AWS EKS Project

REST | 강승환 고동우 유세종 최성민 한시완

목차

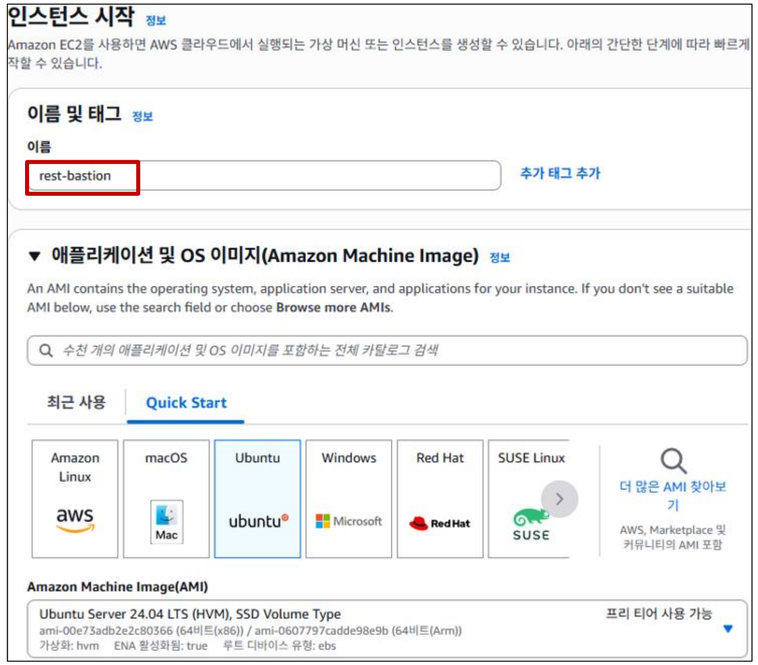
1. Bastion Server 생성 (EC2)

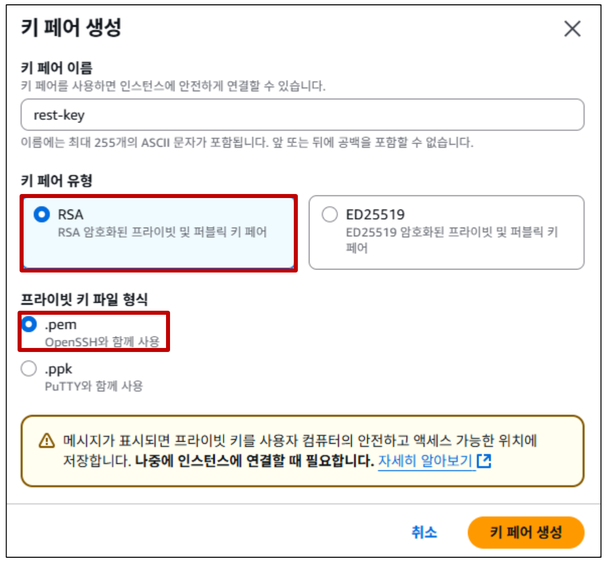
1. Bastion Server 생성 (EC2)

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



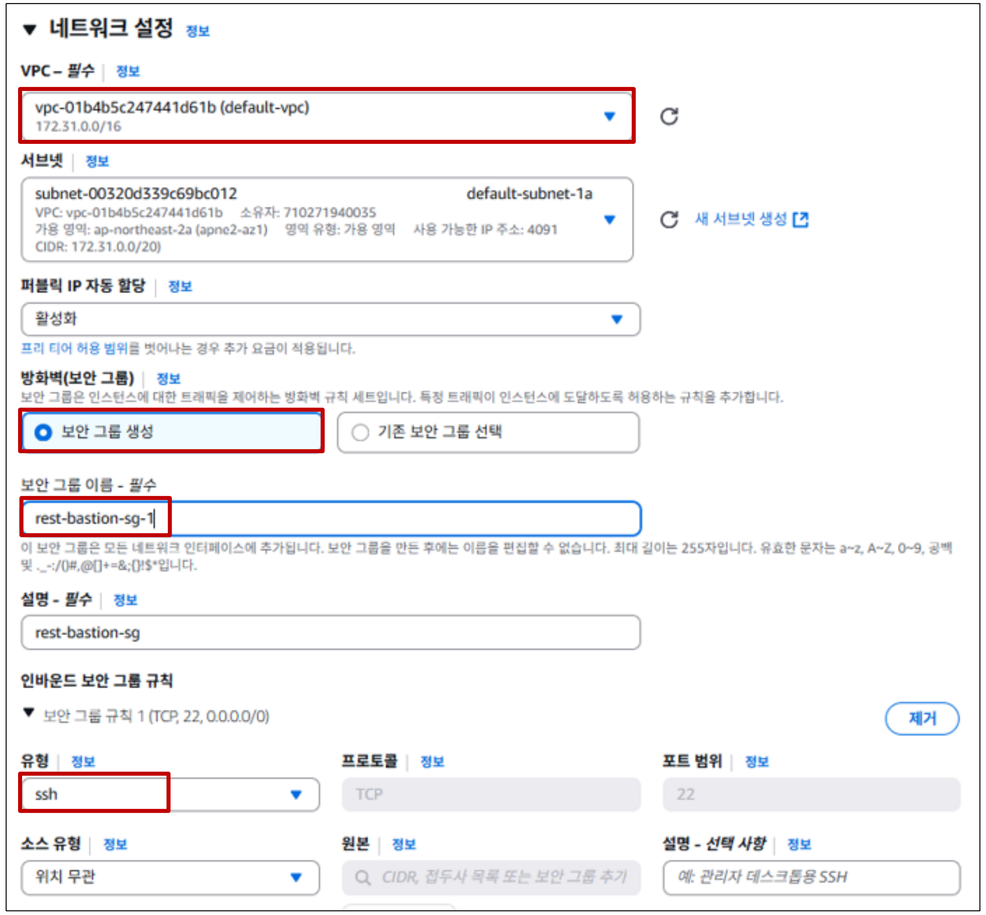


1. Bastion Server 생성 (EC2)

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)

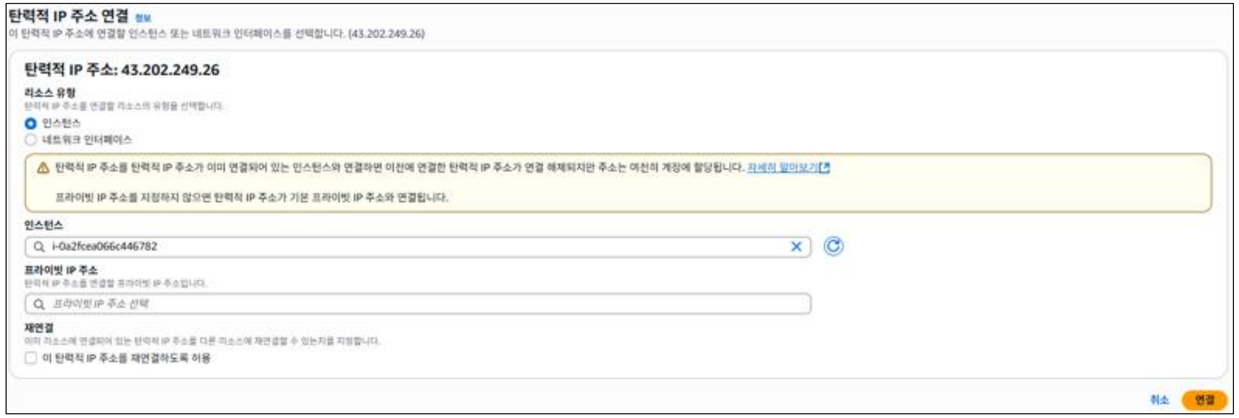
키 페어
- 이름 : rest-key
- 키 페어 유형 : RSA
- 파일 형식 : .pem



1. Bastion Server 생성 (EC2)

1-3. 탄력적 IP 연결

탄력적 IP 생성
- 만든 인스턴스로 연결
- 퍼블릭 IPv4 주소 확인

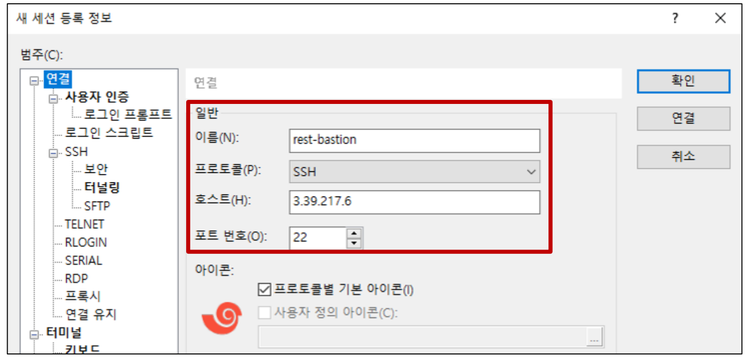


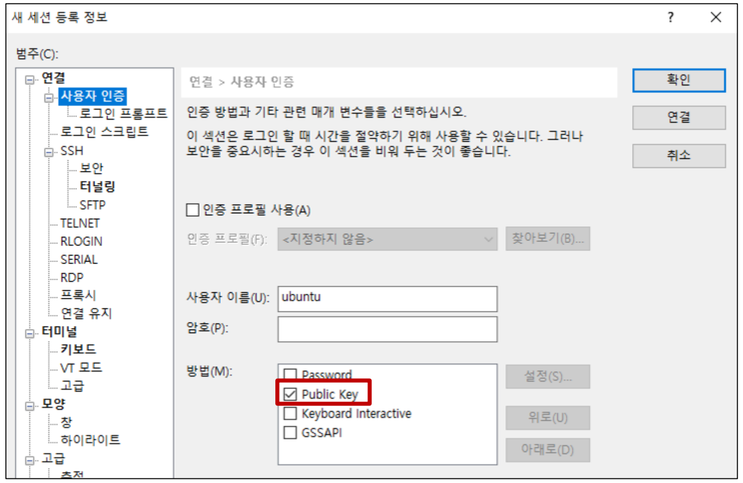
1. Bastion Server 생성 (EC2)

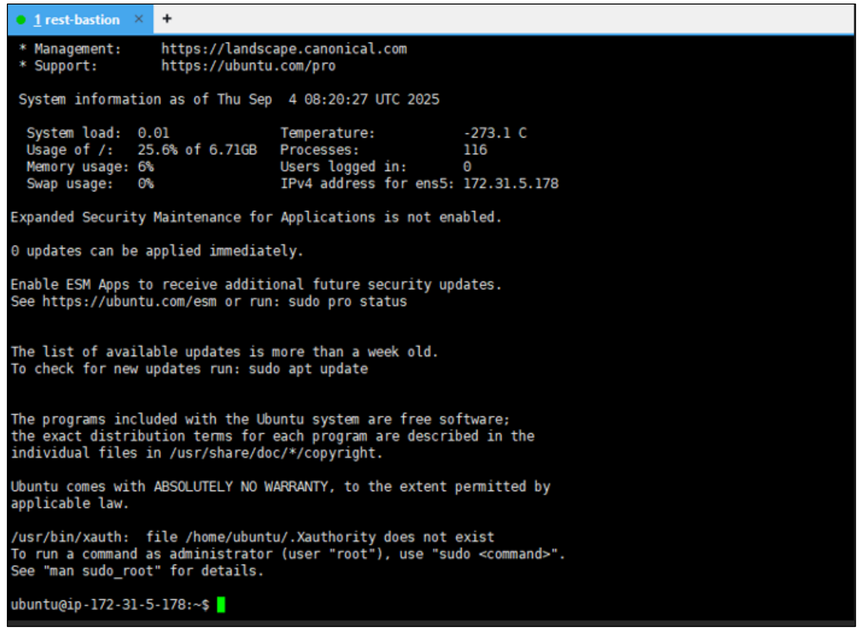
1-4. Xshell 연결

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

사용자 인증
- Public Key 사용
  - rest-key 가져오기







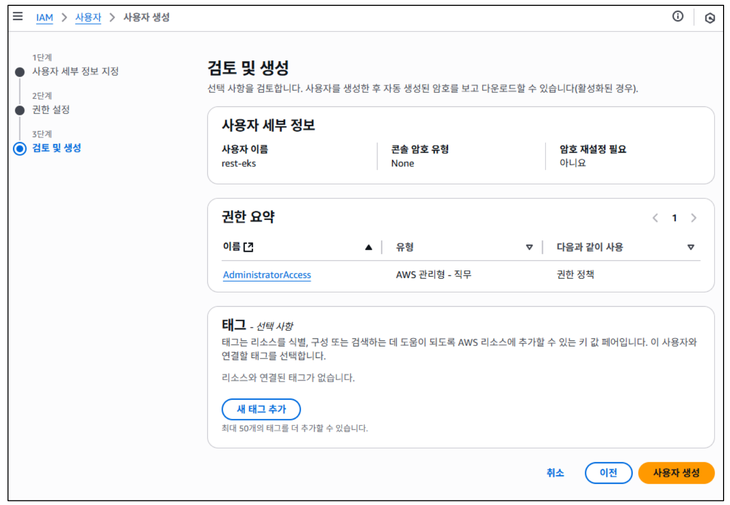
2. Bastion Server 환경 구성

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

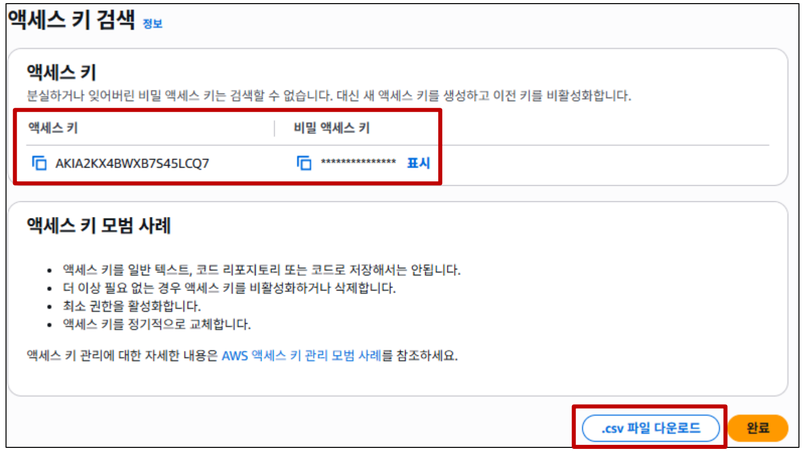


2. Bastion Server 환경 구성

2-2. 액세스 키 생성

액세스 키 모범 사례 및 대안
- Command Line Interface(CLI) 선택

액세스 키 검색
- .csv 파일 다운로드



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

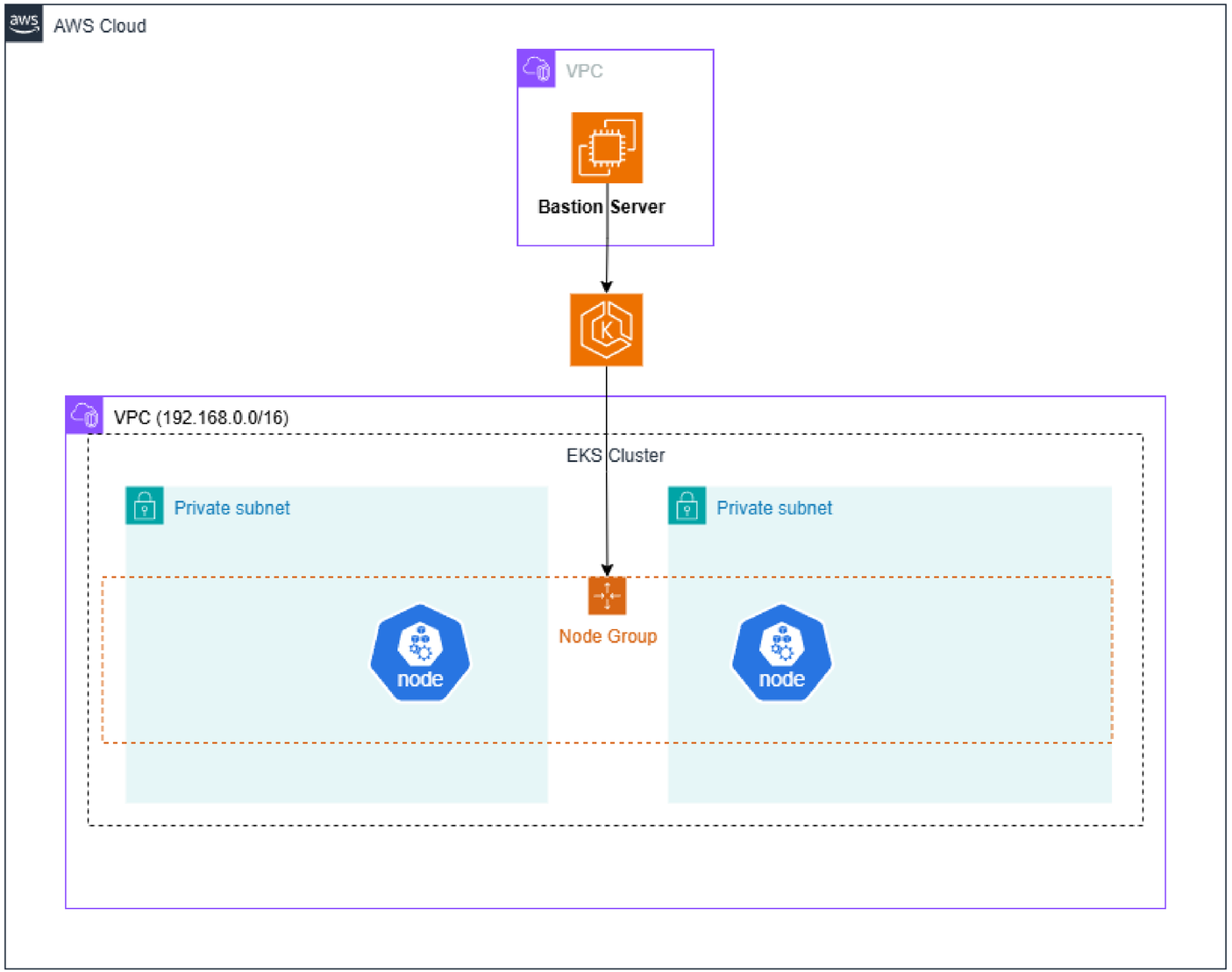
```
ubuntu@ip-172-31-5-178:~$ aws configure
AWS Access Key ID [None]: AKIA2KX4BWXB7S45LCQ7
AWS Secret Access Key [None]: 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
Default region name [None]: ap-northeast-2
Default output format [None]: json
ubuntu@ip-172-31-5-178:~$ aws sts get-caller-identity
{
    "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/ku
bectl
  % 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



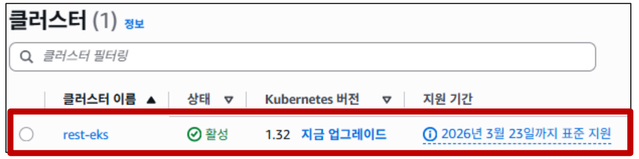
3. EKS Cluster 생성

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
```




```
2025-09-04 08:54:41 [✔]  all EKS cluster resources for "rest-eks" have been created
2025-09-04 08:54:41 [ℹ]  nodegroup "rest-ng" has 2 node(s)
2025-09-04 08:54:41 [ℹ]  node "ip-192-168-53-119.ap-northeast-2.compute.internal" is ready
2025-09-04 08:54:41 [ℹ]  node "ip-192-168-6-69.ap-northeast-2.compute.internal" is ready
2025-09-04 08:54:41 [ℹ]  waiting for at least 2 node(s) to become ready in "rest-ng"
2025-09-04 08:54:41 [ℹ]  nodegroup "rest-ng" has 2 node(s)
2025-09-04 08:54:41 [ℹ]  node "ip-192-168-53-119.ap-northeast-2.compute.internal" is ready
2025-09-04 08:54:41 [ℹ]  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 [ℹ]  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-3abbec1
ip-192-168-6-69.ap-northeast-2.compute.internal     Ready    <none>   7m45s   v1.32.7-eks-3abbec1
```

4. Load Balancer Controller 생성

4. Load Balancer Controller 생성

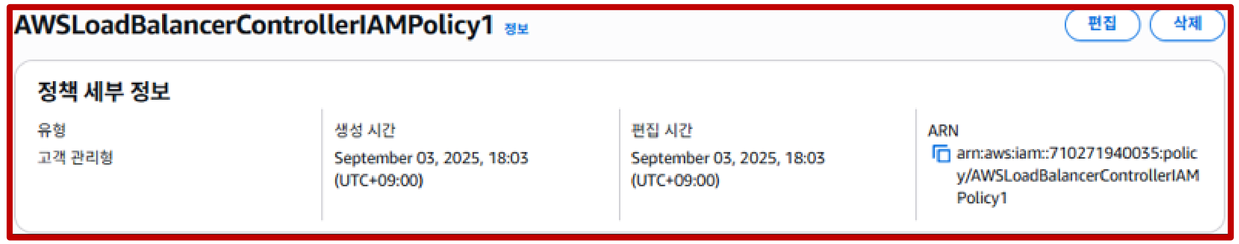
4-1. Helm 설치 및 정책 생성

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
```



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 [ℹ]  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 [ℹ]  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 [ℹ]  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 [ℹ]  building iamserviceaccount stack "eksctl-rest-eks-addon-iamserviceaccount-kube-system-aws-load-balancer-controller"
2025-09-04 09:27:41 [ℹ]  deploying stack "eksctl-rest-eks-addon-iamserviceaccount-kube-system-aws-load-balancer-controller"
2025-09-04 09:27:41 [ℹ]  waiting for CloudFormation stack "eksctl-rest-eks-addon-iamserviceaccount-kube-system-aws-load-balancer-controller"
2025-09-04 09:28:11 [ℹ]  waiting for CloudFormation stack "eksctl-rest-eks-addon-iamserviceaccount-kube-system-aws-load-balancer-controller"
2025-09-04 09:28:11 [ℹ]  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
```

```yaml
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

```yaml
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
```

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

```yaml
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

```yaml
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
```

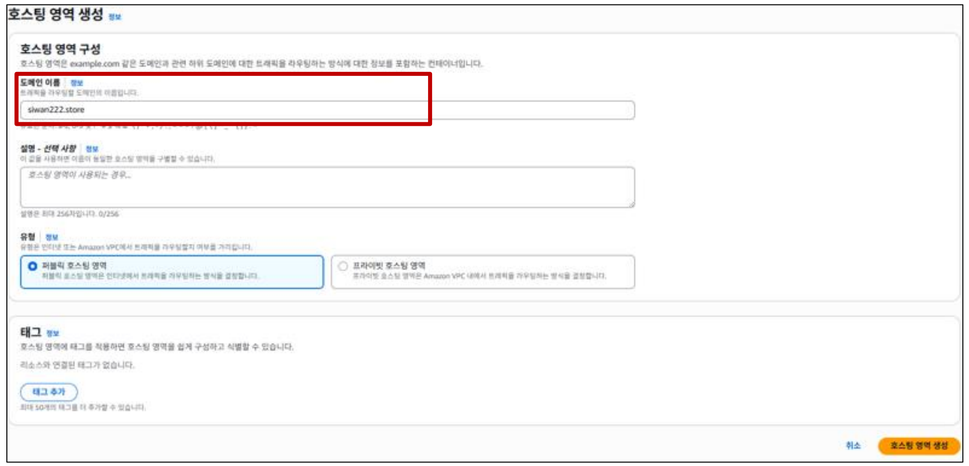
6. Load Balancer 배포 - ALB

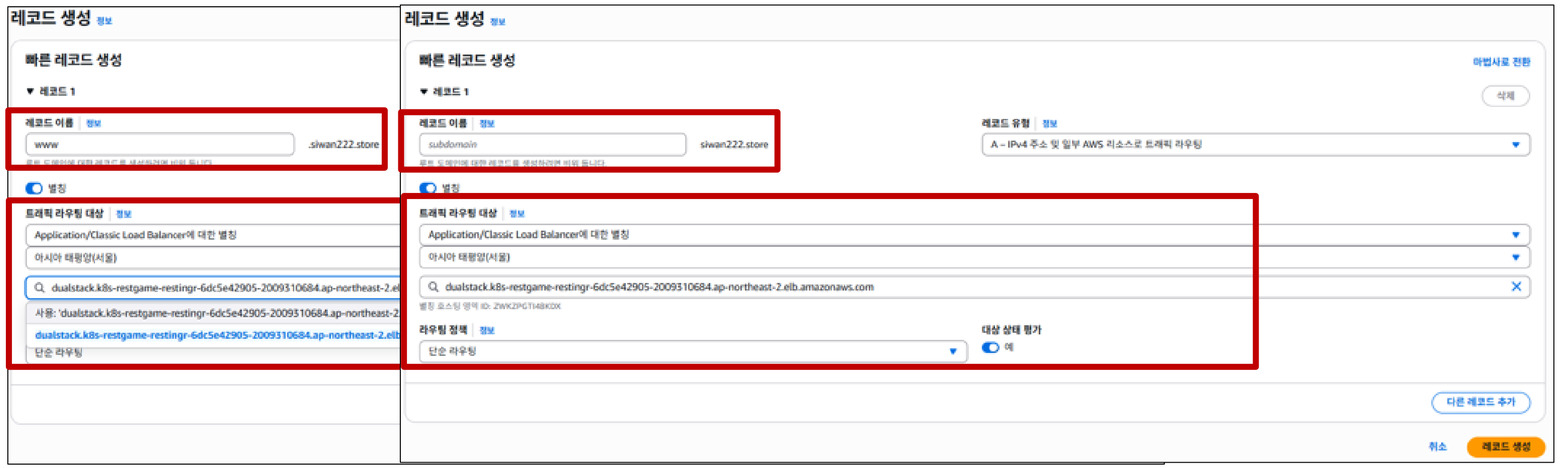
6-3. Route53 설정

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





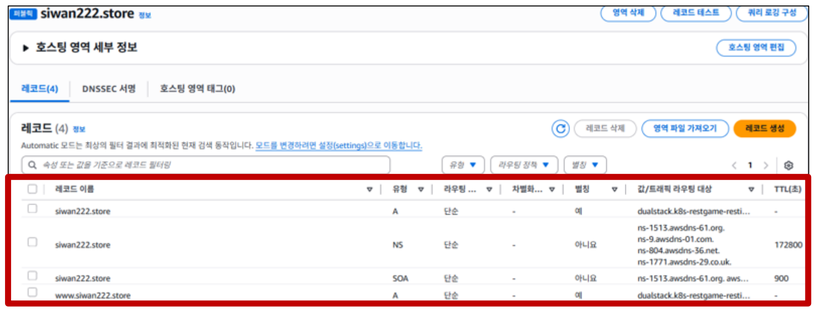
6. Load Balancer 배포 - ALB

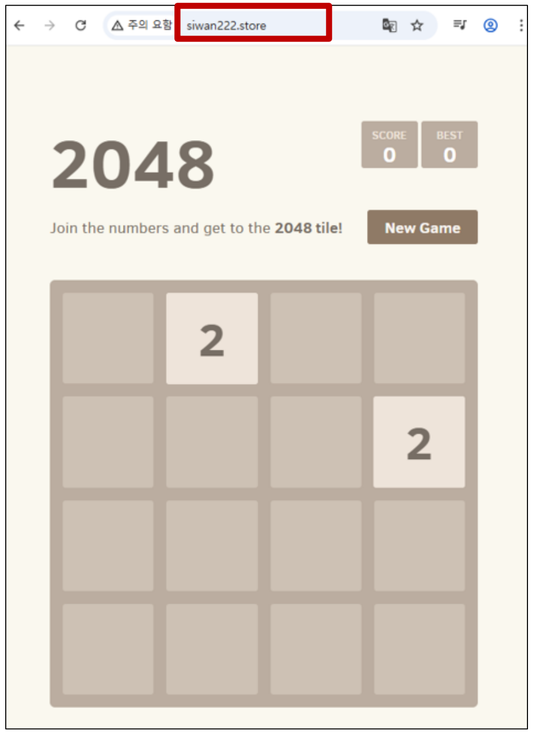
6-3. Route53 설정

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





감사합니다