**Yoojeong Choi**



**Data Life Cycle on AWS**

*Lab 0-2. Modify Aurora and navigate Load Generator*

**Table of Contents**

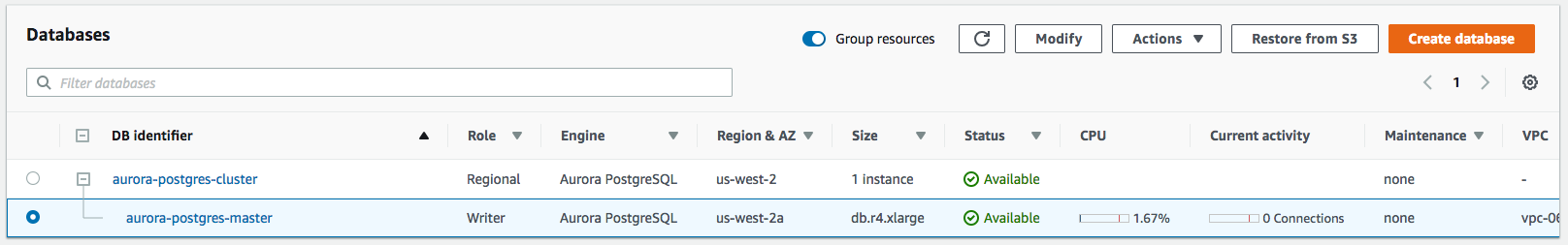
[**모니터링 대시보드 구성을 위한 RDS 설정** 3](#_Toc526250467)

[**Load Generator 확인 및 수행** 6](#_Toc526250468)

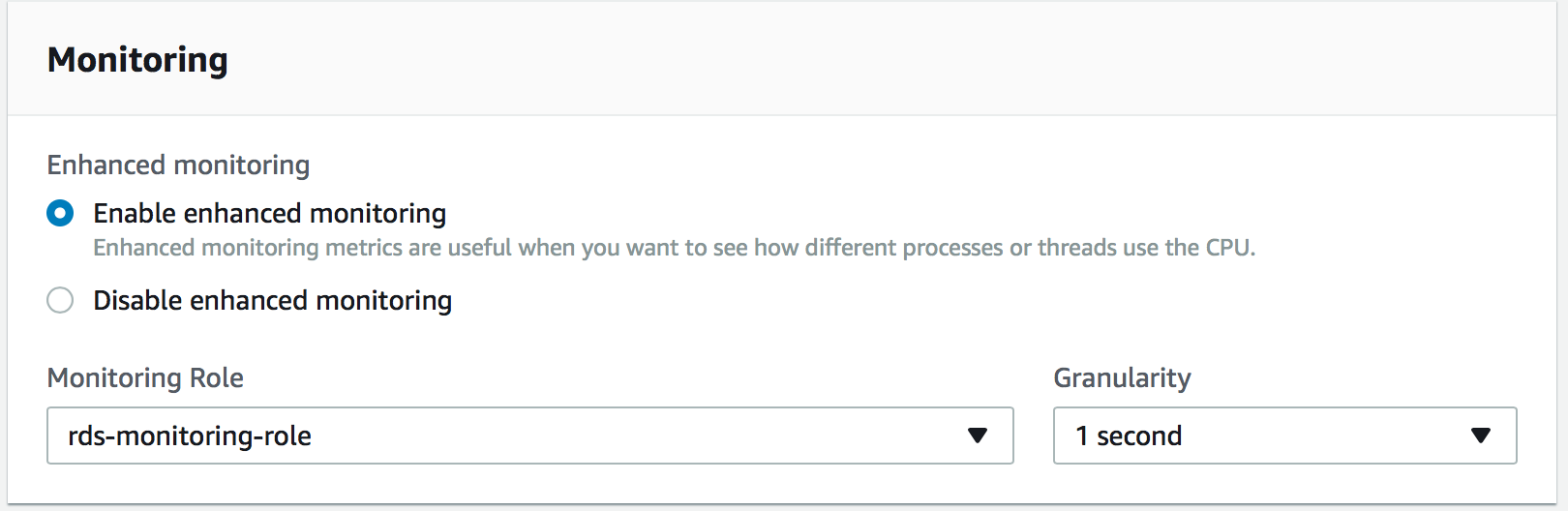
# **모니터링 대시보드 구성을 위한 RDS 설정**

이번 단계에서는 이후 Elasticsearch Service를 활용한 모니터링 대시보드 구성을 위한 RDS 설정을 진행합니다. 모니터링을 위해 Enhanced Monitoring 및 Performance Insight 를 설정합니다.

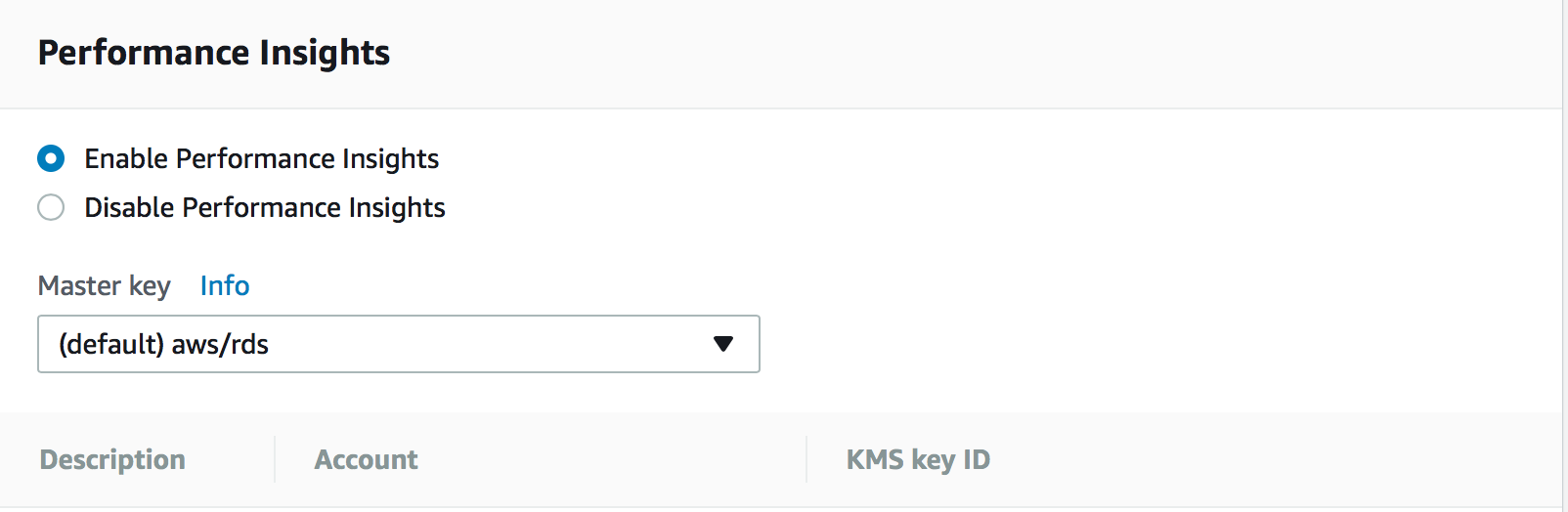
1. AWS Management Console 에서 RDS 서비스로 이동합니다.
2. RDS 서비스 화면에서 ‘aurora-postgres-master’를 선택 후 **Modify**를 선택합니다.



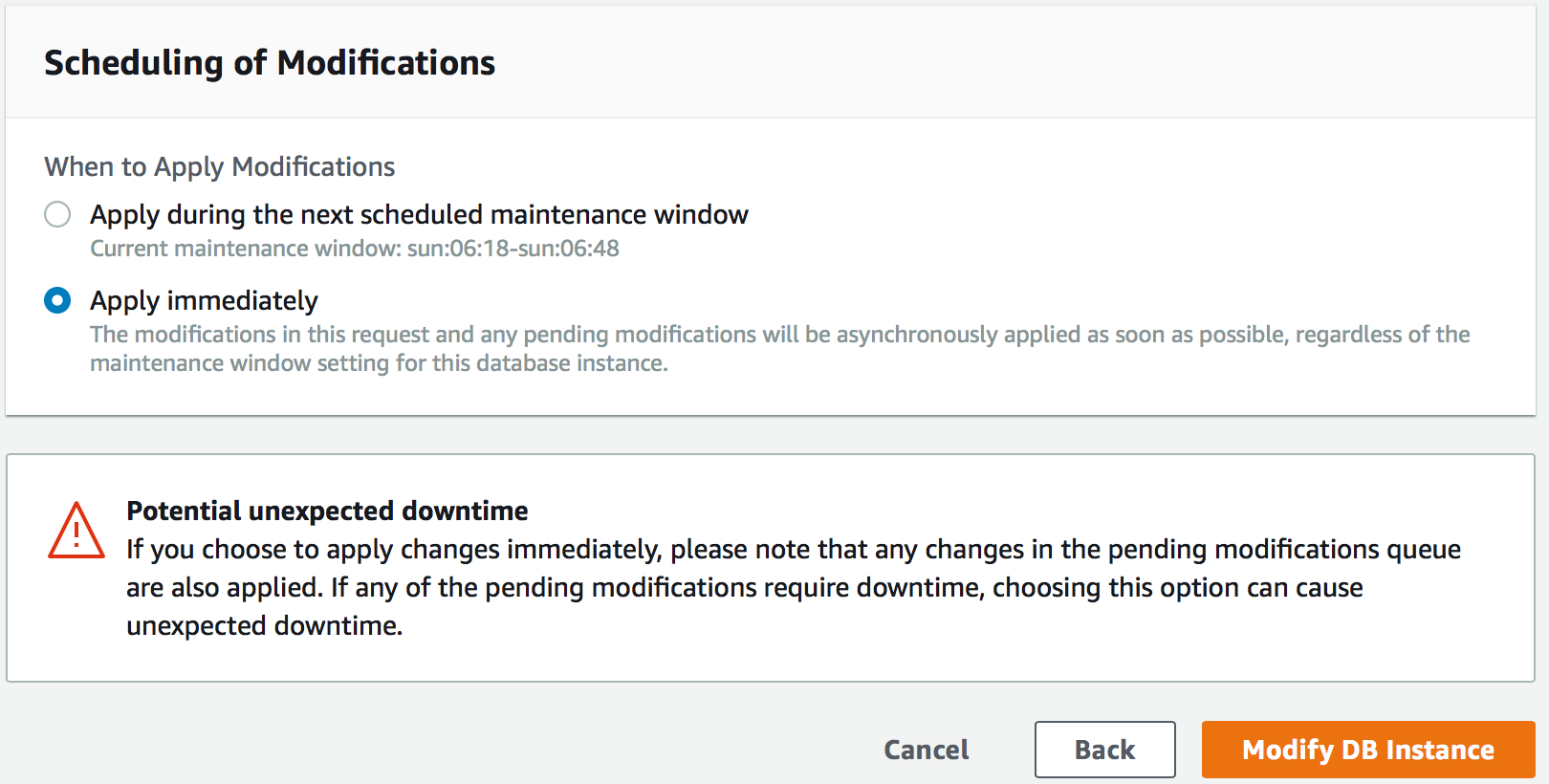
1. Monitoring 섹션의 **Enable enhanced monitoring** 선택 후, Monitoring Role은 rds-monitoring-role선택, **Granularity**는 원하는 수집 간격을 선택합니다.



1. Performance Insights섹션의 **Enable Performance Insights**를 선택 후, KMS의 key를 선택합니다. (특별히 지정하지 않는 경우, default 선택)



1. **Continue** 클릭 하고 다음화면에서, **Apply immediately**를 선택하고 **Modify DB Instance**를 클릭합니다.



# **Load Generator 확인 및 수행**

1. AWS Management Console 에서 EC2 서비스로 이동합니다.
2. CloudFormation으로 생성된 ‘datalab-load-generatorInstance’ 인스턴스의 Public IP확인 후 EC2에 접속합니다.

|  |
| --- |
| 186590d03cf1:Downloads yoojeong$ ssh -i oregon.pem ec2-user@54.186.166.0  Last login: Thu May 3 05:23:51 2018 from 205.251.233.176  \_\_| \_\_|\_ )  \_| ( / Amazon Linux 2 AMI  \_\_\_|\\_\_\_|\_\_\_|  https://aws.amazon.com/amazon-linux-2/  [ec2-user@ip-10-150-2-146 ~]$ |

1. Dynamodb event 테이블의 데이터를 로딩합니다. (총 5건)

|  |
| --- |
| [ec2-user@ip-10-150-1-248 .aws]$ aws dynamodb batch-write-item --request-items <file:///fs1/scripts/data/batch-write-event-table.json>  {  "UnprocessedItems": {}  }  [ec2-user@ip-10-150-1-248 .aws]$ aws dynamodb scan --table-name=event --select COUNT  {  "Count": 5,  "ScannedCount": 5,  "ConsumedCapacity": null  }  [ec2-user@ip-10-150-1-248 .aws]$ |

1. Load Generator 폴더 구조는 아래와 같습니다.

* /fs1/bin/ : 실행 Shell 파일
* /fs1/scripts/python/ : 워크로드 application python파일
* /fs1/Log/: 워크로드별 Log 파일

1. Aurora 접속 정보를 Update 해야합니다. **vi /fs1/scripts/python/database.ini** 를 통해 구성을 업데이트 합니다. 다른 정보는 그대로 두고, host 정보만 ‘aurora-postgres-master’의 endpoint로 변경하고 저장합니다.

|  |
| --- |
| [postgresql]  host=[**Aurora Endpoint**]  database=postgresDB  user=postgres\_dba  password=postgres123 |

1. Load generator 실행파일을 살펴보기 위해 **cat /fs1/bin/run\_all.sh** 을 수행합니다.

Filename 별 워크로드의 session개수와 think time을 설정할 수 있으며, 워크로드는 아래와 같습니다.

[[**RDS-Aurora PostgreSQL**]

* [SELECT] query\_item\_detail : 특정 item 상세 조회
* [SELECT] query\_item\_list : 특정 category item 리스트 조회
* [SELECT] query\_my\_info : 나의 정보 조회(구매건수, 쿠폰 보유 수)
* [SELECT] query\_sub\_categories : sub category 리스트 조회
* [INSERT] purchase: 구매
* [SELECT] query\_top\_10\_orders : 구매량 기준 상위 top 10 주문 조회
* [GET] query\_sub\_categories\_cache : sub category 리스트 조회 (**Elasticache Redis 적용**)

[[**NoSQL-DynamoDB**]

* [PUT] ddb\_apply\_event : event 응모
* [SCAN] ddb\_scan\_events : 현재 오픈되어있는 event 조회
* [SCAN] ddb\_scan\_events\_dax : 현재 오픈되어 있는 event 조회 **(DAX 조회**)

|  |
| --- |
| #!/bin/bash  #sh run\_generator.sh file\_name session\_cnt think\_time"  #Aurora  sh run\_generator.sh query\_item\_detail 2 1 &  sh run\_generator.sh query\_item\_list 2 1 &  sh run\_generator.sh query\_my\_info 3 1 &  sh run\_generator.sh query\_sub\_categories 1 1 &  sh run\_generator.sh purchase 1 1 &  sh /fs1/bin/run\_generator.sh query\_top\_10\_orders 2 1 &  #query Elasticache  #sh run\_generator.sh query\_sub\_categories\_cache 1 1 &  #DDB  sh run\_generator.sh ddb\_apply\_event 1 1 &  sh run\_generator.sh ddb\_scan\_events 1 1 &  #query Dax  #sh run\_generator.sh ddb\_scan\_events\_dax 1 1 & |

1. 별다른 설정 변경 없이 스크립트를 수행하도 무방합니다. **sh /fs1/bin/run\_all.sh** 를 수행하여 Load Generator를 실행합니다. 실행 확인은 **ps -ef | grep python** 명령어를 통한 Python 프로세스 개수를 통해 확인할 수 있습니다.

|  |
| --- |
| [ec2-user@ip-10-150-1-248 bin]$ sh /fs1/bin/run\_all.sh  query\_item\_detail 2 processes have forked  query\_item\_list 2 processes have forked  query\_sub\_categories 1 processes have forked  query\_my\_info 3 processes have forked  purchase 1 processes have forked  query\_top\_20\_orders 2 processes have forked  ddb\_apply\_event 1 processes have forked  ddb\_scan\_events 1 processes have forked  [ec2-user@ip-10-150-1-248 bin]$ ps -ef|grep python  ec2-user 4851 1 0 01:44 pts/3 00:00:00 python /fs1/scripts/python/query\_item\_detail.py 1  ec2-user 4852 1 0 01:44 pts/3 00:00:00 python /fs1/scripts/python/query\_item\_list.py 1  ec2-user 4853 1 0 01:44 pts/3 00:00:00 python /fs1/scripts/python/query\_item\_detail.py 1  ec2-user 4854 1 0 01:44 pts/3 00:00:00 python /fs1/scripts/python/query\_item\_list.py 1  ec2-user 4855 1 0 01:44 pts/3 00:00:00 python /fs1/scripts/python/query\_my\_info.py 1  ec2-user 4856 1 0 01:44 pts/3 00:00:00 python /fs1/scripts/python/query\_sub\_categories.py 1  ec2-user 4857 1 0 01:44 pts/3 00:00:00 python /fs1/scripts/python/query\_my\_info.py 1  ec2-user 4858 1 0 01:44 pts/3 00:00:00 python /fs1/scripts/python/purchase.py 1  ec2-user 4859 1 0 01:44 pts/3 00:00:00 python /fs1/scripts/python/query\_my\_info.py 1  ec2-user 4860 1 0 01:44 pts/3 00:00:00 python /fs1/scripts/python/query\_top\_20\_orders.py 1  ec2-user 4861 1 0 01:44 pts/3 00:00:00 python /fs1/scripts/python/query\_top\_20\_orders.py 1  ec2-user 4862 1 0 01:44 pts/3 00:00:06 python /fs1/scripts/python/ddb\_apply\_event.py 1  ec2-user 4863 1 0 01:44 pts/3 00:00:04 python /fs1/scripts/python/ddb\_scan\_events.py 1  ec2-user 5827 5803 0 02:14 pts/8 00:00:00 grep --color=auto python |

1. 수행 Log를 확인합니다. (예: query\_my\_info) Log는 각 워크로드별로 폴더내 1분단위로 생성됩니다.

|  |
| --- |
| [ec2-user@ip-10-150-1-248 ~]$ cd /fs1/Log/queryMyInfo  [ec2-user@ip-10-150-1-248 queryMyInfo]$ tail -f 201805031035.log  coupon\_cnt:4, order\_cnt:490843: Execution time: 1.89766693115 seconds  coupon\_cnt:1, order\_cnt:491719: Execution time: 2.2637758255 seconds  coupon\_cnt:4, order\_cnt:895084: Execution time: 1.92225694656 seconds |

1. 실습 종료 시까지 해당 Load 프로세스는 그대로 유지합니다.