APEX

AWS EKS 프로젝트

[2025.09.04 ~ 2025.10.21]

대우직업능력개발원
Daewoo Development Institute for Vocational Ability

APEX

이정일, 서재권, 임재근, 김민호

APEX

<목차>

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

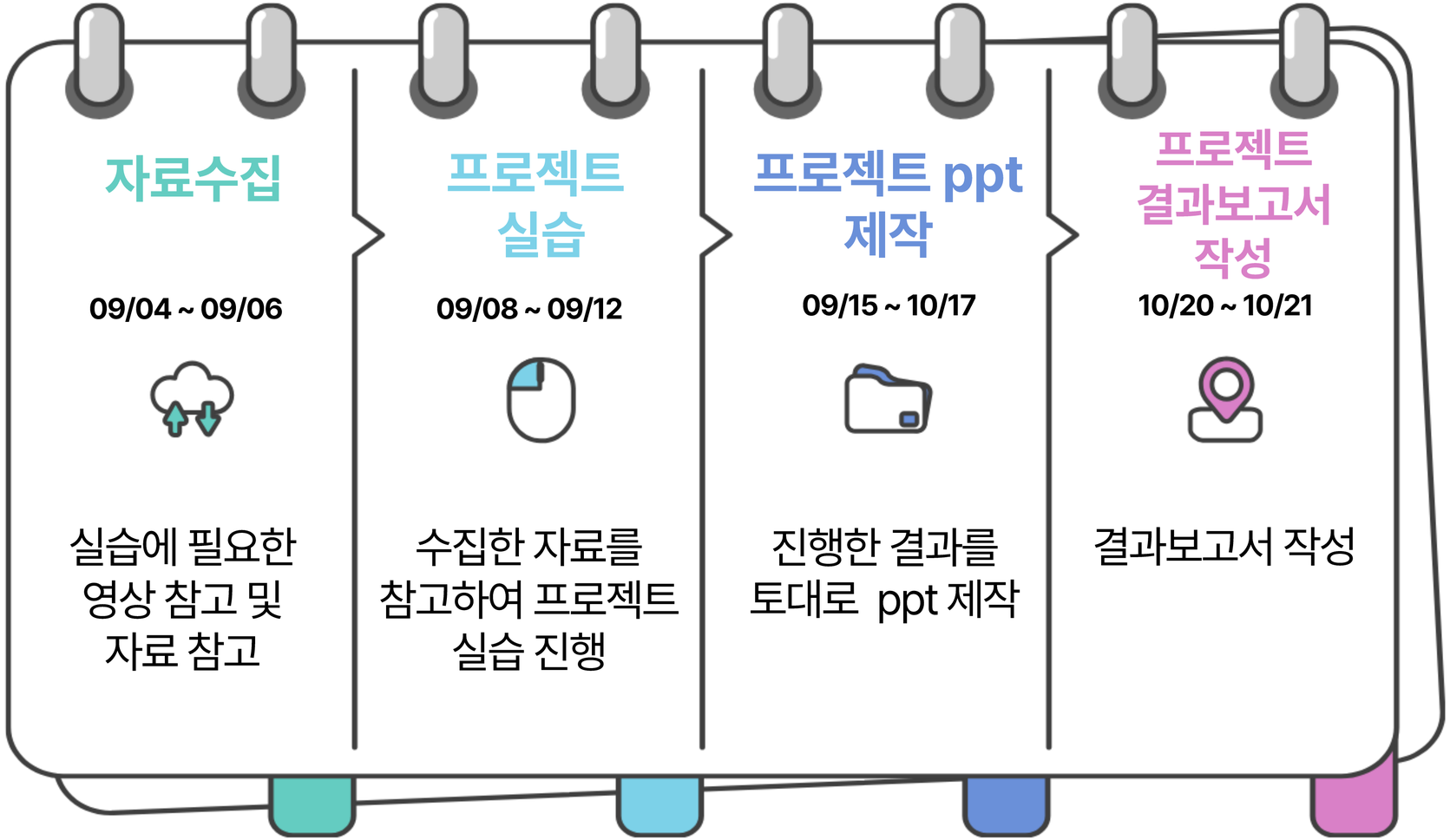
프로젝트 구조

- 계획 -> 시스템 구성 및 구축 -> 최종 결과

2. 프로젝트 팀 구성 및 역할

팀원 , 역할, 담당 역할			
<이정일>	<임재근>	<서재권>	<김민호>
• 자료 조사 • 프로젝트 실습 진행	• 자료 조사 • 프로젝트 실습 진행	• 프로젝트 보고서 작성 • PPT 제작	• 프로젝트 보고서 작성 • PPT 제작
#조장	#팀원	#팀원	#팀원

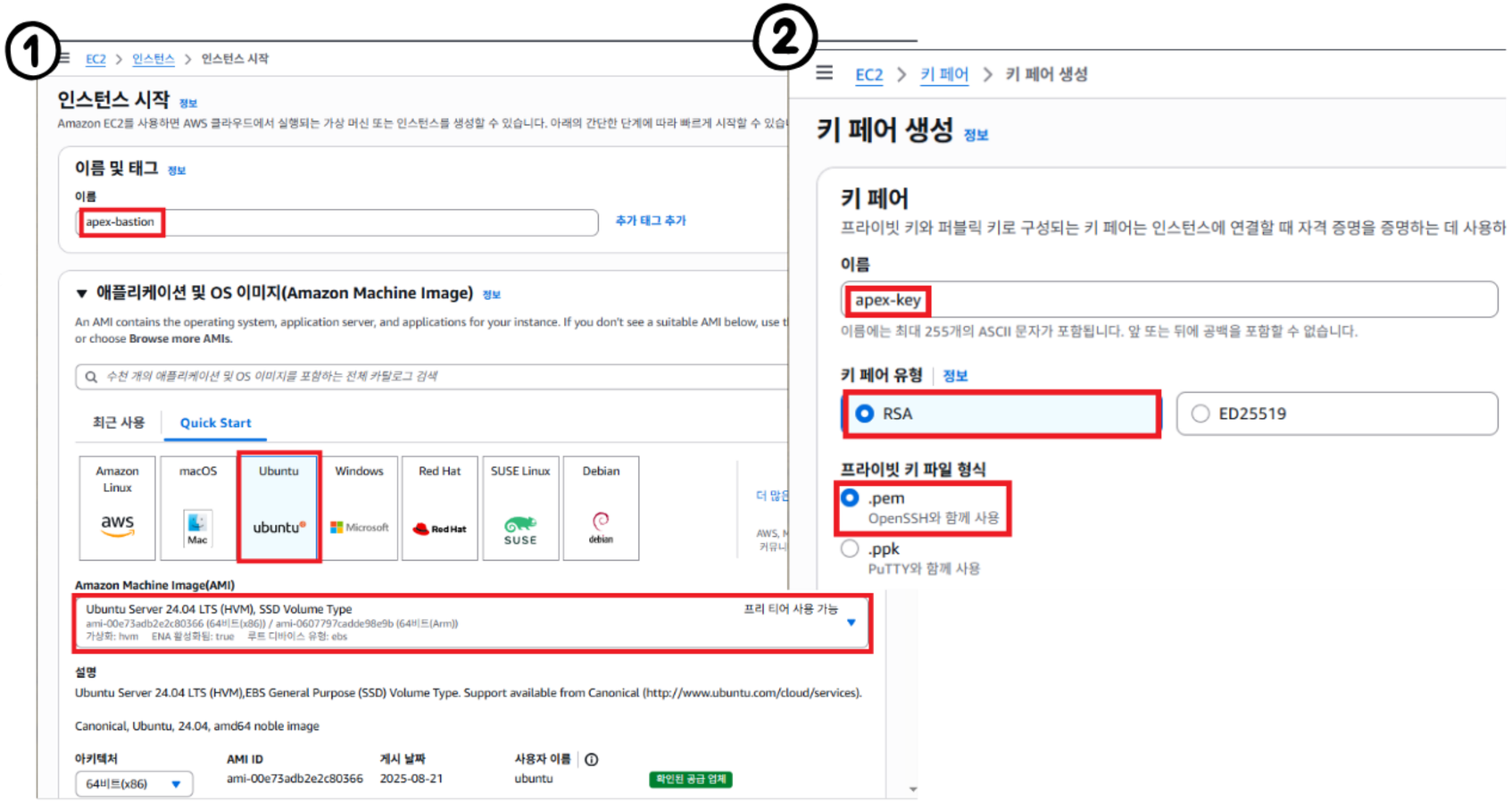
3. 프로젝트 수행절차 및 방법



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



4. 프로젝트 수행결과 | Bastion Server 생성

1-2. 인스턴스 생성(2)

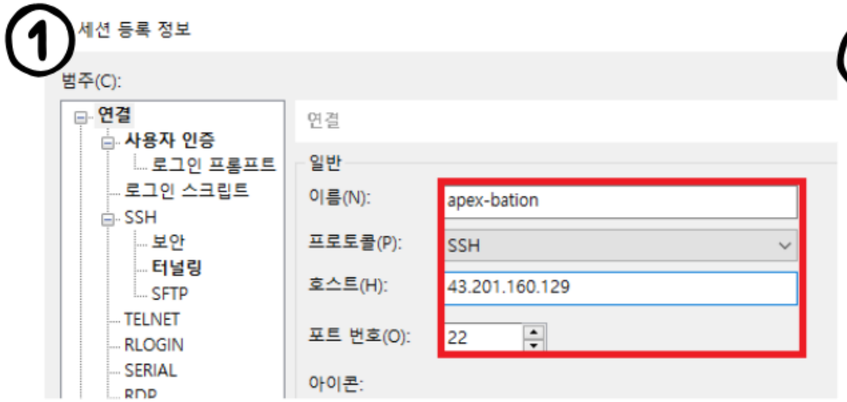
- 네트워크 설정
  - VPC : 기본값
  - 보안그룹 생성
    - 이름 : apex-bastion-sg
- 생성 유형 : ssh (위치무관)

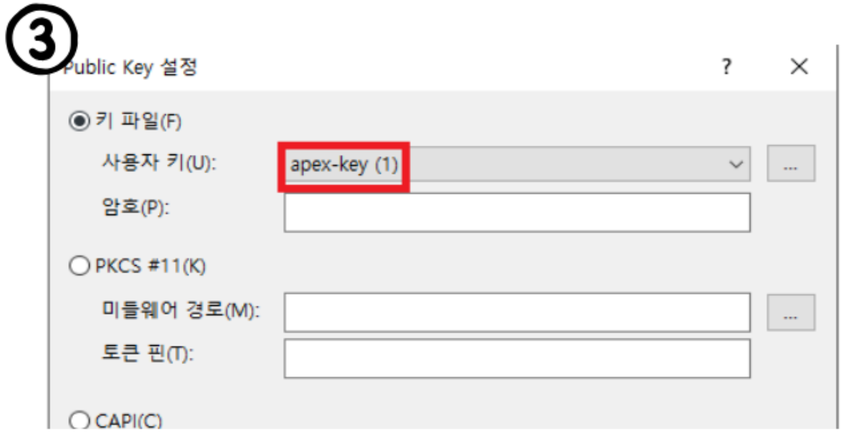
확인

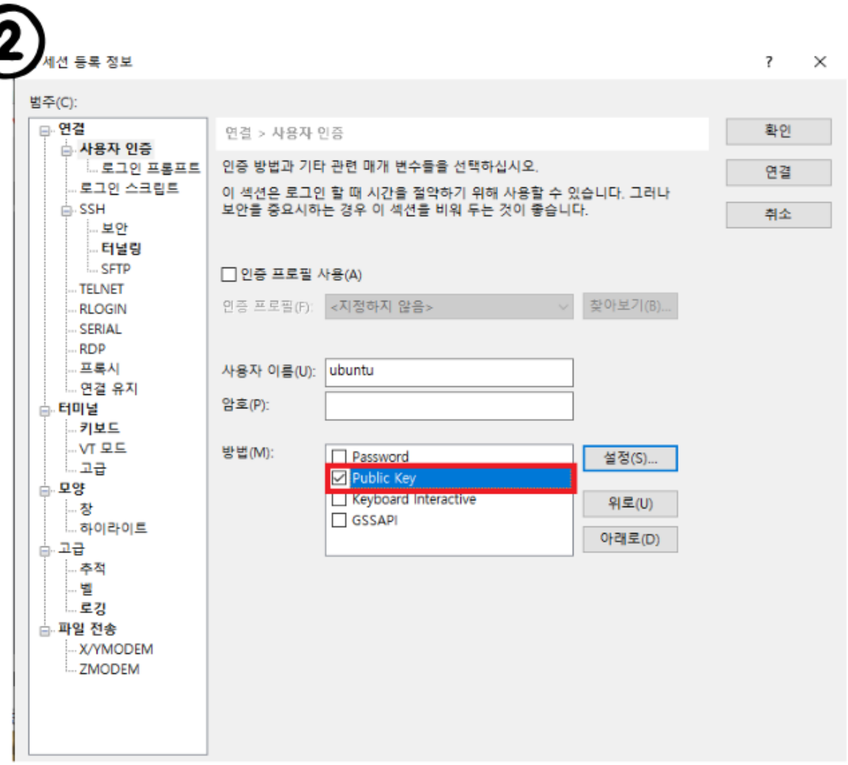
4. 프로젝트 수행결과 | Bastion Server 생성

1-3. XSHELL 연결

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







4. 프로젝트 수행결과 | Bastion Server 환경구성

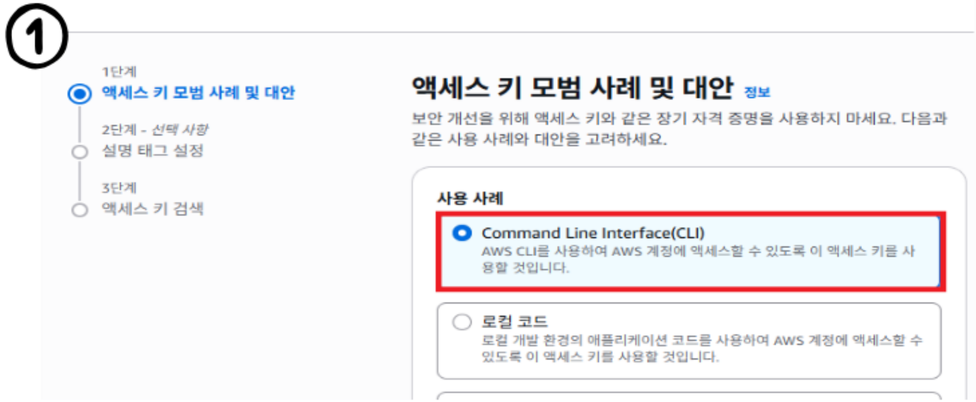
2-1. iam 사용자 생성

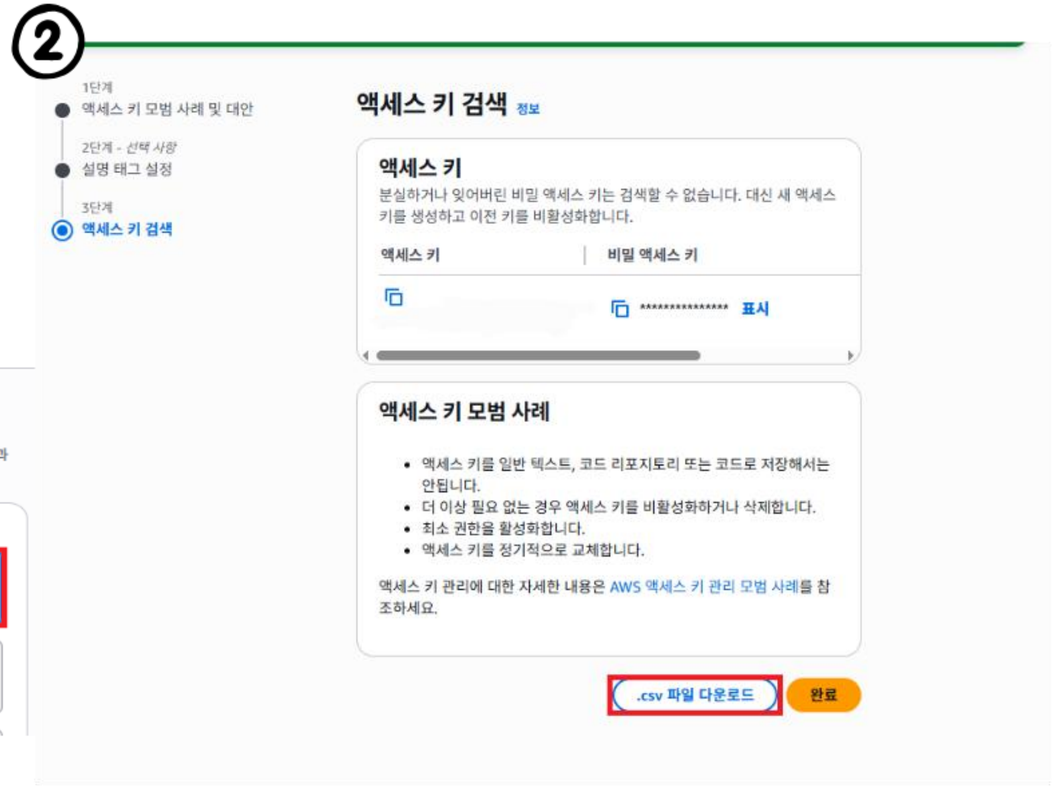
- 사용자 세부 정보 지정
  - 이름 : apex-eks-user
- 권한 설정
  - 권한 옵션
    - 직접 정책 연결
  - 정책 검색
    - AdministratorAcess 체크
- 검토 및 생성

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
  - 태그 미설정
- 검토 및 생성
  - .csv 파일 다운로드
- 생성 확인





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/1.31.0/2024-0
9-12/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 p
ort?
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/down
load/eksctl_$PLATFORM.tar.gz"
ubuntu@ip-172-31-7-236:~$ curl -sL "https://github.com/eksctl-io/eksctl/releases/latest/downl
oad/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, 디폴로이먼트 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:192.168.96.0/19
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)

작업 | 로드 밸런서 생성

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 --
licas=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-
                                PORT(S)          AGE
kubernetes   ClusterIP      10.100.0.1       <none>
                                443/TCP          14m
webtest      LoadBalancer   10.100.171.182   afabfd132
ortheast-2.elb.amazonaws.com   80:31424/TCP     8s
ubuntu@ip-172-31-7-236:~$
```

AWSLoadBalancerControllerIAMPolicy 정보

편집 | 삭제

정책 세부 정보

유형
고객 관리형

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

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

ARN
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
  - AWSLoadBalnacerControllerIAMPolicy
  - documnet 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:~$
```

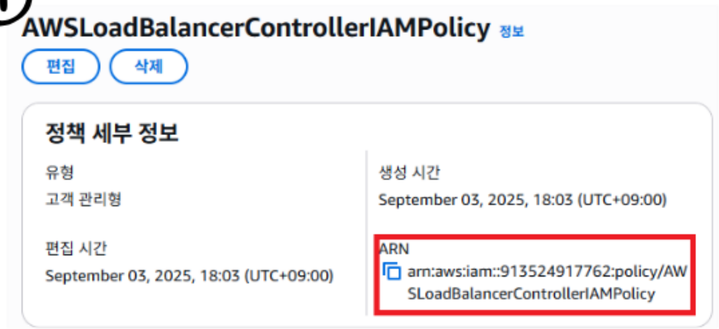
③
```
ubuntu@ip-172-31-7-236:~$ aws iam create-policy \
 --policy-name AWSLoadBalancerControllerIAMPolicy \
 --policy-document file://iam_policy.json
```

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



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

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/AmazonEKSLoad
BalancerControllerRole
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-con
troller \
  -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-samlep-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
```

```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 apex-service.yaml

```
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 서비스 생성 후 확인

```
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
                                                              NOMINATED NODE   READINESS GATES
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>           <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>           <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>           <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 2048.yaml
  - 이름 : deployment-2048
  - 포트 : 80
- 서비스 생성
  - 이름 : service-2048
  - 타입 : NodePort

①

```
ubuntu@ip-172-31-7-236:~$ vi 2048_full.yaml
```

```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
```

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

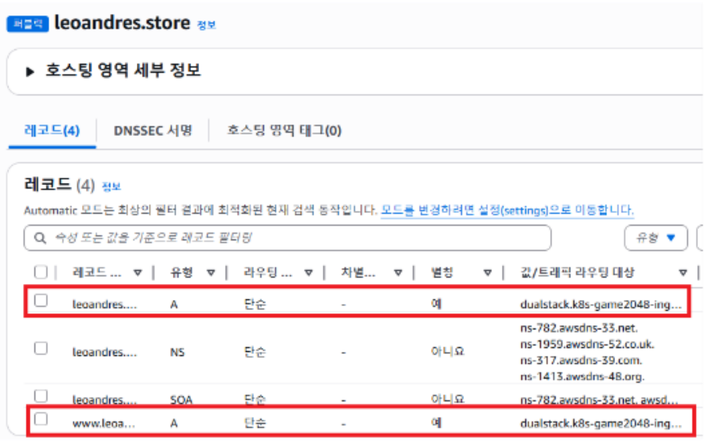
```
ubuntu@ip-172-31-7-236:~$ kubectl get ingress -n game-2048
NAME           CLASS   HOSTS   ADDRESS   PORTS   AGE
ingress-2048   alb     *                 80      2m10s
```

```yaml
---
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 mor
e...)
Annotations:  alb.ingress.kubernetes.io/scheme: internet-facing
              alb.ingress.kubernetes.io/target-type: ip
```

③



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

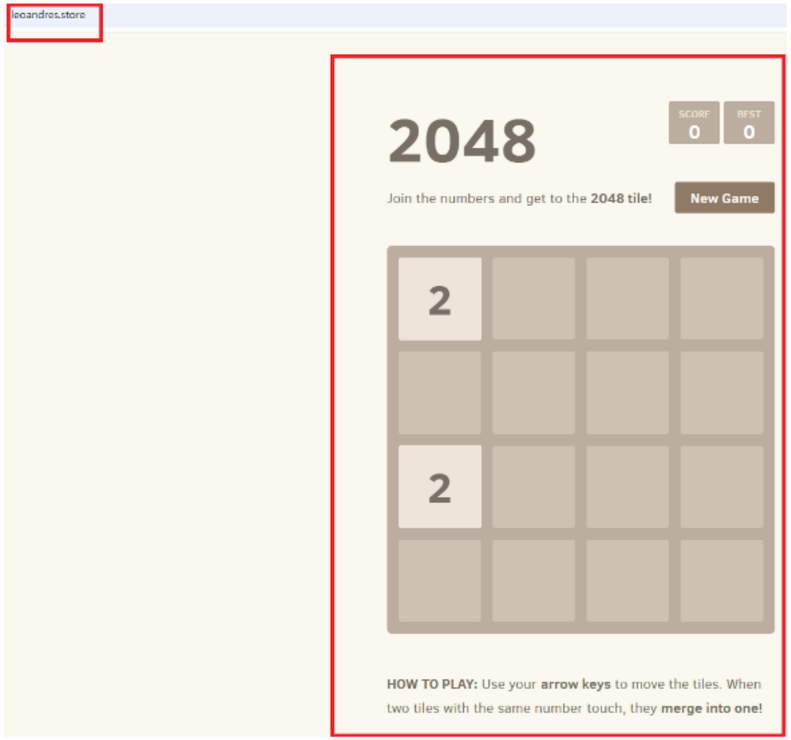
①

```
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.c
om
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

6. 기타 활동 자료

APEX

감사합니다.