

# AWS 실습

JUHYEMIN

# 1. 로그인 및 IAM 유저 설정

동일한 액세스 유형 및 권한을 사용하여 한 번에 여러 사용자를 추가할 수 있습니다. [자세히 알아보기](#)

사용자 이름\*

[다른 사용자 추가](#)

## AWS 액세스 유형 선택

해당 사용자가 AWS에 액세스하는 방법을 선택합니다. 마지막 단계에서는 액세스 키와 자동 생성된 비밀번호가 제공됩니다. [자세히 알아보기](#)

액세스 유형\*  프로그래밍 방식 액세스

AWS API, CLI, SDK 및 기타 개발 도구에 대해 액세스 키 ID 및 비밀 액세스 키(를) 활성화

**차후 cli 사용을 위해 프로그래밍 방식으로 선택**

사용자가 AWS Management Console에 로그인할 수 있도록 허용하는 비밀번호(를) 활성화합니다.

콘솔 비밀번호\*  자동 생성된 비밀번호

사용자 지정 비밀번호

비밀번호 재설정 필요  사용자가 다음에 로그인할 때 비밀번호 생성 요청

사용자가 비밀번호를 변경할 수 있도록 허용하는 IAMUserChangePassword 정책을 자동으로 가져옵니다.

\*필수

취소

[다음: 권한](#)

# 사용자 추가

## ▼ 권한 설정

 그룹에 사용자 추가

 기존 사용자에서 권한 복사

 기존 정책 직접 연결

기존 그룹에 사용자를 추가하거나 새 그룹을 생성합니다. 그룹을 사용하여 직무별로 사용자의 권한을 관리하는 것이 좋습니다. [자세히 알아보기](#)

## 그룹에 사용자 추가

[그룹 생성](#) [새로 고침](#)

그룹	연결된 정책
<input type="checkbox"/> Sparcs	AdministratorAccess
<input checked="" type="checkbox"/> ConsoleAdmin	AmazonEC2FullAccess 및 6 이상

1. 그룹 이름 : ConsoleAdmin  
권한: AdministratorAccess (맨 위)

2. 그룹 이름 : S3Admin  
권한: : AmazonS3FullAccess

2 결과 표시

# 1. 로그인 및 IAM 유저 설정



성공

아래에 표시된 사용자를 생성했습니다. 사용자 보안 자격 증명을 보고 다운로드할 수 있습니다. AWS Management Console 로그인을 위한 사용자 지침을 이메일로 보낼 수도 있습니다. 지금이 이 자격 증명을 다운로드할 수 있는 마지막 기회입니다. 하지만 언제든지 새 자격 증명을 생성할 수 있습니다.

AWS Management Console 액세스 권한이 있는 사용자가 <https://sparcs-juhyemin.signin.aws.amazon.com/console>에 로그인할 수 있습니다.

↓ .csv 다운로드

Console login link

<https://sparcs-juhyemin.signin.aws.amazon.com/console>

사용자	비밀번호	이메일 로그인 지점
▶  juhyemin	***** 표시	이메일 전송

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

The screenshot shows the AWS Management Console interface. At the top, the 'aws' logo and a '서비스' (Services) dropdown menu are visible, with the dropdown menu circled in red. Below the header, the 'AWS Management Console' title is displayed. On the left side, there is a sidebar with 'AWS 서비스' (AWS Services) and a list of '최근 방문한 서비스' (Recently visited services) including IAM, CloudFormation, and Billing. The 'EC2' icon in this list is circled in red. The main content area shows a list of Amazon Machine Images (AMIs) with columns for AMI ID, description, and architecture. The 'Ubuntu Server 16.04 LTS (HVM), SSD Volume Type' AMI is highlighted with a red border. The '선택' (Select) button for this AMI is also visible.

AMI ID	Description	Architecture
ami-095ca789e0549777d	Amazon Linux 2 AMI (HVM), SSD Volume Type	64비트(x86)
ami-0be3e6f84d3b968cd	Amazon Linux AMI 2018.03.0 (HVM), SSD Volume Type	64비트(x86)
ami-0794a2d1e6d99117a	Ubuntu Server 18.04 LTS (HVM), SSD Volume Type	64비트(x86)
ami-0a25005e83c56767a	Ubuntu Server 16.04 LTS (HVM), SSD Volume Type	64비트(x86)
ami-09f603257550ed748	SUSE Linux Enterprise Server 15 SP1 (HVM), SSD Volume Type	64비트(x86)

### 1) 단계1: Xenial 선택

서비스 -> EC2 -> 인스턴스 -> 인스턴스 시작

## 2-1. EC2

### 단계 2: 인스턴스 유형 선택

Amazon EC2는 각 사용 사례에 맞게 최적화된 다양한 인스턴스 유형을 제공합니다. 인스턴스는 애플리케이션을 실행할 수 있는 가상 서버입니다. 이러한 인스턴스에는 CPU, 메모리, 스토리지 및 네트워크 용량의 다양한 조합이 있으며, 애플리케이션에 사용할 적절한 리소스 조합을 제공합니다. 인스턴스 유형과 이 인스턴스 유형이 컴퓨팅 요건을 충족하는 방식에 대해 자세히 알아보기

필터링 기준: **모든 인스턴스 유형** | 현재 세대 | **별 표시/숨기기**

현재 선택된 항목: **t2.micro (Variable ECU, 1 vCPUs, 2.5 GHz, Intel Xeon Family, 1 GB 메모리, EBS 전용)**

	그룹	유형	vCPUs	메모리 (GB)	인스턴스 스토리지 (GB)	EBS 최적화 사용 가능	네트워크 성능
<input type="checkbox"/>	General purpose	t2.nano	1	0.5	EBS 전용	-	낮음에서 중간
<input checked="" type="checkbox"/>	General purpose	t2.micro	1	1	EBS 전용	-	낮음에서 중간
<input type="checkbox"/>	General purpose	t2.medium	2	4	EBS 전용	-	낮음에서 중간
<input type="checkbox"/>	General purpose	t2.large	2	8	EBS 전용	-	낮음에서 중간
<input type="checkbox"/>	General purpose	t2.xlarge	4	16	EBS 전용	-	보통
<input type="checkbox"/>	General purpose	t2.2xlarge	8	32	EBS 전용	-	보통
<input type="checkbox"/>	General purpose	t3.nano	2	0.5	EBS 전용	예	Up to 5 GIGabit
<input type="checkbox"/>	General purpose	t3.micro	2	1	EBS 전용	예	Up to 5 GIGabit
<input type="checkbox"/>	General purpose	t3.small	2	2	EBS 전용	예	Up to 5 GIGabit
<input type="checkbox"/>	General purpose	t3.medium	2	4	EBS 전용	예	Up to 5 GIGabit
<input type="checkbox"/>	General purpose	t3.large	2	8	EBS 전용	예	Up to 5 GIGabit
<input type="checkbox"/>	General purpose	t3.xlarge	4	16	EBS 전용	예	Up to 5 GIGabit

2) 단계2: Free Tier 선택

### 단계 3: 인스턴스 세부 정보 구성

도메인 조인 디렉터리 ⓘ

디렉터리 없음 | 새 디렉터리 생성

IAM 역할 ⓘ

없음 | 새 IAM 역할 생성

CPU 옵션 ⓘ

CPU 옵션 지정

종료 방식 ⓘ

중지

최대 절전 중지 동작 ⓘ

추가 종료 동작으로 최대 절전 모드를 활성화

종료 방지 기능 활성화 ⓘ

우발적인 종료로부터 보호

모니터링 ⓘ

CloudWatch 세부 모니터링 활성화

추가 요금이 발생합니다.

취소 | 이전 | 검토 및 시작 | 다음: 스토리지 추가

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:

설명:

유형 ⓘ	프로토콜 ⓘ	포트 범위 ⓘ	소스 ⓘ	설명 ⓘ
SSH ▾	TCP	22	사용자 지정 ▾ 0.0.0.0/0	예: SSH for Admin Desktop ✕
사용자 지정 TCP ▾	TCP	80	사용자 지정 ▾ 0.0.0.0/0	예: SSH for Admin Desktop ✕
사용자 지정 TCP ▾	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!)**



## 2-1. EC2

### 기존 키 페어 선택 또는 새 키 페어 생성



키 페어는 AWS에 저장하는 퍼블릭 키와 사용자가 저장하는 프라이빗 키 파일로 구성됩니다. 이 둘을 모두 사용하여 SSH를 통해 인스턴스에 안전하게 접속할 수 있습니다. Windows AMI의 경우 인스턴스에 로그인하는 데 사용되는 암호를 얻으려면 프라이빗 키 파일이 필요합니다. Linux AMI의 경우, 프라이빗 키 파일을 사용하면 인스턴스에 안전하게 SSH로 연결할 수 있습니다.

참고: 선택한 키 페어가 이 인스턴스에 대해 승인된 키 세트에 추가됩니다. 퍼블릭 AMI에서 기존 키 페어 제거에 대해 자세히 알아보십시오.

새 키 페어 생성

키 페어 이름

Sparcs\_AWS\_Seminar

키 페어 다운로드

계속하려면 먼저 프라이빗 키 파일(\*.pem 파일)을 다운로드해야 합니다. 액세스할 수 있는 안전한 위치에 저장합니다. 파일은 생성되고 나면 다시 다운로드할 수 없습니다.

취소 인스턴스 시작

- 이후 putty로 .pem 을 저장
  - 또는 리눅스에 저장



```
juhyemin@LAPTOP-CQHNN6N:~/wheel/sparcs_key_pair$ ls
Sparcs_AWS_Seminar.pem
juhyemin@LAPTOP-CQHNN6N:~/wheel/sparcs_key_pair$
```

4) 새 키 페어 생성  
반드시 은밀한 곳에 저장해야 함!!!!



## 2-1. EC2

1) EC2 인스턴스에는 동적 IP가 할당됨

퍼블릭 IPv4 주소

 100.27.20.253 | [개방 주소법](#) 

퍼블릭 IPv4 DNS

 ec2-100-27-20-253.compute-1.amazonaws.com | [개방 주소법](#) 

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.

```
cloud-init/ aws/ aws-heat-on-sak/ t/ ec2/ master/ release-notes
=====
* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:        https://ubuntu.com/advantage

* Introducing self-healing high availability clusters in MicroK8s.
  Simple, hardened, Kubernetes for production, from RaspberryPi to DC.

  https://microk8s.io/high-availability

17 packages can be updated.
0 updates are security updates.

*** System restart required ***

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

ubuntu@ip-172-31-90-162:~$
```

## 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](http://[public_ip]:3000)

**Express**

Welcome to Express

## 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
```

## Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to [nginx.org](http://nginx.org).  
Commercial support is available at [nginx.com](http://nginx.com).

*Thank you for using nginx.*

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
```

### 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
```

```
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";  
    }  
}
```



Edit src/App.js and save to reload.

[Learn React](#)

## 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](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]
```

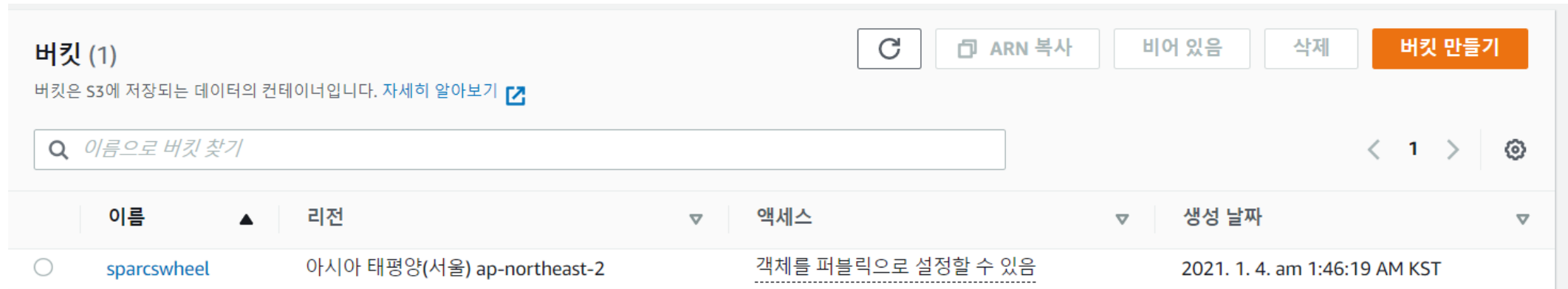
```
region=ap-northeast-2
```

```
output=json
```

### 3. S3 : 버킷 생성

#### 1) 버킷

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



The screenshot shows the AWS S3 console interface. At the top, there are several action buttons: a refresh icon, 'ARN 복사', '비어 있음', '삭제', and a prominent orange '버킷 만들기' button. Below these is a search bar with the placeholder text '이름으로 버킷 찾기'. A table below the search bar lists the bucket details:

이름	리전	액세스	생성 날짜
sparcswheel	아시아 태평양(서울) ap-northeast-2	객체를 퍼블릭으로 설정할 수 있음	2021. 1. 4. am 1:46:19 AM KST

### 3. S3 : 버킷 생성

#### 2) 버킷의 정적 파일 설정

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

정적 웹 사이트 호스팅

- 비활성화
- 활성화

중스티 으혀

- 객체에 대한 요청 디너역간  
요청을 다른 버킷 또는 도메인으로 리디렉션합니다. 자세히 알아보기 [↗](#)

#### 3) 버킷 권한 설정

권한 -> 버킷 정책 -> 정책 생성기 -> ...

**i** 고객이 웹 사이트 엔드포인트의 콘텐츠에 액세스할 수 있게 하려면 모든 콘텐츠를 공개적으로 읽기 가능하도록 설정해야 합니다. 이렇게 하려면, 버킷에 대한 S3 퍼블릭 액세스 차단 설정을 편집하며 됩니다. 자세한 내용은 [Amazon S3 퍼블릭 액세스 차단 개요](#) [↗](#) 참조하십시오.

인덱스 문서

웹 사이트의 홈 페이지 또는 기본 페이지를 지정합니다.

index.html

오류 문서

오류가 발생하면 반환됩니다.

index.html

### 3. S3 : 버킷 생성 – 정책 설정

#### Step 1: Select Policy Type

A Policy is a container for permissions. The different types of policies you can create are an [IAM Policy](#), an [S3 Bucket VPC Endpoint Policy](#), and an [SQS Queue Policy](#).

Select Type of Policy

#### Step 2: Add Statement(s)

A statement is the formal description of a single permission. See [a description of elements](#) that you can use in sta

Effect  Allow  Deny

Principal

Use a comma to separate multiple values.

AWS Service   All Services ('\*')

Use multiple statements to add permissions for more than one service.

Actions   All Actions ('\*')

Amazon Resource Name (ARN)

ARN should follow the following format: `arn:aws:s3:::<bucket_name>/<key_name>`.  
Use a comma to separate multiple values.

[Add Conditions \(Optional\)](#)

Add Statement

### 3. S3 : 버킷 생성 – 정책 설정

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


## 버킷 정책 편집

### 버킷 정책

JSON으로 작성된 버킷 정책은 버킷에 저장된 객체에 대한 액세스 권한을 제공합니다. 버킷 정책은 다른 계정이 소유한 객체에는 적용되지 않습니다. [자세히 알아보기](#)

[정책 예제](#) [정책 생성기](#)

버킷 ARN

 arn:aws:s3:::sparcswheel

### 정책

```
1 {
2   "Id": "Policy1609692773084",
3   "Version": "2012-10-17",
4   "Statement": [
5     {
6       "Sid": "Stmt1609692764477",
7       "Action": [
8         "s3:GetObject"
9       ],
10      "Effect": "Allow",
11      "Resource": "arn:aws:s3:::sparcswheel/",
12      "Principal": "*"
13    }
14  ]
15 }
```

### 3. S3 : 버킷을 EC2로 접근

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

## 4. Document DB 생성

### 1) 개념

Document DB는 클러스터 & 인스턴스로 나누어 처리

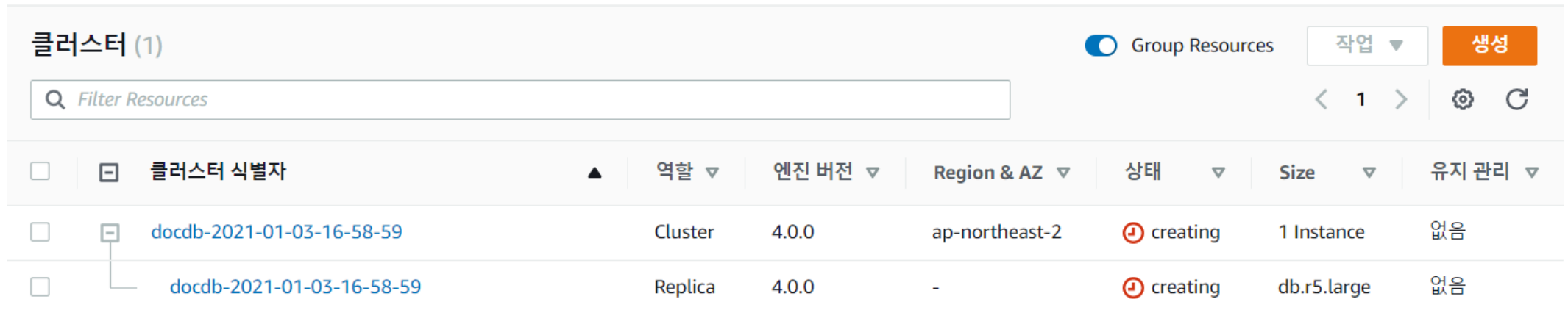
저장 -> 클러스터 : 데이터를 관리하는 저장소

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



The screenshot shows the Amazon DocumentDB console interface. At the top, there is a search bar labeled '클러스터 (1)' and a 'Group Resources' toggle. Below the search bar is a table listing the cluster and its replicas. The cluster is in the 'creating' state, and the replica is also in the 'creating' state.

<input type="checkbox"/>	<input type="checkbox"/>	클러스터 식별자	역할	엔진 버전	Region & AZ	상태	Size	유지 관리
<input type="checkbox"/>	<input type="checkbox"/>	docdb-2021-01-03-16-58-59	Cluster	4.0.0	ap-northeast-2	creating	1 Instance	없음
<input type="checkbox"/>	<input type="checkbox"/>	docdb-2021-01-03-16-58-59	Replica	4.0.0	-	creating	db.r5.large	없음



#### 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 트래픽 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)

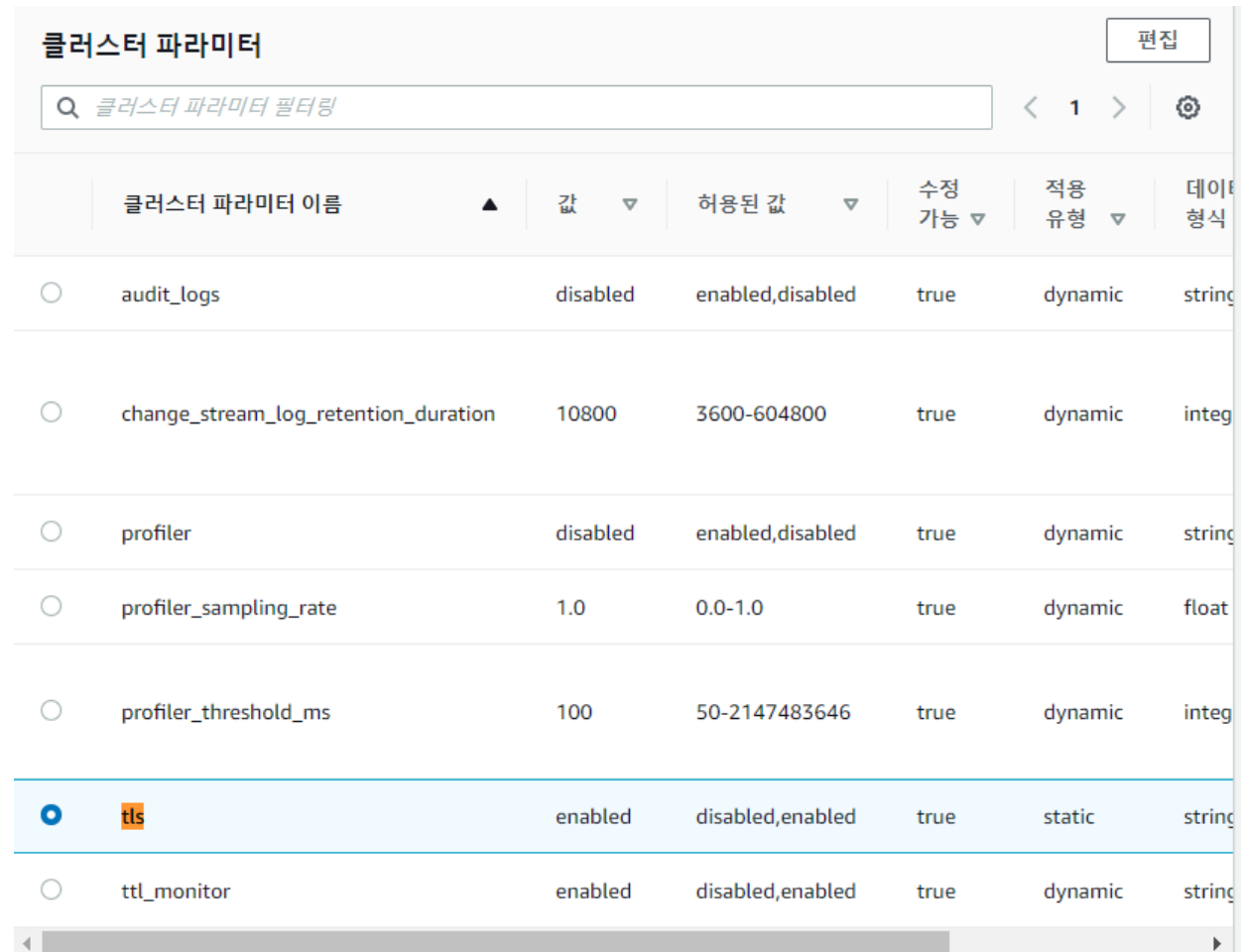
Wget <https://s3.amazonaws.com/rds-downloads/rds-combined-ca-bundle.pem>

DocumentDB 클러스터 클릭 -> 상세 정보: 연결에서 mongo 셸 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



The screenshot shows the AWS Management Console interface for DocumentDB Parameter Groups. The title is '클러스터 파라미터' (Cluster Parameters) with a '편집' (Edit) button. A search bar contains '클러스터 파라미터 필터링'. Below is a table of parameters. The 'tls' parameter is highlighted in blue and is currently set to 'enabled'. The table columns are: Cluster Parameter Name, Value, Allowed Values, Modifiable, Application Type, and Data Type.

클러스터 파라미터 이름	값	허용된 값	수정 가능	적용 유형	데이터 형식
<input type="radio"/> audit_logs	disabled	enabled,disabled	true	dynamic	string
<input type="radio"/> change_stream_log_retention_duration	10800	3600-604800	true	dynamic	integ
<input type="radio"/> profiler	disabled	enabled,disabled	true	dynamic	string
<input type="radio"/> profiler_sampling_rate	1.0	0.0-1.0	true	dynamic	float
<input type="radio"/> profiler_threshold_ms	100	50-2147483646	true	dynamic	integ
<input checked="" type="radio"/> <b>tls</b>	enabled	disabled,enabled	true	static	string
<input type="radio"/> ttl_monitor	enabled	disabled,enabled	true	dynamic	string

#### 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.amazonaws.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

```
ubuntu@ip-172-31-31-199:~/express-mongo-boilerplate$ pm2 startup
[PM2] Init System found: systemd
[PM2] To setup the Startup Script, copy/paste the following command:
sudo env PATH=$PATH:/home/ubuntu/.nvm/versions/node/v14.15.3/bin /home/ubuntu/.nvm/versions/node/v14.15.3/lib/node_modules/pm2/bin/pm2 startup systemd -u ubuntu --hp /home/ubuntu
```

## 5. ELB

### 1) ELB 생성

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

#### 1단계: Load Balancer 구성

##### 기본 구성

Load Balancer를 구성하려면 이름을 입력하고, 체계를 선택하고, 하나 이상의 가용 영역을 지정합니다. 기본 구성은 선택한 네트워크에서의 인터넷 경계 Load Balancer 및 포트입니다.

Load Balancer에서 활성화할 가용 영역을 지정합니다. Load Balancer는 지정한 가용 영역에 위치한 대상으로만 트래픽을 라우팅합니다. 가용 영역당 1개의 서브넷만 지정할 수 있습니다. Load Balancer의 가용성을 높이려면 2개 이상의 가용 영역에서 서브넷을 지정해야 합니다.

이름 ⓘ wheel\_seminar

체계 ⓘ  
 인터넷 경계  
 내부

IP 주소 유형 ⓘ ipv4

VPC ⓘ vpc-5592053d (172.31.0.0/16) (기본값) ⌵

##### 가용 영역

ap-northeast-2a subnet-6b4e2e03 ⌵

IPv4 주소 ⓘ CIDR에서 할당 172.31.0.0/20

ap-northeast-2b subnet-846d9fff ⌵

IPv4 주소 ⓘ CIDR에서 할당 172.31.32.0/20

ap-northeast-2c subnet-f68554ba ⌵

IPv4 주소 ⓘ CIDR에서 할당 172.31.16.0/20

##### 리스너

## 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

보안 그룹 ID	이름	설명
<input checked="" type="checkbox"/> sg-bc0160d7	default	default VPC security group
<input type="checkbox"/> sg-0b4c812a94840264a	http-ssh	HTTP and SSH. SSH are only accepted from KAIST, home
<input checked="" type="checkbox"/> sg-055b68d0eda3d3413	launch-wizard-1	launch-wizard-1 created 2021-01-04T00:19:57.524+09:00
<input type="checkbox"/> sg-0a685173e88710c5a	ldap	LDAP Server
<input type="checkbox"/> 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

이름

대상 유형  인스턴스  
 IP  
 Lambda 함수

프로토콜

포트

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

경로

#### 고급 상태 검사 설정

포트  트래픽 포트  
 재정의

정상 임계값

비정상 임계값

제한 시간  초

간격  초

성공 코드



## 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