

Storage System

Backup

❖ AWS를 이용한 백업

✓ 백업의 형태

- 온프레미스 환경의 데이터를 AWS로 백업
- AWS에 구축한 시스템을 백업하는 경우

✓ 백업의 개요

- 사내 시스템은 온프레미스 환경에 기업 사이트는 AWS 환경에 위치
- 백업 대상은 용량이 큰 이미지 파일, 데이터베이스, 각 서버의 로그 파일, 파일 서버에 있는 파일
- 로그 파일과 데이터베이스 파일은 장기간 보관
- 비용을 절감하면서도 운영에 손이 많이 가지 않는 방식으로 진행

✓ 인프라 핵심 설계 사항

- 스토리지 게이트웨이를 이용한 자동 백업: 온프레미스 환경에 스토리지 게이트웨이를 사용하여 백업용 스토리지를 만들어 S3에 자동 백업
- S3과 글레이셔로 수명 주기 관리: 로그 파일을 S3의 기능으로 백업하면서 온라인 보관 기간을 넘은 파일을 글레이셔에 아카이브
- 용량의 대부분을 차지하는 이미지 파일과 데이터베이스는 S3에 백업: 스토리지 게이트웨이에 부과되는 종량제 요금을 줄이기 위해 용량이 큰 데이터는 스크립트를 사용하여 S3에 백업

Backup

❖ 온프레미스 환경의 데이터 백업

- ✓ 온프레미스 환경의 데이터를 AWS에 백업하는 경우는 백업 대상이나 예산에 따라 적절한 설계 패턴이 달라짐
- ✓ 일반적인 방법
 - 오브젝트 스토리지 서비스인 S3(Amazon Simple Storage Service)를 명령행 인터페이스를 통해 이용하는 패턴
 - ◆ 백업 대상이 파일인 경우에 유효하며 간편하면서도 저렴
 - ◆ S3는 범용적인 파일 저장 서비스로서 온라인 파일 서버처럼 사용할 수 있음
 - ◆ S3는 서로 다른 가용 영역(AZ)에 3중화되어 있으며 99.999999999% 라고 하는 높은 내구성
 - 가상 서버인 EC2와 가상 스토리지인 EBS로 구축한 백업용 서버에 데이터를 동기화시키는 패턴
 - ◆ 첫 번째 방법보다는 비용이 들지만 데이터베이스의 백업을 유연하게 실행할 수 있음
 - ◆ 백업 대상 EC2에서 유틸리티를 실행시키면 DB를 온라인 백업할 수 있음
 - 자동 백업 서비스인 AWS 스토리지 게이트웨이(Storage Gateway)를 이용하는 패턴
 - ◆ 첫 번째나 두 번째 방법보다 적은 노력으로 자동 백업 환경을 구축 할 수 있으면서도 백업 관리의 일원화도 가능
 - ◆ 스토리지 게이트웨이는 온프레미스 환경에 있는 서버에 스토리지 관리 전용 소프트웨어를 설치하여 AWS와 접속하는 서비스

S3

❖ S3 개요

- ✓ S3(Simple Storage Service)는 인터넷 스토리지 서비스
- ✓ 용량에 관계없이 파일을 저장할 수 있고 웹 (HTTP 프로토콜)에서 파일에 접근할 수 있으며 안정성이 뛰어나고 가용성이 높으며 무제한 확장이 가능
- ✓ 대용량의 파일 저장을 EC2 와 EBS를 통해 구축한다면 상당히 많은 비용이 들고 노력이 요구되지만 S3는 저장 용량이 무한대이고 파일 저장에 최적화되어 있기 때문에 용량을 추가하거나 성능을 높이는 작업을 하지 않아도 되고 비용도 EC2 와 EBS로 구축하는 것보다 훨씬 저렴
- ✓ EC2 와 EBS로 정적 웹 서비스(HTML과 JavaScript로만 구성된 웹사이트)를 구축한다면 일일이 EC2 와 EBS를 생성해 높은 요금을 낼 필요 없이 S3에서 바로 정적 웹 서비스(정적 웹 호스팅)를 사용하는 것이 가능
- ✓ S3 자체가 수천 대 이상의 매우 성능이 좋은 웹 서버로 구성되어 있어서 EC2와 EBS로 구축했을 때처럼 자동 횡적 확장(Auto Scaling)이나 부하 분산(Load Balancing) 신경 쓰지 않아도 됨
- ✓ 동적 웹 페이지(ASP, JSP, PHP, Ruby on Rails 등) 와 정적 웹 페이지가 섞여 있다면 동적 웹 페이지만 EC2에서 서비스하고 정적 웹 페이지는 S3를 이용하면 성능도 높일 수 있고 비용도 절감할 수 있음
- ✓ S3는 파일 업로드/다운로드를 모두 HTTP 프로토콜로 처리하기 때문에 별도의 클라이언트나 ActiveX 설치없이 웹 브라우저에서 바로 사용 가능
- ✓ Amazon S3는 대부분의 회사에서 사용해서 오래전 구축한 파일 서버 형태를 탈피하고 있는데 넷플릭스가 수십억 시간에 달하는 콘텐츠를 Amazon S3를 통해 서비스하고 있으며 Airbnb는 10 페타 바이트가 넘는 사용자 사진 그리고 백업 데이터 와 정적 파일을 모두 Amazon S3에 저장해서 사용하고 있음

S3

❖ S3 개요

✓ 기본 개념

● 객체

- ◆ 파일과 메타 데이터로 이루어진 데이터가 저장되는 기본 단위
- ◆ 기본적으로 키(Key)가 객체의 이름이며 값(Value)이 객체의 데이터
- ◆ 객체 하나의 크기는 1Byte ~ 5TB
- ◆ 메타 데이터는 MIME형식으로 확장자를 통해 자동 설정되고 사용자가 임의로 지정
- ◆ MIME(Multipurpose Internet Mail Extensions)이란 파일 변환을 의미하는데 이메일과 함께 동봉할 파일을 텍스트 문자로 전환해서 이메일 시스템을 통해 전달하기 위해 개발되었기 때문에 이름에 Internet Mail Extension이 들어가며 현재는 웹을 통해 여러 형태의 파일을 전달하는 데 쓰이고 있음

● Bucket

- ◆ Amazon S3에서 생성할 수 있는 최상위 디렉토리
- ◆ Bucket은 리전 별로 생성되며 계정 별로 100개까지 생성이 가능
- ◆ Bucket 안에 객체가 저장되고 디렉토리 생성 가능(객체 이름이 디렉터리 경로까지 포함)
- ◆ 저장 가능한 객체의 개수 와 저장 가능한 용량은 무제한
- ◆ 접속 제어 및 권한 관리가 가능
- ◆ <http://examplebucket.s3.amazonaws.com/helloworld.jpg> 처럼 URL로 접근 가능

S3

❖ S3 개요

✓ 기본 개념

- 내구성과 가용성: 1년 기준으로 99.999999999% 내구성, 99.99% 가용성을 가지고 있음
- 내구성은 데이터가 유실되지 않는 것을 의미하고 가용성은 언제나 정상적으로 사용 가능한 상태
- 내구성이 다른 2가지 스토리지 옵션
 - ◆ 표준 스토리지(Standard Storage): 일반적인 스토리지 옵션으로 99.999999999% 내구성을 가지고 있지만 AWS 내부적으로도 이런 높은 내구성을 유지하려면 그만큼 비용이 많이 들게 되고 요금도 높아지기 때문에 유실되면 안 되는 중요한 데이터 저장에 권장
 - ◆ 낮은 중복 스토리지(RRS - Reduced Redundancy Storage): 표준 스토리지보다는 낮은 99.99%의 내구성을 가지고 있는데 데이터를 복제한 사본의 수를 줄여 비용을 낮추었기 때문에 요금이 저렴한데 원본 데이터를 다른 곳에 가지고 있거나 동영상이나 이미지의 썸네일 등 원본에서 다시 생성할 수 있는 데이터 저장에 적합
- 요금: 저장 용량과 데이터 전송량, HTTP 요청(Request) 개수로 책정

S3

❖ S3 Bucket 생성

- ✓ IAM 사용자 생성 및 권한 설정 - 외부에서 사용을 하고자 하는 경우에 생성

사용자 세부 정보 지정

사용자 세부 정보

사용자 이름

adam

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 [도움 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

S3

❖ S3 Bucket 생성

- ✓ IAM 사용자 생성 및 권한 설정 - 외부에서 사용을 하고자 하는 경우에 생성

권한 설정

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

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 (2/1181)

서 사용자에게 연결할 정책을 하나 이상 선택합니다.

필터링 기준 유형: 모든 유형 12 개 일치 < 1 >

☐	정책 이름	유형	연결된 엔터티
☐	AmazonDMSRedshiftS3Role	AWS 관리형	0
☒	AmazonS3FullAccess	AWS 관리형	0

S3

❖ S3 Bucket 생성

- ✓ IAM 사용자 생성 및 권한 설정 - 외부에서 사용을 하고자 하는 경우에 생성

검토 및 생성

선택 사항을 검토합니다. 사용자를 생성한 후 자동 생성된 암호를 보고 다운로드할 수 있습니다(활성화된 경우).

사용자 세부 정보

사용자 이름
adam

콘솔 암호 유형
None

암호 재설정 필요
아니오

권한 요약

< 1 >

이름



유형



다음과 같이 사용



[AmazonS3FullAccess](#)

AWS 관리형

권한 정책

[CloudFrontFullAccess](#)

AWS 관리형

권한 정책

S3

❖ S3 Bucket 생성

- ✓ IAM 사용자 생성 및 권한 설정 - 외부에서 사용을 하고자 하는 경우에 생성
 - ✓ IAM 사용자 Access Key 및 Secret Key 생성
 - <https://console.aws.amazon.com/iam/>에서 IAM 콘솔 열기
 - 탐색 메뉴에서 사용자를 선택
 - IAM 사용자 이름(확인란이 아님)을 선택
 - Security credentials(보안 자격 증명) 탭을 연 다음 Create access key(액세스 키 생성)를 선택
 - 새 액세스 키를 보려면 [Show]를 선택
 - ◆ 액세스 키 ID: AKIAIOSFODNN7EXAMPLE
 - ◆ 보안 액세스 키: wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY
 - 키 페어 파일을 다운로드하려면 [Download .csv file]을 선택한 후 안전한 위치에 키와 함께 .csv 파일을 저장

S3

❖ S3 Bucket 생성

- ✓ IAM 사용자 생성 및 권한 설정 - 외부에서 사용을 하고자 하는 경우에 생성
- ✓ IAM 사용자 Access Key 및 Secret Key 생성

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

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

S3

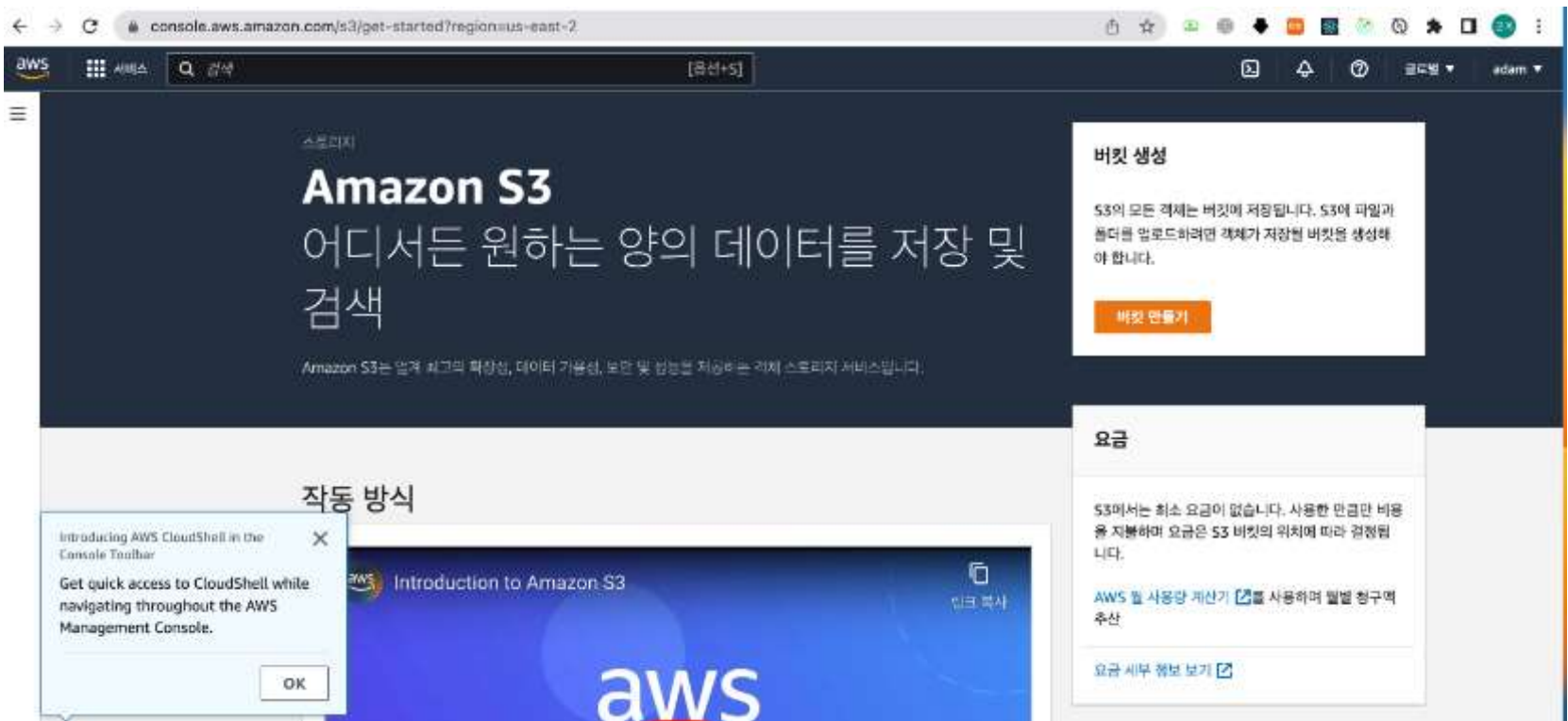
❖ S3 Bucket 생성

- ✓ IAM 사용자 생성 및 권한 설정 - 외부에서 사용을 하고자 하는 경우에 생성
 - IAM 사용자 Access Key 및 Secret Key 생성

Access key ID	Secret access key			
AKIAZKP7ENI	Rje6Hea+o8FTlxO0/taoRTNG1WT3vfuLAcfZhKrt			

S3

- ❖ S3 Bucket 생성
 - ✓ Bucket 생성
 - <https://console.aws.amazon.com/s3>



S3

❖ S3 Bucket 생성

✓ Bucket 생성

● Bucket 만들기 – 이름 과 리전 설정



S3

- ❖ S3 Bucket 생성
 - ✓ Bucket 생성
 - Bucket 만들기 – ACL 설정(ACL 활성화됨 선택)

객체 소유권 Info

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

 **ACL 비활성화 관련 향후 권한 변경 사항**
2023년 4월부터는 S3 콘솔을 사용하여 버킷을 생성할 때 ACL을 비활성화하기 위해 더 이상 s3:PutBucketOwnershipControls 권한이 필요하지 않습니다. [자세히 알아보기](#)

S3

❖ S3 Bucket 생성

✓ Bucket 생성

● Bucket 만들기 – 퍼블릭 액세스 차단 설정(차단 설정 해제)

이 버킷의 퍼블릭 액세스 차단 설정

퍼블릭 액세스는 ACL(엑세스 제어 목록), 버킷 정책, 액세스 지정 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 지정에만 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 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.

S3

- ❖ S3 Bucket 생성
 - ✓ Bucket 생성
 - Bucket 만들기 – 기타 옵션 설정

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

태그 (0) - 선택 사항

버킷 태그를 사용하여 스토리지 비용을 추적하고 버킷을 구성할 수 있습니다. [자세히 알아보기](#) 

이 버킷과 연결된 태그가 없습니다.

태그 추가

S3

- ❖ S3 Bucket 생성
 - ✓ Bucket 생성
 - Bucket 만들기 – 기타 옵션 설정

기본 암호화 [Info](#)

서버 측 암호화는 이 버킷에 저장된 새 객체에 자동으로 적용됩니다.

암호화 키 유형 [Info](#)

- ☒ Amazon S3 관리형 키(SSE-S3)
- ☐ AWS Key Management Service 키(SSE-KMS)

버킷 키

KMS 암호화가 이 버킷의 새 객체를 암호화하는 데 사용되는 경우 버킷 키는 AWS KMS에 대한 호출을 줄여 암호화 비용을 줄입니다. [자세히 알아보기](#) 

- ☐ 비활성화
- ☒ 활성화

S3

- ❖ S3 Bucket 생성
 - ✓ Bucket 생성
 - Bucket 만들기

Amazon S3 X

버킷 'iustudygyberadam'(1) 생성됨
파일 및 폴더를 업로드하거나 추가 버킷 설정을 구성하려면 [세부 정보 보기]를 선택하세요.

세부 정보 보기 X

Amazon S3 > 버킷

계정 스냅샷
Storage Lens는 스토리지 사용량 및 비용 추세를 대한 기사를 제공합니다. 자세히 알아보기

Storage Lens 대시보드 보기

버킷 (1) Info
버킷은 S3에 저장되는 데이터의 컨테이너입니다. 자세히 알아보기

이름으로 버킷 찾기

이름 ▲ AWS 리전 ▼ 액세스 ▼ 생성 날짜 ▼

iustudygyberadam	아시아 태평양(서울) ap-northeast-2	버킷 및 객체가 퍼블릭이 아님	2023. 4. 3. am 7:42:44 AM KST
------------------	----------------------------	------------------	-------------------------------

S3

❖ S3 Bucket 생성

✓ 설정

- 외부에서 Bucket을 사용할 수 있도록 [권한] – [버킷 정책] 에서 [편집]을 눌러서 Bucket 정책 추가

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PublicListGet",
      "Effect": "Allow",
      "Principal": "*",
      "Action": [
        "s3:List*",
        "s3:Get*",
        "s3:Put*",
        "s3:Delete*"
      ],
      "Resource": [
        "arn:aws:s3:::Bucket이름",
        "arn:aws:s3:::Bucket이름/*"
      ]
    }
  ]
}
```

S3

❖ S3 Bucket 생성

✓ 설정

- [권한] – [CORS] 에서 CORS 정책 추가

```
[
  {
    "AllowedHeaders": [
      "*"
    ],
    "AllowedMethods": [
      "GET",
      "PUT",
      "POST",
      "DELETE"
    ],
    "AllowedOrigins": [
      "*"
    ],
    "ExposeHeaders": ["Access-Control-Allow-Origin"]
  }
]
```

S3

- ❖ S3 Bucket 생성
 - ✓ 파일 업로드



S3

- ❖ S3 Bucket 생성
 - ✓ 파일 다운로드

객체 개요	
소유자 ggangpae1	S3 URI  s3://itstudybucket/Lambda.pptx
AWS 리전 아시아 태평양(도쿄) ap-northeast-1	Amazon 리소스 이름(ARN)  arn:aws:s3:::itstudybucket/Lambda.pptx
마지막 수정 2023. 10. 24. am 7:37:33 AM KST	엔터티 태그(Etag)  4cd77ba64e179fb16f40604669ba5ece
크기 1.8MB	객체 URL  https://itstudybucket.s3.ap-northeast-1.amazonaws.com/Lambda.pptx
유형 pptx	
키  Lambda.pptx	

S3

❖ django 에서의 파일 업로드

✓ 프로젝트 생성

- 프로젝트 디렉토리 생성: pythons3
- 가상 환경 생성
 - ◆ `python3 -m venv myenv`
 - ◆ `source myenv/bin/activate`
- 필요한 패키지 설치
 - ◆ `pip install Django`
 - ◆ `pip install django-cors-headers`
 - ◆ `pip install mysqlclient`
- 프로젝트 생성: `django-admin startproject s3project .`
- 애플리케이션 생성: `python manage.py startapp s3application`

S3

❖ django 에서의 파일 업로드

✓ 프로젝트 설정

● s3project/settings.py 파일을 수정

```
# SECURITY WARNING: don't run with debug turned on in production!  
DEBUG = True
```

```
ALLOWED_HOSTS = ['*']  
# Application definition
```

```
INSTALLED_APPS = [  
    "django.contrib.admin",  
    "django.contrib.auth",  
    "django.contrib.contenttypes",  
    "django.contrib.sessions",  
    "django.contrib.messages",  
    "django.contrib.staticfiles",  
    "s3application",  
    "corsheaders", #CORS 관련 추가  
]
```

S3

❖ django 에서의 파일 업로드

✓ 프로젝트 설정

- s3project/settings.py 파일을 수정

```
MIDDLEWARE = [  
    'corsheaders.middleware.CorsMiddleware', # CORS 관련 추가 (가장 상단에 위치)  
    "django.middleware.security.SecurityMiddleware",  
    "django.contrib.sessions.middleware.SessionMiddleware",  
    "django.middleware.common.CommonMiddleware",  
    "django.middleware.csrf.CsrfViewMiddleware",  
    "django.contrib.auth.middleware.AuthenticationMiddleware",  
    "django.contrib.messages.middleware.MessageMiddleware",  
    "django.middleware.clickjacking.XFrameOptionsMiddleware",  
]
```

S3

❖ django 에서의 파일 업로드

✓ 프로젝트 설정

● s3project/settings.py 파일을 수정

#CORS 관련 추가

CORS_ALLOW_ALL_ORIGINS = True #(모든 포트 허용)

#HTTP methods 추가

```
CORS_ALLOW_METHODS = (  
    "DELETE",  
    "GET",  
    "OPTIONS",  
    "PATCH",  
    "POST",  
    "PUT",  
)
```

S3

❖ django 에서의 파일 업로드

✓ 프로젝트 설정

● s3project/settings.py 파일을 수정

Database

<https://docs.djangoproject.com/en/4.1/ref/settings/#databases>

```
DATABASES = {
```

```
    'default': {
```

```
        'ENGINE': 'django.db.backends.mysql',
```

```
        'NAME': 'adam',
```

```
        'USER': 'root',
```

```
        'PASSWORD': 'wnddkd', # mariaDB 설치 시 입력한 root 비밀번호 입력
```

```
        'HOST': '127.0.0.1',
```

```
        'PORT': "
```

```
    }
```

```
}
```

```
TIME_ZONE = "Asia/Seoul"
```

S3

- ❖ django 에서의 파일 업로드
 - ✓ 프로젝트 설정
 - M1 맥북을 사용하는 경우
 - ◆ pip install pymysql
 - ◆ myproject/settings.py 파일에 추가

```
import pymysql
pymysql.install_as_MySQLdb()
```

S3

❖ django 에서의 파일 업로드

✓ 모델 생성

- s3application 디렉토리의 models.py 파일 수정 – 데이터베이스 테이블 과 연동할 모델 생성

```
from django.db import models
```

```
class FileContent(models.Model):
```

```
    title = models.TextField(max_length=40, null=True)
```

```
    imgfile = models.ImageField(null=True, upload_to="", blank=True)
```

```
    content = models.TextField()
```

```
    def __str__(self):
```

```
        return self.title
```

S3

- ❖ django 에서의 파일 업로드
 - ✓ 모델 생성
 - migration
 - # 마이그레이션 파일 만들기
`python manage.py makemigrations`
 - # 마이그레이션 적용
`python manage.py migrate`
 - 데이터베이스에 테이블이 만들어 졌는지 확인

S3

❖ django 에서의 파일 업로드

✓ 요청 처리

- s3application의 views.py 파일을 수정

```
from .models import FileContent
from django.http import JsonResponse
from django.views.decorators.csrf import csrf_exempt
```

```
@csrf_exempt
def fileUpload(request):
    if request.method == 'POST':
        title = request.POST['title']
        content = request.POST['content']
        img = request.FILES["imgfile"]
        fileContent = FileContent(
            title=title,
            content=content,
            imgfile=img,
        )
        print(fileContent)
        fileContent.save()
        return JsonResponse({"result":"success"})
    else:
        return JsonResponse({"result":"fail"})
```

S3

❖ django 에서의 파일 업로드

✓ 요청 처리

● urls.py 파일을 수정

```
from django.urls import path, include
from django.conf.urls.static import static
from django.conf import settings
```

```
# fileupload/urls.py
from django.urls import path
from fileupload import views
```

```
urlpatterns = [
    path('fileupload/', views.fileUpload, name="fileupload"),
]
```

S3

❖ django 에서의 파일 업로드

✓ 요청 처리

- settings.py 파일에 파일을 저장하기 위한 옵션을 추가

```
import os
MEDIA_URL = '/media/'
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
```

S3

- ❖ django 에서의 파일 업로드
 - ✓ 프로젝트를 실행해서 파일 업로드 확인
 - 실행: `python manage.py runserver`
 - Postman 에서 테스트

POST 127.0.0.1:8000/fileupload/ Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

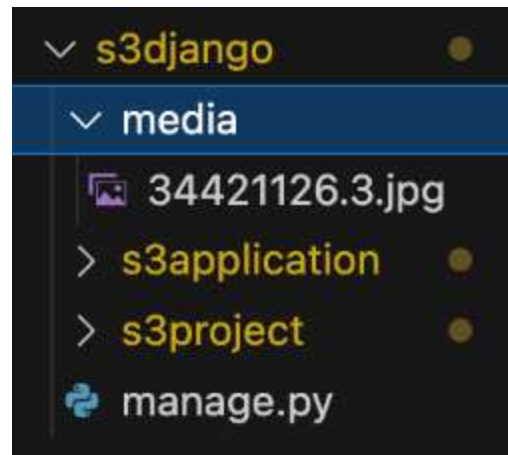
☐ none ☒ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

Key	Value	Description
☒ title	Text 필수지	
☒ content	Text 드림라	
☒ imgfile	File 34421126.3.jpg	

Key Value Description

S3

- ❖ django 에서의 파일 업로드
 - ✓ 프로젝트를 실행해서 파일 업로드 확인
 - 프로젝트에서 파일 업로드 확인



S3

- ❖ django 에서의 파일 업로드
 - ✓ 프로젝트를 실행해서 파일 업로드 확인
 - 데이터베이스 확인
- ```
SELECT *
FROM s3application_filecontent;
```



| s3application_filecontent 1 X                                    |    |       |                |         |
|------------------------------------------------------------------|----|-------|----------------|---------|
| SELECT * FROM s3application_filecontent   Enter a SQL expression |    |       |                |         |
|                                                                  | id | title | imgfile        | content |
| 1                                                                | 1  | 배수지   | 34421126.3.jpg | 드림하     |

# S3

- ❖ django 에서의 파일 업로드
  - ✓ AWS에 업로드
    - 패키지 설치

```
pip install boto3
```

```
pip install django-storages
```

# S3

## ❖ django 에서의 파일 업로드

### ✓ AWS에 업로드

#### ● settings.py 파일에 추가

# AWS Setting

AWS\_REGION = 'ap-northeast-1' #AWS서버의 지역

AWS\_STORAGE\_BUCKET\_NAME = 'itstudybucket' #생성한 Bucket 이름

AWS\_ACCESS\_KEY\_ID = 'AKIA264FV5MWPNFTR67R' #액세스 키 ID

AWS\_SECRET\_ACCESS\_KEY = 'Dkhqua9VeWRvYINfP5yZQxuUc/ZHLI6W5IJ++0D2' #액세스 키 PW

#Bucket이름.s3.AWS서버지역.amazonaws.com 형식

AWS\_S3\_CUSTOM\_DOMAIN = '%s.s3.%s.amazonaws.com' %  
(AWS\_STORAGE\_BUCKET\_NAME, AWS\_REGION)

# Static Setting

STATIC\_URL = "https://%s/static/" % AWS\_S3\_CUSTOM\_DOMAIN

STATICFILES\_STORAGE = 'storages.backends.s3boto3.S3Boto3Storage'

# S3

## ❖ django 에서의 파일 업로드

### ✓ AWS에 업로드

#### ● settings.py 파일에 추가

# Media Setting

MEDIA\_URL = "https://%s/meida/" % AWS\_S3\_CUSTOM\_DOMAIN

DEFAULT\_FILE\_STORAGE = 'storages.backends.s3boto3.S3Boto3Storage'

# Default primary key field type

# <https://docs.djangoproject.com/en/4.1/ref/settings/#default-auto-field>

# S3

- ❖ django 에서의 파일 업로드
  - ✓ AWS에 업로드
    - AWS 와 연결: `python manage.py collectstatic`
      - ◆ Bucket을 확인하면 static 파일이 업로드 되고 admin 이라는 디렉토리가 생성된 것을 확인할 수 있음

# S3

- ❖ django 에서의 파일 업로드
  - ✓ AWS에 업로드
    - 프로젝트를 실행하고 파일 업로드를 수행한 후 버킷을 확인



# S3

- ❖ django 에서의 파일 업로드
  - ✓ AWS에 업로드되는 파일 이름 수정
    - views.py 파일의 업로드 처리 메서드 수정

```
import uuid

@csrf_exempt
def fileUpload(request):
 if request.method == 'POST':
 title = request.POST['title']
 content = request.POST['content']
 img = request.FILES["imgfile"]
 idx = img.name.rfind(".")
 if idx != 1:
 img.name = img.name[:idx] + str(uuid.uuid4()) + img.name[idx:]
 else:
 img.name = img.name + str(uuid.uuid4())
 fileupload = FileUpload(
 title=title,
 content=content,
 imgfile=img,
)
 fileupload.save()
 return redirect('fileupload')
```

# S3

## ❖ django 에서의 파일 업로드

### ✓ AWS에 업로드되는 파일 이름 수정

#### ● views.py 파일의 업로드 처리 메서드 수정

else:

```
fileuploadForm = FileUploadForm
```

```
context = {
```

```
 'fileuploadForm': fileuploadForm,
```

```
}
```

```
return render(request, 'fileupload.html', context)
```

# S3

- ❖ django 에서의 파일 업로드
  - ✓ AWS에 업로드되는 파일 이름 수정
    - 프로젝트를 실행하고 파일 업로드를 수행한 후 버킷을 확인



# S3

## ❖ django 에서의 파일 업로드

### ✓ AWS 이미지 출력

- views.py 파일에 기본 요청 처리 메서드 추가

```
def main(request):
```

```
 pictures = FileUpload.objects.all()
```

```
 return render(request, 'main.html', {'result': pictures})
```

# S3

## ❖ django 에서의 파일 업로드

### ✓ AWS 이미지 출력

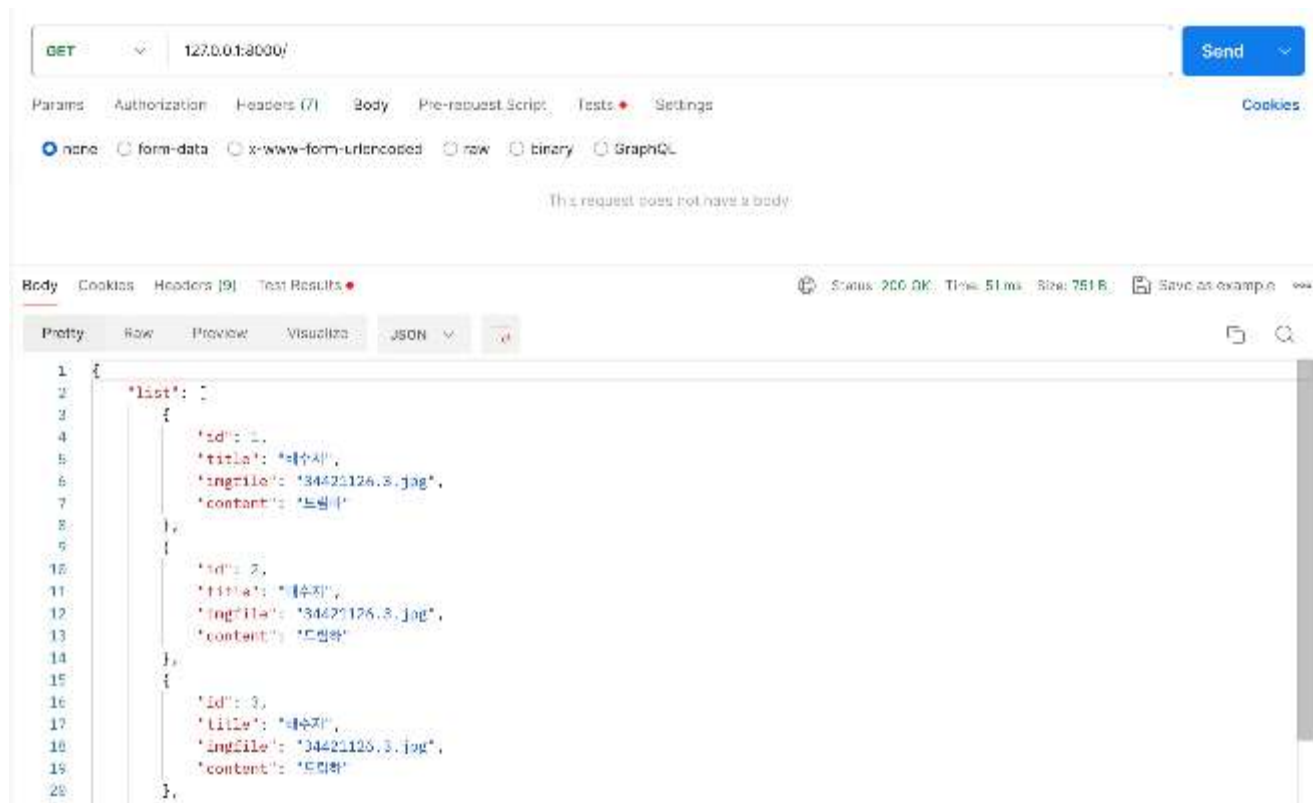
#### ● urls.py 파일 수정

```
from django.urls import path
from s3application import views
```

```
urlpatterns = [
 path("", views.main),
 path('fileupload/', views.fileUpload, name="fileupload"),
]
```

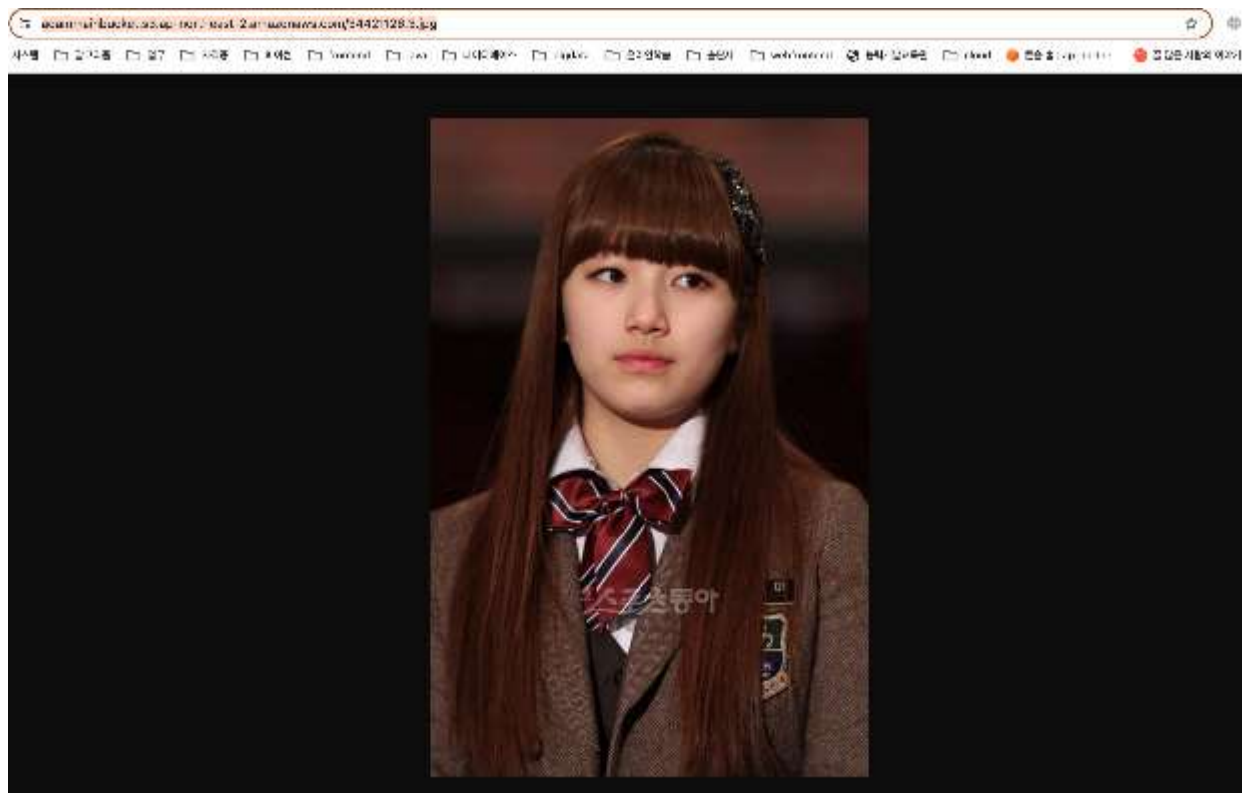
# S3

- ❖ django 에서의 파일 업로드
  - ✓ AWS 이미지 출력
  - 프로젝트 수행 후 기본 요청 수행 후 결과 확인



# S3

- ❖ django 에서의 파일 업로드
  - ✓ AWS 이미지 출력
    - 브라우저에서 imgfile 앞에 버킷의 EndPoint를 추가해서 다운로드 되는지 확인



# S3

## ❖ Spring Boot 프로젝트

### ✓ 파일 업로드

- Spring Boot 프로젝트 생성 – Spring Web, Lombok 의 의존성만 설정
- 프로젝트의 build.gradle에 S3 사용을 위한 의존성 추가  
implementation 'io.awspring.cloud:spring-cloud-starter-aws:2.3.1'

# S3

## ❖ Spring Boot 프로젝트

### ✓ 파일 업로드

- application.properties 파일에 S3 설정을 추가

# AWS Account Credentials (AWS 접근 키)

cloud.aws.credentials.accessKey=AKIAWN5SS7IWEYL55SWG

cloud.aws.credentials.secretKey=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

# AWS S3 bucket Info (S3 Bucket정보)

cloud.aws.s3.bucket=adamsoft

cloud.aws.region.static=us-east-1

cloud.aws.stack.auto=false

# file upload max size (파일 업로드 크기 설정)

spring.servlet.multipart.max-file-size=20MB

spring.servlet.multipart.max-request-size=20MB

# S3

## ❖ Spring Boot 프로젝트

### ✓ 파일 업로드

- 업로드 용량 초과 예외를 처리할 예외 처리 클래스 생성

```
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Component;
import org.springframework.web.ErrorResponse;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.bind.annotation.RestControllerAdvice;
import org.springframework.web.multipart.MaxUploadSizeExceededException;
```

## ❖ Spring Boot 프로젝트

## ✓ 파일 업로드

- 업로드 용량 초과 예외를 처리할 예외 처리 클래스 생성

```
@Component
@RestControllerAdvice
public class FileUploadFailedException{
 @ExceptionHandler(MaxUploadSizeExceededException.class)
 protected ResponseEntity<ErrorResponse>
 handleMaxUploadSizeExceededException(
 MaxUploadSizeExceededException e) {

 ErrorResponse response = ErrorResponse.builder(e, HttpStatus.BAD_REQUEST,
"용량 초과").build();
 return new ResponseEntity<>(response, HttpStatus.BAD_REQUEST);
 }
}
```

# S3

## ❖ Spring Boot 프로젝트

### ✓ 파일 업로드

- 업로드할 파일 경로를 위한 클래스를 생성

```
public class CommonUtils {
 private static final String FILE_EXTENSION_SEPARATOR = ".";
 private static final String CATEGORY_PREFIX = "/";
 private static final String TIME_SEPARATOR = "_";

 public static String buildFileName(String category, String originalFileName) {
 int fileExtensionIndex =
 originalFileName.lastIndexOf(FILE_EXTENSION_SEPARATOR);
 String fileExtension = originalFileName.substring(fileExtensionIndex);
 String fileName = originalFileName.substring(0, fileExtensionIndex);
 String now = String.valueOf(System.currentTimeMillis());

 return category + CATEGORY_PREFIX + fileName + TIME_SEPARATOR + now
 + fileExtension;
 }
}
```

# S3

## ❖ Spring Boot 프로젝트

### ✓ 파일 업로드

#### ● 서비스 클래스를 생성

```
import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.AWSStaticCredentialsProvider;
import com.amazonaws.auth.BasicAWSCredentials;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.CannedAccessControlList;
import com.amazonaws.services.s3.model.ObjectMetadata;
import com.amazonaws.services.s3.model.PutObjectRequest;
import jakarta.annotation.PostConstruct;
import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Service;
import org.springframework.web.multipart.MultipartFile;

import java.io.IOException;
import java.io.InputStream;
```

# S3

## ❖ Spring Boot 프로젝트

### ✓ 파일 업로드

#### ● 서비스 클래스를 생성

```
@Slf4j
@RequiredArgsConstructor
@Service
public class AwsS3Service {
 private AmazonS3 amazonS3Client;

 @Value("${cloud.aws.credentials.accessKey}")
 private String accessKey;

 @Value("${cloud.aws.credentials.secretKey}")
 private String secretKey;

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

 @Value("${cloud.aws.region.static}")
 private String region;
```

# S3

## ❖ Spring Boot 프로젝트

### ✓ 파일 업로드

#### ● 서비스 클래스를 생성

```
@PostConstruct
public void setS3Client() {
 AWSCredentials credentials = new BasicAWSCredentials(this.accessKey,
this.secretKey);

 amazonS3Client = AmazonS3ClientBuilder.standard()
 .withCredentials(new AWSStaticCredentialsProvider(credentials))
 .withRegion(this.region)
 .build();
}
```

# S3

## ❖ Spring Boot 프로젝트

### ✓ 파일 업로드

#### ● 서비스 클래스를 생성

```
public String uploadFileV1(String category, MultipartFile multipartFile) {
 boolean result = validateFileExists(multipartFile);
 if(result == false){
 return null;
 }
 String fileName = CommonUtils.buildFileName(category,
multipartFile.getOriginalFilename());
 ObjectMetadata objectMetadata = new ObjectMetadata();
 objectMetadata.setContentType(multipartFile.getContentType());
 try (InputStream inputStream = multipartFile.getInputStream()) {
 amazonS3Client.putObject(new PutObjectRequest(bucketName, fileName,
inputStream, objectMetadata)
 .withCannedAcl(CannedAccessControlList.PublicRead));
 } catch (IOException e) {
 return null;
 }
 return amazonS3Client.getUrl(bucketName, fileName).toString();
}
```

# S3

## ❖ Spring Boot 프로젝트

### ✓ 파일 업로드

#### ● 서비스 클래스를 생성

```
private boolean validateFileExists(MultipartFile multipartFile) {
 boolean result = true;
 if (multipartFile.isEmpty()) {
 result = false;
 }
 return result;
}
}
```

# S3

## ❖ Spring Boot 프로젝트

### ✓ 파일 업로드

#### ● 컨트롤러 클래스를 생성

```
import lombok.RequiredArgsConstructor;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RequestPart;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.multipart.MultipartFile;

@RestController
@RequiredArgsConstructor
public class S3FileController {

 private final AwsS3Service awsS3Service;

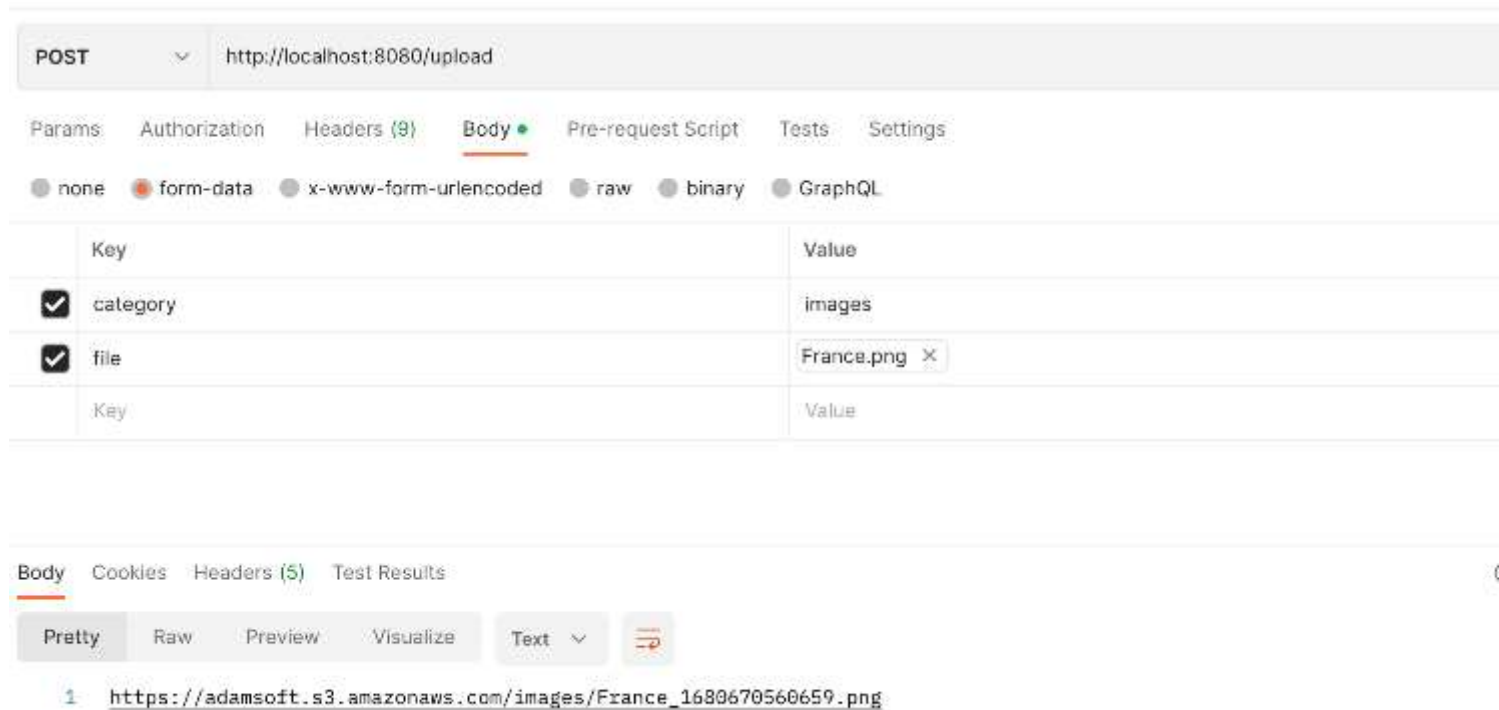
 @PostMapping("/upload")
 public String uploadFile(
 @RequestParam("category") String category,
 @RequestPart(value = "file") MultipartFile multipartFile) {
 return awsS3Service.uploadFileV1(category, multipartFile);
 }
}
```

# S3

## ❖ Spring Boot 프로젝트

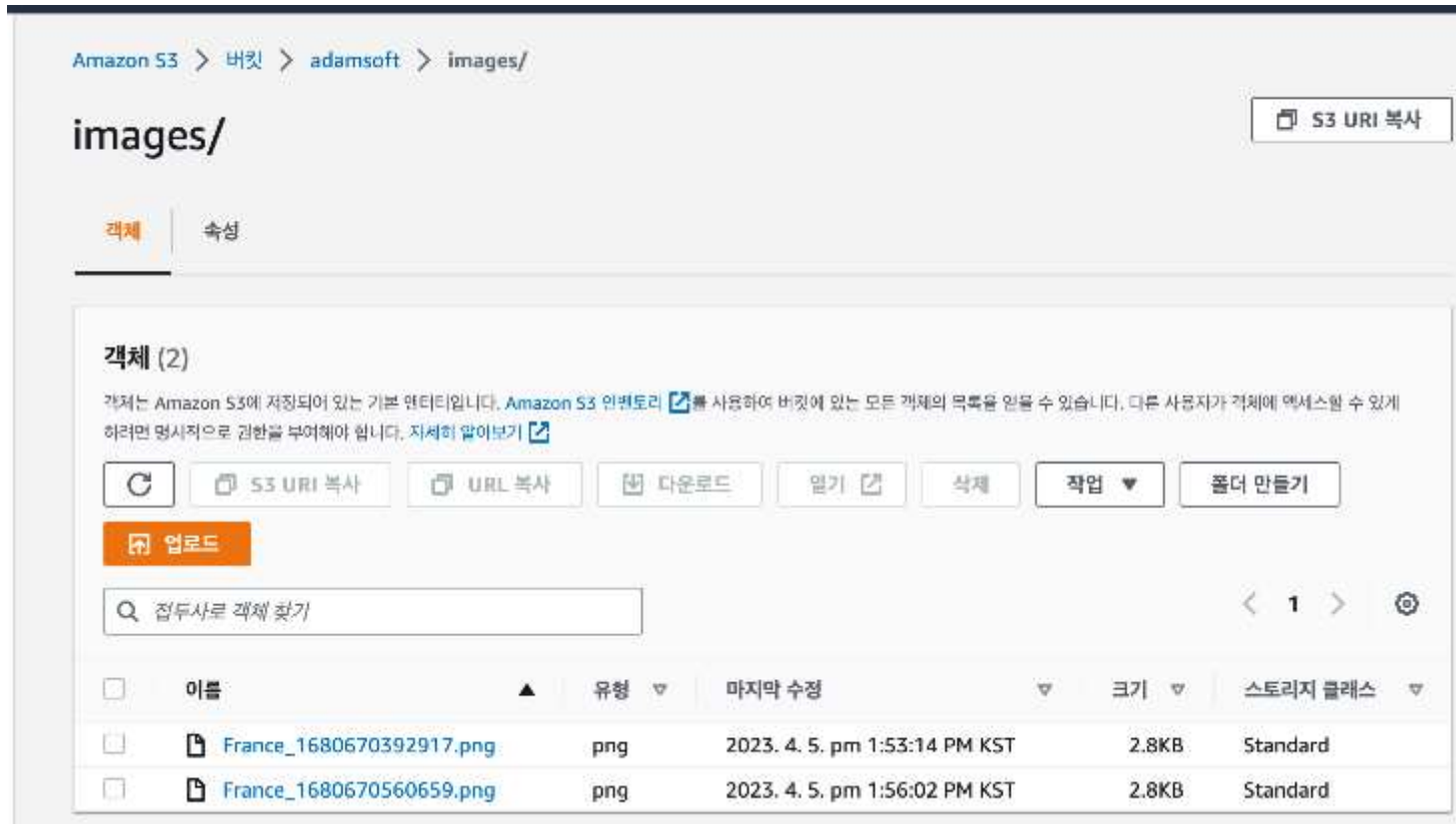
### ✓ 파일 업로드

#### ● 테스트



# S3

- ❖ Spring Boot 프로젝트
  - ✓ 파일 업로드
    - 확인



Amazon S3 > 버킷 > adamsoft > images/

images/ S3 URI 복사

객체 속성

객체 (2)

객체는 Amazon S3에 저장되어 있는 기본 엔티티입니다. [Amazon S3 인벤토리](#)를 사용하여 버킷에 있는 모든 객체의 목록을 얻을 수 있습니다. 다른 사용자가 객체에 액세스할 수 있게 하려면 명시적으로 권한을 부여해야 합니다. [지세히 알아보기](#)

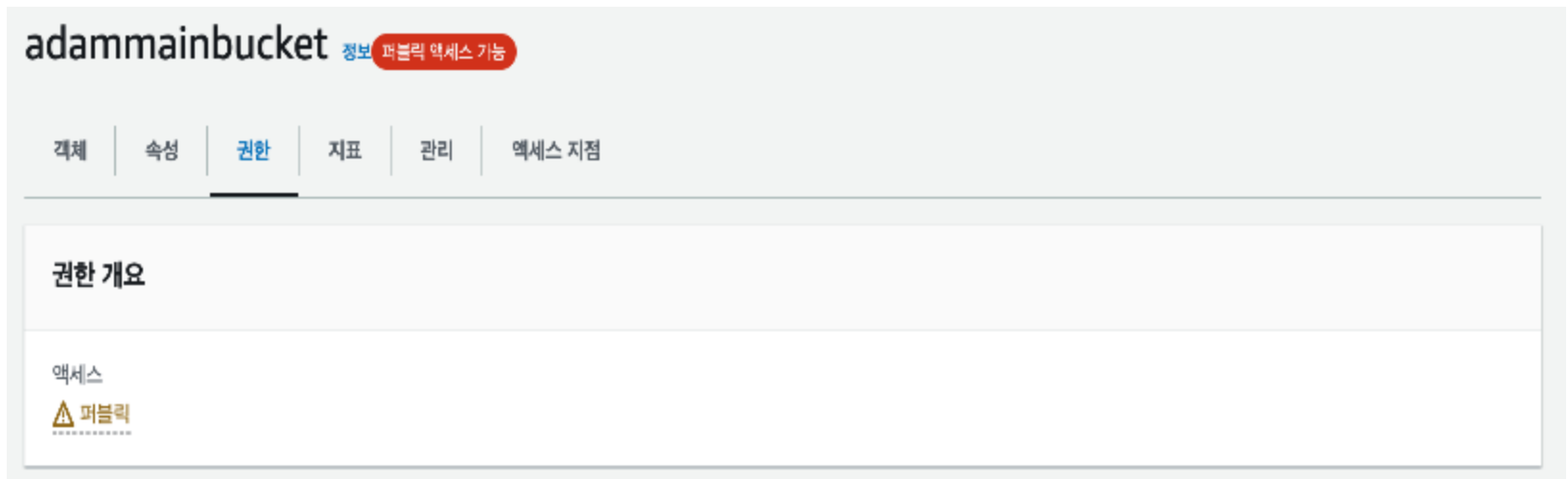
업로드 S3 URI 복사 URL 복사 다운로드 일기 삭제 작업 폴더 만들기

| <input type="checkbox"/> | 이름                                                                                                           | 유형  | 마지막 수정                        | 크기    | 스토리지 클래스 |
|--------------------------|--------------------------------------------------------------------------------------------------------------|-----|-------------------------------|-------|----------|
| <input type="checkbox"/> |  France_1680670392917.png | png | 2023. 4. 5. pm 1:53:14 PM KST | 2.8KB | Standard |
| <input type="checkbox"/> |  France_1680670560659.png | png | 2023. 4. 5. pm 1:56:02 PM KST | 2.8KB | Standard |

# S3

## ❖ Spring Boot 프로젝트

- ✓ ACL 문제로 업로드가 안되는 경우 버킷의 객체 소유권 변경
  - 버킷의 권한 탭 클릭



# S3

## ❖ Spring Boot 프로젝트

- ✓ ACL 문제로 업로드가 안되는 경우 버킷의 객체 소유권 변경
  - 객체 소유권 편집 클릭



# S3

## ❖ Spring Boot 프로젝트

- ✓ ACL 문제로 업로드가 안되는 경우 버킷의 객체 소유권 변경
  - 객체 소유권 수정

### 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 라이더가 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 라이더가 소유합니다.

☐ 객체 라이더

객체 라이더는 객체 소유자로 유지됩니다.

**① 새 객체에 대해서만 객체 소유권을 적용하려면 버킷 정책이 객체 업로드에 bucket-owner-full-control 삽입 ACL을 요구하도록 지정해야 합니다. [자세히 알아보기](#)**

# S3

## ❖ Spring Boot 프로젝트

### ✓ 파일 다운로드

- 다운로드 될 파일의 속성을 설정할 메서드를 CommonUtils 클래스에 추가

```
private static final int UNDER_BAR_INDEX = 1;

public static ContentDisposition createContentDisposition(String
 categoryWithFileName) {
 String fileName = categoryWithFileName.substring(
 categoryWithFileName.lastIndexOf(CATEGORY_PREFIX) +
 UNDER_BAR_INDEX);
 return ContentDisposition.builder("attachment")
 .filename(fileName, StandardCharsets.UTF_8)
 .build();
}
```

# S3

## ❖ Spring Boot 프로젝트

### ✓ 파일 다운로드

#### ● 다운로드 메서드를 Service 클래스에 추가

```
public byte[] downloadFileV1(String resourcePath) {
 boolean result = validateFileExistsAtUrl(resourcePath);
 if(result == false){
 return null;
 }

 S3Object s3Object = amazonS3Client.getObject(bucketName, resourcePath);
 S3ObjectInputStream inputStream = s3Object.getObjectContent();
 try {
 return IOUtils.toByteArray(inputStream);
 } catch (IOException e) {
 return null;
 }
}
```

# S3

## ❖ Spring Boot 프로젝트

### ✓ 파일 다운로드

#### ● 다운로드 메서드를 Service 클래스에 추가

```
private boolean validateFileExistsAtUrl(String resourcePath) {
 boolean result = true;
 if (!amazonS3Client.doesObjectExist(bucketName, resourcePath)) {
 result = false;
 }
 return result;
}
```

# S3

## ❖ Spring Boot 프로젝트

### ✓ 파일 다운로드

#### ● 컨트롤러 클래스에 추가

```
private HttpHeaders buildHeaders(String resourcePath, byte[] data) {
 HttpHeaders headers = new HttpHeaders();
 headers.setContentLength(data.length);
 headers.setContentType(MediaType.APPLICATION_OCTET_STREAM);

 headers.setContentDisposition(CommonUtils.createContentDisposition(resourceP
ath));
 return headers;
}
```

# S3

## ❖ Spring Boot 프로젝트

### ✓ 파일 다운로드

#### ● 컨트롤러 클래스에 추가

```
@GetMapping("/download")
public ResponseEntity<ByteArrayResource> downloadFile(
 @RequestParam("resourcePath") String resourcePath) {
 byte[] data = awsS3Service.downloadFileV1(resourcePath);
 ByteArrayResource resource = new ByteArrayResource(data);
 HttpHeaders headers = buildHeaders(resourcePath, data);

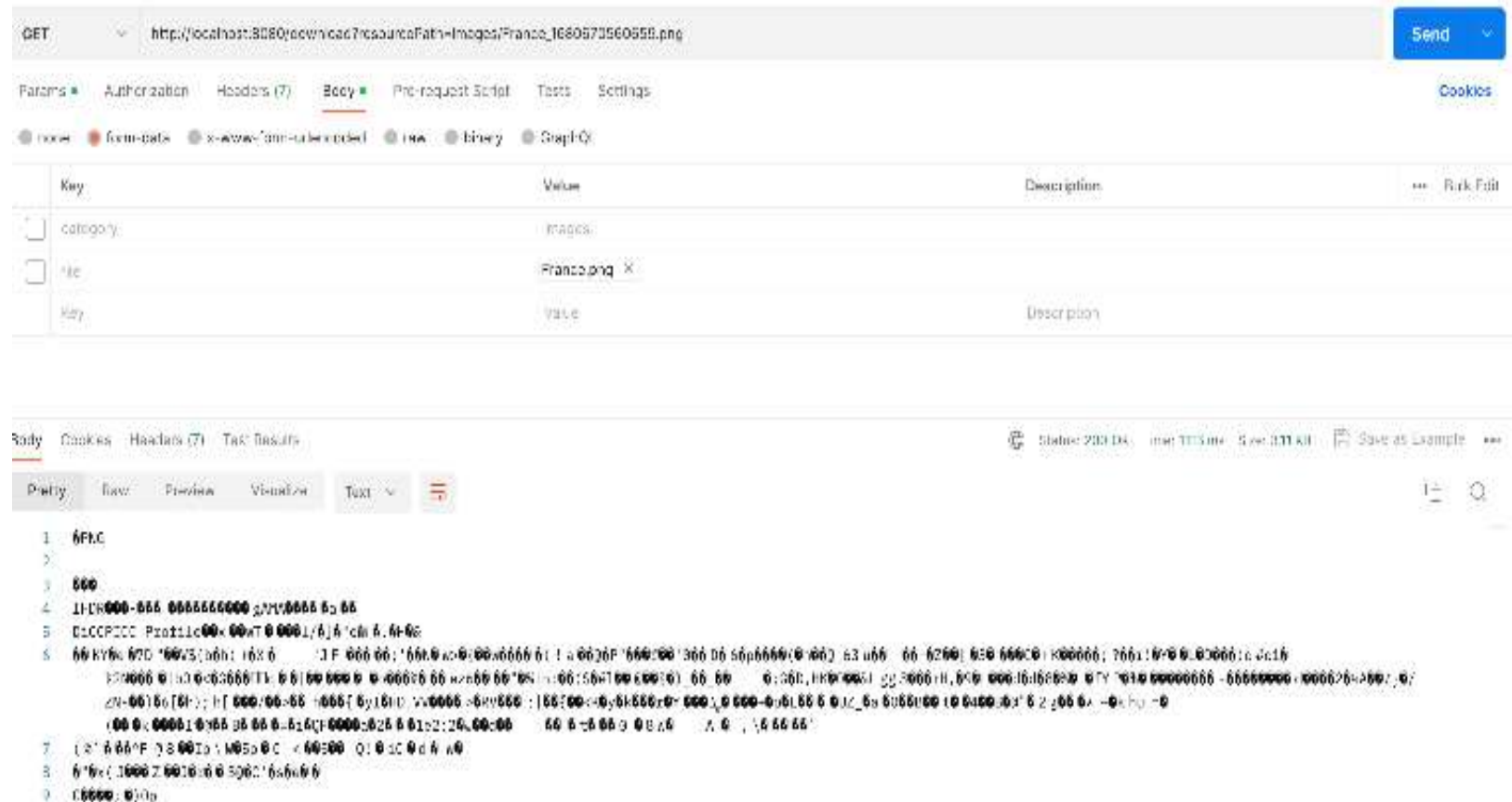
 return ResponseEntity
 .ok()
 .headers(headers)
 .body(resource);
}
```

# S3

## ❖ Spring Boot 프로젝트

✓ 파일 다운로드

● 테스트



# S3

## ❖ Spring Boot 프로젝트

### ✓ 다중 파일 업로드

#### ● 서비스 클래스에 메서드 추가

```
public List<String> uploadFile(String category, List<MultipartFile> multipartFiles) {
 List<String> fileUrls = new ArrayList<>();
```

```
 for (MultipartFile multipartFile : multipartFiles) {
```

```
 String fileName = CommonUtils.buildFileName(category,
multipartFile.getOriginalFilename());
 ObjectMetadata objectMetadata = new ObjectMetadata();
 objectMetadata.setContentType(multipartFile.getContentType());
```

# S3

## ❖ Spring Boot 프로젝트

### ✓ 다중 파일 업로드

#### ● 서비스 클래스에 메서드 추가

```
 try (InputStream inputStream = multipartFile.getInputStream()) {
 amazonS3Client.putObject(new PutObjectRequest(bucketName,
 fileName, inputStream, objectMetadata)
 .withCannedAcl(CannedAccessControlList.PublicRead));
 fileUrls.add(bucketName + "/" + fileName);
 } catch (IOException e) {
 log.warn(e.getMessage());
 }
 }

 return fileUrls;
}
```

# S3

## ❖ Spring Boot 프로젝트

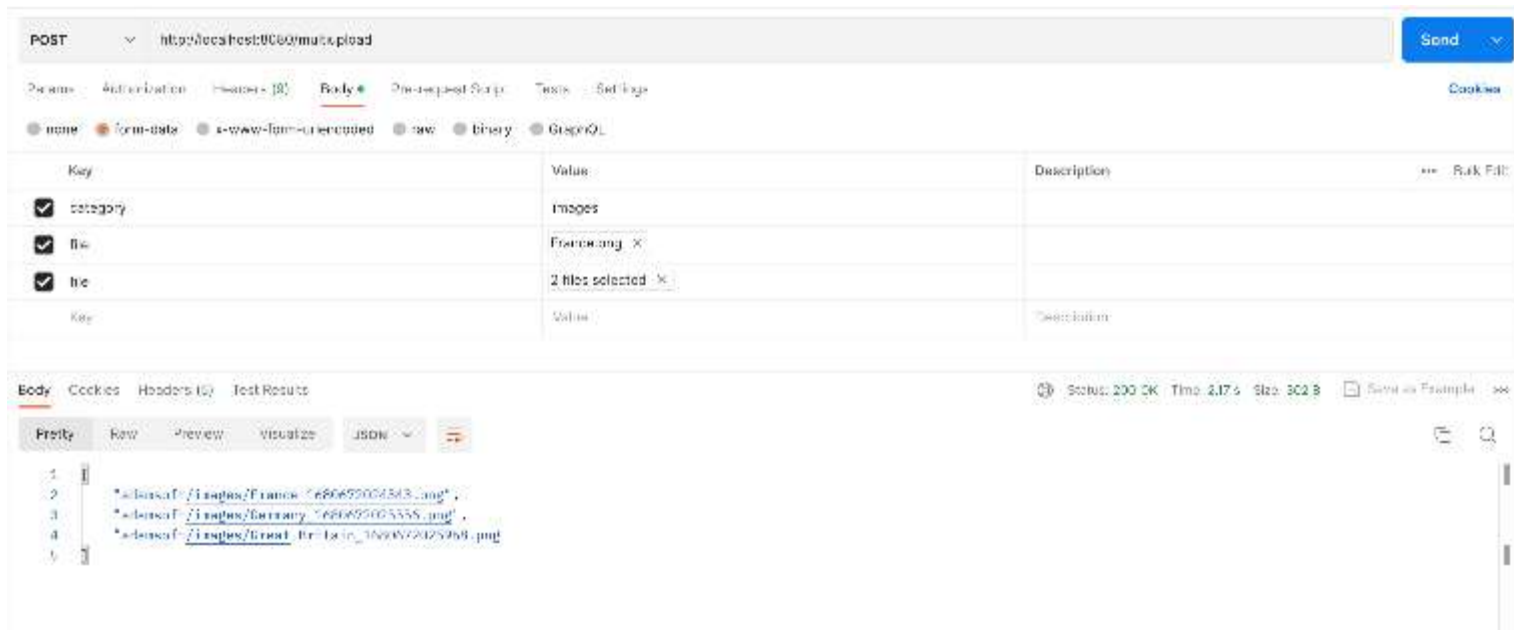
### ✓ 다중 파일 업로드

#### ● 컨트롤러 클래스에 메서드 추가

```
@PostMapping("/multiupload")
public List<String> uploadFile(
 @RequestParam("category") String category,
 @RequestPart(value = "file") List<MultipartFile> multipartFiles) {
 return awsS3Service.uploadFile(category, multipartFiles);
}
```

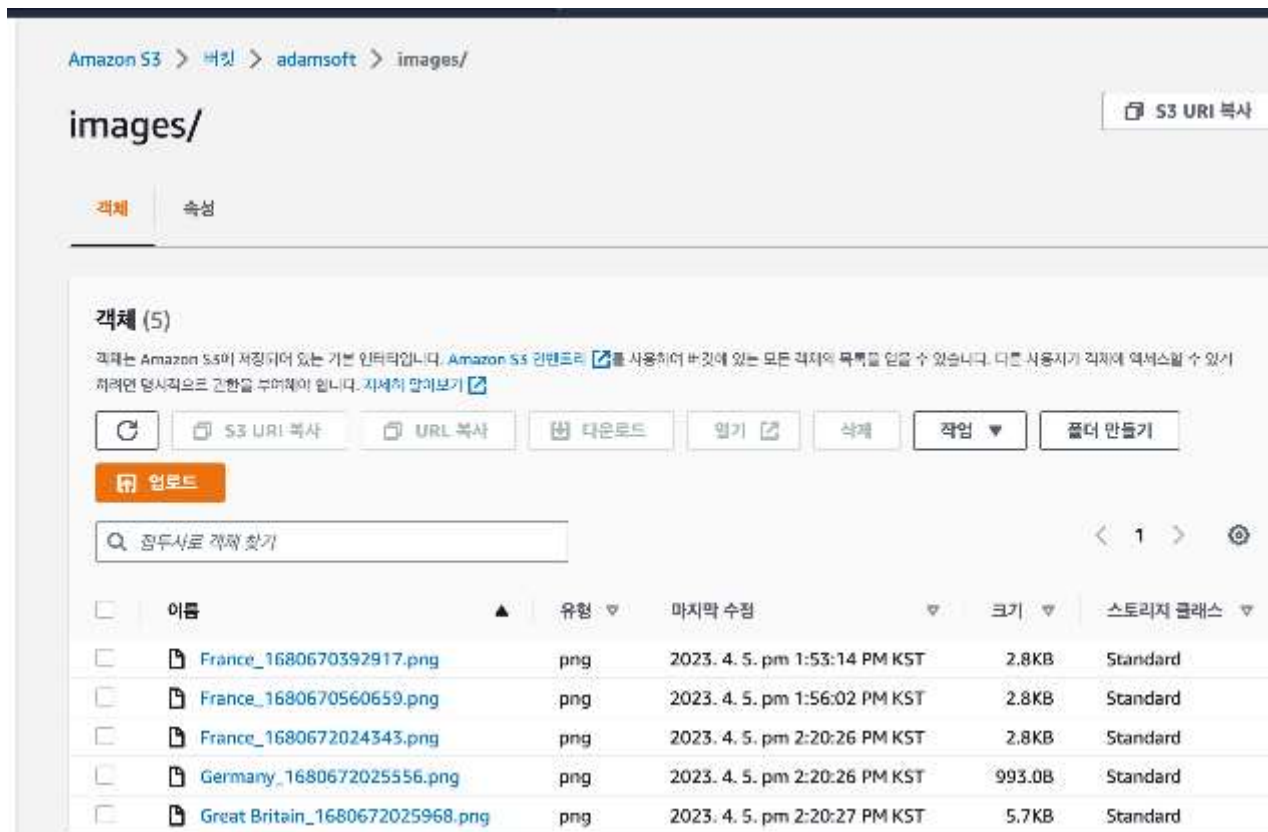
# S3

- ❖ Spring Boot 프로젝트
  - ✓ 다중 파일 업로드
    - 테스트



# S3

- ❖ Spring Boot 프로젝트
  - ✓ 다중 파일 업로드
    - 확인



# S3

- ❖ react 에서 파일 업로드
  - ✓ 프로젝트 생성: `npx create-react-app fileupload`
  - ✓ `cd fileupload`
  - ✓ 라이브러리 설치: `npm install aws-sdk`

# S3

## ❖ react 에서 파일 업로드

### ✓ App.js 파일에 코드 작성

```
import React,{useState} from 'react';
import AWS from 'aws-sdk'
```

```
const S3_BUCKET ='adamsoft';
const REGION ='us-east-1';
```

```
AWS.config.update({
 accessKeyId: 'AKIAWN5SS7IWEYL55SWG',
 secretAccessKey: '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'
})
```

```
const myBucket = new AWS.S3({
 params: { Bucket: S3_BUCKET},
 region: REGION,
})
```

# S3

## ❖ react 에서 파일 업로드

### ✓ App.js 파일에 코드 작성

```
function App() {
 const [progress , setProgress] = useState(0);
 const [selectedFile, setSelectedFile] = useState(null);

 const handleFileInput = (e) => {
 setSelectedFile(e.target.files[0]);
 }

 const uploadFile = (file) => {

 const params = {
 ACL: 'public-read',
 Body: file,
 Bucket: S3_BUCKET,
 Key: file.name
 };
 }
}
```

# S3

## ❖ react 에서 파일 업로드

### ✓ App.js 파일에 코드 작성

```
myBucket.putObject(params)
 .on('httpUploadProgress', (evt) => {
 setProgress(Math.round((evt.loaded / evt.total) * 100))
 })
 .send((err) => {
 if (err) console.log(err)
 })
}

return <div>
 <div>Native SDK File Upload Progress is {progress}%</div>
 <input type="file" onChange={handleFileInput}/>
 <button onClick={() => uploadFile(selectedFile)}> Upload to S3</button>
</div>
}

export default App;
```

# S3

- ❖ react 에서 파일 업로드
  - ✓ 프로젝트 실행 후 확인: npm start

# 정적 웹 사이트 CI/CD

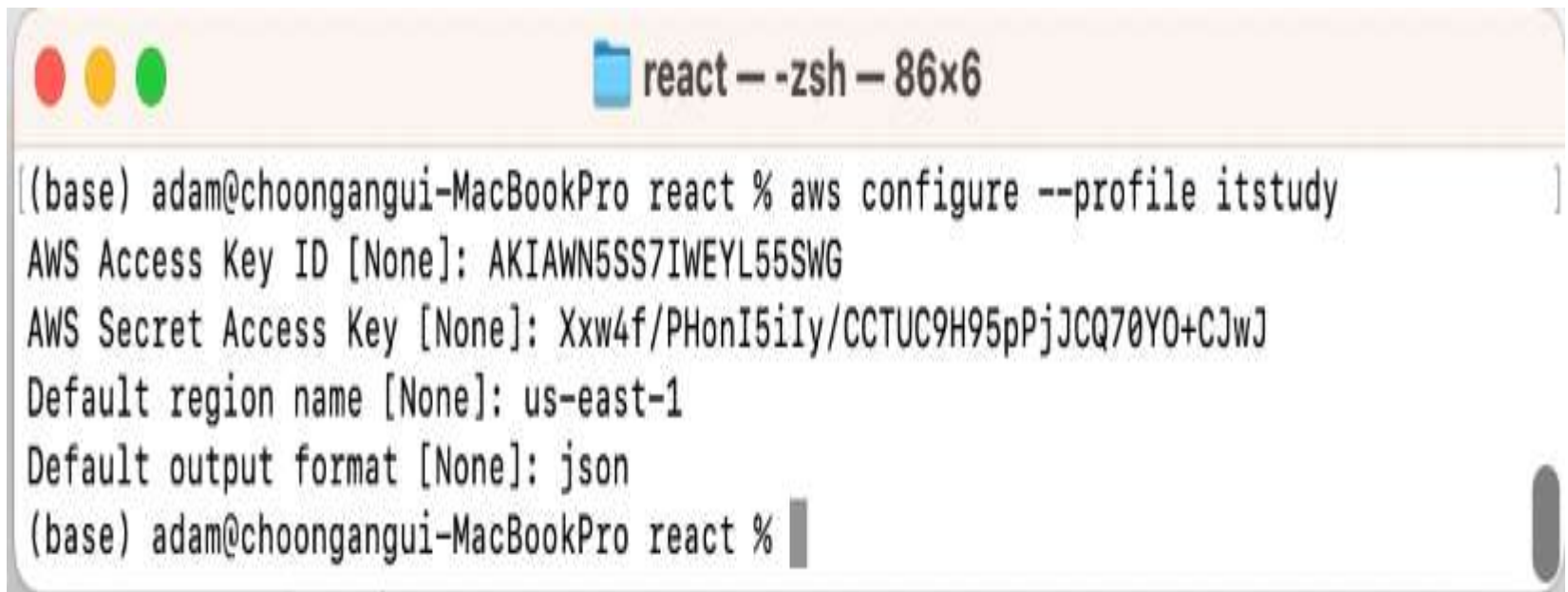
## ❖ 정적 웹 사이트 호스팅

- ✓ AWS Command Interface 설치

- [https://docs.aws.amazon.com/ko\\_kr/cli/latest/userguide/getting-started-install.html](https://docs.aws.amazon.com/ko_kr/cli/latest/userguide/getting-started-install.html)

- ✓ 확인: `aws --version`

- ✓ 유저 추가



```
react — zsh — 86x6
[(base) adam@choongangui-MacBookPro react % aws configure --profile itstudy]
AWS Access Key ID [None]: AKIAWN5SS7IWEYL55SWG
AWS Secret Access Key [None]: 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
Default region name [None]: us-east-1
Default output format [None]: json
(base) adam@choongangui-MacBookPro react %
```

# 정적 웹 사이트 CI/CD

## ❖ 정적 웹 사이트 호스팅

- ✓ S3 Bucket 설정 – 속성 탭에서 수정

### 정적 웹 사이트 호스팅 편집 Info

#### 정적 웹 사이트 호스팅

이 리소스를 사용하여 웹 사이트를 호스팅하거나 콘텐츠를 리디렉션합니다. 자세히 알아보기 [🔗](#)

정적 웹 사이트 호스팅

☐ 비활성화

☒ 활성화

호스팅 유형

☒ 정적 웹 사이트 호스팅

☐ 객체에 대한 요청 리디렉션

☒ 고객 웹 사이트 콘텐츠의 콘텐츠 액세스할 수 있게 하려면 모든 콘텐츠를 공개적으로 읽기 가능하도록 설정해야 합니다. 이렇게 하려면, 버킷에 대한 S3 퍼블릭 액세스 차단 설정을 편집하면 됩니다. 자세한 내용은 [Amazon S3 퍼블릭 액세스 차단 사용](#) [🔗](#) 참조하십시오.

인덱스 문서

웹 사이트의 홈 페이지 또는 기본 페이지를 지정합니다.

오류 문서 - 선택 사항

오류가 발생하면 반환됩니다.

리디렉션 규칙 - 선택 사항

JSON으로 작성된 리디렉션 규칙은 특정 콘텐츠에 대한 웹 페이지 요청을 자동으로 리디렉션합니다. 자세히 알아보기 [🔗](#)

# 정적 웹 사이트 CI/CD

## ❖ 정적 웹 사이트 호스팅

- ✓ react 프로젝트 생성 - `npx create-react-app staticweb`
- ✓ 프로젝트 빌드 - `npm run build`
- ✓ 업로드 - `aws s3 sync ./build s3://Bucket이름 --profile=계정명`
  - package.json 파일의 scripts에 추가하면 `yarn deploly` 나 `npm run deploy`로 배포 가능  
"deploy": "aws s3 sync ./build s3://Bucket이름 --profile=계정이름"
- ✓ S3의 엔드 포인트에 접속해서 확인

# 정적 웹 사이트 CI/CD

## ❖ 정적 웹 사이트 호스팅

### ✓ Git Hub Action 을 이용한 CI/CD 구현

- 프로젝트 내에 .github/workflows/?.yml

name: CI/CD hdev\_client to AWS S3

on:

push:

branches:

- main

# 정적 웹 사이트 CI/CD

## ❖ 정적 웹 사이트 호스팅

### ✓ Git Hub Action 을 이용한 CI/CD 구현

- 프로젝트 내에 .github/workflows/?.yml

jobs:

deploy:

runs-on: ubuntu-latest

steps:

- name: 코드 체크아웃

uses: actions/checkout@v3

- name: AWS IAM 사용자 설정

uses: aws-actions/configure-aws-credentials@v2

with:

aws-access-key-id: \${{ secrets.AWS\_ACCESS\_KEY\_ID }}

aws-secret-access-key: \${{ secrets.AWS\_SECRET\_ACCESS\_KEY }}

aws-region: \${{ secrets.AWS\_REGION }}

# 정적 웹 사이트 CI/CD

## ❖ 정적 웹 사이트 호스팅

### ✓ Git Hub Action 을 이용한 CI/CD 구현

#### ● 프로젝트 내에 ./github/workflows/?.yml

- name: 리액트 빌드

run: |

npm install

npm run build

- name: 빌드한 파일 S3에 업로드

run: aws s3 sync build/ s3://\${{ secrets.AWS\_S3\_BUCKET }} --acl public-read

env:

AWS\_ACCESS\_KEY\_ID: \${{ secrets.AWS\_ACCESS\_KEY\_ID }}

AWS\_SECRET\_ACCESS\_KEY: \${{ secrets.AWS\_SECRET\_ACCESS\_KEY }}

# 정적 웹 사이트 CI/CD

- ❖ 정적 웹 사이트 호스팅
  - ✓ Git Hub Action 을 이용한 CI/CD 구현
    - Git Hub 의 Settings 에 Secret Key 등록



# 정적 웹 사이트 CI/CD

- ❖ 정적 웹 사이트 호스팅
  - ✓ Git Hub Action 을 이용한 CI/CD 구현
    - Git Hub 의 Settings 에 Secret Key 등록

---

## Security

 Code security and analysis

 Deploy keys

 Secrets and variables 

Actions

Codespaces

Dependabot

---

# 정적 웹 사이트 CI/CD

- ❖ 정적 웹 사이트 호스팅
  - ✓ Git Hub Action 을 이용한 CI/CD 구현
    - Git Hub 의 Settings 에 Secret Key 등록

## Repository secrets

This repository has no secrets.

New repository secret

# 정적 웹 사이트 CI/CD

- ❖ 정적 웹 사이트 호스팅
  - ✓ Git Hub Action 을 이용한 CI/CD 구현
    - Git Hub 의 Settings 에 Secret Key 등록

## Actions secrets / New secret

Name \*

AWS\_S3\_BUCKET

Secret \*

itstudyfront

Add secret

# 정적 웹 사이트 CI/CD

- ❖ 정적 웹 사이트 호스팅
  - ✓ Git Hub Action 을 이용한 CI/CD 구현
    - Git Hub 의 Settings 에 Secret Key 등록

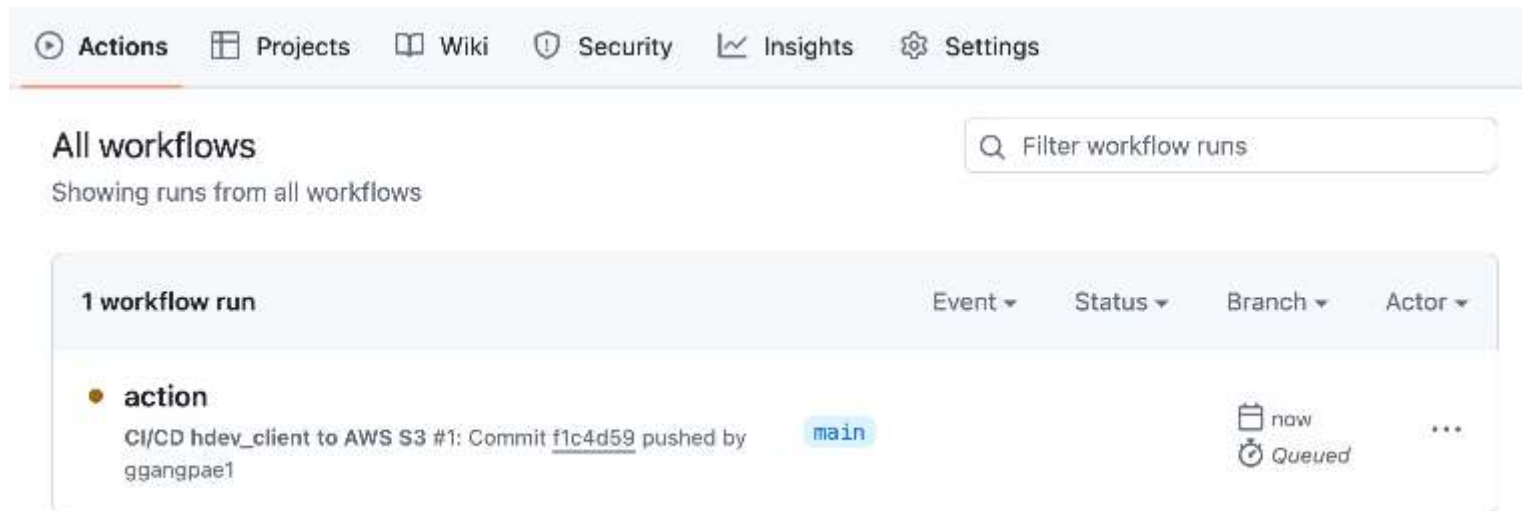
## Repository secrets

New repository secret

| Name ↕                                                                                                   | Last updated |                                                                                      |                                                                                      |
|----------------------------------------------------------------------------------------------------------|--------------|--------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------|
|  AWS_ACCESS_KEY_ID      | now          |   |   |
|  AWS_REGION             | now          |   |   |
|  AWS_S3_BUCKET          | 1 minute ago |   |   |
|  AWS_SECRET_ACCESS_KEY | now          |  |  |

# 정적 웹 사이트 CI/CD

- ❖ 정적 웹 사이트 호스팅
  - ✓ Git Hub Action 을 이용한 CI/CD 구현
    - 코드를 업로드하고 Git Hub에서 Action 확인



# 정적 웹 사이트 CI/CD

## ❖ 정적 웹 사이트 호스팅

### ✓ Git Hub Action 을 이용한 CI/CD 구현

#### ● ACL 문제가 발생하면 버킷의 객체 소유권 변경

**객체 소유권 편집** 정보

**객체 소유권**  
다른 AWS 계정에서 이 버킷에 작성한 객체의 소유권 및 액세스 제어 목록(ACL)의 사용을 재어립니다. 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 라이더가 업로드하는 데이터를 소유하도록 해야 하는 경우가 아니라면 **ACL을 비활성화**하는 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 선택**  
이 버킷에 작성된 새 객체가 bucket-owner-full-control 삽입 ACL을 지정하는 경우 새 객체는 버킷 소유자가 소유합니다. 그렇지 않은 경우 객체 라이더가 소유합니다.

☐ **객체 라이더**  
객체 라이더는 객체 소유자로 유지됩니다.

① 새 객체에 대해서만 객체 소유권을 적용하려면 버킷 정책이 객체 업로드에 bucket-owner-full-control 삽입 ACL을 요구하도록 지정해야 합니다. [자세히 알아보기](#)

# 정적 웹 사이트 CI/CD

## ❖ 정적 웹 사이트 호스팅

### ✓ Git Hub Action 을 이용한 CI/CD 구현

- app.js 코드를 수정하고 다시 업로드하고 반영되었는지 확인

```
import React, {Fragment} from 'react';
```

```
function App() {
 return (
 <Fragment>
 <h1>70-79뉴프렌즈</h1>
 </Fragment>
);
}
export default App;
```

# 정적 웹 사이트 CI/CD

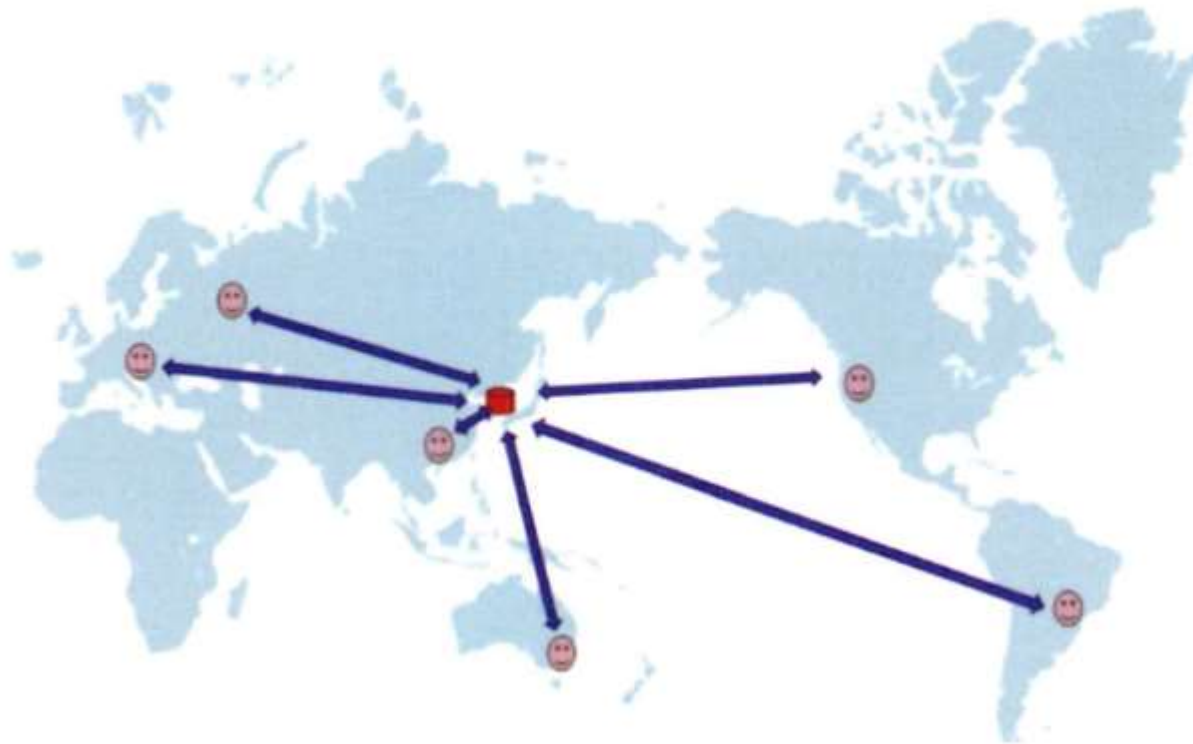
## ❖ 정적 웹 사이트 호스팅

### ✓ CloudFront

- 웹 페이지 호스트 와 네트워크 서버 설정을 위한 서비스
- CDN
  - ◆ 콘텐츠 전송 네트워크(Content Delivery Network)는 리소스 요청이 발생했을 때 사용자가 현재 어디에서 요청하는지에 따라 리소스를 전달해주는 분산 네트워크 시스템(Distributed Network System)
  - ◆ CDN의 목적은 Edge Location을 통해 사용자에게 최소한의 지연 시간과 최고의 성능으로 콘텐츠를 보여주기 위함
    - 최초로 웹사이트 호스팅이 이루어지고 있는 곳을 Origin이라고 하는데 이 웹사이트는 전 세계에서 다양한 사용자가 방문
    - 어떤 지역에서 사용자가 웹사이트 요청을 하는 경우 Origin에서 사용자 요청에 응답하여 콘텐츠를 전달하면 문제가 발생할 수 있음
    - 사용자가 Origin Server에서 멀어진다면 콘텐츠를 받는 데 지연 시간(latency)이 발생
    - 많은 사용자가 동시 다발적으로 요청하게 된다면 지연 시간은 더 길어지는데 이런 문제를 해결하기 위해서 CDN이 등장

# S3

- ❖ 정적 웹 사이트 호스팅
  - ✓ CloudFront
    - CDN



# S3

## ❖ 정적 웹 사이트 호스팅

### ✓ CloudFront

#### ● CDN

- ◆ CDN은 많은 요청이 오가는 지역을 근거로 Edge Location을 생성
- ◆ Edge Location은 Origin Server에서 가지고 있는 콘텐츠 복사본을 가지고 있으며 콘텐츠 요청을 전달받을 때 Origin Server에서 콘텐츠를 전달하는 것이 아니라 사용자와 가장 가까운 Edge Location을 통해 콘텐츠를 전송



# S3

## ❖ 정적 웹 사이트 호스팅

### ✓ CloudFront

#### ● Edge Location

- ◆ Edge Location은 CloudFront CDN에 의해 전 세계 곳곳에 생성
- ◆ 각각의 Edge Location은 전 세계 사용자에게 최소한의 지연 시간을 발생시키기 위해 존재
- ◆ CloudFront에 Contents Server가 Hosting 되면 사용자의 요청을 받을 시 사용자가 위치한 가장 가까운 Edge Location을 참조하여 콘텐츠를 전달
- ◆ Edge Location은 Origin Server에 있는 콘텐츠 원본을 캐시에 보관
- ◆ 사용자가 웹사이트를 요청할 시 캐시에 요청 정보가 있지 않다면 일차적으로 Origin Server에서 콘텐츠를 가져온 후 캐시에 보관하여 사용자에게 전달
- ◆ 추후 사용자가 똑같은 요청을 할 시 Origin Server가 아닌 Cache로부터 콘텐츠를 전달받기 때문에 속도가 빠른 장점을 가지고 있음

# S3

## ❖ 정적 웹 사이트 호스팅

### ✓ CloudFront

#### ● Origin Server

- ◆ 최초 웹 서버가 구현되고 돌아가고 있는 곳이 Origin Server
- ◆ 웹 페이지를 꾸미기 위해 필요한 HTML, CSS, JavaScript 파일이 있는 S3 Bucket이나 EC2 인스턴스가 Origin Server
- ◆ CloudFront는 Origin Server를 통해 콘텐츠를 전달받고 사용자에게 전달.
- ◆ S3 Bucket 단독으로 웹사이트 호스팅을 하게 되면 정적 웹사이트 호스팅만 가능하지만 CloudFront를 사용하여 동적 웹사이트 호스팅을 구현할 수 있음

# S3

## ❖ 정적 웹 사이트 호스팅

### ✓ CloudFront

#### ● 장점

- ◆ Edge Location을 만들고 캐시에 원본 콘텐츠를 담고 있기 때문에 사용자에게 콘텐츠를 전달할 때 처리량이 줄고 지연 시간이 거의 없음
- ◆ http 와 https를 모두 지원하기 때문에 보안이 뛰어남
- ◆ 데이터를 전송할 때만 비용이 청구되기 때문에 웹사이트 호스팅 시 저렴

# S3

## ❖ 정적 웹 사이트 호스팅

### ✓ CloudFront

#### ● 설정 - Cloud Front 배포 생성을 선택

The screenshot shows the Amazon CloudFront console interface. On the left is a navigation sidebar with options like 'Overview', 'Distributions', 'Buckets', 'Logging', 'Cache Policies', 'Origins', 'Invalidations', 'Metrics', 'Tags', and 'Account Settings'. The main content area features the 'Amazon CloudFront' logo and the tagline '짧은 대기 시간과 빠른 전송 속도로 콘텐츠를 안전하게 제공' (Provide content securely with short wait times and fast transfer speeds). Below this, there's a section titled '이점 및 기능' (Benefits and Features) with three columns: '대기 시간 감소' (Reduce latency), '본인 가산' (Self-managed), and '비용 절감' (Cost savings). To the right of the main content are three informational cards: 'CloudFront 시작하기' (Get started with CloudFront), 'AWS 프리 티어' (AWS Free Tier), and '요금(미국)' (Pricing (US)).

**Amazon CloudFront**  
짧은 대기 시간과 빠른 전송 속도로 콘텐츠를 안전하게 제공

Amazon CloudFront는 웹 사이트 대기 시간을 줄이는 빠른 전송 속도를 제공하는 콘텐츠 전송 네트워크(CDN) 서비스입니다.

**이점 및 기능**

- 대기 시간 감소**  
CloudFront는 전 세계에 분포된 225개 이상의 엣지 위치를 통해 콘텐츠를 전달합니다. 이는 100GB에 달하는 월간 트래픽을 처리할 수 있는 POP과 엣지 캐시를 통해 콘텐츠를 전달합니다. CloudFront는 캐시된 콘텐츠를 전달하는 데 더 빠른 전송 속도를 제공합니다. 또한, 전송 지연을 줄여 콘텐츠를 더 빠르게 전달할 수 있습니다.
- 본인 가산**  
자체 관리, 보안, 모니터링 및 업데이트를 위한 CloudFront를 사용합니다. AWS Shield Standard는 추가 비용 없이 DDoS 공격으로부터 CloudFront를 보호해 주며, AWS IAM을 사용하여 액세스 권한을 관리할 수 있습니다.
- 비용 절감**  
CloudFront를 사용하면 AWS의 다른 서비스와 통합하여 비용을 절감할 수 있습니다. CloudFront는 AWS의 다른 서비스와 통합하여 비용을 절감할 수 있습니다.

**CloudFront 시작하기**  
5분 이내에 Amazon S3 버킷, Application Load Balancer, Amazon API Gateway API 등에 대해 CloudFront 배포를 생성하여 콘텐츠를 안전하게 제공할 수 있습니다.

**AWS 프리 티어**  
1TB의 데이터 전송  
10,000,000건의 HTTP 또는 HTTPS 요청  
2,000,000건의 CloudFront 객체 조회  
다양한 기타 무료

**요금(미국)**  
1GB 이하 1TB 초과 데이터 전송 요금  
10MB/월 USD 0.005/GB  
100MB 이하 100MB 초과 데이터 전송 요금

# S3

## ❖ 정적 웹 사이트 호스팅

### ✓ CloudFront

#### ● 설정 - Cloud Front 배포 생성을 선택

##### ◆ 원본 도메인 설정

- 원본 도메인: 오리진을 어떤 것을 선택할 지 설정
- 원본 경로: 하위 디렉토리가 존재하는 경우 선택
- Bucket을 선택하게 되면 OAI 옵션을 설정할 수 있음

The screenshot shows the 'Create Distribution' page in the AWS CloudFront console. The 'Origin' section is expanded, showing the following configuration:

- Origin Name:** A text input field containing 'itstudyfront.s3.amazonaws.com'.
- Protocol:** A dropdown menu with 'HTTP and HTTPS' selected.
- HTTP Port:** A text input field containing '80'.
- HTTPS Port:** A text input field containing '443'.
- SSL Protocol:** A dropdown menu with 'TLSv1.2' selected.

# S3

## ❖ 정적 웹 사이트 호스팅

### ✓ CloudFront

#### ● 설정 - Cloud Front 배포 생성을 선택

#### ◆ 뷰어 프로토콜 정책 수정

**기본 캐시 동작**

경로 떠난 [정보](#)

기본값(\*)

자동으로 각체 압축 [정보](#)

☐ No

☒ Yes

**뷰어**

뷰어 프로토콜 정책

☐ HTTP and HTTPS

☒ Redirect HTTP to HTTPS

☐ HTTPS only

허용된 HTTP 방법

☒ GET, HEAD

☐ GET, HEAD, OPTIONS

☐ GET, HEAD, OPTIONS, PUT, POST, PATCH, DELETE

뷰어 액세스 제한

뷰어 액세스를 제한하는 경우 뷰어는 CloudFront 저장된 URL 또는 서명한 쿼리를 사용하여 사용자의 본인에게 액세스해야 합니다.

☒ No

☐ Yes

캐시 키 및 원본 요청

캐시 정책 및 원본 요청 정책을 사용하여 캐시 키 및 원본 요청을 제어할 것을 권장합니다.

# S3

## ❖ 정적 웹 사이트 호스팅

### ✓ CloudFront

#### ● 설정 - Cloud Front 배포 생성을 선택

##### ◆ 캐시 정책 설정

- 처음 등장하는 경로 패턴은 수정할 수 없음
- 자동으로 객체 압축은 콘텐츠를 오리진에서 전송 시 압축해서 보낼 지에 대한 여부를 정의
- 뷰어 프로토콜 정책에는 HTTP와 HTTPS를 모두 허용할지 HTTP를 HTTPS로 재 전송할지 HTTPS만 허용할지 총 세 가지 선택 사항이 있음
- 일반적으로 HTTPS가 HTTP보다 보안이 훨씬 뛰어난데 SSL(Secure Socket Layer) 인증서를 통해 서버에서 브라우저로 정보가 안전하게 전달될 수 있음
- 허용된 HTTP 방법을 설정

# S3

## ❖ 정적 웹 사이트 호스팅

### ✓ CloudFront

#### ● 설정 - Cloud Front 배포 생성을 선택

#### ◆ 방화벽 과 가격 분류 설정

### 웹 애플리케이션 방화벽(WAF) 정보

☐ 보안 보호 활성화  
AWS WAF를 사용하여 가장 일반적인 웹 위협과 보안 취약성으로부터 애플리케이션을 안전하게 보호하세요. 차단된 요청은 웹 서버에 도달하기 전에 걸러집니다.

☒ 보안 보호 비활성화  
애플리케이션에 AWS WAF의 보안 보호 기능이 필요하지 않은 경우 이 옵션을 선택하세요.

### 설정

#### 가격 분류 정보

차단하려는 광고와 연관된 가격 분류를 선택합니다.

☐ 모든 엣지 로케이션에서 사용(최고의 성능)

☐ 북미 및 유럽만 사용

☒ 북미, 유럽, 아시아, 중동 및 아프리카에서 사용

#### 대체 도메인 이름(CNAME) - 선택 사항

이 리소스에서 제공하는 페이지에 대체 URL에서 사용하는 사용자 정의 도메인 이름을 추가합니다.

항목 추가

③ 대체 도메인 이름 목록을 추가하려면 **대량 편집기**를 사용하십시오.

#### 사용자 정의 SSL 인증서 - 선택 사항

AWS Certificate Manager의 인증서를 연결합니다. 인증서는 반드시 미국 북부(미주) 및 북부 리전(us-east-1)에 있어야 합니다.

인증서 선택

인증서 요청

[인증서 요청](#)

# S3

## ❖ 정적 웹 사이트 호스팅

### ✓ CloudFront

#### ● 설정 - Cloud Front 배포 생성을 선택

#### ◆ 기본값 루트 객체 설정

##### 사용자 정의 SSL 인증서 - 선택 사항

AWS Certificate Manager가 인증서를 인증합니다. 인증서는 반드시 미국 동부(버지니아 북부) 리전(us-east-1)에 있어야 합니다.

인증서 선택

인증서 요청

##### 지원되는 HTTP 버전

추가 HTTP 버전에 대해 지원은 추가됩니다. HTTP/1.0 및 HTTP/1.1이 기본적으로 지원됩니다.

☒ HTTP/2

☐ HTTP/3

##### 기본값 루트 객체 - 선택 사항

본래가 특정 객체 대신 루트 URL(/)을 요청할 때 반환할 객체(파일 이름)입니다.

index.html

##### 표준 로깅

Amazon S3 버킷으로 전송된 쿼리 로그를 가져옵니다.

☒ 끄기

☐ 켜기

##### IPv6

☐ 끄기

☒ 켜기

##### 실명 - 선택 사항

# S3

## ❖ 정적 웹 사이트 호스팅

### ✓ CloudFront

#### ● 확인: CloudFront의 도메인 확인

[일반](#) | [보안](#) | [원본](#) | [동작](#) | [오류 페이지](#) | [무효화](#) | [태그](#)

### 세부 정보

|                                            |                                                                   |                                          |
|--------------------------------------------|-------------------------------------------------------------------|------------------------------------------|
| 배포 도메인 이름<br>d20ja17c8s9c1x.cloudfront.net | ARN<br>arn:aws:cloudfront:641022061021:distribution/EKXSX6M3D48SY | 마지막 수정<br>2023년 12월 4일 오전 9시 11분 42초 UTC |
|--------------------------------------------|-------------------------------------------------------------------|------------------------------------------|

### 설정

원점

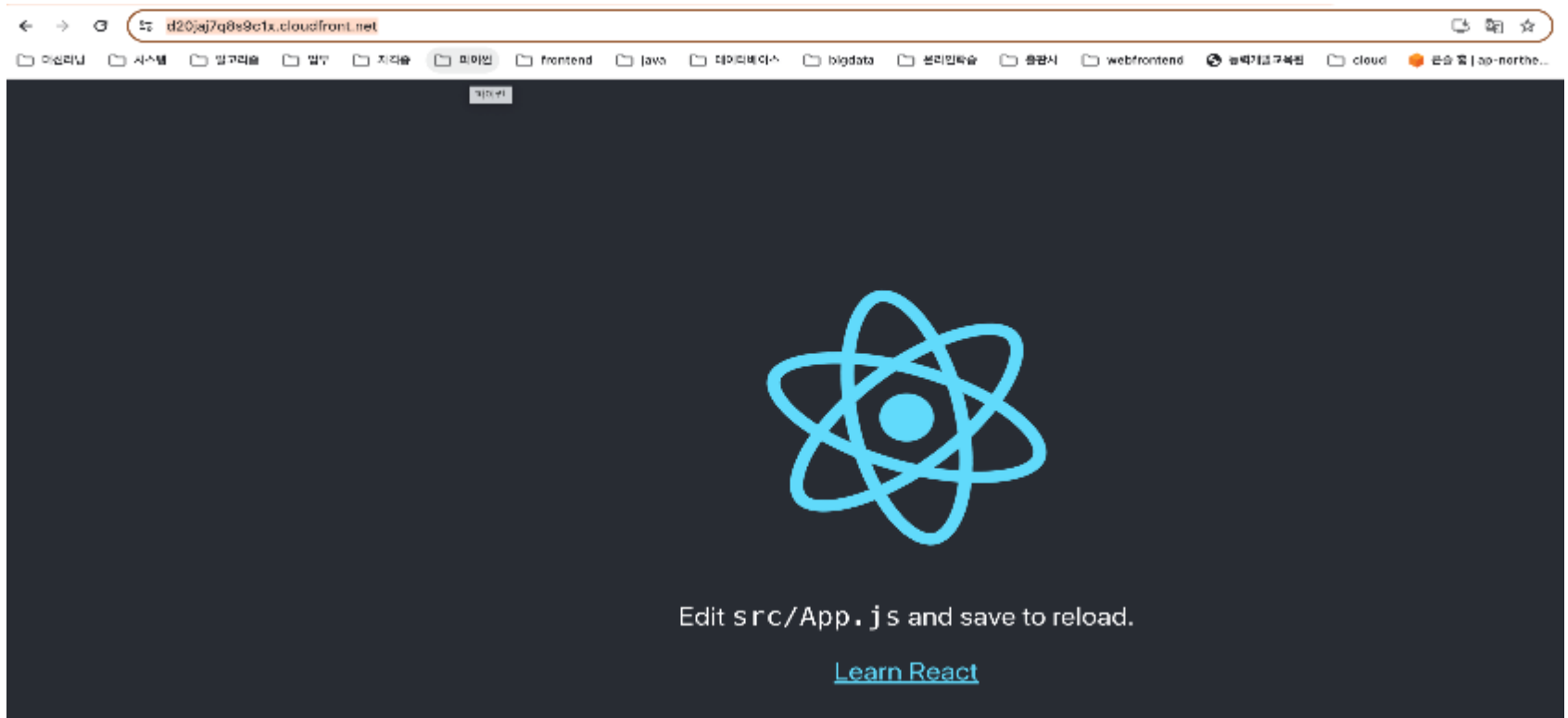
|                                                                                               |                |                                                      |
|-----------------------------------------------------------------------------------------------|----------------|------------------------------------------------------|
| 설명<br>-<br>가격 분구<br>북미, 유럽, 아시아, 중동 및 아프리카에서 사용<br>지원되는 HTTP 버전<br>HTTP/2, HTTP/1.1, HTTP/1.0 | 대체 도메인 이름<br>- | 표준 로깅<br>끄기<br>쿠키 로깅<br>끄기<br>기본 부트 객체<br>index.html |
|-----------------------------------------------------------------------------------------------|----------------|------------------------------------------------------|

### Continuous deployment 정보

Create staging distribution

# S3

- ❖ 정적 웹 사이트 호스팅
  - ✓ CloudFront
    - 확인: 브라우저에서 확인



# S3

## ❖ 정적 웹 사이트 호스팅

### ✓ Git Hub Action 을 이용한 CI/CD 구현

#### ● 프로젝트 내에 ./github/workflows/?.yml

- name: CloudFront 캐시 무력화 설정

uses: chetan/invalidate-cloudfront-action@v2

env:

DISTRIBUTION: \${{ secrets.AWS\_CLOUDFRONT\_ID }}

PATHS: "/\*"

AWS\_REGION: us-east-1

AWS\_ACCESS\_KEY\_ID: \${{ secrets.AWS\_ACCESS\_KEY\_ID }}

AWS\_SECRET\_ACCESS\_KEY: \${{ secrets.AWS\_SECRET\_ACCESS\_KEY }}

# S3

## ❖ 정적 웹 사이트 호스팅

### ✓ Git Hub Action 을 이용한 CICD 구현

- AWS\_CLOUDFRONT\_ID 키를 추가하고 Cloud Front ID 추가
- IAM 사용자에게 CloudFront 권한 추가

# S3

- ❖ 정적 웹 사이트 호스팅
  - ✓ https 인증서 생성
    - AWS Certificate Manager 서비스에 접속해서 인증서 요청을 눌러서 인증서 생성



# S3

## ❖ 정적 웹 사이트 호스팅

### ✓ https 인증서 생성

- AWS Certificate Manager 서비스에 접속해서 인증서 요청을 눌러서 인증서 생성



# S3

- ❖ 정적 웹 사이트 호스팅
  - ✓ https 인증서 생성
    - AWS Certificate Manager 서비스에 접속해서 인증서 요청을 눌러서 인증서 생성

AWS Certificate Manager > 인증서 > 인증서 요청 > 퍼블릭 인증서 요청

## 퍼블릭 인증서 요청

**도메인 이름**  
인증서에 대해 하나 이상의 도메인 이름을 지명합니다.

완전히 정규화된 도메인 이름 [정보](#)

[이 인증서에 다른 이름 추가](#)

이 인증서에 이름을 추가할 수 있습니다. 예를 들어, 'www.example.com'의 도메인 인증서를 요청하는 경우 고객이 두 이름 중 하나를 사이트와 합칠 수 있도록 'example.com'이라는 이름을 추가할 수 있습니다.

**검증 방법** [정보](#)  
도메인 소유권을 증명하기 위한 방법 선택

☒ **DNS 검증 - 권장**  
인증서 요청에서 도메인에 대한 DNS 구성을 수정할 권한이 있는 권위 이 접근을 선택합니다.

☐ **이메일 검증**  
인증서 요청에서 도메인에 대한 DNS 구성을 수정할 권한을 소유하지 않거나 획득할 수 없는 경우 이 옵션을 선택합니다.

# S3

## ❖ 정적 웹 사이트 호스팅

### ✓ https 인증서 생성

- AWS Certificate Manager 서비스에서 인증서를 선택하고 인증서 검증 – Route53에서 레코드 생성 클릭

The screenshot shows the AWS Certificate Manager console for a specific certificate. The certificate is in the 'Issued' state and is associated with the domain 'front.shinhanjason.com'. The console displays the certificate's ARN and provides a link to generate Route 53 records. Below this, a table lists the domain records, including the domain name, status, type (CNAME), and the CNAME value.

**36d08e58-ad25-4ab4-a49c-7dd8d40b8c41**

**인증서 상태**

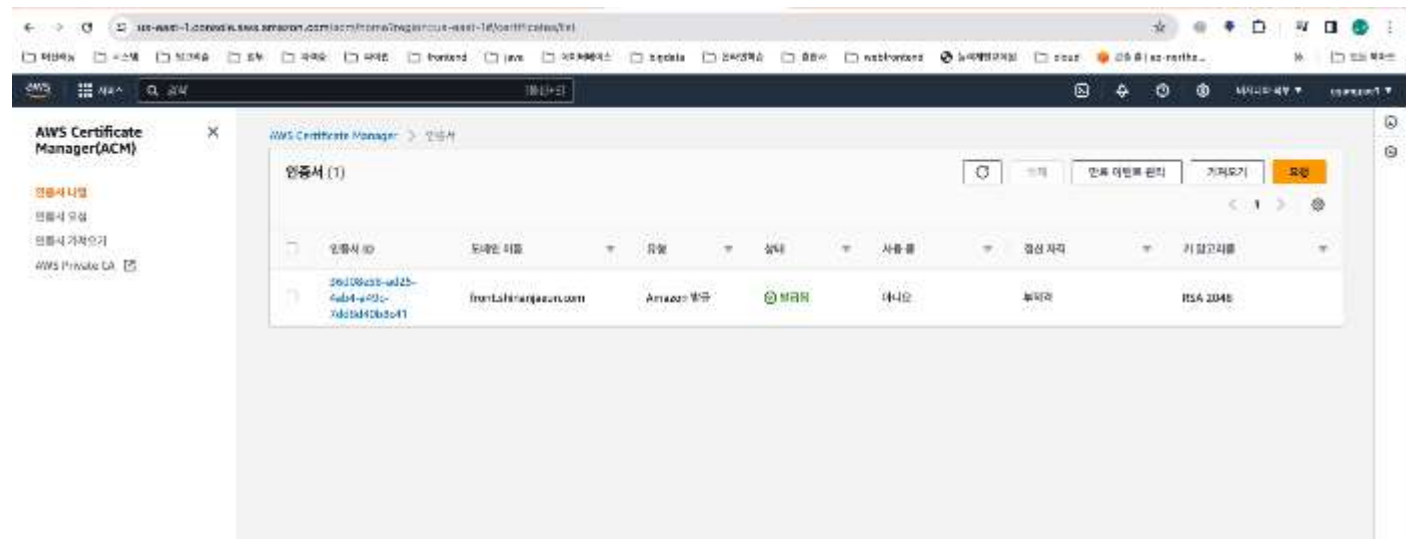
식별자: 36d08e58-ad25-4ab4-a49c-7dd8d40b8c41  
상태: [검증 대기 중](#) [정보](#)  
ARN: [arn:aws:acm:us-east-1:541022061021:certificate/36d08e58-ad25-4ab4-a49c-7dd8d40b8c41](#)  
유형: Amazon [발급](#)

**도메인 (1)** [Route 53에서 레코드 생성](#) [CSV로 내보내기](#)

| 도메인                    | 상태                      | 경선 상태 | 유형    | CNAME 이름                                                         | CNAME 값                                                          |
|------------------------|-------------------------|-------|-------|------------------------------------------------------------------|------------------------------------------------------------------|
| front.shinhanjason.com | <a href="#">검증 대기 중</a> | -     | CNAME | _f11b778af62a81ab74de58ba9ed48cd4.mhlotbpdnt.acm-validations.aws | 0877b778af62a81ab74de58ba9ed48cd4.mhlotbpdnt.acm-validations.aws |

# S3

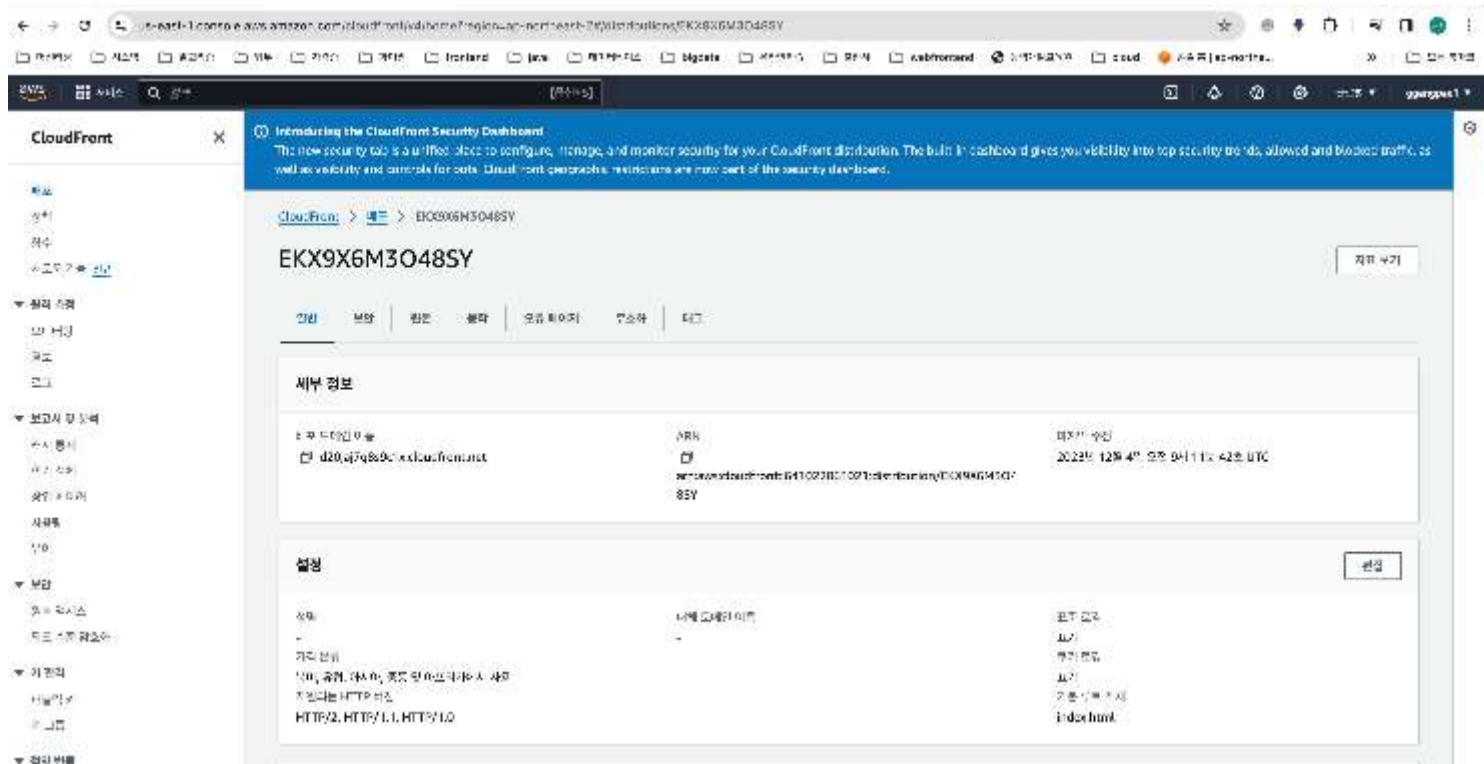
- ❖ 정적 웹 사이트 호스팅
  - ✓ https 인증서 생성
    - 인증서 생성 확인



# S3

## ❖ 정적 웹 사이트 호스팅

- ✓ https 인증서 와 Cloud Front 의 배포 연결
  - Cloud Front에서 배포된 곳에 접속해서 [일반] - [설정] - [편집] 클릭



# S3

## ❖ 정적 웹 사이트 호스팅

### ✓ https 인증서 와 Cloud Front 의 배포 연결

- Cloud Front에서 배포된 곳에 접속해서 [설정] - [편집]

## 설정 편집

### 설정

#### 가격 분류 [정보](#)

지불하려는 최고가와 연관된 가격 분류를 선택합니다.

- ☐ 모든 엣지 로케이션에서 사용(최고의 성능)
- ☐ 북미 및 유럽만 사용
- ☒ 북미, 유럽, 아시아, 중동 및 아프리카에서 사용

#### 대체 도메인 이름(CNAME) - 선택 사항

이 배포에서 제공하는 파일에 대해 URL에서 사용하는 사용자 정의 도메인 이름을 추가합니다.

① 대체 도메인 이름 목록을 추가하려면 [대리 편집기](#)(들) 사용하십시오.

#### 사용자 정의 SSL 인증서 - 선택 사항

AWS Certificate Manager의 인증서를 연결합니다. 인증서는 반드시 미국 동부(버지니아 북부) 리전(us-east-1)에 있어야 합니다.



☒ [front.shinanjaeun.com](#) [인증서 요청](#)

# S3

## ❖ 정적 웹 사이트 호스팅

- ✓ Route53에서 도메인 과 Cloud Front 연결

The screenshot shows the AWS Route 53 console. On the left is a navigation menu with options like '다시보기', '호스팅 영역', '상태 검사', 'IP 기반 라우팅', 'CIDR 블록', '트래픽 흐름', '트래픽 정책', '정책 레코드', '도메인', '등록된 도메인', '요청', and '확인자'. The main panel is titled '호스팅 영역 (2)' and contains a table of hosted zones.

호스팅 영역 (2)  
Automatic 모드는 확실히 주어진 결과에 호스팅될 문서 검색을 제공합니다. 모드를 변경하려면 설정(settings)으로 이동합니다.

새로 생성된 호스팅 영역 생성

속성 또는 값을 기준으로 레코드 필터링

| 호스팅 영역 이름       | 유형  | 생성자      | 레코드 수 | 설명               |
|-----------------|-----|----------|-------|------------------|
| itstudy-ecs.com | 퍼블릭 | Route 53 | 3     |                  |
| shinanjaeun.com | 퍼블릭 | Route 53 | 5     | HostedZone cr... |

# S3

## ❖ 정적 웹 사이트 호스팅

- ✓ Route53에서 도메인 과 Cloud Front 연결

The screenshot shows the AWS Route 53 console for the domain **shinanjeun.com**. The left sidebar contains a navigation menu with the following items: 나시보드, 호스팅 영역, 상단 검사, IP 기반 라우팅, CloudFront, 트래픽 흐름, 프라이빗 DNS, 검색 레코드, 도메인, 등록된 도메인, 기록, 확인자, VPC, 인터넷 게이트웨이, 다중비율의 엔드포인트, 고객, 관리 구성, Outposts, DNS 변경, and 애플리케이션 라우팅. The main panel displays the '레코드 (5)' (Records) tab, which shows a table of DNS records for the domain. The table has columns for Record Name, Type, Weight, Priority, Alias, Value, TTL, and Status. The records are as follows:

| 레코드 이름         | 유형    | 가중치 | 우선순위 | 대체  | 값/프라이빗 라우팅 대상                                                                              | TTL(초) | 상태 |
|----------------|-------|-----|------|-----|--------------------------------------------------------------------------------------------|--------|----|
| shinanjeun.com | NS    | 단순  | -    | 아니오 | ns-1838.awsdns-37.co.uk, ns-1034.awsdns-01.org, ns-324.awsdns-01.net, ns-325.awsdns-40.com | 172800 | -  |
| shinanjeun.com | SOA   | 단순  | -    | 아니오 | ns-1838.awsdns-37.co.uk a...                                                               | 900    | -  |
| api.shinaje... | A     | 단순  | -    | 예   | duelstack-itstudy-server-elb...                                                            | -      | -  |
| ad7a46a52c...  | CNAME | 단순  | -    | 아니오 | ad7a46a52c5a52a0f3b4...                                                                    | 300    | -  |
| 11b17a50a66... | CNAME | 단순  | -    | 아니오 | 0677b778a62a81ab74ce58...                                                                  | 300    | -  |

# S3

## ❖ 정적 웹 사이트 호스팅

- ✓ Route53에서 도메인 과 Cloud Front 연결

Route 53 > 호스팅 영역 > shinanjeun.com > 레코드 생성

### 레코드 생성 정보

빠른 레코드 생성 마법사로 전환

▼ 레코드 1 삭제

레코드 이름 정보

front .shinanjeun.com

루트 도메인에 대한 레코드를 생성하려면 비워 둡니다.

레코드 유형 정보

A - IPv4 주소 및 일부 AWS 리소스로 트래픽 라우팅

☒ 년칭

드래픽 라우팅 대상 정보

CloudFront 배포에 대한 년칭

미국 동부(버지니아 북부)

CloudFront 배포 및 동일한 호스팅 영역의 다른 레코드가 내하 번칭은 전역적 번칭이며 미국 동부(버지니아 북부)에서만 사용할 수 있습니다.

라우팅 정책 정보

단순 라우팅

대상 상태 평가

☐ 아니요

다른 레코드 추가

취소

레코드 생성