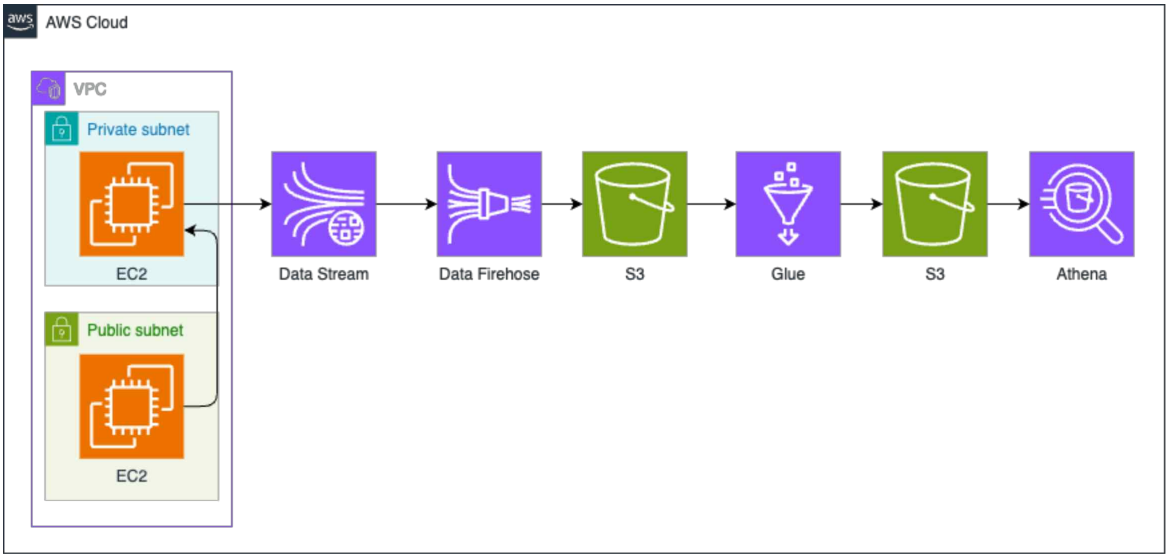
2025 평가경기기능경기대회 과제

직 종 명	클라우드컴퓨팅	과제명	ETL	과제번호	제 2과제
경기시간	4시간	비 번 호		심사위원 확인	(인)

1. 요구사항

요구사항 데이터 기반 의사결정을 강화하기 위해 배치 데이터 처리 파이프라인을 구축하고자 합니다. 트랜잭션 데이터와 액세스 로그를 수집하여 S3에 저장하고, 이 데이터를 심층적으로 분석하기 위한 배치 처리 시스템을 구현해야 합니다. 다양한 형식의 데이터를 통합하고 정규화하여 분석 가능한 형태로 변환해야 하며, 특히 비정형 데이터는 Glue의 커스텀 분류기를 활용하여 처리해야 합니다. 변환된 데이터는 S3에 저장하고 Athena를 통해 쿼리할 수 있는 환경을 구축해야 합니다.

다이어그램



Software Stack

AWS	개발언어/프레임워크
- VPC	- Golang
- EC2	
- Kinesis Series	
- Lambda	
- S3	
- Glue	
- Athena	

2. 선수 유의사항

1) 기계 및 공구 등의 사용 시 안전에 유의하시고, 필요 시 안전장비 및 복장 등을 착용하여 사고를 예방하여 주시기 바랍니다.

2) 작업 중 화상, 감전, 찰과상 등 안전사고 예방에 유의하시고, 공구나 작업도구 사용 시 안전보호구 착용 등 안전수칙을 준수하시기 바랍니다.

3) 작업 중 공구의 사용에 주의하고, 안전수칙을 준수하여 사고를 예방하여 주시기 바랍니다.

4) 경기 시작 전 가벼운 스트레칭 등으로 긴장을 풀어주시고, 작업도구의 사용 시 안전에 주의하십시오.

5) 과제에 명시된 인스턴스와 서비스에서 채점이 진행되므로, 인스턴스를 생성하지 않았거나 종료된 상태면 채점이 불가합니다.

6) 채점은AWS CLI를 통해 진행되므로, 필요한 리소스에 대한 접근이 가능하도록IAM 역할을 EC2 인스턴스에 할당해야 합니다.

7) 별도 언급이 없는 경우 ap-northeast-2 리전에 인프라를 생성하도록 합니다.

8) 과제에 명시된 이름과 태그 등을 정확히 사용해야 AWS CLI 채점이 올바르게 수행됩니다. 리소스 이름과 패키지 설치에 유의하여 작업하십시오.

9) 문제나 채점지에 <> 표시는 변수임을 의미하므로, 선수가 적절히 변경하여 사용해야 합니다.

10) Athena 쿼리를 위해 저장된 S3 데이터에 오류가 있으면 채점에 불이익을 받을 수 있습니다. 데이터 품질을 유지하도록 유의하십시오.

11) 과제에서 요구하는 모든 서비스 구성 요소(VPC, 서브넷, 보안 그룹, IAM 역할 등)를 정확히 구현하십시오, 보안 그룹 등의 설정이 잘못된 경우 채점이 불가할 수 있습니다.

3. 네트워크 구성

VPC 구성

- VPC CIDR : 10.1.0.0/16
- VPC Tag : Name = retail-vpc
- Internet Gateway Tag : Name = retail-igw

LogGenerator subnet A 구성 *서브넷 이름 뒤의 알파벳은 Availability Zone을 의미합니다.

- CIDR : 10.1.1.0/24
- Tag : Name=retail-log-a
- 외부 통신 : NAT G/W를 구성하여 인터넷 접근이 가능하도록 구성
- Route table Tag : Name=retail-log-a-rt
- NAT G/W Tag : Name=retail-natgw-a

Public subnet A 구성 *서브넷 이름 뒤의 알파벳은 Availability Zone을 의미합니다.

- CIDR : 10.1.2.0/24
- Tag : Name=retail-public-a
- 외부 통신 : Internet G/W 를 구성하여 인터넷을 접근
- Route table Tag : Name=retail-public-a-rt

4. Bastion 서버

EC2를 활용해 Bastion 서버를 구성합니다. bastion 서버의 접근을 위해서 TCP (SSH)를 사용합니다. bastion 서버는 외부에서 SSH 프로토콜만을 허용하도록 Security Group을 구성합니다.

- Instance type : t3.large
- AMI : Amazon Linux 2023
- Subnet : retail-public-a

- EC2 인스턴스에 EIP를 연결합니다.
- Tag Name=retail-bastion
- Security gourp name : retail-bastion-sg (SSH를 통해 접근 가능하도록 보안 그룹을 설정)
- EC2 IAM Role : 모든 리소스에 대해 full access를 가지는 role 을 생성하여 붙입니다.
  role 이름은 retail-app-role로 설정합니다.
- 설치 패키지 : aws-cli (AWS Command Line Interface)

5. 로그 생성 어플리케이션

로그 생성 어플리케이션은 retail-bastion 서버를 통해서만 접근 가능하며, 외부에서 직접적인 접근이 불가능하도록 Security Group을 구성합니다. 제공된 binary 파일(access-log-generator, transaction-log-generator)을 실행하여 각각 액세스 로그와 트랜잭션 로그를 생성합니다. 이 binary 파일들은 실행 시 자동으로 /data 폴더와 하위 디렉토리를 생성하여 각각 /data/access와 /data/transaction 경로에 로그 파일을 저장합니다. 생성된 로그는 Kinesis Data Stream으로 전송되어야 합니다. 시스템 재부팅 후에도 로그 생성이 자동으로 재개되도록 systemd 서비스로 구성합니다.

- Instance type: t3.large
- Subnet : retail-log-a
- Tag : Name=retail-log
- EC2 IAM Role : 모든 리소스에 대해 full access를 가지는 retail-app-role 을 붙입니다.
- Security gourp name : retail-log-sg
- 설치 패키지 : aws-kinesis-agent
- 제공된 바이너리 파일은 EC2 인스턴스의 /home/ec2-user/ 경로에 위치하도록 하며, 실행 권한을 부여해야 합니다.
- 로그 생성기가 시스템 재부팅 후에도 자동으로 실행되도록 각 바이너리 파일에 대한 systemd 서비스 파일(access-generator.service 및 transaction-generator.service)을 /etc/systemd/system/ 디렉토리에 다음 템플릿을 사용하여 생성합니다:

```
[Unit]
Description=__________ Log Generator
After=network.target

[Service]
Type=simple
ExecStart=/home/ec2-user/__________-log-generator
WorkingDirectory=/home/ec2-user
Restart=always
RestartSec=5
User=ec2-user
Group=ec2-user

[Install]
WantedBy=multi-user.target
```

- 서비스 파일 생성 후, systemd 관련 명령어를 사용하여 서비스를 등록하고 활성화합니다. (systemctl daemon-reload / enable / start)
- 액세스 로그는 “2025-05-15T10:23:45Z 192.168.1.105:8080 GET /products/electronics 200 user123” 형식의 웹 요청 로그로 생성됩니다.
- 트랜잭션 로그는 {"id":"TX-20230512-1317","timestamp":"2023-05-12T18:20:52Z","customer_name":"abc123","product":"Accessory Alpha","quantity":2,"price":58.54,"store_id":"Store-003","payment_type":"Cash","action":"purchase","category":"accessories"} 형식의 JSON 데이터로 생성됩니다.

6. Amazon Kinesis Data Stream

- 다음 두 개의 Kinesis Data Stream을 생성합니다.
    - retail-access-stream : 액세스 로그 수집용
    - retail-transaction-stream : 트랜잭션 로그 수집용
- 두 스트림 모두 다음 설정을 적용합니다:
    - 데이터 보존기간 : 24시간
    - 샤드 수 : 1개
- Kinesis 에이전트를 사용하여 Kinesis Data Stream에 데이터를 쓰도록 합니다.
    - /etc/aws-kinesis/agent.json 을 열고 편집합니다.
    - 디렉토리 및 파일 권한을 적절히 구성하여 Kinesis Agent가 로그 파일에 접근할 수 있도록 합니다.
    - Kinesis Agent가 시스템 재부팅 후에도 자동으로 실행되도록 systemd 서비스 파일(aws-kinesis-agent.service)을 /etc/systemd/system/ 디렉토리에 다음 템플릿을 사용하여 생성합니다: (sudo journalctl -u aws-kinesis-agent 명령어를 통해 로그를 확인할 수 있습니다.)

```
[Unit]
Description=AWS Kinesis Agent
After=network.target

[Service]
Type=simple
ExecStart=/usr/bin/start-aws-kinesis-agent
Restart=always
RestartSec=5

[Install]
WantedBy=multi-user.target
```

- 서비스 파일 생성 후, systemd 관련 명령어를 사용하여 서비스를 등록하고 활성화합니다. (systemctl daemon-reload / enable / start)

7. Kinesis Data Firehose

- retail-access-delivery와 retail-transaction-delivery 두 개의 Firehose 전송 스트림을 생성합니다. (에러 발생시, 1분 이후에 다시 시도)
- retail-access-delivery의 데이터 소스로는 retail-access-stream을, retail-transaction-delivery의 데이터 소스로는 retail-transaction-stream을 사용합니다.
- retail-data-lake-{계정번호 앞6자리}라는 이름의 S3 버킷을 생성하고 Firehose의 목적지로 설정하고 관련 권한을 부여합니다.
- 트랜잭션 데이터는 raw/transaction/ 경로로 저장되도록 설정합니다.
- 웹 로그 데이터는 raw/access/ 경로로 저장되도록 설정합니다.
- 버퍼 힌트는 크기 1MB, 간격 60초로 설정하여 테스트 중 데이터가 빠르게 S3에 저장되도록 합니다.

8. Glue

데이터베이스 및 카탈로그 구성

- retail_analytics_db라는 이름의 Glue 데이터베이스를 생성합니다.
- 데이터베이스 설명에는 "Online Store Data Analysis Database"라고 명시합니다.

커스텀 분류기 구성

- access-log-classifier라는 이름의 커스텀 분류기를 생성합니다.
- 분류기 유형을 Grok으로 설정합니다.
- Classification 을 grokLog 으로 설정합니다.
- Grok 패턴을 '%{TIMESTAMP_ISO8601:timestamp} %{IP:ip_address}:%{NUMBER:port} %{WORD:method} /%{DATA:path} %{NUMBER:status_code} %{USERNAME:customer_id}'로 설정합니다.

트랜잭션 데이터 크롤러 구성

- retail-transaction-crawler라는 이름의 크롤러를 생성합니다.

- 크롤러의 데이터 소스를 S3의 retail-data-lake-{계정번호 앞6자리}/raw/transaction/ 경로로 설정합니다.
- 크롤러의 대상 데이터베이스를 retail_analytics_db로 설정합니다.
- 크롤러의 테이블 접두사를 raw_로 설정합니다.
- 크롤러에 AWSGlueServiceRole-transaction IAM 역할을 생성한 뒤 할당합니다.
- 트랜잭션 데이터에는 기본 JSON 분류기를 사용하고 crawler를 실행합니다.

액세스 로그 크롤러 구성

- retail-access-crawler라는 이름의 크롤러를 생성합니다.
- 크롤러의 데이터 소스를 S3의 retail-data-lake-{계정번호 앞 6자리}/raw/access/ 경로로 설정합니다.
- 액세스 로그 데이터에는 access-log-classifier 커스텀 분류기를 사용하고 crawler 를 실행합니다.
- 크롤러의 대상 데이터베이스를 retail_analytics_db로 설정합니다.
- 크롤러의 테이블 접두사를 raw_로 설정합니다.
- 크롤러에 AWSGlueServiceRole-access IAM 역할을 생성한 뒤 할당합니다.

트랜젝션 데이터 처리

- process-transaction이라는 이름의 ETL 작업을 생성합니다.
- ETL 작업 유형을 Visual ETL로 생성합니다.
- AWSGlueServiceRole-transaction IAM 역할에 적절한 권한을 할당하고 ETL 작업에 부여합니다.
- Add nodes Sources
    - AWS Glue Data Catalog 의 retail_analytics_db 데이터베이스에서 raw_transaction 테이블 사용
    - IAM : AWSGlueServiceRole-transaction
- Add nodes Transforms
    - SQL Query : Price 필드의 JSON 구조 {double,int} 처리
    - Change Schema : timestamp 필드의 데이터 타입 string을 timestamp로 변경
    - Dervied Column : price와 quantity를 곱하여 total_amount 필드 생성

    - Dervied Column : price 범위에 따라 'Low(<$50)', 'Medium($50-$100)', 'High(>$100)'로 분류하는 price_category 필드 생성
        - 주어진 분류 기준을 참고하되, 실제 데이터에 맞게 적절히 조정하여 모든 레코드가 올바르게 분류되도록 구현
- Add nodes Targets
    - Amazon S3
        - Format : Parquet
        - Compression Type : Snappy
        - S3 Target Location : retail-data-lake-{계정번호 앞6자리}/processed/transaction/
        - Data Catalog update options : 테이블 자동 생성 및 스키마 업데이트 허용
        - Database : retail_analytics_db
        - Table name : processed_transaction

9. Athena

- retail-analytics-workgroup이라는 이름의 Athena 작업 그룹을 생성합니다.
- 작업 그룹의 쿼리 결과 출력 위치를 retail-data-lake-{계정번호 앞6자리}/athena-result/로 설정합니다.

가격 구간별 매출 현황 분석

- 다음 SQL 쿼리를 작성하여 price_category별 주문 건수와 총 매출액을 조회합니다.
    - price_category기준으로 그룹화
    - 각 카테고리별 주문 건수
    - 각 카테고리별 total_amount의 합계
    - total_amount 합계 기준으로 내림차순 정렬
- SELECT, COUNT, SUM, GROUP BY, ORDER BY 사용
- 테이블명: processed_transaction