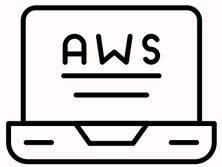
AWS EKS 프로젝트

BSFAN | 김태경
김효은
박종승
윤재영

목차

01 프로젝트 개요



프로젝트 선정 배경 / 기획의도

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

프로젝트 내용

- EKS 클러스터 구축을 위해 클러스터 설계 및 생성
- NLB & ALB 구현을 위한 설계 및 생성

활용 장비 및 재료

- AWS Console
- Xshell 8

프로젝트 구조

- 계획 〉시스템 구성 및 구축〉최종 결과물

02 프로젝트 수행 결과 | Bastion Server 생성

1-1. 인스턴스 생성 (1)

인스턴스
- 이름 : bsfan-bastion
- AMI : Ubuntu Server 22.04 LTS
- 인스턴스 유형 : t2.micro

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

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

bsfan-bastion

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.

수천 개의 애플리케이션 및 OS 이미지를 포함하는 전체 카탈로그 검색

Quick Start

Amazon Linux | macOS | Ubuntu | Windows | Red Hat | SUSE Linux

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

설명

Ubuntu Server 22.04 LTS (HVM),EBS General Purpose (SSD) Volume Type. Support available from Canonical (http://www.ubuntu.com/cloud/services).

2

키 페어 생성

키 페어 이름

키 페어를 사용하면 인스턴스에 안전하게 연결할 수 있습니다.

bsfan-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

02 프로젝트 수행 결과 | Bastion Server 생성

1-2. 인스턴스 생성 (2)

네트워크 설정
- VPC : 기본값
- 보안 그룹 생성
  - 이름 : bsfan-bastion-sg
  - 유형 : ssh (위치무관)

생성 후 확인

1

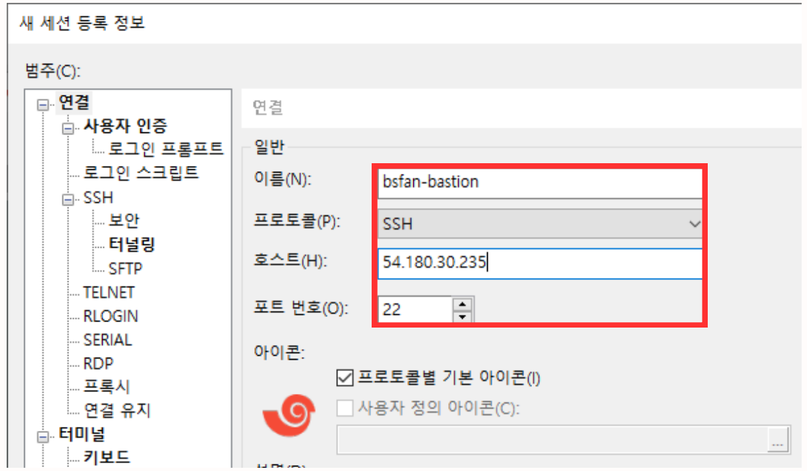
02 프로젝트 수행 결과 | Bastion Server 생성

1-3. Xshell 연결

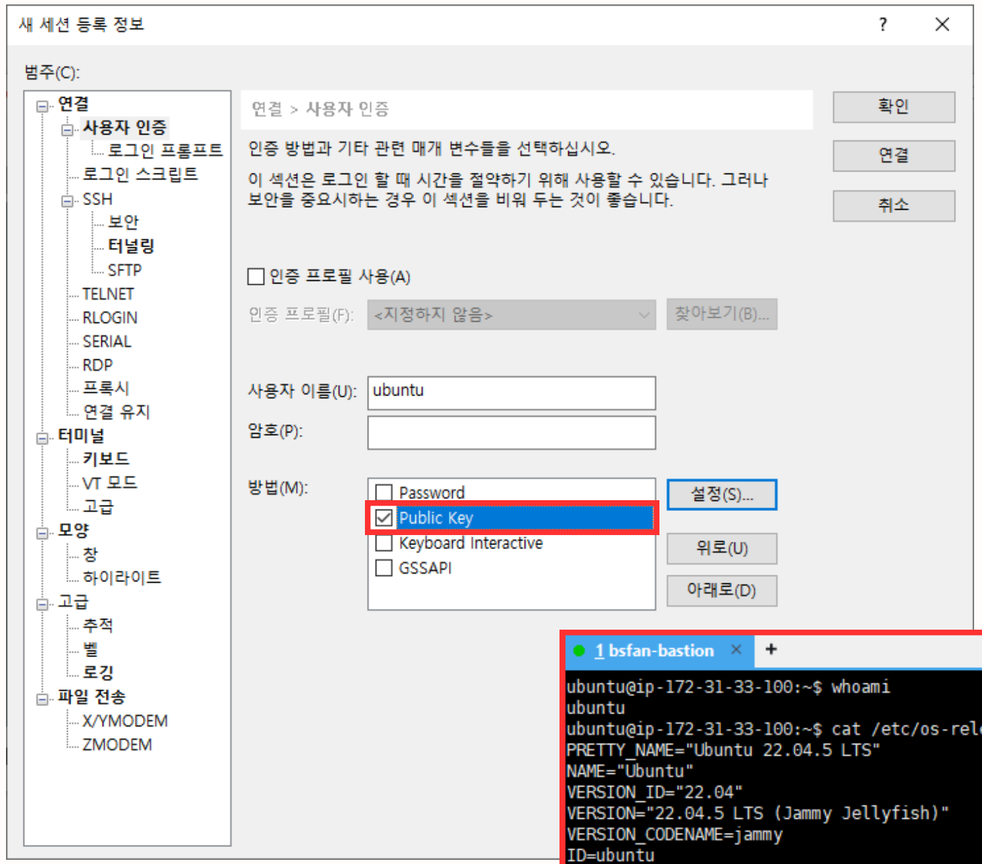
연결
- 이름 : bsfan-bastion
- 호스트 : 54.180.30.235 (퍼블릭 IPv4 주소)
- 포트 번호 : 22

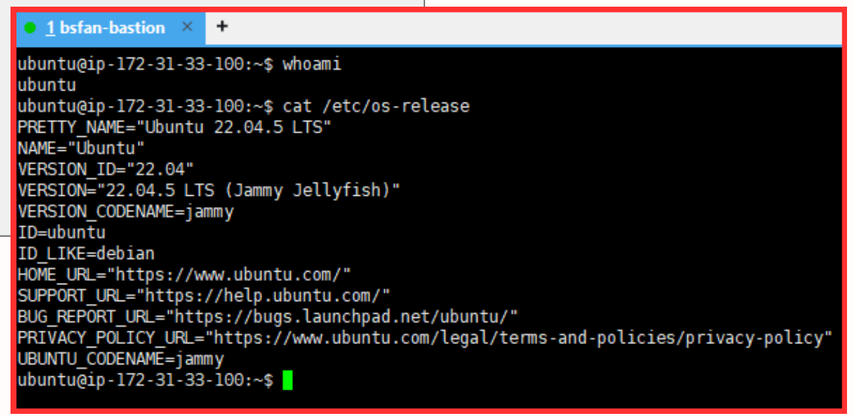
사용자 인증
- Public Key 사용
  - bsfan-key 가져오기

1



2





02 프로젝트 수행 결과 | Bastion Server 환경 구성

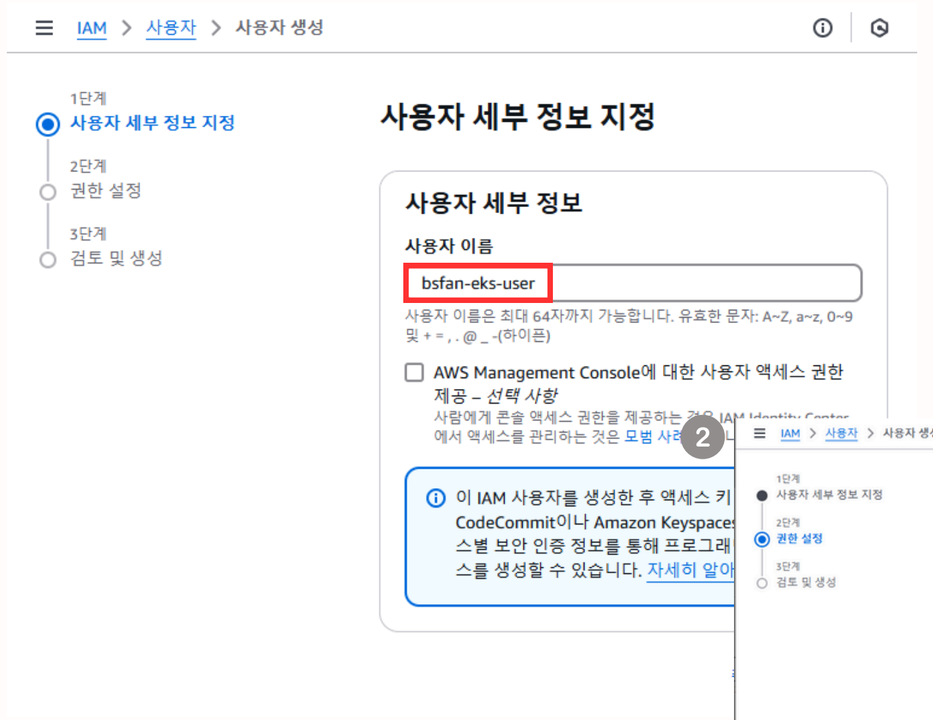
2-1. IAM 사용자 생성

사용자 세부 정보 지정
- 이름 : bsfan-eks-user

권한 설정
- 권한 옵션
  - 직접 정책 연결
- 정책 검색
  - AdministratorAccess 체크

검토 및 생성

1



2

02 프로젝트 수행 결과 | Bastion Server 환경 구성

2-2. 액세스 키 생성

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

설명 태그 설정
- 태그 미 설정

검토 및 생성
- .csv 파일 다운로드

생성 확인

02 프로젝트 수행 결과 | Bastion Server 환경 구성

2-3. 관리 도구 설치

1. AWS CLI 설치
2. 관리시스템에 AWS 계정 등록 및 확인
3. k8s 관리 도구인 kubectl 설치(최신버전)
4. eskctl 설치 및 확인

1
```
ubuntu@ip-172-31-33-100:~$ curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
unzip awscliv2.zip
```

2
```
ubuntu@ip-172-31-33-100:~$ aws configure
AWS Access Key ID [None]: AKIASFUIROQADMCMJGHH
AWS Secret Access Key [None]: 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
Default region name [None]: ap-northeast-2
Default output format [None]: json
ubuntu@ip-172-31-33-100:~$ aws sts get-caller-identity
{
    "UserId": "AIDASFUIROQAL3NJ57WEM",
    "Account": "149536470016",
    "Arn": "arn:aws:iam::149536470016:user/bsfan-eks-user"
}
```

3
```
ubuntu@ip-172-31-33-100:~$ 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  5726k      0  0:00:09  0:00:09 --:--:-- 6894k
ubuntu@ip-172-31-33-100:~$ ls
aws  awscliv2.zip  kubectl
ubuntu@ip-172-31-33-100:~$ chmod +x ./kubectl
ubuntu@ip-172-31-33-100:~$ mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$HOME/bin:$PATH
ubuntu@ip-172-31-33-100:~$ echo 'export PATH=$HOME/bin:$PATH' >> ~/.bashrc
```

4
```
ubuntu@ip-172-31-33-100:~$ ARCH=amd64
ubuntu@ip-172-31-33-100:~$ PLATFORM=$(uname -s)_$ARCH
ubuntu@ip-172-31-33-100:~$ curl -sLO "https://github.com/eksctl-io/eksctl/releases/latest/download/eksctl_$PLATFORM.tar.gz"
ubuntu@ip-172-31-33-100:~$ curl -sL "https://github.com/eksctl-io/eksctl/releases/latest/download/eksctl_checksums.txt" | grep $PLATFORM |
 sha256sum --check
eksctl_Linux_amd64.tar.gz: OK
ubuntu@ip-172-31-33-100:~$ tar -xzf eksctl_$PLATFORM.tar.gz -C /tmp && rm eksctl_$PLATFORM.tar.gz
ubuntu@ip-172-31-33-100:~$ sudo mv /tmp/eksctl /usr/local/bin
```

5
```
ubuntu@ip-172-31-33-100:~$ eksctl version
0.203.0
```

02 프로젝트 수행 결과 | EKS Cluster 생성

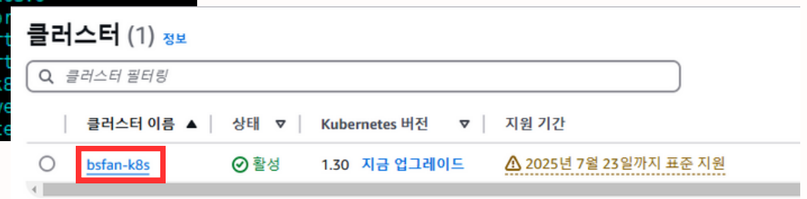
3-1. EKS Cluster 생성

EKS 클러스터 생성 후 확인
- 이름 : bsfan-k8s
- 리전 : ap-northeast-2
- 노드 그룹 : bsfan-k8s-ng
- 노드 : 2

테스트용 서비스, 디플로이먼트 생성 후 확인
- 이름 : webtest
- 타입 : 로드밸런서

1

```
ubuntu@ip-172-31-33-100:~$ eksctl create cluster \
    --name bsfan-k8s \
    --region ap-northeast-2 \
    --with-oidc \
    --nodegroup-name bsfan-k8s-ng \
    --zones ap-northeast-2a,ap-northeast-2c \
    --nodes 2
    --node-type t3.medium \
    --node-volume-size=20 \
    --managed
2025-02-11 02:57:50 [ℹ]  eksctl version 0.203.0
2025-02-11 02:57:50 [ℹ]  using region ap-nor
2025-02-11 02:57:50 [ℹ]  subnets for ap-nort
2025-02-11 02:57:50 [ℹ]  subnets for ap-nort
2025-02-11 02:57:50 [ℹ]  nodegroup "bsfan-k8
2025-02-11 02:57:50 [ℹ]  using Kubernetes ve
2025-02-11 02:57:50 [ℹ]  creating EKS cluste
```



2

```
ubuntu@ip-172-31-33-100:~$ kubectl create deployment webtest --image=nginx:1.14 --port=80 --replicas=3
deployment.apps/webtest created
ubuntu@ip-172-31-33-100:~$ kubectl expose deployment webtest --port=80 --type=LoadBalancer
service/webtest exposed
ubuntu@ip-172-31-33-100:~$ kubectl get sv webtest
error: the server doesn't have a resource type "sv"
ubuntu@ip-172-31-33-100:~$ kubectl get svc webtest
NAME      TYPE           CLUSTER-IP       EXTERNAL-IP
E
webtest   LoadBalancer   10
s
```

3

```
ubuntu@ip-172-31-33-100:~$ kubectl delete svc webtest
kubservice "webtest" deleted
eubuntu@ip-172-31-33-100:~$ kubectl delete deployments.apps webtest
deployment.apps "webtest" deleted
ubuntu@ip-172-31-33-100:~$
```

02 프로젝트 수행 결과 | Load Balancer Controller 생성

4-1. Helm 설치 및 정책 생성

스크립트로 Helm 설치

- curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3

정책 설치

- curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.11.0/docs/install/iam_policy.json

설치된 정책으로 IAM 정책 생성

- AWSLoadBalancerControllerIAMPolicy
- document file : iam_policy.json

1

```
ubuntu@ip-172-31-33-100:~$ 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
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
ubuntu@ip-172-31-33-100:~$
```

2

```
ubuntu@ip-172-31-33-100:~$ curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.11.0/docs/install/iam_policy.json
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  8759  100  8759    0     0  19924      0 --:--:-- --:--:-- --:--:-- 19952
```

3

```
ubuntu@ip-172-31-33-100:~$ aws iam create-policy \
    --policy-name AWSLoadBalancerControllerIAMPolicy \
    --policy-document file://iam_policy.json
{
    "Policy": {
        "PolicyName": "AWSLoadBalancerControllerIAMPolicy",
        "PolicyId": "ANPASFUIROQANCNXPD3YC",
        "Arn": "arn:aws:iam::149536470016:policy/AWSLoadBalancerControllerIAMPolicy",
        "Path": "/",
        "DefaultVersionId": "v1",
        "AttachmentCount": 0,
        "PermissionsBoundaryUsageCount": 0,
        "IsAttachable": true,
        "CreateDate": "2025-02-11T03:44:20+00:00",
        "UpdateDate": "2025-02-11T03:44:20+00:00"
    }
}
```

02 프로젝트 수행 결과 | Load Balancer Controller 생성

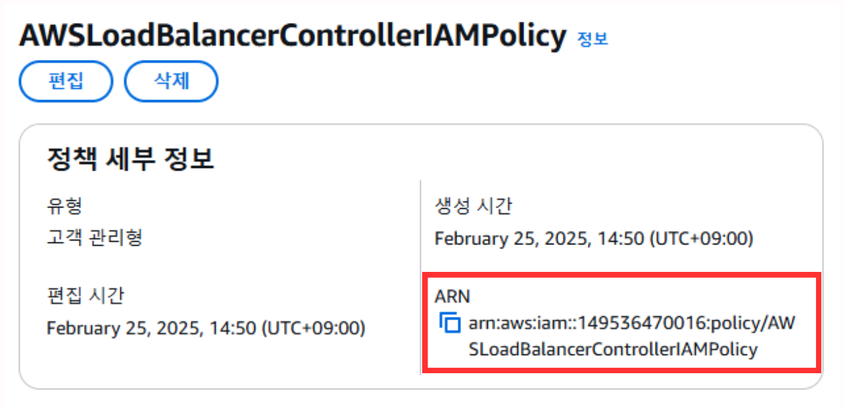
4-2. IAM 서비스 어카운트 생성

ARN 확인
(IAM 〉정책 〉AWSLoadBalancerControllerIAMPolicy)

서비스 어카운트
- 이름 : aws-load-balancer-controller
- 네임스페이스 : kube-system
- 클러스터 : bsfan-k8s

서비스 어카운트 생성 확인

1



2

```
ubuntu@ip-172-31-33-100:~$ eksctl create iamserviceaccount \
  --cluster=bsfan-k8s \
  --namespace=kube-system \
  --name=aws-load-balancer-controller \
  --role-name AmazonEKSLoadBalancerControllerRole \
  --attach-policy-arn=arn:aws:iam::149536470016:policy/AWSLoadBalancer
  --approve
2025-02-11 04:43:06 [ℹ]  1 iamserviceaccount (kube-system/aws-load-bala
2025-02-11 04:43:06 [!]  serviceaccounts that exist in Kubernetes will
2025-02-11 04:43:06 [ℹ]  1 task: {
    2 sequential sub-tasks: {
        create IAM role for serviceaccount "kube-system/aws-load-balanc
        create serviceaccount "kube-system/aws-load-balancer-controlle
    } }2025-02-11 04:43:06 [ℹ]  building iamserviceaccount stack "eksct
ontroller"
2025-02-11 04:43:06 [ℹ]  deploying stack "eksctl-bsfan-k8s-addon-iamse
2025-02-11 04:43:06 [ℹ]  waiting for CloudFormation stack "eksctl-bsfan
er"
2025-02-11 04:43:36 [ℹ]  waiting for CloudFormation stack "eksctl-bsfan
er"
```

3

```
ubuntu@ip-172-31-33-100:~$ 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::149536470016:role/AmazonEKSLoadBalancerControlle
Image pull secrets:  <none>
Mountable secrets:   <none>
Tokens:              <none>
Events:              <none>
```

02 프로젝트 수행 결과 | Load Balancer Controller 생성

4-3. Load Balancer 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-172-31-33-100:~$ helm repo add eks https://aws.github.io/eks-charts
"eks" has been added to your repositories
ubuntu@ip-172-31-33-100:~$ 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!⎈
```

2
```
ubuntu@ip-172-31-33-100:~$ helm install aws-load-balancer-controller eks/aws-load-balancer-controller
  -n kube-system \
  --set clusterName=bsfan-k8s \
  --set serviceAccount.create=false \
  --set serviceAccount.name=aws-load-balancer-controller
NAME: aws-load-balancer-controller
LAST DEPLOYED: Tue Feb 11 04:49:40 2025
NAMESPACE: kube-system
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
AWS Load Balancer controller installed!
```

3
```
NAMESPACE     NAME                                           READY   STATUS
kube-system   aws-load-balancer-controller-79f4b7dc5-b44s2   1/1     Running
kube-system   aws-load-balancer-controller-79f4b7dc5-wg9sm   1/1     Running
```

02 프로젝트 수행 결과 | Load Balancer 배포-NLB

5-1. 샘플 애플리케이션 배포(1)

네임스페이스 생성
- 이름 : nlb-bsfan-app
- 경로 : namespace/nlb-bsfan-app

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

1

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nlb-bsfan-app
  namespace: nlb-bsfan-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-172-31-33-100:~$ kubectl apply -f bsfan-deployment.yaml
deployment.apps/nlb-bsfan-app created
ubuntu@ip-172-31-33-100:~$ vi bsfan-deployment.yaml
ubuntu@ip-172-31-33-100:~$ kubectl get ns
kubeNAME              STATUS   AGE
default           Active   113m
kube-node-lease   Active   113m
kube-public       Active   113m
kube-system       Active   113m
nlb-bsfan-app     Active   3m14s
ubuntu@ip-172-31-33-100:~$ kubectl get  deployments.apps -n nlb-bsfan-app
NAME            READY   UP-TO-DATE   AVAILABLE   AGE
nlb-bsfan-app   3/3     3            3           32s
```

02 프로젝트 수행 결과 | Load Balancer 배포-NLB

5-2. 샘플 애플리케이션 배포(2)

서비스 생성

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

서비스 생성 후 확인

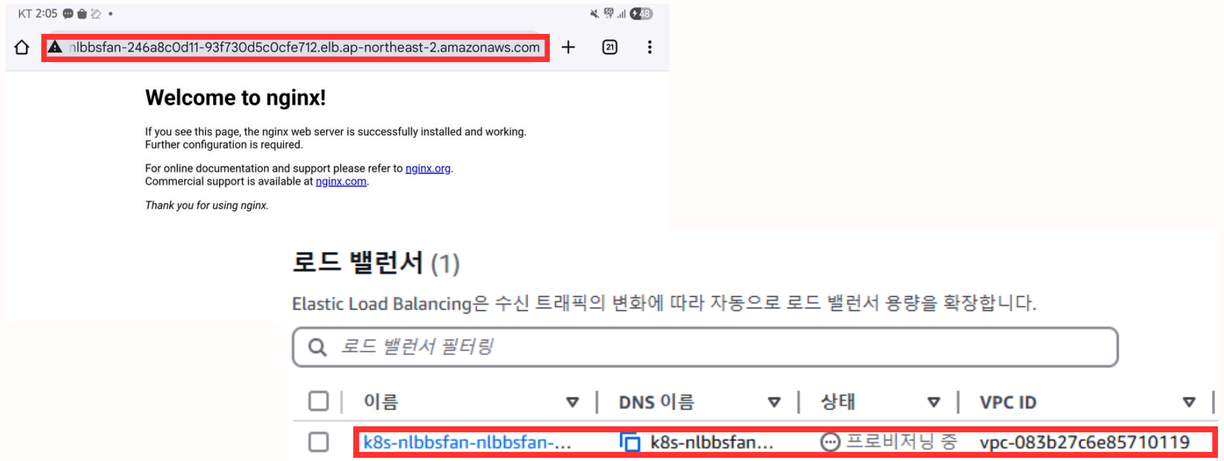
1

```yaml
apiVersion: v1
kind: Service
metadata:
  name: nlb-bsfan-service
  namespace: nlb-bsfan-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-172-31-33-100:~$ kubectl apply -f bsfan-service.yaml
service/nlb-bsfan-service created
ubuntu@ip-172-31-33-100:~$ kubectl get svc -n nlb-bsfan-app nlb-bsfan-service
NAME                TYPE           CLUSTER-IP      EXTERNAL-IP
RT(S)        AGE
nlb-bsfan-service   LoadBalancer   10.100.127.38   k8s-nlbbsfan-nlbbsfan-246a
:31220/TCP   30s
```

3



02 프로젝트 수행 결과 | Load Balancer 배포-ALB

6-1. 샘플 애플리케이션 배포(1)

네임스페이스 생성
- 이름 : bsfan-game-2048

디플로이먼트 생성
- 야물 파일 생성, 적용
  - 이름 : bsfan game 2048.yaml
- 이름 : bsfan-deployment-2048
- 포트 : 80

서비스 생성
- 이름 : bsfan-service-2048
- 타입 : NodePort

1

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

2

```
ubuntu@ip-172-31-33-100:~$ kubectl apply -f bsfan-game-2048.yaml
namespace/bsfan-game-2048 created
deployment.apps/bsfan-deployment-2048 created
service/bsfan-service-2048 created
ingress.networking.k8s.io/bsfan-ingress-2048 created
ubuntu@ip-172-31-33-100:~$ kubectl get pod -n bsfan-game-2048
NAME                                    READY   STATUS    RESTARTS   AGE
bsfan-deployment-2048-855c6b4647-4ns8f   1/1     Running   0          22s
bsfan-deployment-2048-855c6b4647-4skrf   1/1     Running   0          22s
bsfan-deployment-2048-855c6b4647-9d8mq   1/1     Running   0          22s
bsfan-deployment-2048-855c6b4647-kz879   1/1     Running   0          22s
bsfan-deployment-2048-855c6b4647-wnpl2   1/1     Running   0          22s
ubuntu@ip-172-31-33-100:~$ kubectl get svc -n bsfan-game-2048
NAME                 TYPE       CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
bsfan-service-2048   NodePort   10.100.63.67    <none>        80:30194/TCP   38s
```

02 프로젝트 수행 결과 | Load Balancer 배포-ALB

6-2. 샘플 애플리케이션 배포(2)

ingress 생성

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

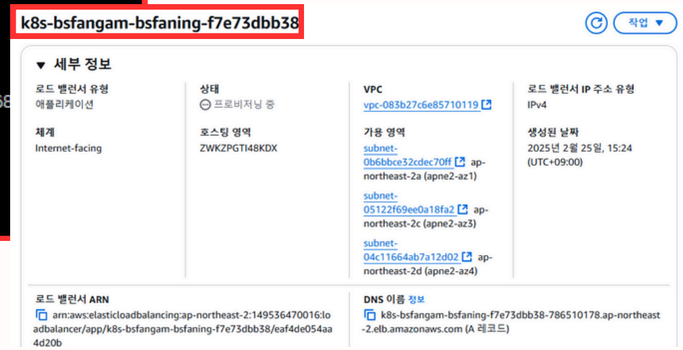
ingress 및 접속 주소 확인

1

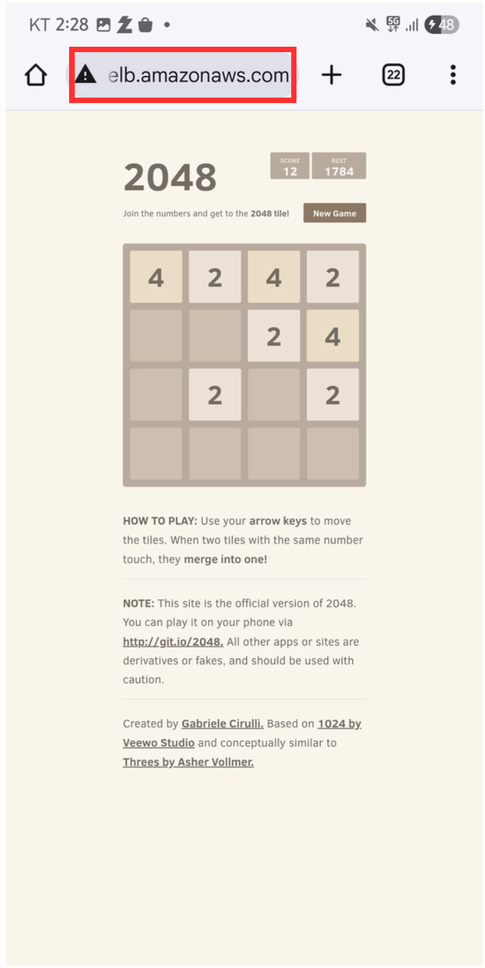
```yaml
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  namespace: bsfan-game-2048
  name: bsfan-ingress-2048
  annotations:
    alb.ingress.kubernetes.io/scheme: internet-facing
    alb.ingress.kubernetes.io/target-type: ip
spec:
  ingressClassName: alb
  rules:
    - http:
        paths: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

2

```
ubuntu@ip-172-31-33-100:~$ kubectl describe -n bsfan-game-2048 ingress bsfan-ingress-2048
Name:             bsfan-ingress-2048
Labels:           <none>
Namespace:        bsfan-game-2048
Address:          k8s-bsfangam-bsfaning-f7e73dbb38-2039286029.ap-northeast-2.elb.amazonaws.com
Ingress Class:    alb
Default backend:  <default>
Rules:
  Host        Path  Backends
  ----        ----  --------
  *
              /   bsfan-service-2048:80 (192.168.5.244:80,192.168.9.64:80,192.168
Annotations:  alb.ingress.kubernetes.io/scheme: internet-facing
              alb.ingress.kubernetes.io/target-type: ip
Events:
  Type    Reason                  Age    From     Message
  ----    ------                  ----   ----     -------
  Normal  SuccessfullyReconciled  2m10s  ingress  Successfully reconciled
```



3



감사합니다