TEAM RENEW

클라우드 기반 정보시스템 구축 전문가 양성

EKS 클러스터 생성 및 ELB 구성

2025.03.06

CONTENTS

목차

OVERVIEW

기획 의도

KUBERNETES

클라우드 환경에서
확장이 가능하고
안정적인 애플리케이션
구축 및 관리

AWS EKS

이론 학습과 함께
실제 환경에서
EKS 클러스터의 생성, 관리
애플리케이션 배포
과정 경험

SERVICE

효율적인 컨테이너
오케스트레이션을 구현하고
로드밸런싱, 인그레스를 통해
안정적이고 확장 가능한
서비스 제공 기반 구축

OVERVIEW

필요성 및 기대 효과

문제점

높은 운영 부담

기존 쿠버네티스 시스템은 자체적으로 컨트롤 플레인을 관리해야 해서 많은 시간과 자원을 소모해야 한다

불안정한 쿠버네티스 시스템

네트워크 구성 및 인증 시스템이 복잡하여 안정적인 서비스를 운영하기가 어렵다

즉각적인 대응이 어려움

장애가 발생했을 때 제때 고치는 것이 힘들다

기대 효과

낮아지는 운영 부담

AWS EKS를 사용한다면 컨트롤 플레인을 AWS가 따로 관리하여 적은 시간과 자원이 소모된다

안정적인 쿠버네티스 시스템

IAM 및 VPC와 연계하여 네트워크 및 인증 시스템을 간소화할 수 있어서 클라우드 환경에서 보다 효율적인 쿠버네티스 시스템을 구축할 수 있다

실무 환경에서의 역량 강화

DevOps 및 클라우드 네이티브 기술을 실무에 적용할 수 있는 역량을 배양할 수 있다

OVERVIEW

목표 및 추진 내용

KUBERNETES

AWS CLI 도구와 EKS를 사용하여
쿠버네티스 클러스터를 구축 및 운영

INGRESS

인그레스 컨트롤러로 트래픽 관리

CI/CD

CI/CD 파이프라인을 통해
컨테이너 애플리케이션을 자동 배포

SECURITY

AWS 보안 기술을 적용해
안전하고 효율적인 클라우드 환경 구현

OVERVIEW

목표 및 추진 내용

01

AWS CLI 및 EKS 설치

- AWS CLI 및 EKS 관련 도구 설치
- AWS 리소스를 원격으로 자동화하거나 제어할 수 있도록 환경 설정
- 해당 액세스 키 설정

02

쿠버네티스 클러스터 구축 및 운영

- EKS로 쿠버네티스 클러스터 구축
- 클러스터 및 워커 노드 등을 설정하여 다양한 서비스 통합

03

인그레스 컨트롤러를 통한 트래픽 관리

- 쿠버네티스 클러스터 내의 ALB 컨트롤러와 인그레스 설정 활용
- 애플리케이션에 대한 트래픽을 분배하고 로드 밸런싱 구현
- 고가용성과 안정적인 서비스 운영 실현

OVERVIEW

목표 및 추진 내용

04

클러스터 자동화 관리

- 자동화된 클러스터 관리 시스템 구축
- 클러스터 상태 모니터링

05

CI/CD 파이프라인 구축

- CI/CD 파이프라인 구축
- 파이프라인을 EKS 환경에 통합

06

AWS 보안 및 데이터 보호

- AWS 환경에서
  데이터 보호, 접근 제어
  모니터링 및 검사
  네트워크 보안
  애플리케이션 보안 등
  기술적 조치 이용

TEAM BUILDING

팀 구성 및 역할

팀원명	역할	추진 전략
황준서	아이디어 선정 및 보고서 점검 프로젝트 방향 설정 및 점검	AWS EKS 환경의 전반적인 점검 및 프로젝트 방향 제시
박소정	문서 작성 및 검토 애플리케이션 배포 및 관리	보고서를 체계적으로 작성 및 검토 애플리케이션 배포, 운영 및 모니터링
안웅렬		
권택	아이디어 구체화 및 설계 지원 관련 자료 수집 및 문서화	클러스터 설계를 위한 아이디어 구체화 AWS 관련 자료를 수집 보고서 작성 준비
윤승원		

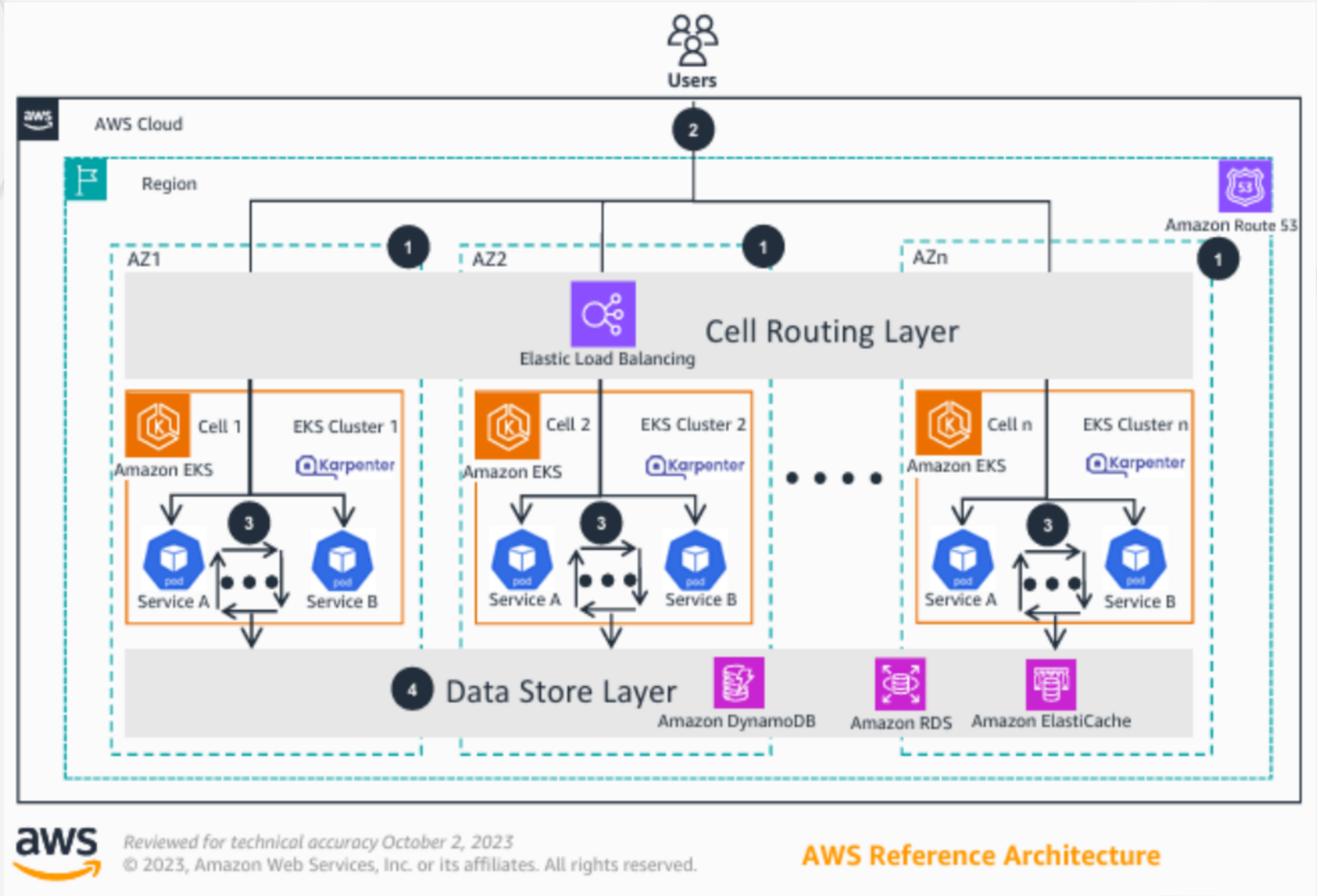
ROADMAP

수행 절차 및 방법

1	2	3
프로젝트 계획 수립	기본 인프라 구축	EKS 클러스터 구축
4	**5**	**6**
애플리케이션 배포 환경 구성	성능 최적화 및 테스트	최종 검증 및 문서화

PROJECT

프로젝트 구성도



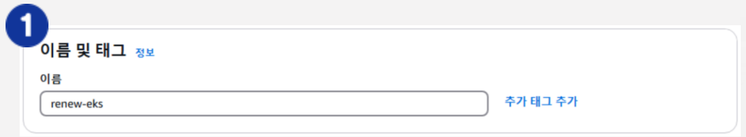
PROJECT

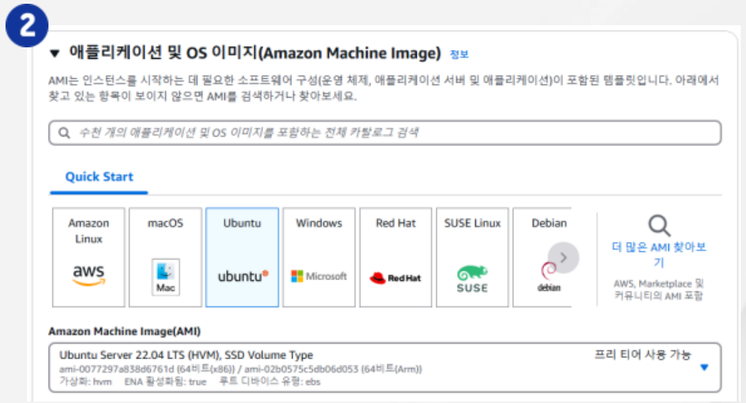
Bastion Server 설치

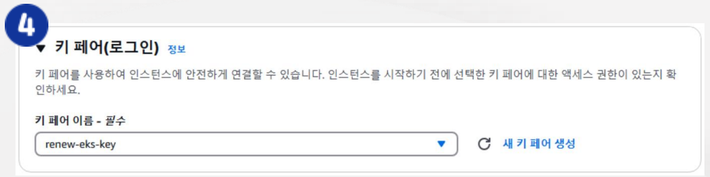
설치

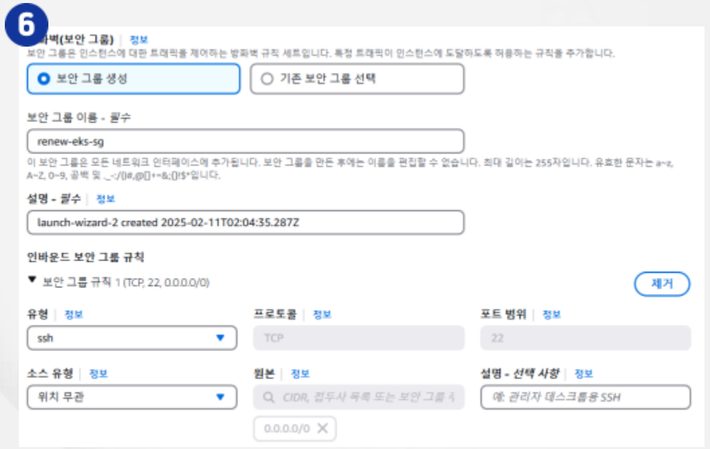
인스턴스 시작

- 이름 : renew-eks
- AMI : Ubuntu Server 22.04 LTS
- 유형 : t2.micro
- 키 페어 : renew-eks-key
- vpc : team3-vpc
- 서브넷 : teams-sub-a
- 보안 그룹 이름 : renew-eks-sg
- 보안 그룹 규칙 : ssh (위치 무관)









PROJECT

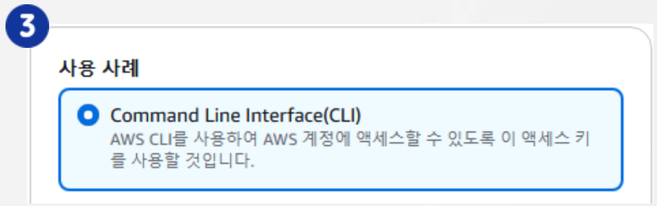
서버 환경 구성

IAM 설정

사용자 세부 정보 지정
- 이름 : renew-eks-mgr-user

권한 설정
- 권한 옵션 : 직접 정책 연결
- 권한 정책 : Administrator

액세스 키 설정
- 사용 사례 : CLI
- 설명 태그 값 : renew-eks-iam-accesskey



PROJECT

서버 환경 구성

AWS CLI 설치

Unzip 설치
- sudo apt-get install -y unzip

AWS CLI 파일 다운로드 및 압축 해제
- 다운로드 : awscliv2.zip 파일 경로
- 압축 해제 : unzip awscliv2.zip

설치 여부 확인
- AWS CLI 설치 확인 : sudo ./aws/install
- 버전 확인 : aws --version

1
```
ubuntu@ip-192-168-0-41:~$ sudo apt-get install -y unzip
```

2
```
ubuntu@ip-192-168-0-41:~$ curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -
o "awscliv2.zip"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 64.7M  100 64.7M    0     0  20.7M      0  0:00:03  0:00:03 --:--:-- 20.7M
```

3
```
ubuntu@ip-192-168-0-41:~$ unzip awscliv2.zip
```

4
```
ubuntu@ip-192-168-0-41:~$ sudo ./aws/install
You can now run: /usr/local/bin/aws --version
ubuntu@ip-192-168-0-41:~$ aws --version
aws-cli/2.24.1 Python/3.12.6 Linux/6.8.0-1021-aws exe/x86_64.ubuntu.22
```

AWS 계정 등록

AWS 계정 등록
- 액세스 키 ID, 비밀번호 : Bastion Server
- 리전명 : ap-northeast-2
- 출력 형식 : json

확인 : aws sts get-caller-identity

1
```
ubuntu@ip-192-168-0-41:~$ aws configure
AWS Access Key ID [None]: AKIAQPOMUL3CQOYRBTVB
AWS Secret Access Key [None]: 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
Default region name [None]: ap-northeast-2
Default output format [None]: json
```

2
```
ubuntu@ip-192-168-0-41:~$ aws sts get-caller-identity
{
    "UserId": "AIDAQPOMUL3CQTBXJWP7Y",
    "Account": "033178279621",
    "Arn": "arn:aws:iam::033178279621:user/renew-eks-mgr-user"
}
```

PROJECT

서버 환경 구성

Kubectl 설치

바이너리 다운로드

- 다운로드 : Linux 버전 Kubernetes 1.32.0 파일

체크섬 다운로드

- 다운로드 : Linux 버전 Kubernetes 1.32.0 체크섬 파일

적용

- 실행 권한 적용 : chmod +x ./kubectl
- 복사 : mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$HOME/bin:$PATH
- Bash 셸 사용 : echo 'export PATH=$HOME/bin:$PATH' >> ~/.bashrc

1
```
ubuntu@ip-192-168-0-41:~$ curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.32.0/2024-12-20/bin/linux/amd64/kubectl
```

2
```
ubuntu@ip-192-168-0-41:~$ curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.32.0/2024-12-20/bin/linux/amd64/kubectl.sha256
```

3
```
ubuntu@ip-192-168-0-41:~$ chmod +x ./kubectl
```

4
```
ubuntu@ip-192-168-0-41:~$ mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$HOME/bin:$PATH
```

5
```
ubuntu@ip-192-168-0-41:~$ echo 'export PATH=$HOME/bin:$PATH' >> ~/.bashrc
```

Eksctl 설치

설치

- 다운로드 : eksctl 유닉스 경로 참조

확인

- 버전 확인 : eksctl version

1
```
ubuntu@ip-192-168-0-41:~$ ARCH=amd64
ubuntu@ip-192-168-0-41:~$ PLATFORM=$(uname -s)_$ARCH
ubuntu@ip-192-168-0-41:~$ curl -sLO "https://github.com/eksctl-io/eksctl/releases/latest/download/eksctl_$PLATFORM.tar.gz"
ubuntu@ip-192-168-0-41:~$ curl -sL "https://github.com/eksctl-io/eksctl/releases/latest/download/eksctl_checksums.txt" | grep $PLATFORM | sha256sum --check
eksctl_Linux_amd64.tar.gz: OK
ubuntu@ip-192-168-0-41:~$ tar -xzf eksctl_$PLATFORM.tar.gz -C /tmp && rm eksctl_$PLATFORM.tar.gz
ubuntu@ip-192-168-0-41:~$ sudo mv /tmp/eksctl /usr/local/bin
```

2
```
ubuntu@ip-192-168-0-41:~$ eksctl version
0.203.0
```

PROJECT

클러스터 생성

클러스터 및 서비스 생성

클러스터 생성

- 이름 : k8s-renew
- 리전명 : ap-northeast-2
- 노드 그룹 이름 : k8s-ng
- 영역 : ap-northeast-2a, ap-northeast-2c
- 노드 개수 : 2
- 노드 타입 : t3.medium
- 노드 크기 : 20

서비스 생성

- Deployment 생성 : kubectl create deploy webtest --image=nginx:1.14 --replicas=3 --port=80
- Service 생성 : kubectl expose deploy webtest --port=80 --type=loadbalancer
- 접속 : webtest의 External-IP를 통해 접속

1

```
ubuntu@ip-192-168-0-41:~$ eksctl create cluster \
    --name k8s-renew \
    --region ap-northeast-2 \
    --with-oidc \
    --nodegroup-name k8s-ng \
    --zones ap-northeast-2a,ap-northeast-2c \
    --nodes 2
    --node-type t3.medium \
    --node-volume-size=20 \
    --managed
```

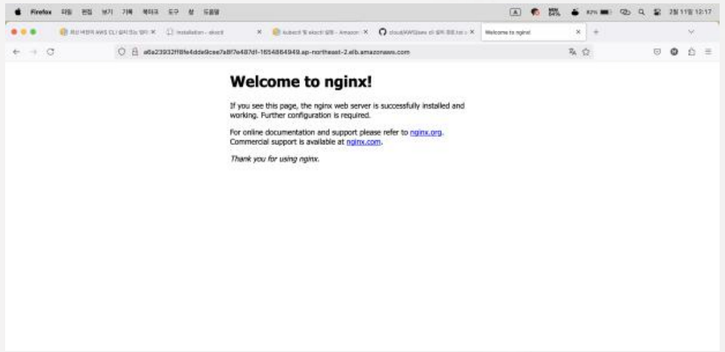
2

```
ubuntu@ip-192-168-0-41:~$ kubectl create deploy webtest --image=nginx1.14 --replicas=3 --port=80
deployment.apps/webtest created
ubuntu@ip-192-168-0-41:~$ kubectl get deploy
NAME      READY   UP-TO-DATE   AVAILABLE   AGE
webtest   0/3     3            0           13s
```

3

```
ubuntu@ip-192-168-0-41:~$ kubectl expose deploy webtest --port=80 --type=LoadBalancer
service/webtest exposed
ubuntu@ip-192-168-0-41:~$ kubectl get svc
NAME         TYPE           CLUSTER-IP       EXTERNAL-IP
                                PORT(S)        AGE
kubernetes   ClusterIP      10.100.0.1       <none>
                                443/TCP        23m
webtest      LoadBalancer   10.100.235.33    a6a23932ff8fe4dde9cee7a8f7e487df-1654864949.ap-
northeast-2.elb.amazonaws.com   80:31386/TCP   7s
```

4



PROJECT

Load Balancer 설치

Load Balancer 설치

HELM 설치

- 파일 다운로드 : Helm 파일이 있는 경로
- 사용자에게 권한 부여 : chmod 700 get_helm.sh
- 설치 : ./get_helm.sh

IAM 정책 생성

- 정책 다운로드 : load balancer controller 파일이 있는 경로
- 정책 이름 : AWSLoadBalancerControllerPolicy
- 정책 파일 : iam_policy.json

IAM 역할 생성

- 클러스터 : k8s-renew
- 네임스페이스 : kube-system
- 이름 : aws-load-balancer-controller
- 역할 이름 : AmazonEKSLoadBalancerControllerRole
- 접근 : aws:iam::111122223333:policy/AWSLoadBalancerControllerIAMPolicy (1111...3333 부분을 계정 ID로 변경)

1

```
ubuntu@ip-192-168-0-41:~$ curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3
ubuntu@ip-192-168-0-41:~$ chmod 700 get_helm.sh
ubuntu@ip-192-168-0-41:~$ ./get_helm.sh
Downloading https://get.helm.sh/helm-v3.17.0-linux-amd64.tar.gz
Verifying checksum... Done.
Preparing to install helm into /usr/local/bin
helm installed into /usr/local/bin/helm
```

2

```
ubuntu@ip-192-168-0-41:~$ aws iam create-policy \
    --policy-name AWSLoadBalancerControllerIAMPolicy \
    --policy-document file://iam_policy.json
{
    "Policy": {
        "PolicyName": "AWSLoadBalancerControllerIAMPolicy",
        "PolicyId": "ANPAQPOMUL3CVAQOLBR34",
        "Arn": "arn:aws:iam::033178279621:policy/AWSLoadBalancerControllerIAMPolicy",
        "Path": "/",
        "DefaultVersionId": "v1",
        "AttachmentCount": 0,
        "PermissionsBoundaryUsageCount": 0,
        "IsAttachable": true,
        "CreateDate": "2025-02-11T03:20:51+00:00",
        "UpdateDate": "2025-02-11T03:20:51+00:00"
    }
}
```

3

```
ubuntu@ip-192-168-0-41:~$ eksctl create iamserviceaccount \
  --cluster=k8s-renew \
  --namespace=kube-system \
  --name=aws-load-balancer-controller \
  --role-name AmazonEKSLoadBalancerControllerRole \
  --attach-policy-arn=arn:aws:iam::033178279621:policy/AWSLoadBalancerControllerIAMPolicy \
  --approve
```

PROJECT

Load Balancer 설치

Load Balancer 설치

설치
- 리포지토리 추가 : helm repo add eks https://aws.github.io/eks-charts
- 업데이트 : helm repo update eks
- 클러스터 이름 : k8s-renew
- 서비스 계정 생성 : false
- 서비스 계정 이름 : aws-load-balancer-controller

확인 : kubectl get deployment -n kube-system aws-load-balancer-controller

1
```
ubuntu@ip-192-168-0-41:~$ helm repo add eks https://aws.github.io/eks-charts
"eks" has been added to your repositories
```

2
```
ubuntu@ip-192-168-0-41:~$ helm repo update eks
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "eks" chart repository
Update Complete. ⎈Happy Helming!⎈
```

3
```
ubuntu@ip-192-168-0-41:~$ helm install aws-load-balancer-controller eks/aws-load-balancer-controller \
  -n kube-system \
  --set clusterName=k8s-renew \
  --set serviceAccount.create=false \
  --set serviceAccount.name=aws-load-balancer-controller
NAME: aws-load-balancer-controller
LAST DEPLOYED: Tue Feb 11 03:29:48 2025
NAMESPACE: kube-system
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
AWS Load Balancer controller installed!
```

4
```
ubuntu@ip-192-168-0-41:~$ kubectl get deployment -n kube-system aws-load-balancer-controller
NAME                           READY   UP-TO-DATE   AVAILABLE   AGE
aws-load-balancer-controller   2/2     2            2           87s
```

PROJECT

Load Balancer 배포

NLB 생성

NLB 생성

- 네임스페이스 생성 : renew-nlb-app
- Deployment 생성 : sample-deployment.yaml 파일 작성 후 적용
- 서비스 생성 : sample-service.yaml 파일 작성 후 적용

1

```
ubuntu@ip-192-168-0-41:~$ kubectl create namespace renew-nlb-app
namespace/renew-nlb-app created
```

2

```
ubuntu@ip-192-168-0-41:~$ vi sample-deployment.yaml
ubuntu@ip-192-168-0-41:~$ kubectl apply -f sample-deployment.yaml
deployment.apps/renew-nlb-app created
apiVersion: apps/v1
kind: Deployment
metadata:
  name: renew-nlb-app
  namespace: renew-nlb-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: public.ecr.aws/nginx/nginx:1.23
          ports:
            - name: tcp
              containerPort: 80
```

3

```
ubuntu@ip-192-168-0-41:~$ vi sample-service.yaml
ubuntu@ip-192-168-0-41:~$ kubectl apply -f sample-service.yaml
service/nlb-sample-service created
apiVersion: v1
kind: Service
metadata:
  name: nlb-sample-service
  namespace: renew-nlb-app
  annotations:
    service.beta.kubernetes.io/aws-load-balancer-type: external
    service.beta.kubernetes.io/aws-load-balancer-nlb-target-type: ip
    service.beta.kubernetes.io/aws-load-balancer-scheme: internet-facing
spec:
  ports:
    - port: 80
      targetPort: 80
      protocol: TCP
  type: LoadBalancer
  selector:
    app: nginx
```

PROJECT

Load Balancer 배포

ALB 생성

ALB 생성
- 파일 다운로드 : 2048_full.yaml 파일이 있는 경로
- Deployment 확인 : kubectl get pod -n game-2048
- Ingress 확인 : kubectl get ingress -n game-2048

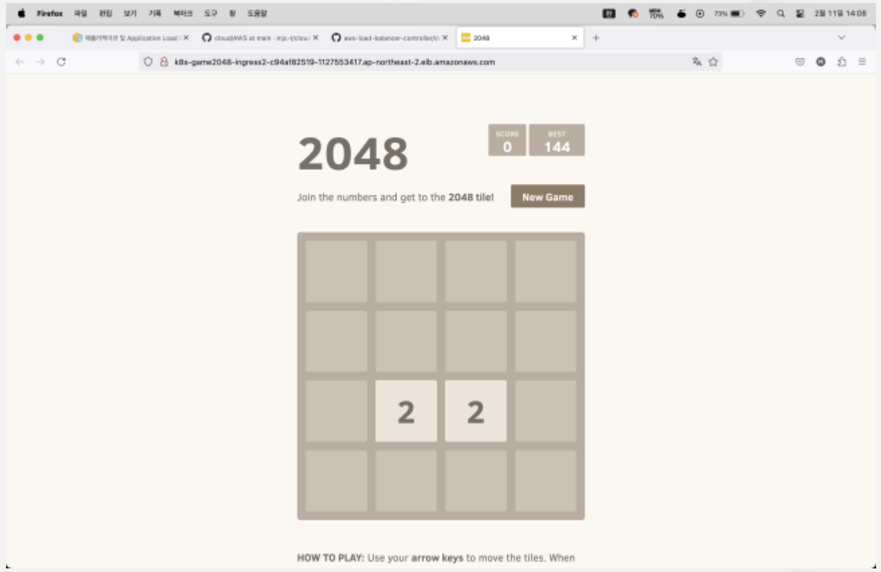
2048 화면 출력
- 화면 출력 : Ingress의 주소를 복사하여 화면 출력

1
```
ubuntu@ip-192-168-0-41:~$ kubectl apply -f https://raw.githubusercontent.com/kubernetes-sigs/
aws-load-balancer-controller/v2.11.0/docs/examples/2048/2048_full.yaml
namespace/game-2048 created
deployment.apps/deployment-2048 created
service/service-2048 created
ingress.networking.k8s.io/ingress-2048 created
```

2
```
ubuntu@ip-192-168-0-41:~$ kubectl get po -n 2048
No resources found in 2048 namespace.
```

3
```
ubuntu@ip-192-168-0-41:~$ kubectl get po -n game-2048
NAME                              READY   STATUS    RESTARTS   AGE
deployment-2048-85f8c7d69-dcgj2   1/1     Running   0          84s
deployment-2048-85f8c7d69-dhx5p   1/1     Running   0          84s
deployment-2048-85f8c7d69-fd9w8   1/1     Running   0          84s
deployment-2048-85f8c7d69-h8qcv   1/1     Running   0          84s
deployment-2048-85f8c7d69-vcvmn   1/1     Running   0          84s
```

4



EXPECTED EFFECT

기대 효과

운영 효율 향상

- 실제 클라우드 환경에서의 쿠버네티스 운영 경험 축적
- AWS의 클라우드 서비스와 쿠버네티스 활용 능력 획득
- 컨테이너의 운영 효율성 향상 및 서비스 안정성 확보

쉬운 작업 처리

- AWS에서 EKS를 효과적으로 처리 가능
- 쿠버네티스 클러스터를 관리 및 배포하는 실무 능력 향상
- 클라우드 환경에서의 쿠버네티스 클러스터를 쉽게 처리

EXPECTED EFFECT

활용 방안

1. 기업 내 클라우드 환경을 구축 및 운영을 하여 효율적으로 컨테이너 및 노드 관리를 할 수 있게 되어서 운영 및 유지보수 효율성이 향상될 수 있다.

2. 다양한 클라우드 서비스를 통해 애플리케이션 확장 가능성 및 운영 환경이 제공되어 쉽게 관리할 수 있다.

3. 서비스 간 트래픽 조정을 통해 지속적인 운용 최적화 및 성능 개선을 가능하게 하여 확장성과 유연성을 강화할 수 있다.

TEAM RENEW

감사합니다

권택 박소정 안웅렬 윤승원 황준서