S3 환경설정

- `boto3.generate_presigned_url()` 은 **S3에 실제 접근하는 게 아니라, AWS 계정의 키로 URL을 '서명'하는 행위**
- 그 서명을 하려면, **"서명할 키"**, 즉 `AWS_ACCESS_KEY_ID` / `AWS_SECRET_ACCESS_KEY`가 필요
- 이 키는 **S3를 소유한 AWS 계정에서 발급 →** 유효한 presigned URL을 생성

⚙️ 설정 방식

① AWS 콘솔 → IAM → 사용자 생성

1. 이름 예: `ec2-s3-presigner`
2. **S3 접근 전용 최소 권한 정책** 부여(IAM)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["s3:GetObject", "s3:PutObject"],
      "Resource": ["arn:aws:s3:::visioninapp-bucket/*"]
    },
    {
      "Effect": "Allow",
      "Action": ["s3:ListBucket"],
      "Resource": ["arn:aws:s3:::visioninapp-bucket"]
    }
  ]
}
```

3. **Access Key / Secret Key** 발급

   (CSV 다운로드)

---

② EC2 API 안에 환경변수로 설정

또는 `.env` 파일 만들어두고 Python이 실행될 때 불러오기:

```
AWS_ACCESS_KEY_ID=AKIAxxxxxxxxxxxx
AWS_SECRET_ACCESS_KEY=xxxxxxxxxxxxxxxxxxxxxx
REGION_NAME=ap-northeast-2
BUCKET_NAME=xxxx-bucket
```

---

③ boto3 라이브러리 사용 방법

IAM 사용자 사용법 (로컬 파일 사용한 예시일 뿐, 실제로는 수정 필요)

```
import boto3
import os

# AWS 자격증명
AWS_ACCESS_KEY_ID = "YOUR_ACCESS_KEY"
AWS_SECRET_ACCESS_KEY = "YOUR_SECRET_KEY"
REGION_NAME = "ap-northeast-2"
BUCKET_NAME = "YOUR_BUCKET_NAME"

# 조회 & 다운로드 대상 (S3에 이미 존재하는 Key)
SAVE_KEY = "test/best.pt"

# 로컬 다운로드 경로
LOCAL_SAVE_PATH = os.path.basename(SAVE_KEY)  # ex) best.pt

# 업로드할 로컬 파일 경로 (예시로 직접 지정)
LOCAL_UPLOAD_PATH = "result/my_model.pt"  # ✅ 업로드할 로컬 파일 경로
UPLOAD_KEY = "result/my_model.pt"         # ✅ 업로드될 S3 내 경로/이름

# boto3 클라이언트 생성
s3 = boto3.client(
    "s3",
    aws_access_key_id=AWS_ACCESS_KEY_ID,
    aws_secret_access_key=AWS_SECRET_ACCESS_KEY,
    region_name=REGION_NAME
)
```

```
# ================================
# 1️⃣ S3 객체 메타데이터 조회
# ================================
try:
    response = s3.head_object(Bucket=BUCKET_NAME, Key=SAVE_KEY)
    print("✅ [조회 성공]")
    print(f"파일명: {SAVE_KEY}")
    print(f"파일 크기: {response['ContentLength']} bytes")
    print(f"마지막 수정일: {response['LastModified']}")
except Exception as e:
    print("❌ [조회 실패]:", e)

# ================================
# 2️⃣ S3 객체 다운로드
# ================================
try:
    s3.download_file(BUCKET_NAME, SAVE_KEY, LOCAL_SAVE_PATH)
    print(f"✅ [다운로드 완료] → {LOCAL_SAVE_PATH}")
    print(f"📁 로컬 경로: {os.path.abspath(LOCAL_SAVE_PATH)}")
except Exception as e:
    print("❌ [다운로드 실패]:", e)

# ================================
# 3️⃣ S3 객체 업로드 (직접 이름 지정)
# ================================
try:
    if not os.path.exists(LOCAL_UPLOAD_PATH):
        raise FileNotFoundError(f"로컬 업로드 파일이 존재하지 않습니다: {LOCAL_
UPLOAD_PATH}")

    s3.upload_file(LOCAL_UPLOAD_PATH, BUCKET_NAME, UPLOAD_KEY)
    print(f"✅ [업로드 완료] → s3://{BUCKET_NAME}/{UPLOAD_KEY}")
    print(f"📤 업로드한 로컬 파일: {LOCAL_UPLOAD_PATH}")
except Exception as e:
    print("❌ [업로드 실패]:", e)
```

async 버전 (aioboto3)

```
import aioboto3
import asyncio
import os

AWS_ACCESS_KEY_ID = "YOUR_ACCESS_KEY"
AWS_SECRET_ACCESS_KEY = "YOUR_SECRET_KEY"
REGION_NAME = "ap-northeast-2"
BUCKET_NAME = "YOUR_BUCKET_NAME"

SAVE_KEY = "test/best.pt"
LOCAL_SAVE_PATH = os.path.basename(SAVE_KEY)
LOCAL_UPLOAD_PATH = "result/my_model.pt"
UPLOAD_KEY = "result/my_model.pt"

async def main():
    # aioboto3 클라이언트 (비동기 컨텍스트)
    async with aioboto3.client(
        "s3",
        aws_access_key_id=AWS_ACCESS_KEY_ID,
        aws_secret_access_key=AWS_SECRET_ACCESS_KEY,
        region_name=REGION_NAME
    ) as s3:

        # 1️⃣ 메타데이터 조회
        try:
            response = await s3.head_object(Bucket=BUCKET_NAME, Key=SAV
E_KEY)
            print("✅ [조회 성공]")
            print(f"파일명: {SAVE_KEY}")
            print(f"파일 크기: {response['ContentLength']} bytes")
            print(f"마지막 수정일: {response['LastModified']}")
        except Exception as e:
            print("❌ [조회 실패]:", e)

        # 2️⃣ 다운로드
        try:
            await s3.download_file(BUCKET_NAME, SAVE_KEY, LOCAL_SAVE_P
ATH)
```

```
            print(f"✅ [다운로드 완료] → {LOCAL_SAVE_PATH}")
            print(f"📁 로컬 경로: {os.path.abspath(LOCAL_SAVE_PATH)}")
        except Exception as e:
            print("❌ [다운로드 실패]:", e)

        # 3️⃣ 업로드
        try:
            if not os.path.exists(LOCAL_UPLOAD_PATH):
                raise FileNotFoundError(f"로컬 업로드 파일이 존재하지 않습니다: {LOCAL_UPLOAD_PATH}")

            await s3.upload_file(LOCAL_UPLOAD_PATH, BUCKET_NAME, UPLOAD_KEY)
            print(f"✅ [업로드 완료] → s3://{BUCKET_NAME}/{UPLOAD_KEY}")
        except Exception as e:
            print("❌ [업로드 실패]:", e)

if __name__ == "__main__":
    asyncio.run(main())
```

Presigned URL

```
import boto3

s3 = boto3.client("s3")
url = s3.generate_presigned_url(
    "get_object",
    Params={"Bucket": "visioninapp-bucket", "Key": "datasets/train.zip"},
    ExpiresIn=3600
)
print(url)
```

> presigned URL은 네 AWS 계정의 키로 서명됐기 때문에,
>
> EC2가 어느 클라우드/어느 계정에 있든지 상관없이 “HTTP로 접근 가능한 임시 S3 링크”가 만들어짐.

- 이 IAM 사용자는 오직 **presigned URL 생성용 / S3 접근용**으로만 쓰기
- 절대 FullAccess 주지 말기
- **키는 .env 또는 AWS Parameter Store / Secret Manager 등에서 안전하게 보관**

✅ 요약

> 모델 조회/다운/업로드 방식
>
> **S3를 가진 계정에서 발급한 Access Key ID / Secret Access Key를 EC2 환경변수에 등록해두고, boto3가 그걸 써서 presigned URL을 생성하는 구조.**
>
> EC2가 다른 계정에 있어도, S3 접근 없이 "서명된 다운로드 링크"를 만들어 GPU 서버나 RabbitMQ로 넘길 수 있다.

모델 학습 아키텍처