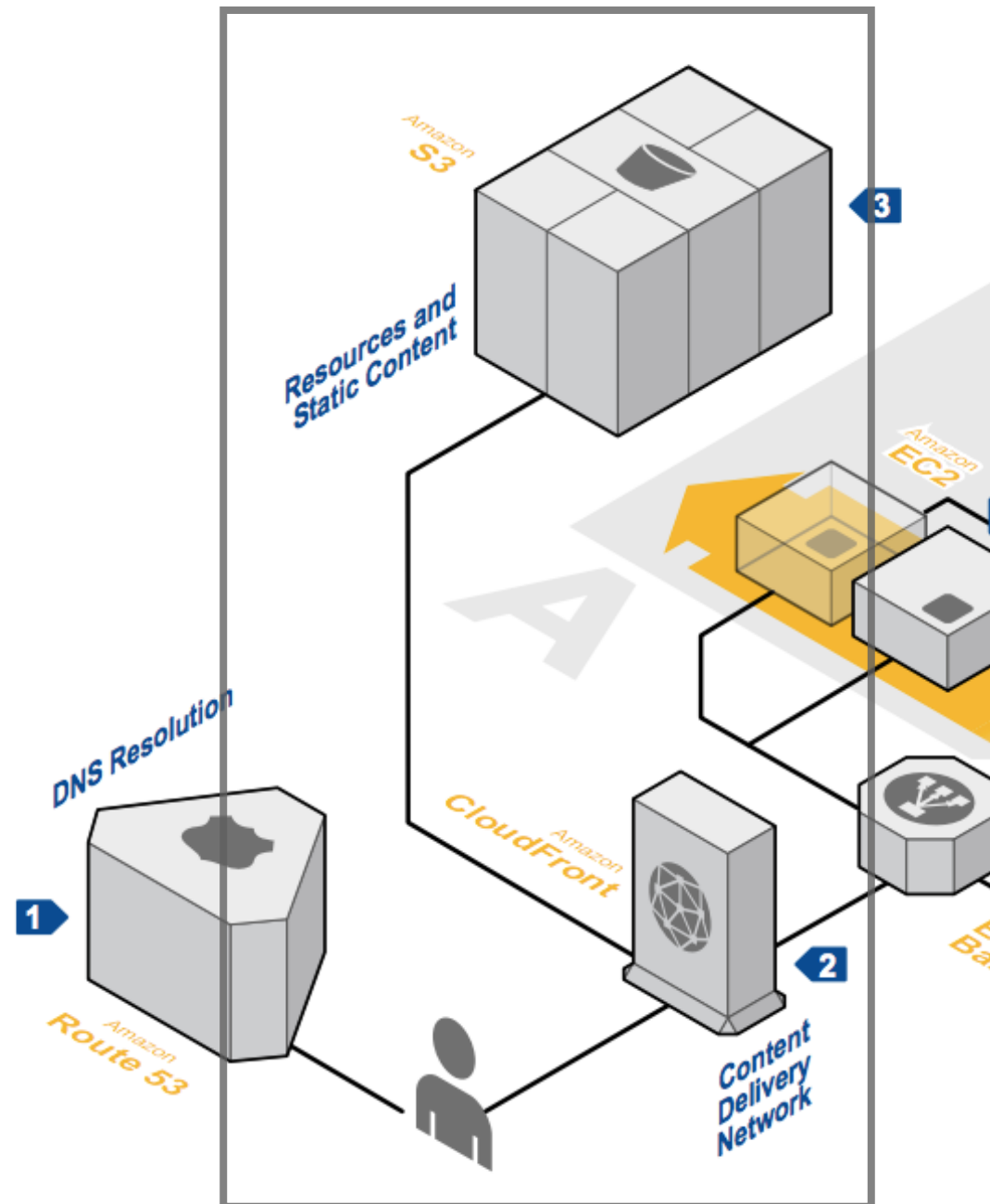
웹 서비스 아키텍처



웹 서비스 아키텍처



웹 서비스 아키텍처



정적 파일(CSS, Image, etc...) 제공 아키텍처

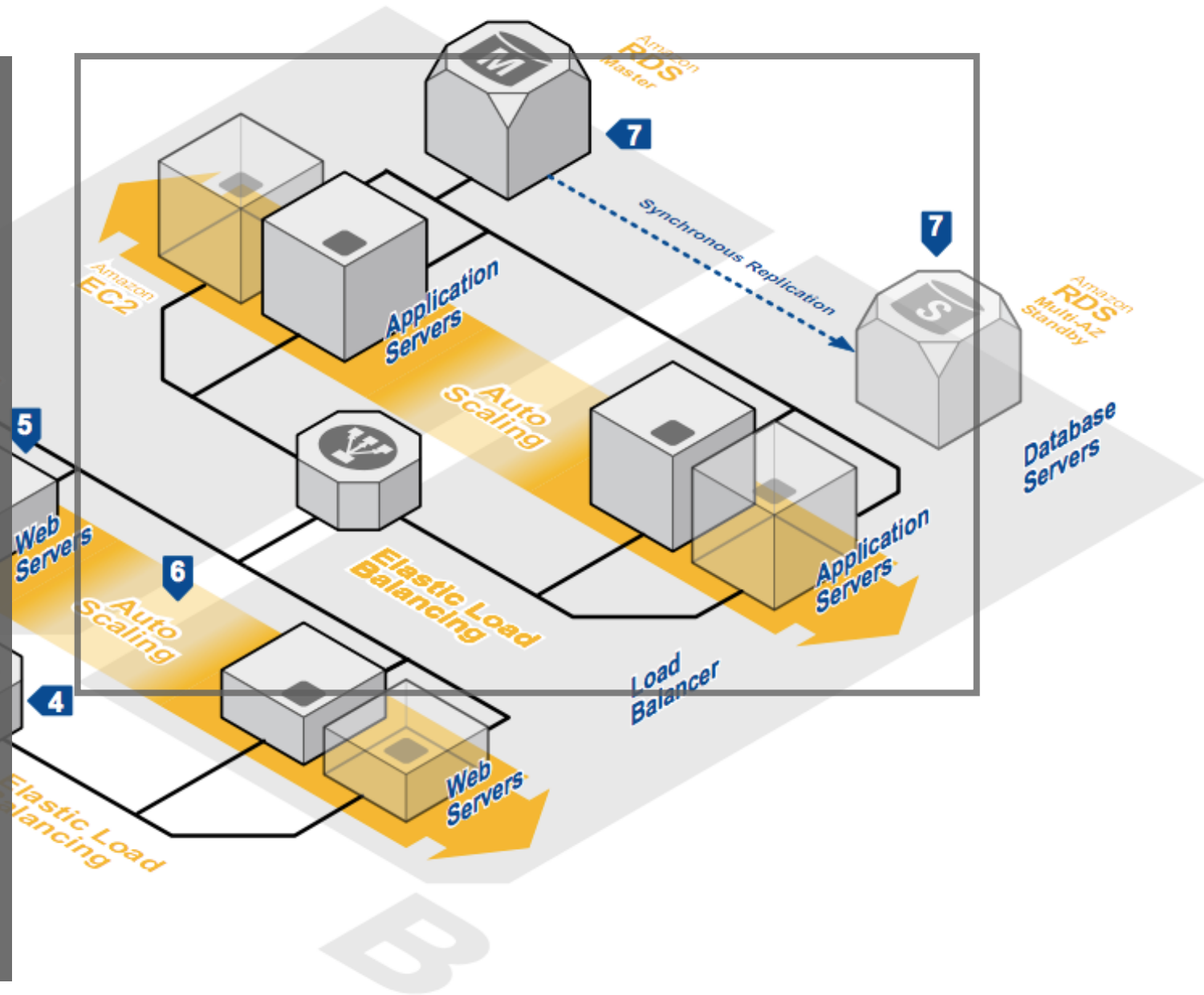
- CDN(AWS CloudFront)
 - 콘텐츠 전송 네트워크
 - 클라이언트에서 좀 더 빠르게 콘텐츠를 받아 보고 웹 서버에 요청을 분산하기 위해
- Static file server(AWS S3)
 - 정적 파일을 저장 / 제공하기 위한 서버
 - 여러 클라이언트나 서버에서 접근하여 사용

메인 웹 서버(NginX)의 부하를 막기 위해 구축

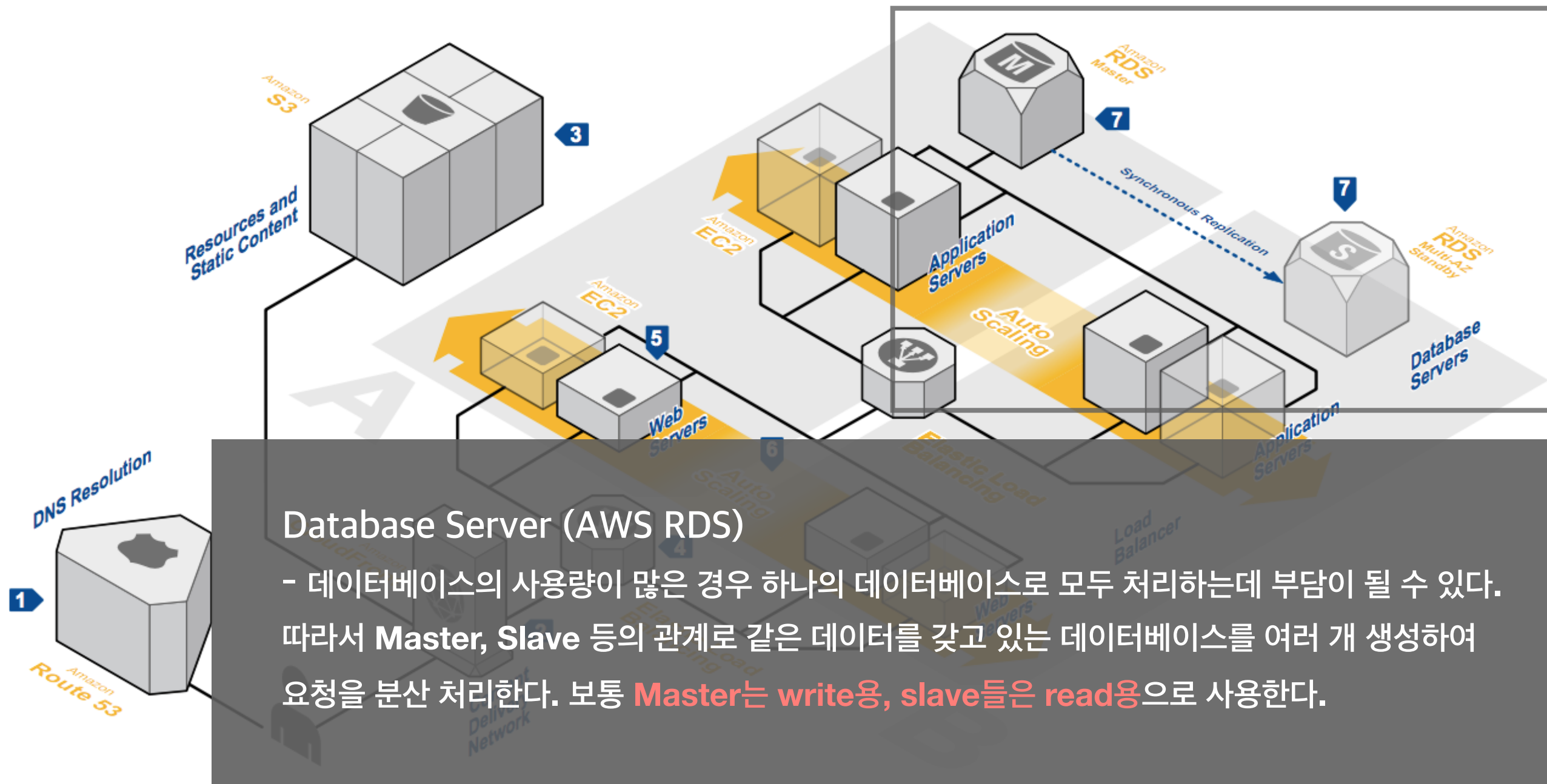
웹 서비스 아키텍처

WAS(Web application server)

- 동적 데이터를 제공하기 위한 서버
- Nginx를 proxy 서버로 설정하여 WAS 영역과 연결하는 경우가 많음
- 개발팀이 사용하는 언어, 프레임워크가 제각각이라 **DevOps 영역에서 효율적**으로 설계, 운영해야함



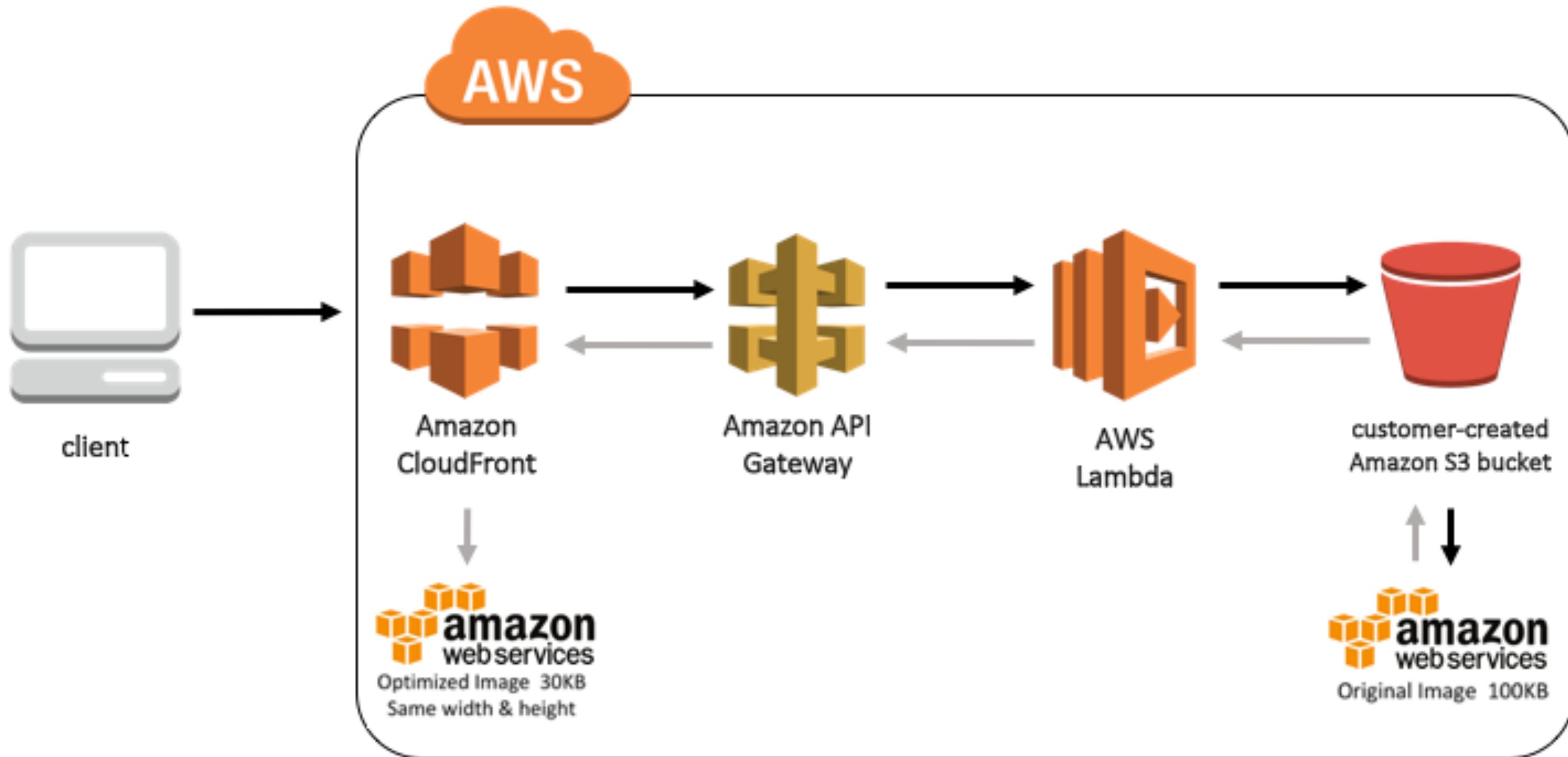
웹 서비스 아키텍처



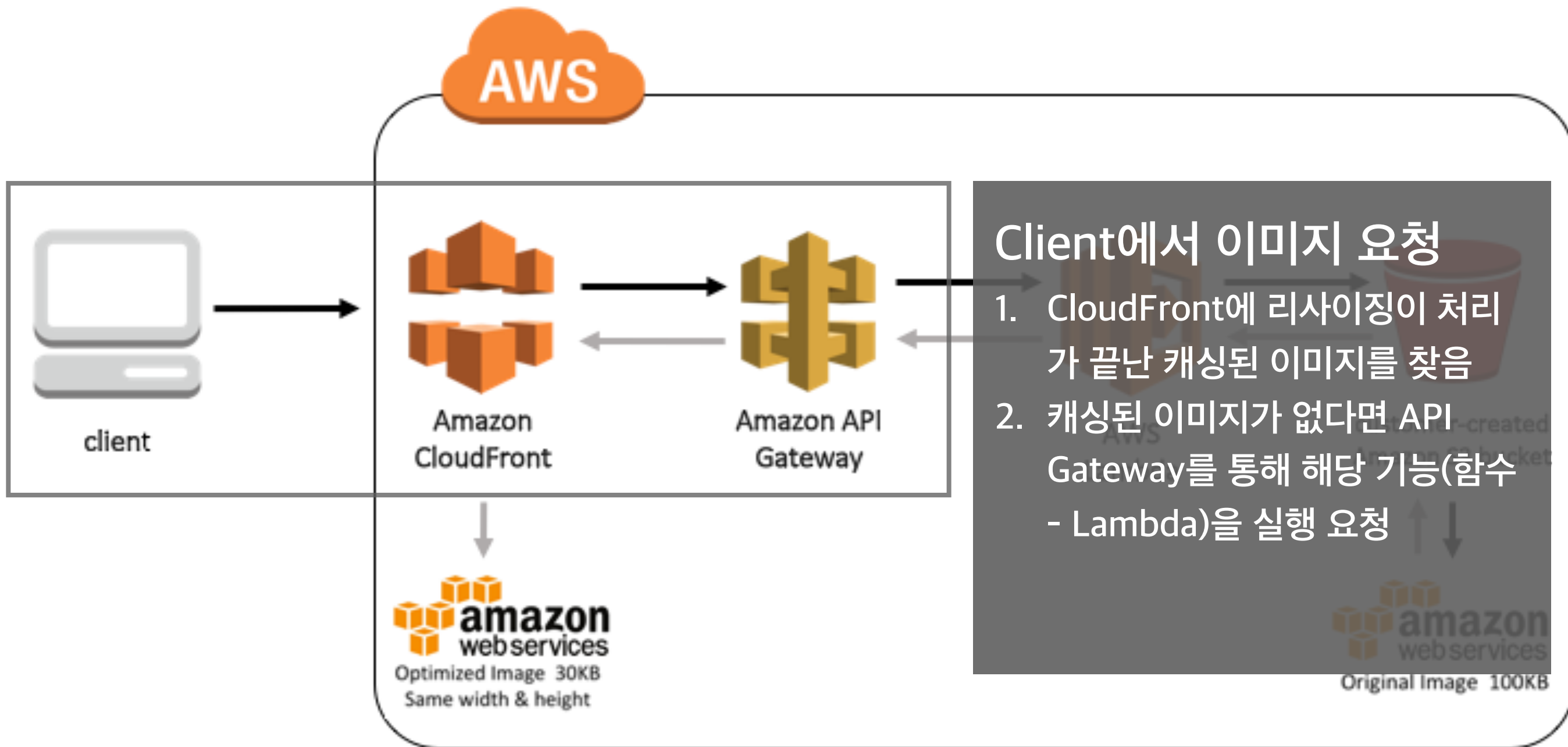
서버리스 아키텍처

- BaaS(Backend as a Service), FaaS(Function as a Service)
- 서버 로직을 별도로 써드 파티 업체가 구현한 것을 쓰거나, 코드는 작성하지만 그 실행은 외부 업체를 통해 하는 것
- 장점
 - 서버가 정말 필요한 경우에만 사용하여 비용을 절감할 수 있다.
 - 코드 외적인 내용을 처리할 필요가 없어서 운영 비용을 절감할 수 있다.
 - 스케일을 키우는데 걱정할 필요가 없다.
 - 불필요한 작업(서버 구성, 서버 관리)을 줄일 수 있다.
- 단점
 - 제공되는 환경 내에서만 일을 처리할 수 있기 때문에 튜닝이나 외부 라이브러리 등 접근 측면에서 제한이 있을 수 있다.

서버리스 아키텍처(이미지 리사이징)



서버리스 아키텍처(이미지 리사이징)

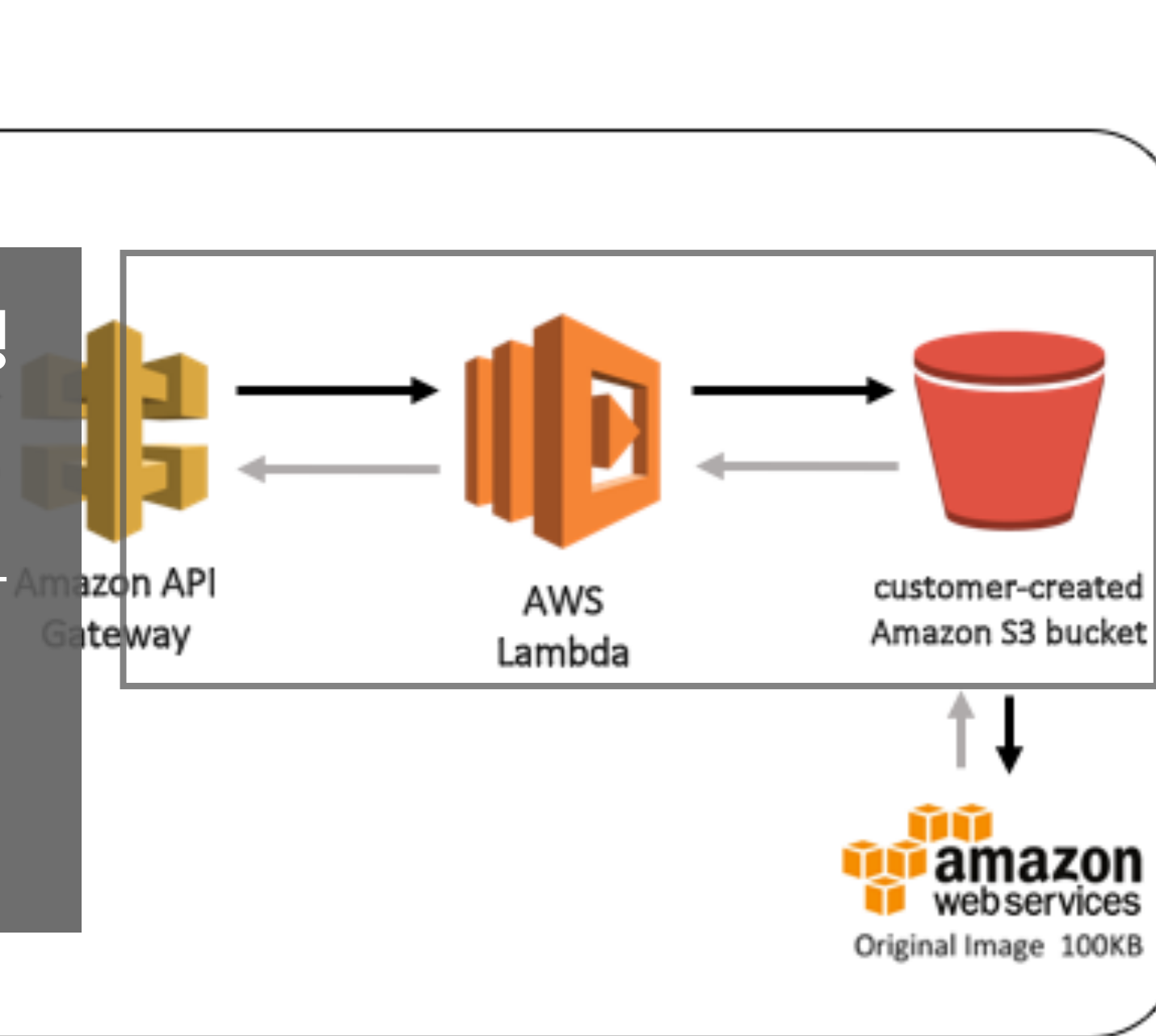


서버리스 아키텍처(이미지 리사이징)

Lambda(FaaS) 리사이징 기능 실행

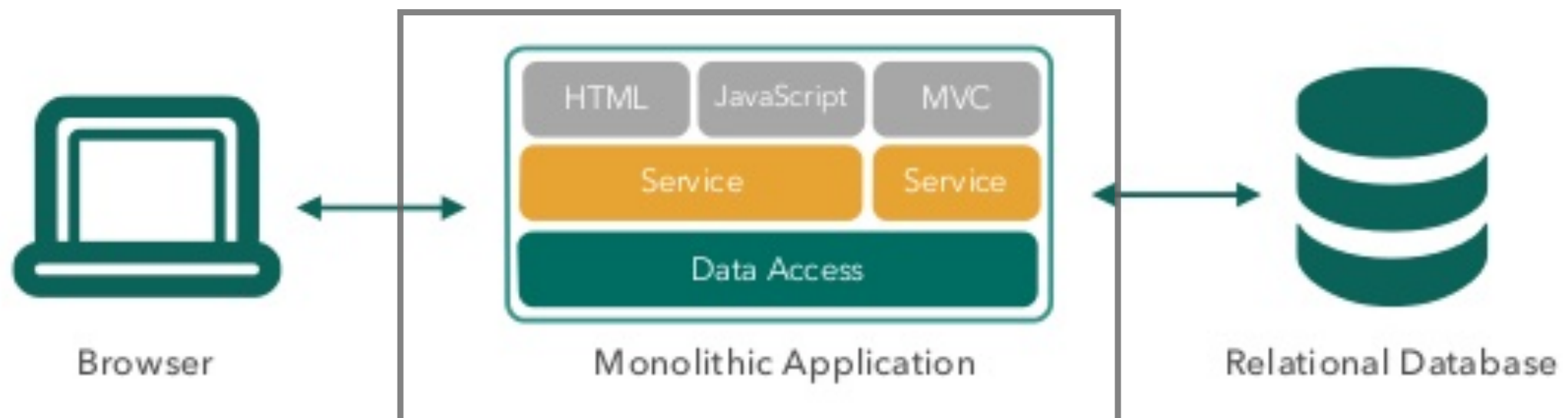
1. API Gateway에서 리사이징 기능을 하는 Lambda 트리거 실행
2. 기능을 수행하고 리사이징 된 이미지를 클라이언트에 전달 후 S3 버킷에 저장
3. CloudFront에서 클라이언트에게 전달하고 해당 이미지 캐싱


Optimized Image 30KB
Same width & height



모노리틱 아키텍처

Monolithic Architecture



제품 전체를 하나로 합쳐서 관리

모노리틱 아키텍처

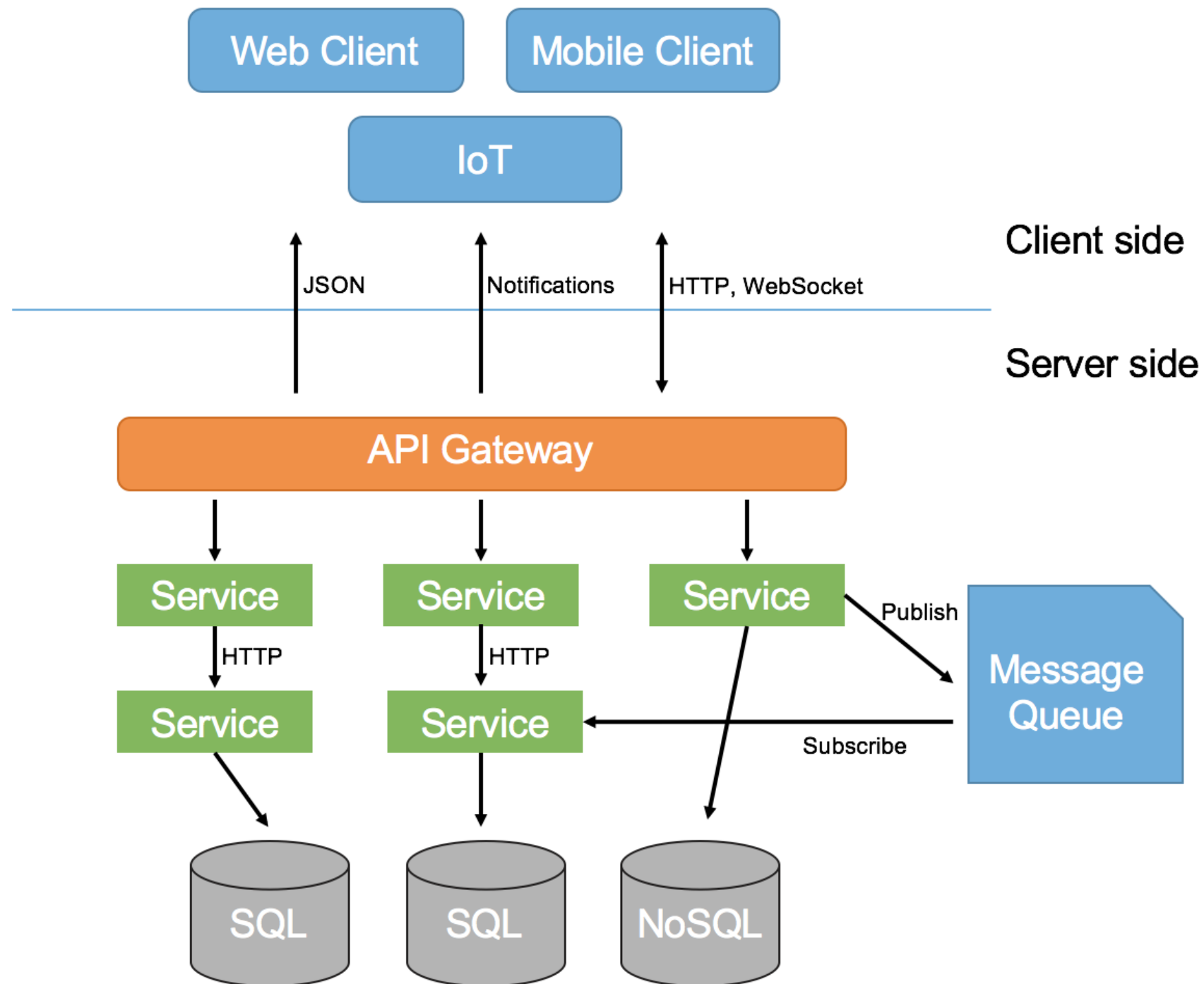
- 장점

- 서버를 구성하는데 간단하다.
- 기술 스택이 통일되어있기 때문에 개발하기도 쉽다.
- 하나의 어플리케이션만 배포하면 된다.
 - 관리하기 쉬워진다.
- 스케일 아웃에 유리하다.

- 단점

- 하나의 큰 덩어기로 코드가 되어있기 때문에 수정하는데 비용이 매우 커진다.
- 한번 정해진 기술, 버전을 굉장히 오래 써야 한다.

마이크로서비스 아키텍처



마이크로서비스 아키텍처

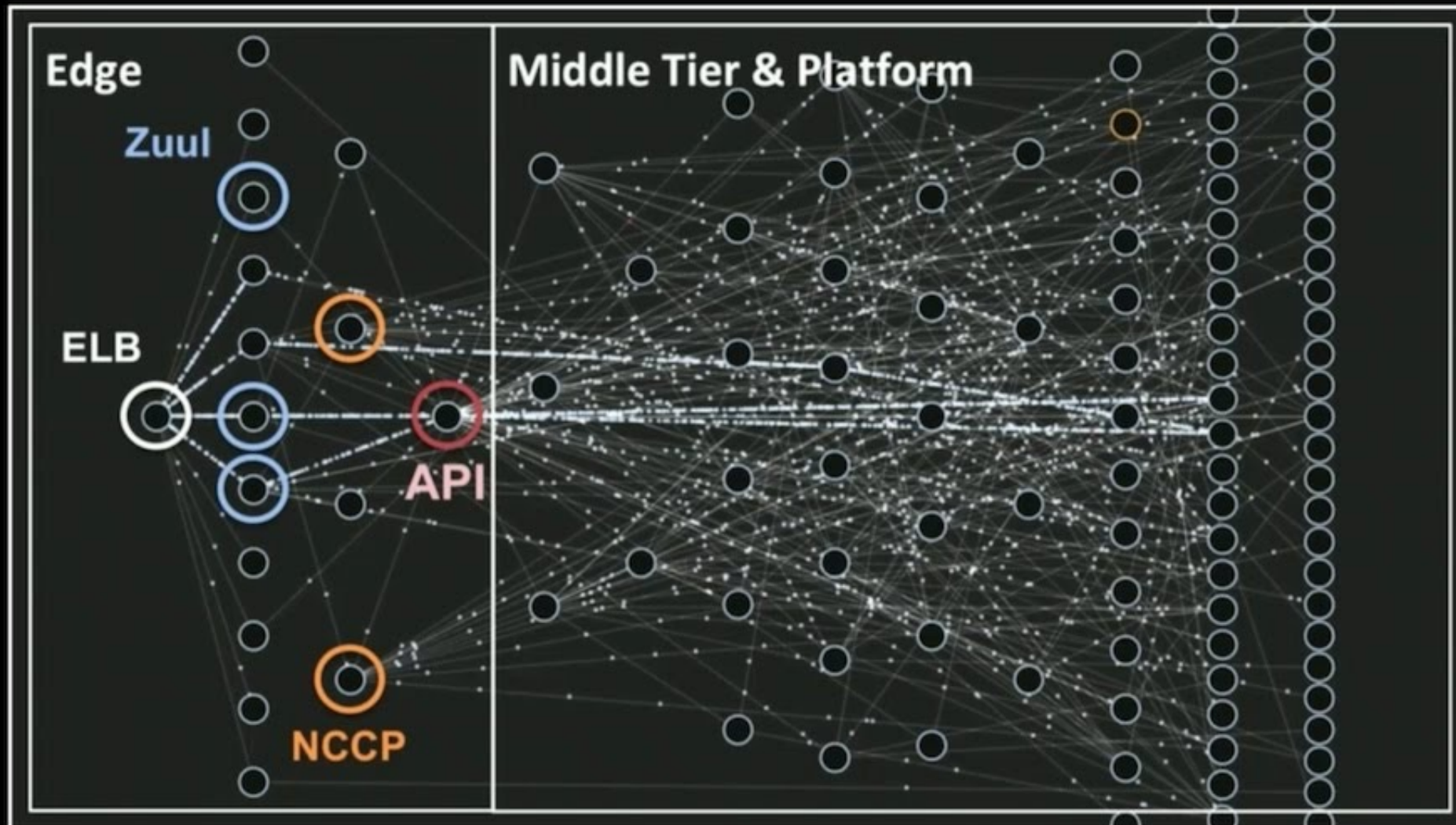
- 장점

- 제품이 하나의 기술, 버전에 묶여있지 않아도 된다.
- 도메인별 어플리케이션들이 느슨하게 연결되어있기 때문에 추후 아키텍처 변화에 유연하게 대처할 수 있다.
- 여러 사람들이 큰 규모의 제품을 개발할때 각 서비스간의 스펙만 맞추고 개발을 진행하면 되기 때문에 편하다.
- 전체 배포를 하지 않아도 되기 때문에 배포하기가 더 쉬워진다.
 - 페이스북 등 대형 서비스들도 하루에 수십, 수백번 배포가 가능한 이유다

- 단점

- 복잡도가 올라간다.
- 규칙 없이 막 사용했다간 사람보다 더 많은 기술 스택이 쌓일 수 있다.
- 서비스들간 연동 테스트하기가 힘들다
- 테스트, 배포, 모니터링 자동화가 잘 되어있어야만 가능하다.

마이크로서비스 아키텍처 - Netflix



Filmed at
QCon San Francisco 2016

Brought to you by
InfoQ

마이크로서비스 아키텍처

- 사람이 적고, 제품이 작을 땐 가볍고 빠르게 처리할 수 있는 monolithic이 더 낫다.
- 하지만 사람이 많아지고 제품의 규모가 커지는 경우에는 microservice는 선택이 아닌 필수가 된다.
- Microservice를 하기 위해선 배포, 테스트, 모니터링 자동화가 꼭 되어있어야 한다.

많이 사용하는 제품들 - Cloud service

- 서비스

- Amazon Web Service
- Google Cloud Platform
- Naver Cloud Platform
- Azure
- KT Cloud

- 차이점

- 어떤 서비스를 제공하는지
- 안정성, 비용, 제공되는 지역(리전)
- 법, 비용, 오랜 시간동안 직접 구축하여 씬 등의 이유로 직접 IDC에 구축하는 경우도 있음

많이 사용하는 제품들 - HTTP Web server

- 서비스
 - NginX
 - Apache
 - IIS (Windows 서버)
- 차이점
 - 제공되는 OS
 - 성능
 - 연동되는 Application Server

많이 사용하는 제품들 - WAS

- 서비스
 - Java
 - Tomcat, JBoss, Jetty
 - Node.js
 - PM2 (Process manager)
 - impress
 - Python(WSGI)
 - GUnicorn, uWSGI
 - Ruby
 - Phusion Passenger, Puma
- 차이점
 - 지원하는 언어

많이 사용하는 제품들 - Cache

- 서비스
 - in-memory - 데이터를 메모리에 저장, Read/Write
 - Memcached
 - Redis
- 차이점
 - 제공되는 기능
 - key/value store 넘어서 database, message broker 등
 - 제공되는 데이터 구조

많이 사용하는 제품들 - Monitoring

- 서비스
 - AWS Cloudwatch
 - NHN Pinpoint(<http://naver.github.io/pinpoint/>)
 - Opensource APM(Application performance monitoring)
 - ELK - Elastic search, Logstash, Kibana(<https://www.elastic.co/kr/elk-stack>)
 - Log monitoring
 - Grafana(<https://grafana.com/>)
 - Metric(cpu, ram, disk, etc...) monitoring

많이 사용하는 제품들 - CI/CD

- 서비스
 - **Jenkins CI** (<https://jenkins.io/>)
 - Circle CI (<https://circleci.com/>)
 - AWS Codepipeline

2주차 과제

AWS Route53을 통해 로드밸런서 DNS와 퍼블릭 도메인(도메인이 있을 경우에만) 연결

AWS ELB - sticky session 설정하기

Auto scaling group + ELB + NginX 실습 내용 삭제하고 복습하기