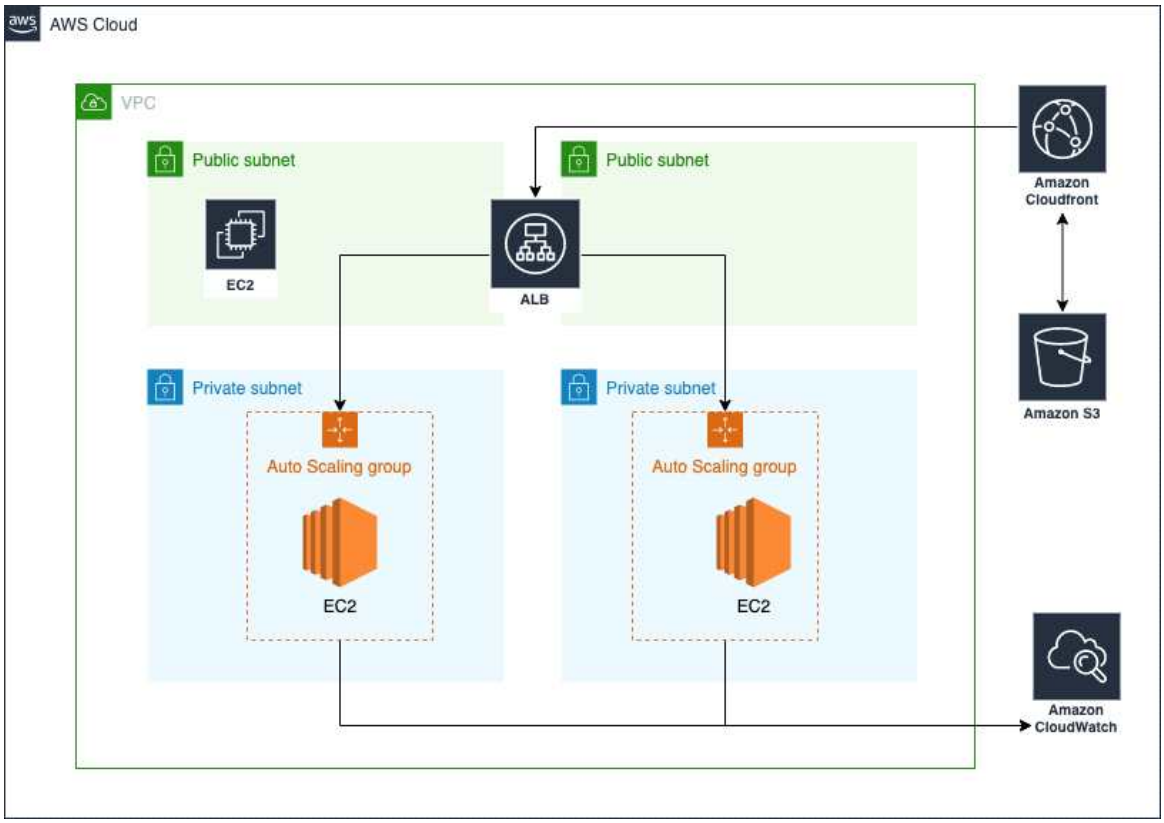
2021 대전광역시 제56회 전국기능경기대회 과제

직 종 명	클라우드컴퓨팅	과 제 명	Web Service Provisioning	과제번호	제1과제
경기시간	4시간	비 번 호		심사위원 확 인	(인)

1. 요구사항

AWS 서비스를 이용하여 웹서비스를 운영할 수 있는 클라우드 플랫폼을 구성 하고자 합니다. 주어진 아키텍처를 바탕으로 고가용성과 성능 등 여러 가지 요소를 고려하여 웹 어플리케이션이 구동 할 수 있는 클라우드 플랫폼을 구축 하여야 합니다. AWS에서 제공하는 솔루션을 통해 보다 빠르고 안정성 있게 구축 하는 것이 당신의 업무입니다.

다이어그램



Software Stack

- AWS
  - VPC
  - EC2(Autoscaling)
  - LoadBalancer
  - S3
  - CloudFront
  - CloudWatch
- 개발언어/프레임워크
  - Python / Flask

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) 선수의 계정에는 비용제한이 존재하며, 이보다 더 높게 과금될 시 계정 사용이 불가능할 수 있습니다.
6) 문제에 제시된 괄호박스 <>는 변수를 뜻함으로 선수가 적절히 변경하여 사용해야 합니다.
7) 문제의 효율을 위해 Security Group의 80/443 outbound는 anyopen하여 사용할 수 있도록 합니다.
8) Bastion EC2는 채점시 사용되기 때문에 종료되어 불이익을 받지 않도록 주의해 주시기 바랍니다.
9) 과제 종료 시 진행 중인 테스트를 모두 종료하여 서버에 부하가 발생 하지 않도록 합니다.

3. Cloud Netowkring

클라우드 인프라에 대해 네트워크 레벨의 격리 및 분리가 가능하도록 아래 요구사항을 참고하여 VPC를 구성 합니다. Private subnet은 각각 route table을 따로 가져야 합니다. Public Subnet은 하나의 route table을 가집니다. 이름 가장 뒤에 붙은 알파벳은 ZONE을 의미 합니다.

VPC 정보

- VPC CIDR : 10.1.0.0/16
- VPC Tag : Name=wsi-vpc

Private A subnet 정보

- CIDR : 10.1.0.0/24
- Tag : Name=wsi-private-a
- 외부 통신 : NAT G/W를 구성하여 인터넷 접근이 가능하도록 구성
- Route table Tag : Name=wsi-private-a-rt

Private B subnet 정보

- CIDR : 10.1.1.0/24
- Tag : Name=wsi-private-b
- 외부 통신 : NAT G/W를 구성하여 인터넷 접근이 가능하도록 구성
- Route table Tag : Name=wsi-private-b-rt

Public A subnet 정보

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

Public B subnet 정보

- CIDR : 10.1.3.0/24
- Tag : Name=wsi-public-b
- 외부 통신 : Internet G/W를 구성하여 인터넷을 접근
- Route table Tag : Name=wsi-public-rt

4. S3 hosting

아래 정보를 이용하여 S3 버킷을 생성하고, 지급 받은 s3.tar에 있는 파일들을 모두 s3에 업로드 합니다. 해당 파일들은 이후 CloudFront를 통해 접근할 수 있어야 합니다.

- 버킷이름 : wsi-<비번호>-<4자리 임의 영문>-web-static

5. Bastion 서버

EC2를 활용해 Bastion 서버를 구성합니다. 해당 서버는 public 존에 위치하고 stop 후 재시작 하더라도 public ip가 변경돼서는 안 됩니다. 외부에서 SSH 접근은 해당 bastion 서버를 통해서 접근합니다. **채점 시에도 사용함으로 인스턴스가 종료되어 불이익 받지 않도록 합니다.**
채점 시 Bastion 어떤 유저에서도 awscli 명령어를 호출 하여도 PowerUser policy 권한을 갖도록 설정해야 합니다.

- EC2 type : t3.small
- 이미지 : Amazon Linux2
- 권한 : AWS PowerUser policy
- 설치 패키지 : awscli, jq, curl
- Tag : Name=wsi-bastion-ec2

6. 웹 어플리케이션

해당 과제에서 배포하여 사용할 웹 어플리케이션 입니다. HTTP GET./v1/color 하나의 API를 가지고 있으며, 입력 받는 쿼리 스트링에 따라 다른 응답값을 전송합니다. name이라는 파라미터를 받으면 몇몇 색에 대하여 RGB 코드를 반환합니다. 추천 실행 명령어는 환경에 따라 실행되지 않을 수 있습니다. 단순 가이드이며 실행 시 추천 명령어를 사용하지 않아도 무방합니다. nginx uWSGI 등 여러 쓰레드를 이용해 워크로드를 받는 등 더 좋은 방법으로 어플리케이션을 실행해도 무방합니다. 제공된 어플리케이션 바이너리는 EC2에서 다운로드해 사용 가능하도록 S3 버킷을 생성하여 업로드 합니다. 제공된 바이너리는 소스코드는 수정해서 사용하지 않습니다.

아래 정보를 웹서버 구축 시 활용합니다.

```
#!/usr/bin/python3
from flask import Flask, abort, request, jsonify
```

- S3 이름(Artifactory) : wsi-<비번호>-<4자리 임의값>-artifactory
- 개발언어 및 프레임워크 : Python3 / Flask
- 로그 정보 : /var/log/app/app.log에 저장 되도록 코드에서 구현됨
- 추천 실행 명령어 : nohup python3 app.py &
- Health check API : HTTP GET /health
- 제공 API : HTTP GET /v1/color
  호출 예) http://localhost:8080/v1/color?name=red&hash=abcd
  응답 예) {code: fc4a07, name: orange}
  (value는 다를 수 있으나 code, name key를 가짐)
- Listen port : TCP 8080
- 라이브러리 정보

```
#!/usr/bin/python3
from flask import Flask, abort, request, jsonify
```

7. 오토스케일링

웹 어플리케이션이 EC2에서 구동할 수 있도록 구성합니다. 오토스케일링 그룹을 통해서 EC2가 생성되도록 합니다. 많은 워크로드를 받을 시 자동으로 EC2가 늘어나야 하며, 자동화를 통해 별다른 구성없이도 웹서버가 동작 하도록 합니다. 인스턴스가 새로 생성 되더라도 자동으로 웹 어플리케이션이 구동 되지 않으면 동작하지 않는 걸로 간주됩니다. 해당 서버에서 문제에서 생성한 artifactory에 있는 파일을 다운로드 할 수 있어야 합니다. 다만, 업로드 권한은 없도록 구성해야 합니다. 채점 중 서버가 삭제될 수 있으며, autoscaling size를 0으로 설정할 수도 있습니다. protection 옵션 등으로 서버가 절대로 종료되지 않는 설정은 하지 않습니다.

- 오토스케일링그룹/EC2 이름 : wsi-web-api-asg
- 이미지 : Amazon Linux2
- EC2 type : t3.small
- 설치 패키지 : jq, curl
- Tag : Name=wsi-web-api-asg
- Scale-up : 평균 CPU 사용량 30 이상시 scale-up 되도록 구성
- 고가용성 : EC2 한대가 죽더라도 서비스가 동작 하도록 구성
- EC2 Tag : Name=wsi-web-api-asg (생성된 EC2 이름도 같은 Name 태그를 가짐)
- Time : 최대 6분 안에 인스턴스가 새로 생성 되거나 종료 되도록 구성

8. 로드밸런서

ALB를 이용하여 워크로드를 여러 대 wsi-web-api-asg EC2로 분산하도록 구성합니다. wsi-webapp-asg 서버는 로드밸런서를 통해서만 HTTP 요청을 받아야 합니다. 다른 외부 PC나 Bastion에서 로드밸런서를 거치지 않고 직접 HTTP 요청시 요청이 차단되어야 합니다.

- Network facing : 인터넷 망에서 LB로 접근 가능하도록 구성
- Listen : HTTP 80을 통해 접근 가능하도록 구성
- 이름 : wsi-web-api-alb
- Tag : Name-wsi-web-api-alb

9. Cloud Front

CloudFront를 통하여 웹서비스 접근이 가능하도록 합니다. 캐싱을 통해 유저가 브라우저를 통해 CloudFront의 주소에 접근하여 보다 빠른속도로 웹서비스를 이용할 수 있도록 합니다. 4번에서 구성한 S3에 업로드 되어 있는 정적파일들은 캐싱이 되어야 합니다. 하지만 8번에서 구성한 ALB로의 요청에 대해서는 캐싱하지 않고, Query String도 모두 origin으로 전달해야 합니다.

- 프로토콜 : CloudFront에 접근시 HTTPS를 통하여 접근 가능하도록 구성
- Origin : S3와 ALB 두개의 origin을 가지도록 구성
- Edge : 한국뿐만 아니라 전 세계의 유저가 빠른 속도로 접근 가능하도록 구성
- Tag : Name=wsi-web-cdn
- 기타 : 채점시 오동작 예방으로 IPv6는 비활성화 하고, 하나의 CloudFront만 생성

10. 로그

서버에서 발생 하는 로그를 저장합니다. wsi-web-api-asg서버에서 남기는 app 로그가 CloudWatch Logs에 저장 되도록 구성합니다.

- 로그그룹 이름 : /aws/ec2/wsi
- 로그 스트림 이름 : api_<EC2 ID>
  예) api_i-xxxx0000

11. 모니터링

CloudWatch를 이용해 시스템을 모니터링 할 수 있도록 합니다. 흩어져 있는 메트릭을 한 곳에서 보기 위하여 CloudWatch에서 Dash board를 생성합니다. 아래 메트릭 정보들을 하나의 dash board에서 확인 할 수 있어야 합니다. 하나의 DashBoard는 여러 개의 그래프로 구성되고, 하나의 그래프를 하나 이상을 메트릭을 포함할 수 있습니다. 모든 그래프의 period는 1분으로 설정합니다.

- Dash board 이름 : WSI_API

- HTTP ERROR 그래프
  - 그래프 이름 : HTTP_ERROR
  - 서버의 5XX 응답값과, LB의 5XX 응답값에 대한 메트릭 포함
  - 에러의 개수를 집계함으로 값은 sum으로 계산되어 표기돼야 함

- HTTP 요청 count 그래프
  - 그래프 이름 : HTTP_COUNT
  - LB로 유입 되는 HTTP 요청 개수에 대한 메트릭 포함
  - 요청 개수를 집계함으로 값은 sum으로 계산되어 표기돼야 함

- HTTP 응답 시간 그래프
  - 그래프 이름 : RESPONSE_TIME
  - LB로 들어오는 HTTP 요청의 평균 응답 시간 메트릭 포함

- Autoscaling EC2 CPU 그래프
  - 그래프 이름 : API_CPU
  - wsi-web-api-asg 인스턴스의 평균 CPU 사용량 메트릭 포함