



AWS EKS Project

REST | 강승환 고동우 유세종 최성민 한시완

목차

1. Bastion Server 생성 (EC2)

2. Bastion Server 환경 구성

3. EKS Cluster 생성

4. Load Balancer Controller 생성

5. Load Balancer 배포 - NLB

6. Load Balancer 배포 - ALB

1. Bastion Server 생성 (EC2)

1. Bastion Server 생성 (EC2)

1-1. 인스턴스 생성(1)

인스턴스

- 이름 : rest-bastion
- AMI : Ubuntu Server 24.04 LTS
- 인스턴스 유형 : t3.medium

키 페어

- 이름 : rest-key
- 키 페어 유형 : RSA
- 파일 형식 : .pem

인스턴스 시작

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

이름 및 태그

이름

rest-bastion

추가 태그 추가

▼ 애플리케이션 및 OS 이미지(Amazon Machine Image)

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 field or choose [Browse more AMIs](#).

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

더 많은 AMI 찾아보기

AWS, Marketplace 및 커뮤니티의 AMI 포함

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

▼ 인스턴스 유형

정보 | 조언 받기

인스턴스 유형

t3.medium

패밀라: t3 2 vCPU 4 GiB 메모리 현재 세대: true

온디맨드 Windows 기본 요금: 0.0704 USD 시간당

온디맨드 SUSE 기본 요금: 0.1083 USD 시간당 온디맨드 RHEL 기본 요금: 0.0808 USD 시간당

온디맨드 Ubuntu Pro 기본 요금: 0.0555 USD 시간당

온디맨드 Linux 기본 요금: 0.052 USD 시간당

☒ 모든 세대

인스턴스 유형 비교

소프트웨어가 사전 설치된 AMI에는 추가 비용이 적용됩니다.

▼ 키 페어(로그인)

정보

키 페어를 사용하여 인스턴스에 안전하게 연결할 수 있습니다. 인스턴스를 시작하기 전에 선택한 키 페어에 대한 액세스 권한이 있는지 확인하세요.

키 페어 이름 - 필수

rest-key

새 키 페어 생성

키 페어 생성

키 페어 이름

키 페어를 사용하면 인스턴스에 안전하게 연결할 수 있습니다.

rest-key

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

1. Bastion Server 생성 (EC2)

1-2. 인스턴스 생성(2)

네트워크 설정

- VPC : 기본값
- 서브넷 : ap-northeast-2a
- 보안 그룹 생성
 - 이름 : rest-bastion-sg-1
 - 유형 : ssh (위치 무관)

키 페어

- 이름 : rest-key
- 키 페어 유형 : RSA
- 파일 형식 : .pem

▼ 네트워크 설정 정보

VPC - 필수 | 정보

vpc-01b4b5c247441d61b (default-vpc)
172.31.0.0/16

↻

서브넷 | 정보

subnet-00320d339c69bc012
VPC: vpc-01b4b5c247441d61b 소유자: 710271940035
가용 영역: ap-northeast-2a (apne2-az1) 영역 유형: 가용 영역 사용 가능한 IP 주소: 4091
CIDR: 172.31.0.0/20

default-subnet-1a
↻ 새 서브넷 생성

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 추가합니다.

☒ 보안 그룹 생성 ☐ 기존 보안 그룹 선택

보안 그룹 이름 - 필수

rest-bastion-sg-1

이 보안 그룹은 모든 네트워크 인터페이스에 추가됩니다. 보안 그룹을 만든 후에는 이름을 편집할 수 없습니다. 최대 길이는 255자입니다. 유효한 문자는 a~z, A~Z, 0~9, 공백 및 _-./!@#%&*~()\$*입니다.

설명 - 필수 | 정보

rest-bastion-sg

인바운드 보안 그룹 규칙

▼ 보안 그룹 규칙 1 (TCP, 22, 0.0.0.0/0)

제거

유형 | 정보

ssh

프로토콜 | 정보

TCP

포트 범위 | 정보

22

소스 유형 | 정보

위치 무관

원본 | 정보

Q CIDR, 접두사 목록 또는 보안 그룹 추가

설명 - 선택 사항 | 정보

예: 관리자 데스크톱용 SSH

1. Bastion Server 생성 (EC2)

1-3. 탄력적 IP 연결

탄력적 IP 생성

- 만든 인스턴스로 연결
- 퍼블릭 IPv4 주소 확인

탄력적 IP 주소 연결

이 탄력적 IP 주소에 연결할 인스턴스 또는 네트워크 인터페이스를 선택합니다. (43.202.249.26)

탄력적 IP 주소: 43.202.249.26

리소스 유형

탄력적 IP 주소를 연결할 리소스의 유형을 선택합니다.

인스턴스

네트워크 인터페이스

탄력적 IP 주소를 탄력적 IP 주소가 이미 연결되어 있는 인스턴스와 연결하면 이전에 연결한 탄력적 IP 주소가 연결 해제되지만 주소는 여전히 계정에 할당됩니다. 자세히 알아보기

프라이빗 IP 주소를 지정하지 않으면 탄력적 IP 주소가 기본 프라이빗 IP 주소와 연결됩니다.

인스턴스

이-0a2fcea066c446782

프라이빗 IP 주소

탄력적 IP 주소를 연결할 프라이빗 IP 주소입니다.

프라이빗 IP 주소 선택

재연결

이미 리소스에 연결되어 있는 탄력적 IP 주소를 다른 리소스에 재연결할 수 있는지를 지정합니다.

☐ 이 탄력적 IP 주소를 재연결하도록 허용

취소

연결

탄력적 IP 주소 (2) 정보

속성 또는 태그별로 탄력적 IP 주소 찾기

<input type="checkbox"/>	Name	할당된 IPv4 주소	유형	할당 ID	역방향 DNS 레코드	연결된 인스턴스 ID
<input type="checkbox"/>	eksctl-rest-eks-cluster/NATIP	3.36.228.42	퍼블릭 IP	eipalloc-03bd616b88164abe9	-	-

1. Bastion Server 생성 (EC2)

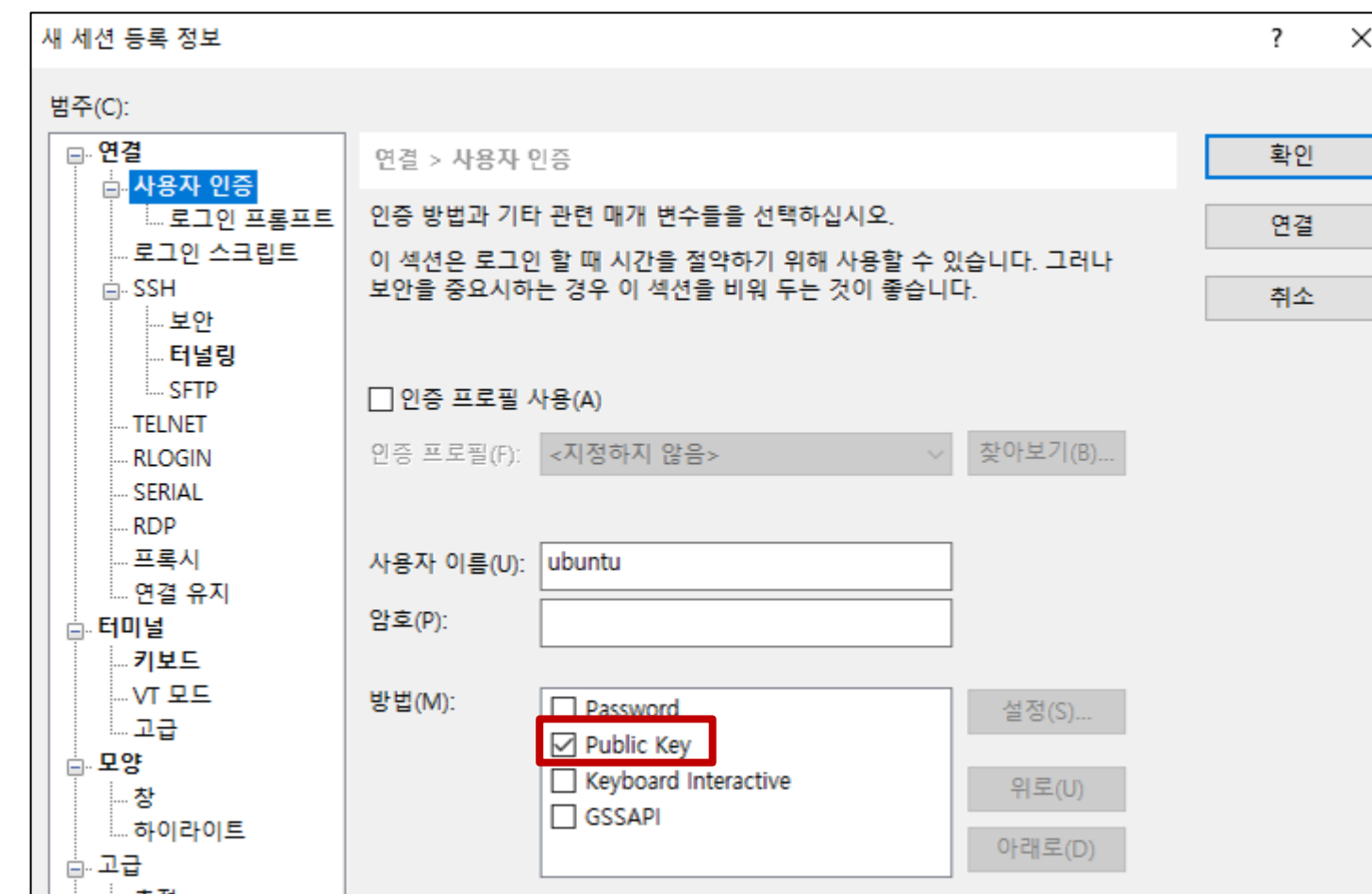
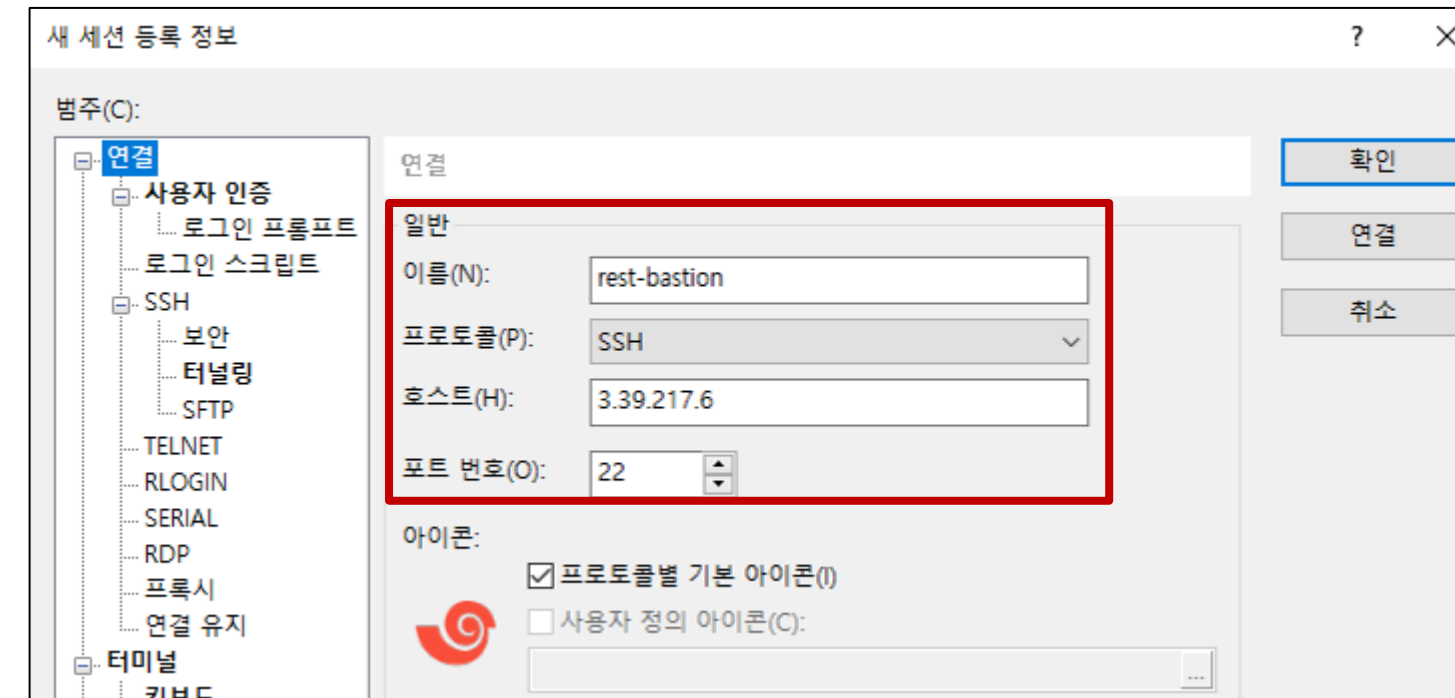
1-4. Xshell 연결

연결

- 이름 : rest-bastion
- 호스트 : 3.39.217.6
- 포트 번호 : 22

사용자 인증

- Public Key 사용
 - rest-key 가져오기



```
1 rest-bastion x +
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/pro

System information as of Thu Sep 4 08:20:27 UTC 2025

System load: 0.01      Temperature: -273.1 C
Usage of /: 25.6% of 6.71GB Processes: 116
Memory usage: 6%      Users logged in: 0
Swap usage: 0%        IPv4 address for ens5: 172.31.5.178

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-172-31-5-178:~$
```

2. Bastion Server 환경 구성

2. Bastion Server 환경 구성

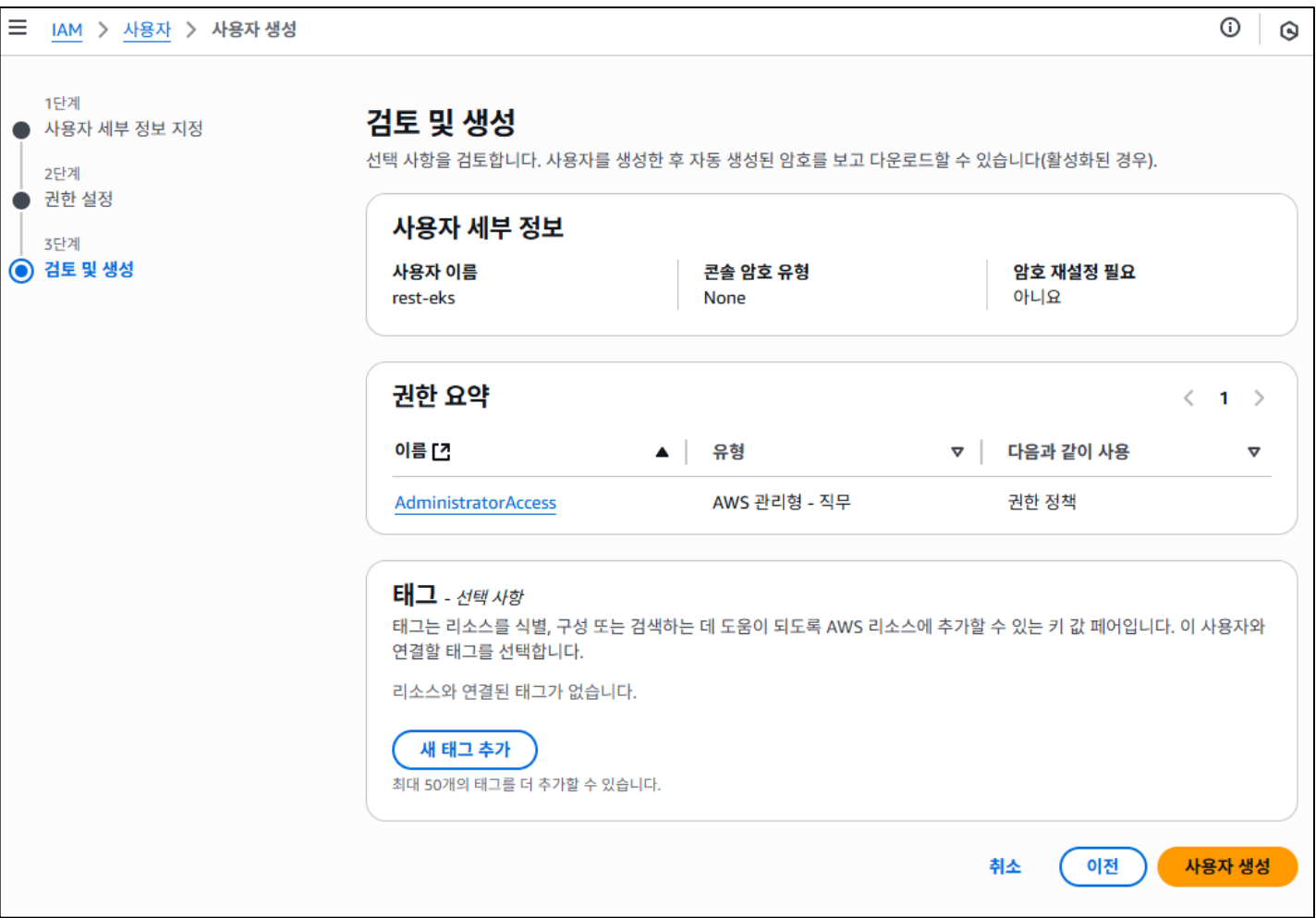
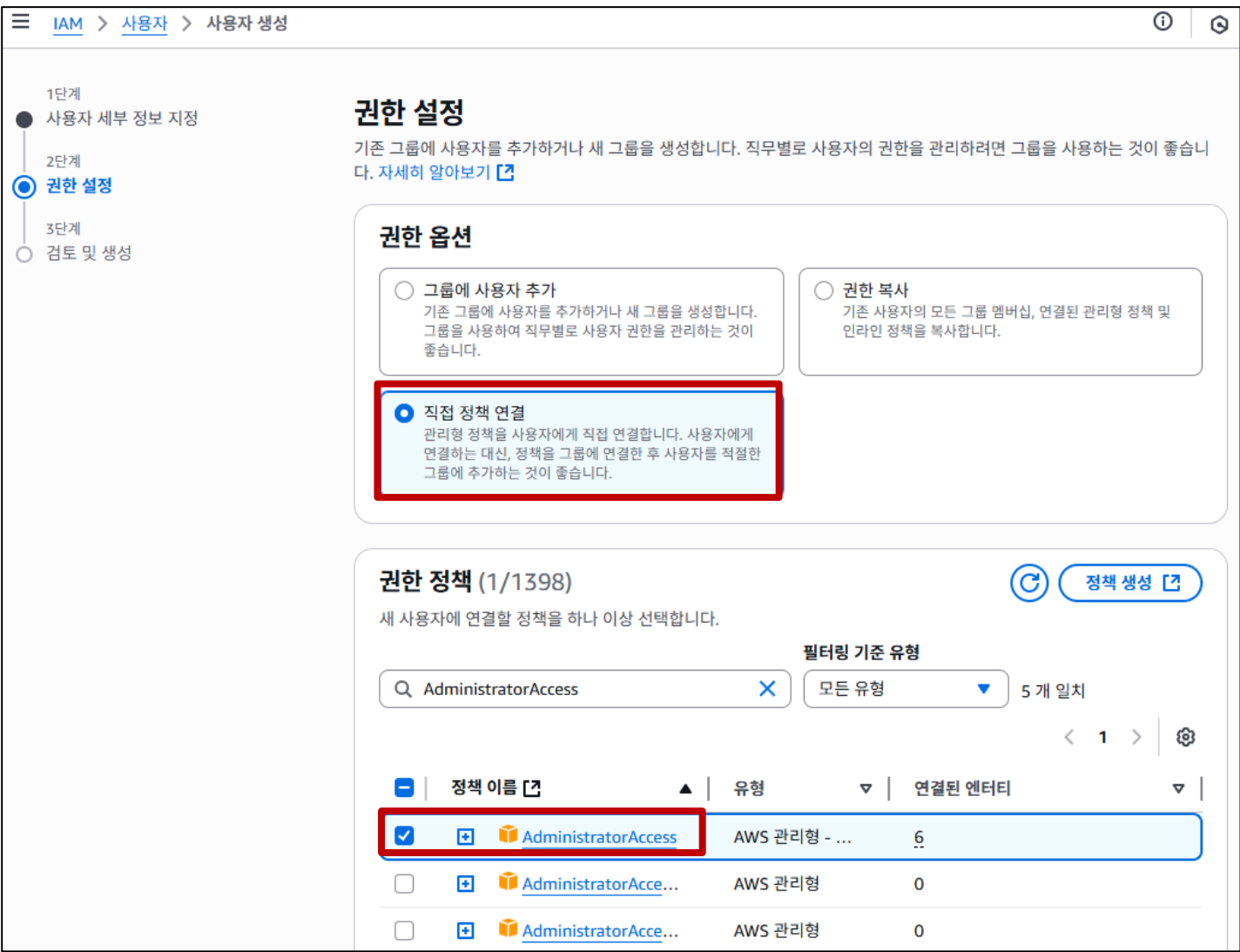
2-1. IAM

사용자 세부 정보 지정

- 사용자 이름 : rest-eks

권한 설정

- 권한 옵션
 - 직접 정책 연결
- 정책 검색
 - AdministratorAccess 선택



2. Bastion Server 환경 구성

2-2. 액세스 키 생성

액세스 키 모범 사례 및 대안

- Command Line Interface(CLI) 선택

액세스 키 검색

- .csv 파일 다운로드

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.

☐ AWS 컴퓨팅 서비스에서 실행되는 애플리케이션

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

☐ 서버 파티 서비스

AWS 리소스를 모니터링 또는 관리하는 서버 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.

취소

다음

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

AKIA2KX4BWXB7S45LCQ7

***** 표시

액세스 키 모범 사례

- 액세스 키를 일반 텍스트, 코드 리포지토리 또는 코드로 저장해서는 안됩니다.
- 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

2. Bastion Server 환경 구성

2-3. 관리 도구 설치(1)

1. AWS CLI 설치
2. 관리시스템에 AWS 계정 등록 및 확인
3. k8s 관리 도구인 kubectl 설치
4. eksctl 설치 및 확인

```
ubuntu@ip-172-31-5-178:~$ 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 59.2M  100 59.2M    0     0  221M      0 --:--:-- --:--:-- --:--:-- 221M
```

```
ubuntu@ip-172-31-5-178:~$ sudo ./aws/install
You can now run: /usr/local/bin/aws --version
```

```
ubuntu@ip-172-31-5-178:~$ aws --version
aws-cli/2.28.23 Python/3.13.7 Linux/6.14.0-1011-aws exe/x86_64.ubuntu.24
```


2. Bastion Server 환경 구성

2-3. 관리 도구 설치(2)

1. AWS CLI 설치
2. 관리시스템에 AWS 계정 등록 및 확인
3. k8s 관리 도구인 kubectl 설치
4. eksctl 설치 및 확인

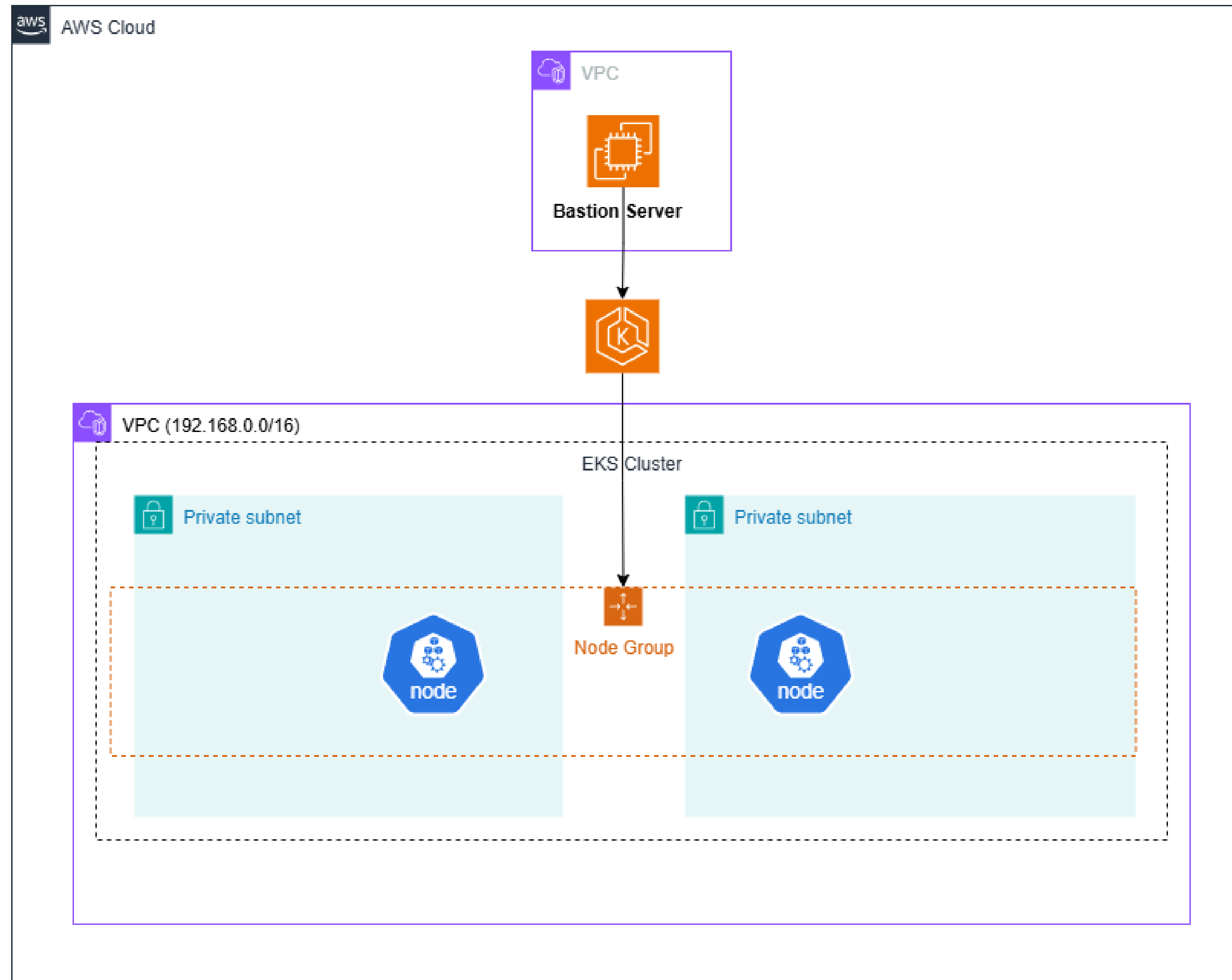
```
ubuntu@ip-172-31-5-178:~$ aws configure
AWS Access Key ID [None]: AKIA2KX4BWXB7S45LCQ7
AWS Secret Access Key [None]: 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
Default region name [None]: ap-northeast-2
Default output format [None]: json
ubuntu@ip-172-31-5-178:~$ aws sts get-caller-identity
{
  "UserId": "AIDA2KX4BWXBQDLEANKYU",
  "Account": "710271940035",
  "Arn": "arn:aws:iam::710271940035:user/hansw"
}
```

```
ubuntu@ip-172-31-12-163:~$ curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.31.0/2024-09-12/bin/linux/amd64/kubect
l
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total     Spent    Left  Speed
100 53.7M  100 53.7M    0     0  17.1M      0  0:00:03  0:00:03 --:--:-- 17.1M
ubuntu@ip-172-31-12-163:~$ chmod +x ./kubectl
ubuntu@ip-172-31-12-163:~$ mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$HOME/bin:$PATH
```

```
ubuntu@ip-172-31-5-178:~$ ARCH=amd64
ubuntu@ip-172-31-5-178:~$ PLATFORM=$(uname -s)_$ARCH
ubuntu@ip-172-31-5-178:~$ curl -sLO "https://github.com/eksctl-io/eksctl/releases/latest/download/eksctl_${PLATFORM}.ta
r.gz"
ubuntu@ip-172-31-5-178:~$ curl -sL "https://github.com/eksctl-io/eksctl/releases/latest/download/eksctl_checksums.txt
" | grep $PLATFORM | sha256sum --check
eksctl_Linux_amd64.tar.gz: OK
ubuntu@ip-172-31-5-178:~$ tar -xzf eksctl_${PLATFORM}.tar.gz -C /tmp && rm eksctl_${PLATFORM}.tar.gz
ubuntu@ip-172-31-5-178:~$ sudo mv /tmp/eksctl /usr/local/bin
ubuntu@ip-172-31-5-178:~$ eksctl version
0.214.0
```

3. EKS Cluster 생성

3. EKS Cluster - 순서도



3. EKS Cluster 생성

3-1. EKS Cluster 생성

EKS 클러스터 생성 후 확인

- 이름 : rest-eks
- 리전 : ap-northeast-2
- 노드 그룹 : rest-ng
- 노드 : 2

```
ubuntu@ip-172-31-12-163:~$ eksctl create cluster \
--name rest-eks \
--region ap-northeast-2 \
--with-oidc \
--nodegroup-name rest-ng \
--zones ap-northeast-2a,ap-northeast-2c \
--nodes 2 \
--node-type t3.medium \
--node-volume-size=20 \
--managed
```

클러스터 (1) 정보				
Q 클러스터 필터링				
클러스터 이름 ▲	상태 ▼	Kubernetes 버전 ▼	지원 기간	
<input type="radio"/> rest-eks	🟢 활성	1.32 지금 업그레이드	📅 2026년 3월 23일까지 표준 지원	

```
2025-09-04 08:54:41 [✓] all EKS cluster resources for "rest-eks" have been created
2025-09-04 08:54:41 [□] nodegroup "rest-ng" has 2 node(s)
2025-09-04 08:54:41 [□] node "ip-192-168-53-119.ap-northeast-2.compute.internal" is ready
2025-09-04 08:54:41 [□] node "ip-192-168-6-69.ap-northeast-2.compute.internal" is ready
2025-09-04 08:54:41 [□] waiting for at least 2 node(s) to become ready in "rest-ng"
2025-09-04 08:54:41 [□] nodegroup "rest-ng" has 2 node(s)
2025-09-04 08:54:41 [□] node "ip-192-168-53-119.ap-northeast-2.compute.internal" is ready
2025-09-04 08:54:41 [□] node "ip-192-168-6-69.ap-northeast-2.compute.internal" is ready
2025-09-04 08:54:41 [✓] created 1 managed nodegroup(s) in cluster "rest-eks"
2025-09-04 08:54:42 [□] kubectl command should work with "/home/ubuntu/.kube/config", try 'kubectl get nodes'
2025-09-04 08:54:42 [✓] EKS cluster "rest-eks" in "ap-northeast-2" region is ready
```

```
ubuntu@ip-172-31-5-178:~$ kubectl get no
NAME                                                    STATUS    ROLES    AGE    VERSION
ip-192-168-53-119.ap-northeast-2.compute.internal      Ready     <none>    7m45s  v1.32.7-eks-3abbe1
ip-192-168-6-69.ap-northeast-2.compute.internal        Ready     <none>    7m45s  v1.32.7-eks-3abbe1
```

4. Load Balancer Controller 생성

4. Load Balancer Controller 생성

4-1. Helm 설치 및 정책 생성

1. Helm 설치 및 확인
- 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
2. eksctl을 사용하여 IAM 역할 생성
- curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.13.3/docs/install/iam_policy.json
 - aws iam create-policy \ --policy-name AWSLoadBalancerControllerIAMPolicy \ --policy-document file://iam_policy.json
3. AWS IAM에서 정책 확인

```
ubuntu@ip-172-31-5-178:~$ curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3
ubuntu@ip-172-31-5-178:~$ chmod 700 get_helm.sh
ubuntu@ip-172-31-5-178:~$ ./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-172-31-5-178:~$ helm --help
The Kubernetes package manager

Common actions for Helm:

- helm search: search for charts
- helm pull: download a chart to your local directory to view
- helm install: upload the chart to Kubernetes
- helm list: list releases of charts

Environment variables:

| Name | Description
```

```
ubuntu@ip-172-31-5-178:~$ 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 26712 0 --:--:-- --:--:-- --:--:-- 26727
ubuntu@ip-172-31-5-178:~$ aws iam create-policy \ --policy-name AWSLoadBalancerControllerIAMPolicy \ --policy-document file://iam_policy.json
```

AWSLoadBalancerControllerIAMPolicy1 정보 편집 삭제

정책 세부 정보

유형	생성 시간	편집 시간	ARN
고객 관리형	September 03, 2025, 18:03 (UTC+09:00)	September 03, 2025, 18:03 (UTC+09:00)	arn:aws:iam:710271940035:policy/AWSLoadBalancerControllerIAMPolicy1

4. Load Balancer Controller 생성

4-2. 서비스 어카운트 생성

서비스 어카운트

- 클러스터 : rest-eks
- 네임스페이스 : kube-system
- 이름 : aws-load-balancer-controller

ARN 확인

- IAM > 정책 >
AWSLoadBalancerControllerIAMPolicy

```
ubuntu@ip-172-31-5-178:~$ export cluster_name=rest-eks
ubuntu@ip-172-31-5-178:~$ oidc_id=$(aws eks describe-cluster --name $cluster_name --query "cluster.identity.oidc.issuer" --output text | cut -d '/' -f 5)
ubuntu@ip-172-31-5-178:~$ echo $oidc_id
15DBF00C410BB68AA3991C716B17A504
ubuntu@ip-172-31-5-178:~$
ubuntu@ip-172-31-5-178:~$ eksctl utils associate-iam-oidc-provider --cluster $cluster_name --approve
2025-09-04 09:24:40 [✓] IAM Open ID Connect provider is already associated with cluster "rest-eks" in "ap-northeast-2"
```

```
ubuntu@ip-172-31-5-178:~$ eksctl create iamserviceaccount \
--cluster=rest-eks \
--namespace=kube-system \
--name=aws-load-balancer-controller \
--role-name AmazonEKSLoadBalancerControllerRole \
--attach-policy-arn=arn:aws:iam::710271940035:policy/AWSLoadBalancerControllerIAMPolicy \
--approve
2025-09-04 09:27:41 [✓] 1 iamserviceaccount (kube-system/aws-load-balancer-controller) was included (based on the include/exclude rules)
2025-09-04 09:27:41 [!] serviceaccounts that exist in Kubernetes will be excluded, use --override-existing-serviceaccounts to override
2025-09-04 09:27:41 [✓] 1 task: {
  2 sequential sub-tasks: {
    create IAM role for serviceaccount "kube-system/aws-load-balancer-controller",
    create serviceaccount "kube-system/aws-load-balancer-controller",
  } }
2025-09-04 09:27:41 [✓] building iamserviceaccount stack "eksctl-rest-eks-addon-iamserviceaccount-kube-system-aws-load-balancer-controller"
2025-09-04 09:27:41 [✓] deploying stack "eksctl-rest-eks-addon-iamserviceaccount-kube-system-aws-load-balancer-controller"
2025-09-04 09:27:41 [✓] waiting for CloudFormation stack "eksctl-rest-eks-addon-iamserviceaccount-kube-system-aws-load-balancer-controller"
2025-09-04 09:28:11 [✓] waiting for CloudFormation stack "eksctl-rest-eks-addon-iamserviceaccount-kube-system-aws-load-balancer-controller"
2025-09-04 09:28:11 [✓] created serviceaccount "kube-system/aws-load-balancer-controller"
```

```
ubuntu@ip-172-31-5-178:~$ 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::710271940035:role/AmazonEKSLoadBalancerControllerRole
```

4. Load Balancer Controller 생성

4-3. LB Controller 설치

Helm Repository 등록 및 업데이트

- helm repo add eks <https://aws.github.io/eks-charts>
- helm repo update eks

AWS LB Controller 설치

- 클러스터 이름 : rest-eks
- 이름 : aws-loadbalancer-controller

LB Controller 배포 상태 확인

```
ubuntu@ip-172-31-5-178:~$ helm repo add eks https://aws.github.io/eks-charts
"eks" has been added to your repositories
ubuntu@ip-172-31-5-178:~$ 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-172-31-5-178:~$ helm install aws-load-balancer-controller eks/aws-load-balancer-controller \
-n kube-system \
--set clusterName=rest-eks \
--set serviceAccount.create=false \
--set serviceAccount.name=aws-load-balancer-controller
NAME: aws-load-balancer-controller
LAST DEPLOYED: Thu Sep  4 09:36:18 2025
NAMESPACE: kube-system
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
AWS Load Balancer controller installed!
```

```
ubuntu@ip-172-31-5-178:~$ 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           2m54s
```

```
ubuntu@ip-172-31-5-178:~$ kubectl get pod -n kube-system
NAME                                READY   STATUS    RESTARTS   AGE
aws-load-balancer-controller-ff94bbff9-fl5bb  1/1     Running   0          3m40s
aws-load-balancer-controller-ff94bbff9-xkdn6  1/1     Running   0          3m40s
aws-node-hvljx                             2/2     Running   0          46m
aws-node-ktt4p                             2/2     Running   0          46m
coredns-844d8f59bb-52twq                  1/1     Running   0          49m
coredns-844d8f59bb-bwjjc                  1/1     Running   0          49m
kube-proxy-5t8kg                           1/1     Running   0          46m
kube-proxy-cs7w5                           1/1     Running   0          46m
metrics-server-67b599888d-ln92m            1/1     Running   0          49m
metrics-server-67b599888d-n577b            1/1     Running   0          49m
```

5. Load Balancer 배포 - NLB

5. Load Balancer 배포 - NLB

5-1. 샘플 애플리케이션 배포(1)

네임스페이스 생성

- 이름 : nlb-rest-app
- 경로 : namespace/nlb-rest-app

디플로이먼트 생성

- 야물 파일 생성, 적용
 - 이름 : rest-deployment.yaml
- 이름 : nlb-rest-app
- 이미지 : nginx:1.23
- 포트 : 80

```
ubuntu@ip-172-31-5-178:~$ kubectl create namespace nlb-rest-app
namespace/nlb-rest-app created
ubuntu@ip-172-31-5-178:~$ kubectl get namespace
NAME                STATUS   AGE
default             Active   56m
kube-node-lease     Active   56m
kube-public         Active   56m
kube-system         Active   56m
nlb-rest-app        Active   13s
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nlb-rest-app
  namespace: nlb-rest-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
```

```
ubuntu@ip-172-31-5-178:~$ kubectl apply -f rest-deployment.yaml
deployment.apps/nlb-rest-app created
ubuntu@ip-172-31-5-178:~$ kubectl get deployments.apps -n nlb-rest-app
NAME                READY   UP-TO-DATE   AVAILABLE   AGE
nlb-rest-app        3/3     3             3           22s
```

5. Load Balancer 배포 - NLB

5-2. 샘플 애플리케이션 배포(2)

서비스 생성

- 야물 파일 생성, 적용
 - 이름 : rest-service.yaml
- 이름 : nlb-rest-service
- 포트 : 80
- 타입 : 로드밸런서

서비스 생성 후 확인

```
apiVersion: v1
kind: Service
metadata:
  name: nlb-rest-service
  namespace: nlb-rest-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
```

```
ubuntu@ip-172-31-5-178:~$ kubectl apply -f rest-service.yaml
service/nlb-rest-service created
ubuntu@ip-172-31-5-178:~$ kubectl get svc -n nlb-rest-app nlb-rest-service
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
nlb-rest-service	LoadBalancer	10.100.7.124	<pending>	80:30282/TCP	24s

로드 밸런서 (1) 작업 ▼ 로드 밸런서 생성 ▼

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

<input type="checkbox"/>	이름 ▼	상태 ▼	유형 ▼	체계 ▼	IP 주소 유형 ▼	VPC ID ↗ ▼
<input type="checkbox"/>	k8s-nlbresta-nlbrests-b...	✓ 활성	network	Internet-facing	IPv4	vpc-0225b351509612138

6. Load Balancer 배포 - ALB

6. Load Balancer 배포 - ALB

6-1. 샘플 애플리케이션 배포(1)

- 네임스페이스 생성**
- 이름 : rest-game-2048
- 디플로이먼트 생성**
- 야물 파일 생성, 적용
 - 이름 : rest-game-2048.yaml
 - 이름 : rest-game-2048
 - 포트 : 80
- 서비스 생성**
- 이름 : rest-game-2048
 - 타입 : NodePort

```
apiVersion: v1
kind: Namespace
metadata:
  name: rest-game-2048
---
apiVersion: apps/v1
kind: Deployment
metadata:
  namespace: rest-game-2048
  name: rest-deployment-2048
spec:
  selector:
    matchLabels:
      app.kubernetes.io/name: rest-app-2048
  replicas: 5
  template:
    metadata:
      labels:
        app.kubernetes.io/name: rest-app-2048
    spec:
      containers:
        - image: public.ecr.aws/l6m2t8p7/docker-2048:latest
          imagePullPolicy: Always
          name: rest-app-2048
          ports:
            - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  namespace: rest-game-2048
  name: rest-service-2048
spec:
  ports:
    - port: 80
      targetPort: 80
      protocol: TCP
  type: NodePort
  selector:
    app.kubernetes.io/name: rest-app-2048
```

```
ubuntu@ip-172-31-12-163:~$ kubectl apply -f rest-game-2048.yaml
namespace/rest-game-2048 created
deployment.apps/rest-deployment-2048 created
service/rest-service-2048 created
ingress.networking.k8s.io/rest-ingress-2048 created
ubuntu@ip-172-31-12-163:~$ kubectl get pod -n rest-game-2048
```

NAME	READY	STATUS	RESTARTS	AGE
rest-deployment-2048-7d7b4844c6-7w95q	1/1	Running	0	17s
rest-deployment-2048-7d7b4844c6-9vqqb	1/1	Running	0	17s
rest-deployment-2048-7d7b4844c6-fdwf6	1/1	Running	0	17s
rest-deployment-2048-7d7b4844c6-fnkls	1/1	Running	0	17s
rest-deployment-2048-7d7b4844c6-xl9rl	1/1	Running	0	17s

6. Load Balancer 배포 - ALB

6-2. 샘플 애플리케이션 배포(2)

Ingress 생성

- 야물 파일 생성, 적용
 - 이름 : rest-ingress-2048.yaml
- 이름 : rest-game-2048
- 포트 : 80
- 클래스네임 : alb

Ingress 확인

- 생성한 로드밸런서의 DNS 이름복사
- 브라우저에서 확인하면 2048 게임 확인 가능

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  namespace: rest-game-2048
  name: rest-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: rest-service-2048
            port:
              number: 80
```

```
ubuntu@ip-172-31-12-163:~$ kubectl describe -n rest-game-2048 ingress rest-ingress-2048
Name:                rest-ingress-2048
Labels:               <none>
Namespace:            rest-game-2048
Address:              k8s-restgame-restingr-6dc5e42905-2009310684.ap-northeast-2.elb.amazonaws.com
Ingress Class:        alb
Default backend:      <default>
Rules:
  Host      Path  Backends
  ----      -
  *          /    rest-service-2048:80 (192.168.17.240:80,192.168.31.77:80,192.168.38.127:80 + 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  89s   ingress   Successfully reconciled
```

k8s-restgame-restingr-6dc5e42905

▼ 세부 정보

로드 밸런서 유형

애플리케이션

상태

🟢 활성

체계

Internet-facing

호스팅 영역

ZWKZPGTI48KDX

로드 밸런서 ARN

arn:aws:elasticloadbalancing:ap-northeast-2:710271940035:loadbalancer/app/k8s-restgame-restingr-6dc5e42905/83ec8fabd5960e81

VPC

vpc-0225b351509612138

가용 영역

subnet-057df4667321b8d94 ap-northeast-2c (apne2-az3)

subnet-07ca175dfd2efb8e4 ap-northeast-2a (apne2-az1)

로드 밸런서 IP 주소 유형

IPv4

생성된 날짜

2025년 9월 5일, 19:30 (UTC+09:00)

DNS 이름 정보

k8s-restgame-restingr-6dc5e42905-2009310684.ap-northeast-2.elb.amazonaws.com (A 레코드)

6. Load Balancer 배포 - ALB

6-3. Route53 설정

Route53 생성

- 도메인 이름 : siwan222.store

Route53 레코드 생성

- 레코드 이름 : www
- 별칭 : Application/Classic Load Balancer
- 레코드 이름 : (공백)
- 별칭 : Application/Classic Load Balancer

호스팅 영역 생성

호스팅 영역 구성

호스팅 영역은 example.com 같은 도메인과 관련 하위 도메인에 대한 트래픽을 라우팅하는 방식에 대한 정보를 포함하는 컨테이너입니다.

도메인 이름

트래픽을 라우팅할 도메인의 이름입니다.

siwan222.store

설명 - 선택 사항

이 값을 사용하면 이름이 동일한 호스팅 영역을 구별할 수 있습니다.

호스팅 영역이 사용되는 경우.

유형

☒ 퍼블릭 호스팅 영역

☐ 프라이빗 호스팅 영역

퍼블릭 호스팅 영역은 인터넷에서 트래픽을 라우팅하는 방식을 설정합니다.

프라이빗 호스팅 영역은 Amazon VPC 내에서 트래픽을 라우팅하는 방식을 설정합니다.

태그

호스팅 영역에 태그를 사용하면 호스팅 영역을 쉽게 구성하고 식별할 수 있습니다.

리소스와 연결된 태그가 없습니다.

태그 추가

최대 50개의 태그를 더 추가할 수 있습니다.

취소

호스팅 영역 생성

레코드 생성

빠른 레코드 생성

▼ 레코드 1

레코드 이름

www

siwan222.store

별칭

Application/Classic Load Balancer에 대한 별칭

아시아 태평양(서울)

Q, dualstack.k8s-restgame-restingr-6dc5e42905-2009310684.ap-northeast-2.elb.amazonaws.com

사용: 'dualstack.k8s-restgame-restingr-6dc5e42905-2009310684.ap-northeast-2.elb.amazonaws.com'

dualstack.k8s-restgame-restingr-6dc5e42905-2009310684.ap-northeast-2.elb.amazonaws.com

단순 라우팅

레코드 생성

빠른 레코드 생성

▼ 레코드 1

레코드 이름

subdomain

siwan222.store

별칭

Application/Classic Load Balancer에 대한 별칭

아시아 태평양(서울)

Q, dualstack.k8s-restgame-restingr-6dc5e42905-2009310684.ap-northeast-2.elb.amazonaws.com

발장 호스팅 영역 ID: ZWKCZPGT148K00X

라우팅 정책

단순 라우팅

대상 상태 평가

☒ 예

다른 레코드 추가

취소

레코드 생성

6. Load Balancer 배포 - ALB

6-3. Route53 설정

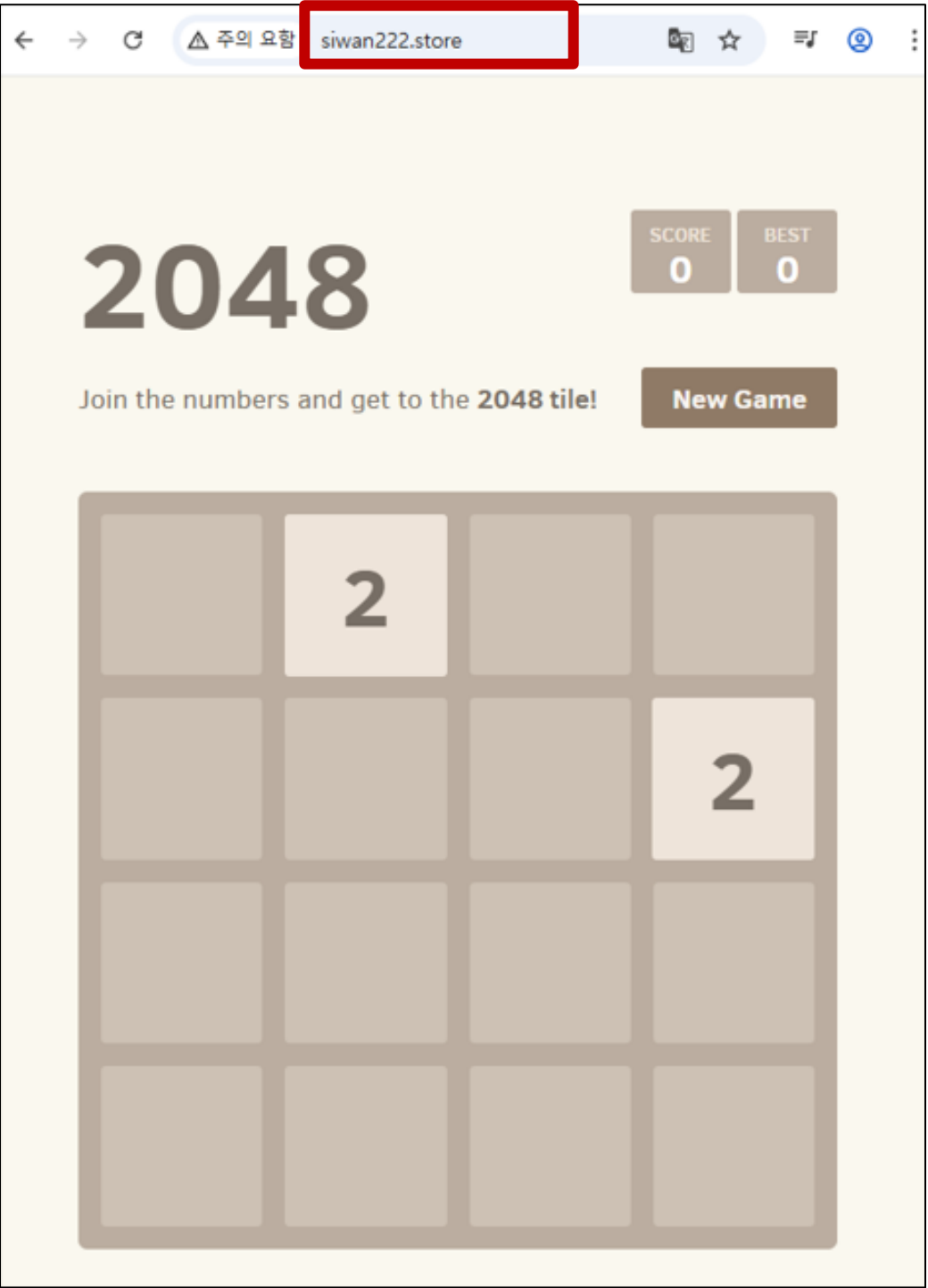
생성한 도메인 가비아에 DNS 등록

- ns-1513.awsdns-61.org
- ns-9.awsdns-01.com
- ns-804.awsdns-36.net
- ns-1771.awsdns-29.co.ur

도메인주소로 ALB 서비스 동작 확인

siwan222.store									
호스팅 영역 세부 정보									
레코드(4)									
Automatic 모드는 최상의 필터 결과에 최적화된 현재 검색 동작입니다. 모드를 변경하려면 설정(settings)으로 이동합니다.									
속성 또는 값을 기준으로 레코드 필터링									
<input type="checkbox"/>	레코드 이름	유형	라우팅 ...	자별화...	별칭	값/트래픽 라우팅 대상	TTL(초)		
<input type="checkbox"/>	siwan222.store	A	단순	-	예	dualstack.k8s-restgame-resti...	-		
<input type="checkbox"/>	siwan222.store	NS	단순	-	아니요	ns-1513.awsdns-61.org, ns-9.awsdns-01.com, ns-804.awsdns-36.net, ns-1771.awsdns-29.co.uk	172800		
<input type="checkbox"/>	siwan222.store	SOA	단순	-	아니요	ns-1513.awsdns-61.org, aws...	900		
<input type="checkbox"/>	www.siwan222.store	A	단순	-	예	dualstack.k8s-restgame-resti...	-		

네임서버 설정	
1차	ns-1513.awsdns-61.org
2차	ns-9.awsdns-01.com
3차	ns-804.awsdns-36.net
4차	ns-1771.awsdns-29.co.uk
5차	데이터 없음
6차	데이터 없음
7차	데이터 없음



감사합니다