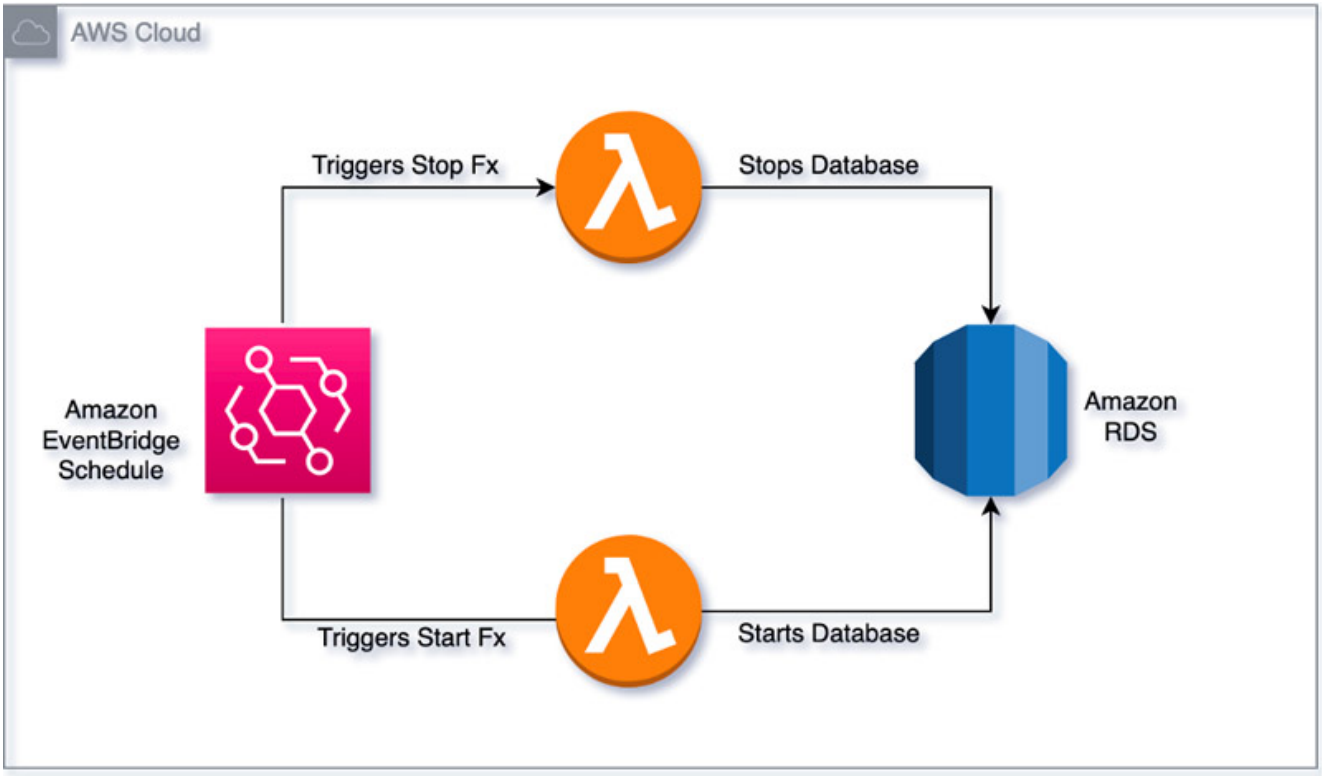
[AWS] Lambda + EventBridge 를 통한 RDS 데이터베이스를 자동으로 중지 및 실행하기

RDS 인스턴스는 DB 인스턴스 시간, 프로비저닝된 스토리지, 백업 스토리지, I/O 요청, 프로비저닝된 IOPS 및 데이터 전송에 대해 청구됩니다. 사용하지 않는 리소스들을 중지하지 않을 경우 지속적으로 비용이 발생하며 이 문서에서는 Amazon EventBridge와 AWS Lambda의 조합을 사용하여 일정에 따라 **모든 지역**에 대하여 미사용 RDS 인스턴스를 중지/시작하기 위한 자동화된 솔루션을 구현하는 방법을 살펴보겠습니다.



단계 :

1. 사전 구성

2. IAM 정책 및 역할 생성

3. Lambda 함수 생성

4. EventBridge 규칙 생성

5. 테스트

1. 사전 구성

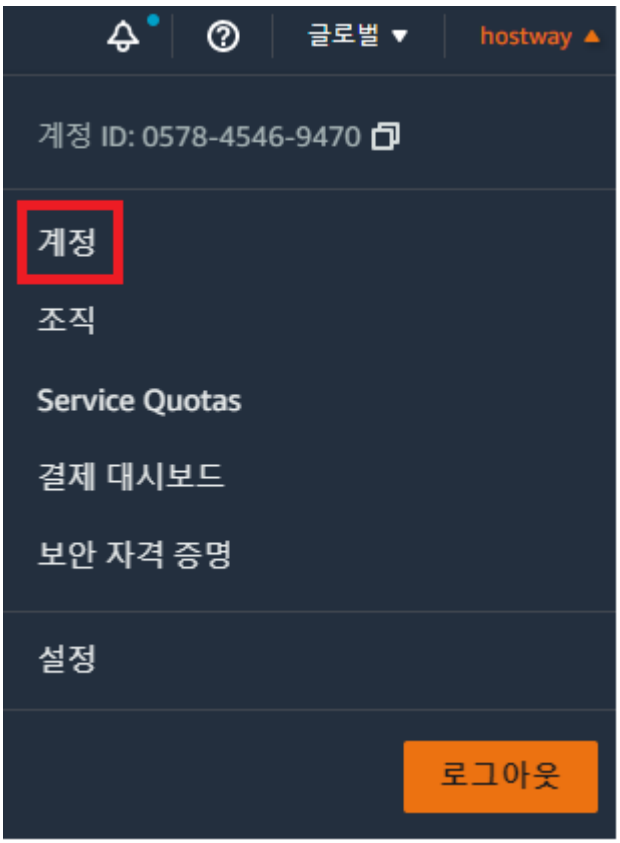
AWS 서비스에 프로그래밍 방식으로 연결하려면 엔드포인트를 사용해야 합니다.

그러나 일부 리전(opt-in)들은 기본적으로 비활성화 되어 있으며 해당 리전들의 DB 인스턴스 정보를 가져오기 위해선 활성화를 진행해주어야 합니다.

opt-in region

- 아프리카(케이프타운)
- 아시아 태평양(홍콩)
- 유럽(밀라노)
- 중동(바레인)
- 아시아 태평양(자카르타)

1. 권한이 있는 계정으로 콘솔에 로그인 후 우측 상단 계정 선택 – 계정- AWS 리전 비활성화 되어있는 5개의 리전들을 활성화 상태로 변경합니다.



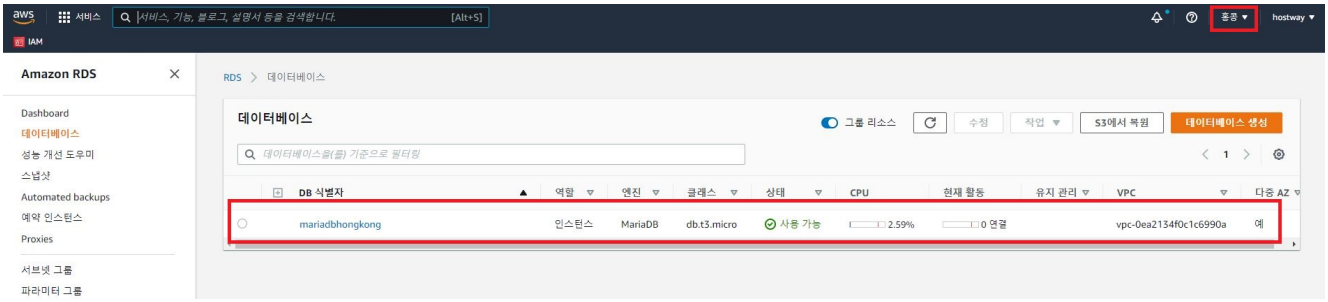
리전	상태	작업
중동 (바레인)	비활성화됨	활성화
아시아 태평양 (자카르타)	비활성화됨	활성화
아프리카 (케이프타운)	비활성화됨	활성화
아시아 태평양 (홍콩)	비활성화됨	활성화
유럽 (밀라노)	비활성화됨	활성화

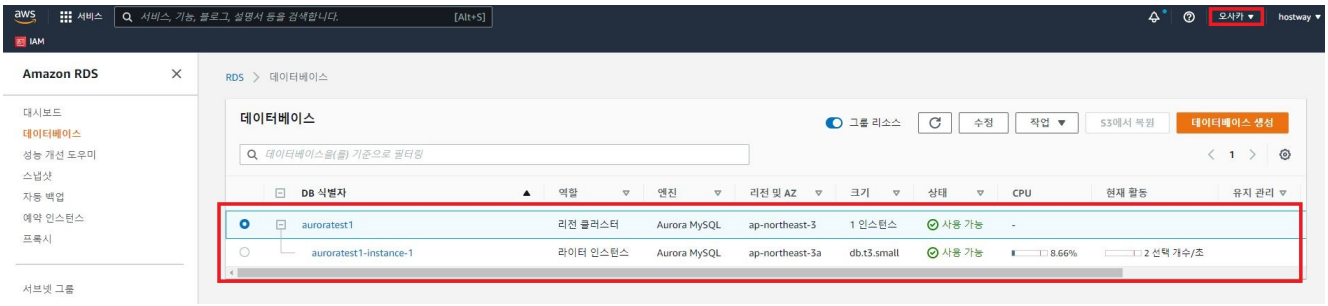
2. 테스트를 진행할 DB 인스턴스를 생성합니다. 총 3개 리전에 대하여 테스트 해보겠습니다.

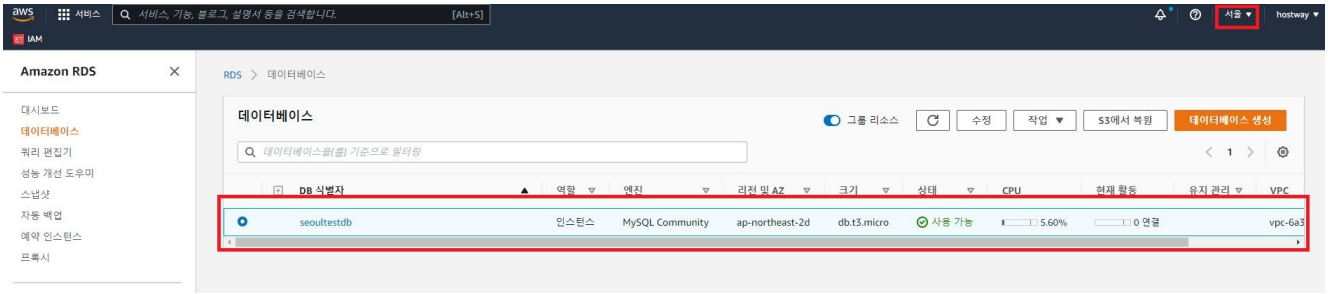
홍콩 : MariaDB

오사카 : AWS Aurora

서울 : Mysql

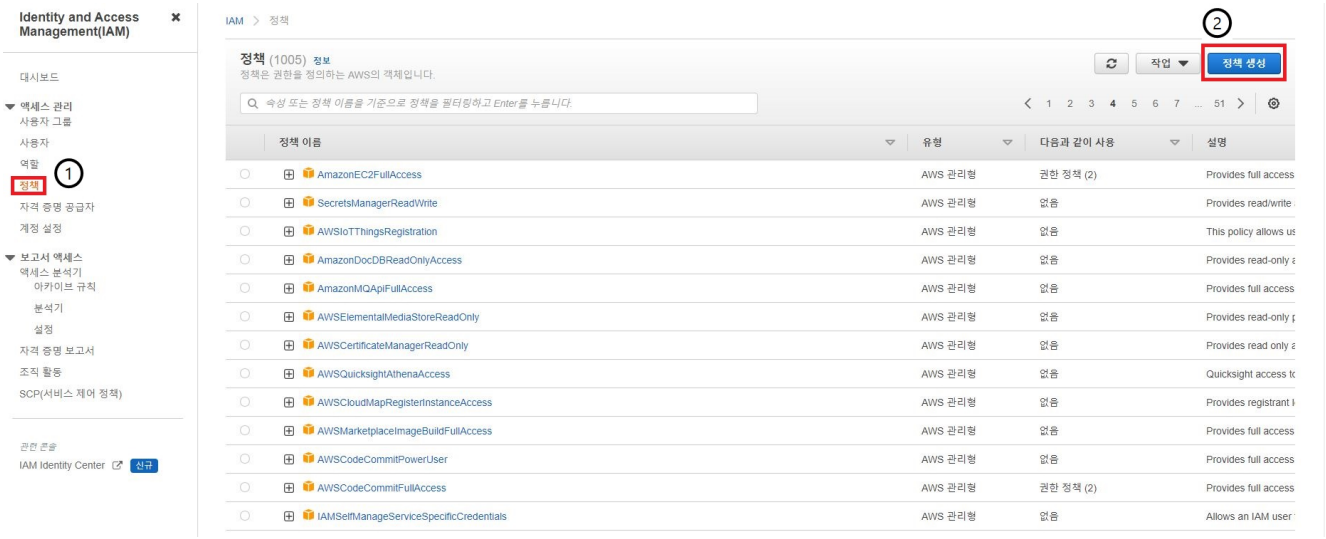




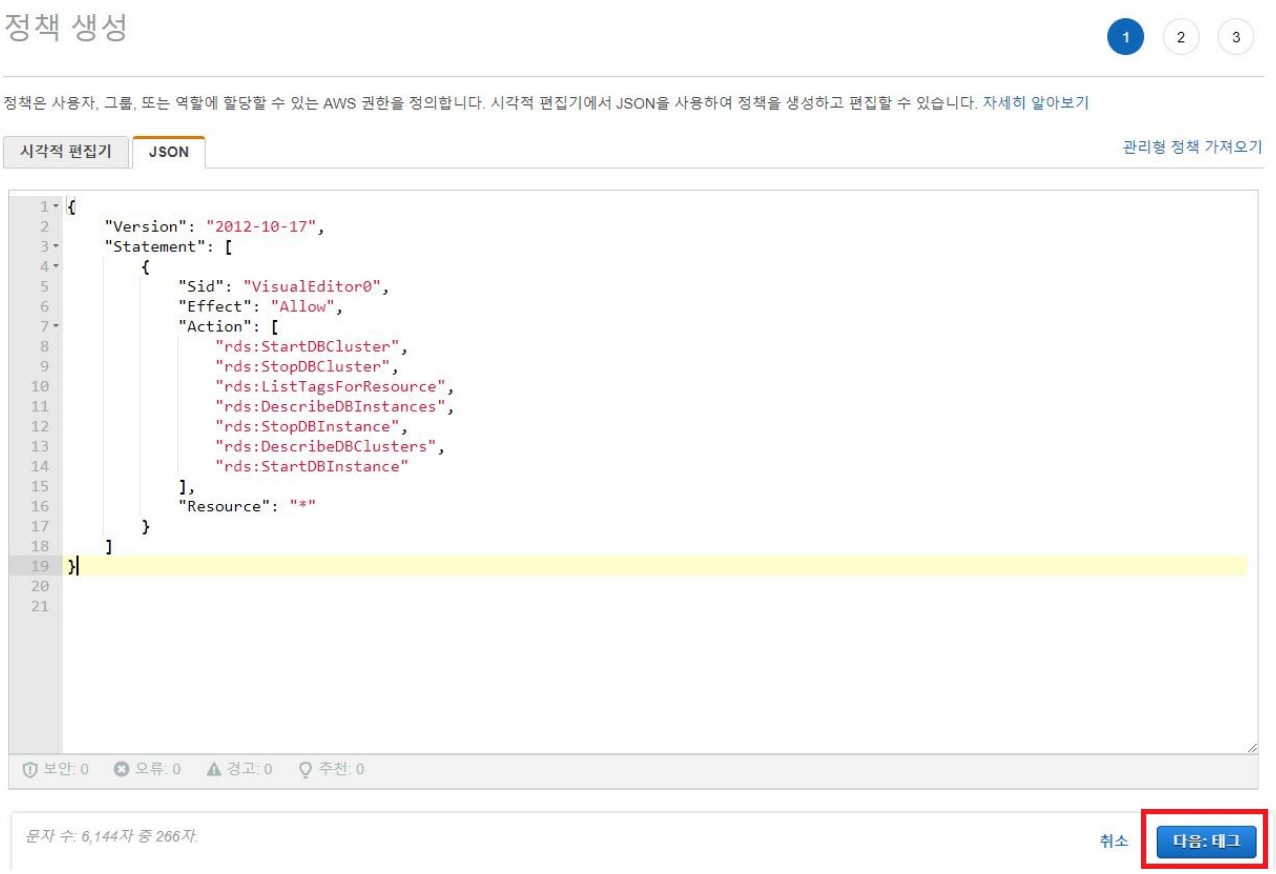


2. IAM 정책 및 역할 생성

1. RDS Stop/Start 역할을 위한 정책을 생성합니다. IAM 콘솔에서 정책 – 정책 생성



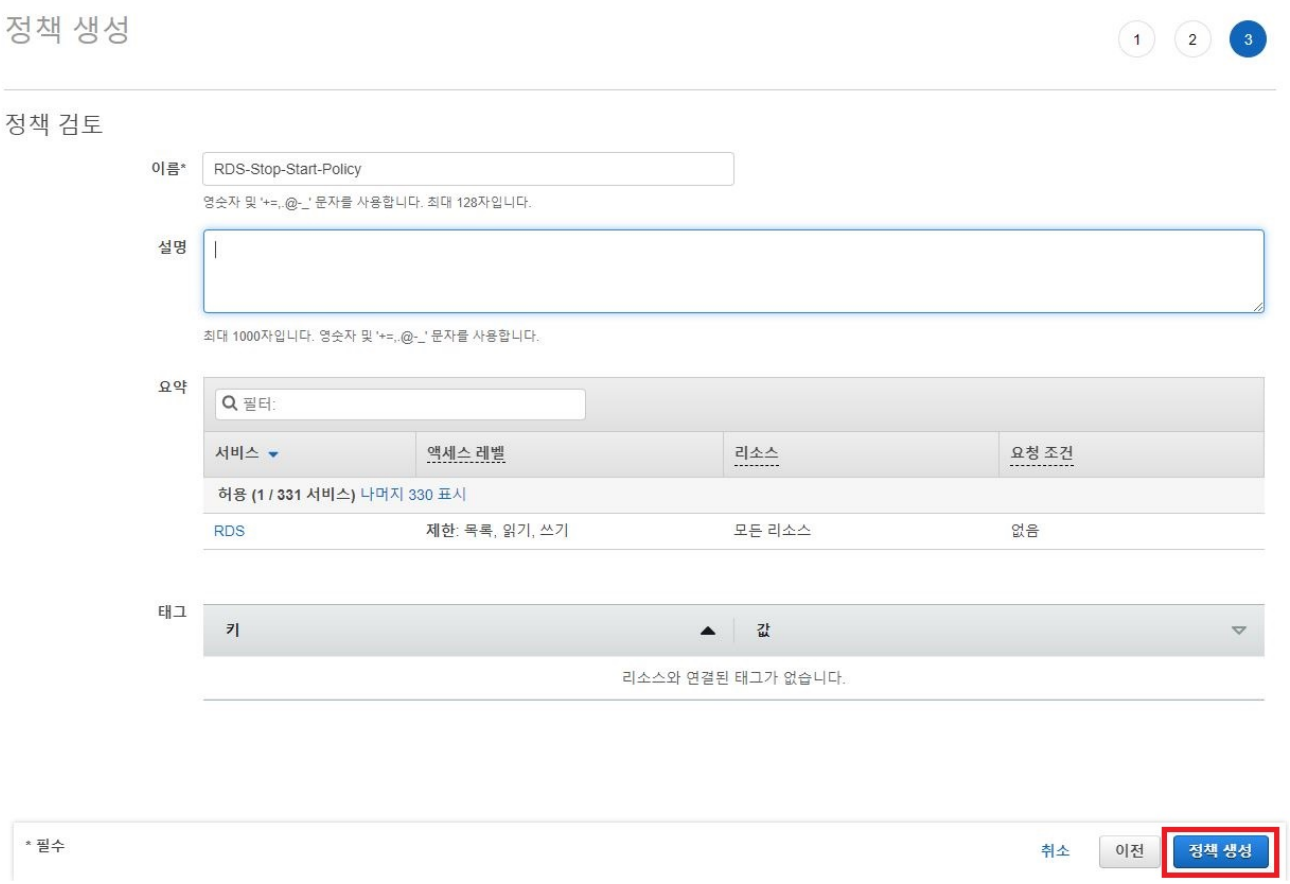
2. JSON 편집기에서 IAM 권한을 연결합니다.



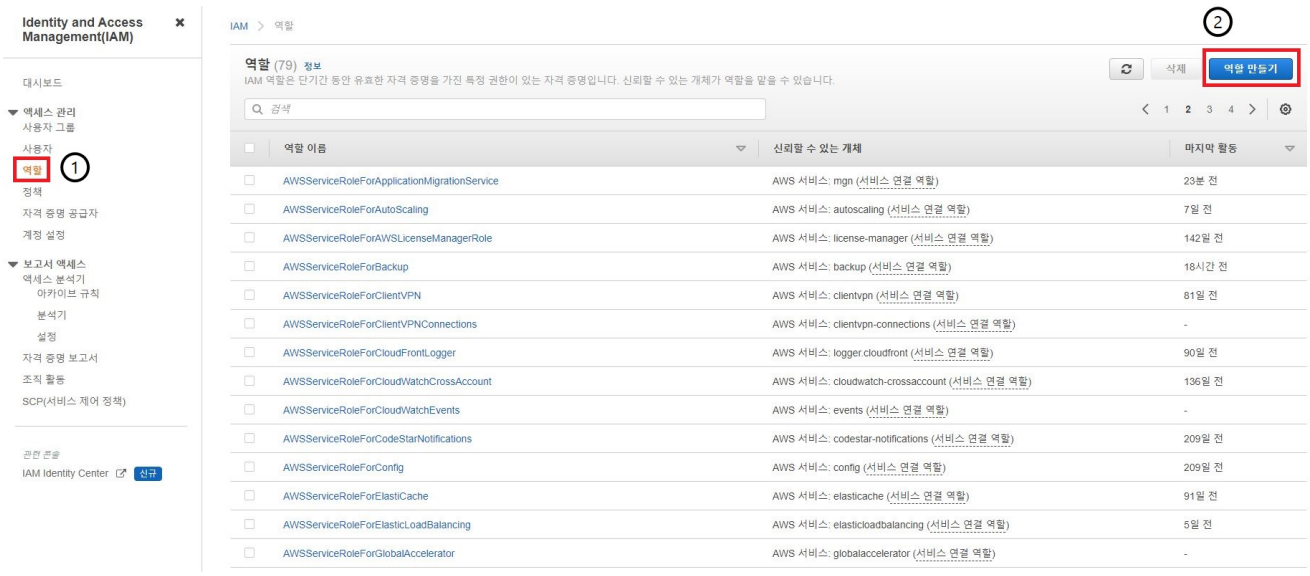

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
```

```
            "Effect": "Allow",
            "Action": [
                "rds:StartDBCluster",
                "rds:StopDBCluster",
                "rds:ListTagsForResource",
                "rds:DescribeDBInstances",
                "rds:StopDBInstance",
                "rds:DescribeDBClusters",
                "rds:StartDBInstance"
            ],
            "Resource": "*"
        }
    ]
}
```

3. RDS-Stop-Start-Policy 이름 입력 후 정책 생성을 클릭합니다.

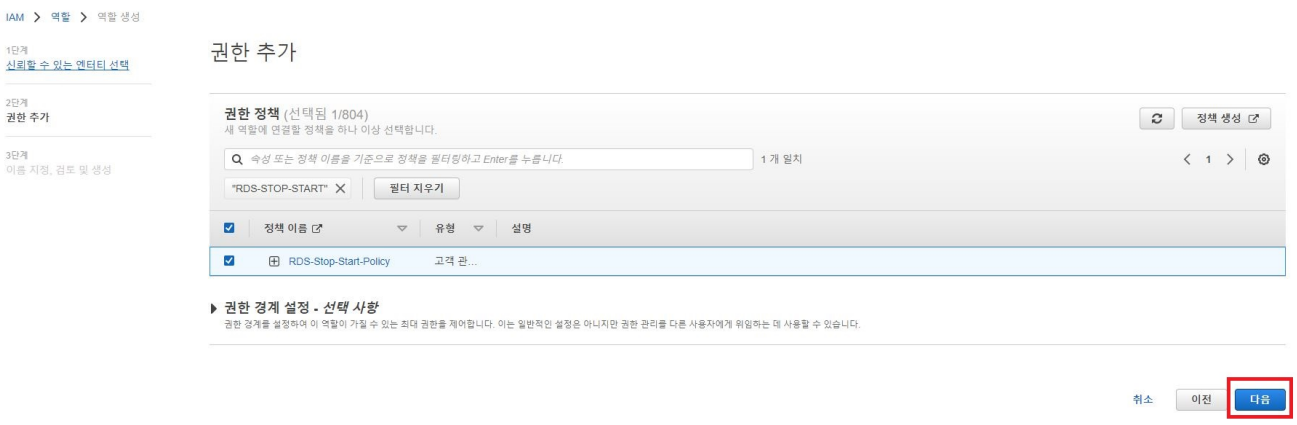


4. RDS Stop/Start 를 위한 역할을 생성합니다. IAM 콘솔에서 역할 – 역할 만들기

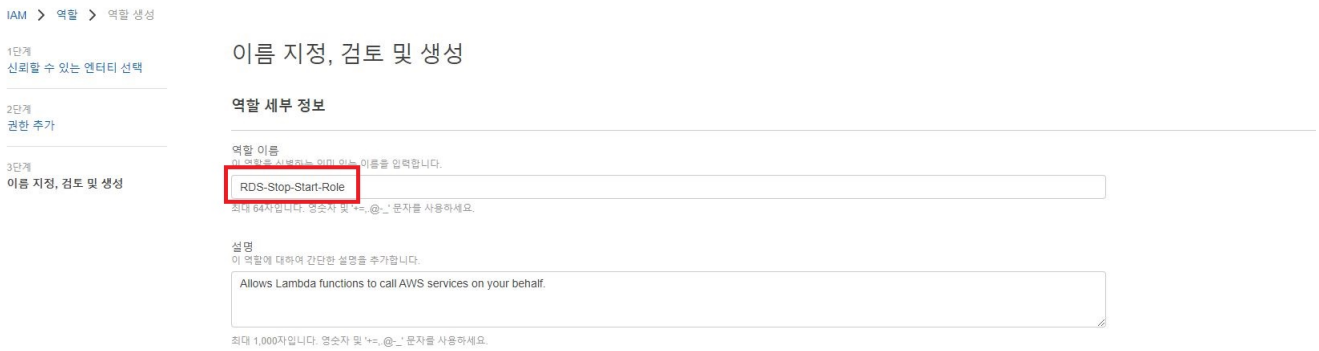


5. AWS Lambda를 신뢰할 수 있는 엔터티로 선택 후 다음을 클릭합니다.

6. 생성한 정책 RDS-Stop-Start-Policy 필터링 후 선택 – 다음을 클릭합니다.



7. RDS-Stop-Start-Role 이름 입력 후 역할 생성을 클릭합니다.

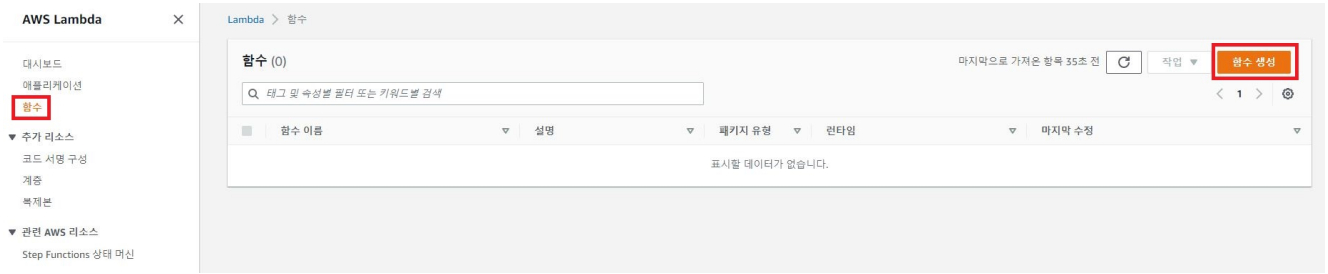


3. Lambda 함수 생성

AWS Lambda 함수를 생성하여 RDS DB 인스턴스를 Stop 시키도록 하겠습니다.

Region은 어느 지역이던 상관 없으나 여기선 서울 Region에 생성해보겠습니다.

1. RDS 인스턴스를 Stop할 Lambda 함수를 생성합니다. Lambda 콘솔 함수 – 함수 생성

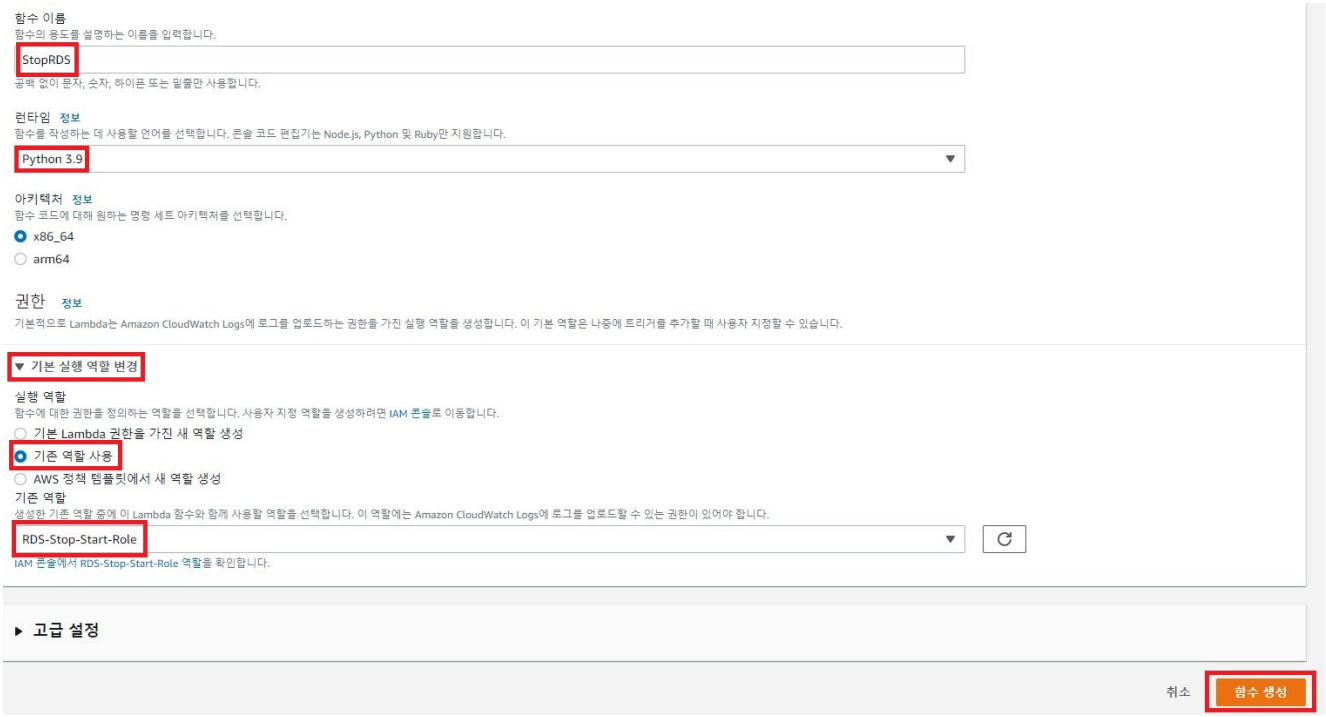


2. 다음과 같이 입력 후 함수를 생성합니다.

함수 이름 : StopRDS

런타임 : Python 3.9

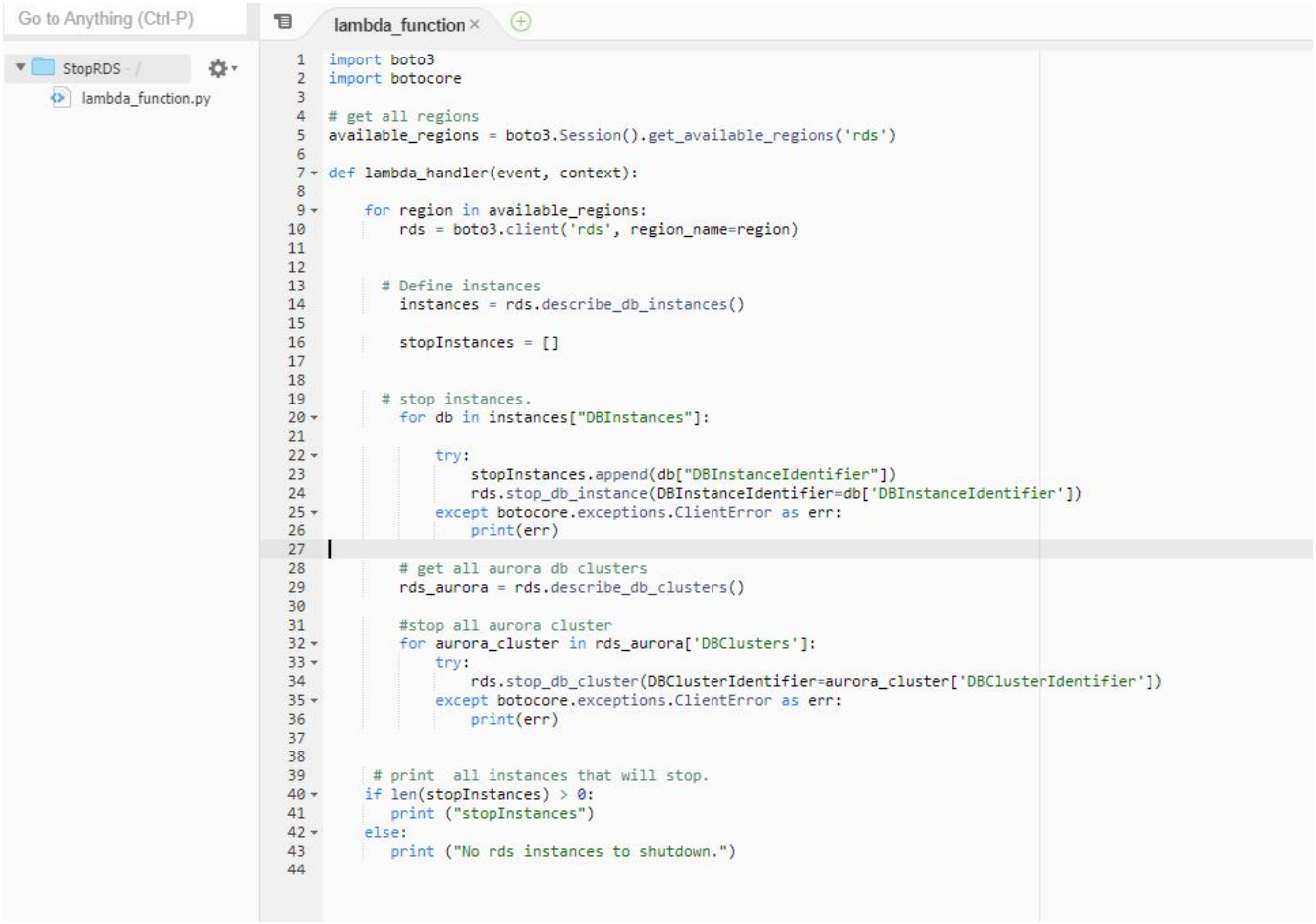
권한 : 기본 실행 역할 변경 – 기존 역할 사용 – 생성한 역할 RDS-Stop-Start-Role 선택



3. Boto3, Botocore를 이용하여 Python 코드 작성을 진행합니다.

Boto : AWS에서 Python 프로그래밍 언어의 사용을 개선하도록 설계된 소프트웨어 개발 키트(SDK) 이며 가장 최신 버전인 Boto3 까지 나와있습니다.

Botocore : AWS 도구에 대해 기본적인 액세스를 제공하며 low-level의 클라이언트 요청을 만들고 API에서 결과를 가져옵니다.




```
# StopRDS 함수 코드

import boto3
import botocore

# get all regions
available_regions                                                =
boto3.Session().get_available_regions('rds')

def lambda_handler(event, context):

    for region in available_regions:
        rds = boto3.client('rds', region_name=region)


      # Define instances
        instances = rds.describe_db_instances()

        stopInstances = []


```

```
      # stop instances.
        for db in instances["DBInstances"]:

            try:

 stopInstances.append(db["DBInstanceIdentifier"])

 rds.stop_db_instance(DBInstanceIdentifier=db['DBInstanceIdent
ifier'])
            except botocore.exceptions.ClientError as err:
                print(err)

        # get all aurora db clusters
        rds_aurora = rds.describe_db_clusters()

        # stop all aurora cluster
        for aurora_cluster in rds_aurora['DBClusters']:
            try:

rds.stop_db_cluster(DBClusterIdentifier=aurora_cluster['DBClus
terIdentifier'])
            except botocore.exceptions.ClientError as err:
                print(err)


     # print  all instances that will stop.
    if len(stopInstances) > 0:
       print ("stopInstances")
    else:
       print ("No rds instances to shutdown.")
```

stop_db_instance API는 Aurora 클러스터의 일부인 인스턴스를 종료할 수 없기 때문에 별도로 Aurora DB에 대하여 stop_db_cluster API 호출을 해주어야 합니다.

StartRDS 함수 작성을 하고자 할 경우 아래 코드를 입력하여 함수를 생성하면 됩니다.

```
# StartRDS 함수 코드

import boto3
import botocore

```

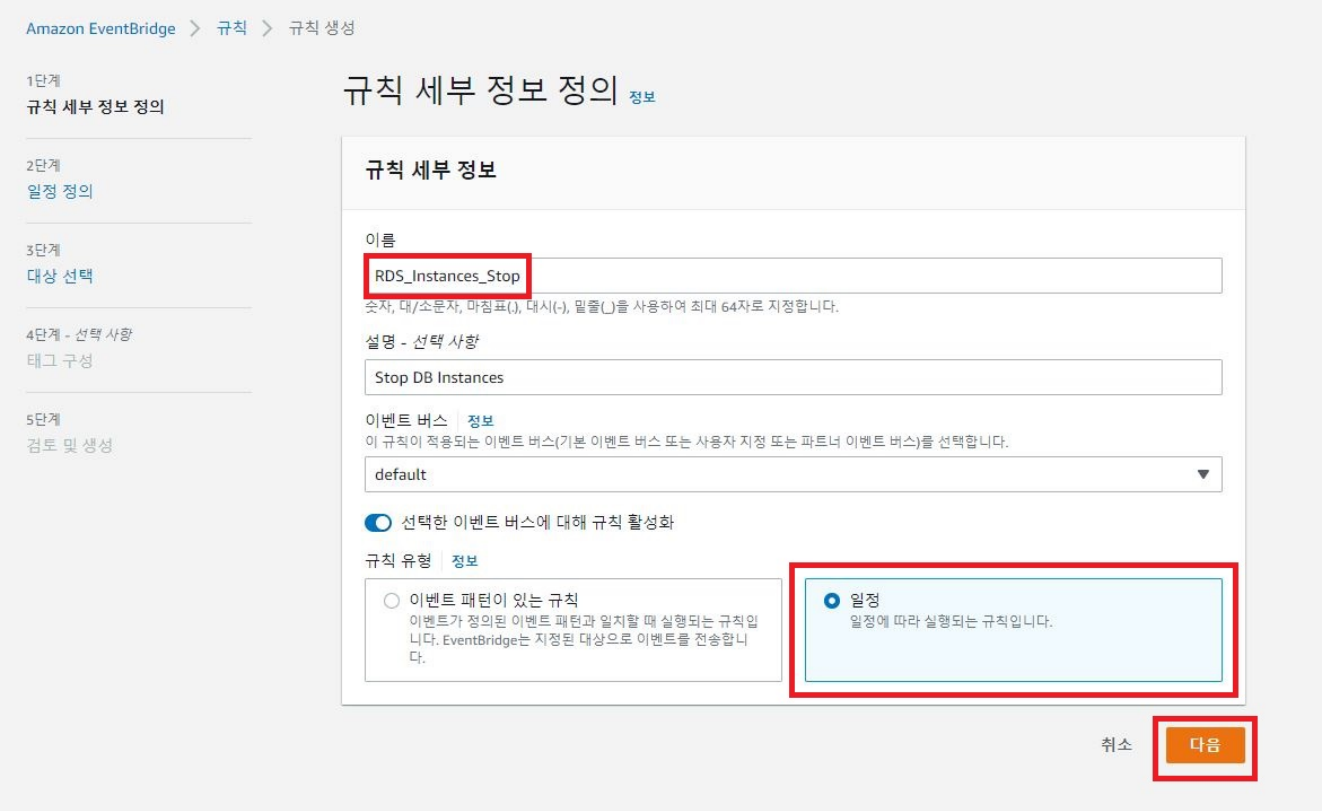
```
# get all regions
available_regions                                          =
boto3.Session().get_available_regions('rds')

def lambda_handler(event, context):

    for region in available_regions:
        rds = boto3.client('rds', region_name=region)


      # Define instances
        instances = rds.describe_db_instances()

        startInstances = []


      # start instances.
        for db in instances["DBInstances"]:

            try:

 startInstances.append(db["DBInstanceIdentifier"])

 rds.start_db_instance(DBInstanceIdentifier=db['DBInstanceIden
tifier'])
            except botocore.exceptions.ClientError as err:
                print(err)

        # get all aurora db clusters
        rds_aurora = rds.describe_db_clusters()

        # stop all aurora cluster
        for aurora_cluster in rds_aurora['DBClusters']:
            try:

 rds.start_db_cluster(DBClusterIdentifier=aurora_cluster['DBCl
usterIdentifier'])
            except botocore.exceptions.ClientError as err:
                print(err)


```

```
 # print  all instances that will stop.
if len(startInstances) > 0:
   print ("startInstances")
else:
   print ("No rds instances to start.")
```

4. EventBridge 규칙 생성

1. Amazon EventBridge 콘솔에서 규칙 – 규칙 생성을 클릭합니다.

2. 규칙 세부 정보에서 RDS_Instances_Stop 규칙 이름으로 입력 및 규칙 유형 일정 선택 후 다음을 클릭합니다.

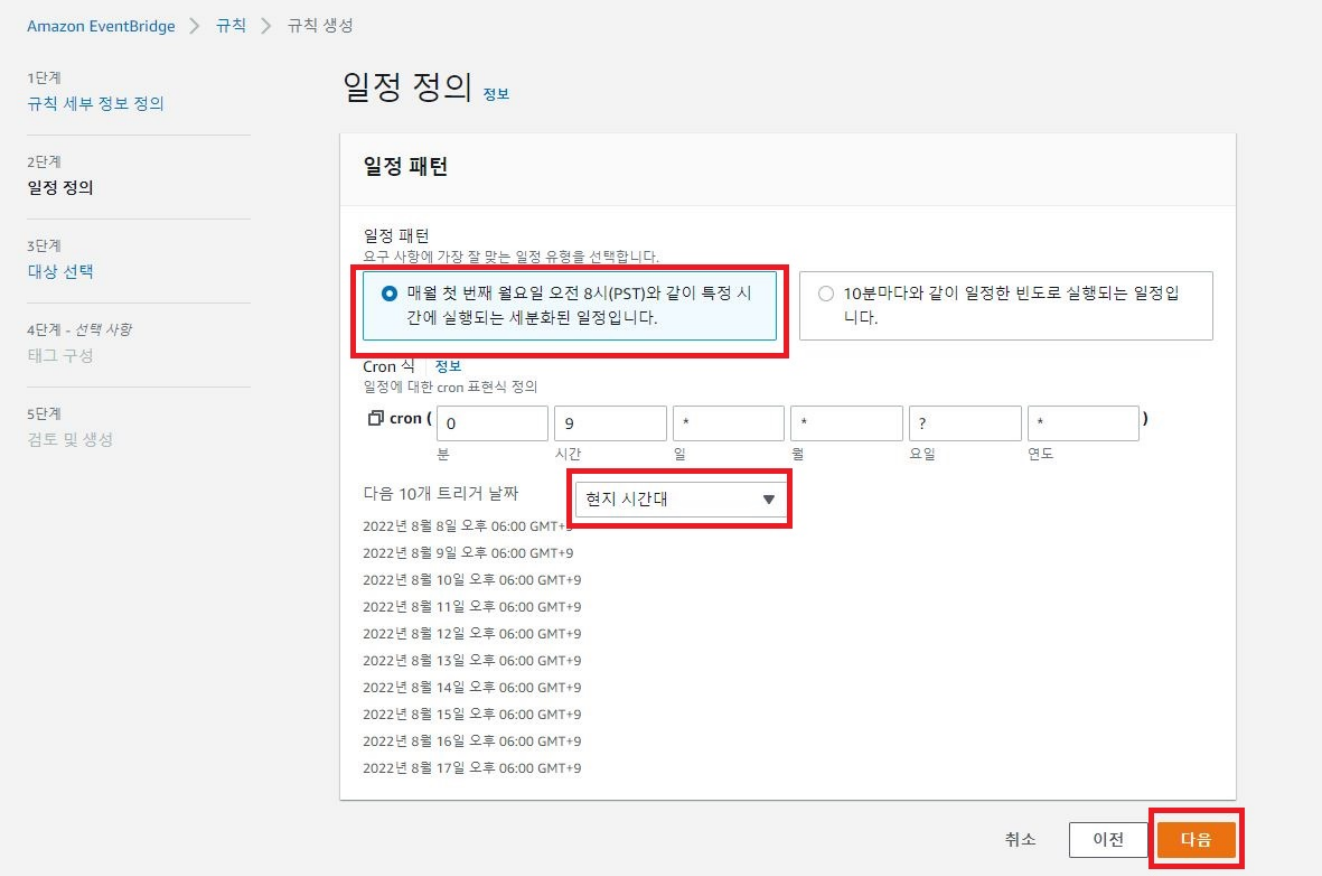


3. Cron 표현식을 작성합니다. 0 9 * * ? *

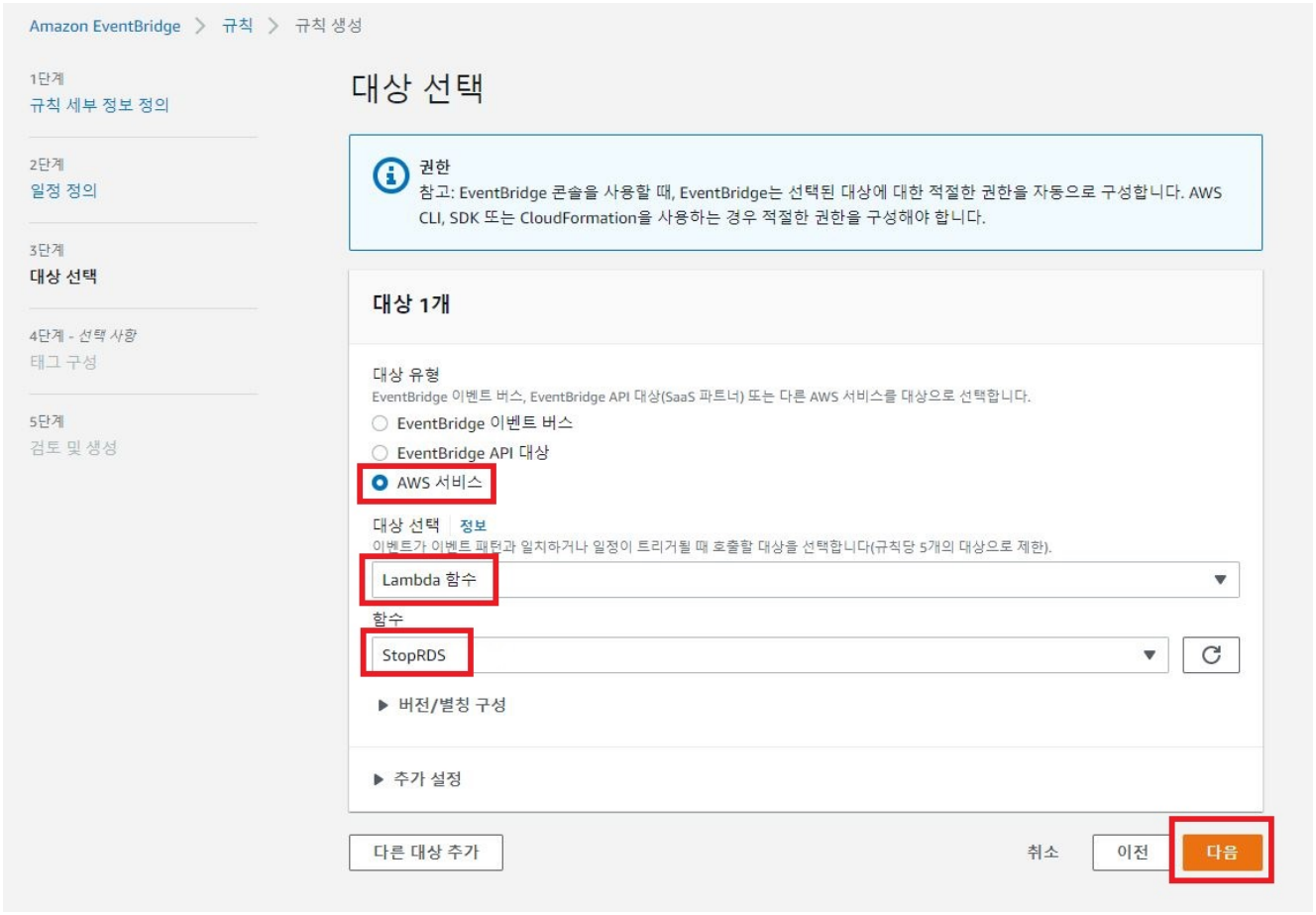
아래는 매일 오후 6시에 Lambda 함수를 트리거하는 표현식입니다.

자세한 cron 표현식은 공식 문서에 잘 나와 있습니다.

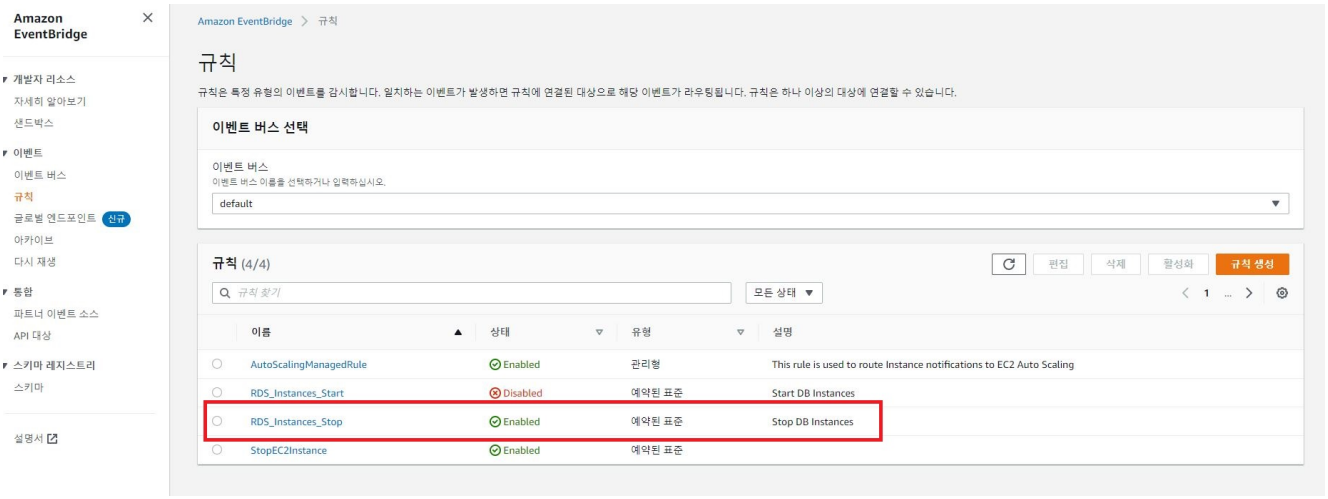
https://docs.aws.amazon.com/ko_kr/lambda/latest/dg/services-cloudwatchevents-expressions.html



4. AWS 서비스 선택 후 StopRDS 함수를 대상으로 지정한 뒤 다음을 클릭합니다.



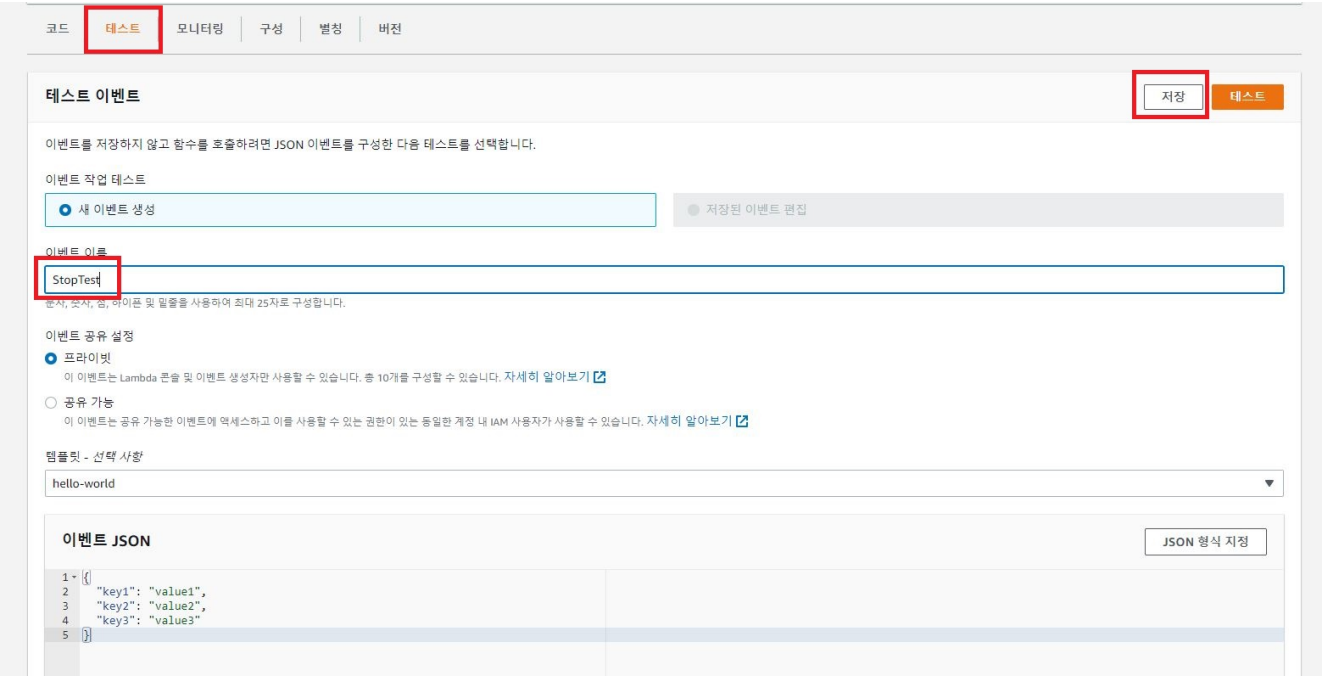
5. EventBridge 규칙이 정상적으로 생성되었습니다.



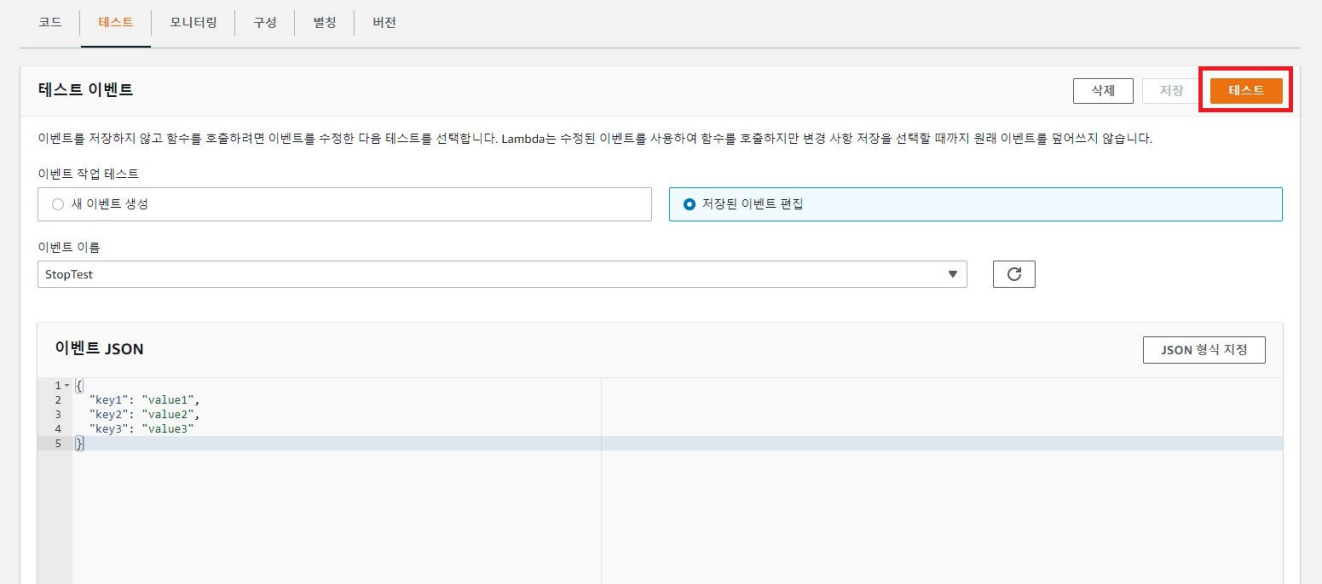
5. 테스트

생성한 Lambda 함수, EventBridge 규칙으로 3개 Region DB 들에 대하여 테스트를 진행해보겠습니다.

1. Amazon Lambda 콘솔에서 StopRDS 선택 후 테스트를 클릭, StopTest 이름을 작성한 뒤 기본 템플릿 hello-world 로 저장합니다.

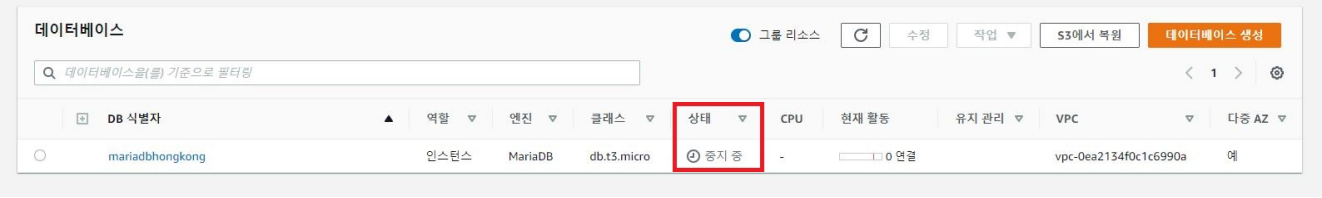


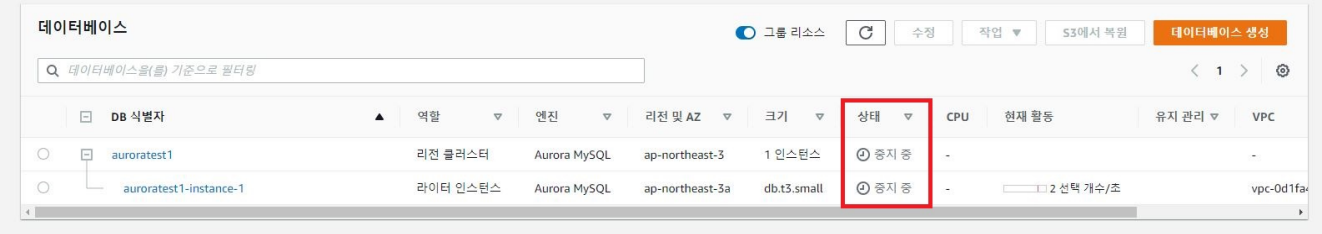
2. 저장된 템플릿 에서 테스트를 클릭합니다.

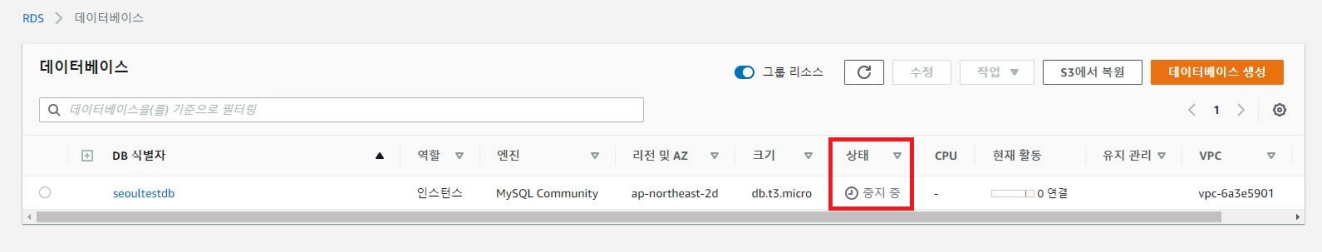


3. 테스트가 정상적으로 성공했다면 다음과 같은 화면을 볼 수 있습니다.

4. 이제 3개 리전에 대한 RDS 데이터베이스를 확인해보겠습니다.







서울 리전 Mysql 인스턴스, opt-in 리전인 홍콩 Mssql 정상적으로 Stop 되고 있으며 오사카 리전 Aurora DB Cluster도 정상적으로 Stop 중인 상태를 확인할 수 있습니다.

감사합니다.