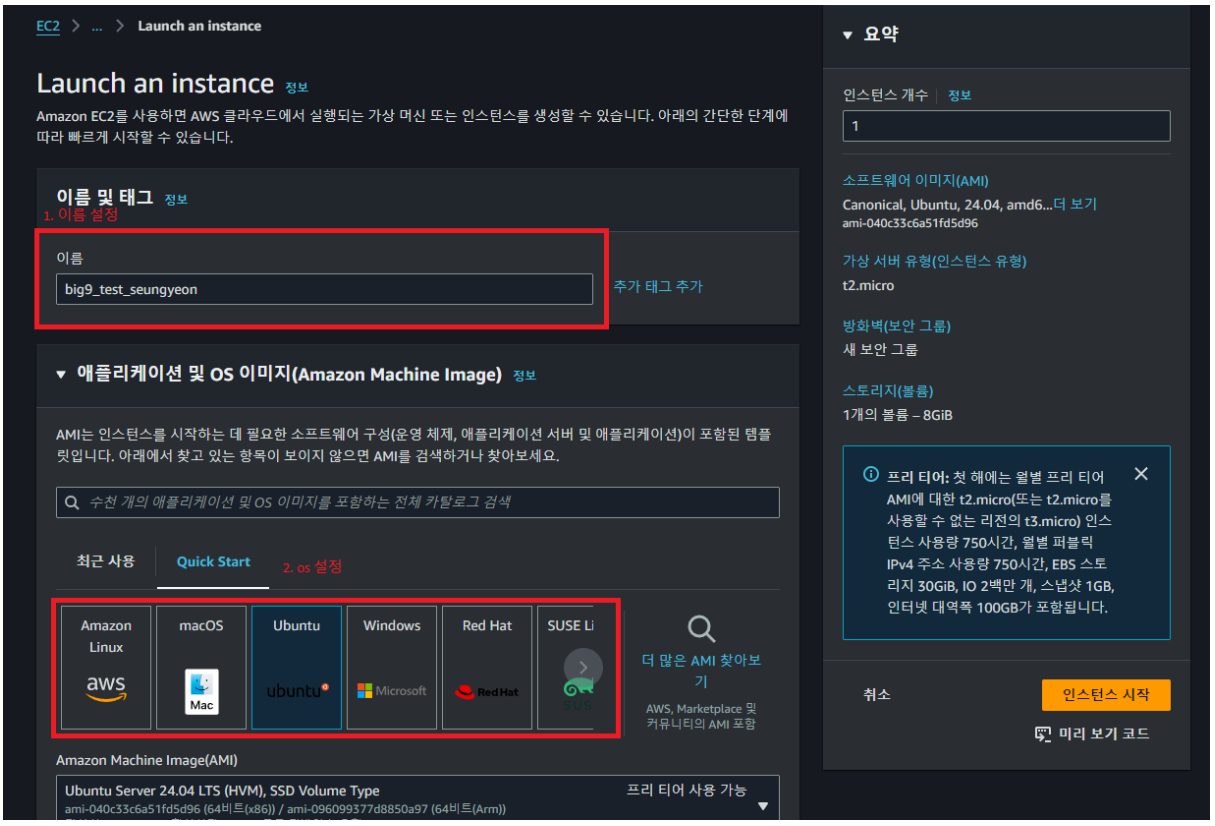
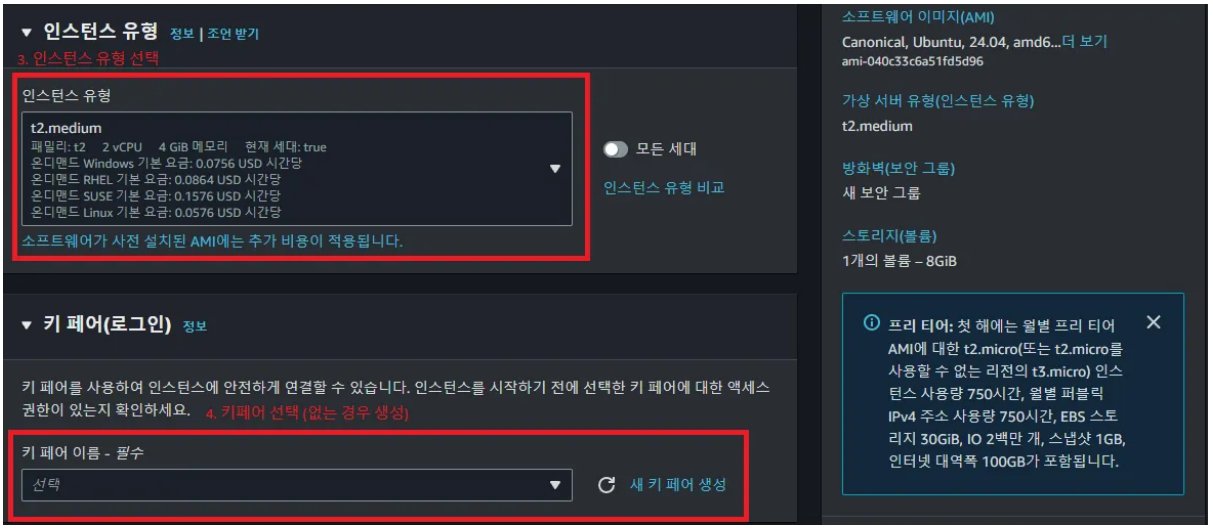
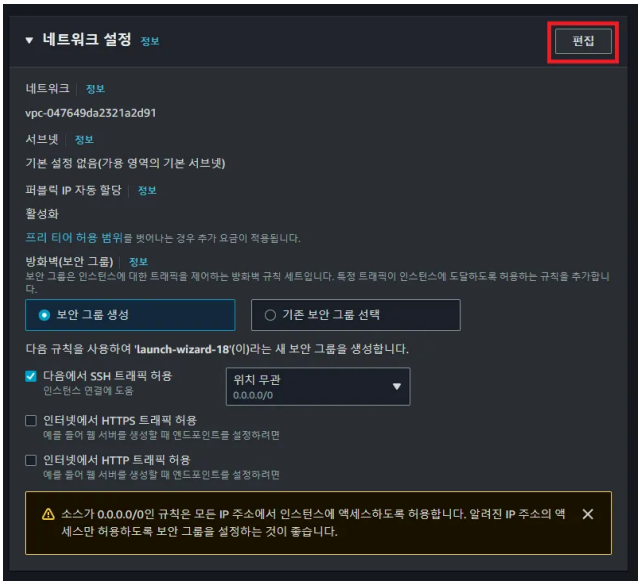
AWS_실습_빅데이터9기_정승연

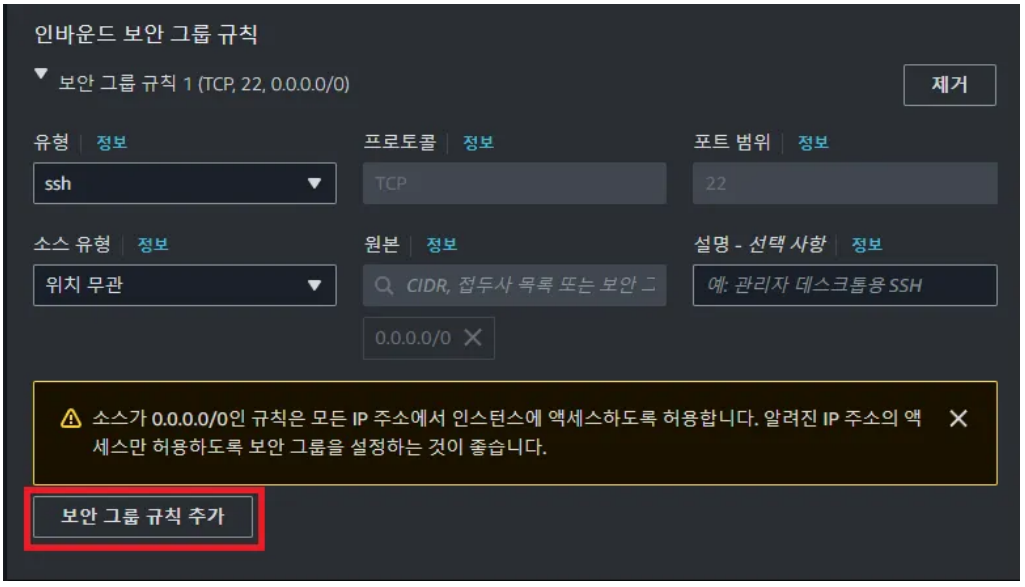
가. AWS 환경 구축하기

1. EC2 생성

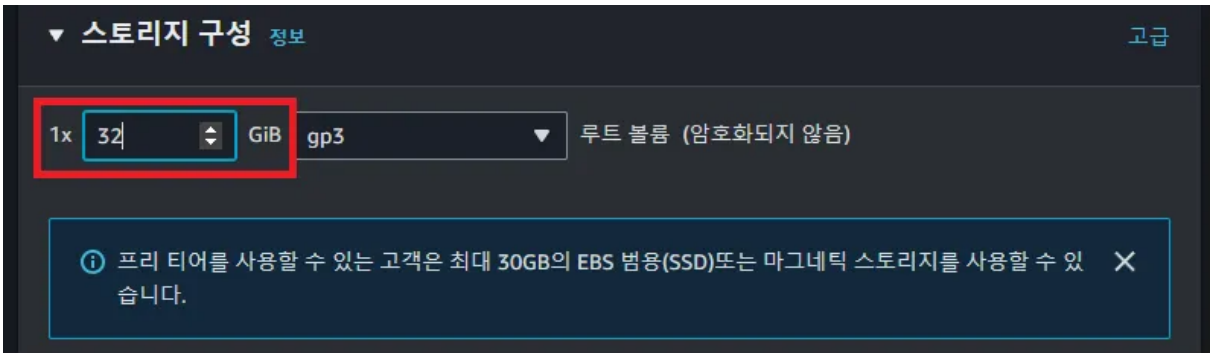






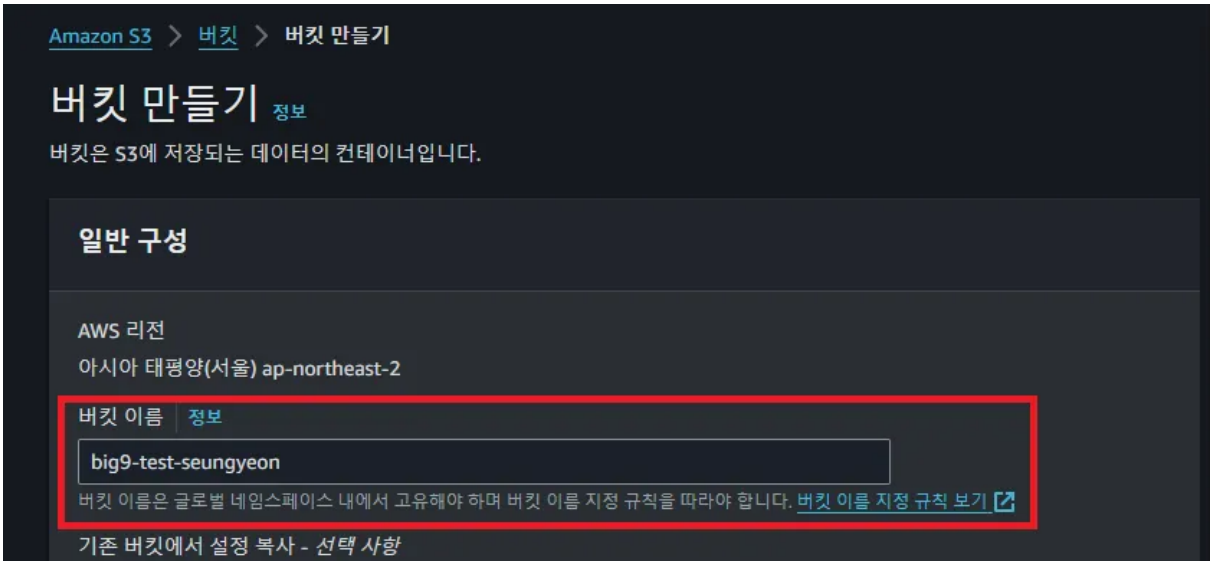


- 보안 그룹 : ssh(22), http(80), https(443), 5000포트, 8888포트 규칙 추가
- 5000포트, 8888포트의 경우 유형을 사용자 정의 TCP 지정 후 포트에 임의로 추가



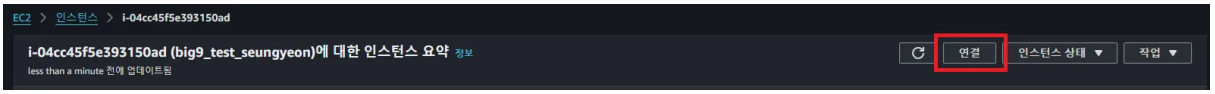
- 스토리지 : 32GiB 이상

2. S3 생성



3. EC2에 Python 개발 환경 구축

1. EC2 인스턴스 연결



2. conda 설치 (버전 확인하기)

```
wget https://repo.anaconda.com/archive/Anaconda3-2024.06-1-Linux-x86_64.sh
```

3. bash script 경로 설정

```
bash Anaconda3-2024.06-1-Linux-x86_64.sh
```

4. conda base 환경 실행

```
conda activate base
```

5. jupyter, ipykernel 설치

```
pip install jupyter ipykernel
```

6. 경로 이동

```
cd /home/ubuntu/anaconda3/
```

7. 주피터 노트북 실행

```
jupyter notebook
```

8. boto3를 통해 S3 bucket으로 데이터 업로드 기능 구현
    a. .env 파일을 생성하여 자신의 키 파일을 따로 관리할 것

```
ACCESS_KEY = your_access_key
SECRET_KEY = your_secret_key
region = your_region

S3_BUCKET = your_s3_bucket_name
```

b. 데이터 업로드 함수 구현

```
import os
from dotenv import load_dotenv
import boto3

# .env 파일에서 환경 변수 로드
load_dotenv()

# 환경 변수를 가져와 boto3에서 사용
s3 = boto3.client(
    's3',
    aws_access_key_id=os.getenv('AWS_ACCESS_KEY_ID'),
    aws_secret_access_key=os.getenv('AWS_SECRET_ACCESS_KEY')
)

# S3 버킷에 파일 업로드
bucket_name = 'S3_BUCKET'
file_name = 'your_file.csv'
s3.upload_file(file_name, bucket_name, file_name)
```

나. AWS 데이터 분석 서비스 파이프라인 구축하기

1. EC2 인스턴스 연결
2. EC2 인스턴스에서 S3 Bucket 데이터(csv) 다운로드 기능 구현
    a. python3와boto3가 설치되지 않은 경우 설치 후 진행(`sudo apt install python3 boto3`)
    b. 파일 생성 (`nano your_file_name.py`)

```
import boto3

s3 = boto3.client('s3')
bucket_name = 'your_s3_bucket_name'  # S3 버킷 이름
file_name = 'your_file.csv'  # 다운로드할 파일 이름
local_file_path = '/home/ubuntu/your_file.csv'  # 다운로드할 로컬 경로
```

```
s3.download_file(bucket_name, file_name, local_file_path)
print(f'{file_name}가 {local_file_path}에 다운로드되었습니다.')
```

2. EC2 인스턴스에서 분석 코드 구현

```
import pandas as pd

local_file_path = '/home/ubuntu/your_file.csv'  # 다운로드 파일 경로
data = pd.read_csv(local_file_path)

# 데이터 정보 출력
print(data.info())

# 추가 분석 코드 여기에 작성
```

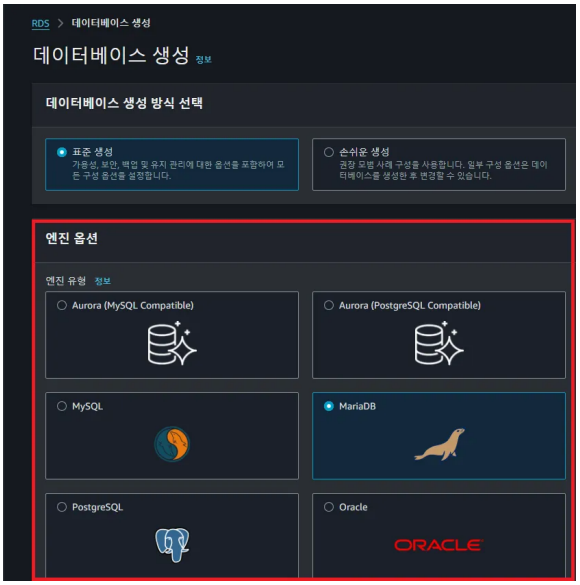
3. EC2 인스턴스에서 S3 Bucket으로 데이터 업로드 기능 구현

```
import boto3

# S3 클라이언트 생성
s3 = boto3.client('s3')

# 업로드할 파일 이름과 버킷 이름
file_name = 'your_file.csv'
bucket_name = 'your_s3_bucket_name'

# S3에 파일 업로드
s3.upload_file(file_name, bucket_name, file_name)

print(f'{file_name} has been uploaded to {bucket_name}.')
```

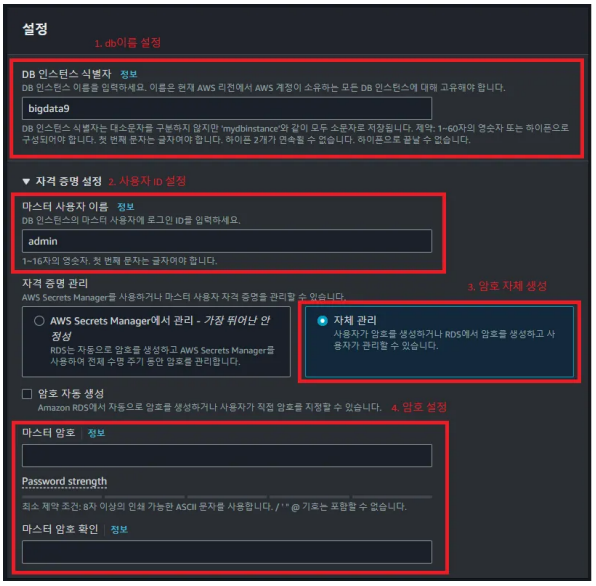
- 함수 실행 방법: `python3 your_file_name.py`

다. RDS에 데이터 적재하기

1. RDS 인스턴스 생성



- 사용하는 데이터베이스 엔진 선택



2. '나' 부분에서 발생하는 데이터 EC2에서 RDS로 데이터 적재 진행
   a. EC2 인스턴스 연결 후 RDS 연결
      - mariadb로 연결한다는 가정 하에 아래 코드 작성 후 설정한 암호 입력

      `mariadb -h {rds_endpoint} -u {username(e.g. admin)} -p -P {port(e.g. 3306}`

   b. 데이터베이스 사용 `use {db name(e.g. bigdata9)};`

   c. 테이블 생성

```
CREATE TABLE your_table_name (
    id INT AUTO_INCREMENT PRIMARY KEY,
    column1 VARCHAR(255),
```

```
    column2 VARCHAR(255)
);
```

d. 데이터 적재 함수 구현

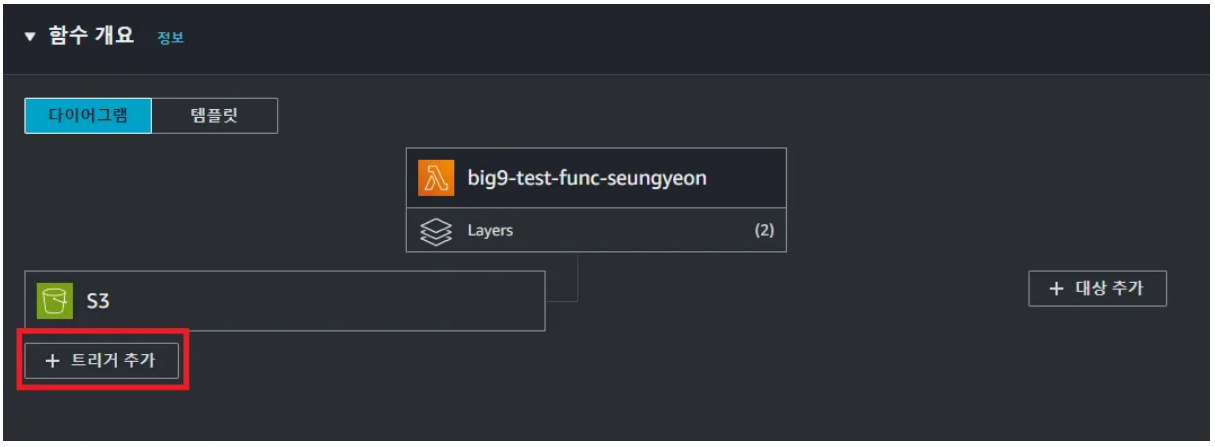
```
import pandas as pd
import mysql.connector
from mysql.connector import Error

# CSV 파일 읽기
df = pd.read_csv('your_file.csv')  # CSV 파일 경로

# 데이터 타입 변환 (필요에 따라 조정)
df['Column1'] = df['Column1'].astype(str)
df['Column2'] = df['Column2'].astype(str)

# RDS 데이터베이스 연결 설정
try:
    connection = mysql.connector.connect(
        host= 'your_rds_endpoint',  # rds 엔드포인트로 교체
        database='your_database_name',  # 데이터베이스 이름으로 교체
        user='your_user_name',  # 사용자 이름으로 교체
        password='your_password_name'  # 비밀번호로 교체
    )

    if connection.is_connected():
        print("Connected to RDS")

        # 데이터베이스에 데이터 적재
        for index, row in df.iterrows():
            cursor = connection.cursor()
            cursor.execute("""
                INSERT INTO output (Column1, Column2)
                VALUES (%s, %s)
            """, (row['Column1'], row['Column2']))
            # 데이터 형식에 맞게 변경
            connection.commit()

        print("Data uploaded successfully")

except Error as e:
    print(f"Error: {e}")
```

```
finally:
    if connection.is_connected():
        cursor.close()
        connection.close()
        print("RDS connection closed")
```

라. Lambda를 활용한 자동화

1. Lambda 함수 생성

2. Lambda 함수 트리거 생성



3. Lambda 함수 Layer 생성

4. Python 시각화 코드 작성 및 Image File S3 적재 자동화

```
import json
import boto3
import pandas as pd
import matplotlib.pyplot as plt
import os

s3 = boto3.client('s3')

def lambda_handler(event, context):
    # S3에서 객체 가져오기
    bucket_name = event['Records'][0]['s3']['bucket']['name']

    # 여러 개의 파일을 처리하기 위한 반복문
    for record in event['Records']:
        object_key = record['s3']['object']['key']

        # S3에서 CSV 파일 다운로드
        response = s3.get_object(Bucket=bucket_name, Key=object_key)
        data = pd.read_csv(response['Body'])

        # 데이터 시각화
        plt.figure(figsize=(10, 6))
        plt.plot(data['Column1'], data['Column2'])
        plt.title(f'Data Visualization for {object_key}')
        plt.xlabel('Column1')
        plt.ylabel('Column2')

        # 이미지 파일 저장
        image_path ='/tmp/' + object_key.split('.')[0] + '_vis.png'
        plt.savefig(image_path)
        plt.close()

        # S3에 이미지 업로드
        s3.upload_file(
            image_path,
            bucket_name,
```

```
            object_key.split('.')[0] + '_vis.png'
        )

    return {
        'statusCode': 200,
        'body': json.dumps('Images uploaded to S3 successfully!')
    }
```

- S3 버킷에 파일 업로드 또는 수정 시 이벤트를 감지하여 파일의 데이터를 읽어 그래프로 구현한 후 S3 버킷에 이미지 파일 적재