

AWS EKS Project

KDT 클라우드보안 [2025.02.11 – 2025.03.06]

STEAM

김민지 010-7313-2639

kmj73132639@gmail.com

진승우 010-2929-1267

shyngwoo@gmail.com

현룡관 010-5689-2162

gusfydrhks@gmail.com

설예림 010-8285-6476

yerim7480@gmail.com

CONTENTS

1 프로젝트 개요

2 프로젝트 팀 구성 및 역할

3 수행 절차 및 방법

3-1. EKS 란?

3-2. 관리 서버 구축

3-3. Bastion Server 환경 구성

3-4. EKS 클러스터 생성

3-5. 로드밸런서 컨트롤러 생성

3-6. 로드밸런서 배포 – NLB, ALB

01. 프로젝트 개요



프로젝트 및 선정 배경 / 기획의도

기존 클러스터 운영 방식의 문제점을 개선

- 운영 및 유지보수의 복잡성
- 확장성
- 비용 및 보안 부담

=> EKS를 기반으로 한 자동화된 관리
시스템 도입



프로젝트 내용

- Amazon EKS 클러스터를 구성하고 VPC
네트워크를 설정해 안정적 운영 환경 구축
- AWS Load Balancer Controller를
설치하고 서비스를 배포해 트래픽을
효과적으로 분산
- IAM 서비스 어카운트와 역할 관리를 연동해
보안성 강화



활용장비

- AWS
- Xshell 8
- Vmware Workstation 17
 - Ubuntu 24.04



프로젝트 구조

- 로드밸런서 컨트롤러를 통해 트래픽을
효율적으로 분산
- 서비스 어카운트를 체계적으로 관리

02. 프로젝트 팀 구성 및 역할

훈련생	역할	담당 역할
김민지	문서 작성	프로젝트 계획서 및 보고서 작성
진승우	조장	프로젝트 수행
현룡관	작업	프로젝트 수행
설예림	PPT	프로젝트 결과보고서 PPT 작성

03. 수행절차 및 방법

3-1. EKS 란?

Amazon Elastic Kubernetes Service(Amazon EKS)는 Amazon Web Services(AWS) 및 자체 데이터 센터에서 Kubernetes 클러스터의 가용성과 확장성을 운영하고 유지할 필요가 없는 관리형 Kubernetes 서비스입니다.

Amazon EKS의 기능

기계 학습에 적합

- GPU 기반 인스턴스 및 AWS Neuron을 비롯한 다양한 컴퓨팅 옵션을 지원해 고성능 훈련과 지연시간이 짧은 추론이 가능하여 프로덕션 환경에서 모델이 효율적으로 실행됩니다.

하이브리드 배포

- *Amazon EKS Connector*를 사용하여 Amazon EKS 콘솔에서 모든 Kubernetes 클러스터와 해당 리소스를 볼 수 있습니다.

컴퓨팅

- 모든 범위의 Amazon EC2 인스턴스 유형과 Nitro 및 Graviton과 같은 AWS 혁신 기술을 활용하여 워크로드에 맞게 컴퓨팅을 최적화할 수 있습니다.



3-1. EKS 란?

Amazon EKS의 기능

네트워킹

- Amazon EKS는 Amazon VPC와 통합되어 Amazon EKS 클러스터에서 자체 Amazon VPC 보안 그룹 및 [네트워크 액세스 제어 목록](#)(ACL)을 사용할 수 있습니다.

보안

- 클러스터 및 애플리케이션을 보호하기 위해 AWS Identity and Access Management(IAM)와 통합됩니다. Amazon EKS를 사용하면 AWS IAM 권한을 Kubernetes 역할 기반 액세스 제어(RBAC)에 쉽게 매핑할 수 있습니다.

Observability

- 모니터링, 로깅 및 감사 기능을 위해 AWS Managed Service for Prometheus(AMP), Amazon CloudWatch, Amazon CloudTrail, Amazon GuardDuty와 통합됩니다.



3-2. 관리서버 구축

EC2 인스턴스 생성

- 이름: k8s-mgr-system
- AMI: ubuntu22.04 버전
- 키: k8s-mgr-key
- 보안그룹: k8s-mgr-sg

1 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

k8s-mgr-system-steam

추가 태그 추가

▼ 애플리케이션 및 OS 이미지(Amazon Machine Image) 정보

AMI는 인스턴스를 시작하는 데 필요한 소프트웨어 구성(운영 체제, 애플리케이션 서버 및 애플리케이션)이 포함된 템플릿입니다. 아래에서 찾고 있는 항목이 보이지 않으면 AMI를 검색하거나 찾아보세요.

Q 수천 개의 애플리케이션 및 OS 이미지를 포함하는 전체 카탈로그 검색

Quick Start



더 많은 AMI 찾아보기
AWS, Marketplace 및 커뮤니티의 AMI 포함

Amazon Machine Image(AMI)

Ubuntu Server 22.04 LTS (HVM), SSD Volume Type

ami-0077297a838d6761d (64비트(x86)) / ami-02b0575c5db06d053 (64비트(Arm))
가상화: hvm ENA 활성화됨: true 루트 디바이스 유형: ebs

프리 티어 사용 가능

2 키 페어 생성

키 페어 이름

키 페어를 사용하면 인스턴스에 안전하게 연결할 수 있습니다.

k8s-mgr-key-steam

이름에는 최대 255개의 ASCII 문자가 포함됩니다. 앞 또는 뒤에 공백을 포함할 수 없습니다.

키 페어 유형

☒ RSA

RSA 암호화된 프라이빗 및 퍼블릭 키 페어

☐ ED25519

ED25519 암호화된 프라이빗 및 퍼블릭 키 페어

프라이빗 키 파일 형식

☒ .pem

OpenSSH와 함께 사용

☐ .ppk

PuTTY와 함께 사용

⚠ 메시지가 표시되면 프라이빗 키를 사용자 컴퓨터의 안전하고 액세스 가능한 위치에 저장합니다. 나중에 인스턴스에 연결할 때 필요합니다. [자세히 알아보기](#)

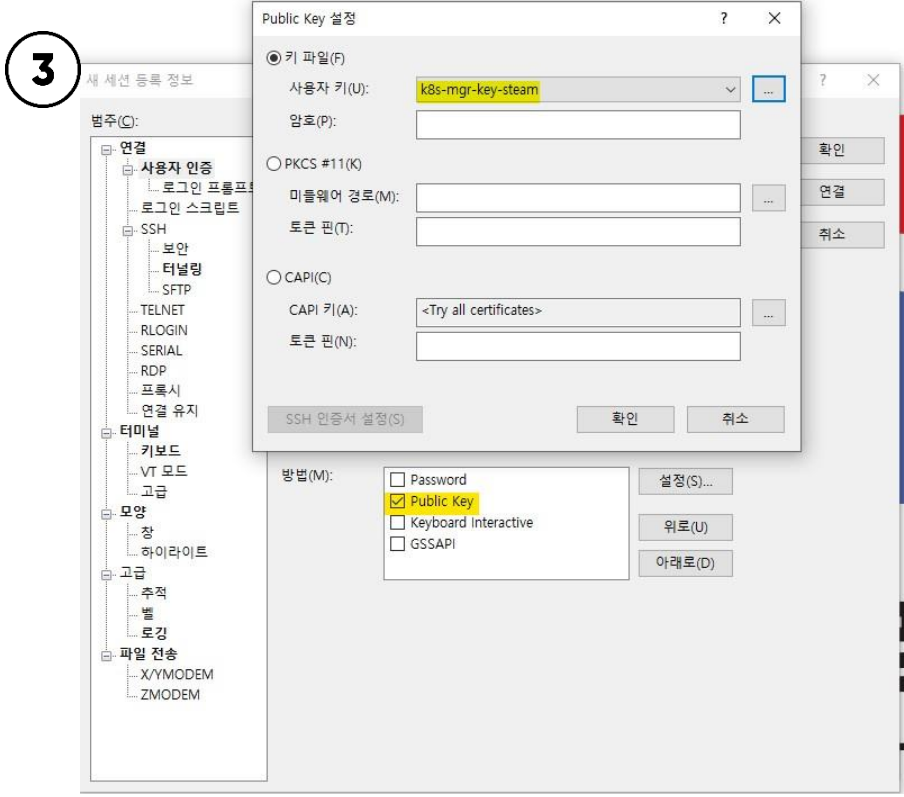
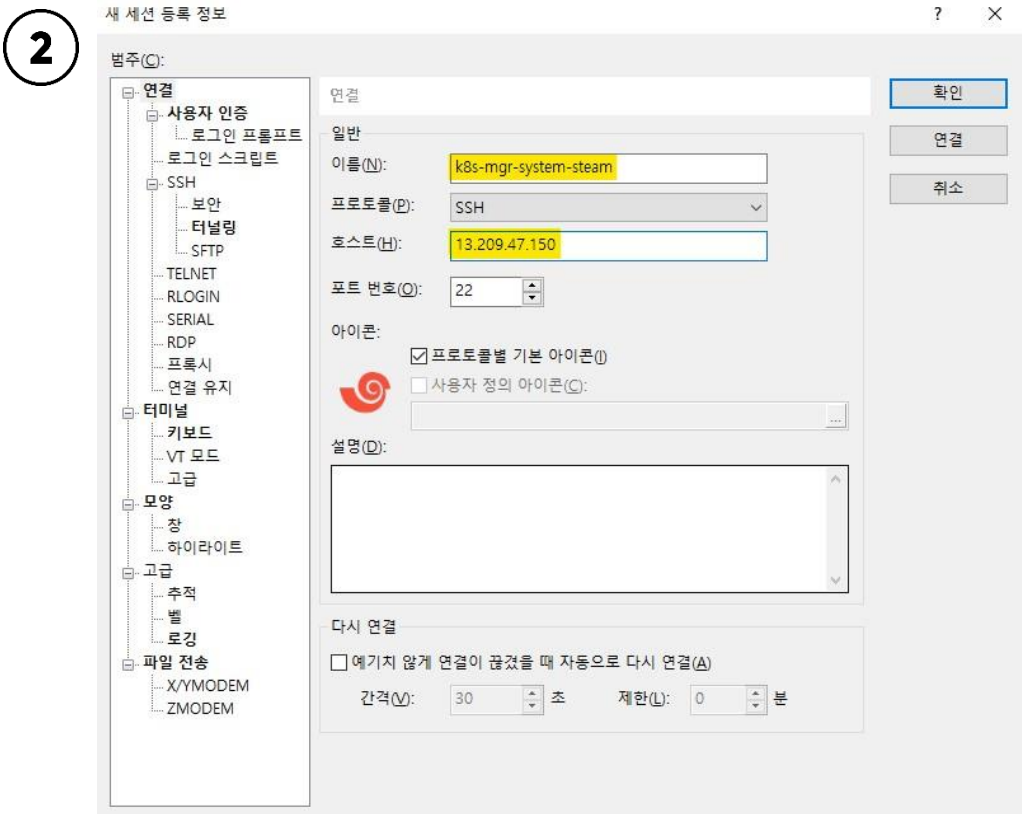
취소

키 페어 생성

3-2. 관리서버 구축

EC2 인스턴스 생성

- 이름: k8s-mgr-system
- AMI: ubuntu22.04 버전
- 키: k8s-mgr-key
- 보안그룹: k8s-mgr-sg

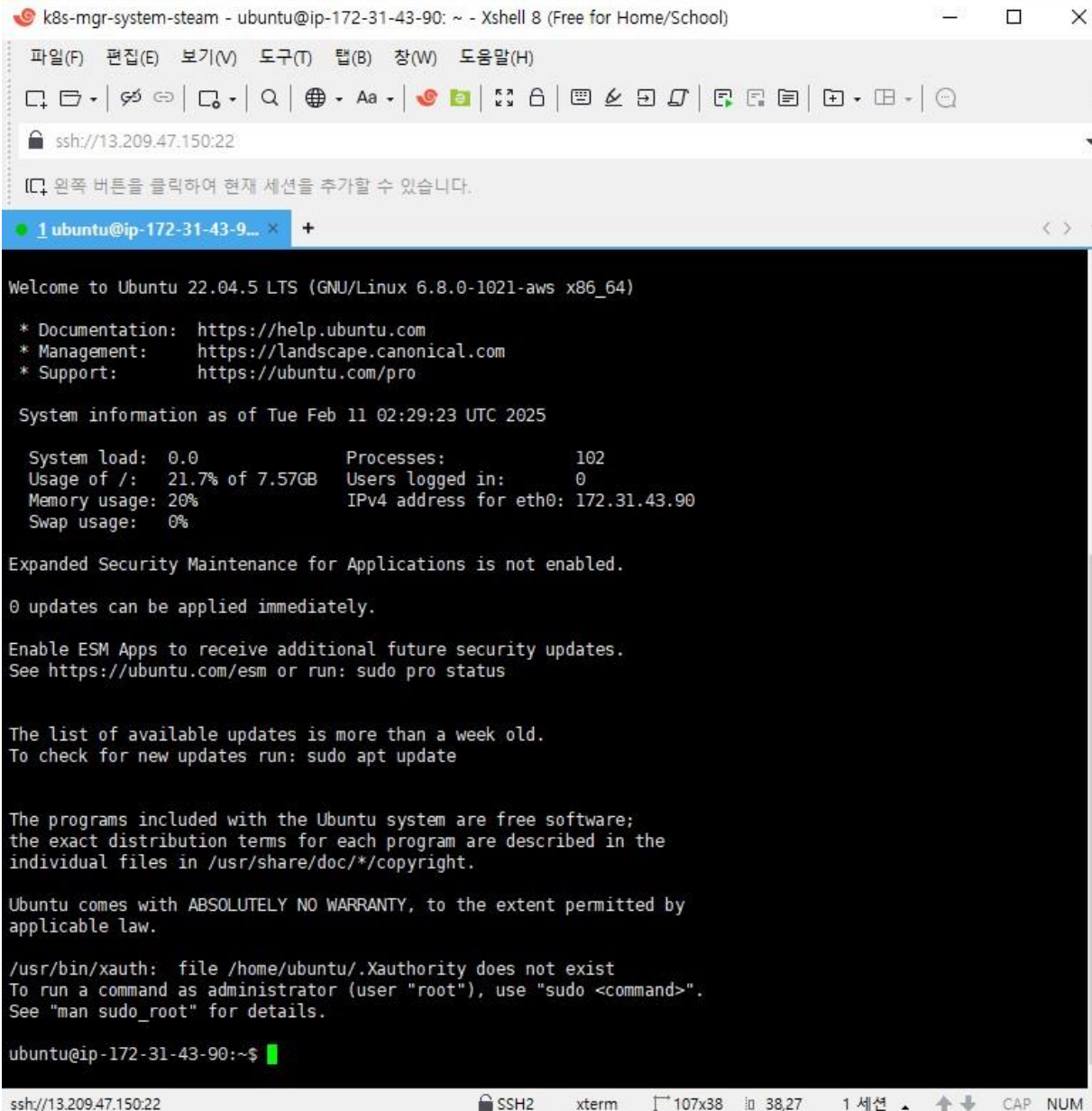


3-2. 관리서버 구축

EC2 인스턴스 생성

- 이름: k8s-mgr-system
- AMI: ubuntu22.04 버전
- 키: k8s-mgr-key
- 보안그룹: k8s-mgr-sg

1



```
k8s-mgr-system-steam - ubuntu@ip-172-31-43-90: ~ - Xshell 8 (Free for Home/School)
파일(F) 편집(E) 보기(V) 도구(T) 탭(B) 창(W) 도움말(H)
ssh://13.209.47.150:22
원쪽 버튼을 클릭하여 현재 세션을 추가할 수 있습니다.
1 ubuntu@ip-172-31-43-90:~$
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 6.8.0-1021-aws x86_64)

* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:        https://ubuntu.com/pro

System information as of Tue Feb 11 02:29:23 UTC 2025

System load:  0.0      Processes:            102
Usage of /:   21.7% of 7.57GB   Users logged in:      0
Memory usage: 20%      IPv4 address for eth0: 172.31.43.90
Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

/usr/bin/xauth: file /home/ubuntu/.Xauthority does not exist
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-43-90:~$
```

Xshell에 연결하여 계속 진행

3-3. Bastion Server 환경구성

관리시스템에 aws 계정 등록 ① 사용자 세부 정보 지정

- 1. AWS IAM 계정 생성
 - 사용자 이름: eks-mgr-user
 - 권한: 직접 정책 연결 - 두번째 administratorAccess
 - 보안자격증명 - 액세스키만들기 - CLI (체크박스체크)
 - .csv 파일 반드시 다운로드!!

사용자 세부 정보

사용자 이름

eks-mgr-steam

사용자 이름은 최대 64자까지 가능합니다. 유효한 문자: A~Z, a~z, 0~9 및 +, =, ., @, _ (하이픈)

☐ AWS Management Console에 대한 사용자 액세스 권한 제공 - 선택 사항

사람에게 콘솔 액세스 권한을 제공하는 경우 IAM Identity Center에서 액세스를 관리하는 것은 모범 사례입니다.

이 IAM 사용자를 생성한 후 액세스 키 또는 AWS CodeCommit이나 Amazon Keyspaces에 대한 서비스별 보안 인증 정보를 통해 프로그래밍 방식 액세스를 생성할 수 있습니다. 자세히 알아보기

취소

다음

② 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.

☐ 권한 복사

기존 사용자의 모든 그룹 멤버십, 연결된 관리형 정책 및 인라인 정책을 복사합니다.

☒ 직접 정책 연결

관리형 정책을 사용자에게 직접 연결합니다. 사용자에게 연결하는 대신, 정책을 그룹에 연결한 후 사용자를 적절한 그룹에 추가하는 것이 좋습니다.

권한 정책 (1/1322)

새 사용자에게 연결할 정책을 하나 이상 선택합니다.

검색

필터링 기준 유형

모든 유형

< 1 2 3 4 5 6 7 ... 67 >

정책 이름	유형	연결된 엔터티
<input type="checkbox"/> AccessAnalyzerServiceRolePolicy	AWS 관리형	0
<input checked="" type="checkbox"/> AdministratorAccess	AWS 관리형 - 직무	5
<input type="checkbox"/> AdministratorAccess-Amplify	AWS 관리형	0
<input type="checkbox"/> AdministratorAccess-AWSElasticBeanstalk	AWS 관리형	0

3-3. Bastion Server 환경구성

관리시스템에 aws 계정 등록

1. AWS IAM 계정 생성
 - 사용자 이름: eks-mgr-user
 - 권한: 직접 정책 연결 - 두번째 administratorAccess
 - 보안자격증명 - 액세스키만들기 - CLI (체크박스체크)
 - .csv 파일 반드시 다운로드!!

1

[홈](#)
>
[사용자](#)
>
[eks-mgr-steam](#)
>
[엑세스 키 만들기](#)

1단계

● 엑세스 키 모범 사례 및 대안

○ 2단계 - 선택 사항
설명 태그 설정

○ 3단계
엑세스 키 검색

엑세스 키 모범 사례 및 대안 정보

보안 개선을 위해 엑세스 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.

☐ AWS 랩퓨팅 서비스에서 실행되는 애플리케이션
Amazon EC2, Amazon ECS 또는 AWS Lambda와 같은 AWS 랩퓨팅 서비스에서 실행되는 애플리케이션 코드를 사용하여 AWS 계정에 액세스할 수 있도록 이 액세스 키를 사용할 것입니다.

☐ 서드 파티 서비스
AWS 리소스를 모니터링 또는 관리하는 서드 파티 애플리케이션 또는 서비스에 액세스할 수 있도록 이 액세스 키를 사용할 것입니다.

☐ AWS 외부에서 실행되는 애플리케이션
이 액세스 키를 사용하여 AWS 리소스에 액세스해야 하는 AWS 외부의 데이터 센터 또는 기타 인프라에서 실행 중인 워크로드를 인증할 것입니다.

☐ 기타
귀하의 사용 사례가 여기에 나열되어 있지 않습니다.

☒ **권장되는 대안**

- 브라우저 기반 CLI인 [AWS CloudShell](#)을 사용하여 명령을 실행합니다. [자세히 알아보기](#).
- [AWS CLI V2](#)를 사용하고 IAM 자격 증명 센터의 사용자를 통한 인증을 활성화합니다. [자세히 알아보기](#).

확인

☒ 위의 권장 사항을 이해했으며 액세스 키 생성을 계속하려고 합니다.

2

[홈](#)
>
[사용자](#)
>
[eks-mgr-steam](#)
>
[엑세스 키 만들기](#)

2단계

○ 1단계
엑세스 키 모범 사례 및 대안

● 2단계 - 선택 사항
설명 태그 설정

○ 3단계
엑세스 키 검색

엑세스 키 검색 정보

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

엑세스 키

비밀 액세스 키

AKIA5CBDQ5D34U4AANHR

***** 표시

엑세스 키 모범 사례

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

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

csv 파일 다운로드

완료

3-3. Bastion Server 환경구성

관리시스템에 aws 계정 등록

2. AWS CLI 설치

- `sudo apt update`
- `sudo apt install -y unzip`
- https://docs.aws.amazon.com/ko_kr/cli/latest/userguide/getting-started-install.html
- `curl`
"https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip"
`-o "awscliv2.zip"`
`unzip awscliv2.zip`
`sudo ./aws/install`

```
1 ubuntu@ip-172-31-43-90:~$ sudo apt-get install -y unzip
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Suggested packages:
  zip
The following NEW packages will be installed:
  unzip
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 175 kB of archives.
After this operation, 386 kB of additional disk space will be used.
Get:1 http://ap-northeast-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 unzip amd64 6.0-26ubuntu
3.2 [175 kB]
Fetched 175 kB in 0s (10.6 MB/s)
Selecting previously unselected package unzip.
```

```
3 ubuntu@ip-172-31-43-90:~$ sudo ./aws/install
You can now run: /usr/local/bin/aws --version
```

2

2. AWS CLI 설치

AWS CLI를 설치하는 방법

시작하기
서비스 가이드
개발자 도구
AI 리소스

nd Line <

설치

slc/Docker

인증

:

다음 방법 중 하나를 사용하여 AWS CLI를 설치할 수 있습니다.

- 설치할 버전을 지정할 수 있으므로 버전 제어에는 명령줄 설치 관리자 프로그램이 적합합니다. 이 옵션은 자동 업데이트되지 않으며 업데이트할 때마다 새 설치 관리자를 다운로드하여 이전 버전을 덮어써야 합니다.
- 공식적으로 지원되는 **snap** 패키지는 스냅 패키지가 자동으로 새로 고쳐지므로 항상 최신 버전의 AWS CLI를 사용할 수 있는 좋은 옵션입니다. 마이너 버전의 AWS CLI를 선택하는 기능이 기본적으로 지원되지 않으므로 팀에서 버전을 고정해야 하는 경우 최적의 설치 방법이 아닙니다.

Command line installer - Linux x86 (64-bit)

Command line - Linux ARM

Snap package

AWS CLI의 현재 설치를 업데이트하려면 업데이트할 때마다 새 설치 관리자를 다운로드하여 이전 버전을 덮어씁니다. 명령줄에서 다음 단계에 따라 Linux에 AWS CLI를 설치합니다.

다음은 기본 설치를 제공하는 단일 복사 및 붙여넣기 그룹의 빠른 설치 단계입니다. 안내 지침은 다음 단계를 참조하세요.

① 참고

(선택 사항) 다음 명령 블록은 다운로드의 무결성을 먼저 확인하지 않고 AWS CLI를 다운로드하고 설치합니다. 다운로드 무결성을 확인하려면 아래의 단계별 지침을 사용하세요.

AWS CLI를 설치하려면 다음 명령을 실행합니다.

```
$ curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
unzip awscliv2.zip
sudo ./aws/install
```

3-3. Bastion Server 환경구성

관리시스템에 aws 계정 등록

3. 관리 시스템에 aws 계정 등록

- aws configure
- 받은 키들과 리전 입력 리전은: ap-northeast-2, format은 json
- 확인용: aws sts get-caller-identity

4. k8s 관리툴인 kubectl 설치(최신버전)

- kubectl 설치 경로: https://docs.aws.amazon.com/ko_kr/eks/latest/userguide/install-kubectl.html
- 복사
- 2~5번 진행
- 확인: kubectl version

①

```
ubuntu@ip-172-31-43-90:~$ aws configure
AWS Access Key ID [None]: AKIA5CBDQ5D34U4AANHR
AWS Secret Access Key [None]: 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
Default region name [None]: ap-northeast-2
Default output format [None]: json
```

②

```
ubuntu@ip-172-31-43-90:~$ aws sts get-caller-identity
{
  "UserId": "AIDA5CBDQ5D3Z6WMK3TXA",
  "Account": "897722673399",
  "Arn": "arn:aws:iam::897722673399:user/eks-mgr-steam"
}
```

- ③ **Linux(AMD64)**
1. Amazon S3에서 클러스터의 Kubernetes 버전에 대한 kubectl 바이너리를 다운로드합니다.
- Kubernetes 1.32

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.32.0/2024-12-20/bin/linux/amd64/kubectl
```

④

```
ubuntu@ip-172-31-43-90:~$ curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.32.0/2024-12-20/bin/linux/amd64/kubectl
% Total    % Received % Xferd Average Speed   Time    Time     Time  Current
           % Dload  % Upload   Total   Spent    Left  Speed
100 54.6M  100 54.6M    0     0  5333k      0  0:00:10  0:00:10 --:--:-- 6156k
```

3-3. Bastion Server 환경구성

관리시스템에 aws 계정 등록

4. k8s 관리툴인 kubectl 설치(최신버전)

- kubectl 설치 경로:
https://docs.aws.amazon.com/ko_kr/eks/latest/userguide/install-kubectl.html
- 복사
- 2~5번 진행
- 확인: kubectl version

1. 2. (선택사항) 해당 바이너리의 SHA-256 체크섬을 사용하여 다운로드한 바이너리를 확인합니다.
 - a. 디바이스의 하드웨어 플랫폼용 명령을 사용하여 Amazon S3using에서 클러스터의 Kubernetes 버전에 대한 SHA-256 체크섬을 다운로드합니다.
 - Kubernetes 1.32

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.32.0/2024-12-20/bin/linux/amd64/kubectl.sha256
```
 2. b. 다음 명령 중 하나를 사용하여 다운로드한 바이너리의 SHA-256 체크섬을 확인합니다.

```
sha256sum -c kubectl.sha256
```

or

```
openssl sha1 -sha256 kubectl
```
3.

```
ubuntu@ip-172-31-43-90:~$ sha256sum -c kubectl.sha256
kubectl: OK
ubuntu@ip-172-31-43-90:~$ openssl sha1 -sha256 kubectl
SHA2-256(kubectl)= 3d46610aleba3dd47e4879f69effa481f56517d7a4a57bb3ebf7498f4df2e99e
```


3-3. Bastion Server 환경구성

관리시스템에 aws 계정 등록

4. k8s 관리툴인 kubectl 설치(최신버전)

- kubectl 설치 경로:
https://docs.aws.amazon.com/ko_kr/eks/latest/userguide/install-kubectl.html
- 복사
- 2~5번 진행
- 확인: kubectl version

① 3. 바이너리에 실행 권한을 적용합니다.

```
chmod +x ./kubectl
```

4. 바이너리를 PATH의 폴더에 복사합니다. kubectl 버전이 이미 설치된 경우 \$HOME/bin/kubectl을 생성하고 \$HOME/bin이 \$PATH로 시작하도록 해야 합니다.

```
mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$HOME/bin:$PATH
```

5. (선택사항) 셸 초기화 파일에 \$HOME/bin 경로를 추가하면 셸을 열 때 구성됩니다.

참고

이 단계에는 Bash 셸을 사용한다고 가정합니다. 다른 셸을 사용하는 경우, 특정 셸 초기화 파일을 사용하도록 명령을 변경하세요.

```
echo 'export PATH=$HOME/bin:$PATH' >> ~/.bashrc
```

②

```
ubuntu@ip-172-31-43-90:~$ chmod +x ./kubectl
ubuntu@ip-172-31-43-90:~$ mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$HOME/bin:$PATH
ubuntu@ip-172-31-43-90:~$ echo 'export PATH=$HOME/bin:$PATH' >> ~/.bashrc
ubuntu@ip-172-31-43-90:~$ kubectl version
Client Version: v1.32.0-eks-5ca49cb
Kustomize Version: v5.5.0
```


3-3. Bastion Server 환경구성

관리시스템에 aws 계정 등록

5. eksctl 설치

- 아래로 내려서 설치 링크 타고 들어가기
- 한 줄씩 차례대로 입력
- 확인: eksctl version

1



7. 처음 몇 글자를 입력한 후 Tab 키를 사용하여 kubect1 하위 명령을 완료할 수 있는 자동 완성 구성을 고려합니다. 자세한 내용은 Kubernetes 설명서의 [Kubect1 autocomplete](#)를 참조하세요.

eksctl 설치

eksctl CLI는 EKS 클러스터 작업에 사용됩니다. 이는 많은 개별 작업을 자동화합니다. eksctl 설치에 대한 지침은 eksctl 설명서의 [설치](#)를 참조하세요.

eksctl을 사용하는 경우 사용 중인 IAM 보안 주체에 Amazon EKS IAM 역할, 서비스 연결 역할, AWS CloudFormation, VPC, 관련 리소스를 사용할 수 있는 권한이 있어야 합니다. 자세한 내용은 IAM 사용 설명서의 [서비스 연결 역할 사용](#)을 참조하세요. 이 가이드의 모든 단계를 동일한 사용자로 완료해야 합니다. 현재 사용자를 확인하려면 다음 명령을 실행합니다.

2

For Unix

To download the latest release, run:

```
# for ARM systems, set ARCH to: `arm64`, `armv6` or `armv7`  
ARCH=amd64  
PLATFORM=$(uname -s)_$ARCH  
  
curl -sLO "https://github.com/eksctl-io/eksctl/releases/latest/download/eksctl_${P  
PLATFORM.tar.gz"  
  
# (Optional) Verify checksum  
curl -sL "https://github.com/eksctl-io/eksctl/releases/latest/download/eksctl_checl  
PLATFORM.tar.gz.sha256sum"  
  
tar -xzf eksctl_${PLATFORM}.tar.gz -C /tmp && rm eksctl_${PLATFORM}.tar.gz  
  
sudo mv /tmp/eksctl /usr/local/bin
```

3

```
ubuntu@ip-172-31-43-90:~$ ARCH=amd64  
ubuntu@ip-172-31-43-90:~$ PLATFORM=$(uname -s)_$ARCH  
ubuntu@ip-172-31-43-90:~$ curl -sLO "https://github.com/eksctl-io/eksctl/releases/latest/download/eksctl_${P  
PLATFORM.tar.gz"
```

4

```
ubuntu@ip-172-31-43-90:~$ curl -sL "https://github.com/eksctl-io/eksctl/releases/latest/download/eksctl_checksums.txt" | grep  
Usage: grep [OPTION]... PATTERNS [FILE]...  
Try 'grep --help' for more information.  
ubuntu@ip-172-31-43-90:~$ $PLATFORM | sha256sum --check  
Linux amd64: command not found  
sha256sum: 'standard input': no properly formatted SHA256 checksum lines found  
ubuntu@ip-172-31-43-90:~$ tar -xzf eksctl_${PLATFORM}.tar.gz -C /tmp && rm eksctl_${PLATFORM}.tar.gz  
ubuntu@ip-172-31-43-90:~$ sudo mv /tmp/eksctl /usr/local/bin  
ubuntu@ip-172-31-43-90:~$ eksctl version  
0.203.0
```

3-4. EKS 클러스터 생성

클러스터 생성

```
eksctl create cluster ₩  
--name k8s-demo ₩  
--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
```

- cloud formation, ec2인스턴스 에서 생성중인걸 확인
- EKS - 컴퓨팅 - 노드그룹 생성확인, VPC 생성 확인
- kubectl get no / kubectl get po -A

1

ubuntu@ip-172-31-43-90:~\$ eksctl create cluster \
--name k8s-steam \
--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

인스턴스 (3) 정보

Name	인스턴스 ID	인스턴스 상태	인스턴스 유형	상태 검사	정보 상태	가용 영역	피블릭 IPv4 DNS	피블릭 IPv4 ...	탄력적
k8s-mgr-system-steam	i-01b3b4e525d98e8c9	실행 중	t2.micro	2/2개 검사 통과...	경보 보기	ap-northeast-2c	ec2-13-209-47-150.ap-...	13.209.47.150	-
k8s-steam-k8s-ng-Node	i-01e30149cc0ef14cf	실행 중	m5.large	3/3개 검사 통과...	경보 보기	ap-northeast-2c	ec2-15-164-171-163.ap-...	15.164.171.163	-
k8s-steam-k8s-ng-Node	i-05f7c9425969f77c9	실행 중	m5.large	3/3개 검사 통과...	경보 보기	ap-northeast-2a	ec2-43-203-219-30.ap-...	43.203.219.30	-

3

스택 (3)

스택 이름	스택 ID	상태	생성 시간	업데이트한 시간	삭제한 시간
eksctl-k8s-steam-nodegroup-k8s-ng	arn:aws:cloudformation:ap-no...	CREATE_COMPLETE	2025-02-11 13:34:59 UTC+0900	-	-
eksctl-k8s-steam-addon-vpc-cni	arn:aws:cloudformation:ap-no...	CREATE_COMPLETE	2025-02-11 13:34:18 UTC+0900	-	-
eksctl-k8s-steam-cluster	arn:aws:cloudformation:ap-no...	CREATE_COMPLETE	2025-02-11 13:24:12 UTC+0900	-	-

4

클러스터 (1) 정보

클러스터 이름	상태	Kubernetes 버전	지원 기간	정책 업그레이드	생성됨	공급자
k8s-steam	활성	1.30 지금 업그레이드	2025년 7월 23일까지 표준 지원	확장됨	17분 전	EKS

5

ubuntu@ip-172-31-43-90:~\$ kubectl get no
NAME
ip-192-168-15-114.ap-northeast-2.compute.internal
ip-192-168-60-246.ap-northeast-2.compute.internal
STATUS
Ready
Ready
ROLES
<none>
<none>
AGE
6m53s
6m55s
VERSION
v1.30.8-eks-aeac579
v1.30.8-eks-aeac579

3-4. EKS 클러스터 생성

클러스터 생성

- kubectl create deployment webtest --image=nginx:1.14 --port=80 --replicas=3
- kubectl get deploy,po
- kubectl expose deployment webtest --port=80 --type=LoadBalancer
- kubectl get svc / 접속되는지 확인
- kubectl delete svc webtest
- kubectl delete deploy webtest

①

```
ubuntu@ip-172-31-43-90:~$ kubectl create deployment webtest --image=nginx:1.14 --port=80 --replicas=3
deployment.apps/webtest created
ubuntu@ip-172-31-43-90:~$ kubectl get deploy,po
NAME                                READY    UP-TO-DATE    AVAILABLE    AGE
deployment.apps/webtest             3/3      3              3             3m4s

NAME                                READY    STATUS    RESTARTS    AGE
pod/webtest-6d754887d7-26thr        1/1      Running   0            3m4s
pod/webtest-6d754887d7-2jtxc        1/1      Running   0            3m4s
pod/webtest-6d754887d7-9mbbx        1/1      Running   0            3m4s
ubuntu@ip-172-31-43-90:~$ kubectl expose deployment webtest --port=80 --type=LoadBalancer
service/webtest exposed
ubuntu@ip-172-31-43-90:~$ kubectl get svc
NAME                                TYPE                CLUSTER-IP    EXTERNAL-IP
kubernetes                          ClusterIP           10.100.0.1     <none>
webtest                             LoadBalancer        10.100.158.144 a96107ceec9aa432e9e1f5b5982886bc-1805272758.ap-northeast-2.elb.amazonaws.com
```

②

```
ubuntu@ip-172-31-43-90:~$ kubectl delete svc webtest
service "webtest" deleted
ubuntu@ip-172-31-43-90:~$ kubectl delete deploy webtest
deployment.apps "webtest" deleted
```


3-5. 로드밸런서 컨트롤러 생성

Helm을 활용하여 Load Balancer Controller 생성

1단계: eksctl을 사용하여 IAM 역할 생성

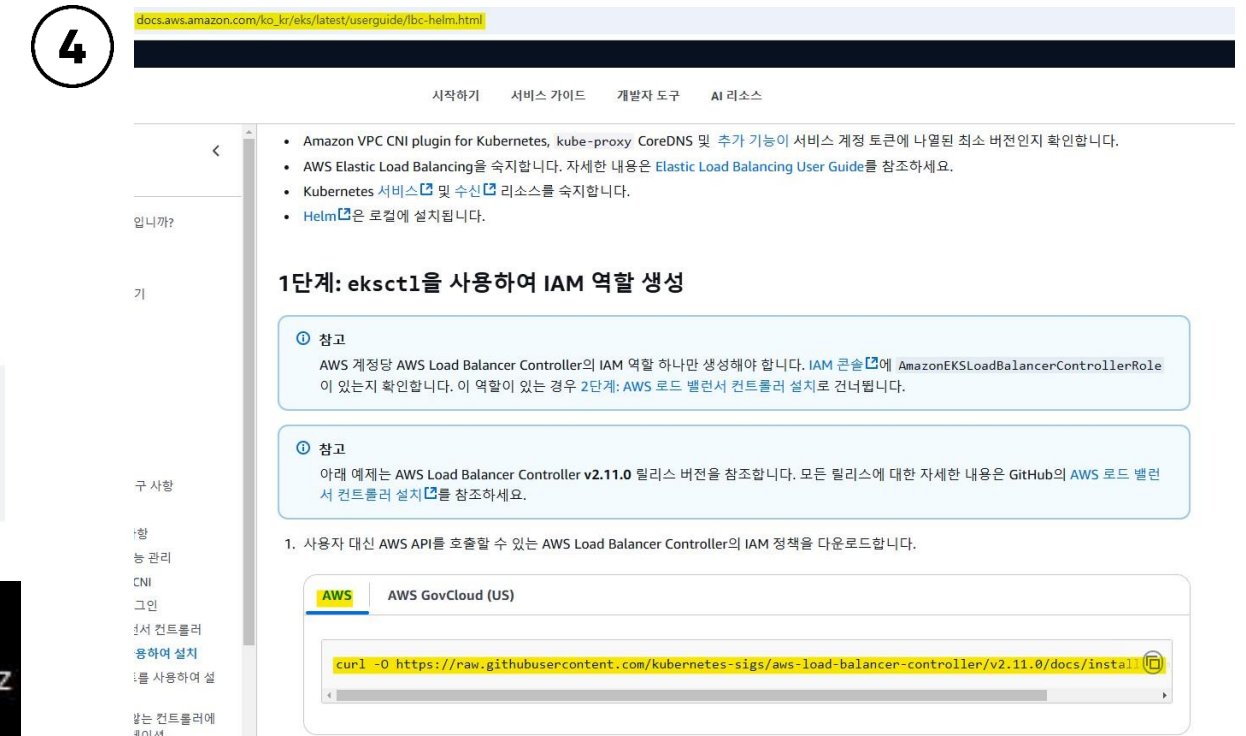
- 헬름설치 검색
- 스크립트로 3줄
- https://docs.aws.amazon.com/ko_kr/eks/latest/userguide/lbc-helm.html (1, 2번)



1 Helm
https://helm.sh/docs> intro > install
[헬름 설치하기]
헬름 설치하기 이 가이드는 헬름 CLI를 설치하는 방법을 설명한다. 헬름은 소스 또는 미리-빌드된(pre-built) 바이너리 릴리스로 설치할 수 있다.

2
\$ curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3
\$ chmod 700 get_helm.sh
\$./get_helm.sh

3
ubuntu@ip-172-31-43-90:~\$ chmod 700 get_helm.sh
ubuntu@ip-172-31-43-90:~\$./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



5 2. 이전 단계에서 다운로드한 정책을 사용하여 IAM 정책을 만듭니다.

```
aws iam create-policy \
  --policy-name AWSLoadBalancerControllerIAMPolicy \
  --policy-document file://iam_policy.json
```

3-5. 로드밸런서 컨트롤러 생성

Helm을 활용하여 Load Balancer Controller 생성

1단계: eksctl을 사용하여 IAM 역할 생성

- 확인: AWS IAM
- export cluster_name=k8s-demo
- oidc_id=\$(aws eks describe-cluster --name \$cluster_name --query "cluster.identity.oidc.issuer" --output text | cut -d '/' -f 5)
- echo \$oidc_id (출력 안되면 eksctl utils associate-iam-oidc-provider --cluster \$cluster_name --approve)
- IAM - 정책 - ARN복사 -> arn부분을 바꾸기

① 정책 (1/1322) 정보

정책은 권한을 정의하는 AWS의 객체입니다.

정책 이름

▲

유형

+

AWSLoadBalancerControllerIAMPolicy

고객 관리형

②

```
ubuntu@ip-172-31-43-90:~$ export cluster_name=k8s-steam
oidc_id=$(aws eks describe-cluster --name $cluster_name --query "cluster.identity.oidc.issuer" --output text | cut -d '/' -f 5)
echo $oidc_id
8306D0DD72D2FBCDFF61846508643C53
```

③

AWSLoadBalancerControllerIAMPolicy 정보

편집삭제

정책 세부 정보

ARN이(가) 복사됨

arn:aws:iam::050752633979:policy/AWSLoadBalancerControllerIAMPolicy

정책 세부 정보

유형

고객 관리형

생성 시간

February 10, 2025, 14:20 (UTC+09:00)

편집 시간

February 10, 2025, 14:20 (UTC+09:00)

권한

연결된 엔터티

태그

정책 버전

마지막 액세스

3-5. 로드밸런서 컨트롤러 생성

Helm을 활용하여 Load Balancer Controller 생성

2단계: AWS Load Balancer Controller 설치

- 1번, 2번 수행
- 3번 (클러스터이름만 수정 k8s-demo)
- helm install aws-load-balancer-controller eks/aws-load-balancer-controller -n kube-system --set clusterName=k8s-demo --set serviceAccount.create=false --set serviceAccount.name=aws-load-balancer-controller
- 확인: kubectl get deployment -n kube-system aws-load-balancer-controller

① 2단계: AWS Load Balancer Controller 설치

1. eks-charts 차트 Helm 리포지토리를 추가합니다. AWS에서는 이 리포지토리를 GitHub에 유지합니다.

```
helm repo add eks https://aws.github.io/eks-charts
```

2. 최신 차트가 적용되도록 로컬 리포지토리를 업데이트합니다.

```
helm repo update eks
```

3. AWS Load Balancer Controller를 설치합니다.

Amazon EC2 인스턴스 메타데이터 서비스(IMDS)에 대해 제한적인 액세스 권한이 있는 Amazon EC2 노드에 컨트롤러를 배포하거나 Fargate 또는 Amazon EKS Hybrid Nodes에 배포하는 경우, 다음 helm 명령에 다음 플러그를 추가합니다.

- --set region=region-code
- --set vpcId=vpc-xxxxxxxx

my-cluster를 해당 클러스터의 이름으로 바꿉니다. 다음 명령에서 aws-load-balancer-controller는 이전 단계에서 생성한 Kubernetes 서비스 계정입니다.

차트 Helm 구성에 관한 자세한 내용은 GitHub에서 values.yaml을 참조하세요.

```
helm install aws-load-balancer-controller eks/aws-load-balancer-controller \
  -n kube-system \
  --set clusterName=my-cluster \
  --set serviceAccount.create=false \
  --set serviceAccount.name=aws-load-balancer-controller
```

```
ubuntu@ip-172-31-43-90:~$ helm repo add eks https://aws.github.io/eks-charts
"eks" has been added to your repositories
ubuntu@ip-172-31-43-90:~$ 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-172-31-43-90:~$ 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          57s
```


3-6. 로드밸런서 배포 - NLB, ALB

1. NLB

- https://docs.aws.amazon.com/ko_kr/eks/latest/userguide/network-load-balancing.html
- 샘플 애플리케이션을 배포합니다.
 - `kubectl get ns nlb-sample-app`
 - 2번 a, b, c 쪽 진행 (확인: `kubectl get po -n nlb-sample-app`)
 - 3번 진행
 - 4번 진행
- EC2 - 로드 밸런서 생성 확인 (DNS를 복사해 external - IP 에 붙여넣기)
- 로드밸런서 - 대상그룹

```
1 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
```

```
2 ubuntu@ip-172-31-43-90:~$ kubectl apply -f sample-deployment.yaml
deployment.apps/nlb-sample-app created
```

```
3 ubuntu@ip-172-31-43-90:~$ kubectl get po -n nlb-sample-app
NAME                                READY   STATUS    RESTARTS   AGE
nlb-sample-app-fccbb75cd-67sfl      1/1     Running   0           2m30s
nlb-sample-app-fccbb75cd-ds2td      1/1     Running   0           2m30s
nlb-sample-app-fccbb75cd-gz9p5      1/1     Running   0           2m30s
```

4

3. IP 대상에 대한 로드 밸런싱을 수행하는 인터넷이 연결된 Network Load Balancer를 사용하여 서비스를 생성합니다.
- a. 다음 내용을 컴퓨터에 `sample-service.yaml` 파일이라는 이름의 파일에 저장합니다. Fargate 노드에 배포하는 경우 `service.beta.kubernetes.io/aws-load-balancer-scheme: internet-facing` 줄을 제거합니다.

```
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
```

- b. 매니페스트를 클러스터에 적용합니다.

```
kubectl apply -f sample-service.yaml
```

3-6. 로드밸런서 배포 - NLB, ALB

1. NLB

- https://docs.aws.amazon.com/ko_kr/eks/latest/userguide/network-load-balancing.html
- 샘플 애플리케이션을 배포합니다.
 - `kubectl get ns nlb-sample-app`
 - 2번 a, b, c 쪽 진행 (확인: `kubectl get po -n nlb-sample-app`)
 - 3번 진행
 - 4번 진행
- EC2 - 로드 밸런서 생성 확인 (DNS를 복사해 external - IP 에 붙여넣기)
- 로드밸런서 - 대상그룹

①

```
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-172-31-43-90:~$ kubectl apply -f sample-service.yaml
service/nlb-sample-service created
```

③ 로드 밸런서 (1)

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

로드 밸런서 필터링

이름	DNS 이름	상태	VPC ID	가용 영역	유형	생성된 날짜
k8s-nlbsampl-nlbsampl...	k8s-nlbsampl-nlbsampl-f99df529e-c5fdda4a9b...	활성	vpc-0bf4d5fda9c7be	2 가용 영역	network	2025년 2월 11일, 15:49 (UTC+09:00)

④ k8s-nlbsampl-nlbsampl-f7aee5047c

세부 정보

aws:elasticloadbalancing:ap-northeast-2:897722673399:targetgroup/k8s-nlbsampl-nlbsampl-f7aee5047c/j2f834570e708cc

대상 유형: IP

프로토콜: 포트: TCP: 80

VPC: vpc-0bf4d5fda9c7be

IP 주소 유형: IPv4

로드 밸런서: k8s-nlbsampl-nlbsampl-f99df529e

대상 개	정상	비정상	사용되지 않음	초기	드레이닝
3	3	0	0	0	0

가용 영역별 대상 배포

이래의 등록된 대상 테이블에 적용된 해당 필드를 보려면 이 테이블에서 값을 선택합니다.

대상 | 모니터링 | 상태 검사 | 속성 | 태그

등록된 대상 (3)

IP 주소	포트	영역	상태 확인	상태 확인 세부 정보	관리상 재정의	재정의 세부 정보
192.168.17.207	80	ap-northeast-2a (apne2-a...	Healthy	-	No override	No override is currently a...
192.168.32.32	80	ap-northeast-2c (apne2-a...	Healthy	-	No override	No override is currently a...
192.168.14.17	80	ap-northeast-2a (apne2-a...	Healthy	-	No override	No override is currently a...

3-6. 로드밸런서 배포 - NLB, ALB

1. ALB

- 2048_full aws

1

aws

Amazon EKS

기 전에 모든 대상의 상태가 정상일 때까지 기다립니다. 모든 대상이 healthy가 될 때까지 몇 분 정도 걸릴 수 있습니다. 대상은 healthy 상태로 변경하기 전에 unhealthy 상태일 수 있습니다.

7. `curl k8s-default-samplese-xxxxxxxxxxxx-xxxxxxxxxxxxx.elb.region-code.amazonaws.com`

예제 출력은 다음과 같습니다.

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
[...]
```

8. 샘플 배포, 서비스 및 테임스페이스가 완료되면 제거합니다.

`kubectl delete namespace nlb-sample-app`

다음 주제: [애플리케이션 로드 밸런싱](#)
이전 주제: [Horizontal Pod Autoscaler](#)

2. 게임 2048을 샘플 애플리케이션으로 배포하여 AWS Load Balancer Controller가 수신 객체의 결과로 AWS ALB를 생성하는지 확인합니다. 배포하려는 서버넷 유형에 대한 단계를 완료합니다.
- a. IPv6 패밀리로 생성한 클러스터의 Pods에 배포하는 경우 다음 단계로 건너뛴다.
- 퍼블릭::

2

```
ubuntu@ip-172-31-43-90:~$ 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
```

3

```
ubuntu@ip-172-31-43-90:~$ kubectl get pods -n game-2048
NAME                                READY   STATUS    RESTARTS   AGE
deployment-2048-85f8c7d69-d5jmh    1/1     Running   0           58s
deployment-2048-85f8c7d69-dq9cj    1/1     Running   0           58s
deployment-2048-85f8c7d69-jltgj    1/1     Running   0           58s
deployment-2048-85f8c7d69-mtfhh    1/1     Running   0           58s
deployment-2048-85f8c7d69-rzp9b    1/1     Running   0           58s
ubuntu@ip-172-31-43-90:~$ kubectl get svc -n game-2048
NAME      TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)          AGE
service-2048  NodePort    10.100.155.76 <none>        80:31943/TCP     68s
ubuntu@ip-172-31-43-90:~$ kubectl get ing ingress-2048 -n game-2048
NAME      CLASS  HOSTS      ADDRESS                                                                 PORTS   AGE
ingress-2048  alb    *          k8s-game2048-ingress2-56fd5b4c55-2035839758.ap-northeast-2.elb.amazonaws.com  80      81s
```

4

k8s-game2048-ingress2-56fd5b4c55

세부 정보

로드 밸런서 유형
애플리케이션

상태
프로비저닝 중

가용 영역
subnet-01fc0cb03da69dfba ap-northeast-2c (apne2-az3)
subnet-017428d51470ee326 ap-northeast-2a (apne2-az1)

로드 밸런서 IP 주소 유형
IPv4
생성된 날짜
2025년 2월 11일, 15:56 (UTC+09:00)

로드 밸런서 ARN
arn:aws:elasticloadbalancing:ap-northeast-2:897722673399:loadbalancer/app/k8s-game2048-ingress2-56fd5b4c55/0d0eaa8c82efe465

DNS 이름 정보
k8s-game2048-ingress2-56fd5b4c55-2035839758.ap-northeast-2.elb.amazonaws.com (A 레코드)

리스너 및 규칙

리스너는 구성된 프로토콜 및 포트에서 연결 요청을 확인합니다. 리스너가 수신한 트래픽은 기본 작업 및 기타 추가 규칙에 따라 라우팅됩니다.

리스너 필터링

프로토콜: 포트	기본 작업	규칙	ARN	보안 정책	기본 SSL/TLS 인증서	mTLS	트러스트
HTTP:80	고정 응답 반환 <ul style="list-style-type: none">응답 코드: 404응답 본문응답 콘텐츠 유형: text/plain	2개 규칙	ARN	해당되지 않음	해당되지 않음	해당되지 않음	해당되지 않음

감사합니다.