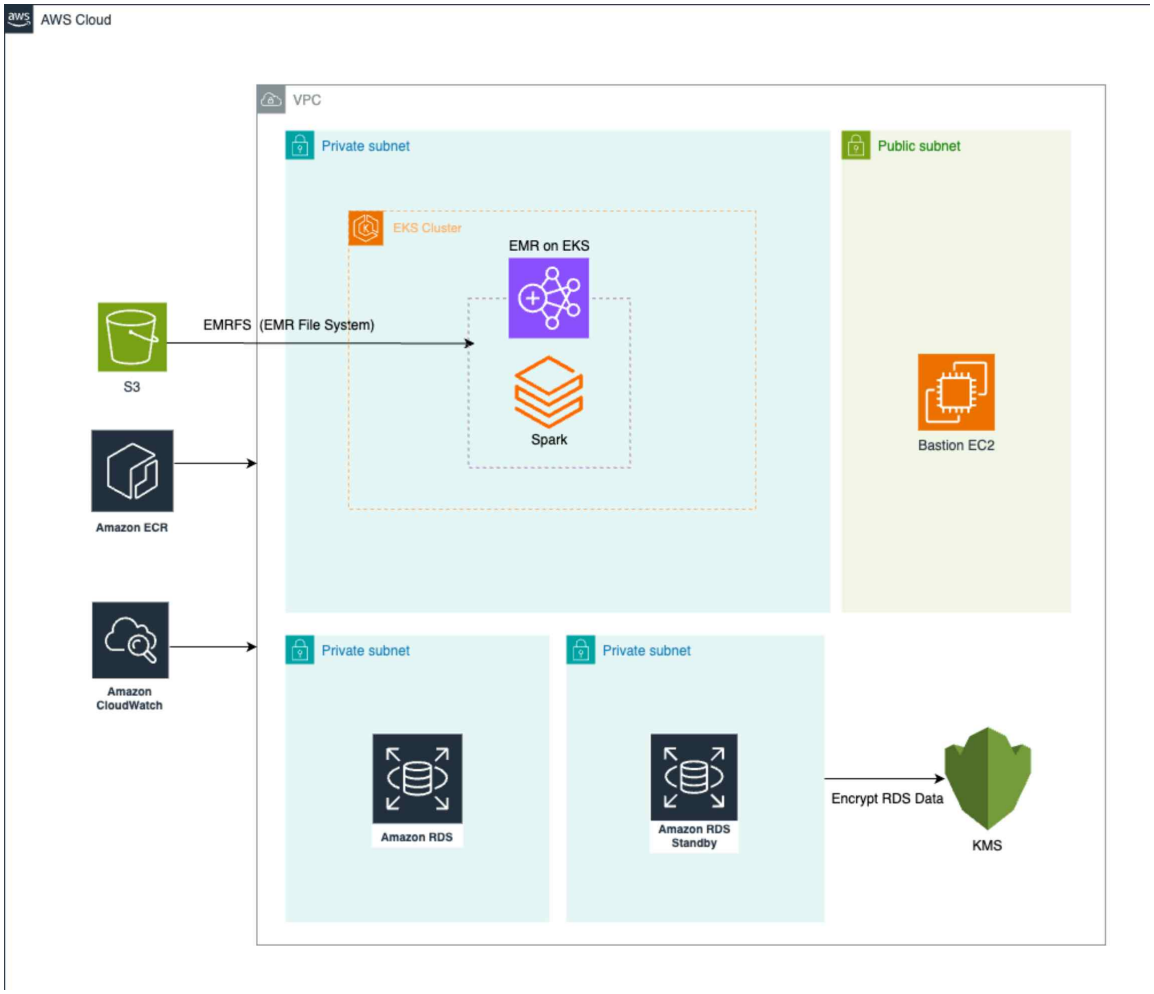
2025년도 평가경기기능경기대회 과제

직 종 명	클라우드컴퓨팅	과 제 명	EMR on EKS	과 제 번 호	제 1 과 제
경 기 시 간	4 시 간	비 번 호		심 사 위 원 확 인	(인)

1. 요구사항

당신은 WorldSkills 금융회사의 데이터 엔지니어링 팀에 합류하여 금융 트랜잭션 분석 시스템의 인프라 설계와 구현을 담당하게 되었습니다. 이 시스템은 AWS 클라우드에서 구현하려고 합니다. 특히 EMR on EKS를 활용하여 확장성과 고가용성을 높이고자 합니다. 주어진 아키텍처를 분석하고 클라우드 설계원칙인 고가용성, 확장성, 비용, 보안 등을 잘 고려하여 인프라를 구축하세요.

다이어그램



Software Stack

AWS	개발언어/프레임워크
• VPC • EC2 • EKS • ECR • RDS • KMS • S3 • CloudWatch • EMR	• Python • DSL

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 있는 bastion server에서 대부분의 채점이 이루어지므로 인스턴스를 생성하지 않았거나 종료된 상태면 채점이 불가합니다.
6) 선수 계정에는 비용의 제한이 존재함으로 이를 고려해 경기 진행에 불이익이 없도록 유의합니다.
7) 모든 리소스는 서울(ap-northeast-2) 리전에 구성합니다.
8) 과제에 적힌 이름과 태그 등을 이용해 awscli 채점을 수행합니다. 이름과 패키지 설치에 유의하여 작업합니다.
9) 문제나 채점지에 <> 는 변수 임으로 선수가 변경하여 사용하여야 합니다.
10) 문제에서 언급하지 않은 설정은 기본값으로 유지합니다.
11) 제공자료는 수정 없이 사용합니다. 제공자료를 수정해서 사용하면 불이익을 받을 수 있습니다.

3. 네트워킹

1) VPC 구성

- Name Tag=ws-vpc
- CIDR: 10.0.0.0/16

2) Subnet 구성

(1) 퍼블릭 서브넷 A

- 가용영역: ap-northeast-2a
- CIDR: 10.0.0.0/24
- Name Tag=ws-public-a
- 라우팅테이블: ws-public-rtb
- Internet GW (ws-igw)를 통하여 통신

(2) 퍼블릭 서브넷 B

- 가용영역: ap-northeast-2b
- CIDR: 10.0.1.0/24
- Name Tag=ws-public-b
- 라우팅테이블: ws-public-rtb
- Internet GW (ws-igw)를 통하여 통신

(3) 프라이빗 서브넷 A

- 가용영역: ap-northeast-2a
- CIDR: 10.0.2.0/24
- Name Tag=ws-private-a
- 라우팅테이블: ws-private-rtb-a
- NAT GW (ws-nat-a)를 통하여 통신

(4) 프라이빗 서브넷 B

- 가용영역: ap-northeast-2b
- CIDR: 10.0.3.0/24
- Name Tag=ws-private-b
- 라우팅테이블: ws-private-rtb-b
- NAT GW (ws-nat-b)를 통하여 통신

(5) 프라이빗 서브넷 C

- 가용영역: ap-northeast-2a

- CIDR: 10.0.4.0/24
- Name Tag=ws-private-c
- 라우팅테이블: ws-private-rtb-c
- NAT GW (ws-nat-a)를 통하여 통신

(6) 프라이빗 서브넷 D

- 가용영역: ap-northeast-2b
- CIDR: 10.0.5.0/24
- Name Tag=ws-private-d
- 라우팅테이블: ws-private-rtb-d
- NAT GW (ws-nat-b)를 통하여 통신

4. Bastion Server

EC2를 사용하여 Bastion 서버를 생성합니다. 이 인스턴스는 모든 AWS의 권한을 사용할 수 있어야 하며, SSH로 서버에 접근 가능해야 하지만 기본 SSH 포트를 사용하면 안됩니다. 인스턴스를 재부팅해도 IP가 변경되어서는 안됩니다.

- Name Tag: ws-bastion-2025
- OS Image: Amazon Linux 2023
- Instance type: c5.large
- Subnet: ws-public-a
- 설치 패키지: kubectl, aws, jq, eksctl, docker
- 포트번호: 2222

5. Storage

데이터 저장을 위해 S3를 구성합니다. S3 버킷은 KMS로 암호화되어야 하며, 버전 관리가 활성화되어야 합니다. 제공된 파일들이 아래 폴더구조에 맞춰 구성된 S3 버킷에 업로드되어야 합니다(필요한 폴더는 직접 생성합니다). 이때, credit_card_analysis.py에서 "your-account-id"를 본인의 accountID를 수정한후 업로드하세요. 또한, 분석 결과를 저장하기 위한 /output 하위폴더를 생성하세요.

- 버킷 이름: ws-cc-raw-data-<accountID>
- 버전 관리: 활성화

- 서버 측 암호화: SSE-KMS (키 별칭: ws-secret-key)

폴더구조

```
ws-cc-raw-data-{AWS_ACCOUNT_ID}/
│
├── data/
│   └── creditcard.csv
│
├── templates/
│   ├── driver-template.yaml
│   └── executor-template.yaml
│
├── output/
│   └── (empty directory)
│
├── create_tables.sql
└── credit_card_analysis.py
```

6. Relational Database

eks로 분석한 데이터들을 저장할 관계형 데이터베이스로 PostgreSQL을 사용합니다. AWS에서 사용할 수 있는 Fully-managed RDS Service인 Aurora and RDS를 활용해 데이터가 저장될 수 있어야 합니다. PostgreSQL의 포트는 기본 포트가 되어서는 안됩니다. 다중 AZ배포를 활성화하여 고가용성을 보장해야 하고, 이 DB 볼륨에 저장되는 모든 데이터는 KMS(ws-secret-key)를 이용하여 암호화되어야 합니다. rds 인스턴스 서브넷 그룹은 ws-private-c, ws-private-d에 지정되어야 합니다. 제공받은 create_tables.sql 파일을 생성한 rds에 실행하여 table들을 생성합니다.

- Instance Name: ws-db-instance
- 엔진: PostgreSQL
- 템플릿: 개발/테스트
- 버전: latest
- 포트번호: 5433
- Instance Class: db.t4g.medium
- 마스터 사용자 이름: wsadmin
- 마스터 암호 설정: 자체 관리 키 wspassword123
- 초기 데이터베이스 이름: fraud_detection

7. KMS

AWS KMS를 사용하여 데이터 암호화에 사용할 키를 생성합니다. 이 키는 S3에 저장되는 데이터와 RDS의 데이터를 암호화하는데 사용됩니다. 키는 자동 키 회전이 활성화되어야 합니다.

- 키 이름: ws-secret-key

8. 쿠버네티스

Amazon EKS를 사용하여 컨테이너화된 애플리케이션을 실행하고 데이터를 불러와 분석합니다. 워크로드 분리를 위해 카펜터를 사용하여 분석 어플리케이션 Pod를 관리하며 카펜터는 ws-addon 노드 그룹에 배포되어야 합니다. 카펜터를 통해 관리하는 emr 관련 노드들은 모두 ws-emr 네임스페이스를 사용하며 그 외 기본 시스템들은 ws-system 노드그룹에서 실행되도록 합니다. ws-addon과 ws-emr은 taints 옵션을 사용하여 맞는 파드만 배치되도록 관리합니다. 모든 쿠버네티스 관련 서비스들은 ws-private-a와 ws-private-b 서브넷 위에서 실행되어야 합니다.

1) 이미지 저장소 구성

컨테이너 이미지 저장소로 AWS ECR을 사용합니다. 이 레포지토리는 KMS로 암호화되며 보안을 위해 푸시할때 스캔이 되어야 합니다. 또한 같은 태그의 이미지가 업로드 되지 않도록 합니다.

- 레포지토리 네임스페이스: ws-emr
- 리포지토리 이름: spark
- 암호화 구성: AWS KMS: ws-secret-key

2) 클러스터 구성

- 클러스터 이름: ws-eks-cluster
- Kubernetes 버전: 1.32
- EKS 자율모드 - 비활성화
- 클러스터 API 서버는 VPC 외부에서 접속할 수 없도록 합니다.
- 클러스터에서 발생하는 API 서버 로그를 CloudWatch로 전송해야 합니다.

3) 노드 그룹 생성

- ws-addon 노드그룹 생성
    - 이름: ws-addon

    - 노드 IAM 역할: ws-eks-node-role
    - 인스턴스 유형: t3.xlarge
    - 원하는크기:2 / 최소크기:2 / 최대크기: 3
    - 노드 라벨: key=nodegroup/value=addon
    - taints: key=node-role.kubernetes.io/ws-addon, effect=NoSchedule
- ws-system 노드 그룹 생성
    - 이름: ws-system
    - 노드 IAM 역할: ws-eks-node-role
    - 인스턴스 유형: t3.medium(기본값)
    - 원하는크기:2 / 최소크기:2 / 최대크기: 3
    - 노드 라벨: key=nodegroup/value=system
- 네임 스페이스: ws-emr (EMR 관련 모든 리소스는 이 네임스페이스에 생성)

4) Karpenter 구성

EKS 클러스터의 노드 관리를 위해 Karpenter를 사용하여 구현해야 합니다. 카펜터는 ws-addon 노드 그룹 및 karpenter 네임스페이스 위에서만 배포되어야 합니다. 또한 카펜터가 생성하는 모든 노드는 ws-emr 네임스페이스를 가지도록 taints를 설정해야 하며 emr job을 수행하는 모든 노드는 카펜터에 의해서만 생성되고 관리되어야 합니다.

- Karpenter Contoller 설치
    - 네임스페이스: karpenter
    - 카펜터 컨트롤러 역할: ws-karpenter-controller-role
- NodePool 구성 조건
        - **노드 라벨**:
            - nodegroup: emr
            - eks.amazonaws.com/compute-type: emr
        - **노드 테인트**:
            - key=eks.amazonaws.com/compute-type, value=emr (NoSchedule 효과)
        - **인스턴스 요구사항**:
            - 아키텍처: amd64
            - 운영체제: Linux
            - 용량 유형: on-demand

        - 인스턴스 패밀리: m5, m5d, c5, c5d, c4, r4
        - CPU 코어: 4, 8, 16, 32
        - 가용 영역: ap-northeast-2a, ap-northeast-2b
    - **리소스 제한**:
        - CPU: 최대 40 코어
        - 메모리: 최대 160Gi
    - 노드 공실 시간: 90초

9. EMR on EKS 구성

EMR on EKS를 사용하여 Spark를 통해 S3에 저장된 데이터를 분석합니다. Spark를 실행하기 위한 제공받은 도커파일을 ws-emr/spark 레포지토리에 푸시해야 하며 처리된 데이터는 RDS에 저장되어야 합니다. 작업 실행에 필요한 모든 권한은 IAM 역할을 통해 제공되어야 하며, 데이터 처리 과정에서 발생하는 모든 로그는 CloudWatch에 저장되어야 합니다. 가상 클러스터는 지정된 EKS 클러스터 내에서 실행되어야 합니다.

1) 가상 클러스터 생성

- 가상 클러스터 이름: ws-virtual-cluster
- EKS 클러스터: ws-eks-cluster
- 네임스페이스: ws-emr

2) 작업 실행 역할 구성

- 역할 이름: ws-job-execution-role
- S3를 읽고/쓸수 있어야하며, CloudWatch Log 쓰기, RDS, KMS의 키(ws-secret-key) 사용 권한이 있어야 합니다.

3) 컨테이너 이미지 구성

EMR on EKS는 기본적으로 Spark 작업 실행을 위한 이미지와 로깅을 위한 이미지를 제공하지만, 이 과제에서는 특화된 요구사항을 충족하기 위해 커스텀 이미지를 사용해야 합니다. 제공받은 도커 파일을 불러와 이미지를 빌드, ECR에 푸시가 이루어져야 합니다.

4) 분석 작업 제출 스크립트 생성 및 실행

제공된 도커파일을 통해 이미지 빌드가 완료된 EMR on EKS에서 Spark 작업을 실행시켜 S3에 저장된 credit_card_analysis.py를 통해 creditcard.csv를 분석합니다. 작업 제출 스크립트는 직접 작성하며 반드시 ws-emr 네임스페이스에 잡 실행 노드가 올라가도록 해야 합니다. 파이썬 분석 코드 실행 완료시 S3와 RDS에 모두 저장되도록 되어 있으므로 적절한 권한 부여 및 연결을 해주어야 합니다. 잡 제출은 아래 가이드를 참고하여 제출합니다.

- 참고사항
  - 가이드에 작성되어 있는 부분은 모두 필수로 포함하되 추가적으로 작성해야 하는 부분이 있을 수 있음
  - Job Driver와 Job Executor에도 Toleration을 적용해주어야 함 (s3에 저장된 yaml 파일의 링크를 달아줄 것)
  - *아래 첨부된 쉘스크립트에 문제가 있다면 다음 링크를 참고하여 풀이하세요: https://childlike-second-a2f.notion.site/9-4-1fe2e1362ad8809b8d4bdfde1e6c4d70?pvs=4

```
cat > submit-spark-job.sh << 'EOF'
#!/bin/bash

# 환경 변수 설정
export AWS_DEFAULT_REGION=
export ACCOUNT_ID=
export VIRTUAL_CLUSTER_ID=
export JOB_EXECUTION_ROLE_ARN=
export SPARK_IMAGE_URI=
export S3_BUCKET=

# RDS_HOST 변경 필수
export RDS_HOST=
export RDS_PASSWORD=

TOLERATIONS=

# 작업 제출
aws emr-containers start-job-run \
  --virtual-cluster-id $VIRTUAL_CLUSTER_ID \
  --name credit-card-fraud-detection \
  --execution-role-arn $JOB_EXECUTION_ROLE_ARN \
  --release-label emr-6.9.0-latest \
  --job-driver "{\"sparkSubmitJobDriver\": {\"entryPoint\": \
"sparkSubmitParameters\": \" --conf spark.kubernetes.tolerations=${TOLERATIONS} \
--conf spark.kubernetes.driver.podTemplateFile= \
--conf spark.kubernetes.executor.podTemplateFile= \
--configuration-overrides "{\"applicationConfiguration\": [{\"classification\": \"spark-
defaults\", \"properties\": {\"spark.kubernetes.container.image\": \"${SPARK_IMAGE_URI}\"}},
{\"classification\": \"spark-env\", \"properties\": {}, \"configurations\":
[{\"classification\":\"export\", \"properties\": {\"RDS_PASSWORD\":
\"${RDS_PASSWORD}\",\"RDS_HOST\": \"${RDS_HOST}\"}}]}]
  --region $AWS_DEFAULT_REGION
EOF
```