

# 해피캣 포팅 매뉴얼

---



## 포팅 매뉴얼

### 목차

1. 사용 버전
  - 프로젝트 사용 도구
  - 프로젝트 개발 환경
2. 설정 파일 및 환경 변수 정보
  - 프론트엔드 설정 파일
  - 프론트엔드 환경 변수 정보
  - 백엔드 설정 파일
3. 배포 환경 설정
  - EC2 설정
  - NginX & SSL
  - Docker설치
  - Nginx sites-enabled 설정
4. 빌드 및 배포

- 프론트엔드
- 백엔드
- FLASK
- 포트개방

#### 5. 외부서비스

- AWS S3 버킷 생성 및 권한 설정
- 도커 허브 이용법

---

## 사용버전

### 프로젝트 사용 도구

- 이슈 관리 : Jira
- 형상 관리 : GitLab
- 커뮤니케이션 : Notion, MatterMost
- UI/UX : Figma

### 프로젝트 개발 환경

#### Front-End

- Visual Studio Code - 1.85.2
- Vue - 5.0.8
- Node.js - 20.11.0
- npm - 10.4.0
- Bootstrap - 5.3.2

#### Back-End

- IntelliJ IDEA
- SpringBoot - 2.7.18
- JAVA - Oracle JDK 1.8
- Spring Security
- JWT

- AWS S3
- nginx

## **DataBase**

- MariaDB - 11.4.0

## **AI**

- Python - 3.10.9
- Flask - 2.2.3
- OpenCV - 4.6.0.66
- YOLOv8 - 8.0.181
- CoLab

## **AR**

- UnityAR
- Blender
- 3DMax

## **CI/CD**

- Ubuntu - 20.04.6 LTS
  - Docker - 1.5.2
  - Nginx - 1.18.0-0ubuntu1.4
  - SSL
  - Gunicorn - 20.0.4-3
- 

## **2. 설정파일 및 환경 변수 정보**

### **프론트엔드 설정파일**

vite.config.js

```

import { fileURLToPath, URL } from 'node:url'

import { defineConfig } from 'vite'
import vue from '@vitejs/plugin-vue'
import { VitePWA } from 'vite-plugin-pwa'

// https://vitejs.dev/config/
export default defineConfig({
  workboxOptions: {
    include: [/^index\.html$/, /\.css$/, /\.js$/, /^manifest\.json$/,
    exclude: []
  },
  publicPath: '/',
  plugins: [
    vue(),
    VitePWA({
      registerType: 'autoUpdate',
      includeAssets: ['favicon.ico', 'robots.txt', 'apple-touch-icon'],
      manifest: {
        name: 'POP-CON',
        short_name: 'POP-CON',
        icons: [
          {
            src: '/img/icons/pwa-192x192.png',
            sizes: '192x192',
            type: 'image/png',
            purpose: 'any',
          },
          {
            src: '/img/icons/pwa-512x512.png',
            sizes: '512x512',
            type: 'image/png',
            purpose: 'any',
          },
          {
            src: '/img/icons/pwa-maskable-192x192.png',
            sizes: '192x192',
            type: 'image/png',
            purpose: 'maskable',
          },
          {
            src: '/img/icons/pwa-maskable-512x512.png',

```

```

        sizes: '512x512',
        type: 'image/png',
        purpose: 'maskable',
      },
    ],
    start_url: '/',
    display: 'standalone',
    background_color: '#ffffff',
    theme_color: '#ffffff',
  },
  workbox: {
    globPatterns: ['**/*.{js,css}', 'index.html'],
    runtimeCaching: [
      {
        urlPattern: /\.png$/,
        handler: 'CacheFirst',
        options: {
          cacheName: 'png-cache',
          expiration: {
            maxEntries: 10,
            maxAgeSeconds: 60 * 60 * 24 * 365,
          },
        },
      },
      {
        urlPattern: /\.json$/,
        handler: 'StaleWhileRevalidate',
        options: {
          cacheName: 'json-cache',
          cacheableResponse: {
            statuses: [200],
          },
        },
      },
    ],
  },
},
resolve: {
  alias: {
    '@': fileURLToPath(new URL('./src', import.meta.url))
  }
}

```

```

    },
    optimizeDeps: {
      entries: [],
    }
  })

```

## 프론트엔드 환경변수

env

```
VITE_VUE_API_URL=https://i10c211.p.ssafy.io:8085
```

## 백엔드 설정파일

application.properties

```

# DataSource
#spring.datasource.hikari.driver-class-name=com.mariadb.cj.jdbc.Driver
#spring.datasource.hikari.jdbc-url=jdbc:mariadb://localhost:3306/popcon
spring.datasource.hikari.username=root
spring.datasource.hikari.password=c211happycat10
spring.datasource.driver-class-name=net.sf.log4jdbc.sql.jdbcapi.Driver
spring.datasource.url=jdbc:log4jdbc:mariadb://i10c211.p.ssafy.io:3306/popcon

# mybatis
mybatis.mapper-locations=mapper/**/*.xml
mybatis.type-aliases-package=com.ssafy.popcon.**.dto

# logging
logging.level.root=info
logging.level.com.ssafy=debug

# sql ?? ?? ??
logging.level.jdbc.sqlonly=off
logging.level.jdbc.sqltiming=info
logging.level.jdbc.resultsettable=info
logging.level.jdbc.audit=off
logging.level.jdbc.resultset=off
logging.level.jdbc.connection=off

spring.config.import=aws.properties,log4jdbc.log4j2.properties

```

```

spring.jwt.secret=vmfhaltmskdlstkfkdgodyroqkfwkdbalroqkfwkdbalafhrmdl

# Email
spring.mail.host=smtp.gmail.com
spring.mail.port=587
spring.mail.username=ssafy10c211@gmail.com
spring.mail.password=elfu eeep dxfm ovjc
spring.mail.properties.mail.smtp.starttls.enable=true

#spring.servlet.multipart.max-file-size=10MB
#spring.servlet.multipart.max-request-size=10MB

```

#### aws.properties

```

# S3
#cloud.aws.credentials.accessKey=AKIAYS2NV46LOPOLDWP2
#
#cloud.aws.credentials.secretKey=7J4q12q8//NLq+yhBoeMBfepUw7hvBdreKOr

cloud.aws.credentials.accessKey=AKIAYS2NV46LK4IRGDL2
cloud.aws.credentials.secretKey=AVbiXM1o3GKPpbc2pRSQbaDzMbZGgxn8vKjXl
cloud.aws.s3.bucket=popcon-s3-bucket
cloud.aws.region.static=ap-southeast-2
cloud.aws.stack.auto=false

```

### 3. 배포 환경 설정

#### EC2 설정

##### 1. EC2 접속(CMD)

```
ssh -i I10C211T.pem ubuntu@i10c211.p.ssafy.io
```

##### 2. 서버 시간 변경

```
sudo timedatectl set-timezone Asia/Seoul
```

##### 3. 미러 서버 변경

```

sudo vi /etc/apt/sources.list
:s/ ap-northeast-2.ec2.archive.ubuntu.com/mirror.kakao.com/

```

2번째 명령어는 밑에 사진처럼 떼을 때 입력해주고, **esc**누르고 **:wq**하면 다시 터미널로

```
ubuntu@ip-172-26-2-127: ~  
## Note, this file is written by cloud-init on first boot of an instance  
## modifications made here will not survive a re-bundle.  
## if you wish to make changes you can:  
## a.) add 'apt_preserve_sources_list: true' to /etc/cloud/cloud.cfg  
##    or do the same in user-data  
## b.) add sources in /etc/apt/sources.list.d  
## c.) make changes to template file /etc/cloud/templates/sources.list.tpl  
  
# See http://help.ubuntu.com/community/UpgradeNotes for how to upgrade to  
# newer versions of the distribution.  
deb http://ap-northeast-2.ec2.archive.ubuntu.com/ubuntu/ focal main restricted  
# deb-src http://ap-northeast-2.ec2.archive.ubuntu.com/ubuntu/ focal main restricted  
  
## Major bug fix updates produced after the final release of the  
## distribution.  
deb http://ap-northeast-2.ec2.archive.ubuntu.com/ubuntu/ focal-updates main restricted  
# deb-src http://ap-northeast-2.ec2.archive.ubuntu.com/ubuntu/ focal-updates main restricted  
  
## N.B. software from this repository is ENTIRELY UNSUPPORTED by the Ubuntu  
## team. Also, please note that software in universe WILL NOT receive any  
## review or updates from the Ubuntu security team.  
deb http://ap-northeast-2.ec2.archive.ubuntu.com/ubuntu/ focal universe  
# deb-src http://ap-northeast-2.ec2.archive.ubuntu.com/ubuntu/ focal universe  
deb http://ap-northeast-2.ec2.archive.ubuntu.com/ubuntu/ focal-updates universe  
# deb-src http://ap-northeast-2.ec2.archive.ubuntu.com/ubuntu/ focal-updates universe  
  
## N.B. software from this repository is ENTIRELY UNSUPPORTED by the Ubuntu  
## team, and may not be under a free licence. Please satisfy yourself as to  
## your rights to use the software. Also, please note that software in
```

#### 4. 패키지 업데이트 및 업그레이드

```
sudo apt update
```

위 명령어 입력시 아래처럼 오류가 나오면 다음 코드 실행

```
Err:11 http://ppa.launchpad.net/certbot/certbot/ubuntu focal Release  
404 Not Found [IP: 172.26.2.127]  
E: The repository 'http://ppa.launchpad.net/certbot/certbot/ubuntu focal Release' does not have a Release file.  
N: Updating from such a repository can't be done securely, and is therefore disabled by default.  
N: See apt-secure(8) manpage for repository creation and user configuration details.
```

```
sudo add-apt-repository --remove ppa:certbot/certbot
```

위 코드 입력하면 아래 사진처럼 나오고 **ENTER**누르면 됩니다.

```
ubuntu@ip-172-26-2-127:~$ sudo add-apt-repository --remove ppa:certbot/certbot  
The PPA has been DEPRECATED.  
  
To get up to date instructions on how to get certbot for your systems, please see https://certbot.eff.org/docs/install.html.  
More info: https://launchpad.net/~certbot/+archive/ubuntu/certbot  
Press [ENTER] to continue or Ctrl-c to cancel removing it.
```

다시 `sudo apt update`하고, `sudo apt upgrade` 하면 끝

#### 5. 가상 메모리 할당



```
free -h
```

현재 메모리 용량 확인 명령어

```
ubuntu@ip-172-26-2-127:~$ free -h
               total        used        free      shared  buff/cache   available
Mem:            15Gi       899Mi       12Gi          1.0Mi       1.9Gi       14Gi
Swap:            0B           0B           0B
```

스왑 파일을 사용하여 Amazon EC2 인스턴스의 스왑 공간으로 메모리 할당

Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스에서 스왑 파일로 사용할 메모리를 할당하려고 합니다. 어떻게 해야 하나요?

 <https://repost.aws/ko/knowledge-center/ec2-memory-swap-file>



위 내용 대로 진행하면 됨.

현재 저의 노트북 램은 **16GB** (코치세션에서 램 절반할당 권장)

따라서 8GB를 할당 시키려고한다.

```
sudo dd if=/dev/zero of=/swapfile bs=128M count=64
```

```
ubuntu@ip-172-26-2-127:~$ sudo dd if=/dev/zero of=/swapfile bs=128M count=64
64+0 records in
64+0 records out
8589934592 bytes (8.6 GB, 8.0 GiB) copied, 46.7766 s, 184 MB/s
```

스왑 파일의 읽기 및 쓰기 권한을 업데이트

```
sudo chmod 600 /swapfile
```

Linux 스왑 영역을 설정

```
sudo mkswap /swapfile
```

```
ubuntu@ip-172-26-2-127:~$ sudo chmod 600 /swapfile
ubuntu@ip-172-26-2-127:~$ sudo mkswap /swapfile
Setting up swspace version 1, size = 8 GiB (8589930496 bytes)
no label, UUID=c69ea35c-d4a8-4536-9859-6546f66b1f52
ubuntu@ip-172-26-2-127:~$
```

스왑 공간에 스왑 파일을 추가하여 스왑 파일을 즉시 사용

```
sudo swapon /swapfile
```

프로시저가 성공적인지 확인

```
sudo swapon -s
```

```
ubuntu@ip-172-26-2-127:~$ sudo swapon /swapfile
ubuntu@ip-172-26-2-127:~$ sudo swapon -s
```

Filename	Type	Size	Used	Priority
/swapfile	file	8388604	0	-2

```
ubuntu@ip-172-26-2-127:~$
```

```
/etc/fstab 파일을 편집하여 부팅 시 스왑파일을 시작
편집기에서 파일을 연다
sudo vi /etc/fstab
파일 끝에 다음 줄을 새로 추가하고 파일을 저장한 다음 종료
/swapfile swap swap defaults 0 0
나올때 역시 :wq
```

[illegible]

free -h 할당되었는지 확인해보자!

```

ubuntu@ip-172-26-2-127:~$ free -h
              total        used        free      shared  buff/cache   available
Mem:           15Gi       891Mi       4.6Gi       1.0Mi       10Gi       14Gi
Swap:          8.0Gi           0B       8.0Gi

```

## NginX&SSL

## 방화벽 설정

방화벽 확인

```
sudo ufw status
```

방화벽 허용

```
sudo ufw allow {80}
```

```
ubuntu@ip-172-26-2-127:~$ sudo ufw status
Status: active

To Action From
--
22 ALLOW Anywhere
8989 ALLOW Anywhere
443 ALLOW Anywhere
22 (v6) ALLOW Anywhere (v6)
8989 (v6) ALLOW Anywhere (v6)
443 (v6) ALLOW Anywhere (v6)
```

```
ubuntu@ip-172-26-2-127:~$ sudo ufw allow 80
Rule added
Rule added (v6)
ubuntu@ip-172-26-2-127:~$ sudo ufw status
Status: active

To Action From
--
22 ALLOW Anywhere
8989 ALLOW Anywhere
443 ALLOW Anywhere
80 ALLOW Anywhere
22 (v6) ALLOW Anywhere (v6)
8989 (v6) ALLOW Anywhere (v6)
443 (v6) ALLOW Anywhere (v6)
80 (v6) ALLOW Anywhere (v6)
```

## NginX 설치

- 오픈소스 웹 서버 프로그램
- 외에도 Apache, IIS, LiteSpeed 등이 있음.

Nginx설치

```
sudo apt install nginx -y
```

NginX 상태 확인

```
sudo systemctl status nginx
```

```
ubuntu@ip-172-26-2-127:~$ sudo systemctl status nginx
● nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2024-02-07 22:42:43 KST; 46s ago
     Docs: man:nginx(8)
  Main PID: 74293 (nginx)
    Tasks: 5 (limit: 19165)
   Memory: 6.8M
    CGroup: /system.slice/nginx.service
            └─74293 nginx: master process /usr/sbin/nginx -g daemon on; master_process on;
               └─74294 nginx: worker process
                 └─74295 nginx: worker process
                   └─74296 nginx: worker process
                     └─74297 nginx: worker process

Feb 07 22:42:43 ip-172-26-2-127 systemd[1]: Starting A high performance web server and a reverse proxy server...
Feb 07 22:42:43 ip-172-26-2-127 systemd[1]: Started A high performance web server and a reverse proxy server.
```

default파일로 NginX 환경설정

```
sudo vi /etc/nginx/sites-available/default
```

## SSL설정

```
# letsencrypt 설치
sudo apt-get install letsencrypt
# Certbot 설치
sudo apt-get install certbot python3-certbot-nginx
```

### Certbot Nginx 연결

```
sudo certbot --nginx
# 저는 이메일 입력 약관동의 이메일 수신동의 등이 안나왔다.
# 곧바로 도메인 입력 i10c211.p.ssafy.io 입력하고
# http입력시 리다이렉트 여부 2를 선택했다. (HTTP로 입력해도 HTTPS로 바꿔주는 선택
```

```
ubuntu@ip-172-26-2-127:~$ sudo certbot --nginx
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Plugins selected: Authenticator nginx, Installer nginx
No names were found in your configuration files. Please enter in your domain
name(s) (comma and/or space separated) (Enter 'c' to cancel): i10c211.p.ssafy.io
Obtaining a new certificate
Performing the following challenges:
http-01 challenge for i10c211.p.ssafy.io
Waiting for verification...
Cleaning up challenges
Deploying Certificate to VirtualHost /etc/nginx/sites-enabled/default

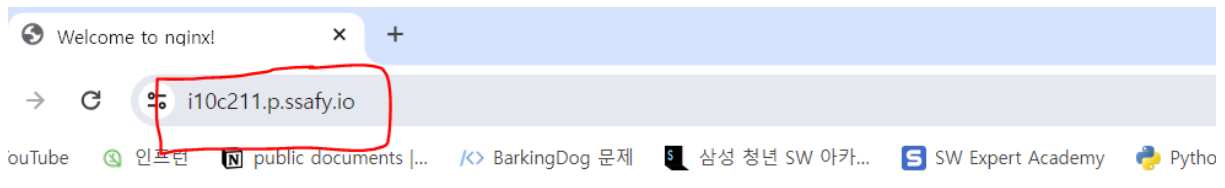
Please choose whether or not to redirect HTTP traffic to HTTPS, removing HTTP access.
-----
1: No redirect - Make no further changes to the webserver configuration.
2: Redirect - Make all requests redirect to secure HTTPS access. Choose this for
new sites, or if you're confident your site works on HTTPS. You can undo this
change by editing your web server's configuration.
-----
Select the appropriate number [1-2] then [enter] (press 'c' to cancel): 2
Redirecting all traffic on port 80 to ssl in /etc/nginx/sites-enabled/default
-----
Congratulations! You have successfully enabled https://i10c211.p.ssafy.io

You should test your configuration at:
https://www.ssllabs.com/ssltest/analyze.html?d=i10c211.p.ssafy.io
-----

IMPORTANT NOTES:
- Congratulations! Your certificate and chain have been saved at:
  /etc/letsencrypt/live/i10c211.p.ssafy.io/fullchain.pem
  Your key file has been saved at:
  /etc/letsencrypt/live/i10c211.p.ssafy.io/privkey.pem
  Your cert will expire on 2024-05-07. To obtain a new or tweaked
  version of this certificate in the future, simply run certbot again
  with the "certonly" option. To non-interactively renew *all* of
  your certificates, run "certbot renew"
- If you like Certbot, please consider supporting our work by:

  Donating to ISRG / Let's Encrypt: https://letsencrypt.org/donate
  Donating to EFF: https://eff.org/donate-le

ubuntu@ip-172-26-2-127:~$
```



## 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](https://nginx.org).  
Commercial support is available at [nginx.com](https://nginx.com).

*Thank you for using nginx.*

## Docker 설치

- apt package index 업데이트

```
sudo apt-get update
```

- Repository 등록을 위한 필요 패키지 설치

```
sudo apt-get install \
    ca-certificates \
    curl \
    gnupg \
    lsb-release
```

- Docker 공식 GPG-Key 추가

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg
```

- Docker Repository 추가

```
echo \
"deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyr
$(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker
```

- Docker engine 설치

```
sudo apt-get update
sudo apt-get install docker-ce docker-ce-cli containerd.io
```

## Ngix sites-enabled 설정

```
server {
    listen 80 default_server;
    listen [::]:80 default_server;

    root /var/www/html;

    # Add index.php to the list if you are using PHP
    index index.html index.htm index.nginx-debian.html;

    server_name _;

    include /etc/nginx/conf.d/service-url.inc;

    location / {
        # First attempt to serve request as file, then
        # as directory, then fall back to displaying a 404.
        #try_files $uri $uri/ =404;
        proxy_pass $service_url;
    }
    location /static {
        alias /path/to/your/flask_app/static; # Flask 애플리케이션의 static
    }
}

server {
    listen 443 ssl http2;
    server_name i10c211.p.ssafy.io;

    # ssl 인증서 적용하기
    ssl_certificate /etc/letsencrypt/live/i10c211.p.ssafy.io/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/i10c211.p.ssafy.io/privatekey.pem;
    ssl_protocols TLSv1 TLSv1.1 TLSv1.2;

    location / { # location 이후 특정 url을 처리하는 방법을 정의(여기서는 / ->
        #proxy_pass https://localhost:9001; # Request에 대해 어디로 리다이렉트
```

```

    proxy_pass http://127.0.0.1:8081;
    proxy_set_header Host $http_host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto $scheme;
}

}

server {
    listen 8085 ssl http2;
    server_name i10c211.p.ssafy.io;

    # ssl 인증서 적용하기
    ssl_certificate /etc/letsencrypt/live/i10c211.p.ssafy.io/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/i10c211.p.ssafy.io/privatekey.pem;
    ssl_protocols TLSv1 TLSv1.1 TLSv1.2;

    location / { # location 이후 특정 url을 처리하는 방법을 정의(여기서는 / ->
        #proxy_pass https://localhost:9001; # Request에 대해 어디로 리다이렉트
        proxy_pass http://127.0.0.1:8080;
        proxy_set_header Host $http_host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }

}

server {
    listen 5005 ssl http2;
    server_name i10c211.p.ssafy.io;

    # ssl 인증서 적용하기
    ssl_certificate /etc/letsencrypt/live/i10c211.p.ssafy.io/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/i10c211.p.ssafy.io/privatekey.pem;
    ssl_protocols TLSv1 TLSv1.1 TLSv1.2;

    location /upload { # location 이후 특정 url을 처리하는 방법을 정의(여기서는
        #proxy_pass https://localhost:9001; # Request에 대해 어디로 리다이렉트

```

```

proxy_pass http://127.0.0.1:5000;
proxy_set_header Host $http_host;
proxy_set_header X-Real-IP $remote_addr;
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
proxy_set_header X-Forwarded-Proto $scheme;
}

}

server {
    if ($host = i10c211.p.ssafy.io) {
        return 308 https://$host$request_uri;
    } # managed by Certbot

    listen 80;
    server_name i10c211.p.ssafy.io;
    return 404; # managed by Certbot
}

```

## 4. 빌드 및 배포

### 프론트엔드

1. Node.js 환경에서 FE\ 디렉토리로 이동
2. npm install을 통해 package\*.json에 정의된 패키지를 다운로드
  - 참고) 로컬에서 확인할 때 이용하는 코드

```

npm install
npm run build; serve -s dist

```

3. 해당 폴더에서 아래의 명령어를 입력하여 배포 버전 파일 생성

```

npm run build

```

4. 프론트엔드 프로젝트 하위에 Dockerfile 생성

```

FROM node:20.11.0

RUN apt-get update && apt-get install -y xsel

```



```

WORKDIR /home/ubuntu/S10P12C211/FE

COPY package*.json ./

ADD . .

RUN npm install

COPY . .
RUN npm run build
EXPOSE 8081

CMD ["serve", "-s", "dist", "-l", "8081"]

```

#### 5. EC2 서버에서 도커 빌드, 컨테이너 띄우기

```

$ sudo docker build -t front:latest .
$ sudo docker run -d -p 8081:8081 --name front front:latest

```

#### 6. 도커 컨테이너에 대한 방화벽 규칙 추가

```

$ sudo ufw-docker allow front 8081

```

## 백엔드

#### 1. gradle build

```

./ gradlew build

```

#### 2. 백엔드 프로젝트 하위에 Dockerfile 생성

```

FROM openjdk:8-jdk

ARG JAR_FILE=./build/libs/*.jar

COPY ${JAR_FILE} app.jar

ENTRYPOINT ["java", "-jar", "/app.jar"]

```

### 3. 로컬에서 도커 이미지 생성

```
docker build -t popcondocker/backend .
```

- 윈도우에서 docker desktop 이 켜져있어야 함
- 본인의 도커허브아이디/레포지토리 명 작성

### 4. 생성한 도커 이미지를 도커 허브에 push

```
docker login  
  
docker push popcondocker/backend
```

### 5. ec2 접속 후 docker image pull

```
// 도커 실행  
sudo systemctl start docker  
  
// 이미지 pull  
sudo docker pull popcondocker/backend
```

### 5. 실행

```
sudo docker run -d -p 8080:8080 popcondocker/backend
```

## FLASK

- 참고) 로컬에서 확인할 때 이용하는 코드

```
set FLASK_APP=app  
flask run --debug
```

### 1. Flask 프로젝트 하위에 Dockerfile 생성

```

FROM python:3.9

# 작업 디렉토리를 설정합니다.
WORKDIR /home/ubuntu/S10P12C211/Flask

# 필요한 파일을 복사하고 패키지를 설치합니다.
COPY requirements.txt .
RUN apt-get update
RUN apt-get -y install libgl1-mesa-glx
RUN pip install --no-cache-dir -r requirements.txt
RUN pip install flask_cors

#YOLOv8 custom model사용하기 위한 추가 코드
RUN pip install --upgrade ultralytics
RUN pip install --force-reinstall ultralytics

# gunicorn을 설치하기 위한 pip 명령어
RUN pip install gunicorn

# Flask 애플리케이션 파일을 복사합니다.
COPY app.py .

# Flask 애플리케이션을 실행합니다.
CMD ["gunicorn", "app:app", "-w", "4", "--bind", "0.0.0.0:5000"]

```

## 2. EC2 서버에서 도커 빌드, 컨테이너 띄우기

```

$sudo docker build -t flask:latest .
$sudo docker run -d -p 5000:5000 --name flask flask:latest

```

## 3. 도커 컨테이너에 대한 방화벽 규칙 추가

```

$sudo ufw-docker allow flask 5000

```

## 포트 개방

```
ubuntu@ip-172-26-2-127:~$ sudo ufw status
```

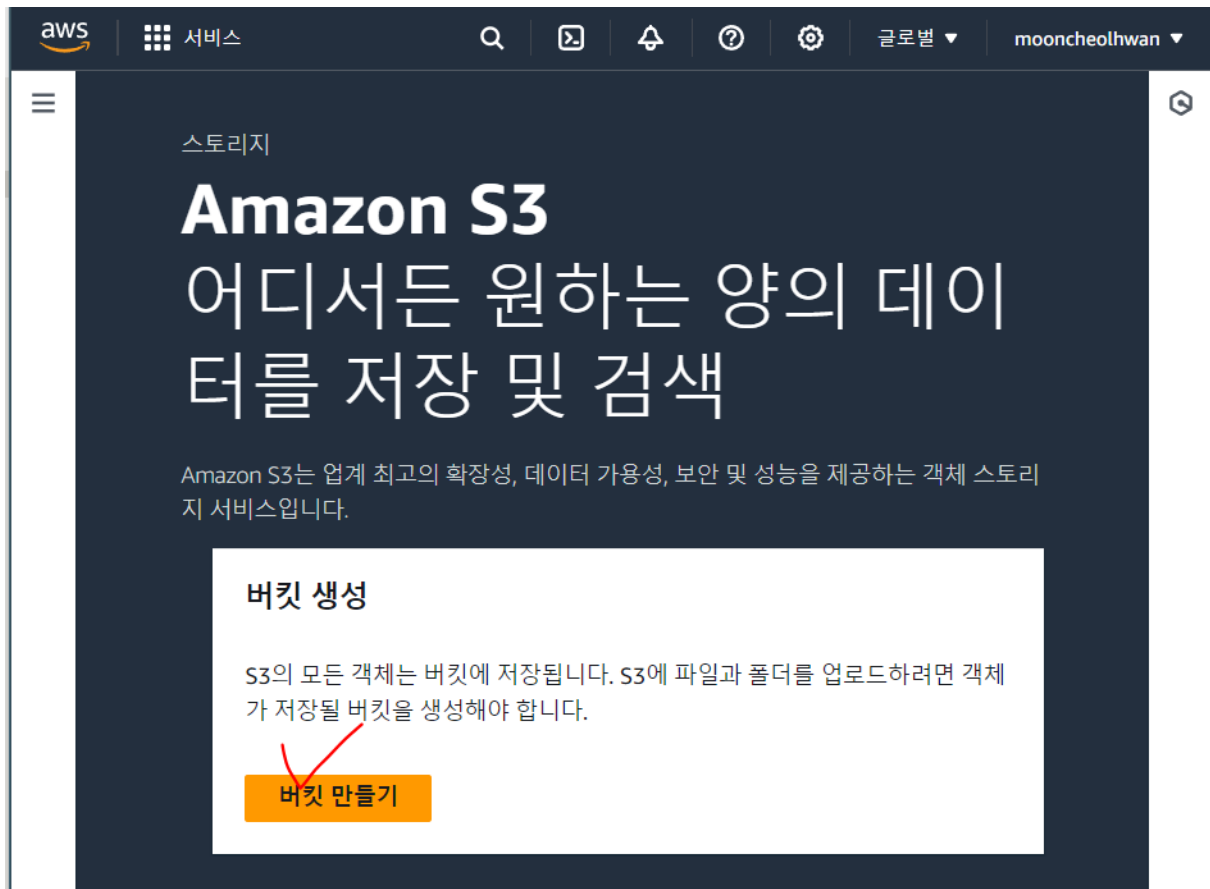
```
Status: active
```

To	Action	From
--	-----	----
22	ALLOW	Anywhere
8989	ALLOW	Anywhere
443	ALLOW	Anywhere
80	ALLOW	Anywhere
3306/tcp	ALLOW	Anywhere
3306	ALLOW	Anywhere
8081	ALLOW	Anywhere
5000	ALLOW	Anywhere
8085	ALLOW	Anywhere
5005	ALLOW	Anywhere
22 (v6)	ALLOW	Anywhere (v6)
8989 (v6)	ALLOW	Anywhere (v6)
443 (v6)	ALLOW	Anywhere (v6)
80 (v6)	ALLOW	Anywhere (v6)
3306/tcp (v6)	ALLOW	Anywhere (v6)
3306 (v6)	ALLOW	Anywhere (v6)
8081 (v6)	ALLOW	Anywhere (v6)
5000 (v6)	ALLOW	Anywhere (v6)
8085 (v6)	ALLOW	Anywhere (v6)
5005 (v6)	ALLOW	Anywhere (v6)
3306/tcp	ALLOW OUT	Anywhere
3306/tcp (v6)	ALLOW OUT	Anywhere (v6)
172.17.0.2 8080/tcp	ALLOW FWD	Anywhere
172.17.0.3 5000/tcp	ALLOW FWD	Anywhere

## 외부서비스

### AWS S3 버킷 생성 및 권한 설정

- 버킷 만들기



- AWS 리전은 어떤 지역의 AWS 서버가 파일을 호스팅 할 지 결정

## 버킷 만들기 정보

버킷은 S3에 저장되는 데이터의 컨테이너입니다. [자세히 알아보기](#)

### 일반 구성

AWS 리전

✓ 아시아 태평양(서울) ap-northeast-2 ▼

버킷 이름 정보

✓ popcon

버킷 이름은 글로벌 네임스페이스 내에서 고유해야 하며 버킷 이름 지정 규칙을 따라야 합니다. [버킷 이름을 지정 규칙 보기](#)

기존 버킷에서 설정 복사 - 선택 사항

다음 구성의 버킷 설정만 복사됩니다.

버킷 선택

형식: s3://bucket/prefix

#### • 객체 소유권 설정

### 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.

#### ☒ 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.

객체 소유권

버킷 소유자 적용

#### • 퍼블릭 액세스 차단 설정

- 실무에서 사용할 경우, 모든 액세스 차단 혹은 ACL을 이용하여 액세스 차단해주는 것이 보안을 위해 좋다

☐ **모든 퍼블릭 액세스 차단**

이 설정을 활성화하면 아래 4개의 설정을 모두 활성화한 것과 같습니다. 다음 설정 각각은 서로 독립적입니다.

- ☐ **새 ACL(액세스 제어 목록)을 통해 부여된 버킷 및 객체에 대한 퍼블릭 액세스 차단**  
S3은 새로 추가된 버킷 또는 객체에 적용되는 퍼블릭 액세스 권한을 차단하며, 기존 버킷 및 객체에 대한 새 퍼블릭 액세스 ACL 생성을 금지합니다. 이 설정은 ACL을 사용하여 S3 리소스에 대한 퍼블릭 액세스를 허용하는 기존 권한을 변경하지 않습니다.
- ☐ **임의의 ACL(액세스 제어 목록)을 통해 부여된 버킷 및 객체에 대한 퍼블릭 액세스 차단**  
S3은 버킷 및 객체에 대한 퍼블릭 액세스를 부여하는 모든 ACL을 무시합니다.
- ☒ **새 퍼블릭 버킷 또는 액세스 지점 정책을 통해 부여된 버킷 및 객체에 대한 퍼블릭 액세스 차단**  
S3은 버킷 및 객체에 대한 퍼블릭 액세스를 부여하는 새 버킷 및 액세스 지점 정책을 차단합니다. 이 설정은 S3 리소스에 대한 퍼블릭 액세스를 허용하는 기존 정책을 변경하지 않습니다.
- ☐ **임의의 퍼블릭 버킷 또는 액세스 지점 정책을 통해 부여된 버킷 및 객체에 대한 퍼블릭 및 교차 계정 액세스 차단**  
S3은 버킷 및 객체에 대한 퍼블릭 액세스를 부여하는 정책을 사용하는 버킷 또는 액세스 지점에 대한 퍼블릭 및 교차 계정 액세스를 무시합니다.




**모든 퍼블릭 액세스 차단을 비활성화하면 이 버킷과 그 안에 포함된 객체가 퍼블릭 상태가 될 수 있습니다.**

정적 웹 사이트 호스팅과 같은 구체적으로 확인된 사용 사례에서 퍼블릭 액세스가 필요한 경우가 아니면 모든 퍼블릭 액세스 차단을 활성화하는 것이 좋습니다.

- ☒ 현재 설정으로 인해 이 버킷과 그 안에 포함된 객체가 퍼블릭 상태가 될 수 있습니다.

- 버킷 버전 관리, 기본 암호화 설정
  - 보안상 기본 암호화도 활성화를 하는게 좋지만, 기본 생성 후 테스트를 위해 비활성화 설정
  - 설정 완료 후 **버킷 만들기** 버튼 클릭

## 버킷 버전 관리

버전 관리는 객체의 여러 버전을 동일한 버킷에서 관리하기 위한 수단입니다. 버전 관리를 사용하여 Amazon S3 버킷에 저장된 모든 객체의 각 버전을 보존, 검색 및 복원할 수 있습니다. 버전 관리를 통해 의도치 않은 사용자 작업과 애플리케이션 장애를 모두 복구할 수 있습니다. [자세히 알아보기](#) 


### 버킷 버전 관리

- ☒ 비활성화
- ☐ 활성화


## 기본 암호화 정보

서버 측 암호화는 이 버킷에 저장된 새 객체에 자동으로 적용됩니다.

### 암호화 유형 정보

- ☒ Amazon S3 관리형 키(SSE-S3)를 사용한 서버 측 암호화
- ☐ AWS Key Management Service 키를 사용한 서버 측 암호화(SSE-KMS)
- ☐ AWS Key Management Service 키를 사용한 이중 계층 서버 측 암호화(DSSE-KMS)  
두 개의 개별 암호화 계층으로 객체를 보호합니다. 요금에 대한 자세한 내용은 [Amazon S3 요금 페이지](#) 의 스토리지 탭에서 **DSSE-KMS** 요금을 참조하세요.

### 버킷 키

SSE-KMS용 S3 버킷 키를 사용하면 AWS KMS에 대한 호출을 줄여 암호화 비용이 절감됩니다. DSSE-KMS에는 S3 버킷 키가 지원되지 않습니다. [자세히 알아보기](#) 

- ☐ 비활성화
- ☒ 활성화



## 범용 버킷 (1) 정보



ARN 복사

비어 있음

삭제


버킷 만들기

버킷은 S3에 저장되는 데이터의 컨테이너입니다. [자세히 알아보기](#)

 이름으로 버킷 찾기

< 1 >



	이름 ▲	AWS 리전 ▼	액세스 ▼	생성 날짜 ▼
	pop-con	아시아 태평양 (서울) ap- northeast-2	객체를 퍼블릭 으로 설정할 수 있음	2024. 2. 8. am 9:20:34 AM KST

- 파일 업로드 테스트

파일 및 폴더 (1 합계, 260.4KB)

제거

파일 추가

폴더 추가

이 테이블의 모든 파일과 폴더가 업로드됩니다.

🔍

이름으로 찾기

<

1

>

☐

이름

▼

폴더

▼

유형

▼

크기

▼

대상 정보

대상

s3://pop-con

▶ 대상 세부 정보

지정된 대상에 저장된 새 객체에 영향을 미치는 버킷 설정.

▶ 권한

다른 AWS 계정에 퍼블릭 액세스 및 액세스 권한을 부여합니다.

▶ 속성

스토리지 클래스, 암호화 설정, 태그 등을 지정합니다.

취소

업로드

✔ 업로드 성공  
아래에서 세부 정보를 확인합니다.

s3://pop-con

✔ 1개 파일, 260.4KB (100.00%)

실패

0개 파일, 0B (0%)

파일 및 폴더

구성

파일 및 폴더 (1 합계, 260.4KB)

🔍 이름으로 찾기

< 1 >

이름	폴더	유형	크기	상태
qq.PNG	-	image/png	260.4KB	✔ 성공

Amazon S3 > 버킷 > pop-con > qq.PNG

qq.PNG 정보

S3 URI 복사

다운로드

열기

속성

권한

버전

객체 개요

소유자

12bc2fb0eee2eacca73fc31763f9df55bb171bd2240d6258ab16d6443694721d

AWS 리전

아시아 태평양(서울) ap-northeast-2

마지막 수정

2024. 2. 8. am 9:22:55 AM KST

크기

260.4KB

유형

PNG

키

S3 URI

s3://pop-con/qq.PNG

Amazon 리소스 이름(ARN)

arn:aws:s3:::pop-con/qq.PNG

엔터티 태그(Etag)

1ae6509911406792b4523f9ef9ba28a0

객체 URL

https://pop-con.s3.ap-northeast-2.amazonaws.com/qq.PNG

- Access Denied 권한 수정
  - 해당하는 Bucket의 권한 - 퍼블릭 액세스 차단(버킷 설정) - 편집 클릭

pop-con
정보

객체
속성
**권한**
지표
관리
액세스 지정

권한 개요

액세스

객체를 퍼블릭으로 설정할 수 있음

퍼블릭 액세스 차단(버킷 설정)

편집

퍼블릭 액세스는 ACL(액세스 제어 목록), 버킷 정책, 액세스 지정 정책 또는 모두를 통해 버킷 및 객체에 부여됩니다. 모든 S3 버킷 및 객체에 대한 퍼블릭 액세스가 차단되었는지 확인하려면 [모든 퍼블릭 액세스 차단]을 활성화합니다. 이 설정은 이 버킷 및 해당 액세스 지정에만 적용됩니다. AWS에서는 [모든 퍼블릭 액세스 차단]을 활성화하도록 권장하지만, 이 설정을 적용하기 전에 퍼블릭 액세스가 없어도 애플리케이션이 올바르게 작동하는지 확인합니다. 버킷 또는 내부 객체에 어느 정도 수준의 퍼블릭 액세스가 필요한 경우 특정 스토리지 사용 사례에 맞게 아래 개별 설정을 사용자 지정할 수 있습니다. [자세히 알아보기](#)

퍼블릭 액세스 차단 편집(버킷 설정)
정보

퍼블릭 액세스 차단(버킷 설정)

퍼블릭 액세스는 ACL(액세스 제어 목록), 버킷 정책, 액세스 지정 정책 또는 모두를 통해 버킷 및 객체에 부여됩니다. 모든 S3 버킷 및 객체에 대한 퍼블릭 액세스가 차단되었는지 확인하려면 [모든 퍼블릭 액세스 차단]을 활성화합니다. 이 설정은 이 버킷 및 해당 액세스 지정에만 적용됩니다. AWS에서는 [모든 퍼블릭 액세스 차단]을 활성화하도록 권장하지만, 이 설정을 적용하기 전에 퍼블릭 액세스가 없어도 애플리케이션이 올바르게 작동하는지 확인합니다. 버킷 또는 내부 객체에 어느 정도 수준의 퍼블릭 액세스가 필요한 경우 특정 스토리지 사용 사례에 맞게 아래 개별 설정을 사용자 지정할 수 있습니다. [자세히 알아보기](#)

☐ 모든 퍼블릭 액세스 차단

이 설정을 활성화하면 아래 4개의 설정을 모두 활성화한 것과 같습니다. 다음 설정 각각은 서로 독립적입니다.

☐ 새 ACL(액세스 제어 목록)을 통해 부여된 버킷 및 객체에 대한 퍼블릭 액세스 차단

S3은 새로 추가된 버킷 또는 객체에 적용되는 퍼블릭 액세스 권한을 차단하며, 기존 버킷 및 객체에 대한 새 퍼블릭 액세스 ACL 생성을 금지합니다. 이 설정은 ACL을 사용하여 S3 리소스에 대한 퍼블릭 액세스를 허용하는 기존 권한을 변경하지 않습니다.

☐ 임의의 ACL(액세스 제어 목록)을 통해 부여된 버킷 및 객체에 대한 퍼블릭 액세스 차단

S3은 버킷 및 객체에 대한 퍼블릭 액세스를 부여하는 모든 ACL을 무시합니다.

☒ 새 퍼블릭 버킷 또는 액세스 지정 정책을 통해 부여된 버킷 및 객체에 대한 퍼블릭 액세스 차단

S3은 버킷 및 객체에 대한 퍼블릭 액세스를 부여하는 새 버킷 및 액세스 지정 정책을 차단합니다. 이 설정은 S3 리소스에 대한 퍼블릭 액세스를 허용하는 기존 정책을 변경하지 않습니다.

☐ 임의의 퍼블릭 버킷 또는 액세스 지정 정책을 통해 부여된 버킷 및 객체에 대한 퍼블릭 및 교차 계정 액세스 차단

S3은 버킷 및 객체에 대한 퍼블릭 액세스를 부여하는 정책을 사용하는 버킷 또는 액세스 지정에 대한 퍼블릭 및 교차 계정 액세스를 무시합니다.

취소

변경 사항 저장

해피캣 포팅 매뉴얼

28



## 확인

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 :

Effect ☒ Allow ☐ Deny

Principal  ✓

Use a comma to separate multiple values.

AWS Service  ☐

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:::${BucketName}/${KeyName}`.  
Use a comma to separate multiple values.

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

✓

You added the following statements. Click the button below to Generate a policy.

Principal(s)	Effect	Action	Resource
• *	Allow	• s3:GetObject • s3:PutObject	arn:aws:s3:::pop-con

## Step 3: Generate Policy

A *policy* is a document (written in the [Access Policy Language](#)) that acts as a container for one or more statements.

✓  [Start Over](#)

- 생성된 Policy JSON Document 복사 후 닫기

JSON Document

elow to edit. To save the policy, copy the text below to a text editor.  
s made below will **not be reflected in the policy generator tool**.

```

d": "Policy1707352573645",
'version": "2012-10-17",
'statement": [
{
  "Sid": "Stmt1707352539518",
  "Action": [
    "s3:GetObject",
    "s3:PutObject"
  ],
  "Effect": "Allow",
  "Resource": "arn:aws:s3:::pop-con",
  "Principal": "*"
}

```


- 위에 복사한 정책 밑에 붙여넣기 한 후 변경사항 저장

## 버킷 정책 편집 정보

### 버킷 정책

JSON으로 작성된 버킷 정책은 버킷에 저장된 객체에 대한 액세스 권한을 제공합니다. 버킷 정책은 다른 계정이 소유한 객체에는 적용되지 않습니다.

#### 버킷 ARN

 arn:aws:s3:::pop-con

### 정책

```

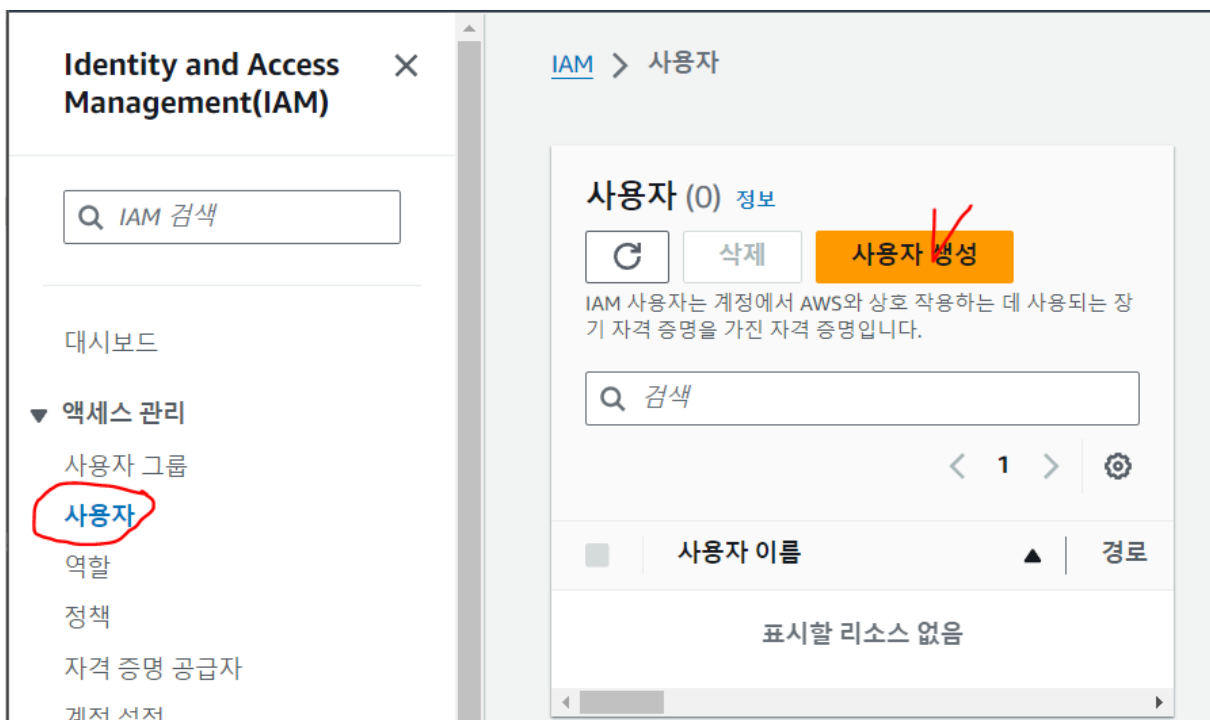
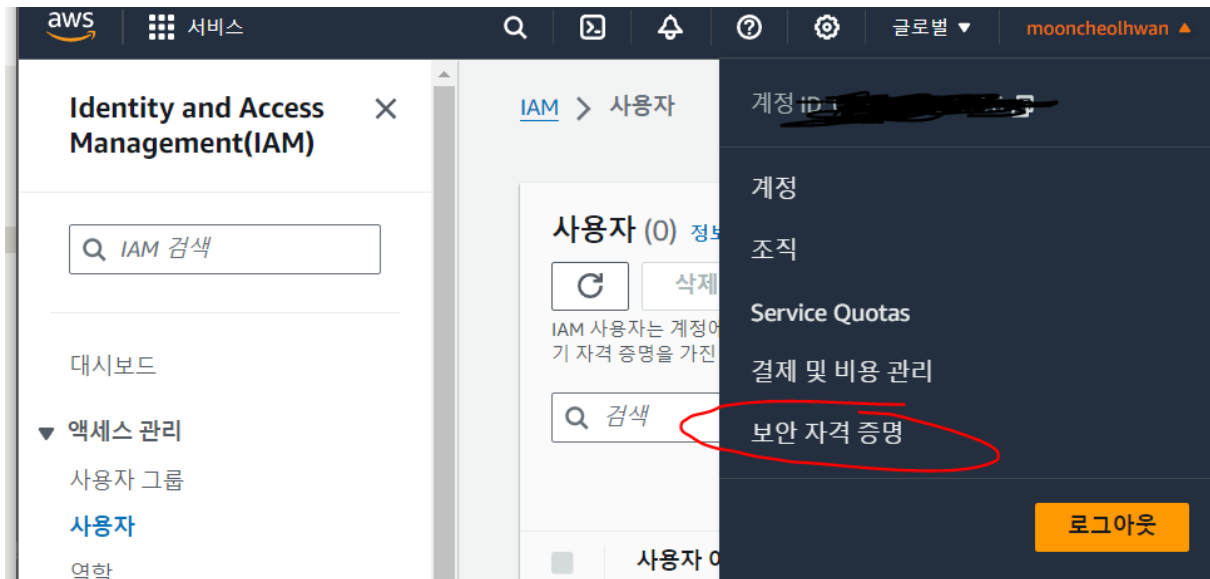
1  {
2    "Id": "Policy1707352573645",
3    "Version": "2012-10-17",
4    "Statement": [
5      {
6        "Sid": "Stmt1707352539518",
7        "Action": [
8          "s3:GetObject",
9          "s3:PutObject"
10       ],
11        "Effect": "Allow",
12        "Resource": "arn:aws:s3:::pop-con",
13        "Principal": "*"
14      }
15    ]

```

복사

## 스프링 부트에서 AWS S3를 사용 설정

## IAM 사용자 생성





## 사용자 세부 정보


사용자 이름

cheolhwan

사용자 이름은 최대 64자까지 가능합니다. 유효한 문자: A~Z, a~z, 0~9 및 +, =, ., @, \_ -(하이픈)

☐ AWS Management Console에 대한 사용자 액세스 권한 제공 - 선택 사항

사용자에게 콘솔 액세스 권한을 제공하는 경우 IAM Identity Center에서 액세스를 관리하는 것은 모범 사례 [입니다](#).

 이 IAM 사용자를 생성한 후 액세스 키 또는 AWS CodeCommit이나 Amazon Keyspaces에 대한 서비스별 보안 인증 정보를 통해 프로그래밍 방식 액세스를 생성할 수 있습니다. [자세히 알아보기](#)

취소

다음

## 권한 정책 (1/1175)

새 사용자에게 연결할 정책을 하나 이상 선택합니다.



정책 생성 [↗](#)

AmazonS3FullAccess



필터링 기준 유형

모든 유형 ▼

1 개 일치

< 1 >



정책 이름 [↗](#)



유형 ▼

연결된 ... ▼



 [AmazonS3FullAccess](#)

AWS 관리형

0

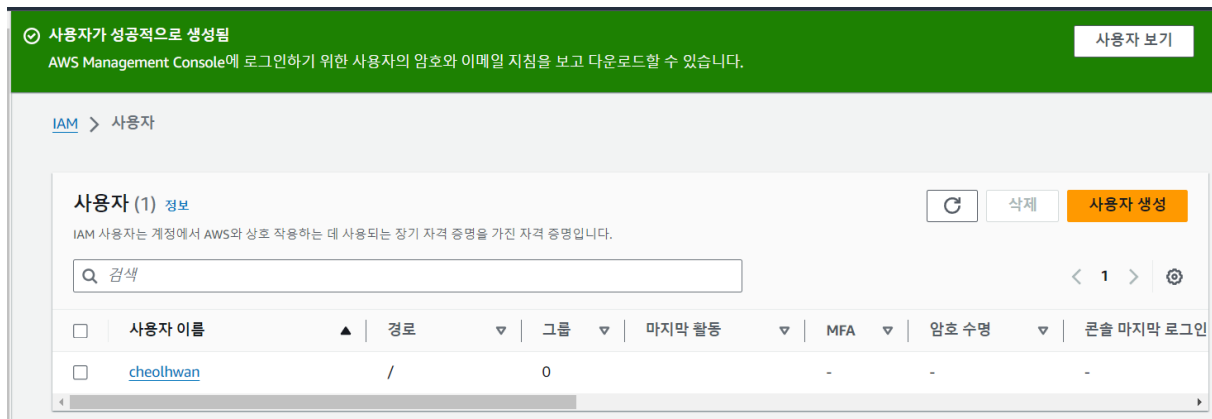
▶ 권한 경계 설정 - 선택 사항

취소

이전

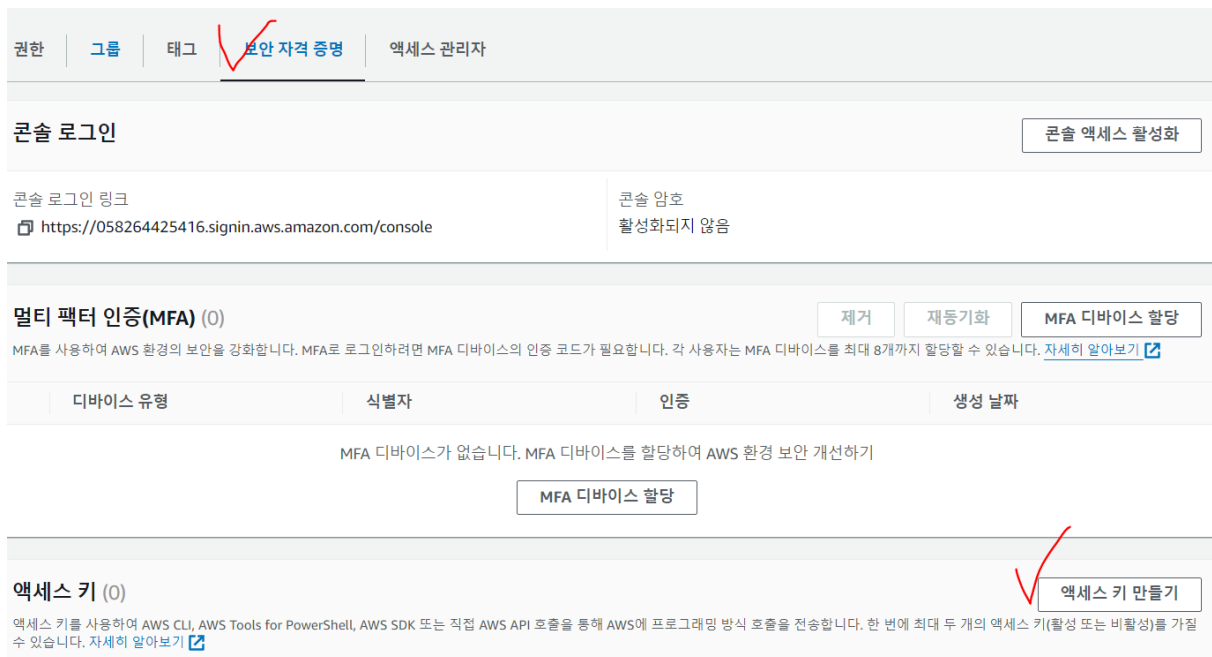
다음

이후 페이지에서 사용자 생성 버튼으로 만들기!



- IAM 사용자에게 Access Key 생성

위에 생성된 사용자 이름 클릭하기



☒ **AWS 외부에서 실행되는 애플리케이션**  
 이 액세스 키를 사용하여 AWS 리소스에 액세스해야 하는 AWS 외부의 데이터 센터 또는 기타 인프라에서 실행 중인 워크로드를 인증할 것입니다.

☐ **기타**  
 귀하의 사용 사례가 여기에 나열되어 있지 않습니다.

**권장되는 대안**  
 IAM Roles Anywhere를 사용하여 AWS 서비스에 액세스하는 비 AWS 워크로드에 대한 임시 보안 인증을 생성할 수 있습니다. [AWS 이외의 워크로드에 대한 액세스를 제공하는 방법에 대해 자세히 알아보세요.](#)

취소

다음

## 설명 태그 설정 - 선택 사항 정보

이 액세스 키에 대한 설명은 이 사용자에게 태그로 연결되고, 액세스 키와 함께 표시됩니다.

**설명 태그 값**  
 이 액세스 키의 용도와 사용 위치를 설명합니다. 좋은 설명은 나중에 이 액세스 키를 자신있게 교체하는 데 유용합니다.

최대 256자까지 가능합니다. 허용되는 문자는 문자, 숫자, UTF-8로 표현할 수 있는 공백 및 `_ . : / = + - @`입니다.

취소

이전

액세스 키 만들기

- 액세스 키 및 비밀 액세스 키는 IAM 권한 생성 후 단 한번만 나오고 그 이후로 **조회 불가능**하니 **반드시 메모**

### 액세스 키

분실하거나 잊어버린 비밀 액세스 키는 검색할 수 없습니다. 대신 새 액세스 키를 생성하고 이전 키를 비활성화합니다.

액세스 키	비밀 액세스 키
AKIAQ3EGUSPEDF44HNPJ	***** <a href="#">표시</a> <b>메모</b>

### 액세스 키 모범 사례

- 액세스 키를 일반 텍스트, 코드 리포지토리 또는 코드로 저장해서는 안 됩니다.
- 더 이상 필요 없는 경우 액세스 키를 비활성화하거나 삭제합니다.
- 최소 권한을 활성화합니다.
- 액세스 키를 정기적으로 교체합니다.

액세스 키 관리에 대한 자세한 내용은 [AWS 액세스 키 관리 모범 사례](#)를 참조하세요.

.csv 파일 다운로드

완료

## Spring Boot 연동

- build.gradle에 implement 추가

```
implementation 'org.springframework.cloud:spring-cloud-starter-aws:2.2.6.RELEASE'
```

- application.yml에 다음 내용 추가
  - cloud.aws.region.static: ap-northeast-2 (한국 리전 사용)
  - bucket 이름이 popcon 일 때

```
cloud:
  aws:
    s3:
      bucket: popcon
      base-url: https://aws-s3-ssafy.s3.ap-northeast-2.amazonaws.com
    credentials:
      access-key: AKIAQ3EGUSPEDF44HNPJ
      secret-key: 비밀엑세스키
    region:
      static: ap-northeast-2
      auto: false
    stack:
      auto: false
```

- 단, **cloud.aws.stack.auto**는 반드시 **false** 설정
  - cloud.aws.stack.auto를 false 설정하지 않으면 다음과 같은 오류 발생
  - EC2에서 Spring Cloud 프로젝트를 실행시키면 기본으로 CloudFormation 구성을 시작하기 때문에 설정한 CloudFormation이 없으면 프로젝트 실행이 되지 않는다. 때문에 해당 기능을 사용하지 않도록 false로 설정

```
Caused by: java.lang.IllegalArgumentException: No valid instance id defined
    at org.springframework.util.Assert.notNull(Assert.java:201)
    ~[spring-core-5.3.9.jar:5.3.9]
    at org.springframework.cloud.aws.core.env.stack.config.AutoDetectingStackNameProvider.autoDetectStackName(AutoDetectingStackNameProvider.java:85) ~[spring-cloud-aws-core-2.2.6.RELEASE.jar:2.2.6.RELEASE]
    at org.springframework.cloud.aws.core.env.stack.config.AutoDetectingStackNameProvider.afterPropertiesSet(AutoDetectingStackNameProvider.java:95)
```

```
kNameProvider.java:70) ~[spring-cloud-aws-core-2.2.6.RELEASE.jar:2.2.6.RELEASE]
```

- AWSS3Config.java 추가

```
package com.ssafy.coach.gi.awss3.config;

import com.amazonaws.auth.AWSStaticCredentialsProvider;
import com.amazonaws.auth.BasicAWSCredentials;
import com.amazonaws.services.s3.AmazonS3Client;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class AWSS3Config {
    @Value("${cloud.aws.credentials.access-key}")
    private String accessKey;
    @Value("${cloud.aws.credentials.secret-key}")
    private String secretKey;
    @Value("${cloud.aws.region.static}")
    private String region;

    @Bean
    public AmazonS3Client amazonS3Client() {
        BasicAWSCredentials awsCredentials = new BasicAWSCredentials(accessKey, secretKey);
        return (AmazonS3Client) AmazonS3ClientBuilder.standard()
            .withRegion(region)
            .withCredentials(new AWSStaticCredentialsProvider(awsCredentials))
            .build();
    }
}
```

- Upload Controller 추가

```
package com.ssafy.coach.gi.awss3.controller;
```

```

import com.amazonaws.services.s3.AmazonS3Client;
import com.amazonaws.services.s3.model.CannedAccessControlList;
import com.amazonaws.services.s3.model.ObjectMetadata;
import com.amazonaws.services.s3.model.PutObjectRequest;
import lombok.RequiredArgsConstructor;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.multipart.MultipartFile;

import java.io.IOException;

@RestController
@RequestMapping("/upload")
public class FileUploadController {
    @Autowired
    AmazonS3Client amazonS3Client;

    @Value("${cloud.aws.s3.bucket}")
    private String bucket;

    @Value("${cloud.aws.s3.base-url}")
    private String baseUrl;

    @PostMapping
    public ResponseEntity<String> uploadFile(@RequestParam("file") MultipartFile file) {
        try {
            String fileName = file.getOriginalFilename();
            String fileUrl = baseUrl + fileName;
            ObjectMetadata metadata = new ObjectMetadata();
            metadata.setContentType(file.getContentType());
            metadata.setContentLength(file.getSize());

            PutObjectRequest putObjectRequest = new PutObjectRequest(
                bucket, fileName, file.getInputStream(), metadata
            );
            putObjectRequest.withCannedAcl(CannedAccessControlList.PublicRead);
            amazonS3Client.putObject(putObjectRequest);
        } catch (IOException e) {
            return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).body(e.getMessage());
        }
        return ResponseEntity.ok(fileUrl);
    }
}

```

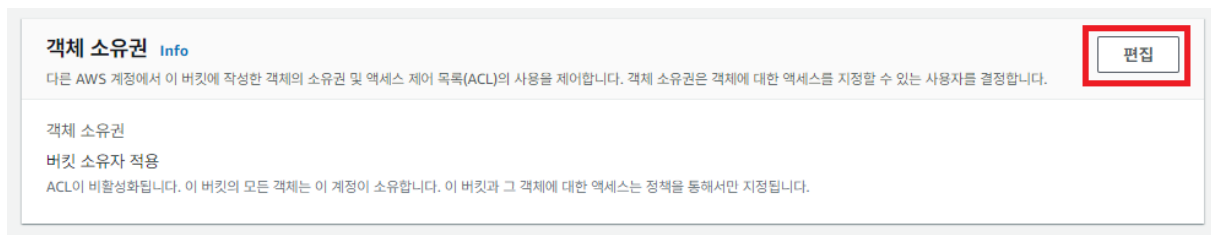
```

        return ResponseEntity.ok(fileUrl);
    } catch (IOException e) {
        e.printStackTrace();
        return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_E
    }
}
}

```

**[ERROR] The bucket does not allow ACLs (Service: Amazon S3; Status Code: 400; Error Code: AccessControlListNotSupported;**

- 버킷 - 권한 - 객체 소유권 - 편집 클릭



- **ACL 활성화됨** 체크
  - **ACL이 복원된다는 것을 확인합니다** 체크
  - **변경 사항 저장** 클릭

객체 소유권 편집 [Info](#)

## 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 검사하기가 용이합니다.



**ACL을 활성화하면 버킷 소유자가 객체 소유권에 대해 적용한 설정이 비활성화됩니다.**

버킷 소유자 적용 설정이 해제되면 ACL(액세스 제어 목록) 및 연결된 권한이 복원됩니다. 소유하지 않은 객체에 대한 액세스는 버킷 정책이 아닌 ACL을 기반으로 합니다.

☒ ACL이 복원된다는 것을 확인합니다.

## 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.



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

취소

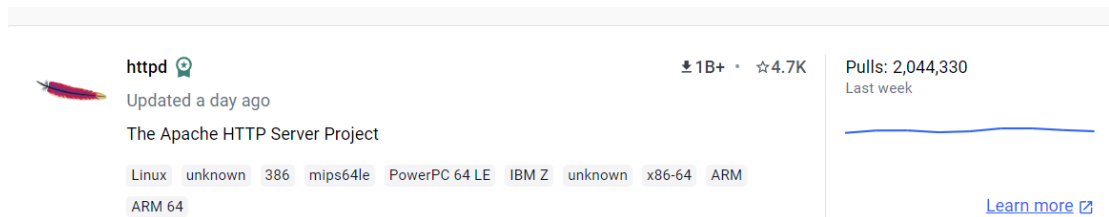
변경 사항 저장

## 도커 허브 이용법

- 도커허브 (레지스트리) 에서 필요한 소프트웨어를 찾게된다.
- 도커허브에서 찾아낸 다운로드해서 가지고 있는 것을 이미지라고 한다.
- 이미지를 실행하는 것을 컨테이너라고 한다.
- 도커 허브에서 이미지를 다운받는 행위를 pull 이라고함
- 이미지를 실행시키는 행위를 run 이라고함
- run을 하게되면 이미지가 컨테이너가 되고, 컨테이너 안에 포함되어있는 실행되도록 조치되어 있는 프로그램이 실행된다.
- 도커허브닷컴에서 이미지를 다운받자.



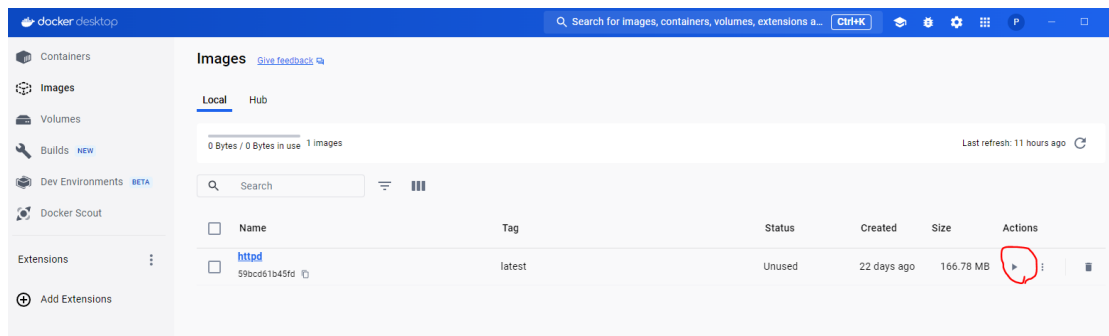
- 아파치 httpd가 있다.



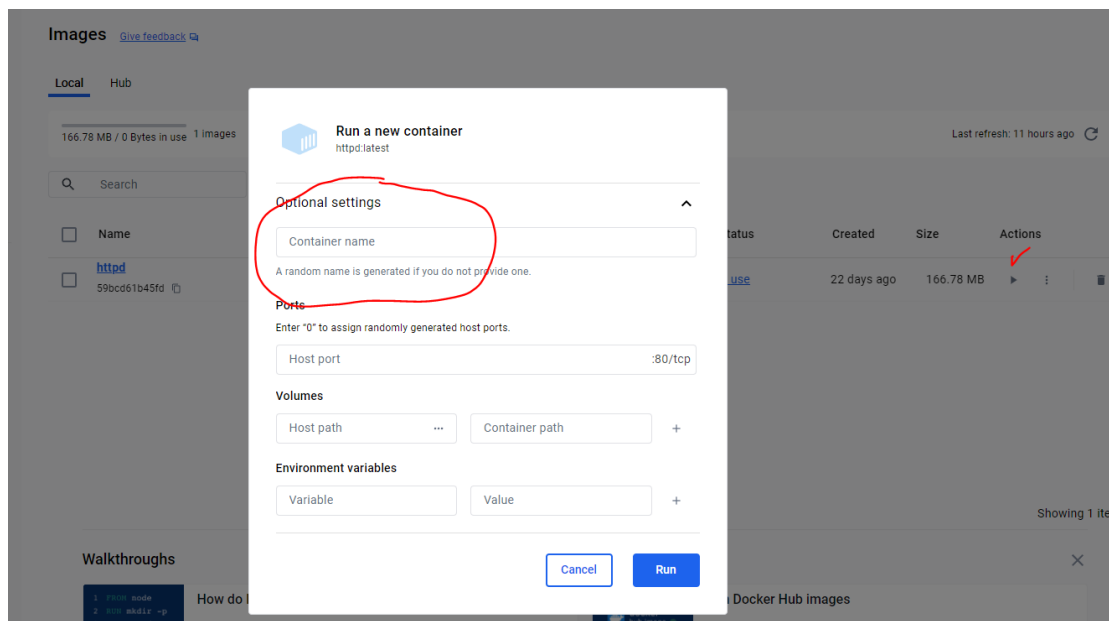
- dock pull

docker pull httpd 다운  
docker images 잘 설치되었는지 확인

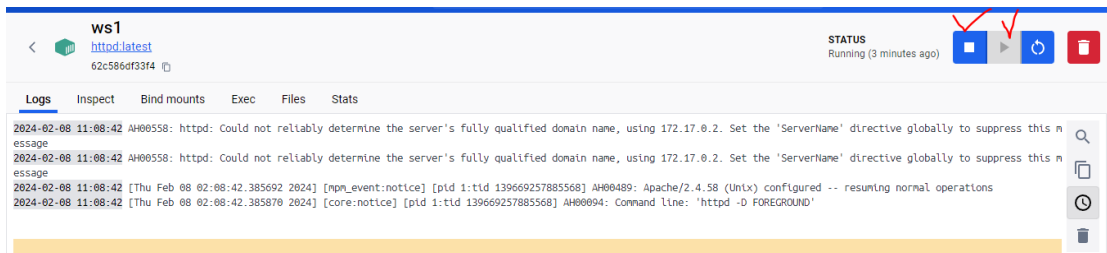
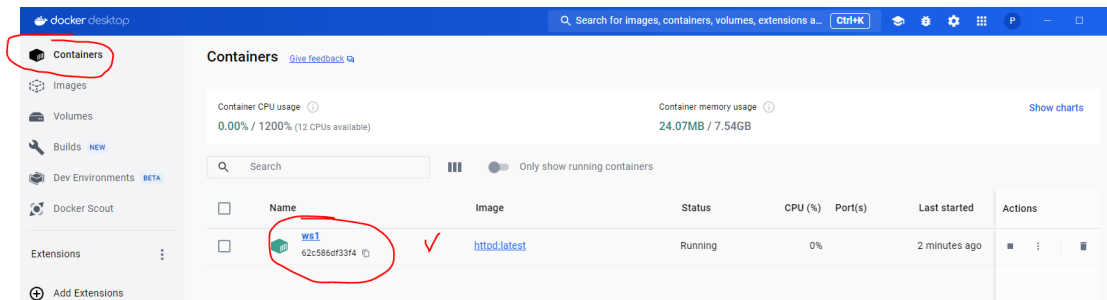
- 도커데스크탑 켜기



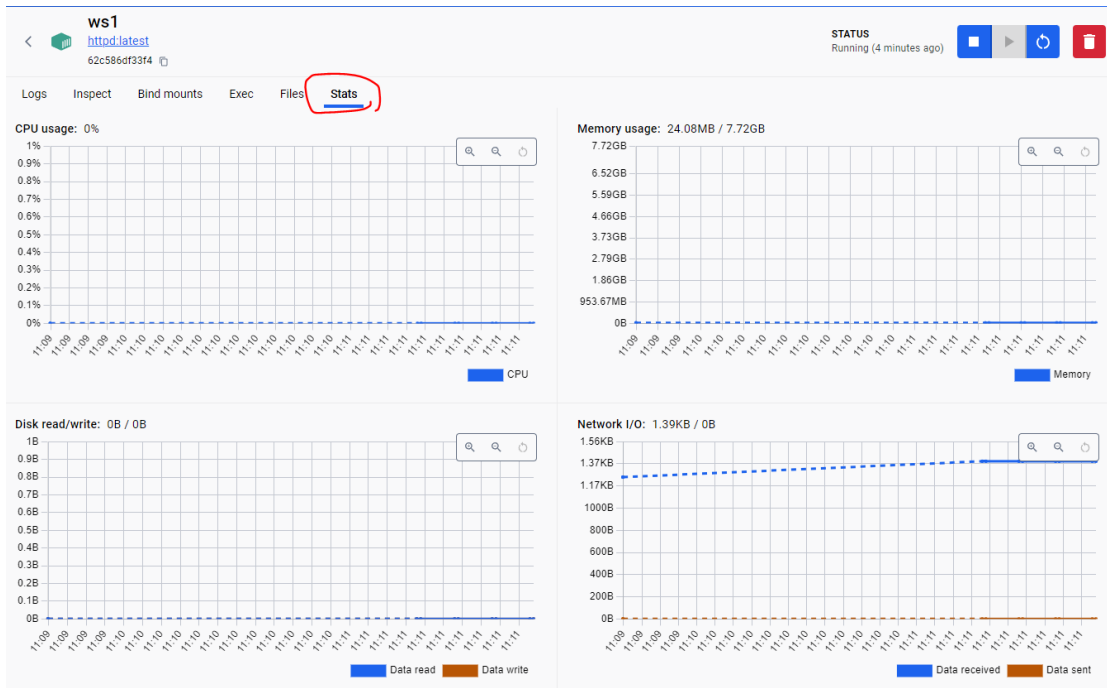
- image에서 httpd run(재생버튼) 누르면 새로운 컨테이너 만들 수 있는 창이 뜬다.



- 생성된 컨테이너 확인하고 더블 클릭하면 돌아가는 형태를 볼 수 있음



- 컴퓨터 사용 현황보려면 Stats누르기



- 도커 run CMD로 할때 코드

```
// 생성된 이미지 리스트보기
docker images
```

```
// 도커 httpd 실행
docker run httpd
```

```
// 도커 컨테이너 확인하는 명령어
docker ps
```

```

// 도커 ws2라는 컨테이너 생성
docker run --name ws2 httpd

// 실행중인 컨테이너 끝내는 명령어
docker stop ws2

// ws2 컨테이너가 꺼졌는 지 확인
docker ps
이때 실행중인 컨테이너만 보여서 ws2가 안보인다

// 실행되고 실행끝난 전체 컨테이너 보고싶을 때
docker ps -a

// 꺼진 컨테이너 실행하는 명령어
docker start ws2
이때 로그가 안보인다.

//로그 보기 위한 명령어
docker logs ws2
보여지고 꺼져버린다.

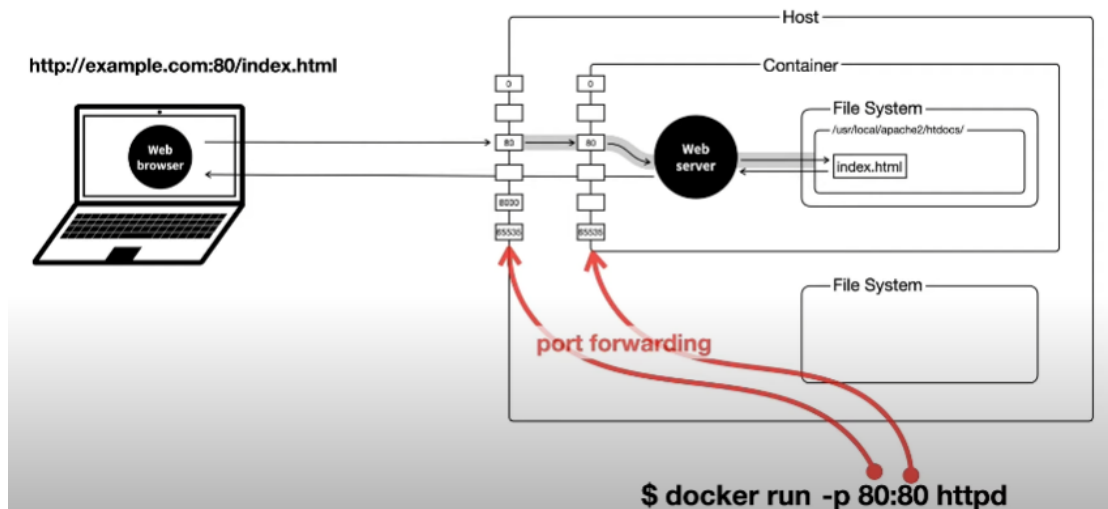
//실시간으로 로그 보기 위한 명령어
docker logs -f ws2

//컨테이너 삭제
docker rm ws2
도커 컨테이너가 stop되어있을 때 삭제가능
// 스탑과 동시에 삭제하는 명령어
docker rm --force ws2

// 이미지 삭제
docker rmi httpd

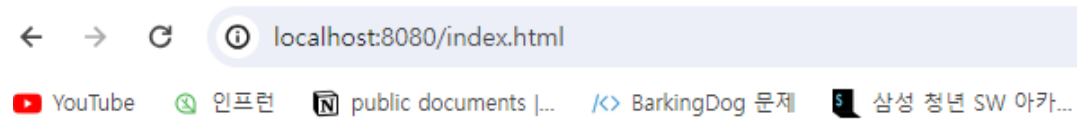
```

- 도커 네트워크를 알아보자.

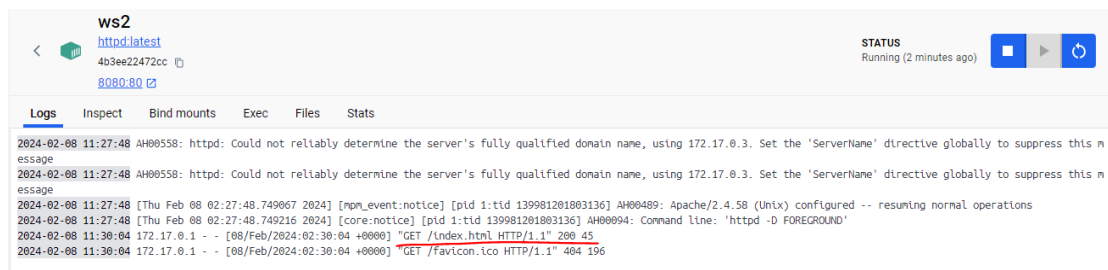


- cmd에서 진행

```
docker ps
docker run --name ws2 -p 8080:80 httpd
```

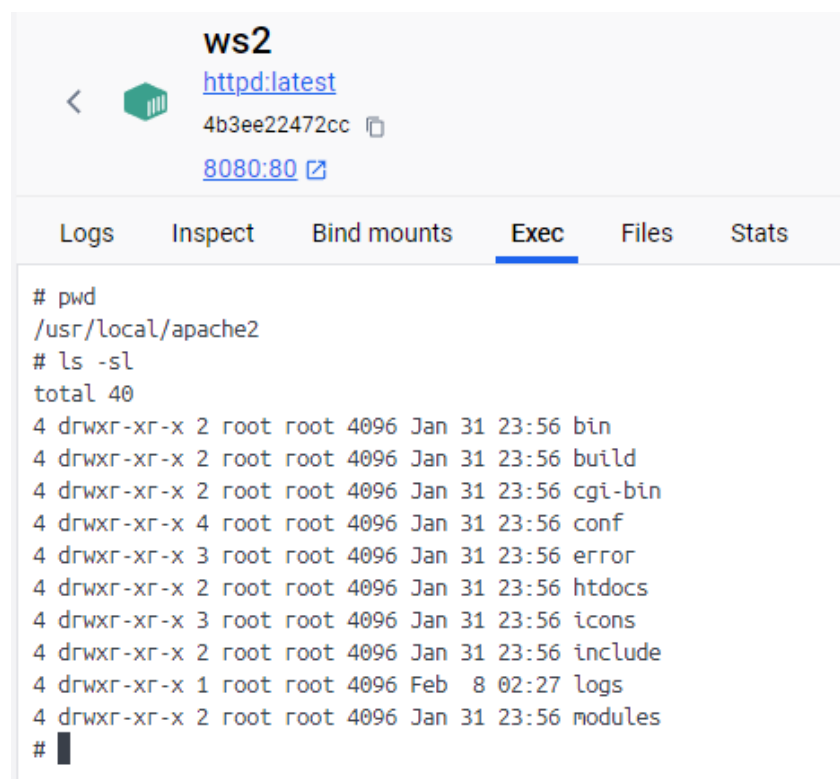
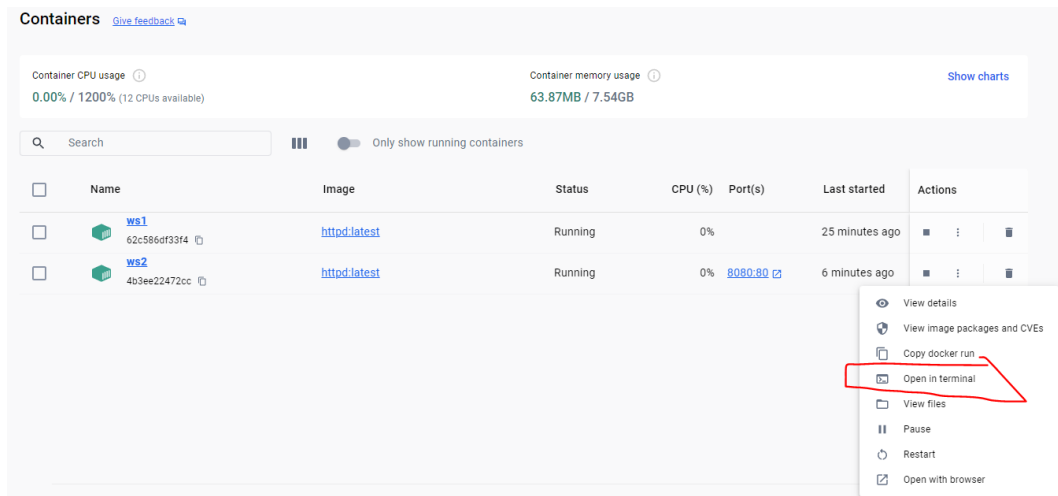


It works!



- 도커 명령어 실행

- index.html을 수정해보자



#### ■ cmd로 실행하는 법

```
docker exec ws3 pwd
docker exec ws3 ls -sl
```

이렇게 명령어를 쓴다.

컨테이너가 지속적으로 연결하여, 명령어 전달하고싶을 때

```
docker exec -it ws3 /bin/sh
```

i는 interactive / t는 tty

본체는 기능이 부족할 땐, sh->bash를 쓰면 되지만, 이미지가 제공안하는 것도

나갈 땐,  
exit

index 수정해보자

```
docker exec -it ws3 /bin/bash  
cd /usr/local/apache2/htdocs/
```

```
C:\Users\SSAFY>docker exec -it ws2 /bin/bash  
root@4b3ee22472cc:/usr/local/apache2# cd /usr/local/apache2/htdocs  
bash: cd: /usr/local/apache2/htdocs/: No such file or directory  
root@4b3ee22472cc:/usr/local/apache2# cd /usr/local/apache2/htdocs/  
root@4b3ee22472cc:/usr/local/apache2/htdocs# ls -al  
total 16  
drwxr-xr-x 2 root    root    4096 Jan 31 23:56 .  
drwxr-xr-x 1 www-data www-data 4096 Jan 31 23:56 ..  
-rw-r--r-- 1      504 staff   45 Jun 11  2007 index.html  
root@4b3ee22472cc:/usr/local/apache2/htdocs# nano index.html  
bash: nano: command not found  
root@4b3ee22472cc:/usr/local/apache2/htdocs# apt update
```

```
root@4b3ee22472cc:/usr/local/apache2/htdocs# apt install nano
```

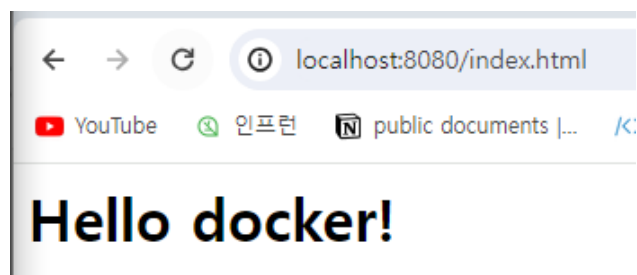
```
Processing triggers for libc-bin (2.30-3-deb12u4) ...  
root@4b3ee22472cc:/usr/local/apache2/htdocs# nano index.html  
root@4b3ee22472cc:/usr/local/apache2/htdocs#
```

관리자: 명령 프롬프트 - docker exec -it ws2 /bin/bash

GNU nano 7.2

```
<html><body><h1>Hello docker!</h1></body></html>
```

수정하고 ctrl+X → y → enter 저장



- 호스트와 컨테이너의 파일시스템 연결

```
<> index.html U x
htdocs > <> index.html > html
<html>
  <body>
    Hello,Docker!
  </body>
</html>

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE 1: zsh
→ Desktop docker run -p 8888:80 -v ~/Desktop/htdocs:/usr/local/apache2/htdocs/ httpd
```