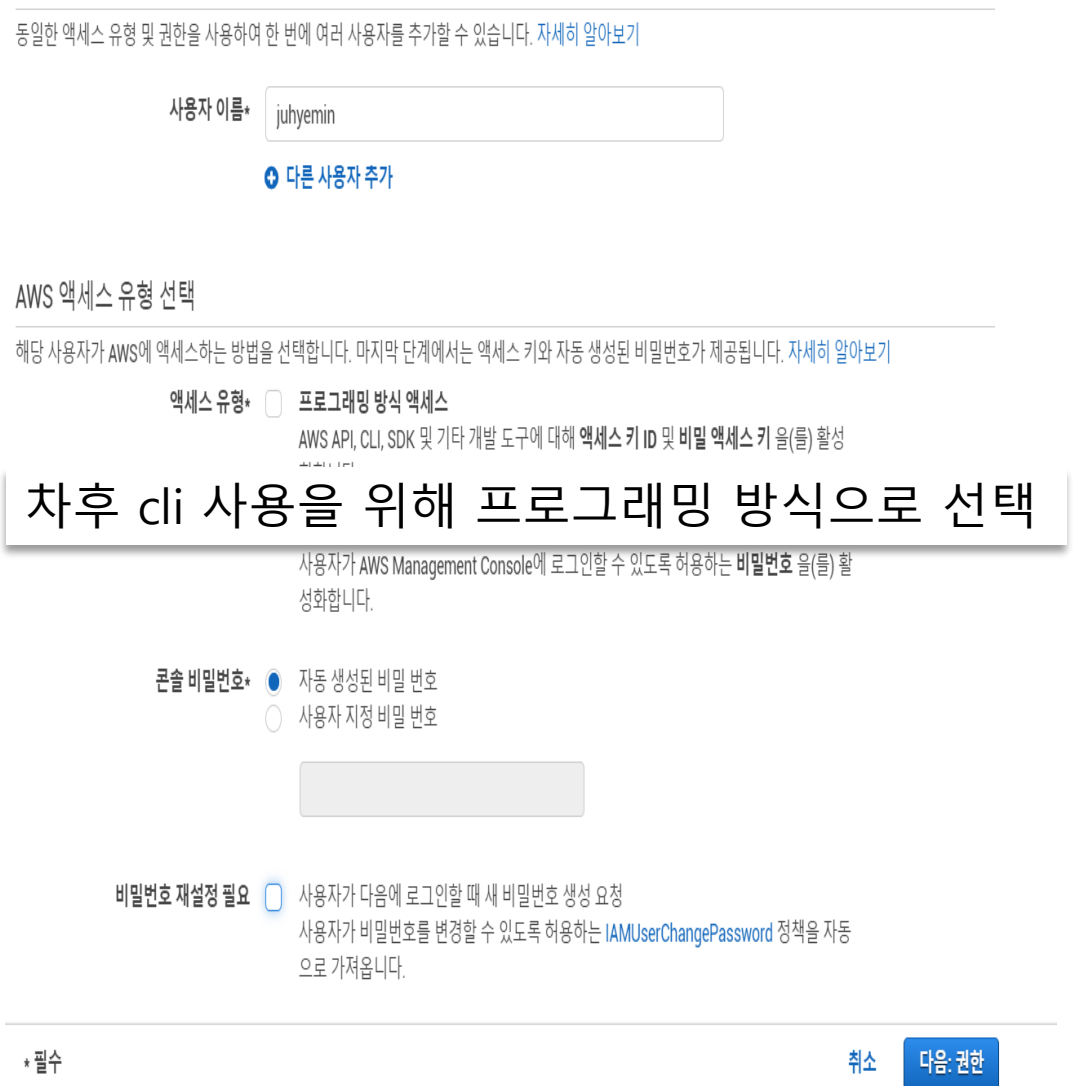
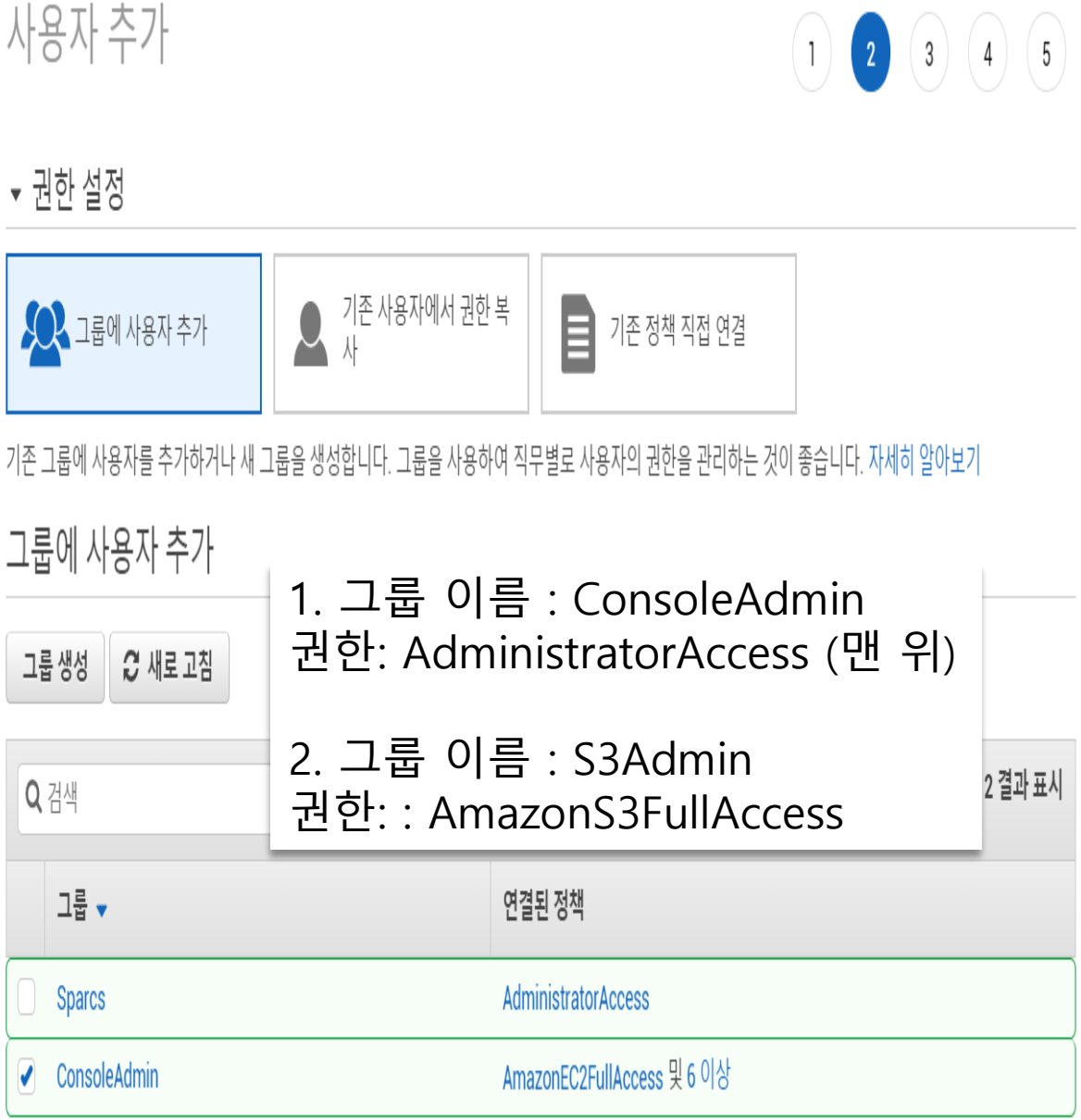
# AWS 실습

JUHYEMIN

# 1. 로그인 및 IAM 유저 설정



# 1. 로그인 및 IAM 유저 설정


Csv에 secret access id 와 pw가 존재

*pip install awscli --upgrade --user*

리눅스 -> awscli 파일 설치

IAM user 설정에서 로그인 활성화 클릭 -> 비밀번호 설정

# 1. 로그인 및 IAM 유저 설정

## 1) Aws credential 설정

*Aws configure*

➔ *Access key id, pw, region name(ap-northeast-2), output format(json) 설정*

## 2) Aws IAM 유저로 로그인

*앞서 설정한 로그인 비밀번호 (access key id, secret~ 와는 다른 것이다!!!)로 iam user 로그인*

2. 웹사이트 구동을 위한 Amazon Web Service

**EC2**

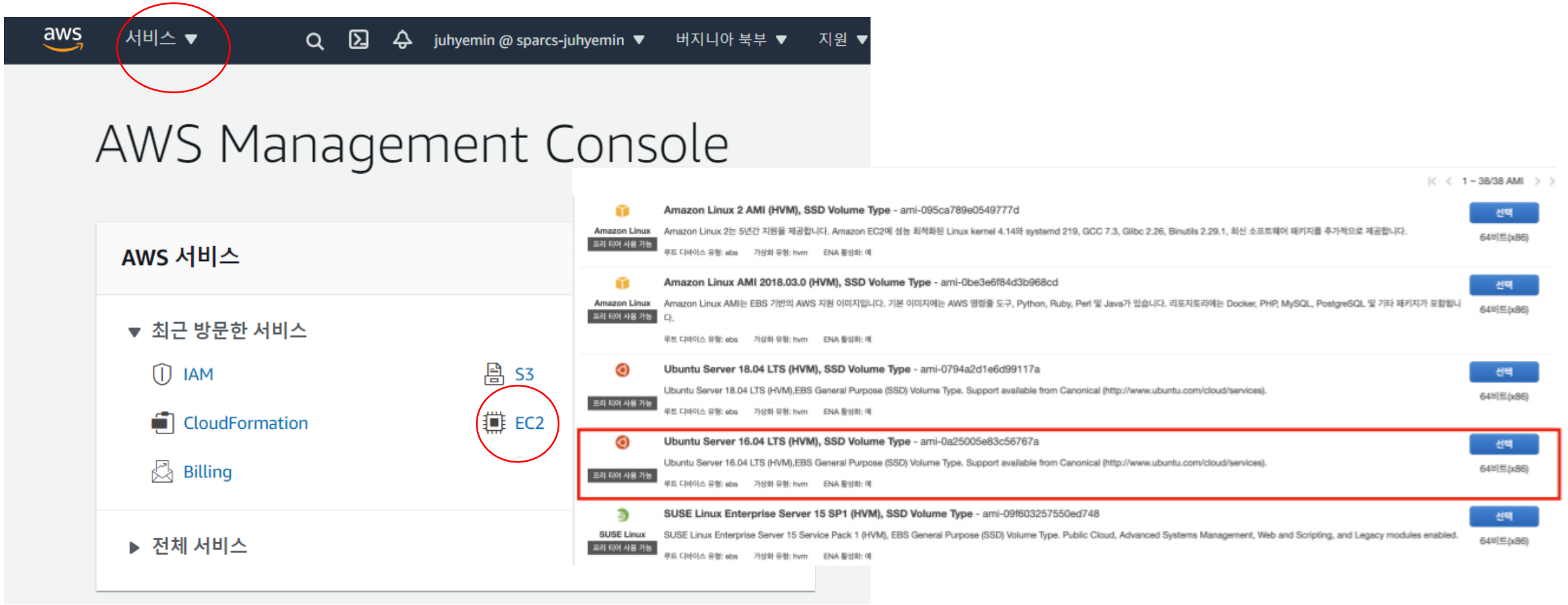
**S3**

**DocumentDB**

**ELB**

**Auto Scaling Group**

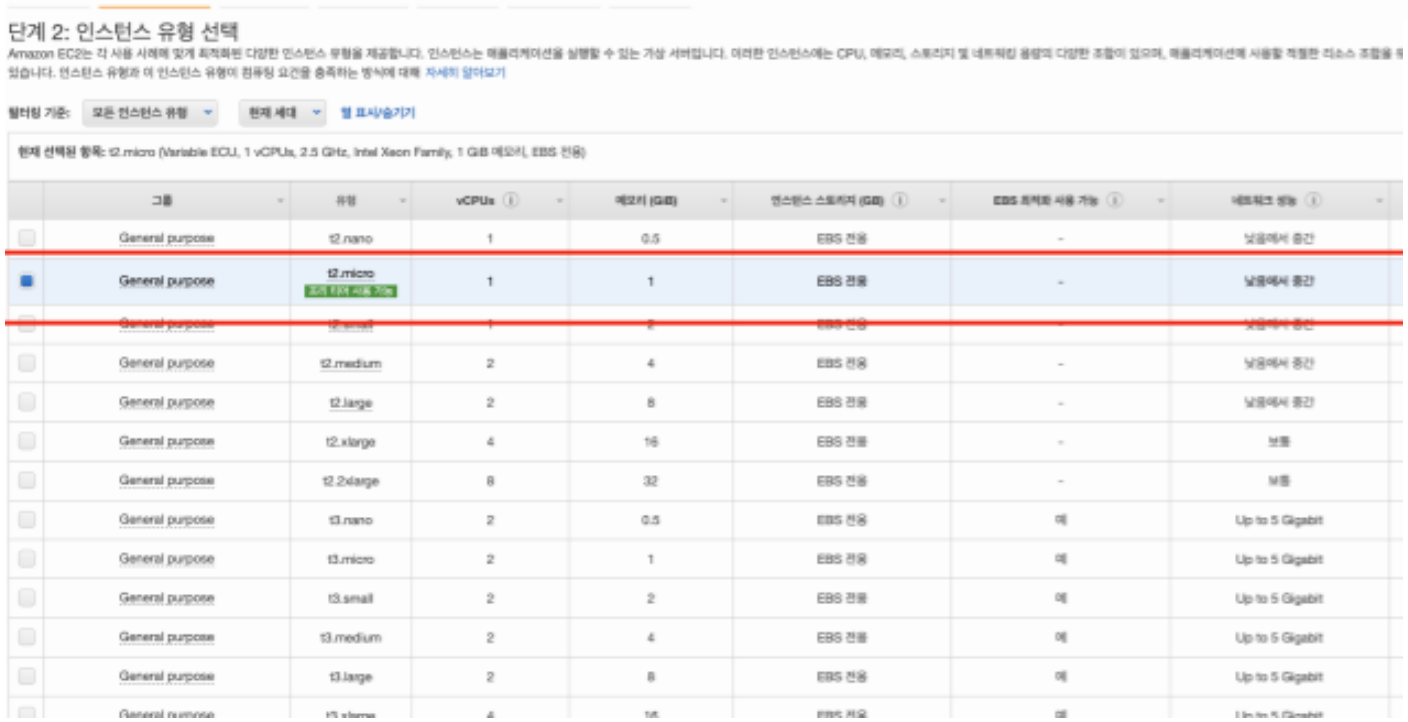
2-1. EC2

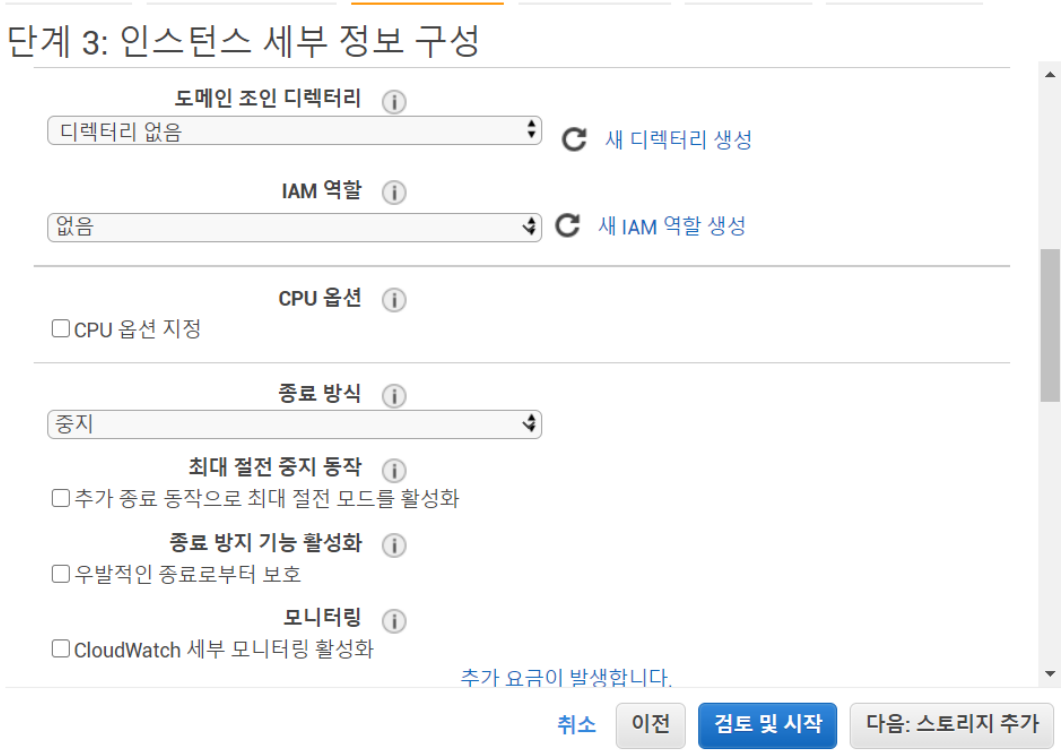



**1) 단계1: Xenial 선택**

서비스 -> EC2 -> 인스턴스 -> 인스턴스 시작

# 2-1. EC2




**2) 단계2: Free Tier 선택**

**3) 단계 3: 인스턴스 세부 정보는 기본으로 설정**

# 2-1. EC2

## 단계 6: 보안 그룹 구성

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

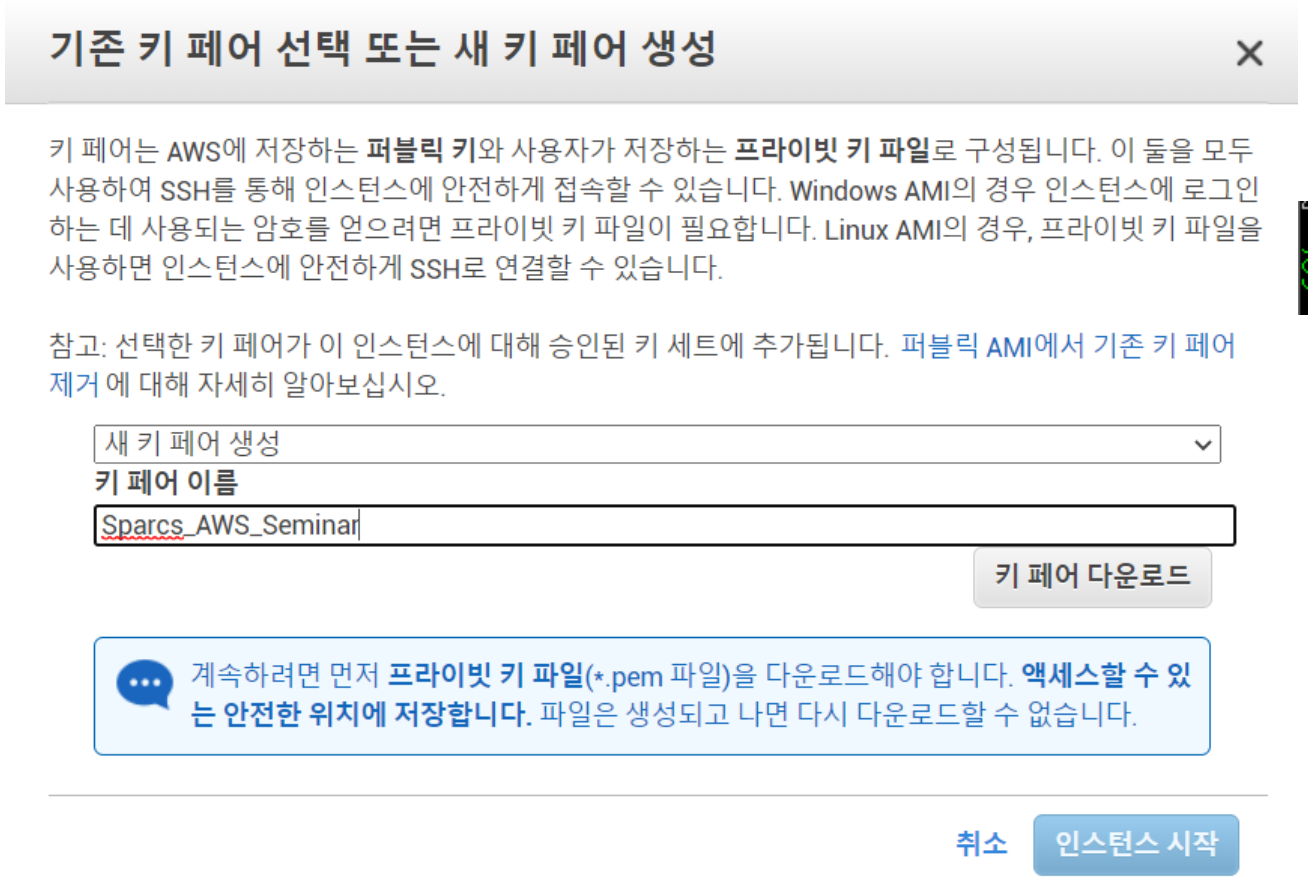
**보안 그룹 이름:** launch-wizard-1

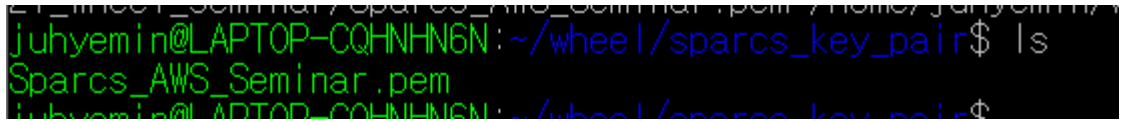
**설명:** launch-wizard-1 created 2021-01-03T23:20:05.200+09:00

| 유형 | 프로토콜 | 포트 범위 | 소스 | 설명 |
|---|---|---|---|---|
| SSH | TCP | 22 | 사용자 지정 0.0.0.0/0 | 예: SSH for Admin Desktop |
| 사용자 지정 T( | TCP | 80 | 사용자 지정 0.0.0.0/0 | 예: SSH for Admin Desktop |
| 사용자 지정 T( | TCP | 433 | 사용자 지정 CIDR, IP 또는 보안 그룹 | 예: SSH for Admin Desktop |

규칙 추가

**3) 단계6: 80번, 433번, 3000번 포트 추가 설정 for web server access**
**소스 : 위치 무관 선택 (웹페이지는 모두에게 이용 가능해야 해서!)**



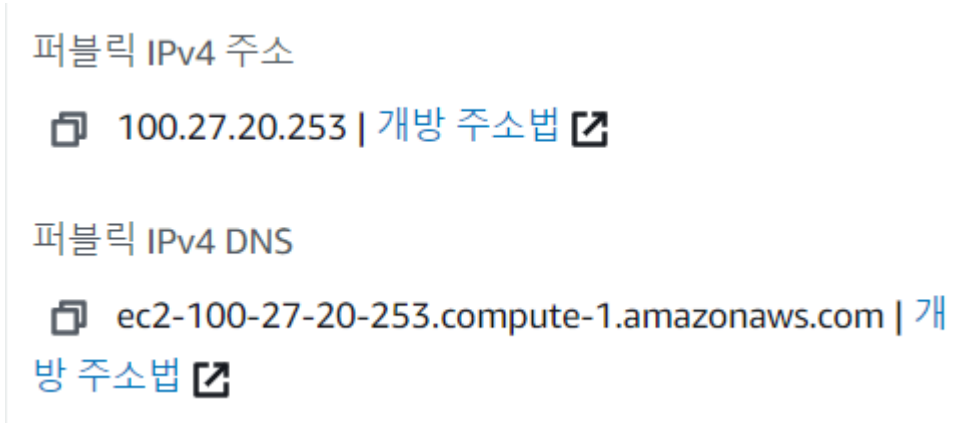

- 이후 putty로 .pem 을 저장
  - 또는 리눅스에 저장

**4) 새 키 페어 생성**

**반드시 은밀한 곳에 저장해야 함!!!!!**

2-1. EC2

**1) EC2 인스턴스에는 동적 IP가 할당됨**



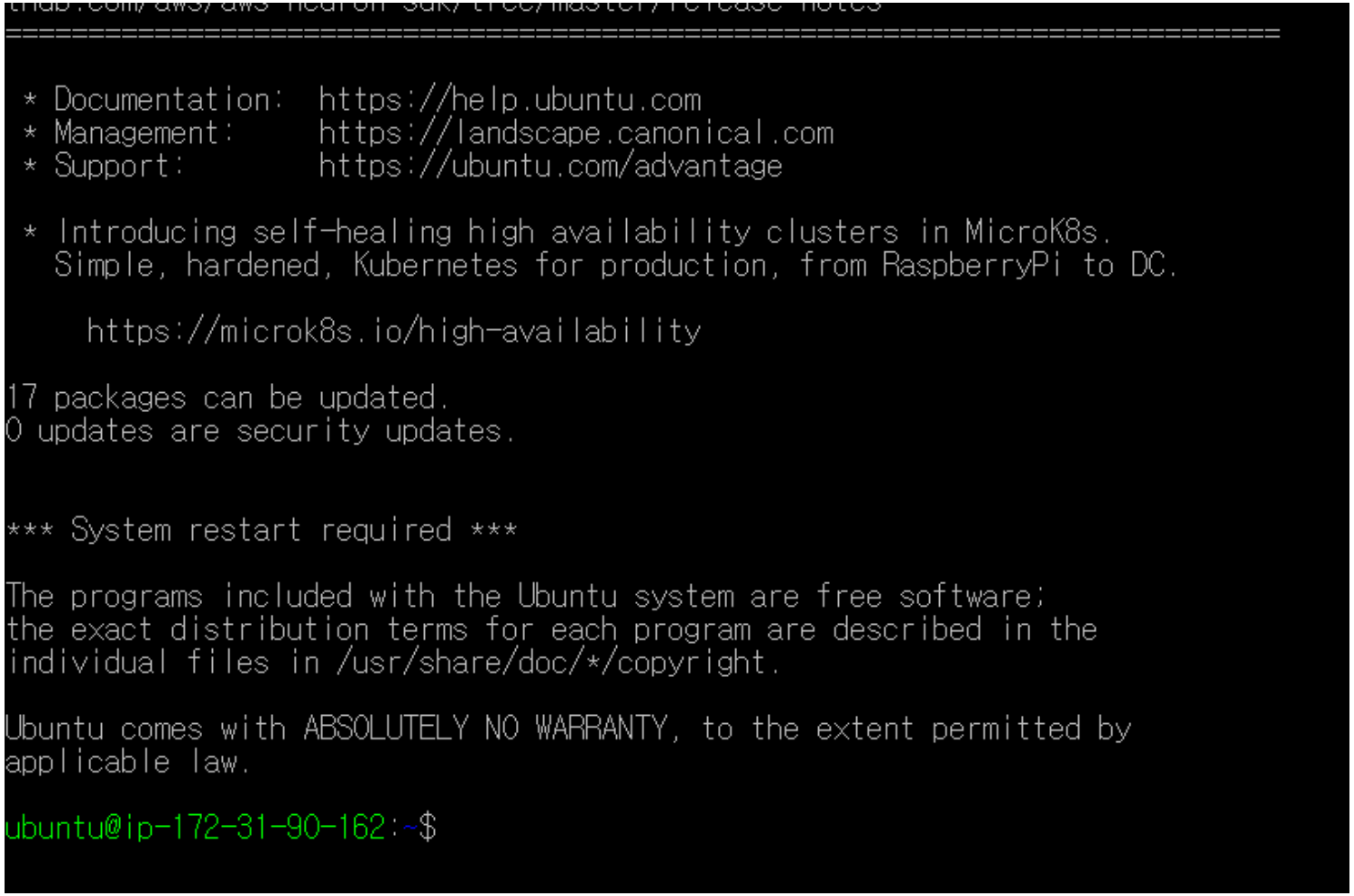

**2) 동적 대신 정적 IP 할당**

**3) Elastic IP -> 새 주소 할당 -> 주소 연결 -> 인스턴스의 private IP에 연결됨**

**4) 인스턴스 클릭 -> IPv4 퍼블릭 IP 복사 -> ssh ubuntu@[public ip] –I [pem 경로]**

2-1. EC2

**만약 ssh 접속 시 Warning이 뜬다면 chmod 400 [파일 이름]을 해주면 됨.**



2-1. EC2

```
ubuntu@ip-172-31-31-199:~$ export NVM_DIR="$HOME/.nvm"
ubuntu@ip-172-31-31-199:~$ [ -s "$NVM_DIR/nvm.sh" ] && ₩. "$NVM_DIR/nvm.sh"  # This loads nvm
ubuntu@ip-172-31-31-199:~$ [ -s "$NVM_DIR/bash_completion" ] && ₩. "$NVM_DIR/bash_completion"
  # This loads nvm bash_completion
ubuntu@ip-172-31-31-199:~$ nvm --version
0.34.0
```

**1) NVM 설치**

curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.34.0/install.sh | bash

**2) Node 설치**

nvm install –lts
※npm install –g yarn으로 패키지 관리자 설치

**3) Create React App -> 리액트 어플리케이션 생성**

npx create-react-app my-react-app
Cd ./my-react-app
yarn && yarn build

**4) Express generator -> express 서버 생성**

Npx express-generator myserver
Cd myserver
yarn

**5) Node Process Manager(PM2) 설치**

Npm install –g pm2

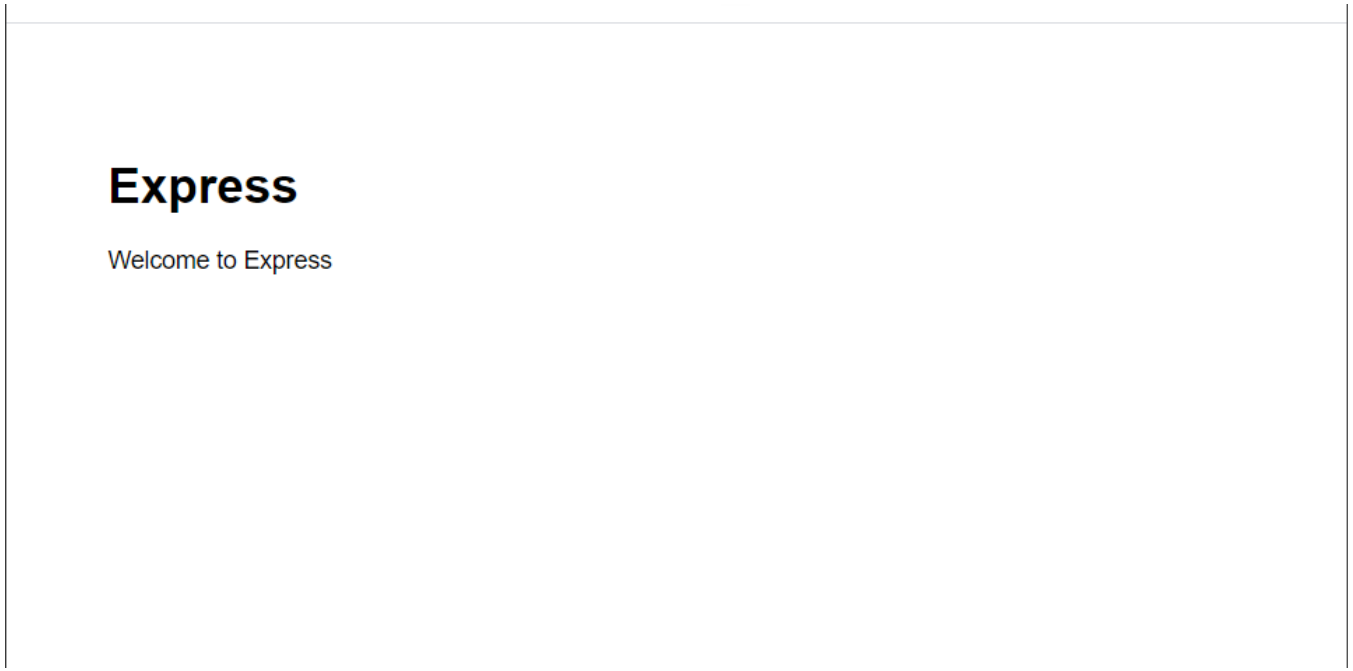
2-1. EC2 : 서버 배포

**1) Myserver 폴더에 접근 -> 서버 pm2로 실행**

DEBUG=myserver:* pm2 start bin/www
**Pm2 list 명령어 사용 -> 서버 작동 체크**

**2) Express 서버는 3000번 포트에서 실행됨**

http://[public_ip]:3000



2-1. EC2 : 프론트 배포

### 1) React 배포

Nginx로 static file 서빙 (nginx는 정적 파일 서빙에 이점을 가짐)

### 2) Nginx 설치

Sudo apt update
Sudo apt install nginx –y
Sudo service nginx status


Nginx는 80번 포트를 사용 -> http://[public_ip]으로 확인

# 2-1. EC2 : 프론트 배포 : nginx 설정

### 1) 기존 설정 파일 삭제

Cd /etc/nginx/
Sudo rm sites-enabled/default

### 2) 새 설정 파일 생성

sudo vi sites-available/my-react-app

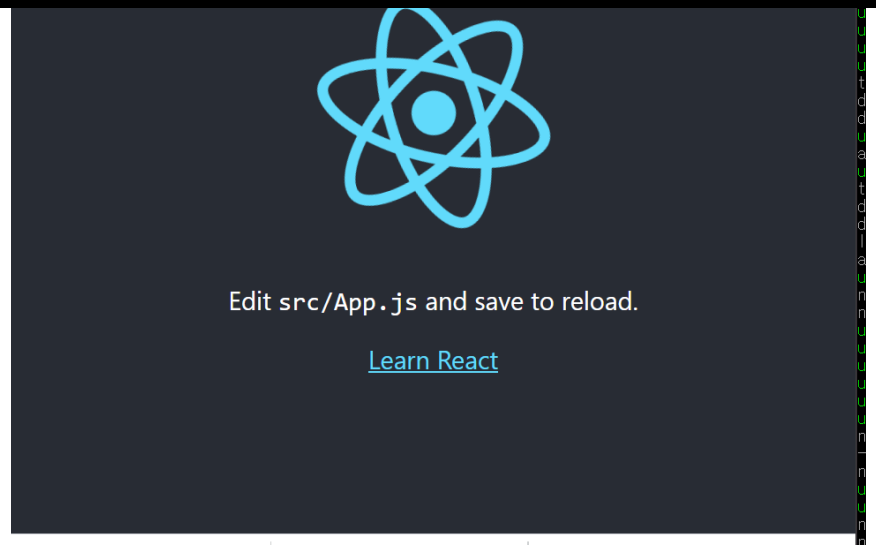
```
server{
    listen 80;
    server_name _;

    root /home/ubuntu/my-react-app/build;
    index index.html;

    location / {
        try_files $uri $uri/ =404;
    }

    location /api {
        proxy_set_header X-Forwarded-For $remote_addr;
        proxy_set_header Host $http_host;
        proxy_pass "http://127.0.0.1:3000";
    }
}
```

### 3) 링크

sudo ln -s /etc/nginx/sites-available/my-reactapp /etc/nginx/sites-enabled/my-react-app

### 4) reload

sudo nginx –t
sudo service nginx reload



2-1. EC2 : 배포 끝!

**1) 라우트 수정 (모든 라우트 앞에 /api 붙이기)**

Cd /home/ubuntu/myserver
Vim app.js

App.use('/api', indexRouter);

App.use('/api/users', usersRouter);

**2) Nginx에서 /api 로 시작하는 요청은 서버로 보냄**

```
server {
    listen 80;
    server_name _;

    root /home/ubuntu/my-react-app/build
    index index.html

    location / {
        try_files $uri $uri/ =404;
    }
    location /api {
        proxy_set_header X-Forwarded-For $remote_addr;
        proxy_set_header Host $http_host;
        proxy_pass "http://127.0.0.1:3000";
    }
}
```

3. S3 : S3로 프론트 배포

**1) AWS cli 설치** https://docs.aws.amazon.com/ko_kr/cli/latest/userguide/cli-chap-install.html

+) Sudo apt install python3-pip 으로 pip 설치
※ aws cli란? ) 명령어 기반으로 aws 서비스를 관리할 수 있는 통합 도구

**2) AWS CLI 다운로드**

pip3 install awscli --upgrade –user

※만약 which aws 시 aws를 찾을 수 없다면 다음 코드를 따라해라
echo 'PATH="/home/ubuntu/.local/bin:$PATH"' >> .bashrc
exec "$SHELL"

**3) AWS CLI 설정** 앞서 IAM 유저를 생성하면서 받은 CSV파일에서 access_key, secret_key를 이용

Cd ~/
mkdir .aws
vi .aws/credentials

[default]
aws_access_key_id=[access_key]
aws_secret_access_key=[secret_key]

3. S3 : S3로 프론트 배포

**4) Config**

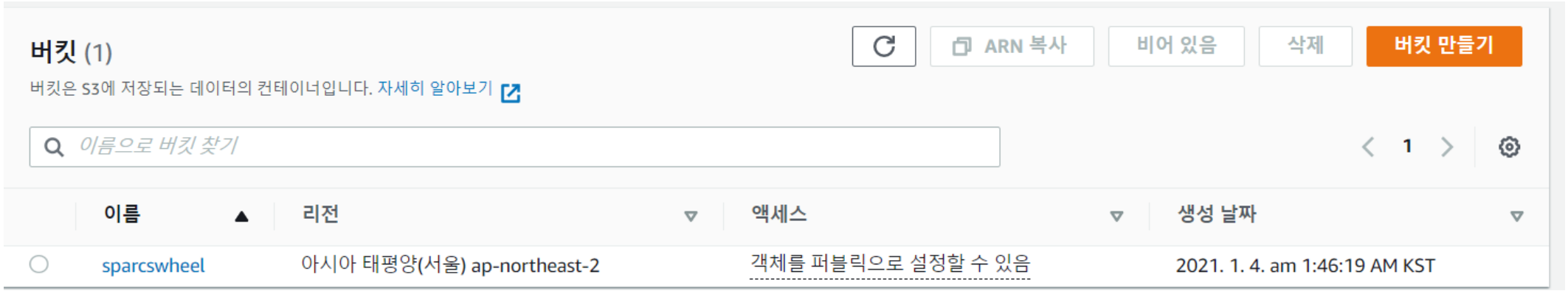
Vi .aws/config

```
[default]
region=ap-northeast-2
output=json
```

# 3. S3 : 버킷 생성

## 1) 버킷

서비스 -> S3 -> 버킷 만들기 -> 버킷 이름 (ex: wheelseminar) -> 서울 리젼 선택 -> 권한설정 : 퍼블릭 엑세스 차단 선택 해제 => 만들기



3. S3 : 버킷 생성

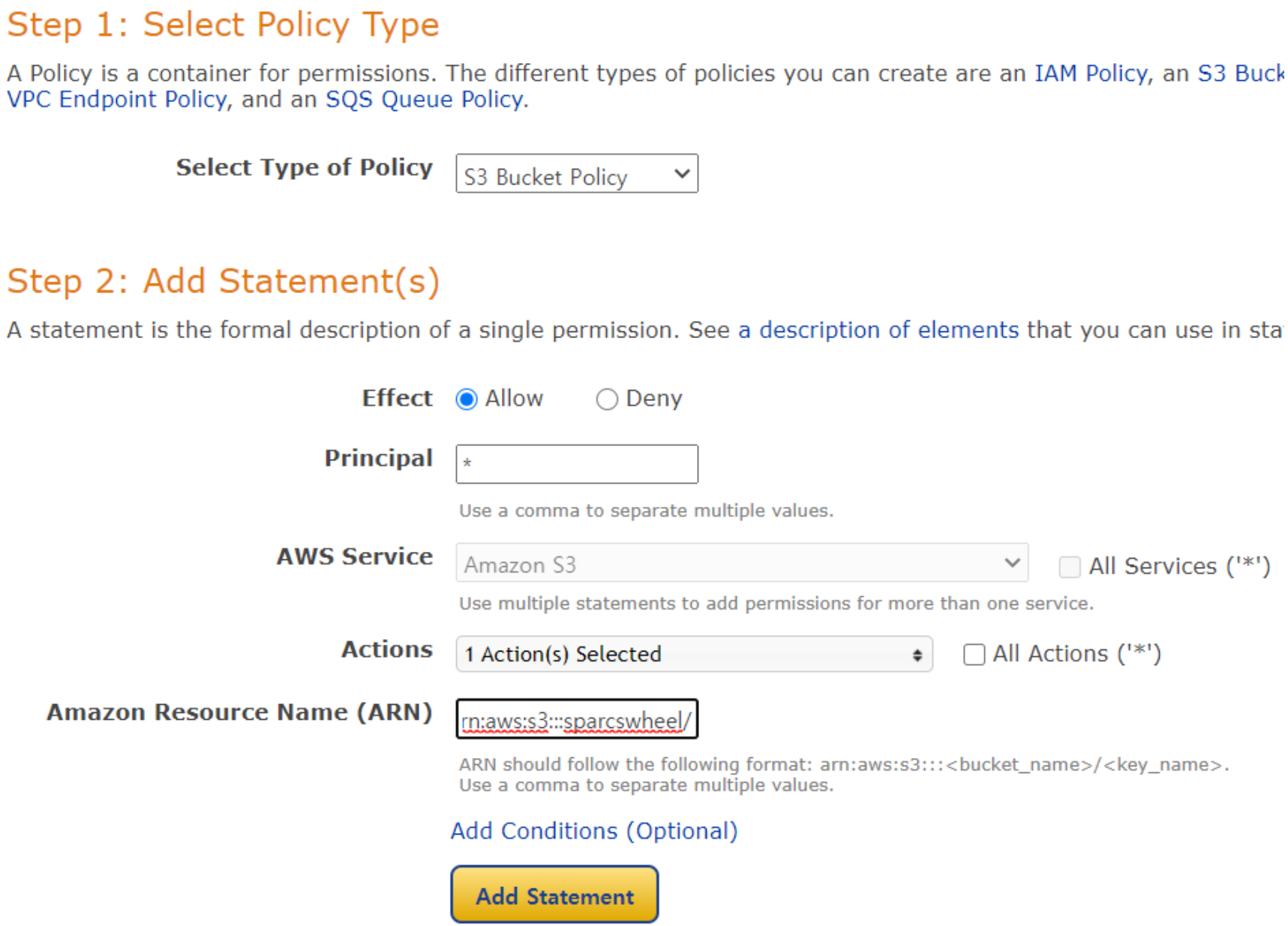
## 2) 버킷의 정적 파일 설정

생성된 버킷 선택 -> 속성 -> 정적 웹 사이트 호스팅 ->
인덱스 문서 : index.html, 오류 문 서 : index.html -> 저장

## 3) 버킷 권한 설정

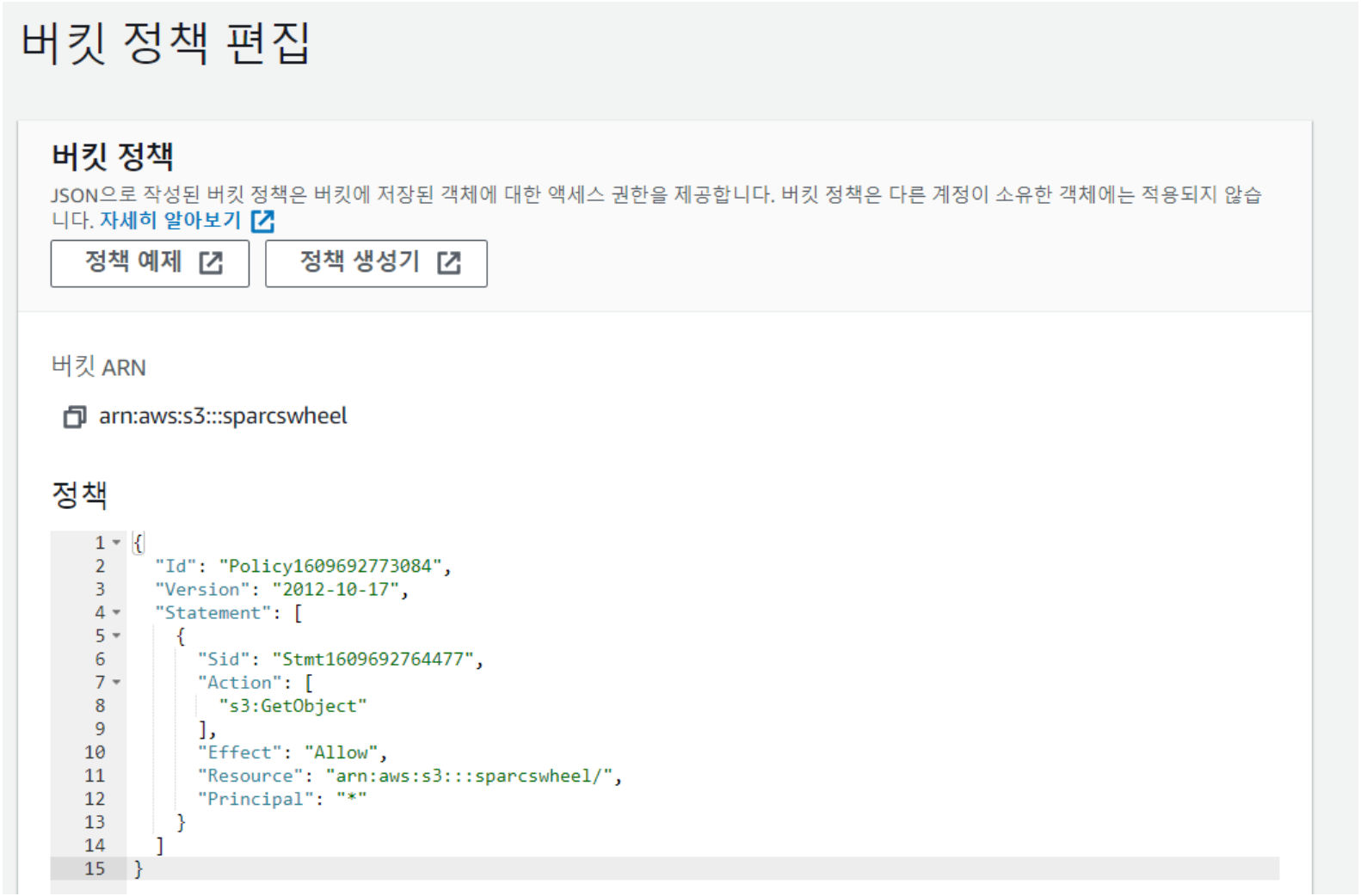
권한 -> 버킷 정책 -> 정책 생성기 -> ...

# 3. S3 : 버킷 생성 – 정책 설정



# 3. S3 : 버킷 생성 – 정책 설정

Step3의 generate policy 를 클릭하면 json documen 팝업창 발생 -> 버킷 정책 편집에 복붙



3. S3 : 버킷을 EC2로 접근

aws s3 sync my-react-app/build s3://[방금_정책_설정한_버킷]
https://[bucket_name].s3.ap-northeast-2.amazonaws.com

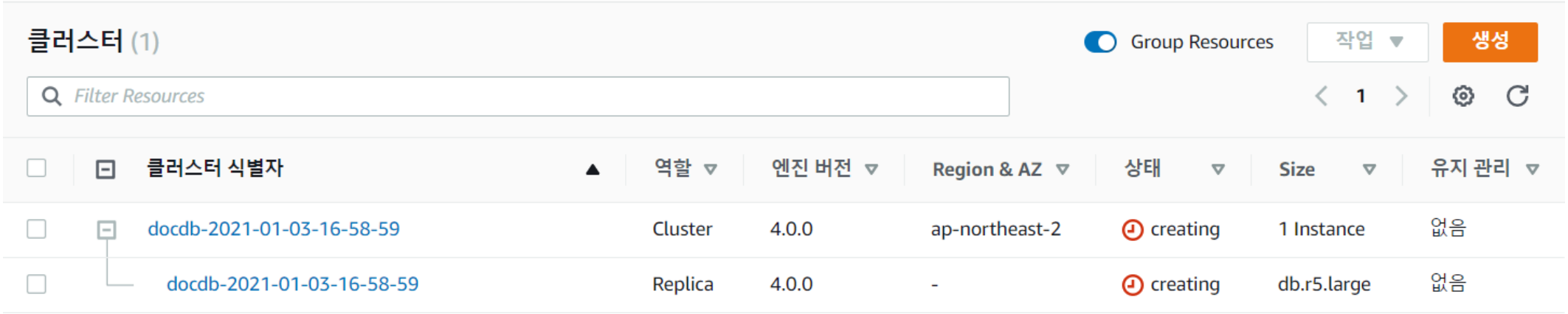
# 4. Document DB 생성

## 1) 개념

Document DB는 클러스터 & 인스턴스로 나누어 처리
저장 -> 클러스터 : 데이터를 관리하는 저장소
처리 -> 인스턴스 : 클러스터의 데이터를 읽고 쓰는 역할

## 2) 생성

서비스 -> Amazon Document DB -> 시작 -> ID, PW 입력-> 인스턴스 개수 1개로 설정
-> 고급 설정-삭제 보호 활성화 설정 해제 -> 클러스터 생성 클릭



4. Document DB 접속 : mongodb-client 설치

1) EC2 접속 후 다운로드

다음 링크를 따라하면 mongo 설치 가능
https://docs.mongodb.com/manual/tutorial/install-mongodb-on-ubuntu/

2) EC2에서 documentdb 접속 위한 보안그룹 재설정

EC2 보안그룹의 default 클릭 -> 인바운드 규칙 편집 -> 규칙 추가 : [유형: 모든 트레픽 소스, 소스: EC2 보안그룹 (ex: launch-wizard-숫자)] -> 규칙 저장

3) TLS 다운 후 mongo shell 연결

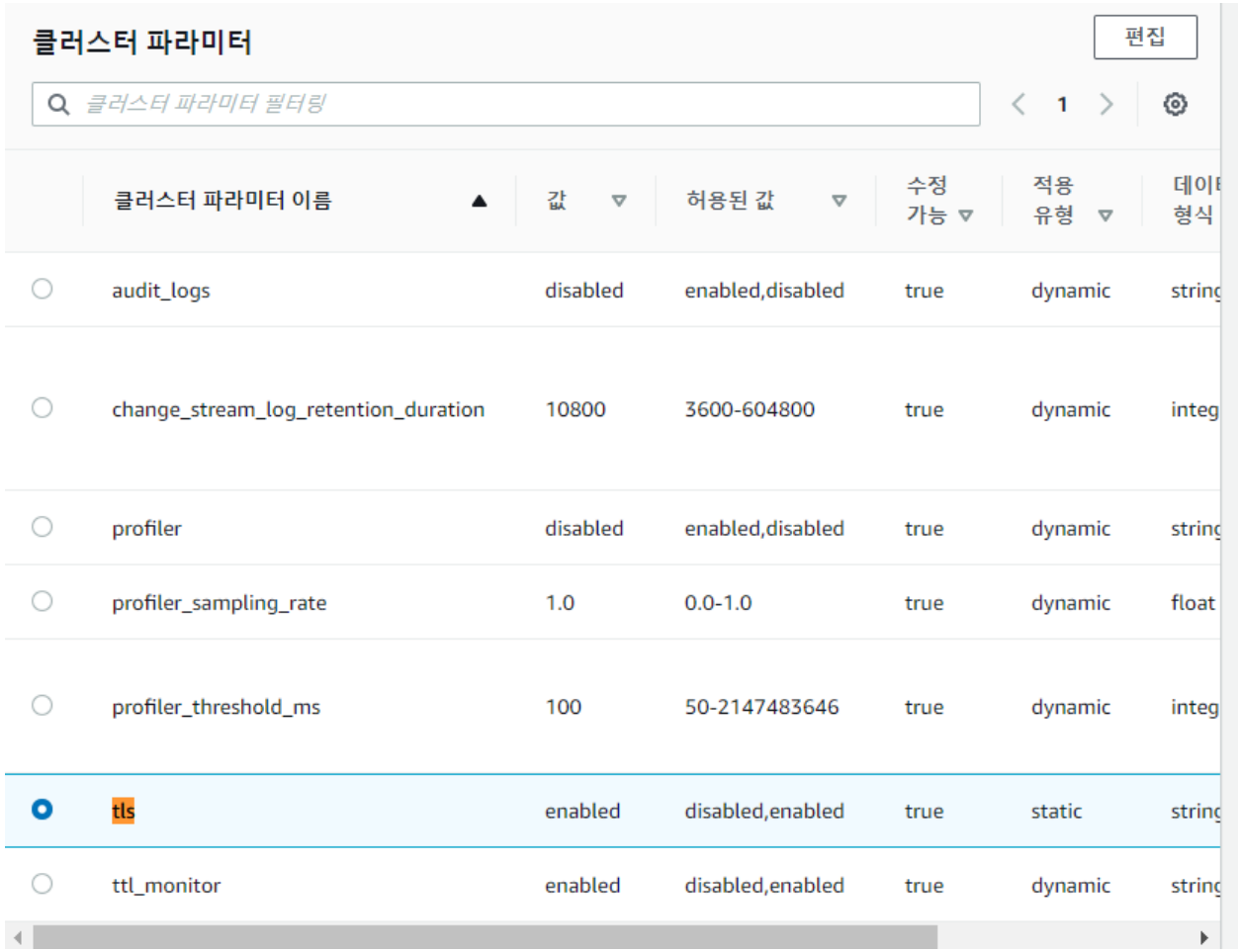
Amazon DocumentDB 퍼블릭 키 다운로드 (TLS 연결을 위해 필수)
Wget https://s3.amazonaws.com/rds-downloads/rds-combined-ca-bundle.pem

DocumentDB 클러스터 클릭 -> 상세 정보: 연결에서 mongo 쉘 연결 방식 복사 -> EC2에서 실행

4. Document DB : TLS -> disabled

*서버에서 document DB 연결 시 TLS 켜져 있으면 귀찮음*

AWS 콘솔 -> DocumentDB -> Parameter Group -> 생성 -> 생성된 paramter group 클릭 -> TLS 설정을 disabled 로 편집



4. Document DB : Node 연결

EC2 인스턴스 접속 -> git clone https://github.com/jungdj/express-mongo-boilerplate
-> cd express-mongo-boilerplate -> yarn

cp config/.env.example config/.env.development

vi config/.env.development

➔ config 값을 아래와 같이 입력하자.
MONGODB_CONNECTION_URL=mongodb://<username>:<password>@<아까 만들었던 aws document 이름>.node.<VPC장소> docdb.amazon.com/<만들고 싶은 db 이름>?~~

Ex)
mongodb://*<sample-user>*:*<password>*@sample-cluster.node.us-east-1.docdb.amazonaws.com/sample-database?ssl=true&replicaSet=rs0&readPreference=secondaryPreferred

pm2 start ecosystem.config.js --env development

4. Document DB : Node 연결 확인

- curl -X GET http://localhost:6001/user/list 입력 시
  []와 같은 response 올 것

- curl -X POST http://localhost:6001/user -d '{ "firstName": " hyemin", "lastName": "Ju" }' 등등

4. Document DB : Node 배포

1) EC2의 myserver 폴더 들어감 -> vim app.js

2) 모든 라우터 앞에 /api 붙이기
Ex) app.use('/api', routes);

➔ Nginx 에서 /api로 시작하는 모든 요청을 서버로 보내기 위함.

3) cd /etc/nginx 로 가서 sites-available/my-react-app 내용 중 location /api 부분을 아래와 같이 바꿔준다.

```
location /api {
    proxy_set_header X-Forwarded-For $remote_addr;
    proxy_set_header Host $http_host;
    proxy_pass "http://127.0.0.1:3000";
    proxy_pass "http://127.0.0.1:6001";
}
```

5. ELB

### 1) 개념

Workload를 여러 시스템으로 분산

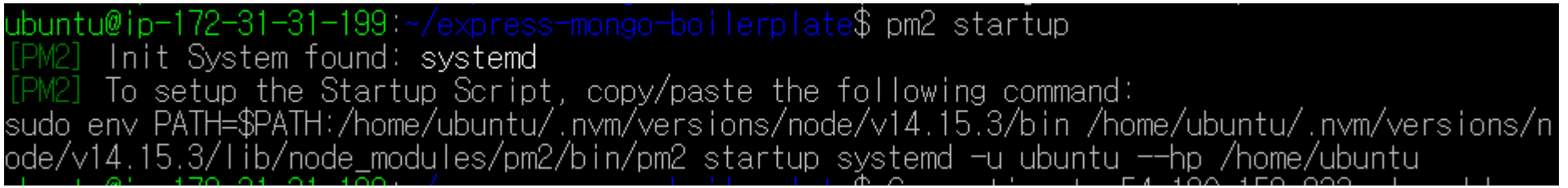
- 트래픽 분산
- 유연한 자원 확보 (요청이 많으면 인스턴스 추가, 그렇지 않으면 인스턴스 삭제)
- 죽은 프로세스 체크
- 안전함

### 2) 이미지 생성

EC2 콘솔 -> 인스턴스 -> 인스턴스 선택 -> 작업 -> 이미지 -> 이미지 생성 -> 마음대로 만들기!

### 3) 시스템 부팅 시 node 서버 자동 실행

Pm2 startup -> 실행 결과를 복붙으로 실행 -> pm2 save



5. ELB

### 1) ELB 생성

EC2 콘솔 -> Load Balancer -> 로드 밸런서 생성 -> Application Load Balancer ->

# 5. ELB

## 1) ELB 생성

### 3단계: 보안 그룹 구성

보안 그룹은 Load Balancer에 대한 트래픽을 제어하는 방화벽 규칙 세트입니다. 이 페이지에서는 특정 트래픽을 Load Balancer에 도달하도록 허용할 규칙을 추가할 수 있습니다. 먼저 새 보안 그룹을 생성할지 아니면 기존 보안 그룹을 선택할지 결정합니다.

**보안 그룹 할당** ○ **새** 보안 그룹 생성
● **기존** 보안 그룹 선택

필터 VPC 보안 그룹

| | 보안 그룹 ID | 이름 | 설명 |
|---|---|---|---|
| ☑ | sg-bc0160d7 | default | default VPC security group |
| ☐ | sg-0b4c812a94840264a | http-ssh | HTTP and SSH. SSH are only accepted from KAIST, home |
| ☑ | sg-055b68d0eda3d3413 | launch-wizard-1 | launch-wizard-1 created 2021-01-04T00:19:57.524+09:00 |
| ☐ | sg-0a685173e88710c5a | ldap | LDAP Server |
| ☐ | sg-0e7724722a0adfe97 | nenw-wheelseminar | nenw-s security group |

### 4단계: 라우팅 구성

Your load balancer routes requests to the targets in this target group using the protocol and port that you specify, and performs health checks on the to all of the listeners configured on this load balancer; you can edit the listeners and add listeners after the load balancer is created.

#### 대상 그룹

대상 그룹: 새 대상 그룹
이름: targets
대상 유형: ● 인스턴스 ○ IP ○ Lambda 함수
프로토콜: HTTP
포트: 80
프로토콜 버전: ● HTTP1
HTTP/1.1을 사용하여 대상으로 요청을 전송합니다. 요청 프로토콜이 HTTP/1.1 또는 HTTP/2일 때 지원됩니다.
○ HTTP2
HTTP/2를 사용하여 대상으로 요청을 전송합니다. 요청 프로토콜이 HTTP/2 또는 gRPC일 때 지원되지만 gRPC 전용 기능은 사용할 수 없습니다.
○ gRPC
gRPC를 사용하여 대상으로 요청을 전송합니다. 요청 프로토콜이 gRPC일 때 지원됩니다.

#### 상태 검사

프로토콜: HTTP
경로: /api/hc

▼ 고급 상태 검사 설정

포트: ● 트래픽 포트 ○ 재정의
정상 임계 값: 2
비정상 임계값: 2
제한 시간: 2 초
간격: 5 초
성공 코드: 200

6. Auto Scaling Group

### 1) Group 생성

EC2 콘솔 -> Auto Scaling Group -> 시작 구성 생성 -> 내 AMI -> [만들어둔 AMI] -> 인스턴스 유형 선택 (인스턴스 세부 정보에 있음) -> 생성 -> 생성한 시작 구성 클릭 -> 보안 그룹 -> TCP 80번 포트 추가 + 소스: ELB 대상 보안 그룹

Auto Scaling Group 생성 -> 시작 구성으로 전환 + 방금 만든 시작 구성 클릭 -> 서브넷 3~4개 추가 -> 그룹 크기 2 , 상태 유예검사 10초, 조건정책 구성: 평균 CPU 사용률 5, 인스턴스 필요시간:0
....
-> 생성