AWS S3 파일업로드

멋쟁이사자처럼 8기 운영진 이경연

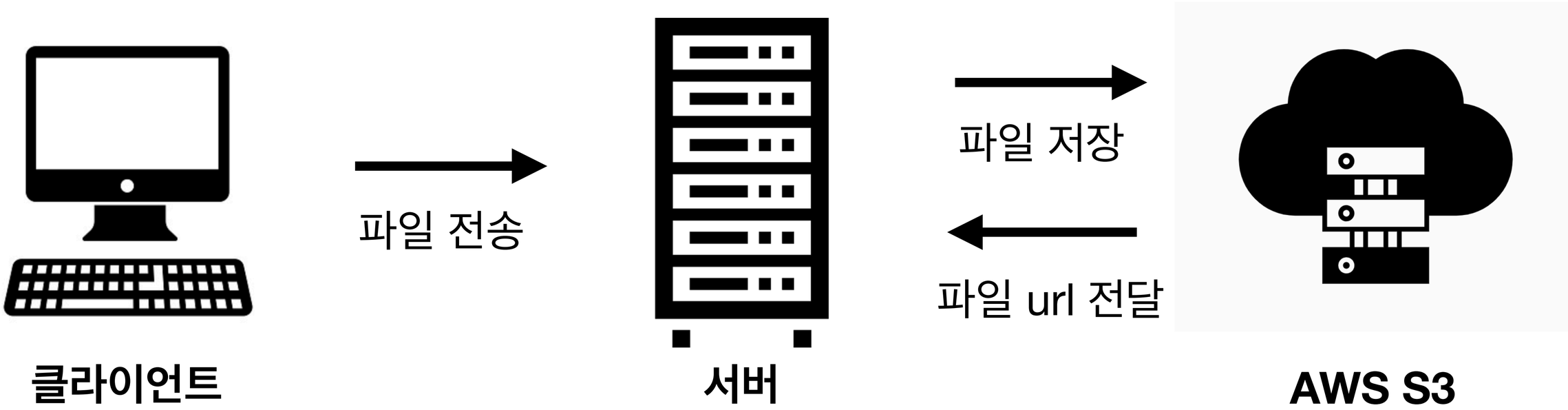
오늘은

- AWS S3에 파일 업로드

저장소를 따로 두는 이유

- 파일(이미지, 문서 등)은 db에 직접 저장하기에는 파일 크기가 너무 큼
- 다른 데이터들(이름, 성별 등)과는 달리 정보 자체를 다룰 일이 거의 없으므로 외부 저장소에 저장한다.

파일 업로드



id	이미지 url	이름	나이
1	https://aws.s3/kwon.jpg	권중훈	25
2	https://aws.s3/cc6656.jpg	박지환	25

프로젝트 생성

- 프로젝트 폴더로 이동
- pipenv shell
- pipenv install django
- django-admin startproject file_upload
- cd file_upload
- django-admin startapp uploader
- settings.py INSTALLED_APPS 에 uploader 추가

django_storages 패키지

- 장고에는 내장된 Storage Class가 있음
- 파일을 다루는데 필요한 기본적 메소드들(open, save, delete 등)이 정의되어있고, Storage CLASS를 상속받아 메소드를 재정의해서 사용한다
- 기본적으로 장고는 FileSystemStorage Class가 Storage Class를 상속받아 로컬 저장소(하드드라이브)에 저장하는 기능을 가지고 있다
- 하지만 우리는 로컬이 아닌 S3에 저장할 것이다

django_storages 패키지

- pipenv shell
- pipenv install django_storages boto3
- settings.py INSTALLED_APPS 에 storages 추가

AWS란?

- https://www.youtube.com/watch?v=NwqoVVKfO8A

- 클라우드 컴퓨팅을 통해 유동적으로 컴퓨팅 리소스(서버 컴퓨터, 저장소, 데이터베이스 등)를 제공하는 서비스

S3란?

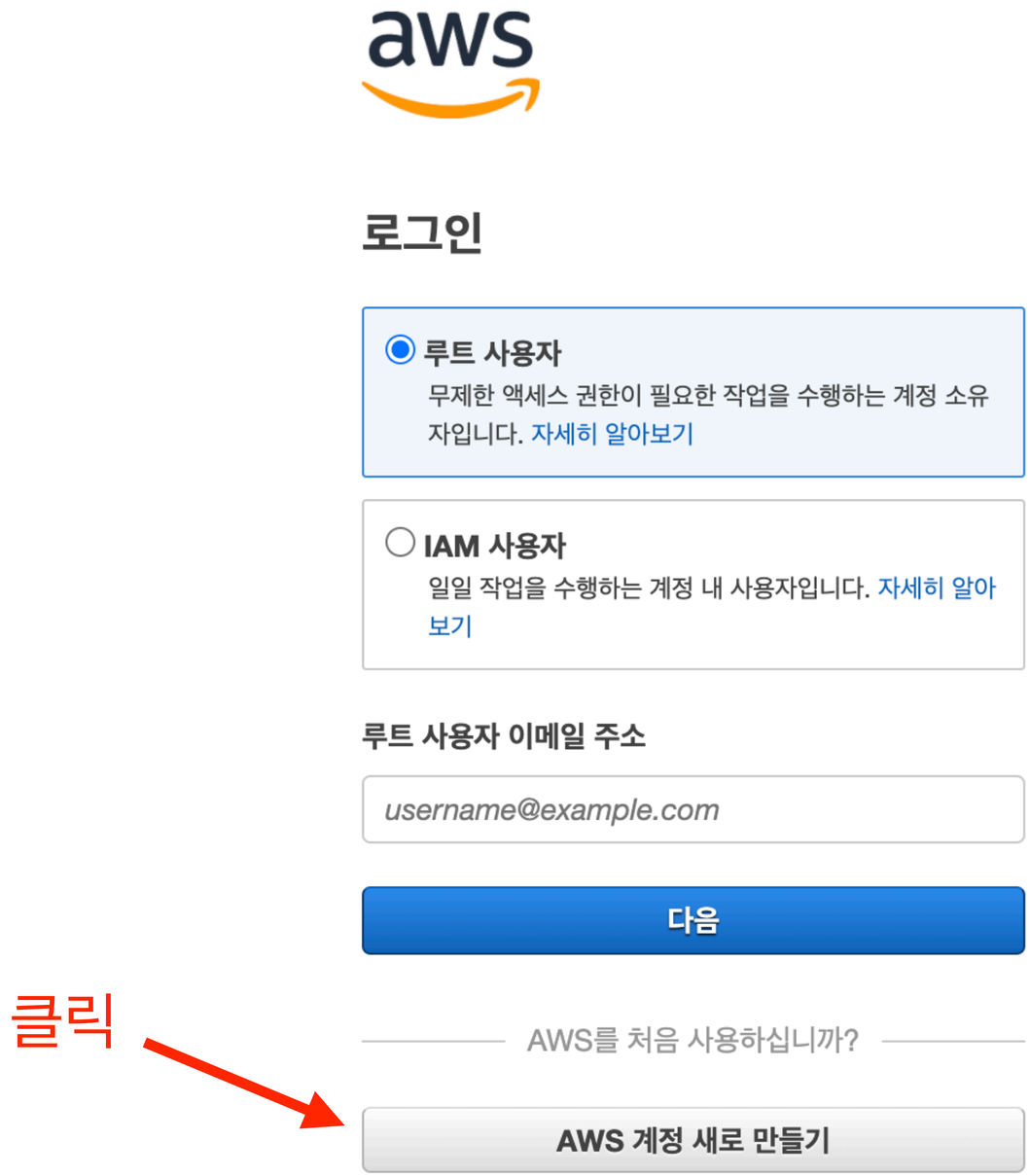
- Simple Storage Service : S3

- 클라우드 스토리지 제공 서비스

- https://aws.amazon.com/ko/s3/?nc=sn&loc=1

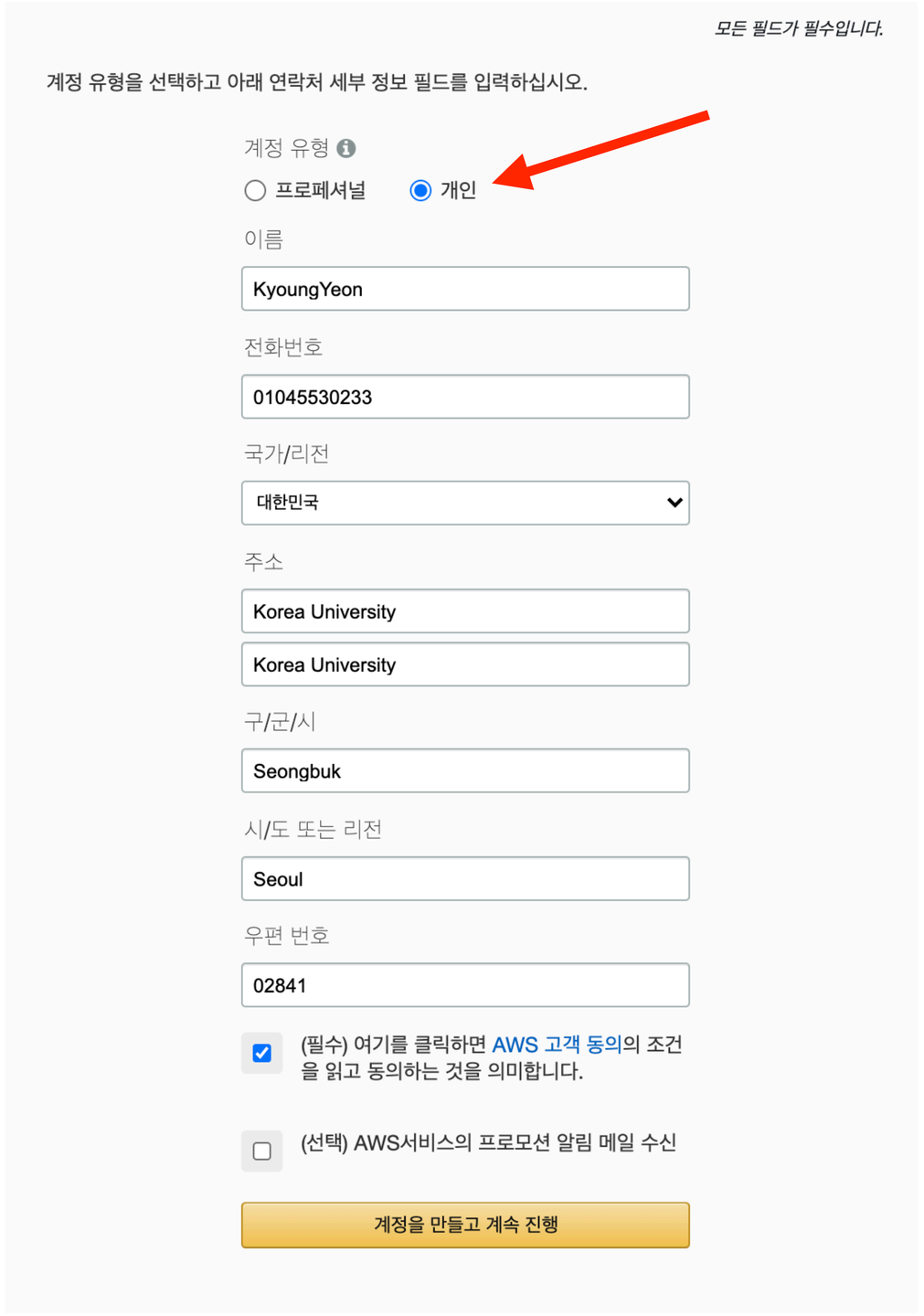
AWS 가입하기

- https://aws.amazon.com/
- 콘솔에 로그인



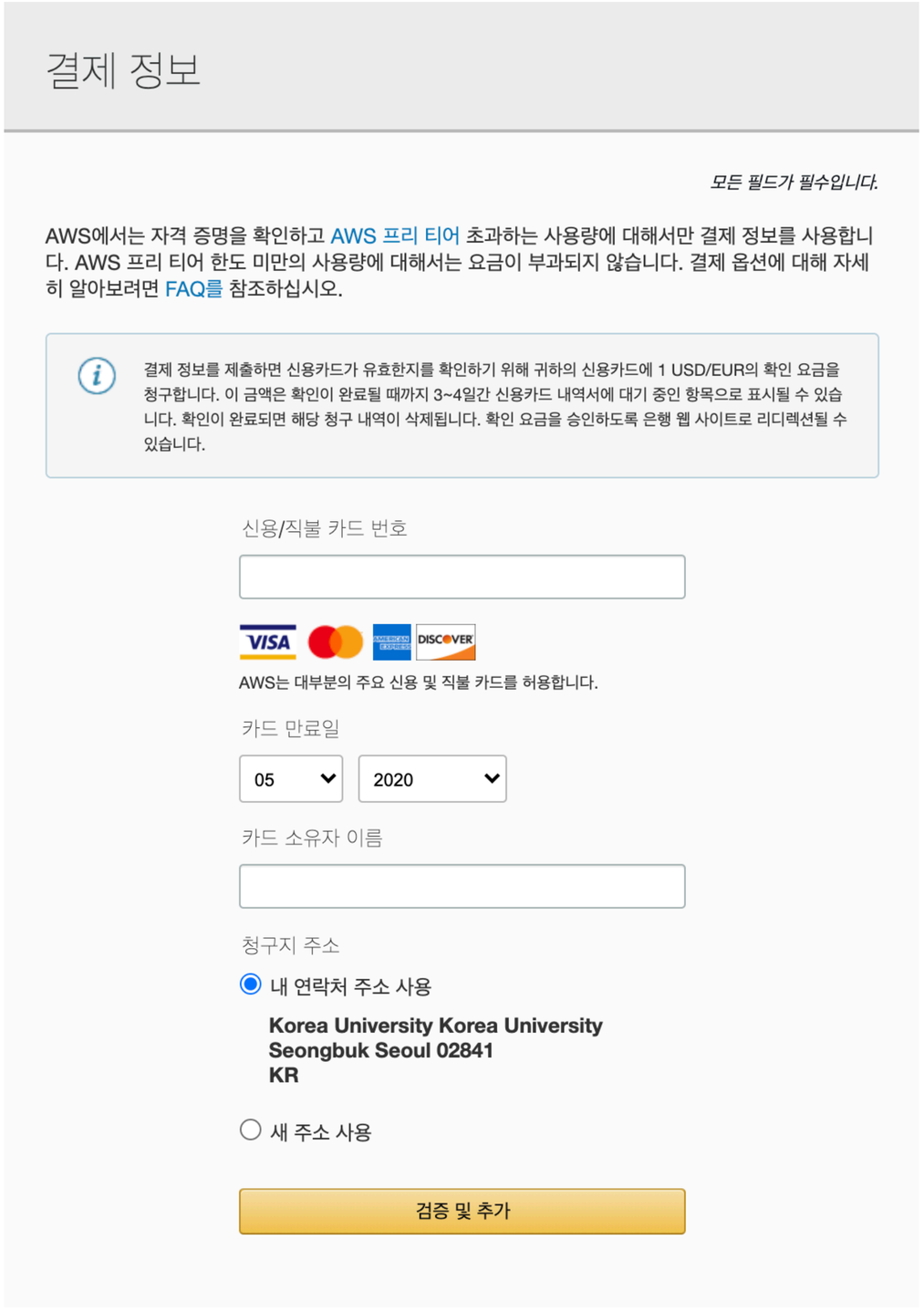
AWS 가입하기

- 아이디 만들고
- 연락처 정보 등록



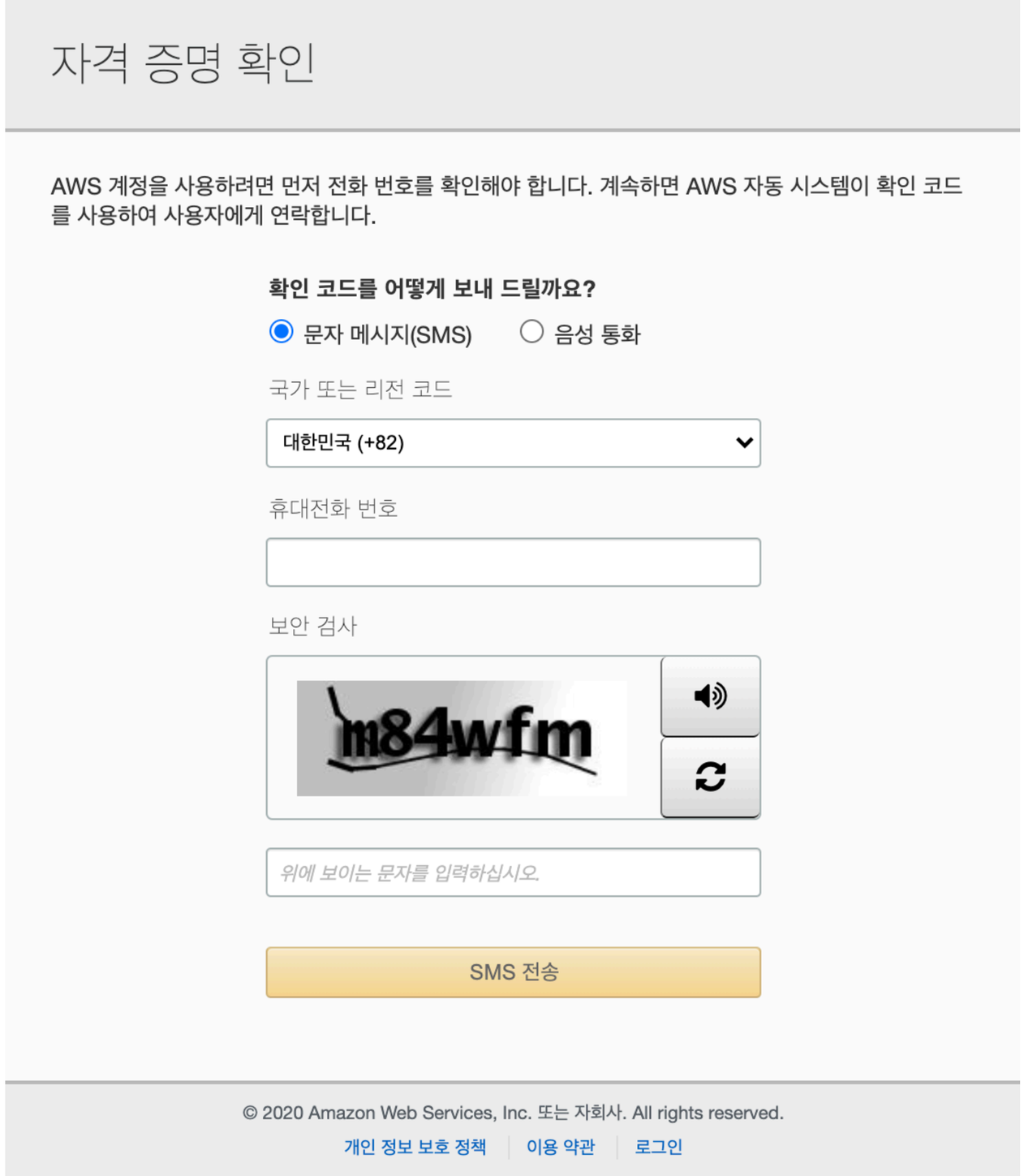
AWS 가입하기

- 결제정보 등록
- 해외결제 가능한 카드
- $1 빠져나갔다가 환급됨



AWS 가입하기

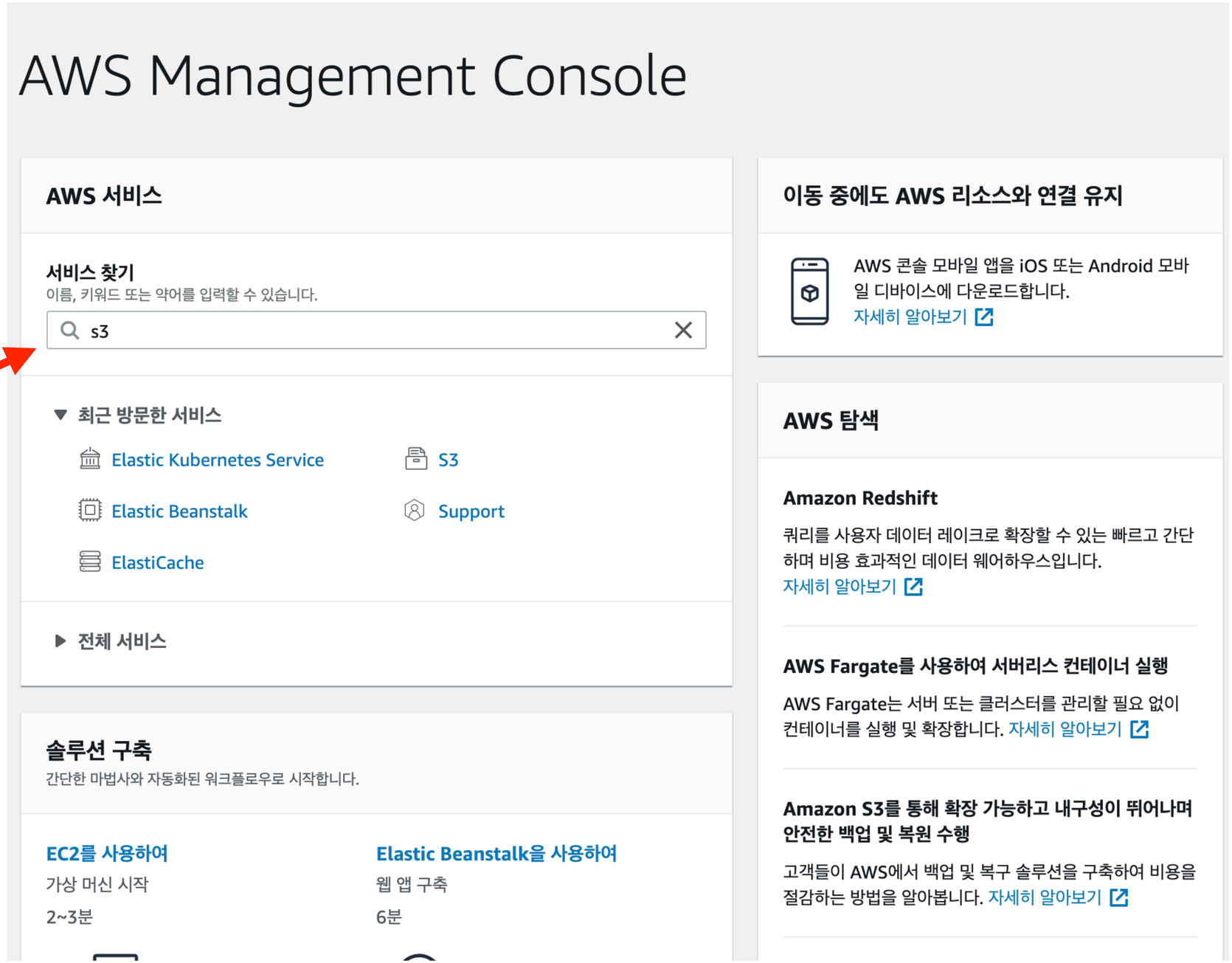
- 자격증명 확인
- 완료 되면 콘솔에 로그인



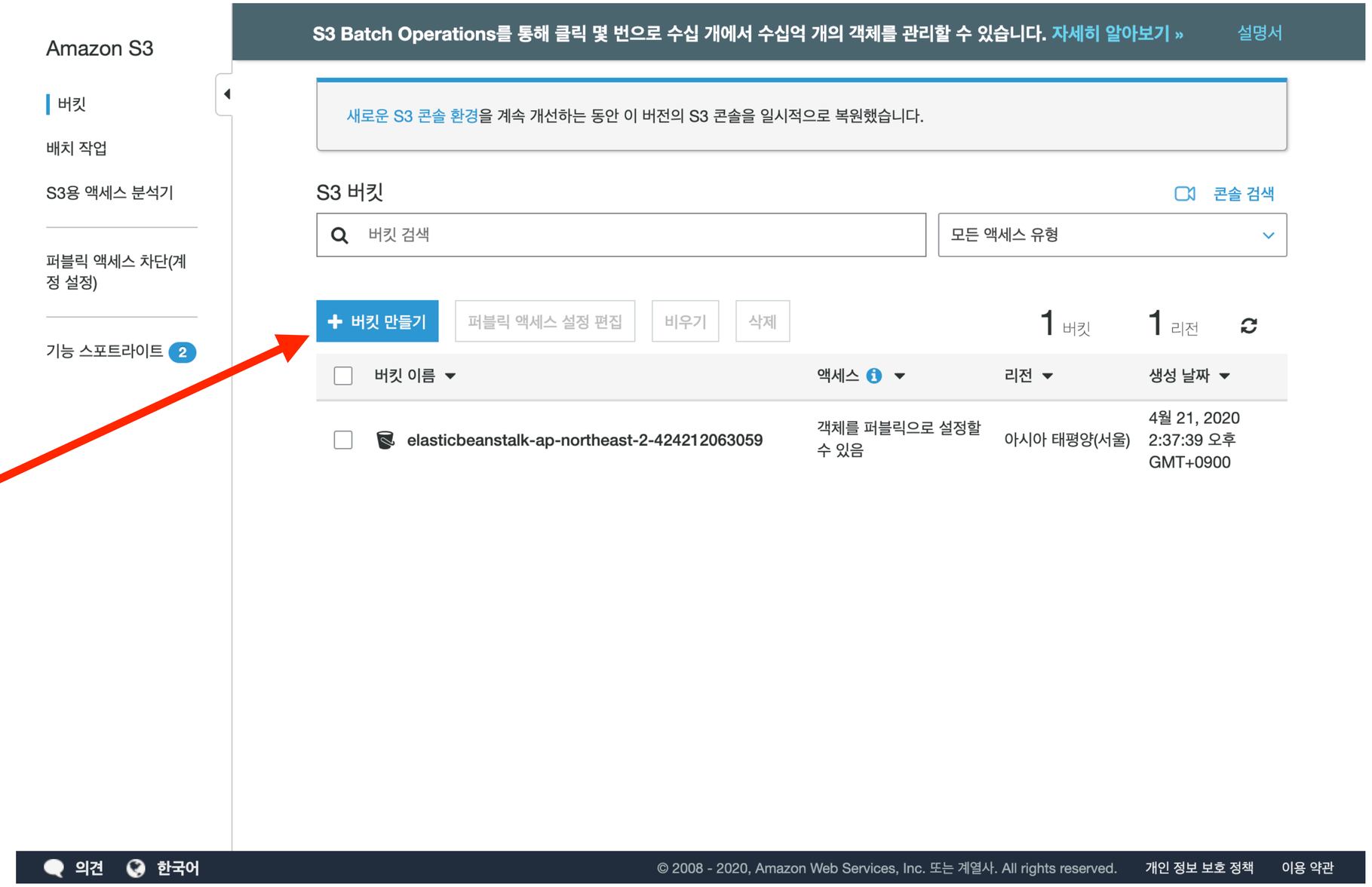
S3 버킷 생성하기

- S3 서비스

S3



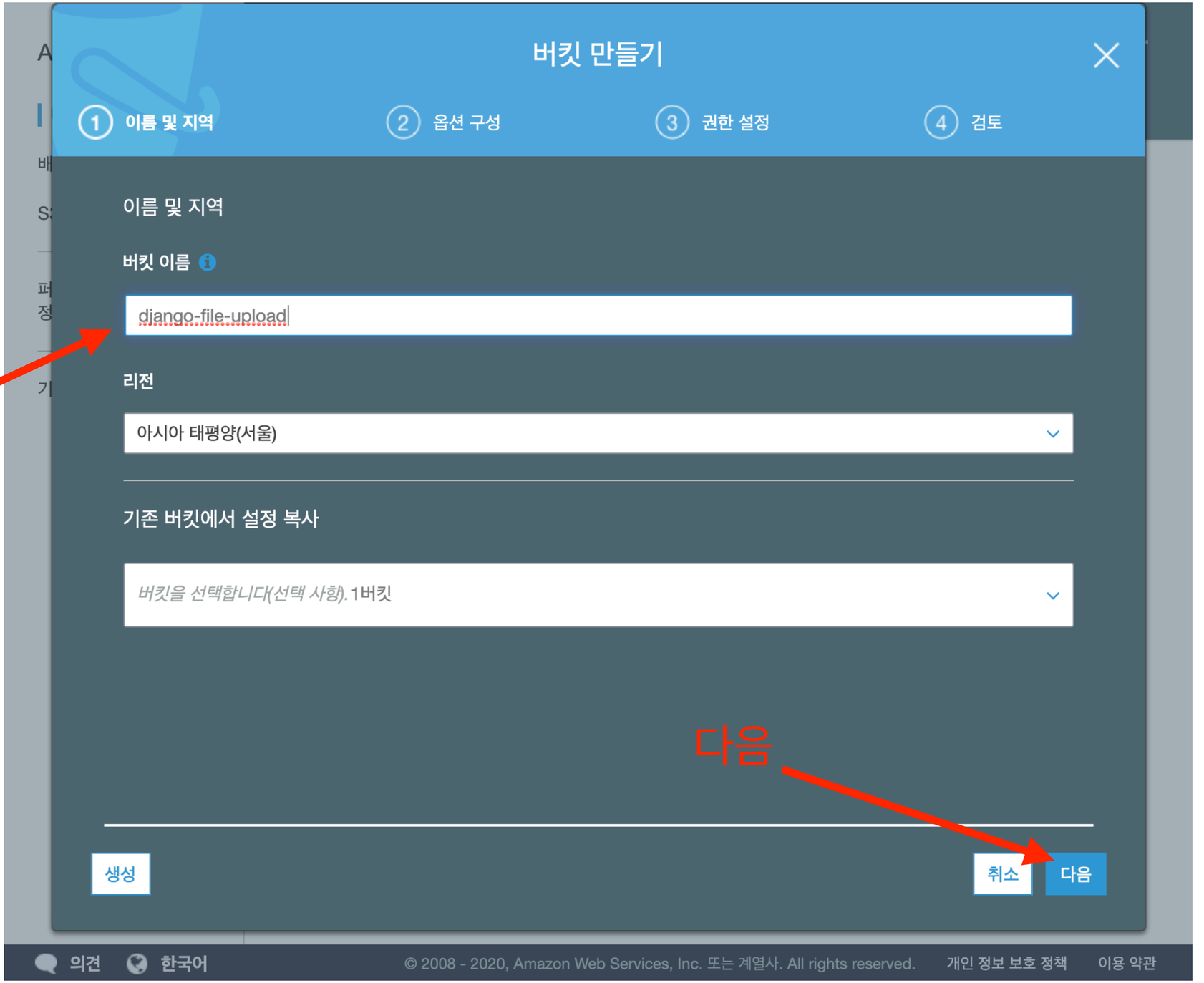
S3 버킷 생성하기



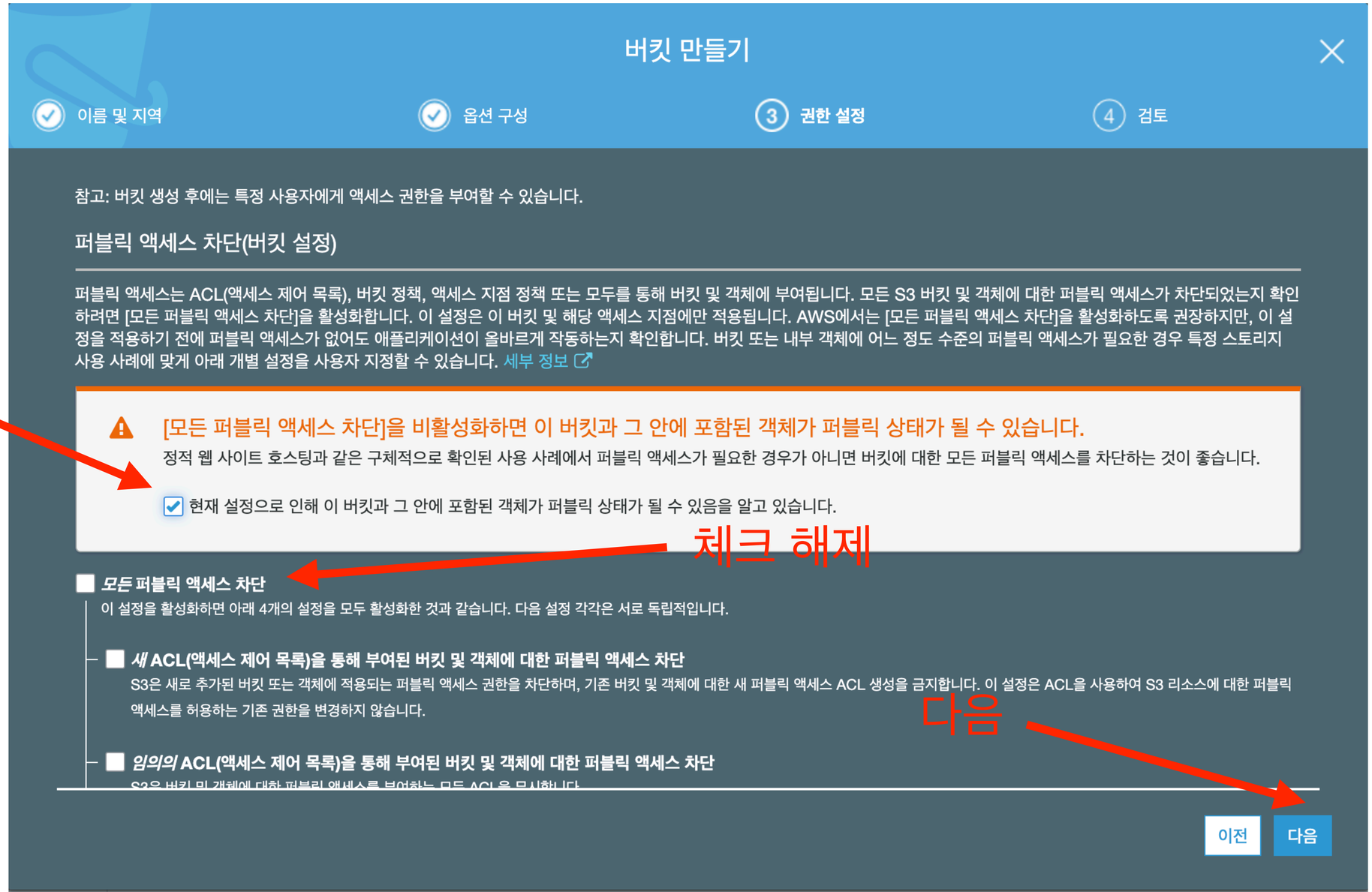
버킷 만들기

S3 버킷 생성하기

본인이 원하는 버킷 이름



S3 버킷 생성하기


버킷 만들기
이름 및 지역
옵션 구성
3 권한 설정
4 검토
참고: 버킷 생성 후에는 특정 사용자에게 액세스 권한을 부여할 수 있습니다.
퍼블릭 액세스 차단(버킷 설정)
퍼블릭 액세스는 ACL(액세스 제어 목록), 버킷 정책, 액세스 지점 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. 세부 정보
[모든 퍼블릭 액세스 차단]을 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 버킷에 대한 모든 퍼블릭 액세스를 차단하는 것이 좋습니다.
현재 설정으로 인해 이 버킷과 그 안에 포함된 객체가 퍼블릭 상태가 될 수 있음을 알고 있습니다.
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
이전
다음
체크
체크 해제
다음

S3 버킷 생성하기

- https://docs.aws.amazon.com/AmazonS3/latest/dev/example-bucket-policies.html

Granting Read-Only Permission to an Anonymous User

The following example policy grants the `s3:GetObject` permission to any public anonymous users. (For a list of permissions and the operations that they allow, see Amazon S3 Actions.) This permission allows anyone to read the object data, which is useful for when you configure your bucket as a website and want everyone to be able to read objects in the bucket. Before you use a bucket policy to grant read-only permission to an anonymous user, you must disable block public access settings for your bucket. For more information, see Setting permissions for website access.

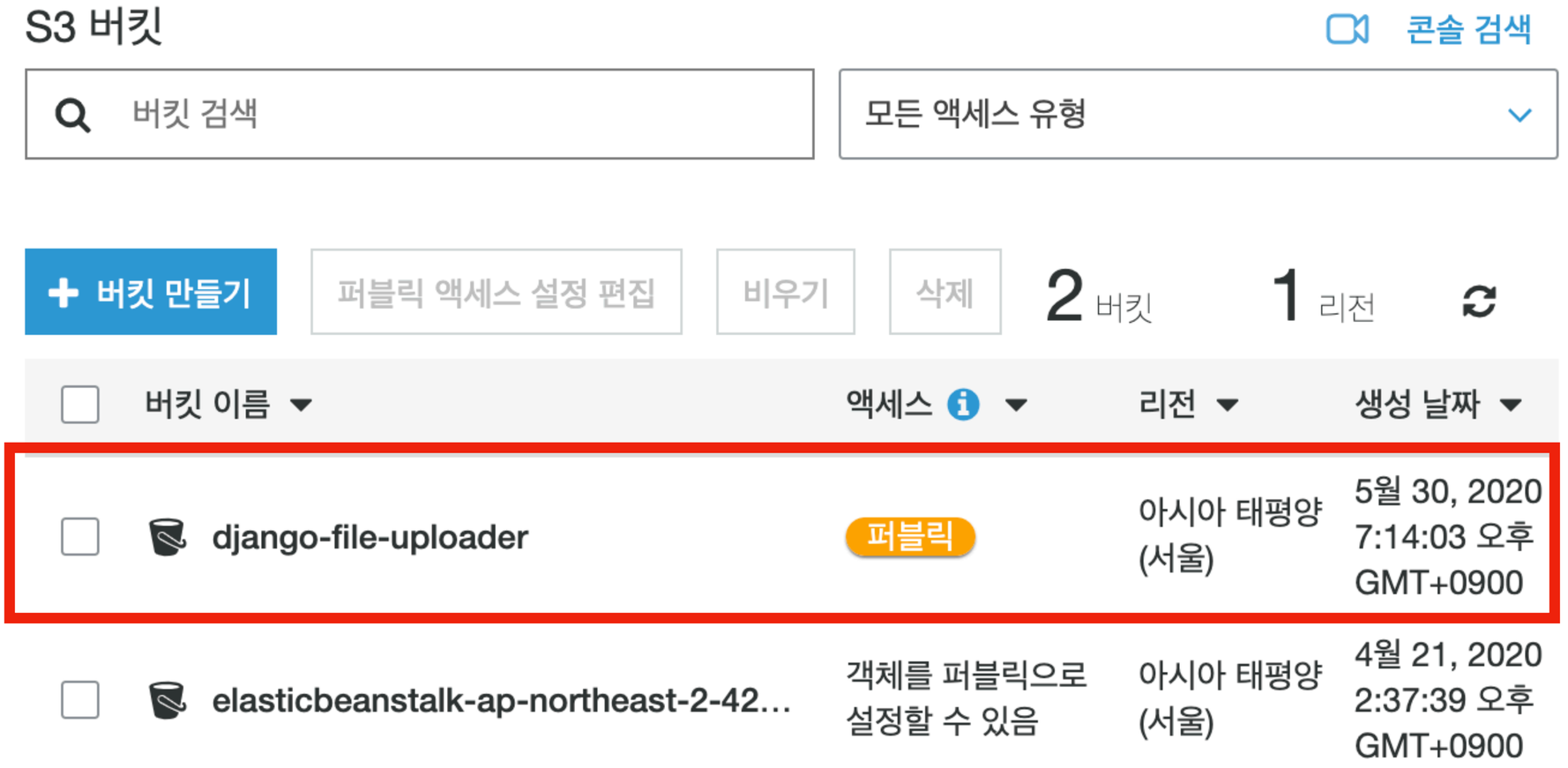
```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Sid":"PublicRead",
      "Effect":"Allow",
      "Principal": "*",
      "Action":["s3:GetObject"],
      "Resource":["arn:aws:s3:::examplebucket/*"]
    }
  ]
}
```

S3 버킷 생성하기

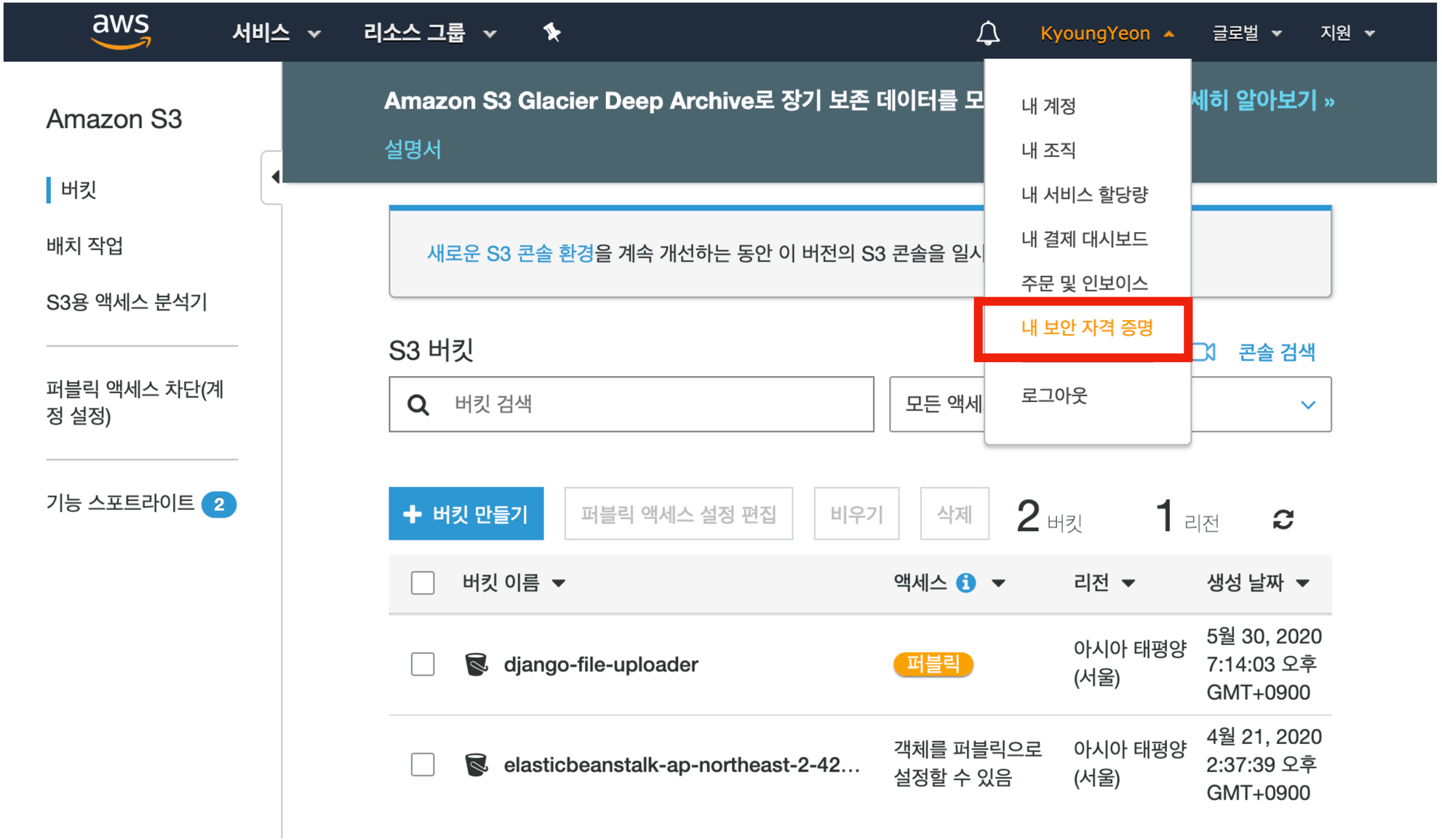

개요
속성
권한
관리
퍼블릭
액세스 지점
퍼블릭 액세스 차단
액세스 제어 목록
버킷 정책
퍼블릭
CORS 구성
버킷 정책 편집기 ARN: arn:aws:s3:::django-file-uploader
아래 텍스트 영역에 새 정책을 추가하거나 기존 정책을 편집하려면 입력합니다.
삭제
취소
저장
{
"Version": "2012-10-17",
"Statement": [
{
"Sid": "PublicRead",
"Effect": "Allow",
"Principal": "*",
"Action": "s3:GetObject",
"Resource": "arn:aws:s3:::django-file-uploader/*"
}
]
}

S3 버킷 생성하기

- 생성 완료



AWS 엑세스키 생성


aws
서비스
리소스 그룹
KyoungYeon
글로벌
지원
Amazon S3
버킷
배치 작업
S3용 액세스 분석기
퍼블릭 액세스 차단(계정 설정)
기능 스포트라이트 2
Amazon S3 Glacier Deep Archive로 장기 보존 데이터를 모
세히 알아보기 »
설명서
내 계정
내 조직
내 서비스 할당량
내 결제 대시보드
주문 및 인보이스
내 보안 자격 증명
로그아웃
새로운 S3 콘솔 환경을 계속 개선하는 동안 이 버전의 S3 콘솔을 일시
S3 버킷
콘솔 검색
버킷 검색
모든 액세
버킷 만들기
퍼블릭 액세스 설정 편집
비우기
삭제
2 버킷
1 리전
버킷 이름
액세스
리전
생성 날짜
django-file-uploader
퍼블릭
아시아 태평양 (서울)
5월 30, 2020 7:14:03 오후 GMT+0900
elasticbeanstalk-ap-northeast-2-42...
객체를 퍼블릭으로 설정할 수 있음
아시아 태평양 (서울)
4월 21, 2020 2:37:39 오후 GMT+0900

AWS 엑세스키 생성

보안 자격 증명

이 페이지를 사용하여 AWS 계정의 자격 증명을 관리합니다. AWS IAM(Identity and Access Management) 사용자에 대한 자격 증명을 관리하려면 IAM 콘솔 을(를) 사용하십시오.

AWS 자격 증명 유형과 사용 방법에 대해 자세히 알아보려면 AWS 일반 참조의 AWS 보안 자격 증명 을(를) 참조하십시오.

▲ 비밀번호

▲ 멀티 팩터 인증(MFA)

▼ 액세스 키(액세스 키 ID 및 비밀 액세스 키)

액세스 키를 사용하여 AWS CLI, PowerShell용 도구, AWS SDK 또는 직접 AWS API 호출을 통해 AWS를 프로그래밍 방식으로 호출합니다. 한 번에 최대 두 개의 액세스 키(활성 또는 비활성)를 가질 수 있습니다. 자세히 알아보기

생성 완료	액세스 키 ID	마지막 사용	마지막으로 사용한 리전	마지막으로 사용한 서비스	상태	작업

새 액세스 키 만들기

루트 사용자 액세스 키는 전체 AWS 계정에 대한 무제한 액세스를 제공합니다. 장기 액세스 키가 필요한 경우 제한된 권한이 부여된 새 IAM 사용자를 생성하고 해당 사용자에 대한 액세스 키를 루트 사용자 액세스 키 대신 생성하는 것이 좋습니다. 자세히 알아보기

▲ CloudFront 키 페어

▲ X.509 인증서

▲ 계정 ID

AWS 엑세스키 생성

- 키 파일 다운로드
- 절대로 외부에 노출되면 안됩니다!!!
  (깃헙, 친구, 부모 등)
- 한번 다운받으면 다시 다운 불가능하니 잘 보관하세요

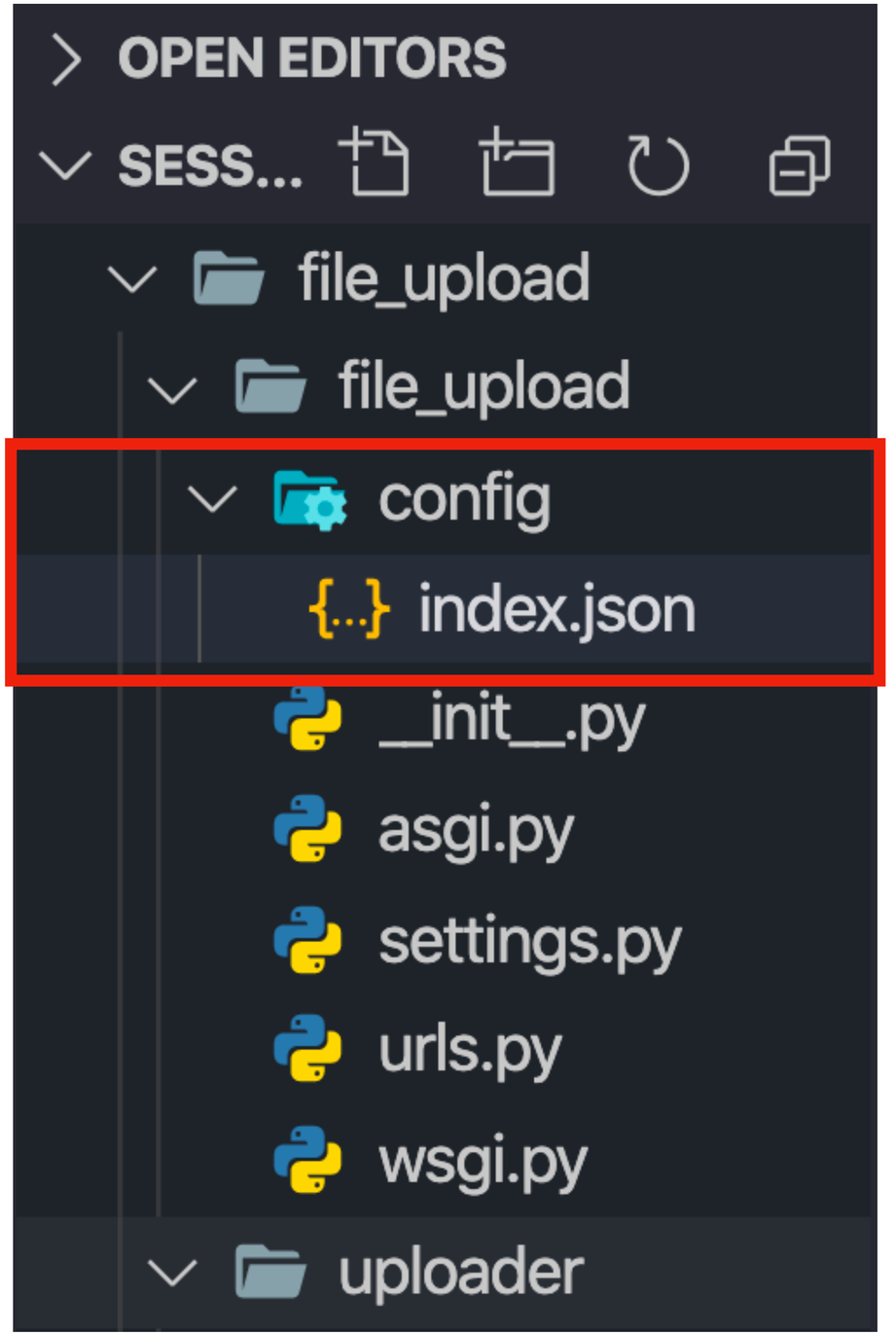
django 세팅

- setting.py에 aws 키를 올리면 안됩니다
- 깃헙에 올릴 때 settings.py는 같이 올라가기 때문
- 그렇다면 어떻게 해야 할까…

config 폴더 생성

- 환경설정 파일을 따로 관리

- file_upload/config/index.json



config/index.json

- 환경설정 파일을 따로 관리
- file_upload/config/index.json

file_upload > file_upload > config > {...} index.json > ...

```
{
    "AWS": {
        "AWS_ACCESS_KEY": "[redacted]",
        "SECRET_ACCESS_KEY": "[redacted]",
        "STORAGE_BUCKET_NAME": "django-file-upload"
    }
}
```

settings.py

- json import

```
13  import os, json
14
15  # Build paths inside the project like this: os.path.join(BASE_DIR, ...)
16  BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
17
18  with open(os.path.join(BASE_DIR, 'file_upload/config/index.json')) as f:
19      secrets = json.loads(f.read())
```

- https://www.pythonforbeginners.com/files/with-statement-in-python

settings.py

- AWS 설정 코드 추가

```
127    # S3 storage
128    DEFAULT_FILE_STORAGE = 'storages.backends.s3boto3.S3Boto3Storage'
129
130    # AWS Access
131    AWS_ACCESS_KEY_ID = secrets['AWS']['AWS_ACCESS_KEY']
132    AWS_SECRET_ACCESS_KEY = secrets['AWS']['SECRET_ACCESS_KEY']
133    AWS_STORAGE_BUCKET_NAME = secrets['AWS']['STORAGE_BUCKET_NAME']
134    AWS_S3_REGION_NAME = 'ap-northeast-2'
135    AWS_S3_FILE_OVERWRITE = False
```

git ignore (오늘 가장 중요한 부분!!!)

- 깃 이그노어파일: 깃이 기록하지 않는 파일/디렉토리 목록
- .gitignore는 .git파일보다 아래에 위치해야 한다

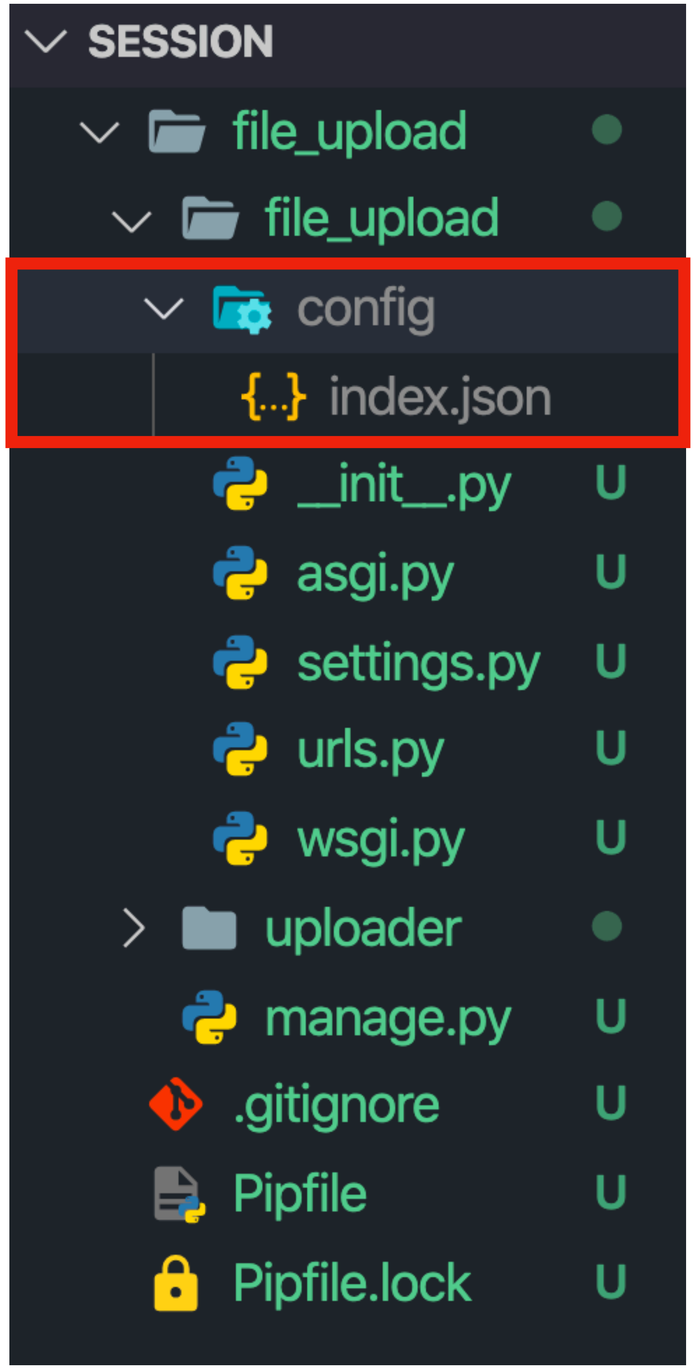
git init

- 깃 이그노어파일: 깃이 기록하지 않는 파일/디렉토리 목록
- 프로젝트 폴더(Pipfile, Pipfile.lock 이 있는 폴더)에서 git init

```
(session) ☁  session  ls
Pipfile       Pipfile.lock file_upload
(session) ☁  session  git init
Initialized empty Git repository in /Users/kyoungyeon/Desktop/session/.git/
(session) ☁  session [master] ⚡
```

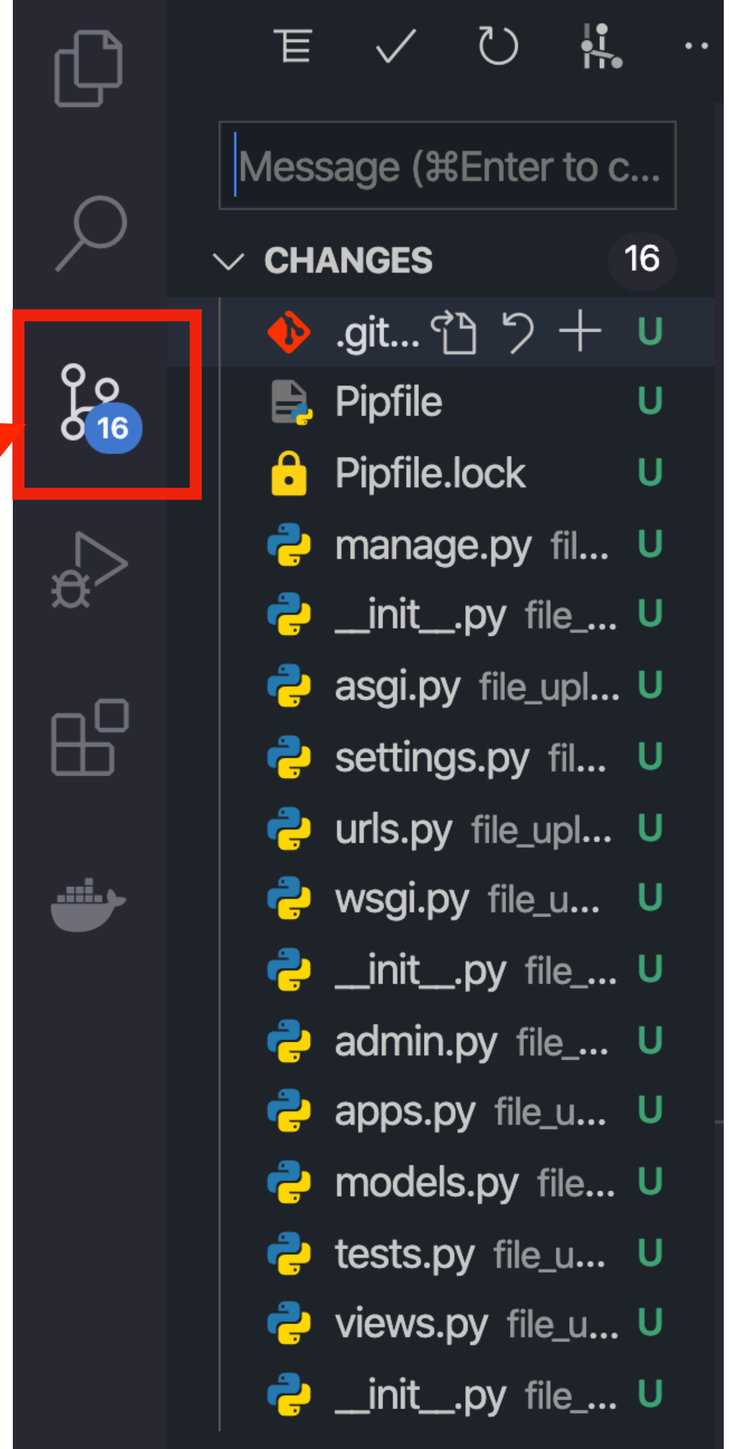
gitignore 확인 1

- config 폴더만 회색이다



gitignore 확인 2

- index.json이 없는 것을 확인



파일 업로드하기 (model)

```
3    # Create your models here.
4    class Post(models.Model):
5        title = models.CharField(max_length=50)
6        img = models.TextField()
```

파일 업로드하기 (template)

```
 8    <body>
 9      <form method="POST" enctype="multipart/form-data">
10        {% csrf_token %}
11        <input type="text" name="title" >
12        <input type="file" name="img" >
13        <button type="submit">업로드</button>
14      </form>
15
16      {% for post in posts %}
17      <div>
18        <h3>{{ post.title }}</h3>
19        <img src="{{ post.img }}" alt="{{ post.title }}">
20      </div>
21      {% endfor %}
22    </body>
23    </html>
```

파일 업로드하기 (views)

- settings.py에서 불러오는 것들도 한줄에 있습니다

```
1    from django.shortcuts import render, redirect
2    from .models import Post
3    from file_upload.settings import AWS_ACCESS_KEY_ID,
     AWS_SECRET_ACCESS_KEY, AWS_STORAGE_BUCKET_NAME,
     AWS_S3_REGION_NAME
4    import boto3
5    from boto3.session import Session
6    from datetime import datetime
```

파일 업로드하기 (views)

```python
def home(request):
    if (request.method == 'POST'):
        file_to_upload = request.FILES.get('img')
        session = Session(
            aws_access_key_id=AWS_ACCESS_KEY_ID,
            aws_secret_access_key=AWS_SECRET_ACCESS_KEY,
            region_name=AWS_S3_REGION_NAME
        )
        s3 = session.resource('s3')
        now = datetime.now().strftime("%Y%H%M%S")

        img_object = s3.Bucket(AWS_STORAGE_BUCKET_NAME).put_object(
            Key = now + file_to_upload.name,
            Body = file_to_upload
        )
        s3_url = 'https://django-file-uploader.s3.ap-northeast-2.amazonaws.com/'
        post = Post.objects.create(
            title = request.POST['title'],
            img = s3_url + now + file_to_upload.name
        )

        return redirect('home')
    posts = Post.objects.all()

    return render(request, 'home.html', { 'posts': posts })
```

파일 업로드하기 (url)

```
16  from django.contrib import admin
17  from django.urls import path
18  from uploader import views
19
20  urlpatterns = [
21      path('admin/', admin.site.urls),
22      path('', views.home, name='home'),
23  ]
```

파일 업로드하기 (utils)

- 너무 긴 코드가 view에 들어가 있다
- 새로운 파일 uploader에 utils.py를 만들자

파일 업로드하기 (utils)

```
1    from .models import Post
2    from file_upload.settings import AWS_ACCESS_KEY_ID,
     AWS_SECRET_ACCESS_KEY, AWS_STORAGE_BUCKET_NAME,
     AWS_S3_REGION_NAME
3    import boto3
4    from boto3.session import Session
5    from datetime import datetime
```

파일 업로드하기 (utils)

```python
def upload_and_save(request, file_to_upload):
    # client 만들기
    session = Session(
        aws_access_key_id=AWS_ACCESS_KEY_ID,
        aws_secret_access_key=AWS_SECRET_ACCESS_KEY,
        region_name=AWS_S3_REGION_NAME
    )
    s3 = session.resource('s3')

    # s3에 object 업로드
    now = datetime.now().strftime("%Y%H%M%S")
    img_object = s3.Bucket(AWS_STORAGE_BUCKET_NAME).put_object(
        Key = now + file_to_upload.name,
        Body = file_to_upload
    )

    # Post model 만들기
    s3_url = 'https://django-file-uploader.s3.ap-northeast-2.amazonaws.com/'
    post = Post.objects.create(
        title = request.POST['title'],
        img = s3_url + now + file_to_upload.name
    )
```

파일 업로드하기 (views)

- 불필요한 import 삭제 / utils에서 함수 import

```
1    from django.shortcuts import render, redirect
2    from .models import Post
3    from .utils import upload_and_save
```

파일 업로드하기 (views)

- 코드가 깔끔해졌다

```
5     def home(request):
6         if (request.method == 'POST'):
7             file_to_upload = request.FILES.get('img')
8             upload_and_save(request, file_to_upload)
9 
10            return redirect('home')
11        posts = Post.objects.all()
12
13        return render(request, 'home.html', { 'posts': posts })
```

참고자료

- https://django-storages.readthedocs.io/en/latest/backends/amazon-S3.html
- https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/s3.html
- https://bitbucket.org/david/django-storages/src/83fa2f0ba20c/storages/backends/s3boto.py
- https://django-storages.readthedocs.io/en/latest/backends/amazon-S3.html

과제

- user 별로 다른 폴더에 이미지 업로드 하기
  HINT: pk
- 해커톤 준비 열심히하기