

AWS_실습_빅데이터9기_정승연

가. AWS 환경 구축하기

1. EC2 생성

Launch an instance

이름 및 태그 정보

1. 이름 설정

이름
big9_test_seungyeon

추가 태그 추가

▼ 애플리케이션 및 OS 이미지(Amazon Machine Image) 정보

AMI는 인스턴스를 시작하는 데 필요한 소프트웨어 구성(운영 체제, 애플리케이션 서버 및 애플리케이션)이 포함된 템플릿입니다. 아래에서 찾고 있는 항목이 보이지 않으면 AMI를 검색하거나 찾아보세요.

Q 수천 개의 애플리케이션 및 OS 이미지를 포함하는 전체 카탈로그 검색

최근 사용 Quick Start 2. os 설정

Amazon Linux macOS Ubuntu Windows Red Hat SUSE Linux

더 많은 AMI 찾아보기
AWS, Marketplace 및 커뮤니티의 AMI 포함

Amazon Machine Image(AMI)

Ubuntu Server 24.04 LTS (HVM), SSD Volume Type
ami-040c33c6a51fd5d96 (64비트(x86)) / ami-096099377d8850a97 (64비트(Arm))

프리 티어 사용 가능

요약

인스턴스 개수 | 정보
1

소프트웨어 이미지(AMI)
Canonical, Ubuntu, 24.04, amd64...더 보기
ami-040c33c6a51fd5d96

가상 서버 유형(인스턴스 유형)
t2.micro

방화벽(보안 그룹)
새 보안 그룹

스토리지(볼륨)
1개의 볼륨 - 8GiB

프리 티어: 첫 해에는 월별 프리 티어 AMI에 대한 t2.micro(또는 t2.micro를 사용할 수 없는 리전의 t3.micro) 인스턴스 사용량 750시간, 월별 퍼블릭 IPv4 주소 사용량 750시간, EBS 스토리지 30GiB, IO 2백만 개, 스냅샷 1GB, 인터넷 대역폭 100GB가 포함됩니다.

취소 인스턴스 시작

미리 보기 코드

▼ 인스턴스 유형 정보 | 초연 받기

3. 인스턴스 유형 선택

인스턴스 유형

t2.medium

패밀리: t2 2 vCPU 4 GiB 메모리 현재 세대: true

온디맨드 Windows 기본 요금: 0.0756 USD 시간당

온디맨드 RHEL 기본 요금: 0.0864 USD 시간당

온디맨드 SUSE 기본 요금: 0.1576 USD 시간당

온디맨드 Linux 기본 요금: 0.0576 USD 시간당

모든 세대

인스턴스 유형 비교

소프트웨어가 사전 설치된 AMI에는 추가 비용이 적용됩니다.

▼ 키 페어(로그인) 정보

키 페어를 사용하여 인스턴스에 안전하게 연결할 수 있습니다. 인스턴스를 시작하기 전에 선택한 키 페어에 대한 액세스 권한이 있는지 확인하세요. 4. 키 페어 선택 (없는 경우 생성)

키 페어 이름 - 필수

선택

새 키 페어 생성

소프트웨어 이미지(AMI)
Canonical, Ubuntu, 24.04, amd64...더 보기
ami-040c33c6a51fd5d96

가상 서버 유형(인스턴스 유형)
t2.medium

방화벽(보안 그룹)
새 보안 그룹

스토리지(볼륨)
1개의 볼륨 - 8GiB

프리 티어: 첫 해에는 월별 프리 티어 AMI에 대한 t2.micro(또는 t2.micro를 사용할 수 없는 리전의 t3.micro) 인스턴스 사용량 750시간, 월별 퍼블릭 IPv4 주소 사용량 750시간, EBS 스토리지 30GiB, IO 2백만 개, 스냅샷 1GB, 인터넷 대역폭 100GB가 포함됩니다.

▼ 네트워크 설정 정보 편집

네트워크 정보
vpc-047649da2321a2d91

서브넷 정보
기본 설정 없음(가용 영역의 기본 서브넷)

퍼블릭 IP 자동 할당 정보
활성화

프리 티어 하용 범위를 벗어나는 경우 추가 요금이 적용됩니다.

방화벽(보안 그룹) 정보
보안 그룹은 인스턴스에 대한 트래픽을 제어하는 방화벽 규칙 세트입니다. 특정 트래픽이 인스턴스에 도달하도록 허용하는 규칙을 추가합니다.

☒ 보안 그룹 생성 ☐ 기존 보안 그룹 선택

다음 규칙을 사용하여 'launch-wizard-18(이)'라는 새 보안 그룹을 생성합니다.

☒ 다음에서 SSH 트래픽 허용
인스턴스 인공에 허용 위치 무관
0.0.0.0/0

☐ 인터넷에서 HTTPS 트래픽 허용
예를 들어 웹 서비스를 생성할 때 엔드포인트를 설정하려면

☐ 인터넷에서 HTTP 트래픽 허용
예를 들어 웹 서비스를 생성할 때 엔드포인트를 설정하려면

⚠ 소스가 0.0.0.0/0인 규칙은 모든 IP 주소에서 인스턴스에 액세스하도록 허용합니다. 알려진 IP 주소의 액세스만 허용하도록 보안 그룹을 설정하는 것이 좋습니다.

인바운드 보안 그룹 규칙

▼ 보안 그룹 규칙 1 (TCP, 22, 0.0.0.0/0) 제거

유형	정보	프로토콜	정보	포트 범위	정보
ssh		TCP		22	

소스 유형	정보	원본	정보	설명 - 선택 사항	정보
위치 무관		Q	CIDR, 접두사 목록 또는 보안 그룹	예: 관리자 데스크톱용 SSH	
			0.0.0.0/0		

⚠ 소스가 0.0.0.0/0인 규칙은 모든 IP 주소에서 인스턴스에 액세스하도록 허용합니다. 알려진 IP 주소의 액세스만 허용하도록 보안 그룹을 설정하는 것이 좋습니다.

보안 그룹 규칙 추가

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

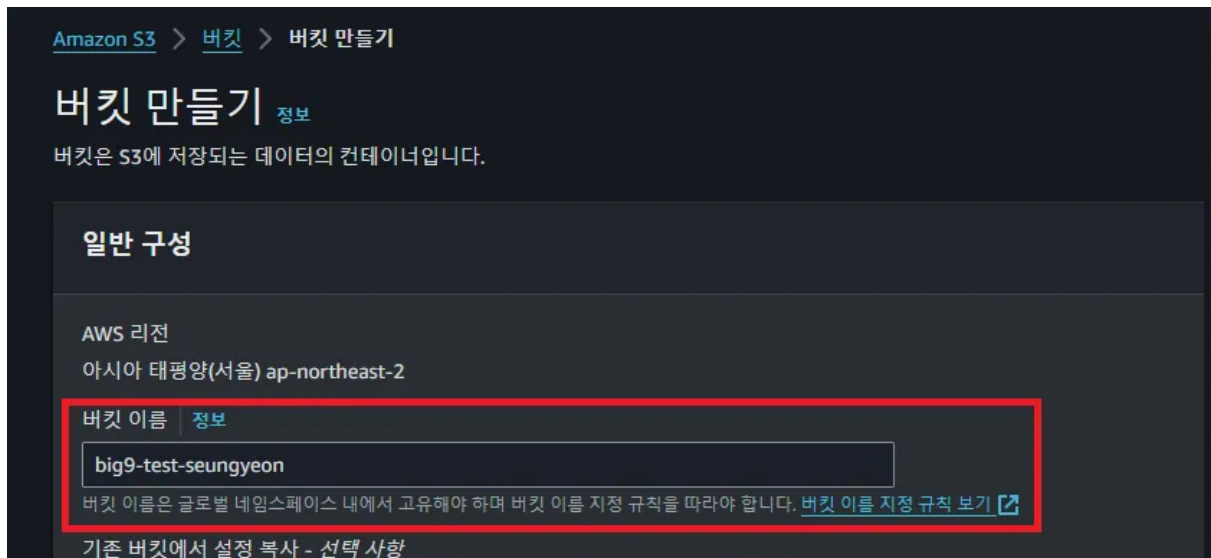
▼ 스토리지 구성 정보 고급

1x 32 GiB gp3 루트 볼륨 (암호화되지 않음)

❗ 프리 티어를 사용할 수 있는 고객은 최대 30GB의 EBS 범용(SSD)또는 마그네틱 스토리지를 사용할 수 있습니다.

- 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

- .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'

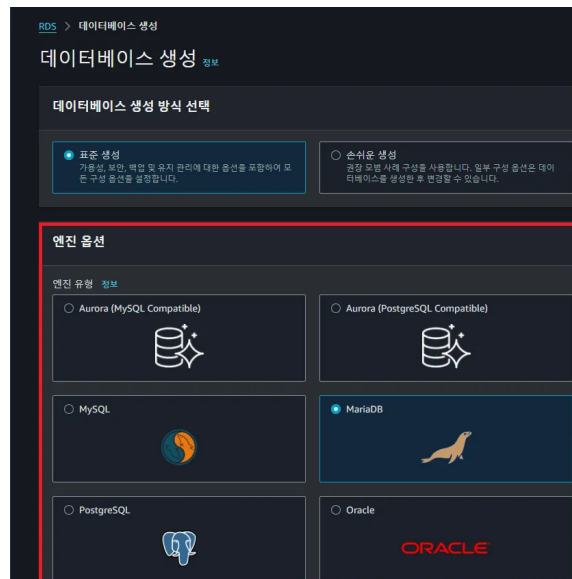
# S3에 파일 업로드
s3.upload_file(file_name, bucket_name, file_name)

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

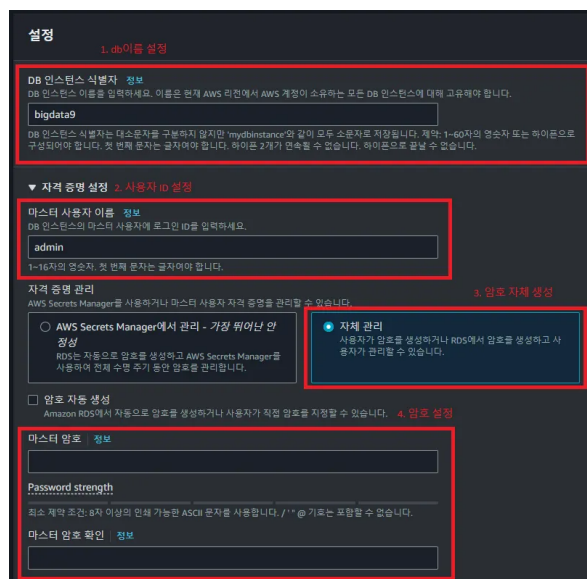
- 함수 실행 방법: `python3 your_file_name.py`

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

The screenshot shows the 'Create Function' page in the AWS Lambda console. The 'Function name' field is filled with 'big9-test-myfunc-seungyeon'. The 'Runtime' dropdown is set to 'Python 3.9'. The 'Architecture' dropdown is set to 'x86_64'. The 'Permissions' section is set to 'Create a new role with the minimum permissions needed to run this function'.

2. Lambda 함수 트리거 생성

The screenshot shows the 'Function Overview' page in the AWS Lambda console. The function name is 'big9-test-func-seungyeon'. The 'Layers' section shows '(2)' layers. The 'Triggers' section shows an S3 trigger. A red box highlights the '+ 트리거 추가' button.

3. Lambda 함수 Layer 생성

계층 정보						원본	Add a layer
병합 주문	이름	계층 버전	호환 런타임	호환 아키텍처	버전 ARN		
1	Klayers-p39-matplotlib	1	python3.6, python3.7, python3.8, python3.9	-	arn:aws:lambda:ap-northeast-2:770693421928:layer:Klayers-p39-matplotlib:1		
2	Klayers-p39-pandas	22	python3.9	x86_64	arn:aws:lambda:ap-northeast-2:770693421928:layer:Klayers-p39-pandas:22		

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