<PostgreSQL 데이터 베이스 적용하기>

- 데이터베이스를 사용하는 이유

1. 영속성 : 서버 재시작이나 장애가 발생해도 데이터가 손실되지 않고 안전하게 보존된다

2. 구조화된 저장과 조회 : 데이터를 논리적으로 설계하고, 복잡한 조건 검색을 빠르게 수행한다

3. 동시성 제어 : 여러 사용자가 동시에 읽기,쓰기 할 때, 일관성을 유지하도록 트랜잭션을 지원한다.

4. 보안, 백업, 복구

[실습] PostgreSQL 데이터베이스 인스턴스 생성하기.

1) 데이터 베이스 생성하기

↳ AWS 라이트 세일에 데이터 베이스 생성 → 지역 선택

2) 버전과 플랜 선택.

3) 데이터 베이스 인스턴스 필수값 입력 후 생성

↳ 리소스 이름에 Database-1 입력

4) 데이터베이스 인스턴스 설정 확인

↳ 사용자 이름, 암호, 데이터 베이스 주소는 반드시 기억해야한다.

[실습] 데이터 베이스 생성하고 파이썬에서 사용하기.

↳ 생성된 PostgreSQL 인스턴스에 파이썬가 사용할 'pybo' 라는 이름의 데이터베이스를 생성한다.

1) 서버에 PostgreSQL 클라이언트 설치하기

sudo apt install postgresql-client

2) 데이터 베이스 생성하기

```
createdb pybo --username=dbmasteruser -h <데이터베이스 주소>
```

↳ 데이터 베이스 주소는 앞에서 확인한 '엔드포인트'를 의미한다.

3) 파이썬에서 데이터 베이스에 접속하기.

↳ 파이썬는 장고로 개발되었으니 psycopg2-binary 모듈이 필요하다.

pip install psycopg2-binary

4) settings/prod.py 파일에 DATABASE 항목 설정하기. (settings/prod.py)

```
DATABASE = {
    'default' = {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': 'pybo',
        'USER': 'dbmasteruser',
        'PASSWORD': 앞은
        'HOST': '  '
        'PORT': '  ',
    }
}
```

↳ git으로 서버에 반영하기.

5) 변경된 데이터 베이스 적용하기.

↳ 서버에서 migrate 수행

6) 슈퍼유저 생성하기.

↳ Gunicorn을 재실행 후 파이썬 기능을 테스트 해보기.

[실습] PgAdmin으로 로컬에서 PostgreSQL 서버 접속하기.

↳ 나중에 서버를 운영하다 보면, 데이터베이스 백업과 같은 작업을 할 수도 있는데 그런 경우 PgAdmin 프로그램은 많은 편의를 제공해준다.

1) AWS 데이터베이스 퍼블릭 모드로 변경하기.

↳ PgAdmin으로 데이터베이스에 접속하려면, AWS의 [네트워킹] 탭에 있는 '네트워크 보안'을 '퍼블릭 모드'로 변경해야 한다.

2) PgAdmin 설치하고 PostgreSQL 서버에 접속하기.

↳ Create - Server 창에서 [General] 탭의 'Name' 항목에 'Pybo' 입력 이어서 [Connection] 탭의 내용을 입력하고 Save.

이후 PgAdmin으로 데이터베이스 작업을 할 수 있다.