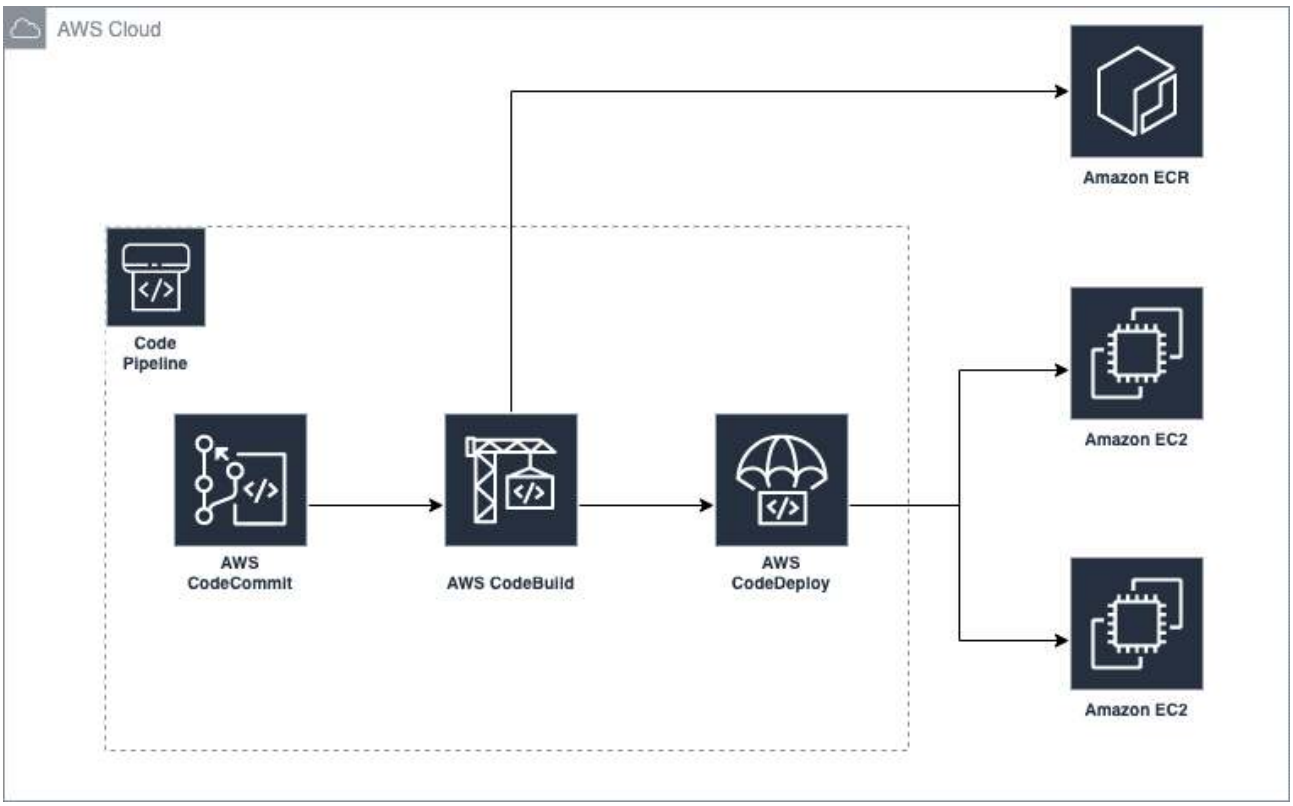
2021 대전광역시 제56회 전국기능경기대회 과제

직 종 명	클라우드컴퓨팅	과 제 명	Automation	과제번호	제2과제
경기시간	4시간	비 번 호		심사위원 확 인	(인)

1. 요구사항

개발 효율을 위하여 CI/CD 파이프라인을 구성하고자 합니다. AWS 환경에서 개발하고 있기 때문에 모두 AWS solution을 활용하여 파이프라인을 구성할 예정입니다. 소스코드 커밋시 자동화 되어 EC2로 배포가 되기도 하고, 경우에 따라 ECR로 도커 이미지를 생성해 업로드할 수 있어야 합니다.

다이어그램



Software Stack

- AWS
  - VPC
  - EC2
  - Code Commit
  - Code Build
  - Code Deploy
  - Code Pipeline
  - ECR
- 개발언어
  - 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.

3. 네트워킹

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.어플리케이션

제공된 app.py는 Python Flask를 통해 개발된 어플리케이션입니다. 채점시 GET /health API를 호출하고 응답값을 체크해 정답 유무를 측정합니다. 해당 API의 로직을 수정 하는 경우 채점에 불이익을 받을 수 있습니다.

5. EC2

Python 어플리케이션 실행할 EC2를 Private subnet에 생성합니다. 서버에 Flask와 같은 라이브러리가 설치되지 않은 상태라도 Continuous Delivery 이후에는 패키지, 라이브러리가 자동으로 설치되고, 어플리케이션을 다운로드 받아 실행되도록 해야 합니다. EC2는 오토스케일링 그룹을 통해 생성하는 것이 아니라 일반 EC2를 생성합니다. Code Deploy가 EC2에 접근 가능하도록 권한을 부여합니다. 권한은 IAM Role을 생성 하고, 인스턴스에 할당하도록 합니다. 어플리케이션의 로그는 따로 남길 필요는 없습니다.
EC2는 총 2대 생성하며 한대의 이름은 wsi-api-1입니다. 다른 EC2의 이름은 wsi-api-2로 생성합니다. 여기서 뜻하는 이름은 EC2 Tag의 Name을 뜻합니다.

- 이미지 : Amazon Linux2
- EC2 Type : t3.small
- IAM role name : wsi-api
- Tag 정보
  - wsi:deploy:group=dev-api
  - 한 서버에는 Name=wsi-api-1로 할당, 다른 서버에는 Name=wsi-api-2로 할당

6. Code commit

어플리케이션의 소스코드를 관리하기 위해 code commit을 사용합니다. code commit repository를 하나 생성하고 제공된 app.py 파일을 업로드 합니다. 해당 repository는 main과 release 두 개의 branch를 가지고 있습니다. 과제 종료 전 최종적으로 main과 release 브랜치의 파일 내용이 같도록 설정해 둡니다. 자동 동기화를 의미하는 것이 아니라 종료 전 한번만 동기화 합니다. 제공받은 app.py는 repository에 src 디렉토리를 생성하고 아래에 위치시킵니다. deploy에 이용하는 appspec파일은 제일 상위에 위치합니다.

- repository 이름 : wsi-api-repo
- default branch : main
- app.py 위치 : src/app.py
- appspec.yml 위치 : appspec.yml

7. Code build

어플리케이션 소스코드를 빌드 하기 위해 code build를 사용합니다. 파이썬(python)으로 된 코드를 컴파일 하여 app.pyc 파일로 나오도록 합니다. cloudwatch log를 활성화 하여 빌드 시작 후 진행상황 로그를 볼 수 있도록 합니다. main 브랜치용 code build와 release 브랜치용 code build 두 개를 생성합니다.

- main branch용 이름 : wsi-api-build
- release branch용 이름 : wsi-api-release

8. Code deploy

빌드 된 산출물이 서버에 복사되어 배포하도록 합니다. appspec.yml 파일을 이용해 배포에 필요한 내용들을 스크립팅 합니다. in-place 배포를 사용해 새로운 바이너리로 배포되도록 합니다. EC2의 Name 태그에 "wsi:deploy:group"="dev-api" 를 가지고 있는 EC2 모두를 배포 대상으로 합니다. 아무것도 설치되지 않은 새로운 EC2가 생성되더라도 deploy가 실행되면 빌드된 산출물이 배포되어 프로세스가 실행되어야 합니다. 배포 과정에서 ALB의 타겟그룹에도 자동으로 attach 되어야 합니다.

- application name : wsi-api
- deployment group name : dev-api
- 배포 유형 : In-place deployment
- 배포 타겟 : EC2 Tag중 "wsi:deploy:group"="dev-api" 를 가지고 있는 EC2 전체

9. Code Pipeline

wsi-api-repo를 Source로 wsi-api-build를 Build로 wsi-api의 dev-api를 Deploy로 가지는 파이프라인을 생성합니다. 각 단계 이름은 각각 Source, Build, Deploy로 설정합니다. wsi-api-repo의 main 브랜치에서 소스코드가 변경되면 별도의 빌드나 배포를 누르지 않더라도 자동으로 빌드되고 서버에 배포되어야 합니다. code commit 이후 최대 15분 이내에 서버 배포까지 모두 완료되어야 합니다.

- pipeline 이름 : wsi-api-pipeline
- Source stage : wsi-api-repo code commit의 main 브랜치
- Build stage : wsi-api-build
- Deploy stage : wsi-api의 dev-api

10. EC2 scripting

EC2를 생성하는 스크립트를 생성합니다. bastion EC2의 /opt/ec2_launch.sh 위치시키며, /opt/ec2_launch.sh wsi-api-30이라고 적으면 wsi-api-3을 파라메터로 받아 해당 Name 태그를 갖는 인스턴스를 생성합니다. 서브넷과 AMI, 다른 Tag 등 다른 정보들은 5번 EC2에 있는 것과 동일하게 설정합니다.

- 스크립트 역할 : 파라메터를 받아 새로운 EC2를 생성 하는 스크립트
- 스크립트 위치 : Bastion EC2의 /opt/ec2_launch.sh
- 수행 방법 : /opt/ec2_launch.sh <이름 태그> (예 /opt/ec2_launch.sh api3)

11. ECR

wsi-api-repo code commit의 release 브랜치 내용이 변경될 시 자동으로 app.py 파일을 app.pyc로 컴파일 하고, 도커 이미지를 생성해 ECR로 업로드 합니다. docker run 명령어로 빌드된 도커 이미지 실행시 별도의 app.pyc 어플리케이션 시작 명령어를 입력하지 않아도 app.pyc 프로세스가 실행될 수 있도록 합니다. 채점은 bastion ec2에서 도커 이미지 다운로드 및 실행할 예정으로 docker 명령어가 설치되어 있어야 합니다. 경기가 끝나기 전 main 브랜치 코드와 release 브랜치 코드를 일치시켜야 합니다. 도커의 이미지 태그는 빌드가 실행되는 날짜기준으로 설정하도록 합니다. 사전에 선수가 하드코딩 하는 것이 아니라 빌드가 일어나는 시점에 자동으로 현재날짜를 태그로 사용하여야 합니다. code commit 이후 최대 5분 이내 ECR 업로드가 끝나야 합니다.

- ECR 이름 : wsi-api-ecr
- 이미지 태그 : 한국시간 기준 현재날짜 - 2자리 년도, 월, 일, 시, 분, 초 형식
  (21년 7월 1일 13시 1분 2초라면 210701130102)
- 최종 ECR 이미지 예시 :
  xxxxxx.dkr.ecr.ap-northeast-2.amazonaws.com/wsi-api-ecr:210702161539