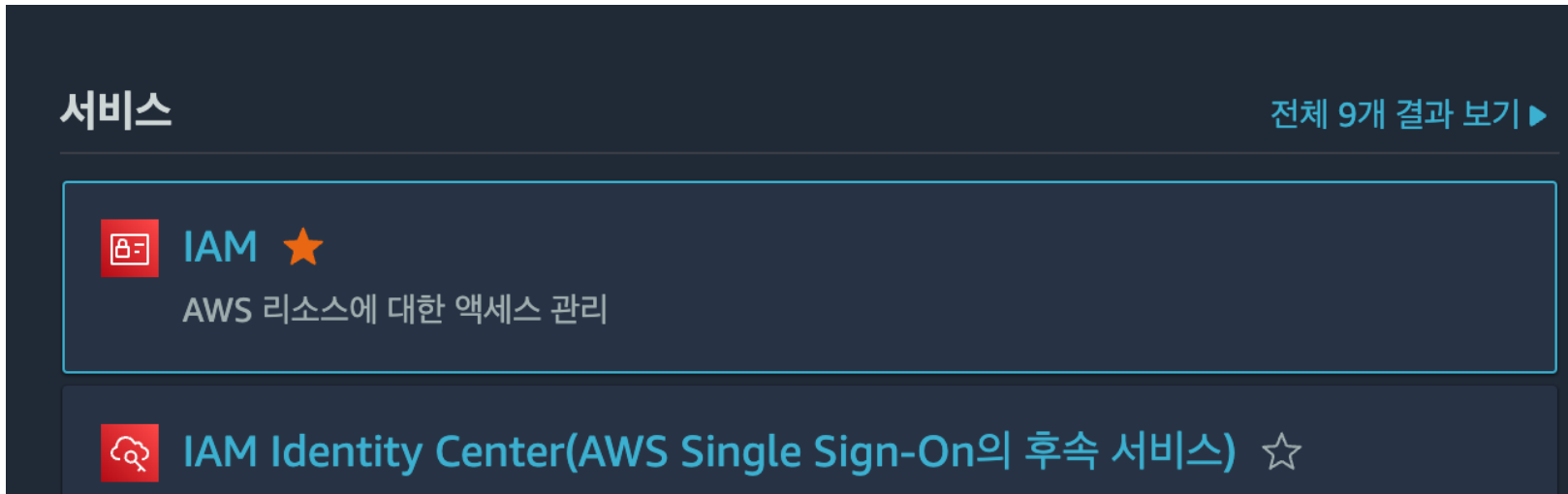


K-Digital Training 웹 풀스택 과정

# 서버 배포

# multer 업로드

# IAM 접속



# 사용자 추가

Identity and Access Management(IAM) X

Q IAM 검색

대시보드

▼ 액세스 관리

사용자 그룹

**사용자**

역할

정책

자격 증명 공급자

계정 설정

↺ 삭제 사용자 생성

< 1 > ⚙

▼ 생성 시간 ▼

# 사용자 추가

## 사용자 세부 정보

사용자 이름

test

사용자 이름은 최대 64자까지 가능합니다. 유효한 문자: A~Z, a~z

☐ AWS Management Console에 대한 사용자 액세스:  
사용자에게 콘솔 액세스 권한을 제공하는 경우 IAM Identity

### 권한 옵션

☐ 그룹에 사용자 추가

기존 그룹에 사용자를 추가하거나 새 그룹을 생성합니다. 그룹을 사용하여 직무별로 사용자 권한을 관리하는 것이 좋습니다.

☐ 권한 복사

기존 사용자의 모든 그룹 멤버십, 연결된 관리형 정책 및 인라인 정책을 복사합니다.

☒ 직접 정책 연결

관리형 정책을 사용자에게 직접 연결합니다. 사용자에게 연결하는 대신, 정책을 그룹에 연결한 후 사용자를 적절한 그룹에 추가하는 것이 좋습니다.

### 권한 정책 (1/1122)

새 사용자에게 연결할 정책을 하나 이상 선택합니다.

Q s3

필터링 기준 유형

모든 유형

11 개 일치

정책 이름

유형

<input type="checkbox"/>	AmazonDMSRedshiftS3Role	AWS 관리형
<input checked="" type="checkbox"/>	AmazonS3FullAccess	AWS 관리형
<input type="checkbox"/>	AmazonS3ObjectLambdaExecutionRolePolicy	AWS 관리형
<input type="checkbox"/>	AmazonS3OutpostsFullAccess	AWS 관리형
<input type="checkbox"/>	AmazonS3OutpostsReadOnlyAccess	AWS 관리형

# 엑세스 키

IAM > 사용자 > test-multier

test-multier 정보 삭제

요약		
ARN arn:aws:iam::103545627881:user/test-multier	콘솔 액세스 비활성화됨	
생성됨 August 23, 2023, 03:28 (UTC+09:00)	마지막 콘솔 로그인 -	엑세스 키 2 엑세스 키 만들기

권한 그룹 태그 보안 자격 증명 액세스 관리자

# 엑세스 키

사용 사례

☒ Command Line Interface(CLI)  
AWS CLI를 사용하여 AWS 계정에 액세스할 수 있도록 이 액세스 키를 사용할 것입니다.


☐ 로컬 코드  
로컬 개발 환경의 애플리케이션 코드를 사용하여 AWS 계정에 액세스할 수 있도록 이 액세스 키를 사용할 것입니다.

☐ AWS 컴퓨팅 서비스에서 실행되는 애플리케이션  
Amazon EC2, Amazon ECS 또는 AWS Lambda와 같은 AWS 컴퓨팅 서비스에서 실행되는 애플리케이션 코드를 사용하여 AWS 계정에 액세스할 수 있도록 이 액세스 키를 사용할 것입니다.

☐ 서버 파티 서비스  
AWS 리소스를 모니터링 또는 관리하는 서버 파티 애플리케이션 또는 서비스에 액세스할 수 있도록 이 액세스 키를 사용할 것입니다.

☐ AWS 외부에서 실행되는 애플리케이션  
애플리케이션을 온프레미스 호스트에서 실행하거나 로컬 AWS 클라이언트 또는 서버 파티 AWS 플러그 인을 사용할 수 있도록 이 액세스 키를 사용할 것입니다.

☐ 기타  
귀하의 사용 사례가 여기에 나열되어 있지 않습니다.

 권장되는 대안

- 브라우저 기반 CLI인 [AWS CloudShell](#)을 사용하여 명령을 실행합니다. [자세히 알아보기](#)
- [AWS CLI V2](#)를 사용하고 IAM 자격 증명 센터의 사용자를 통한 인증을 활성화합니다. [자세히 알아보기](#)

확인

☒ 위의 권장 사항을 이해했으며 액세스 키 생성을 계속하려고 합니다.

취소

다음

다음 버튼으로 진행 -> 액세스 키 만들기 후  
csv파일 다운 받으세요!  
잃어버리면 안됩니다!

# 권한 설정 변경

## S3 버킷 클릭 후 객체 소유권 변경

객체 소유권 정보

다른 AWS 계정에서 이 버킷에 작성한 객체의 소유권 및 액세스 제어 목록(ACL)의 사용을 제어합니다. 객체 소유권은 객체에 대한 액세스를 지정할 수 있는 사용자를 결정합니다.

편집

객체 소유권

다른 AWS 계정에서 이 버킷에 작성한 객체의 소유권 및 액세스 제어 목록(ACL)의 사용을 제어합니다. 객체 소유권은 객체에 대한 액세스를 지정할 수 있는 사용자를 결정합니다.

☐ ACL 비활성화됨(권장)

이 버킷의 모든 객체는 이 계정이 소유합니다. 이 버킷과 그 객체에 대한 액세스는 정책을 통해서만 지정됩니다.

☒ ACL 활성화됨

이 버킷의 객체는 다른 AWS 계정에서 소유할 수 있습니다. 이 버킷 및 객체에 대한 액세스는 ACL을 사용하여 지정할 수 있습니다.

⚠

각 객체의 액세스를 개별적으로 제어하거나 객체 라이터가 업로드하는 데이터를 소유하도록 해야 하는 경우가 아니라면 ACL을 비활성화하는 것이 좋습니다. ACL 대신 버킷 정책을 사용하여 계정 외부의 사용자와 데이터를 공유하면 권한을 관리하고 감사하기가 용이합니다.

객체 소유권

☒ 버킷 소유자 선호

이 버킷에 작성된 새 객체가 bucket-owner-full-control 삽입 ACL을 지정하는 경우 새 객체는 버킷 소유자가 소유합니다. 그렇지 않은 경우 객체 라이터가 소유합니다.

☐ 객체 라이터

객체 라이터는 객체 소유자로 유지됩니다.

ℹ

새 객체에 대해서만 객체 소유권을 적용하려면 버킷 정책이 객체 업로드에 bucket-owner-full-control 삽입 ACL을 요구하도록 지정해야 합니다. [자세히 알아보기](#)

취소

변경 사항 저장

2024-04-30

8



# aws 모듈설치

```
npm install aws-sdk multer-s3@^2
```

```
const aws = require("aws-sdk");  
const multerS3 = require("multer-s3");
```

# multer aws 업로드

```
const express = require("express");
const multer = require("multer");
const aws = require("aws-sdk");
const multerS3 = require("multer-s3");
const app = express();
const PORT = 8000;

//view engine 설정
app.set("view engine", "ejs");
//aws 설정
aws.config.update({
  accessKeyId: "키",
  secretAccessKey: "암호키",
  region: "ap-northeast-2",
});

const s3 = new aws.S3();

const upload = multer({
  storage: multerS3({
    s3: s3,
    bucket: "버킷 주소",
    acl: "public-read",
    metadata: function (req, file, cb) {
      cb(null, { fieldName: file.fieldname });
    },
    key: function (req, file, cb) {
      cb(null, Date.now().toString() + "-" + file.originalname);
    },
  }),
});
```

```
async function fileupload(e) {
  e.preventDefault();
  //파일에 접근
  const file = document.querySelector("#files");
  console.log(file.files);
  //폼 데이터 생성
  const formData = new FormData();
  //파일이 여러개이므로 반복문을 사용
  for (let i = 0; i < file.files.length; i++) {
    formData.append("files", file.files[i]);
  }

  try {
    const res = await axios({
      method: "POST",
      url: "/upload",
      data: formData,
      headers: {
        "Content-Type": "multipart/form-data",
      },
    });
    res.data.forEach((elem) => {
      // resultBox.innerHTML += ``;
      const img = document.createElement("img");
      img.src = elem.location;
      img.style.width = "500px";
      resultBox.appendChild(img);
    });
  } catch (error) {
    console.log(error);
  }
}
```

# 깃 저장소 생성

## Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (\*).

Owner \*



Repository name \*

Great repository names are short and memorable. Need inspiration? How about [legendary-journey](#) ?

Description (optional)



Public

Anyone on the internet can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.

Initialize this repository with:



Add a README file

This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template:None ▾

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License:None ▾

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

You are creating a public repository in your personal account.

Create repository

## 깃 저장소 이름 생성 후

## Public or Private 선택 후

## Create repository 클릭

# 깃 저장소 생성

배포할 프로젝트 에서

Step1 : .gitignore 파일 생성 후 node\_module 폴더 등록

Step2: git init

Step3: git remote add origin 저장소 주소

Step4: git add .

Step5: git commit -m "메세지"

Step6: git push origin main

# Nginx

# Nginx 란?

- 비동기 이벤트 기반 구조의 웹 서버 프로그램
  - 클라이언트로부터 요청을 받았을 때 요청에 맞는 정적 파일을 응답해주는 HTTP Web Server로 활용 된다
  - Apache는 클라이언트 접속 시마다 프로세스 또는 스레드를 생성하는 구조이기 때문에 대량의 클라이언트(1만 이상)가 접속했을 경우 한계가 있음
- => 이를 보완하기 위해 만들어진 것이 Nginx

# nginx 설치, 설정

# EC2 터미널 접속



```
Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

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.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-15-140:~$
```



# Nginx 서버 설치

패키지 저장소 업데이트

```
:~$ sudo apt-get update
```

Nginx 설치

```
:~$ sudo apt-get install nginx
```

sudo : 관리자 권한 실행

apt-get : 우분투, 데비안의 리눅스에서 사용하는 패키지 관리 도구

# Nginx 설정

## **sites-available** 디렉토리

- 서버에 호스팅할 수 있는 여러 웹 사이트 설정 파일이 저장
- 각 설정 파일은 하나의 가상 호스팅을 정의하며, 이를 통해 서로 다른 도메인 또는 서브도메인에 대한 웹 사이트 구성
- 실제로 웹 서버가 사용하는 설정 파일이 아닌, 사용 가능한 설정 목록을 나타냄 (작성만 하는 곳)

## **sites-enabled** 디렉토리

- sites-available에서 작성한 실제로 활성화되는 가상 호스팅 설정 파일이 있는 곳
- 모든 설정 파일들은 심볼릭 링크(Symbolic Link)로 sites-available 디렉토리에 있는 설정 파일과 연결됨

# (선택) default파일 백업

default 파일?

- nginx설정 파일로 nginx 설치 후 welcome to nginx 페이지 실행한다
- 새로 만드는 설정과 겹칠 수 있기 때문에 원본 파일 백업 후 기본 설정 파일 삭제

cp : copy의 줄임말

-r : 하위 디렉토리까지 모두 복사

```
:~$ sudo cp -r /etc/nginx/sites-available/ /etc/nginx/sites-available-origin
```

```
:~$ sudo cp -r /etc/nginx/sites-enabled/ /etc/nginx/sites-enabled-origin
```

# (선택) default파일 삭제

```
:~$ sudo rm /etc/nginx/sites-available/default
```

```
:~$ sudo rm /etc/nginx/sites-enabled/default
```

# 설정 파일 생성

```
:~$ sudo vi /etc/nginx/sites-available/test
```



원하는 설정파일명

확장자가 .conf 생성

# 설정 파일 생성

node.js 일  
경우

```
server {
    listen 80;
    server_name IP or 도메인
    location / {
        proxy_pass http://127.0.0.1:8000;
    }
}
```

← Node프로젝트  
포트 입력

react 일  
경우

```
server {
    listen 80;
    location / {
        root /home/ubuntu/ 리액트 build 경로;
        index index.html index.htm;
        try_files $uri /index.html;
    }
}
```

# 심볼릭 파일 생성

```
:~$ sudo ln -s /etc/nginx/sites-available/test /etc/nginx/sites-enabled/test
```

- 심볼릭 완성 후 nginx 문법 확인

```
ubuntu@ip-:~$ sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

- nginx 재시작

```
:~$ sudo systemctl restart nginx
```

# nodejs 설치



# nodejs 설치

```
:~$ curl -fsSL https://deb.nodesource.com/setup_lts.x | sudo -E bash
```

```
:~$ sudo apt-get install -y nodejs
```

```
:~$ sudo npm i -g pm2
```

\*pm2 란? Node.js 어플리케이션을 쉽게 관리할 수 있게 해주는 Process Manager

\*리눅스 nodejs 설치 저장소

<https://github.com/nodesource/distributions>

# Git Clone 후 서버 실행

```
~$ mkdir 폴더명
~$ cd 폴더명
~폴더명$ git clone 저장소 주소 .
~폴더명$ npm install

~폴더명$ pm2 start index.js
```

# pm2 명령어

- pm2 상태확인  
pm2 status
- 재시작  
pm2 restart id값
- 중지  
pm2 stop id값
- 완전 중지  
pm2 kill
- 로그확인  
pm2 log

# 권한 변경

50X 오류가 나온다면! 외부에서 폴더 읽을 권한이 없어서 생기는 문제!

```
cd /home
```

```
sudo chmod 711 ubuntu/
```

# 권한 설정

- 권한 확인 : ls -l
- 1번째 문자: 종류 (표 참고)
- 2~4번째 문자: 파일 소유자에 대한 권한
- 5~7번째 문자: 파일의 소유 그룹에 대한 권한
- 8~10번째 문자: 타인의 권한

종류	의미
-	파일
d	디렉토리 (폴더)
l	심볼릭 링크

# 권한 의미

- 2~4번째 문자: 파일 소유자에 대한 권한
- 5~7번째 문자: 파일의 소유 그룹에 대한 권한
- 8~10번째 문자: 타인의 권한
- ex) 711: 소유자는 읽기,쓰기,실행 권한, 소유 그룹은 읽기,쓰기, 타인은 읽기 권한 부여

모드(숫자)	모드(알파벳)	권한
4	r	읽기
2	w	쓰기
1	x	실행