

포팅 매뉴얼

1. 기술 스택

구분	기술 스택	상세 내용	버전
공통	형상관리	GitLab	-
	이슈관리	Jira	-
	커뮤니케이션	Mattermost	7.8.6
		Notion	2.3.2
		discord	1.0.9023
		kkakaotalk	-
	기타 편의 툴	Postman	10.20.0
		Swagger	2.1.0
		MobaXterm	23.2
		Dbeaver	23.2.2
	UI/UX	Figma	-
	OS	Android	-
Server	서버	Ubuntu	20.04 LTS
		Windows	10
	플랫폼	Docker	24.0.6
	배포	Docker Desktop	4.21.1
		Jenkins	2.414.3
Back-End	DB	MySQL	8.1.0
		MongoDB	7.0.2
		Redis	7.2.2
		JPA	-
	Java	OpenJDK	17.0.8
		Spring Boot	3.0.8
		Spring 내장 tomcat	3.0.8

	Build	Gradle	8.3
	IDE	IntelliJ IDEA	2023.2.2
AI	전처리	Python	3.7.13
		opencv-python	4.5.5.62
		CUDA	11.7
		tensorflow	1.15.0
		tensorflow-estimator	1.15.1
		tensorflow-gpu	1.15.0
		torch	1.13.1
	VITON-HD	Python	3.8.13
		opencv-python	4.8.1.78
		Tensorflow	2.13.1
		pytorch	1.9.0+cu111
	Openpose	Python	3.7.16
		CUDA	11.5
	IDE	PyCharm	2023.2.3
Front-End	Flutter	Flutter	3.16.0-15.0.pre.36
		dart	3.3.0
	Package	upertino_icons	1.0.2
		animated_splash_screen	1.3.0
		image_picker	1.0.4
		http	0.13.4
		dio	5.3.3
		flutter_secure_storage	9.0.0
		provider	6.0.5
		convert	3.1.1
		http_parser	4.0.2
		camera	0.10.5+5
		path	1.8.0
	IDE	Visual Studio Code	1.84.0

2. 배포 매뉴얼

1) Docker 설치 와 DB 설치 및 연결

1-1 Docker 설치

```
# Docker 설치 전 필요한 패키지 설치
sudo apt-get -y install apt-transport-https ca-certificates curl gnupg-agent software-properties-common

# Docker에 대한 GPG Key 인증 진행
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add

# Docker 레포지토리 등록
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"

# Docker 패키지 설치
sudo apt-get -y update
sudo apt-get -y install docker-ce docker-ce-cli containerd.io

# Docker 일반 유저에게 권한 부여
sudo usermod -aG docker Ubuntu

# Docker 서비스 재시작
sudo service docker restart
exit
```

1-2 MySQL

```
# MySQL Docker 이미지 다운로드
docker pull mysql

# MySQL Docker 컨테이너 생성 및 실행
docker run --name mysql -e MYSQL_ROOT_PASSWORD=<password> -d -p 3306:3306 mysql:latest
```

1-3 Redis

```
# Redis Docker 이미지 다운로드
docker pull redis:latest

# Redis Docker 컨테이너 생성 및 실행
docker run -d -p 6379:6379 --name=redis redis:latest
```

1-4 MongoDB

```
# MongoDB Docker 이미지 다운로드
docker pull mongo

# MongoDB Docker 컨테이너 생성 및 실행
docker run --name mongo -v ~/data:/data/db -d -p 27017:27017 mongo
```

1-5 spring boot의 설정

- application.yml

```
spring:
  autoconfigure:
    exclude: org.springframework.boot.autoconfigure.security.servlet.SecurityAutoConfiguration
  jpa:
    show-sql: true
    properties:
      hibernate:
        format_sql: true
        dialect: org.hibernate.dialect.MySQL8Dialect
    hibernate:
      ddl-auto: update

  datasource:
    url: jdbc:mysql://${DB_HOST}:${MYSQL_PORT}/kkalong
    driver-class-name: com.mysql.cj.jdbc.Driver
    username: ${MYSQL_USERNAME}
    password: ${MYSQL_PASSWORD}
```

```
data:
  mongodb:
    uri:
mongodb://${MONGODB_USERNAME}:${MONGODB_PASSWORD}@${DB_HOST}:${MONGODB_P
ORT}/kksalong

  redis:
    host: ${DB_HOST}
    port: ${REDIS_PORT}
    password: ${REDIS_PASSWORD}
```

- build.gradle

```
dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
    implementation 'org.springframework.boot:spring-boot-starter-web'
    implementation 'org.springframework.boot:spring-boot-starter-validation'
    compileOnly 'org.projectlombok:lombok'

    implementation 'org.springframework.boot:spring-boot-starter-data-redis'
    implementation 'org.springframework.boot:spring-boot-starter-data-mongodb'

    runtimeOnly 'com.mysql:mysql-connector-j'
    annotationProcessor 'org.projectlombok:lombok'
    testImplementation 'org.springframework.boot:spring-boot-starter-test'
    implementation group: 'org.springdoc', name: 'springdoc-openapi-starter-webmvc-ui',
version: '2.1.0'
}
```

2) Nginx 설치와 SSL 인증서 발급 및 적용

2-1 Nginx 설치

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

2-2 SSL 인증서 발급

- ZeroSSL 에서 회원가입 후 New Certificate 를 선택 (<https://zerossl.com>)
- 도메인에 서버 IP 를 입력
- 무료인 90 일 인증서를 선택
- Auto-Generate CSR 선택
- 무료 플랜 선택
- HTTP File Upload 선택
- Download Auth File 파일 다운로드
- 서버에 /.well-known/pki-validation 폴더를 만든다.
- 위에서 만든 pki-validation 폴더 안에 Auth file 을 저장한다
- nginx 설정 파일(/etc/nginx/sites-available/default)을 수정한다.

```
server {  
    listen 80 default_server;  
    listen [::]:80 default_server;  
  
    root /var/www/html;  
  
    index index.html index.htm index.nginx-debian.html;  
  
    server_name _;  
  
    location / {  
        try_files $uri $uri/ =404;  
    }  
}  
  
#추가한 부분  
server {  
    listen 443 ssl;  
  
    ssl on;  
    ssl_certificate /etc/nginx/ssl/combined_certificate.crt;  
    ssl_certificate_key /etc/nginx/ssl/private.key;  
  
    server_name k9c105.p.ssafy.io;  
    access_log /var/log/nginx/nginx.vhost.access.log;  
    error_log /var/log/nginx/nginx.vhost.error.log;
```

```
location / {  
    root /var/www/html;  
    index index index.html index.htm index.nginx-debian.html;  
}  
}
```

- nginx 재시작

```
#nginx 실행 재시작  
sudo systemctl restart nginx  
  
# nginx 상태 확인  
sudo systemctl status nginx
```

3) Jenkins 설치

3-1 Jenkins 설치

```
sudo apt update  
sudo apt install jenkins
```

3-2 Jenkins Pipeline script

```
pipeline {  
    agent any // 사용 가능한 에이전트에서 이 파이프라인 또는 해당 단계를 실행  
  
    environment {  
        DOCKER_IMAGE_NAME_BE = 'leeseungtae/c105-backend' // 도커 허브 레포지토리  
        DOCKER_IMAGE_BE = "  
    }  
  
    stages {  
        stage('Prepare') {  
            steps {  
                sh 'echo "Clonning Repository"  
                git branch: 'develop',  
                credentialsId: 'GitLab',
```

```

        url: 'https://lab.ssafy.com/s09-final/S09P31C105.git'
    }
    post {
        success {
            sh 'echo "Successfully Cloned Repository"'
        }
        failure {
            sh 'echo "Fail Cloned Repository"'
        }
    }
}

stage('Build Gradle') {
    steps {
        sh 'echo "Bulid Gradle Start"'
        dir ('./backend/spring/kkalong') {
            // gradlew이 있어야됨. git clone해서 project를 가져옴.
            sh 'chmod +x gradlew'
            sh './gradlew clean build --exclude-task test'
        }
    }
    post {
        success {
            echo 'gradle build success'
        }
        failure {
            echo 'gradle build failed'
        }
    }
}

stage('Bulid Docker Backend') {
    steps {
        sh 'echo " Image Bulid Start"'
        dir ('./backend/spring/kkalong') {
            script {
                DOCKER_IMAGE_BE = docker.build DOCKER_IMAGE_NAME_BE
            }
        }
    }
    post {

```



```

        failure {
            sh 'echo "Bulid Docker Fail"'
        }
    }
}

stage('Push Docker') {
    steps {
        sh 'echo "Docker Image Push Start"'
        script {
            docker.withRegistry('https://registry.hub.docker.com', "docker-hub") {
                DOCKER_IMAGE_BE.push("latest")
            }
        }
    }
    post {
        success {
            sh 'docker stop spring'
            sh 'docker rmi -f $(docker images -q -f dangling=true)'
        }
        failure {
            error 'This Image Push Fail'
        }
    }
}

stage('Remote Server Docker Pull') {
    steps {
        echo "Pulling Docker Image on Remote Server"
        sh 'docker rmi -f leeseungtae/c105-backend || true'
        sh 'docker stop spring || true'
        sh 'docker rm -f spring || true'
        sh '''
            docker run -i -e TZ=Asia/Seoul ₩
                -e SPRING_PROFILES_ACTIVE=prod ₩
                -e DB_HOST=DB의 호스트₩
                -e MYSQL_PORT= MySQL의 포트₩
                -e MYSQL_USERNAME= MySQL의 계정 이름₩
                -e MYSQL_PASSWORD= MySQL의 계정 비밀번호₩
                -e MONGODB_USERNAME=mongoDB의 계정 이름 ₩
                -e MONGODB_PASSWORD=mongoDB의 비밀 번호₩
        '''
    }
}

```

```

-e MONGODB_PORT=mongoDB의 포트 번호 ₩
-e REDIS_PORT=redis의 포트 번호 ₩
-e REDIS_PASSWORD=redis의 비밀번호 ₩
-e SECURITY_NAME= security의 계정 이름₩
-e SECURITY_PASSWORD= security의 계정 비밀번호₩
-e JWT_SECRET_KEY=JWT 비밀번호 키 ₩
-e JWT_EXPIRATION_MINUTES= JWT 만료 분₩
-e JWT_REFRESH_EXPIRATION_HOURS= JWT 만료 시간 ₩
-e JWT_ISSUER=JWT의 issuer ₩
-e S3_ACCESS_KEY= S3서버 접속 키 ₩
-e S3_SECRET_KEY=S3서버 비밀번호 ₩
-e AI_SERVER_IP=AI서버 IP₩
-e AI_SERVER_PREPROCESS_PORT=AI 서버 포트 ₩
-e AI_SERVER_VITON_PORT= AI VITON 서버 포트 ₩
-e AI_SERVER_OPENPOSE_ADDR= AI OPENPOSE 서버 IP ₩
--name            spring            -p            8761:8761            -d
registry.hub.docker.com/leeseungtae/c105-backend:latest
'''

}
post {
    success {
        echo "Completed Remote Server Docker Pull"
    }
    failure {
        echo "Failed Remote Server Docker Pull"
    }
}

}

}

}

```

3-2 Spring Dockerfile

```
# jdk 이미지 불러오기
FROM openjdk:17-jdk-slim

# Docker Container에서 작업이 이루어지는 위치
WORKDIR /root

ARG JAR_FILE=build/libs/*.jar

# 현재 경로/target/be-0.0.1-SNAPSHOT.jar에 해당하는 파일을
# Docker Container의 WORKDIR 위치로 복사 (이미지 생성할 때 동작)
COPY ${JAR_FILE} app.jar

# COPY된 jar파일 실행하기 (컨테이너 실행할 때 동작)
CMD ["java", "-jar", "./app.jar"]
```

4) S3 환경 구축

- 참고링크: <https://inpa.tistory.com/entry/AWS-%F0%9F%93%9A-S3-%EB%B2%84%ED%82%B7-%EC%83%9D%EC%84%B1-%EC%82%AC%EC%9A%A9%EB%B2%95-%EC%8B%A4%EC%A0%84-%EA%B5%AC%EC%B6%95>
- IAM 참고 링크: <https://velog.io/@chrkb1569/AWS-S3-%EC%A0%81%EC%9A%A9%ED%95%98%EA%B8%B0-IAM-%ED%94%84%EB%A1%9C%EC%A0%9D%ED%8A%B8-%EC%84%A4%EC%A0%95>

4-1 회원 가입

- 해외 결제가 가능한 카드가 필요하다.
- 과정은 하단의 링크를 참고할 것
 - <https://goddaehee.tistory.com/315>

4-2 S3 Bucket 만들기

- 1) <https://s3.console.aws.amazon.com/s3/buckets> 로 들어간다

2) "버킷 만들기"를 누른다.

3) 버킷 이름을 정해주고, 리전을 "아시아 태평양(서울) ap-northeast-2"를 선택한다. 객체 소유권은 비활성화를 선택한다.

Amazon S3 > 버킷 > 버킷 만들기

버킷 만들기

정보

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

일반 구성

버킷 이름

grow-test-bucket

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

AWS 리전

아시아 태평양(서울) ap-northeast-2

기존 버킷에서 설정 복사 - 선택 사항

다음 구성의 버킷 설정만 복사됩니다.

버킷 선택

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

4) "이 버킷의 퍼블릭 액세스 차단 설정"은 이하와 같이 설정한다. 아래의 "현재 설정으로 인해~~"에도 체크해준다.

이 버킷의 퍼블릭 액세스 차단 설정

퍼블릭 액세스는 ACL(엑세스 제어 목록), 버킷 정책, 액세스 지점 정책 또는 모두를 통해 버킷 및 객체에 부여됩니다. 이 버킷 및 해당 객체에 대한 퍼블릭 액세스가 차단되었는지 확인하려면 모든 퍼블릭 액세스 차단을 활성화합니다. 이 설정은 이 버킷 및 해당 액세스 지점에만 적용됩니다. AWS에서는 모든 퍼블릭 액세스 차단을 활성화하도록 권장하지만, 이 설정을 적용하기 전에 퍼블릭 액세스가 없어도 애플리케이션이 올바르게 작동하는지 확인합니다. 이 버킷 또는 내부 객체에 대한 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(엑세스 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(엑세스 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 있음을 알고 있습니다.

5) 나머지 설정은 기본으로 하여 만들기를 하면 끝

4-3. IAM

- IAM 이란 AWS 서버에 액세스 키로 접근할 수 있는 사용자이다.

- AWS 서버에 직접 접근하는 것은 위험할 수 있으므로, 한정된 권한을 부여한 IAM 을 통해 접근하는 것이 권장된다.

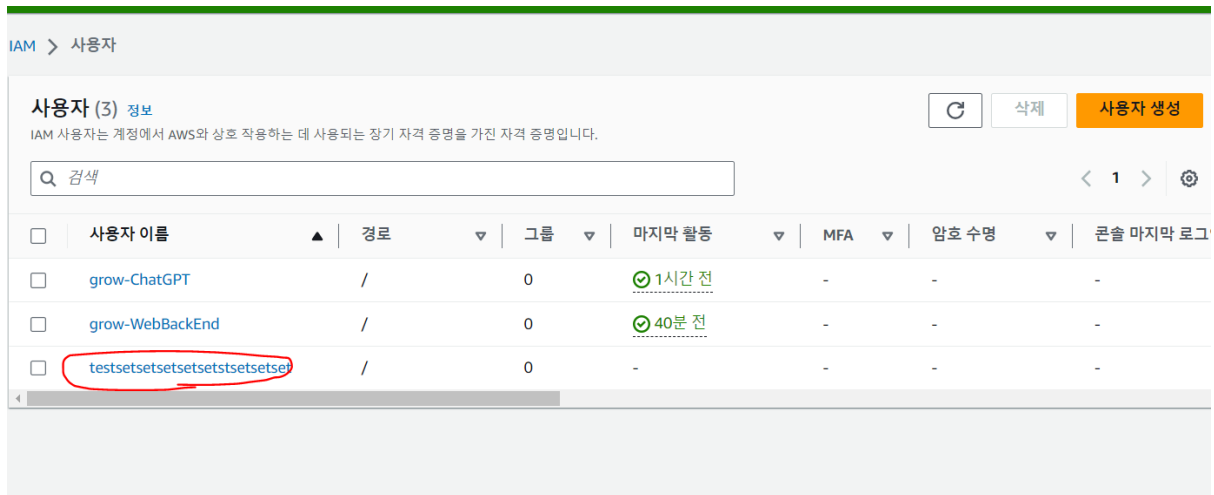
4.3.1. 사용자 생성

1) <https://us-east-1.console.aws.amazon.com/iamv2/home#/home> 로 들어간다.

2) 사용자 밑의 숫자를 누른다.

4.3.2. 액세스 키 생성

1) 앞에서 만든 사용자를 누른다.

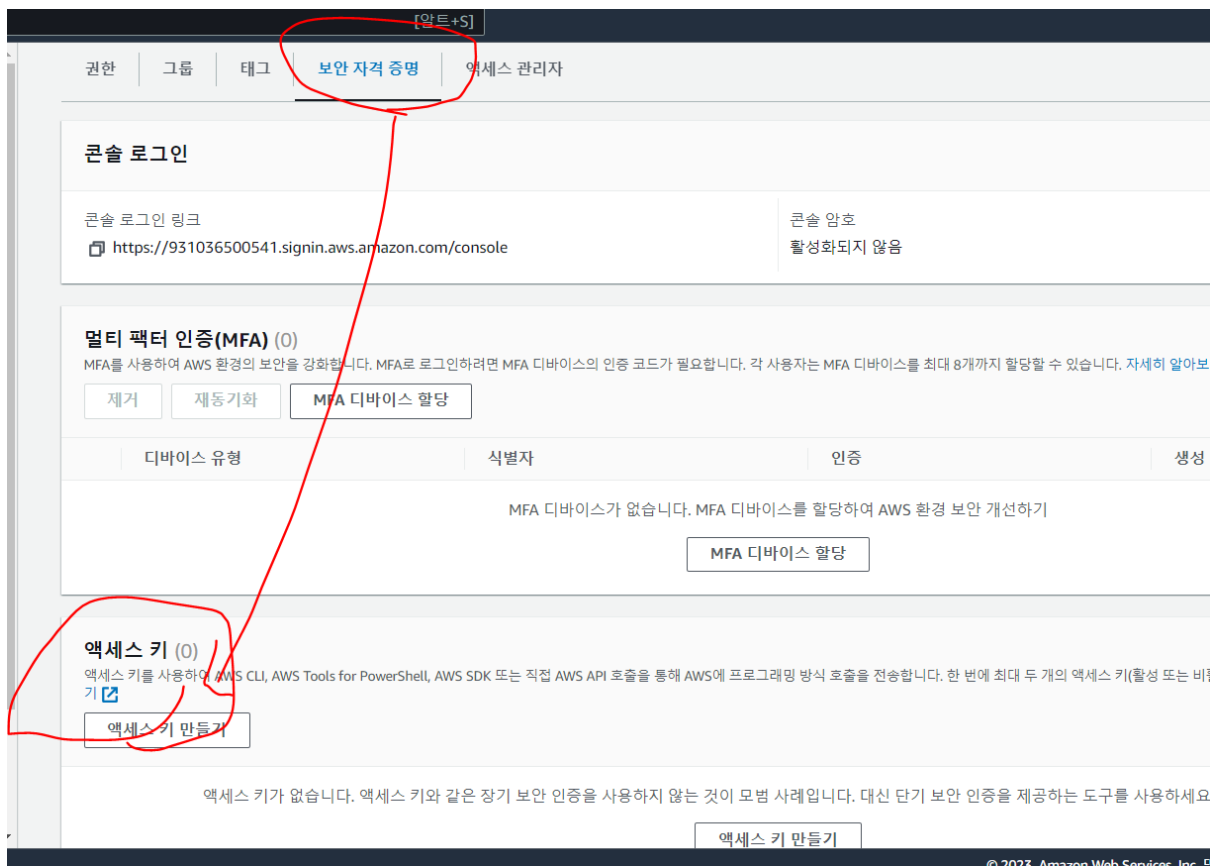


The screenshot shows the AWS IAM console 'Users' page. At the top, there's a search bar and buttons for 'Refresh', 'Delete', and 'Create User'. Below is a table of users:

<input type="checkbox"/>	사용자 이름	경로	그룹	마지막 활동	MFA	암호 수명	콘솔 마지막 로그인
<input type="checkbox"/>	grow-ChatGPT	/	0	1시간 전	-	-	-
<input type="checkbox"/>	grow-WebBackEnd	/	0	40분 전	-	-	-
<input type="checkbox"/>	testsetsetsetsetsetset	/	0	-	-	-	-

The user 'testsetsetsetsetsetset' is circled in red.

2) 보안 자격증명 -> 액세스 키 만들기를 누른다.



The screenshot shows the AWS IAM console 'Security Credentials' page for a user. The '보안 자격 증명' (Security Credentials) tab is selected and circled in red. Below it, there's a section for 'Console Login' and a section for 'Multi-Factor Authentication (MFA)'. The 'Access Keys' section is circled in red, showing 'Access Keys (0)' and a 'Create Access Key' button. A red arrow points from the '보안 자격 증명' tab to the 'Create Access Key' button.

Access Keys (0)

Access keys are used to access AWS CLI, AWS Tools for PowerShell, AWS SDK 또는 직접 AWS API 호출을 통해 AWS에 프로그래밍 방식 호출을 전송합니다. 한 번에 최대 두 개의 액세스 키(활성 또는 비활성)를 생성할 수 있습니다.

[Access Key 만들기](#)

3) 액세스 키 모범 사례 및 대안에서 사례를 선택한다. AWS 상에 서버를 구축했다면 "AWS 컴퓨팅 서비스에서 실행되는 애플리케이션"을 선택한다.

액세스 키 모범 사례 및 대안

정보

보안 개선을 위해 액세스 키와 같은 장기 자격 증명을 사용하지 마세요. 다음과 같은 사용 사례와 대안을 고려하세요.

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.

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

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

☐ 기타
귀하의 사용 사례가 여기에 나열되어 있지 않습니다.

취소

다음

4) 선택 후 "위의 권장 사항을 이해했으며 액세스 키 생성을 계속하려고 합니다"에 체크하고 다음을 누른다.


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.

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

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

☐ 기타
귀하의 사용 사례가 여기에 나열되어 있지 않습니다.

 권장되는 대안

EC2 인스턴스 또는 Lambda 함수와 같은 컴퓨팅 리소스에 IAM 역할을 할당하여 액세스를 위한 임시 보안 인증을 자동으로 제공합니다. [자세히 알아보기](#)

확인

☒ 위의 권장 사항을 이해했으며 액세스 키 생성을 계속하려고 합니다.

취소

다음

5) 액세스키의 설명을 지어준 뒤 "엑세스 키 만들기"를 누른다. 한글은 쓸 수 없다.

IAM > 사용자 > testsetsetsetsetsetset > 액세스 키 만들기

1단계
엑세스 키 모범 사례 및 대안

2단계 - 선택 사항
설명 태그 설정

3단계
엑세스 키 검색

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](#) [엑세스 키 만들기](#)

6) 액세스 키와 비밀 액세스 키를 저장해둔다. 액세스 키는 여기서만 조회가 가능하고, 이후 어떠한 방법으로도 다시 조회할 수 없으므로, 키를 잊어버렸다면 IAM 사용자를 다시 만들어야 하므로 주의.

1단계
엑세스 키 모범 사례 및 대안

2단계 - 선택 사항
설명 태그 설정

3단계
엑세스 키 검색

엑세스 키 검색 정보

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

엑세스 키	비밀 액세스 키
AKI	***** 표시

엑세스 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.

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

[.csv 파일 다운로드](#) [완료](#)

7) 6 에서 얻은 두 액세스 키를 application.yml 과 .env 에 넣는다

4.3.3. 버킷 정책 만들기

1) <http://awspolicygen.s3.amazonaws.com/policygen.html> 에 들어간다.

2) 표시된 부분을 입력한다. principal 은 "*", Actions 는 "GetObject"와 PutObject 두개를 고르고, ARN 은 해당 버킷 페이지 -> 속성에서 확인 가능한 값 뒤에 "/\W*"를 붙인다. (예: arn:aws:s3:::grow-test-bucket -> arn:aws:s3:::grow-test-bucket/\W*)

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 Policy](#), an [SNS VPC Endpoint Policy](#), and an [SQS Queue Policy](#).

Select Type of Policy S3 Bucket 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 statements.

Effect ☒ Allow ☐ Deny

Principal

Use a comma to separate multiple values.

AWS Service Amazon S3 ☐ All Services ('*')

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

Actions 2 Action(s) Selected ☐ 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 \(Optional\)](#)

[Add Statement](#)

GetObject
PutObject 선택

3) Add Statement 를 누른 뒤, 하단의 Generate Policy 를 누른다.

4) 새로 뜬 창의 내용을 복사해둔다.

5) 방금 만든 버킷으로 들어가 권한 탭-> 버킷 정책->편집을 누른다.

grow-test-bucket 정보

객체 | 속성 | **권한** | 지표 | 관리 | 액세스 지정

권한 개요

액세스
객체를 퍼블릭으로 설정할 수 있음

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](#)

[편집](#)

모든 퍼블릭 액세스 차단

비활성

▶ 이 버킷의 개별 퍼블릭 액세스 차단 설정

버킷 정책

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

[편집](#) [삭제](#)

6) 방금 복사한 내용을 붙여넣기 하고 변경사항 저장을 누른다.

7) 버킷 페이지에서 권한탭->"퍼블릭 액세스 차단(버킷 설정)"->편집을 누른다.

8) 아래 사진같이 설정하고 변경사항 저장을 한다.

Amazon S3 > 버킷 > grow-test-bucket > 퍼블릭 액세스 차단 편집(버킷 설정)

퍼블릭 액세스 차단 편집(버킷 설정) 정보

퍼블릭 액세스 차단(버킷 설정)

퍼블릭 액세스는 ACL(엑세스 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 지점에만 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(엑세스 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(엑세스 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

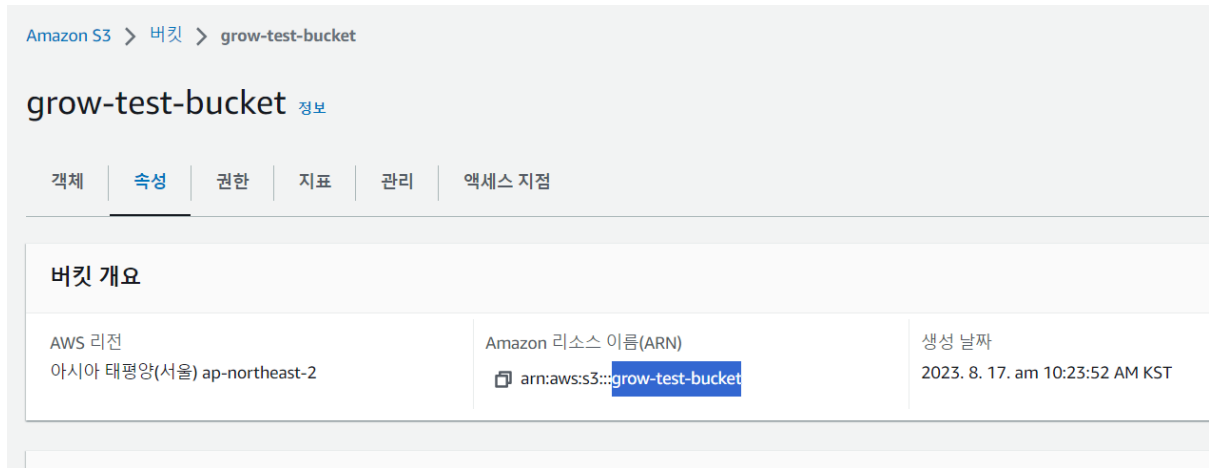
4.3.4. 환경 변수

- AWS_REGION = 버킷을 만들때 정한 지역. 메뉴얼대로 했다면 "ap-northeast-2"이다.

- AWS_ACCESS_KEY = 위에서 얻은 두 액세스 키 중에서 "엑세스 키"의 문자열

- AWS_SECRET_KEY = 위에서 얻은 두 액세스 키 중에서 "비밀 액세스 키"의 문자열

- AWS_BUCKET = 버킷을 만들 때 지어준 이름. 버킷에 들어가서 "속성"탭에서도 확인 할 수 있다.



5) 플러터 앱을 APK 설치 파일로 변환

1. 프로젝트 열기

안드로이드 스튜디오를 열고, Flutter 프로젝트를 엽니다.

2. 터미널 열기

안드로이드 스튜디오 내에서 터미널을 엽니다. 이는 보통 하단의 탭 중 하나로 찾을 수 있습니다.

3. 빌드 모드 선택

APK를 추출하기 위해 다음 중 하나의 명령어를 터미널에 입력합니다.

릴리스 모드: flutter build apk (가장 일반적으로 사용)

릴리스 모드 (분할 APK): flutter build apk --split-per-abi (APK 크기를 줄이기 위해 여러 ABI별로 분할)

디버그 모드: flutter build apk --debug

프로파일 모드: flutter build apk --profile

4. 빌드 과정

명령어를 실행하면, Flutter는 APK를 빌드하기 시작합니다. 이 과정은 몇 분 정도 걸릴 수 있습니다.

5. APK 위치 확인

빌드가 완료되면, 터미널에 빌드된 APK 파일의 경로가 출력됩니다. 일반적으로 이 경로는 `build/app/outputs/flutter-apk/app-release.apk` (릴리스 모드)입니다.

6. APK 파일 찾기

파일 탐색기를 사용하여 해당 경로로 이동하고, 빌드된 APK 파일을 찾습니다.

7. APK 파일 사용

APK 파일을 원하는 장치에 설치하거나 배포합니다.

<추가 사항>

특정 플랫폼에 맞는 APK를 빌드하려면, Flutter 프로젝트의 `pubspec.yaml` 파일에서 해당 플랫폼의 설정을 확인하고 필요에 따라 수정해야 할 수 있습니다.

APK 파일의 크기를 줄이기 위해, ProGuard를 사용하여 코드를 난독화하고 불필요한 코드를 제거하는 것을 고려할 수 있습니다

[프론트엔드 빌드 명령어]

flutter pub get: 프로젝트의 모든 의존성을 설치합니다.

Android용 빌드 **flutter build apk:** Android APK를 릴리스 모드로 빌드합니다.

flutter run android: Android 에뮬레이터나 연결된 디바이스에서 앱을 직접 실행합니다.

웹용 빌드 **flutter build web:** 웹 애플리케이션을 릴리스 모드로 빌드합니다.

flutter run -d chrome: 크롬 브라우저에서 앱을 직접 실행합니다.

데스크톱용 빌드 (Windows, macOS, Linux) **flutter build windows:** Windows용 애플리케이션을 빌드합니다.

flutter build macos: macOS용 애플리케이션을 빌드합니다.

flutter build linux: Linux용 애플리케이션을 빌드합니다.

각 플랫폼에 맞는 flutter run 명령어로 데스크톱 앱을 직접 실행할 수 있습니다.

flutter clean: 빌드 디렉토리를 정리하여 오래된 빌드 파일을 제거합니다.

flutter doctor: Flutter 설치와 관련된 문제를 진단합니다.