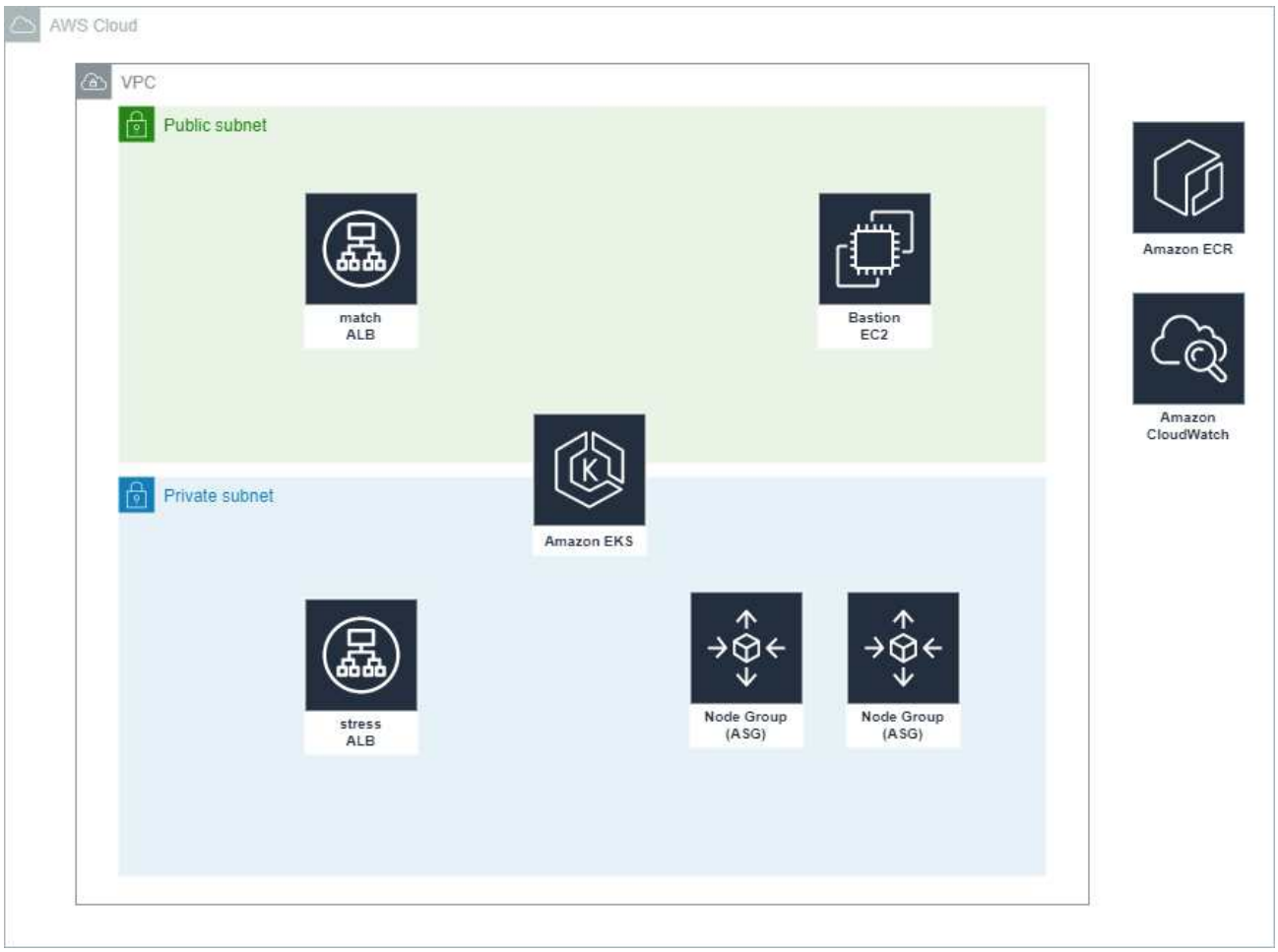
2022 경상남도 제57회 전국기능경기대회

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

1. 요구사항

AWS 서비스 및 쿠버네티스를 활용하여 웹서비스 아키텍쳐를 구성하고자 합니다. 주어진 세부 요구사항과 아키텍쳐를 활용해 고가용성, 성능 등을 고려하여 클라우드 플랫폼을 구축해야 합니다. AWS에 존재하는 관리형 서비스를 통해 빠르고 안정적으로 구축해야 합니다.



S/W stack

AWS
- EKS(Kubernetes 및 Addon)
- EC2
- Load Balancer
- ECR
- CloudWatch

Application
- Golang
- Docker

2. 선수 유의사항

※ 다음 유의사항을 고려하여 요구사항을 완성하시오.

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 비용 제한이 존재합니다. 비용 제한 이상 사용 시 계정의 사용이 불가능할 수 있습니다.
6) 문제에 제시된 괄호는 변수를 뜻함으로 선수가 적절히 변경하여 사용하도록 합니다.
7) 문제의 효율을 위해 Security Group의 80/443 outbound는 anyopen하여 사용하도록 합니다.
8) 과제 종료 시 진행 중인 테스트를 모두 종료하여 서버에 부하가 발생하지 않도록 합니다.
9) 채점 시 Bastion EC2를 사용하오니 종료되어 불이익을 받지 않도록 합니다.
10) 경기 종료 전 각 어플리케이션 별 컨테이너 2대, 노드 2대로 세팅합니다. 채점 시 서버 종료로 인해 불이익을 받지 않도록 주의하시기 바랍니다.
11) Region 지정이 필요한 서비스는 ap-northeast-2로 지정합니다.
12) 8번 문제 쿠버네티스의 경우 Pod가 늘어날시 Node가 무조건 늘어나야 한다는 의미가 아니라, Pod가 늘어나는데 Node의 리소스가 부족할 시 Node가 늘어나야 한다는 의미입니다.
13) 문제 3번 네트워크 구성에 있는 표 컬럼 중 “Name”과 “route table”은 VPC의 Subnet과 Route table Tag 구성시 Name Tag에 대한 Value를 의미 합니다.

3. 네트워크 구성

VPC를 생성하여 클라우드 네트워킹을 구성합니다. 10.0.0.0/16 네트워크를 사용합니다. HA를 고려하여 최소 3개의 az를 가지도록 VPC를 설계합니다. VPC 이름은 skills-vpc입니다. 구성 시 이름(Name)은 Key를 Name으로 갖는 Tag를 의미합니다. 서브넷 구성 시 zero subnet을 허용하며, 서브넷 마스크는 모두 24bit을 사용합니다.
다음을 참고하여 서브넷을 구성합니다.

Name	Internet access	route table	구분
skills-private-a	NAT G/W	skills-private-a-rt	사용 가능한 1번째
skills-private-b	NAT G/W	skills-private-b-rt	사용 가능한 2번째
skills-private-c	NAT G/W	skills-private-c-rt	사용 가능한 3번째
skills-public-a	Internet G/W	skills-public-rt	사용 가능한 4번째
skills-public-b	Internet G/W	skills-public-rt	사용 가능한 5번째
skills-public-c	Internet G/W	skills-public-rt	사용 가능한 6번째

4. 어플리케이션

Match와 Stress 두 개의 어플리케이션이 있습니다. 제공된 binary는 x86 기반 EC2의 Amazon Linux2에서 빌드하고 동작을 확인하였습니다. go version은 go1.16.15 linux/amd64입니다. App 실행시 바인딩 되는 포트는 8080입니다.

- Match

Query string으로 전달한 문자열을 검사하는 API를 제공합니다. Token이라는 key를 전달하면 서버에서 검사 후 응답코드를 반환합니다. 8자 이하 모두 같은 문자를 전달하면 OK이며, 8자를 초과하거나 하나라도 다른 문자가 섞여 있으면 FAIL을 반환합니다.

URL	Response body	Sample
GET /v1/match	{status: OK} {status: FAIL}	OK = /v1/match?token=cccccccc FAIL = /v1/match?token=ccccccccb FAIL = /v1/match?token=aaaaaaaaa
GET /health	{status: OK}	/health

- Stress

Stress 어플리케이션은 로드테스트를 위한 API를 포함합니다. /v1/stress를 호출하면 많은 CPU를 소모하게 됩니다. /v1/random의 경우 응답 반환 시간이 최대 40초까지 걸릴 수 있습니다.

URL	Response	sample
GET /v1/stress	{status: OK}	/v1/stress
GET /v1/random	{status: OK}	/v1/random
GET /health	{status: OK}	/health

5. Bastion

서버 접근 및 채점을 위해 Bastion 서버를 구성합니다. 해당 서버에 접근 불가할 시 채점이 불가함으로 **반드시 SSH를 통한 접속과 권한 문제가 없도록 합니다.** Bastion은 awscli 입력시 Admin Policy에 상응하는 권한을 가지고 있어야 합니다. kubectl 사용 시에도 클러스터 접근에 문제가 없고 모든 권한을 가지고 있어야 합니다.

- EC2 type : c5.large
- 이미지 : Amazon Linux2
- 설치 패키지 : awscli, jq, curl, kubectl
- Tag : Name=skills-bastion

6. 컨테이너라이징

어플리케이션을 컨테이너로 만든 후 쿠버네티스에 배포해야 합니다. 먼저 컨테이너 이미지를 생성한 후 ECR에 업로드하도록 합니다. ECR 타입은 private입니다. 컨테이너가 쿠버네티스에서 실행되어 kubectl exec 명령어로 ssh 접근 시 match app은 match라는 유저로 stress app은 stress라는 유저로 바로 접근되어야 하며, Golang app 프로세스도 해당 유저로 실행되어야 합니다. 컨테이너 이미지에서 curl 명령어를 사용할 수 있도록 패키지를 설치해 둡니다. 버전 역할을 하는 이미지 Tag는 latest로 설정합니다.

- ECR 이름 : match-ecr
- User 이름 : match
- Stress ECR 이름 : stress-ecr
- Stress User 이름 : stress

7. CodeCommit

Kubernetes yaml파일 저장을 위해 codecommit을 생성합니다. Yaml파일 저장용으로만 사용되며 별도의 CI/CD는 필요 없습니다.

- Codecommit name = skills-code

8. 쿠버네티스

어플리케이션을 실행하기 위하여 쿠버네티스 클러스터를 생성합니다. 두 개의 노드 그룹을 생성합니다. Stress, match 어플리케이션 Pod는 skills-app 노드 그룹에 배포되어야 하며, 추가로 설치하는 addon들(CA, Calico 등)은 skills-addon 노드 그룹에 배포되어야 합니다. 단, addon 중 daemonset 형태로 배포되는 컨테이너는 모든 노드에 배포되어야 합니다. match, stress pods 들은 절대로 addon 노드에 배치되어선 안 됩니다. Cluster-Autoscaler를 활용해 Pods들의 CPU가 60%가 넘어갈 시 scale-out 되도록 설정합니다. Pod가 늘어나면 Node들도 자동으로 늘어나야 합니다. Pod들에 skills/version=v1이라는 label을 추가하도록 합니다. Stress와 match 어플리케션을 모든 컨테이너에 지정되어 있어야 하며, deployment.yaml 파일에 해당 label이 정의되어 있어야 합니다. match와 stress 어플리케이션의 리소스(pods, ingress 등) 들은 모두 skills 라는 namespace에 존재해야 합니다.

채점시 쿠버네티스 리소스들은 정의된 yaml 파일로 생성과 삭제를 반복합니다. 설정 누락 없도록 파일을 구성해야 합니다.

- EKS 이름 : skills-cluster
- EKS logging : Control plane 모두 5가지 로그가 저장되어야 함(API server, audit 등)
- EKS version : 1.22.0
- Node group Name: skills-app, skills-addon
- Node group EC2 type = c5.large
- Node group subnet = private subnets
- Node min size = 2 (워크로드가 없을시 2개로 유지되도록 해야 함)
- Node label : skills/dedicated: app, skills/dedicated: addon
- Pod label(match, stress): “skills/version: v1”

9. 배포

Stress와 match 어플리케이션을 K8s에 배포하기 위한 deployment.yaml을 codecommit에서 관리하도록 합니다. Rollout으로 재시작하거나 deployment.yaml을 이용해 배포 시 처리 중인 트레픽이 유실거나 에러 처리되면 안 됩니다. 배포 시간은 최대 5분을 넘지 않도록 합니다. 배포 테스트 시 stress app을 활용하여 채점하며, 초당 10건 이상의 부하를 주입하며 배포 테스트는 하지 않습니다.

10. 보안설정

EKS에서 컨테이너 네트워크 격리를 위해 Calico를 사용합니다. Stress와 match 컨테이너끼리는 쿠버네트스 네트워크 내에서 서로 통신이 불가능해야 합니다. Stress pod에서 match pod 호출이 가능하면 틀린 것으로 간주합니다. 반대의 경우도 마찬가지입니다. 채점 시 Kubernetes의 Service를 통해 접근하며, Kubernetes에서 기본으로 만들어 주는 리눅스 환경변수 Service의 IP를 없애도록 설정하면 안 됩니다. stress와 match 어플리케이션에서 외부로의 DNS, HTTP, HTTPS 접근은 허용합니다. networkpolicy.yaml 파일을 생성해 codecommit에 업로드 하도록 합니다.

(/match/networkpolicy.yaml, /stress/networkpolicy.yaml)

11. LB

Match 어플리케이션과 Stress 어플리케이션의 K8S Ingress 생성 시 자동으로 AWS ALB가 생성되어야 합니다. 보안을 위하여 로드밸런서를 통해 접근 시 주어진 /v1/* API외에 다른 호출은 403으로 응답코드를 내려 주도록 합니다. ALB에서 Listen 하는 포트는 80으로 설정합니다. ingress.yaml파일을 만들어 관리하도록 하며, codecommit에 업로드해 두도록 합니다. 수동 설정을 하지 않도록 하며 kubectl명령으로 삭제 및 재생성 하여도 ingress와 ALB가 자동 생성되어야 합니다. 생성 후 kubectl get ingress match 입력 시 address란에 ALB의 주소가 보여야 합니다. 모든 Load Balancer의 Security Group은 ingress anyopen입니다.

Ingress name	File name in codecommit	Networking
match	/match/ingress.yaml	External
stress	/stress/ingress.yaml	Internal

12. 모니터링

skills-app 노드 그룹의 EC2 CPU, Network(IN/OUT) 정보를 모니터링 가능하도록 CloudWatch의 dashboard 페이지를 구성합니다. 메트릭의 주기는 1분이며 값의 통계는 평균 기준으로 합니다.

- Dashboard 이름 = worker
- CPU Graph 이름 = WORKER CPU
- CPU Graph metric = CPUUtilization
- Network Graph 이름 = WORKER NETWORK
- Network Graph metrics = NetworkOut, NetworkIn

stress application의 모니터링도 dashboard에서 가능하도록 합니다. ALB의 metrics를 통해 Total request count, Average latency, 5xx errors(ELB) 정보를 모니터링 가능하도록 구성합니다. 메트릭 주기는 1분이며, 통계는 메트릭마다 다릅니다.

- Dashboard 이름 = stress
- Total request count Graph 이름 = STRESS REQ COUNT
- Total request count Graph metric = ALB RequestCount
- Average latency Graph 이름 = STRESS RES TIME
- Average latency Grph metric = ALB TargetResponseTime
- 5xx errors Graph 이름 = STRESS 5xx
- 5xx errors Graph metric = HTTPCode_ELB_5XX_Count