



APEX

# AWS EKS 프로젝트

[ 2 0 2 5 . 0 9 . 0 4 ~ 2 0 2 5 . 1 0 . 2 1 ]



APEX  
<목차>



**01**    프로젝트 개요

**04**    프로젝트 수행결과

**02**    프로젝트 팀 구성 및 역할

**05**    현장 교강사 평가

**03**    수행 절차 및 방법

**06**    기타 활동 자료

# 1. 프로젝트 개요

## 프로젝트 및 선정 배경 / 기획의도



- AWS EKS를 활용하여 확장가능하고 안정적인 컨테이너 기반 인프라 구축
- NLB와 ALB를 실현하여 효율적인 트래픽 관리

## 활용장비 및 재료



- aws console
- xshell 8

## 프로젝트 내용



- eks 클러스터 구축을 위해 클러스터 설계 및 생성
- nlb & alb 구현을 위한 설계 및 생성

## 프로젝트 구조



- 계획 -> 시스템 구성 및 구축 -> 최종 결과



## 2. 프로젝트 팀 구성 및 역할

팀원, 역할, 담당 역할

### <이정일>

- 자료 조사
- 프로젝트 실습 진행

#조장

### <임재근>

- 자료 조사
- 프로젝트 실습 진행

#팀원

### <서재권>

- 프로젝트 보고서 작성
- PPT 제작

#팀원

### <김민호>

- 프로젝트 보고서 작성
- PPT 제작

#팀원

### 3. 프로젝트 수행절차 및 방법





## APEX

# 4. 프로젝트 수행결과 | Bastion Server 생성

### 1-1. 인스턴스 생성(1)

- 인스턴스
  - 이름 : apex-bastion
  - AMI : Ubuntu Server 22.04 LTS
  - 인스턴스 유형 : t3.micro
- 키페어
  - 이름 : apex-key
  - 키페어유형 : RSA
  - 파일 형식 : .pem

1

EC2 > 인스턴스 > 인스턴스 시작

인스턴스 시작 정보

Amazon EC2를 사용하면 AWS 클라우드에서 실행되는 가상 머신 또는 인스턴스를 생성할 수 있습니다. 아래의 간단한 단계에 따라 빠르게 시작할 수 있습니다.

이름 및 태그 정보

이름  
apex-bastion 추가 태그 추가

▼ 애플리케이션 및 OS 이미지(Amazon Machine Image) 정보

An AMI contains the operating system, application server, and applications for your instance. If you don't see a suitable AMI below, use the search bar or choose [Browse more AMIs](#).

Q 수천 개의 애플리케이션 및 OS 이미지를 포함하는 전체 카탈로그 검색

최근 사용 Quick Start

Amazon Linux

aws

macOS

Mac

Ubuntu

ubuntu

Windows

Microsoft

Red Hat

Red Hat

SUSE Linux

SUSE

Debian

debian

더 많은 AWS, M 커뮤니

Amazon Machine Image(AMI)

Ubuntu Server 24.04 LTS (HVM), SSD Volume Type  
ami-00e73adb2e2c80366 (64비트(x86)) / ami-0607797cadde98e9b (64비트(Arm))  
가상화: hvm ENA 활성화됨: true 루트 디바이스 유형: ebs 프리 티어 사용 가능

설명  
Ubuntu Server 24.04 LTS (HVM),EBS General Purpose (SSD) Volume Type. Support available from Canonical (<http://www.ubuntu.com/cloud/services>).  
Canonical, Ubuntu, 24.04, amd64 noble image

아키텍처

AMI ID

게시 날짜

사용자 이름

64비트(x86)

ami-00e73adb2e2c80366

2025-08-21

ubuntu

확인된 공급 업체

2

EC2 > 키 페어 > 키 페어 생성

키 페어 생성 정보

키 페어  
프라이빗 키와 퍼블릭 키로 구성되는 키 페어는 인스턴스에 연결할 때 자격 증명을 증명하는 데 사용하

이름  
apex-key  
이름에는 최대 255개의 ASCII 문자가 포함됩니다. 앞 또는 뒤에 공백을 포함할 수 없습니다.

키 페어 유형 정보

RSA

ED25519

프라이빗 키 파일 형식

.pem

OpenSSH와 함께 사용

.ppk

PuTTY와 함께 사용



## APEX

# 4. 프로젝트 수행결과 | Bastion Server 생성

### 1-2. 인스턴스 생성(2)

- 네트워크 설정
  - VPC : 기본값
  - 보안그룹 생성
    - 이름 : apex-bastion-sg
- 생성 유형 : ssh (위치무관)  
확인

① EC2 > 인스턴스 > 인스턴스 시작

▼ 네트워크 설정 정보

VPC - 필수 | 정보  
vpc-03436fd0bf13e073a (default-vpc)  
172.31.0.0/16

서브넷 | 정보  
subnet-09746324ac2dd728c default-subnet-1  
VPC: vpc-03436fd0bf13e073a 소유자: 913524917762 가용 영역: ap-northeast-2a (apne2-az1)  
영역 유형: 가용 영역 사용 가능한 IP 주소: 4091 CIDR: 172.31.0.0/20

퍼블릭 IP 자동 할당 | 정보  
활성화  
프리 티어 허용 범위를 벗어나는 경우 추가 요금이 적용됩니다.

방화벽(보안 그룹) | 정보  
보안 그룹은 인스턴스에 대한 트래픽을 제어하는 방화벽 규칙 세트입니다. 특정 트래픽이 인스턴스에 도달하도록 허용하는 규칙을 추

○ 보안 그룹 생성 ○ 기존 보안 그룹 선택

보안 그룹 이름 - 필수  
apex-bastion-sg  
이 보안 그룹은 모든 네트워크 인터페이스에 추가됩니다. 보안 그룹을 만든 후에는 이름을 편집할 수 없습니다. 최대 길이는 255자입니다.

설명 - 필수 | 정보  
launch-wizard-1 created 2025-09-04T08:25:04.635Z

인바운드 보안 그룹 규칙  
▼ 보안 그룹 규칙 1 (TCP, 22, 0.0.0.0/0)

유형 | 정보  
ssh

소스 유형 | 정보  
위치 무관

프로토콜 | 정보  
TCP

원본 | 정보  
Q CIDR, 접두사 목록 또는 보안 그룹 추가  
0.0.0.0/0 X



## 4. 프로젝트 수행결과 | Bastion Server 생성

### 1-3. XSELL 연결

- 연결
  - 이름 : apex-bastion
  - 호스트 : 43.201.160.129(퍼블릭 IPv4주소)
  - 포트 번호 : 22
- 사용자 인증
  - Public Key 사용
    - apex-key 가져오기

#### ① 세션 등록 정보

범주(C):

연결

사용자 인증

로그인 프롬프트

로그인 스크립트

SSH

보안

터널링

SFTP

TELNET

RLOGIN

SERIAL

RDP

일반

이름(N): apex-bastion

프로토콜(P): SSH

호스트(H): 43.201.160.129

포트 번호(O): 22

아이콘:

#### ② 세션 등록 정보

범주(C):

연결 > 사용자 인증

인증 방법과 기타 관련 매개 변수들을 선택하십시오.

이 섹션은 로그인 할 때 시간을 절약하기 위해 사용할 수 있습니다. 그러나 보안을 중요시하는 경우 이 섹션을 비워 두는 것이 좋습니다.

☐ 인증 프로필 사용(A)

인증 프로필(F): <지정하지 않음> 찾아보기(B)...

사용자 이름(U): ubuntu

암호(P):

방법(M):

☐ Password

☒ Public Key

☐ Keyboard Interactive

☐ GSSAPI

설정(S)...

위로(U)

아래로(D)

확인

연결

취소

#### ③ Public Key 설정

키 파일(F)

사용자 키(U): apex-key (1)

암호(P):

PKCS #11(K)

미들웨어 경로(M):

토큰 핀(T):

CAPIC(C)





## 4. 프로젝트 수행결과 | Bastion Server 환경구성

## 2-1. iam 사용자 생성

- 사용자 세부 정보 지정
  - 이름 : apex-eks-user
- 권한 설정
  - 권한 옵션
    - 직접 정책 연결
  - 정책 검색
    - AdministratorAccess
- 검토 및 생성



**1**

IAM > 사용자 > 사용자 생성

1단계  
● 사용자 세부 정보 지정

### 사용자 세부 정보 지정

**사용자 이름**  
apex-eks-user

사용자 이름은 최대 64자까지 가능합니다. 유효한 문자: A~Z, a~z, 0~9, -, \_, ., =, +, @, ^, `.

☐ AWS Management Console에 대한 사용자 액세스:  
사람에게 콘솔 액세스 권한을 제공하는 경우 IAM Identity Center를 사용합니다.

**이 IAM 사용자를 생성한 후 액세스 키 또는 AWS Keyspaces에 대한 서비스별 보안 인증 정보를 설정합니다. 자세히 알아보기**

2단계  
● 권한 설정

### 권한 설정

기존 그룹에 사용자를 추가하거나 새 그룹을 생성합니다. 직무별로 사용자의 권한을 관리하려면 그룹을 사용하는 것이 좋습니다. [자세히 알아보기](#)

**권한 옵션**

- ☐ 그룹에 사용자 추가  
기존 그룹에 사용자를 추가하거나 새 그룹을 생성합니다. 그룹을 사용하여 직무별로 사용자 권한을 관리하는 것이 좋습니다.
- ☒ 직접 정책 연결  
관리형 정책을 사용자에게 직접 연결합니다. 사용자에게 연결하는 대신, 정책을 그룹에 연결한 후 사용자를 적절한 그룹에 추가하는 것이 좋습니다.
- ☐ 권한 복사  
기존 사용자의 모든 그룹 멤버십, 연결된 관리형 정책 및 인라인 정책을 복사합니다.

**권한 정책 (1/1389)** [정책 생성](#)

새 사용자에게 연결할 정책을 하나 이상 선택합니다.

필터링 기준 유형  
모든 유형

정책 이름	유형	연결된 ...
<input type="checkbox"/> AccessAnalyzerSer...	AWS 관리형	0
<input checked="" type="checkbox"/> AdministratorAccess	AWS 관리형 - 직무	5
<input type="checkbox"/> AdministratorAcce...	AWS 관리형	0

## 4. 프로젝트 수행결과 | Bastion Server 환경구성

### 2-2. 액세스 키 생성

- 액세스 키 모범 사례 및 대안
  - Command Line Interface(CLI)
- 선택 설명 태그 설정
- 토큰 생성 및 설치
- .CSV 파일 다운로드
- 생성 확인

①

1단계 액세스 키 모범 사례 및 대안

2단계 - 선택 사항 설명 태그 설정

3단계 액세스 키 검색

### 액세스 키 모범 사례 및 대안 정보

보안 개선을 위해 액세스 키와 같은 장기 자격 증명을 사용하지 마세요. 다음과 같은 사용 사례와 대안을 고려하세요.

**사용 사례**

- ☒ **Command Line Interface(CLI)**  
AWS CLI를 사용하여 AWS 계정에 액세스할 수 있도록 이 액세스 키를 사용할 것입니다.
- ☐ 로컬 코드  
로컬 개발 환경의 애플리케이션 코드를 사용하여 AWS 계정에 액세스할 수 있도록 이 액세스 키를 사용할 것입니다.

②

1단계 액세스 키 모범 사례 및 대안

2단계 - 선택 사항 설명 태그 설정

3단계 액세스 키 검색

### 액세스 키 검색 정보

**액세스 키**  
분실하거나 잊어버린 비밀 액세스 키는 검색할 수 없습니다. 대신 새 액세스 키를 생성하고 이전 키를 비활성화합니다.

액세스 키 | 비밀 액세스 키

표시

### 액세스 키 모범 사례

- 액세스 키를 일반 텍스트, 코드 리포지토리 또는 코드로 저장해서는 안 됩니다.
- 더 이상 필요 없는 경우 액세스 키를 비활성화하거나 삭제합니다.
- 최소 권한을 활성화합니다.
- 액세스 키를 정기적으로 교체합니다.

액세스 키 관리에 대한 자세한 내용은 [AWS 액세스 키 관리 모범 사례](#)를 참조하세요.

[.csv 파일 다운로드](#) [완료](#)



## 4. 프로젝트 수행결과 | Bastion Server 환경구성

### 2-3. 관리 도구 설치

- AWS CLI 설치
- 관리시스템에 AWS 계정 등록 및 확인
- k8s 관리 도구인 kubectl 설치(최신버전)

①

```
ubuntu@ip-172-31-7-236:~$ sudo apt-get install -y unzip
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Suggested packages:
  zip
```

②

```
ubuntu@ip-172-31-7-236:~$ curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
```

③

```
ubuntu@ip-172-31-7-236:~$ aws configure
```

④

```
ubuntu@ip-172-31-7-236:~$ curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.31.0/2024-09-12/bin/linux/amd64/kubectl
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total   Spent    Left  Speed
100 53.7M  100 53.7M    0     0  9875k      0  0:00:05  0:00:05 --:--:-- 12.2M
ubuntu@ip-172-31-7-236:~$ chmod +x ./kubectl
ubuntu@ip-172-31-7-236:~$ mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$HOME/bin:$PATH
ubuntu@ip-172-31-7-236:~$ kubectl version
Client Version: v1.31.0-eks-a737599
Kustomize Version: v5.4.2
The connection to the server localhost:8080 was refused - did you specify the right host or port?
ubuntu@ip-172-31-7-236:~$ ls
aws  awscliv2.zip  bin  kubectl
ubuntu@ip-172-31-7-236:~$
```

⑤

```
ubuntu@ip-172-31-7-236:~$ ARCH=amd64
ubuntu@ip-172-31-7-236:~$ PLATFORM=$(uname -s)_$ARCH
ubuntu@ip-172-31-7-236:~$ curl -sLO "https://github.com/eksctl-io/eksctl/releases/latest/download/eksctl_${PLATFORM}.tar.gz"
ubuntu@ip-172-31-7-236:~$ curl -sL "https://github.com/eksctl-io/eksctl/releases/latest/download/eksctl_checksums.txt" | grep $PLATFORM | sha256sum --check
eksctl_linux_amd64.tar.gz: OK
ubuntu@ip-172-31-7-236:~$ tar -xzf eksctl_${PLATFORM}.tar.gz -C /tmp && rm eksctl_${PLATFORM}.tar.gz
ubuntu@ip-172-31-7-236:~$ sudo mv /tmp/eksctl /usr/local/bin
ubuntu@ip-172-31-7-236:~$ eksctl version
0.214.0
ubuntu@ip-172-31-7-236:~$
```





## 4. 프로젝트 수행결과 | EKS Cluster 생성

### 3-1. EKS Cluster 생성

- EKS 클러스터 생성 후 확인
  - 이름 : apex-k8s
  - 리전 : ap-northeast-2
  - 노드 그룹 : apex-k8s-ng
  - 노드 : 2
- 테스트용 서비스, 디플로이먼트 생성 후 확인
  - 이름 : webtest
  - 타입 : 로드밸런스

①

```
ubuntu@ip-172-31-7-236:~$ eksctl create cluster
--name apex-eks \
--region ap-northeast-2 \
--with-oidc \
--nodegroup-name apex-ng \
--zones ap-northeast-2a,ap-northeast-2c \
--nodes 2 \
--node-type t3.medium \
2025-09-04 08:44:43 [i] eksctl version 0.214.0
2025-09-04 08:44:43 [i] using region ap-northeast-2
2025-09-04 08:44:43 [i] setting availability zones to [ap-northeast-2b ap-northeast-2c ap-northeast-2d]
2025-09-04 08:44:43 [i] subnets for ap-northeast-2b - public:192.168.0.0/19 private:192.168.0.0/19
2025-09-04 08:44:43 [i] subnets for ap-northeast-2c - public:192.168.32.0/19 private:192.168.128.0/19
2025-09-04 08:44:43 [i] subnets for ap-northeast-2d - public:192.168.64.0/19 private:192.168.160.0/19
2025-09-04 08:44:43 [i] nodegroup "ng-17df2cae" will use " " [AmazonLinux2023/1.32]
```

로드 밸런서 (1)

Elastic Load Balancing은 수신 트래픽의 변화에 따라 자동으로 로드 밸런서 용량을 확장합니다.

로드 밸런서 필터링

이름	상태	유형	체계	IP 주소 유형
afabfd132fc5247b1803...	-	classic	-	-

②

```
ubuntu@ip-172-31-7-236:~$ kubectl create deployment webtest --
--replicas=5
deployment.apps/webtest created
ubuntu@ip-172-31-7-236:~$ kubectl get pods -o wide

NAME                                READY   STATUS    RESTART   NOMINATED NODE   READI
ubuntu@ip-172-31-7-236:~$ kubectl expose deployment webtest --port=80 --type=LoadBalancer
service/webtest exposed
ubuntu@ip-172-31-7-236:~$ kubectl get services

NAME                                TYPE                CLUSTER-IP      EXTERNAL-IP
kubernetes                         ClusterIP           10.100.0.1      <none>
webtest                            LoadBalancer       10.100.171.182  afabfd132
northeast-2.elb.amazonaws.com      80:31424/TCP       8s
ubuntu@ip-172-31-7-236:~$
```

AWSLoadBalancerControllerIAMPolicy 정보

정책 세부 정보	생성 시간
유형	September 03, 2025, 18:03 (UTC+09:00)
고객 관리형	
편집 시간	ARN
September 03, 2025, 18:03 (UTC+09:00)	arn:aws:iam::913524917762:policy/AWSLoadBalancerControllerIAMPolicy

③

```
ubuntu@ip-172-31-7-236:~$ kubectl delete deployment.apps webtest
deployment.apps "webtest" deleted
ubuntu@ip-172-31-7-236:~$
ubuntu@ip-172-31-7-236:~$ kubectl delete svc webtest
service "webtest" deleted
```



## 4. 프로젝트 수행결과 | LoadBalancer Controller 생성

### 4-1. Helm 설치 및 정책 생성

- 스크립트로 Helm 설치
  - `curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3`
- 정책 설치
  - `curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.11.0/docs/install/iam_policy.json`
- 설치된 정책으로 IAM 정책 생성
  - `AWSLoadBalancerControllerIAMPolicy`
  - document file : `iam_policy.json`

```
① ubuntu@ip-172-31-7-236:~$ curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3
ubuntu@ip-172-31-7-236:~$ chmod 700 get_helm.sh
ubuntu@ip-172-31-7-236:~$ ./get_helm.sh
Downloading https://get.helm.sh/helm-v3.18.6-linux-amd64.tar.gz
Verifying checksum... Done.
Preparing to install helm into /usr/local/bin
helm installed into /usr/local/bin/helm
ubuntu@ip-172-31-7-236:~$
```

```
② ubuntu@ip-172-31-7-236:~$ curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.7.2/docs/install/iam_policy.json
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left   Speed
100 8446  100 8446    0     0  29758      0 --:--:-- --:--:-- --:--:-- 29844
ubuntu@ip-172-31-7-236:~$
```

```
③ ubuntu@ip-172-31-7-236:~$ aws iam create-policy \
--policy-name AWSLoadBalancerControllerIAMPolicy \
--policy-document file://iam_policy.json
```



## 4. 프로젝트 수행결과 | LoadBalancer Controller 생성

### 4-2. IAM 서비스 어카운트 생성

- ARN 확인
  - (IAM > 정책 > AWS LoadBalancer Controller IAM Policy)
- 서비스 어카운트
  - 이름 : aws-load-balancer-controller
  - 네임스페이스 : kube-system
  - 클러스터 : apex-eks

①

#### AWSLoadBalancerControllerIAMPolicy 정보

편집

삭제

##### 정책 세부 정보

유형  
고객 관리형

편집 시간  
September 03, 2025, 18:03 (UTC+09:00)

생성 시간  
September 03, 2025, 18:03 (UTC+09:00)

ARN  
 arn:aws:iam::913524917762:policy/AWSLoadBalancerControllerIAMPolicy

②

```
ubuntu@ip-172-31-7-236:~$ kubectl get sa -n kube-system | grep load
aws-load-balancer-controller                                0                27s
ubuntu@ip-172-31-7-236:~$
```

③

```
ubuntu@ip-172-31-7-236:~$ eksctl create iamserviceaccount \
--cluster=fabulous-sculpture-1756975483 \
--namespace=kube-system \
--name=aws-load-balancer-controller \
--role-name AmazonEKSLoadBalancerControllerRole \
--attach-policy-arn=arn:aws:iam::913524917762:policy/AWSLoadBalancerControllerIAMPolicy \
--approve

2025-09-04 09:15:26 [✓] 1 iamserviceaccount (kube-system/aws-load-balancer-controller) was included (based on the include/exclude rules)
2025-09-04 09:15:26 [!] serviceaccounts that exist in Kubernetes will be excluded, use --override-existing-serviceaccounts to override
```

④

```
ubuntu@ip-172-31-7-236:~$ kubectl describe -n kube-system sa aws-load-balancer-controller
Name:               aws-load-balancer-controller
Namespace:          kube-system
Labels:             app.kubernetes.io/managed-by=eksctl
Annotations:        eks.amazonaws.com/role-arn: arn:aws:iam::913524917762:role/AmazonEKSLoadBalancerControllerRole
Image pull secrets:  <none>
Mountable secrets:  <none>
Tokens:             <none>
Events:             <none>
ubuntu@ip-172-31-7-236:~$
```



## 4. 프로젝트 수행결과 | LoadBalancer Controller 생성

### 4-3. LoadBalancer Controller 설치

- Helm 리포지토리 추가 및 업데이트
  - helm repo add eks
  - <https://aws.github.io/eks-charts>
  - helm repo update eks
- LoadBalancer Controller 설치
  - 이름 : aws-loadbalancer-controller

```
① ubuntu@ip-172-31-7-236:~$ helm repo add eks https://aws.github.io/eks-charts
"eks" has been added to your repositories
ubuntu@ip-172-31-7-236:~$ 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!*
ubuntu@ip-172-31-7-236:~$
```

```
② ubuntu@ip-172-31-7-236:~$ helm install aws-load-balancer-controller eks/aws-load-balancer-controller \
-n kube-system \
--set clusterName=fabulous-sculpture-1756975483 \
--set serviceAccount.create=false \
--set serviceAccount.name=aws-load-balancer-controller
NAME: aws-load-balancer-controller
LAST DEPLOYED: Thu Sep  4 09:18:23 2025
NAMESPACE: kube-system
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
AWS Load Balancer controller installed!
ubuntu@ip-172-31-7-236:~$
```

## 4. 프로젝트 수행결과 | Load Balancer 배포-ALB

### 5-1. 샘플 애플리케이션 배포(1)

- 네임스페이스 생성
  - 이름 : nlb-sample-app
- 디플로이먼트 생성
  - 야물 파일 생성, 적용
    - 이름 : apex-deployment.yaml
  - 이름 : nlb-sample-app
  - 이미지 : nginx:1.23
  - 포트 : 80

① `ubuntu@ip-172-31-7-236:~$ kubectl create namespace nlb-sample-app`  
namespace/nlb-sample-app created  
ubuntu@ip-172-31-7-236:~\$

② `ubuntu@ip-172-31-7-236:~$ vi apex-deployment.yaml`

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nlb-sample-app
  namespace: nlb-sample-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-172-31-7-236:~$ kubectl apply -f apex-deployment.yaml`  
deployment.apps/nlb-sample-app created  
ubuntu@ip-172-31-7-236:~\$

## 4. 프로젝트 수행결과 | Load Balancer 배포-ALB

### 5-2. 샘플 애플리케이션 배포(2)

- 서비스 생성
  - 야물 파일 생성, 적용
    - 이름 : apex-service.yaml
  - 이름 : nlb-sample-service
  - 포트 : 80
  - 타입 : LoadBalancer
- 서비스 생성 후 확인

```
① ubuntu@ip-172-31-7-236:~$ vi apex-service.yaml
ubuntu@ip-172-31-7-236:~$
apiVersion: v1
kind: Service
metadata:
  name: nlb-sample-service
  namespace: nlb-sample-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

② ubuntu@ip-172-31-7-236:~$ kubectl apply -f apex-service.yaml
service/nlb-sample-service created
ubuntu@ip-172-31-7-236:~$

ubuntu@ip-172-31-7-236:~$ kubectl get svc -n nlb-sample-app
NAME                                TYPE                CLUSTER-IP      EXTERNAL-IP      PORT(S)          AGE
nlb-sample-service                 LoadBalancer        10.100.206.250   <pending>        80:30576/TCP     18s
ubuntu@ip-172-31-7-236:~$

ubuntu@ip-172-31-7-236:~$ kubectl get pods -n nlb-sample-app -o wide
NAME                                READY   STATUS    RESTARTS   AGE   IP              NODE
nlb-sample-app-7b5b664bbc-6nq4t     1/1     Running   0          4m25s  192.168.64.78   ip-19
2-168-68-30.ap-northeast-2.compute.internal  <none>
nlb-sample-app-7b5b664bbc-dlh9t     1/1     Running   0          4m25s  192.168.92.250  ip-19
2-168-68-30.ap-northeast-2.compute.internal  <none>
nlb-sample-app-7b5b664bbc-jtj9v     1/1     Running   0          4m25s  192.168.29.10   ip-19
2-168-29-197.ap-northeast-2.compute.internal <none>
ubuntu@ip-172-31-7-236:~$
```



## 4. 프로젝트 수행결과 | Load Balancer 배포-ALB

### 6-1. 샘플 애플리케이션 배포(1)

- 네임스페이스 생성
  - 이름 : game-2048
- 디플로이먼트 생성
  - 야물 파일 생성, 적용
    - 이름 : apex game
  - 이름 : deployment-2048
  - 포트 : 80
- 서비스 생성
  - 이름 : service-2048
  - 타입 : NodePort

①

```
ubuntu@ip-172-31-7-236:~$ vi 2048_full.yaml
apiVersion: v1
kind: Namespace
metadata:
  name: game-2048
---
apiVersion: apps/v1
kind: Deployment
metadata:
  namespace: game-2048
  name: deployment-2048
spec:
  selector:
    matchLabels:
      app.kubernetes.io/name: app-2048
  replicas: 5
  template:
    metadata:
      labels:
        app.kubernetes.io/name: app-2048
    spec:
      containers:
        - image: public.ecr.aws/l6m2t8p7/docker-2048:latest
          imagePullPolicy: Always
          name: app-2048
          ports:
            - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  namespace: game-2048
  name: service-2048
spec:
  ports:
    - port: 80
      targetPort: 80
      protocol: TCP
  type: NodePort
  selector:
    app.kubernetes.io/name: app-2048
```

②

```
ubuntu@ip-172-31-7-236:~$ kubectl apply -f 2048_full.yaml
namespace/game-2048 created
deployment.apps/deployment-2048 created
service/service-2048 created
ingress.networking.k8s.io/ingress-2048 created
ubuntu@ip-172-31-7-236:~$ kubectl get pods -n game-2048
NAME                                READY   STATUS             RESTARTS   AGE
deployment-2048-bdbddc878-8vdl6     0/1     ContainerCreating   0          5s
deployment-2048-bdbddc878-c6qxf     0/1     ContainerCreating   0          5s
deployment-2048-bdbddc878-dmxwc     0/1     ContainerCreating   0          5s
deployment-2048-bdbddc878-g4dlt     1/1     Running             0          5s
deployment-2048-bdbddc878-jtcmm     1/1     Running             0          5s
ubuntu@ip-172-31-7-236:~$ kubectl get services -n game-2048
NAME          TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)          AGE
service-2048  NodePort    10.100.163.16 <none>        80:32507/TCP     8s
ubuntu@ip-172-31-7-236:~$ kubectl get targetgroupbindings -n game-2048
```



## APEX

# 4. 프로젝트 수행결과 | Load Balancer 배포-ALB

## 6-2. 샘플 애플리케이션 배포(2)

- ingress 생성
  - 야물 파일 생성, 적용
    - 이름 : ingress-2048.yaml
  - 이름 : game-2048
  - 포트 : 80
  - 클래스네임 : alb
- ingress 및 접속 주소 확인
- Route53

①

```
ubuntu@ip-172-31-7-236:~$ kubectl get ingress -n game-2048
NAME          CLASS  HOSTS  ADDRESS  PORTS  AGE
ingress-2048  alb    *      80       2m10s

---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  namespace: game-2048
  name: ingress-2048
  annotations:
    alb.ingress.kubernetes.io/scheme: internet-facing
    alb.ingress.kubernetes.io/target-type: ip
spec:
  ingressClassName: alb
  rules:
  - http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: service-2048
            port:
              number: 80
```

②

```
ubuntu@ip-172-31-7-236:~$ kubectl describe ingress ingress-2048 -n game-2048
Name:          ingress-2048
Labels:         <none>
Namespace:     game-2048
Address:
Ingress Class:  alb
Default backend: <default>
Rules:
  Host      Path  Backends
  ----      -
  *          /    service-2048:80 (192.168.73.15:80,192.168.94.82:80,192.168.21.81:80 + 2 more...)
Annotations:  alb.ingress.kubernetes.io/scheme: internet-facing
              alb.ingress.kubernetes.io/target-type: ip
```

③

레코드 (4) 정보

Automatic 모드는 최상의 필터 결과에 최적화된 현재 검색 동작입니다. [모드를 변경하려면 설정\(settings\)으로 이동합니다.](#)

Q 속성 또는 값을 기준으로 레코드 필터링

<input type="checkbox"/>	레코드 ...	유형	라우팅 ...	차별...	별칭	값/트래픽 라우팅 대상
<input type="checkbox"/>	leoandres....	A	단순	-	예	dualstack.k8s-game2048-ing...
<input type="checkbox"/>	leoandres....	NS	단순	-	아니요	ns-782.awsdns-33.net. ns-1959.awsdns-52.co.uk. ns-317.awsdns-39.com. ns-1413.awsdns-48.org.
<input type="checkbox"/>	leoandres....	SOA	단순	-	아니요	ns-782.awsdns-33.net. awsd...
<input type="checkbox"/>	www.leoa...	A	단순	-	예	dualstack.k8s-game2048-ing...



## 4. 프로젝트 수행결과 | Load Balancer 배포-ALB

### 6-2. 샘플 애플리케이션 배포(3)

- ingress 생성
  - 야물 파일 생성, 적용
    - 이름 : ingress-2048.yaml
  - 이름 : game-2048
  - 포트 : 80
  - 클래스네임 : alb
- ingress 및 접속 주소 확인

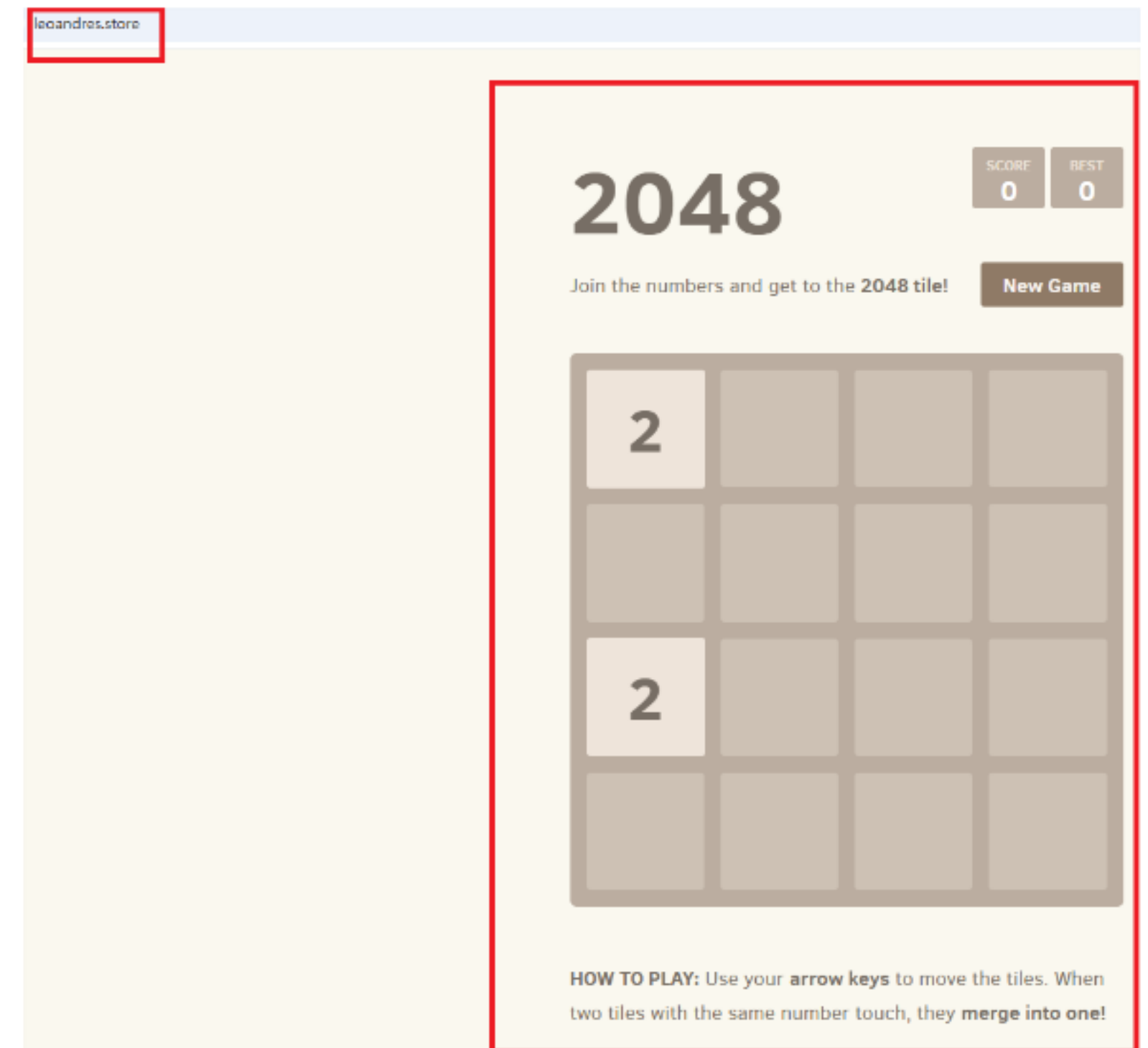
①

```
ubuntu@ip-172-31-0-103:~$ kubectl apply -f 2048_full.yaml
kubectl get pods -n game-2048
kubectl get services -n game-2048
kubectl get targetgroupbindings -n game-2048
namespace/game-2048 unchanged
deployment.apps/deployment-2048 unchanged
service/service-2048 unchanged
ingress.networking.k8s.io/ingress-2048 unchanged
NAME READY STATUS RESTARTS AGE
deployment-2048-bdbddc878-5126n 1/1 Running 0 80m
deployment-2048-bdbddc878-6pb2t 1/1 Running 0 80m
deployment-2048-bdbddc878-8kk7k 1/1 Running 0 80m
deployment-2048-bdbddc878-mdwd6 1/1 Running 0 80m
deployment-2048-bdbddc878-tjwkz 1/1 Running 0 80m
NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE
service-2048 NodePort 10.100.159.88 <none> 80:30771/TCP 80m
NAME SERVICE-NAME SERVICE-PORT TARGET-TYPE AGE
k8s-game2048-service2-f92763c77d service-2048 80 ip 31m
ubuntu@ip-172-31-0-103:~$ kubectl get ingress -n game-2048
NAME CLASS HOSTS ADDRESS PORTS AGE
ingress-2048 alb * k8s-game2048-ingress2-5e89baf5b8-377045271.ap-northeast-2.elb.amazonaws.com 80 81m
```

②

```
ubuntu@ip-172-31-0-103:~$ kubectl describe ingress ingress-2048 -n game-2048
Name: ingress-2048
Labels: <none>
Namespace: game-2048
Address: k8s-game2048-ingress2-5e89baf5b8-377045271.ap-northeast-2.elb.amazonaws.com
Ingress Class: alb
Default backend: <default>
Rules:
  Host Path Backends
  ----
  * / service-2048:80 (192.168.24.113:80,192.168.12.241:80,192.168.55.224:80 + 2 more...)
Annotations: alb.ingress.kubernetes.io/scheme: internet-facing
             alb.ingress.kubernetes.io/target-type: ip
Events:
  Type Reason Age From Message
  ----
  Normal SuccessfullyReconciled 19m ingress Successfully reconciled
  Normal SuccessfullyReconciled 18m ingress Successfully reconciled
  Normal SuccessfullyReconciled 13m ingress Successfully reconciled
```

③







## 5. 현장 교강사 평가

프로젝트 결과물에 대한 완성도 평가

프로젝트 진행 간 잘한 부분 & 아쉬운 부분

프로젝트 결과 피드백

프로젝트 수행 간 느낀 점



APEX

---

## 6. 기타 활동 자료

---



APEX

**감사합니다.**

---