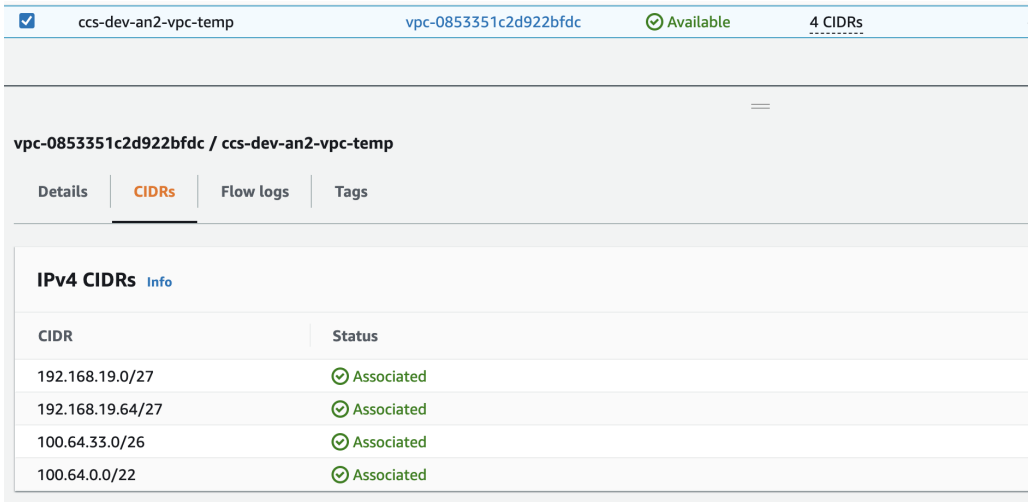
SKT CSS DEV(임시계정) Cluster 구축 가이드

Editing

VPC 정보



public 서브넷 : 192.168.19.0/27

중복서브넷 : 100.64.0.0/22

유니크 서브넷 : 100.64.33.0/26

DB 서브넷 : 192.168.19.64/27

Cluster 생성시 필요한 서브넷

ccs-dev-an2-snet-a-public (subnet-0005d26509951e2ea)

ccs-dev-an2-snet-c-public (subnet-04cded3e7c677f5c6)

ccs-dev-an2-snet-a-priv-eks-node(subnet-07897f60a71b240b0)

ccs-dev-an2-snet-c-priv-eks-node(subnet-001c54c3fcbfb4d54)

Cluster 생성 후 추가될 중복 서브넷

ccs-dev-an2-snet-a-priv-eks-pod(subnet-0240942c723cd1261)

ccs-dev-an2-snet-c-priv-eks-pod(subnet-0710db2c12b75b094)

중복 서브넷 태그 추가

```
Key : kubernetes.io/cluster/ccs-dev-an2-eks-cluster
Value : shared
```

Cluster 구성

eksctl create cluster --config-file createCluster.yaml --profile skt-css-dev

Cluster 생성

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
  name: ccs-dev-an2-eks-cluster
  region: ap-northeast-2
  version: "1.19"
vpc:
  subnets:
    private:
      ap-northeast-2a:
        id: subnet-07897f60a71b240b0
      ap-northeast-2c:
        id: subnet-001c54c3fcbfb4d54
    public:
      ap-northeast-2a:
        id: subnet-0005d26509951e2ea
      ap-northeast-2c:
        id: subnet-04cded3e7c677f5c6
  clusterEndpoints:
    privateAccess: true
    publicAccess: true
```

KubeContext GET

```
aws eks update-kubeconfig --profile skt-css-dev --region ap-northeast-2 --name ccs-dev-an2-eks-cluster
--alias ccs-dev-an2-eks-cluster
```

CNI 플러그인에 대한 사용자 지정 네트워크 구성 활성화

```
kubectl set env daemonset aws-node -n kube-system AWS_VPC_K8S_CNI_CUSTOM_NETWORK_CFG=true
```

작업자 노드를 식별하기 위한 ENIConfig 레이블 추가

```
kubectl set env daemonset aws-node -n kube-system ENI_CONFIG_LABEL_DEF=failure-domain.beta.kubernetes.io/zone
```

ENIConfig 사용자 지정 리소스 정의 설치

```
( |custom-network-test:default) kshong@kshongui-MacBookPro  ~/aws/skt_landing  cat << EOF | kubectl
apply -f -
apiVersion: apiextensions.k8s.io/v1beta1
kind: CustomResourceDefinition
metadata:
  name: eniconfigs.crd.k8s.amazonaws.com
spec:
  scope: Cluster
  group: crd.k8s.amazonaws.com
  version: v1alpha1
  names:
    plural: eniconfigs
    singular: eniconfig
    kind: ENIConfig
EOF
customresourcedefinition.apiextensions.k8s.io/eniconfigs.crd.k8s.amazonaws.com configured
```

모든 서브넷 및 가용 영역에 대해 ENIConfig 사용자 지정 리소스 및 secutiry 그룹 생성

```
cat <<EOF | kubectl apply -f -

apiVersion: crd.k8s.amazonaws.com/v1alpha1

kind: ENIConfig

metadata:

 name: ap-northeast-2a

spec:

  securityGroups:

    - sg-04a928fea05eea5b0

  subnet: subnet-0240942c723cd1261

EOF
```

```
cat <<EOF | kubectl apply -f -

apiVersion: crd.k8s.amazonaws.com/v1alpha1

kind: ENIConfig

metadata:

 name: ap-northeast-2c

spec:

  securityGroups:

    - sg-04a928fea05eea5b0

  subnet: subnet-0710db2c12b75b094

EOF
```

요청 온 노드그룹 정보(황규용 M)

ccs-dev-an2-eks-service-wrk-2-16

2대

30GB

r5.xlarge

노드 그룹 구성

eksctl create nodegroup --config-file skt_css_dev_worker_node.yaml --profile skt-css-dev

worker 노드그룹 생성

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
  name: ccs-dev-an2-eks-cluster
  region: ap-northeast-2
managedNodeGroups:
  - name: ccs-dev-an2-eks-service-wrk-2-16
    instanceType: r5.large
    availabilityZones:
      - ap-northeast-2a
      - ap-northeast-2c
    desiredCapacity: 2
    minSize: 2
    maxSize: 2
    volumeSize: 30
    ssh:
      allow: true
      publicKeyName: ccs-dev-an2-eks-key
    labels:
       role: worker
    privateNetworking: true
    tags:
      nodegroup-role: worker
    iam:
      withAddonPolicies:
        autoScaler: true
        externalDNS: true
        imageBuilder: true
        appMesh: true
        appMeshPreview: true
        albIngress: true
```

Custom Network 적용여부 확인

NAME READY STATUS RESTARTS AGE IP NODE NOMINATED NODE READINESS GATES

coredns-78fb67b999-hcf6h 1/1 Running 0 30s 100.64.0.69 ip-100-64-33-20.ap-northeast-2.compute.internal <none> <none>
coredns-78fb67b999-n6vp2 1/1 Running 0 19s 100.64.3.101 ip-100-64-33-44.ap-northeast-2.compute.internal <none> <none>

Create EFS

AWS Elastic File System을 생성합니다.

- `PROFILE_NAME` : 프로파일 이름
- `REGION` : 지역. 예) ap-northeast-2
- `CLUSTER_NAME` : Create EKS Cluster 에서 생성한 클러스터 이름
- `VPC_ID` : Create VPC and Subnets 에서 생성한 VPC ID 또는 Create EKS Cluster 에서 생성한 VPC ID

```
aws ec2 create-security-group \
--profile {PROFILE_NAME} \
--region {REGION} \
--vpc-id {VPC_ID} \
--group-name {CLUSTER_NAME}-efs-sg \
--description "EFS Security group for {CLUSTER_NAME} cluster."
```

예제)

```
aws ec2 create-security-group \
--profile skt-css-dev \
--region ap-northeast-2 \
--vpc-id vpc-0853351c2d922bfdc \
--group-name ccs-dev-an2-sg-cluster-efs \
--description "EFS Security group for ccs-dev-an2-eks-cluster cluster."
{
    "GroupId": "sg-08d2ca69a9e8ffdf0"
}
```

인바운드 엑세스 권한 승인

- `PROFILE_NAME` : 프로파일 이름
- `REGION` : 지역. 예) ap-northeast-2
- `SECURITY_GROUP_ID` : 위에서 생성한 Security Group ID

```
aws ec2 authorize-security-group-ingress \
--profile {PROFILE_NAME} \
--region {REGION}
--group-id {SECURITY_GROUP_ID} \
--protocol tcp \
--port 2049 \
--cidr 0.0.0.0/0
```

예제)

```
aws ec2 authorize-security-group-ingress \
--profile skt-css-dev \
--region ap-northeast-2 \
--group-id sg-08d2ca69a9e8ffdf0 \
--protocol tcp \
--port 2049 \
--cidr 0.0.0.0/0
```

EFS 생성

- `PROFILE_NAME` : 프로파일 이름
- `REGION` : 지역. 예) ap-northeast-2
- `CLUSTER_NAME` : Create EKS Cluster 에서 생성한 클러스터 이름

```
aws efs create-file-system \
--profile {PROFILE_NAME} \
--region {REGION} \
--tags Key=Name,Value={CLUSTER_NAME}
```

예제)

```
aws efs create-file-system \
--profile skt-css-dev \
--region ap-northeast-2 \
--tags Key=Name,Value=ccs-dev-an2-efs

{
"OwnerId": "580008524617",
"CreationToken": "9c961b0a-98b6-429e-9d8c-b7bdb8b28969",
"FileSystemId": "fs-2174f341",
"FileSystemArn": "arn:aws:elasticfilesystem:ap-northeast-2:580008524617:file-system/fs-2174f341",
"CreationTime": "2021-07-20T18:19:45+09:00",
"LifeCycleState": "creating",
"Name": "ccs-dev-an2-efs",
"NumberOfMountTargets": 0,
"SizeInBytes": {
"Value": 0,
"ValueInIA": 0,
"ValueInStandard": 0
},
"PerformanceMode": "generalPurpose",
"Encrypted": false,
"ThroughputMode": "bursting",
"Tags": [
{
"Key": "Name",
"Value": "ccs-dev-an2-efs"
}
]
}
```

Create a Mount Target

EKS 클러스터에 Join 하는 모든 작업자 노드가 배치되는 각 Subnet 에 대하여 탑재 대상을 생성합니다.

- `PROFILE_NAME` : 프로파일 이름
- `REGION` : 지역
- `FILE_SYSTEM_ID` : 탑재 대상을 생성하려는 파일 시스템의 ID
- `SUBNET_ID` : EC2 인스턴스를 시작한 VPC 서브넷 ID
- `SECURITY_GROUP_ID` : 이전 단계에서 탑재 대상에 대해 생성한 보안 그룹의 ID

```
$ aws efs create-mount-target \
--profile {PROFILE_NAME} \
--region {REGION} \
--file-system-id {FILE_SYSTEM_ID} \
--subnet-id {PRIVATE_SUBNET_ID} \
--security-group {SECURITY_GROUP_ID}
```

예제)

```
$ aws efs create-mount-target \
  --profile skt-css-dev \
  --region ap-northeast-2 \
  --file-system-id fs-2174f341 \
  --subnet-id subnet-07897f60a71b240b0 \
  --security-group sg-08d2ca69a9e8ffdf0

{
"OwnerId": "580008524617",
"MountTargetId": "fsmt-891769e8",
"FileSystemId": "fs-2174f341",
"SubnetId": "subnet-07897f60a71b240b0",
"LifeCycleState": "creating",
"IpAddress": "100.64.33.30",
"NetworkInterfaceId": "eni-08cfc2d86985fb20e",
"AvailabilityZoneId": "apne2-az1",
"AvailabilityZoneName": "ap-northeast-2a",
"VpcId": "vpc-0853351c2d922bfdc"
}


$ aws efs create-mount-target \
  --profile zcp-dev \
  --region ap-northeast-2 \
  --file-system-id fs-2174f341 \
  --subnet-id subnet-001c54c3fcbfb4d54 \
  --security-group sg-08d2ca69a9e8ffdf0

{
"OwnerId": "580008524617",
"MountTargetId": "fsmt-60146a01",
"FileSystemId": "fs-2174f341",
"SubnetId": "subnet-001c54c3fcbfb4d54",
"LifeCycleState": "creating",
"IpAddress": "100.64.33.53",
"NetworkInterfaceId": "eni-06eec03fc7af38989",
"AvailabilityZoneId": "apne2-az3",
"AvailabilityZoneName": "ap-northeast-2c",
"VpcId": "vpc-0853351c2d922bfdc"
}
```

Install EFS Provisioner

> The efs-provisioner allows you to mount EFS storage as PersistentVolumes in kubernetes. It consists of a container that has access to an AWS EFS resource. The container reads a configmap which contains the EFS filesystem ID, the AWS region and the name you want to use for your efs-provisioner. This name will be used later when you create a storage class.
> 보다 자세한 설명은 efs-provisioner를 참고하세요.

efs-provisioner가 mount 되지 않을때

Create EFS 에서 다음의 항목들을 제대로 구성하였는지 확인합니다.

1. eks node group의 security group과 efs의 security group이 같게 지정되어야 합니다.
2. 해당 security group의 inbound rule이 NFS 2049 포트에 대해 모든 리소스에 대하여 열려 있는지 확인합니다.

Attach policy to node instance role

아래 문제를 해결하기 위하여 Node Instance Role 에 AmazonElasticFileSystemReadOnlyAccess 정책을 추가합니다.

couldn't confirm that the EFS file system exists: AccessDeniedException: User: arn:aws:sts::042956191338:assumed-role/eksctl-cloudzcp-mcm-dev-nodegroup-NodeInstanceRole-P42QCK9LS1C9/i-0ca2c2b792976de3a is not authorized to perform: elasticfilesystem:DescribeFileSystems on the specified resource

- CLI
- Management Console

- `PROFILE_NAME` : 프로파일 이름

- `NODE_INSTANCE_ROLE` : Node Group 생성 시에 만들어진 Node Instance Role

```
aws iam attach-role-policy \
--profile {PROFILE_NAME} \
--role-name {NODE_INSTANCE_ROLE} \
--policy-arn arn:aws:iam::aws:policy/AmazonElasticFileSystemFullAccess
```

예제)

```
aws iam attach-role-policy \
--profile skt-css-dev \
--role-name eksctl-ccs-dev-an2-eks-cluster-no-NodeInstanceRole-1XDQHNT01PYHP \
--policy-arn arn:aws:iam::aws:policy/AmazonElasticFileSystemFullAccess
```

Install EFS provisioner with helm

EFS Provisioner를 Helm을 이용하여 설치합니다.

- `FILE_SYSTEM_ID` : Create EFS 에서 생성한 FileSystem ID
- `REGION` : 지역. 예) ap-northeast-2

```
helm install stable/efs-provisioner \
--name zcp-efs-provisioner \
--namespace kube-system \
--set efsProvisioner.efsFileSystemId={FILE_SYSTEM_ID} \
--set efsProvisioner.awsRegion={REGION} \
--set efsProvisioner.path=/zcp \
--set efsProvisioner.provisionerName=cloudzcp.io/aws-efs \
--set efsProvisioner.storageClass.name=efs-zcp
```

예제)

```
helm3 install css-efs-provisioner stable/efs-provisioner \
--namespace kube-system \
--set efsProvisioner.efsFileSystemId=fs-2174f341 \
--set efsProvisioner.awsRegion=ap-northeast-2 \
--set efsProvisioner.path=/ccs \
--set efsProvisioner.provisionerName=css.io/aws-efs \
--set efsProvisioner.storageClass.name=ccs-dev-an2-eks-sc-efs-pvc
```

retain storage class 생성

```
kubectl patch sc ccs-dev-an2-eks-sc-efs-pvc -p '{"metadata":{"name":"ccs-dev-an2-eks-sc-efs-pvc-retain"
},"reclaimPolicy": "Retain"}' --dry-run -o yaml | kubectl create -f -
```

Install ebs csi driver

Policy 파일 다운로드

```
curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-ebs-csi-driver/v0.10.0/docs/example-iam-p
olicy.json
```

Policy 생성

- `PROFILE_NAME` : 프로파일 이름

```
aws iam create-policy --policy-name Amazon_EBS_CSI_Driver \
--policy-document file://example-iam-policy.json --profile skt-css-dev
```

```
{
"Policy": {
"PolicyName": "Amazon_EBS_CSI_Driver",
"PolicyId": "ANPAYOCZTXNE6RI42CZK7",
"Arn": "arn:aws:iam::580008524617:policy/Amazon_EBS_CSI_Driver",
"Path": "/",
"DefaultVersionId": "v1",
"AttachmentCount": 0,
"PermissionsBoundaryUsageCount": 0,
"IsAttachable": true,
"CreateDate": "2021-07-20T11:45:17+00:00",
"UpdateDate": "2021-07-20T11:45:17+00:00"
}
}
```

```
kubectl get cm -n kube-system aws-auth -o yaml
```

```
apiVersion: v1
data:
mapRoles: |
- groups:
- system:bootstrappers
- system:nodes
rolearn: arn:aws:iam::580008524617:role/eksctl-ccs-dev-an2-eks-cluster-no-NodeInstanceRole-1XDQHNT01PYHP
username: system:node:{{EC2PrivateDNSName}}
kind: ConfigMap
metadata:
creationTimestamp: "2021-07-20T08:35:02Z"
managedFields:
- apiVersion: v1
fieldsType: FieldsV1
fieldsV1:
f:data:
.: {}
f:mapRoles: {}
manager: vpcLambda
operation: Update
time: "2021-07-20T08:35:02Z"
name: aws-auth
namespace: kube-system
resourceVersion: "6723"
selfLink: /api/v1/namespaces/kube-system/configmaps/aws-auth
uid: c9b364ce-6df8-4677-9bbd-12e81543f761
```

Node Role 에 Policy 를 연결합니다.

```
aws iam attach-role-policy \
--profile {PROFILE_NAME} \
--role-name eksctl-{CLUSTER_NAME}-nodegroup-NodeInstanceRole-{XXX} \
--policy-arn arn:aws:iam::{CUSTOMER_ACCOUT_ID}:policy/Amazon_EBS_CSI_Driver
```

예제)

```
aws iam attach-role-policy \
--profile skt-css-dev \
--role-name eksctl-ccs-dev-an2-eks-cluster-no-NodeInstanceRole-1XDQHNT01PYHP \
--policy-arn arn:aws:iam::580008524617:policy/Amazon_EBS_CSI_Driver
```

helm 을 이용하여 `aws-ebs-csi-driver` 를 설치합니다.

```
helm3 repo add aws-ebs-csi-driver https://kubernetes-sigs.github.io/aws-ebs-csi-driver

helm3 upgrade -i aws-ebs-csi-driver aws-ebs-csi-driver/aws-ebs-csi-driver ₩
-f values-css.yaml ₩
--namespace kube-system ₩
--set serviceAccount.controller.create=false ₩
--set enableVolumeScheduling=true ₩
--set enableVolumeResizing=true ₩
--set enableVolumeSnapshot=true ₩
```

StorageClass 를 확인합니다.

```
kubectl get sc

ebs-gp3 ebs.csi.aws.com Delete WaitForFirstConsumer true 67s
ebs-gp3-retain ebs.csi.aws.com Retain WaitForFirstConsumer true 67s
```

NLB 구성(nginx ingress controller)

- Helm 차트 : https://github.com/helm/charts/tree/master/stable/nginx-ingress
- Helm 차트 (v1.36.3) : https://github.com/helm/charts/tree/c21cf6826872c1e85a6a2651088d5166f54d3e5f/stable/nginx-ingress

k8s version >= 1.9

k8s 버전이 1.9이상의 클러스터는 k8s.gcr.io/ingress-nginx/controller:v0.45.0으로 직접 설치합니다.

아래 명령어를 통해 helm3 repo를 추가합니다.

```
helm3 repo add ingress-nginx https://kubernetes.github.io/ingress-nginx
```

helm배포를 위한 values.yaml 파일을 작성합니다.

conmmon

1. values-zcp.yaml 을 작성합니다.

```
## nginx configuration
## Ref:
https://github.com/kubernetes/ingress-nginx/blob/master/docs/user-gui
de/nginx-configuration/index.md
##

## Overrides for generated resource names
# See templates/_helpers.tpl
# nameOverride:
# fullnameOverride:

controller:
  name: controller
  image:
    repository: k8s.gcr.io/ingress-nginx/controller
    tag: "v0.45.0"
    digest:
sha256:c4390c53f348c3bd4e60a5dd6a11c35799ae78c49388090140b9d72ccede17
```

```
    pullPolicy: IfNotPresent
    # www-data -> uid 101
    runAsUser: 101
    allowPrivilegeEscalation: true

  # Configures the ports the nginx-controller listens on
  containerPort:
    http: 80
    https: 443

  # Will add custom configuration options to Nginx
https://kubernetes.github.io/ingress-nginx/user-guide/nginx-configura
tion/configmap/
  config:
    enable-access-log-for-default-backend: "true"
    server-tokens: 'false'
    log-format-escape-json: 'true'
    log-format-upstream: '{"time_date": "$time_iso8601",
                           "client": "$remote_addr",
                           "host": "$http_host",
                           "scheme": "$scheme",
                           "request_method": "$request_method",
                           "request_uri": "$uri",
                           "request_id": "$request_id",
                           "status": $status,
                           "upstream_addr": "$upstream_addr",
                           "upstream_status": $upstream_status,
                           "request_time": $request_time,
                           "upstream_response_time":
$upstream_response_time,
                           "upstream_connect_time":
$upstream_connect_time,
                           "upstream_header_time":
$upstream_header_time}'
    compute-full-forwarded-for: "true"
    use-forwarded-headers: "true"

  ## Annotations to be added to the controller config configuration
configmap
  ##
  configAnnotations: {}

  # Will add custom headers before sending traffic to backends
according to
https://github.com/kubernetes/ingress-nginx/tree/master/docs/examples
/customization/custom-headers
  proxySetHeaders: {}

  # Will add custom headers before sending response traffic to the
client according to:
https://kubernetes.github.io/ingress-nginx/user-guide/nginx-configura
tion/configmap/#add-headers
```

```
  addHeaders: {}

  # Optionally customize the pod dnsConfig.
  dnsConfig: {}

  # Optionally change this to ClusterFirstWithHostNet in case you have
'hostNetwork: true'.
  # By default, while using host network, name resolution uses the
host's DNS. If you wish nginx-controller
  # to keep resolving names inside the k8s network, use
ClusterFirstWithHostNet.
  dnsPolicy: ClusterFirst

  # Bare-metal considerations via the host network
https://kubernetes.github.io/ingress-nginx/deploy/baremetal/#via-the-
host-network
  # Ingress status was blank because there is no Service exposing the
NGINX Ingress controller in a configuration using the host network,
the default --publish-service flag used in standard cloud setups does
not apply
  reportNodeInternalIp: false

  # Required for use with CNI based kubernetes installations (such as
ones set up by kubeadm),
  # since CNI and hostport don't mix yet. Can be deprecated once
https://github.com/kubernetes/kubernetes/issues/23920
  # is merged
  hostNetwork: false

  ## Use host ports 80 and 443
  ## Disabled by default
  ##
  hostPort:
    enabled: false
    ports:
      http: 80
      https: 443

  ## Election ID to use for status update
  ##
  electionID: ingress-controller-leader

  ## Name of the ingress class to route through this controller
  ##
  ingressClass: nginx

  # labels to add to the pod container metadata
  podLabels: {}
  #  key: value

  ## Security Context policies for controller pods
  ##
  podSecurityContext: {}
```

```
  ## See
https://kubernetes.io/docs/tasks/administer-cluster/sysctl-cluster/
for
  ## notes on enabling and using sysctls
  ###
  sysctls: {}
  # sysctls:
  #   "net.core.somaxconn": "8192"

  ## Allows customization of the source of the IP address or FQDN to
report
  ## in the ingress status field. By default, it reads the information
provided
  ## by the service. If disable, the status field reports the IP
address of the
  ## node or nodes where an ingress controller pod is running.
  publishService:
    enabled: true
    ## Allows overriding of the publish service to bind to
    ## Must be <namespace>/<service_name>
    ##
    pathOverride: ""

  ## Limit the scope of the controller
  ##
  scope:
    enabled: false
    namespace: ""   # defaults to .Release.Namespace

  ## Allows customization of the configmap / nginx-configmap namespace
  ##
  configMapNamespace: ""   # defaults to .Release.Namespace

  ## Allows customization of the tcp-services-configmap
  ##
  tcp:
    configMapNamespace: ""   # defaults to .Release.Namespace
    ## Annotations to be added to the tcp config configmap
    annotations: {}

  ## Allows customization of the udp-services-configmap
  ##
  udp:
    configMapNamespace: ""   # defaults to .Release.Namespace
    ## Annotations to be added to the udp config configmap
    annotations: {}

  # Maxmind license key to download GeoLite2 Databases
  #
https://blog.maxmind.com/2019/12/18/significant-changes-to-accessing-
and-using-geolite2-databases
  maxmindLicenseKey: ""
```

```
  ## Additional command line arguments to pass to
nginx-ingress-controller
  ## E.g. to specify the default SSL certificate you can use
  ## extraArgs:
  ##   default-ssl-certificate: "<namespace>/<secret_name>"
  extraArgs: {}

  ## Additional environment variables to set
  extraEnvs: []
  # extraEnvs:
  #   - name: FOO
  #     valueFrom:
  #       secretKeyRef:
  #         key: FOO
  #         name: secret-resource

  ## DaemonSet or Deployment
  ##
  kind: Deployment

  ## Annotations to be added to the controller Deployment or DaemonSet
  ##
  annotations: {}
  #  keel.sh/pollSchedule: "@every 60m"

  ## Labels to be added to the controller Deployment or DaemonSet
  ##
  labels: {}
  #  keel.sh/policy: patch
  #  keel.sh/trigger: poll


  # The update strategy to apply to the Deployment or DaemonSet
  ##
  updateStrategy: {}
  #  rollingUpdate:
  #    maxUnavailable: 1
  #  type: RollingUpdate

  # minReadySeconds to avoid killing pods before we are ready
  ##
  minReadySeconds: 0


  ## Node tolerations for server scheduling to nodes with taints
  ## Ref:
https://kubernetes.io/docs/concepts/configuration/assign-pod-node/
  ##
  tolerations:
  - key: "dedicated"
    value: "edge"
  #  - key: "key"
```

```
  #    operator: "Equal|Exists"
  #    value: "value"
  #    effect: "NoSchedule|PreferNoSchedule|NoExecute(1.6 only)"

  ## Affinity and anti-affinity
  ## Ref:
https://kubernetes.io/docs/concepts/configuration/assign-pod-node/#af
finity-and-anti-affinity
  ##
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: dedicated
            operator: In
            values:
            - edge
    podAntiAffinity:
      preferredDuringSchedulingIgnoredDuringExecution:
      - weight: 100
        podAffinityTerm:
          labelSelector:
            matchExpressions:
            - key: app.kubernetes.io/name
              operator: In
              values:
              - ingress-nginx
            - key: app.kubernetes.io/component
              operator: In
              values:
              - controller
          topologyKey: kubernetes.io/hostname

    # # An example of preferred pod anti-affinity, weight is in the
range 1-100
    # podAntiAffinity:
    #   preferredDuringSchedulingIgnoredDuringExecution:
    #   - weight: 100
    #     podAffinityTerm:
    #       labelSelector:
    #         matchExpressions:
    #         - key: app.kubernetes.io/name
    #           operator: In
    #           values:
    #           - ingress-nginx
    #         - key: app.kubernetes.io/instance
    #           operator: In
    #           values:
    #           - ingress-nginx
    #         - key: app.kubernetes.io/component
    #           operator: In
    #           values:
```

```
    #           - controller
    #       topologyKey: kubernetes.io/hostname

    # # An example of required pod anti-affinity
    # podAntiAffinity:
    #   requiredDuringSchedulingIgnoredDuringExecution:
    #   - labelSelector:
    #       matchExpressions:
    #       - key: app.kubernetes.io/name
    #         operator: In
    #         values:
    #         - ingress-nginx
    #       - key: app.kubernetes.io/instance
    #         operator: In
    #         values:
    #         - ingress-nginx
    #       - key: app.kubernetes.io/component
    #         operator: In
    #         values:
    #         - controller
    #     topologyKey: "kubernetes.io/hostname"

  ## Topology spread constraints rely on node labels to identify the
topology domain(s) that each Node is in.
  ## Ref:
https://kubernetes.io/docs/concepts/workloads/pods/pod-topology-sprea
d-constraints/
  ##
  topologySpreadConstraints: []
    # - maxSkew: 1
    #   topologyKey: failure-domain.beta.kubernetes.io/zone
    #   whenUnsatisfiable: DoNotSchedule
    #   labelSelector:
    #     matchLabels:
    #       app.kubernetes.io/instance: ingress-nginx-internal

  ## terminationGracePeriodSeconds
  ## wait up to five minutes for the drain of connections
  ##
  terminationGracePeriodSeconds: 300

  ## Node labels for controller pod assignment
  ## Ref: https://kubernetes.io/docs/user-guide/node-selection/
  ##
  nodeSelector:
    kubernetes.io/os: linux

  ## Liveness and readiness probe values
  ## Ref:
https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle/#con
tainer-probes
  ##
  livenessProbe:
```

```
    failureThreshold: 5
    initialDelaySeconds: 10
    periodSeconds: 10
    successThreshold: 1
    timeoutSeconds: 1
    port: 10254
  readinessProbe:
    failureThreshold: 3
    initialDelaySeconds: 10
    periodSeconds: 10
    successThreshold: 1
    timeoutSeconds: 1
    port: 10254

  # Path of the health check endpoint. All requests received on the
port defined by
  # the healthz-port parameter are forwarded internally to this path.
  healthCheckPath: "/healthz"

  ## Annotations to be added to controller pods
  ##
  podAnnotations: {}

  replicaCount: 1

  minAvailable: 1

  # Define requests resources to avoid probe issues due to CPU
utilization in busy nodes
  # ref:
https://github.com/kubernetes/ingress-nginx/issues/4735#issuecomment-
551204903
  # Ideally, there should be no limits.
  #
https://engineering.indeedblog.com/blog/2019/12/cpu-throttling-regres
sion-fix/
  resources:
  #  limits:
  #    cpu: 100m
  #    memory: 90Mi
    requests:
      cpu: 100m
      memory: 90Mi

  # Mutually exclusive with keda autoscaling
  autoscaling:
    enabled: false
    minReplicas: 1
    maxReplicas: 11
    targetCPUUtilizationPercentage: 50
    targetMemoryUtilizationPercentage: 50

  autoscalingTemplate: []
```

```
  # Custom or additional autoscaling metrics
  # ref:
https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autos
cale/#support-for-custom-metrics
  # - type: Pods
  #   pods:
  #     metric:
  #       name: nginx_ingress_controller_nginx_process_requests_total
  #     target:
  #       type: AverageValue
  #       averageValue: 10000m

  # Mutually exclusive with hpa autoscaling
  keda:
    apiVersion: "keda.sh/v1alpha1"
  # apiVersion changes with keda 1.x vs 2.x
  # 2.x = keda.sh/v1alpha1
  # 1.x = keda.k8s.io/v1alpha1
    enabled: false
    minReplicas: 1
    maxReplicas: 11
    pollingInterval: 30
    cooldownPeriod: 300
    restoreToOriginalReplicaCount: false
    scaledObject:
      annotations: {}
      # Custom annotations for ScaledObject resource
      #  annotations:
      # key: value
    triggers: []
 #     - type: prometheus
 #       metadata:
 #         serverAddress: http://<prometheus-host>:9090
 #         metricName: http_requests_total
 #         threshold: '100'
 #         query:
sum(rate(http_requests_total{deployment="my-deployment"}[2m]))

    behavior: {}
 #     scaleDown:
 #       stabilizationWindowSeconds: 300
 #       policies:
 #       - type: Pods
 #         value: 1
 #         periodSeconds: 180
 #     scaleUp:
 #       stabilizationWindowSeconds: 300
 #       policies:
 #       - type: Pods
 #         value: 2
 #         periodSeconds: 60

  ## Enable mimalloc as a drop-in replacement for malloc.
```

```
  ## ref: https://github.com/microsoft/mimalloc
  ##
  enableMimalloc: true

  ## Override NGINX template
  customTemplate:
    configMapName: ""
    configMapKey: ""

  service:
    enabled: true

    annotations: {}
    labels: {}
    # clusterIP: ""

    ## List of IP addresses at which the controller services are
available
    ## Ref:
https://kubernetes.io/docs/user-guide/services/#external-ips
    ##
    externalIPs: []

    # loadBalancerIP: ""
    loadBalancerSourceRanges: []

    enableHttp: true
    enableHttps: true

    ## Set external traffic policy to: "Local" to preserve source IP
on
    ## providers supporting it
    ## Ref:
https://kubernetes.io/docs/tutorials/services/source-ip/#source-ip-fo
r-services-with-typeloadbalancer
    # externalTrafficPolicy: ""

    # Must be either "None" or "ClientIP" if set. Kubernetes will
default to "None".
    # Ref:
https://kubernetes.io/docs/concepts/services-networking/service/#virt
ual-ips-and-service-proxies
    # sessionAffinity: ""

    # specifies the health check node port (numeric port number) for
the service. If healthCheckNodePort isn’t specified,
    # the service controller allocates a port from your cluster’s
NodePort range.
    # Ref:
https://kubernetes.io/docs/tasks/access-application-cluster/create-ex
ternal-load-balancer/#preserving-the-client-source-ip
    # healthCheckNodePort: 0
```

```
    ports:
      http: 80
      https: 443

    targetPorts:
      http: http
      https: https

    type: LoadBalancer

    # type: NodePort
    # nodePorts:
    #   http: 32080
    #   https: 32443
    #   tcp:
    #     8080: 32808
    nodePorts:
      http: ""
      https: ""
      tcp: {}
      udp: {}

    ## Enables an additional internal load balancer (besides the
external one).
    ## Annotations are mandatory for the load balancer to come up.
Varies with the cloud service.
    internal:
      enabled: false
      annotations: {}

      # loadBalancerIP: ""

      ## Restrict access For LoadBalancer service. Defaults to
0.0.0.0/0.
      loadBalancerSourceRanges: []

      ## Set external traffic policy to: "Local" to preserve source IP
on
      ## providers supporting it
      ## Ref:
https://kubernetes.io/docs/tutorials/services/source-ip/#source-ip-fo
r-services-with-typeloadbalancer
      # externalTrafficPolicy: ""

  extraContainers: []
  ## Additional containers to be added to the controller pod.
  ## See
https://github.com/lemonldap-ng-controller/lemonldap-ng-controller as
example.
  #  - name: my-sidecar
  #    image: nginx:latest
  #  - name: lemonldap-ng-controller
  #    image: lemonldapng/lemonldap-ng-controller:0.2.0
```

```
  #    args:
  #      - /lemonldap-ng-controller
  #      - --alsologtostderr
  #      - --configmap=$(POD_NAMESPACE)/lemonldap-ng-configuration
  #    env:
  #      - name: POD_NAME
  #        valueFrom:
  #          fieldRef:
  #            fieldPath: metadata.name
  #      - name: POD_NAMESPACE
  #        valueFrom:
  #          fieldRef:
  #            fieldPath: metadata.namespace
  #    volumeMounts:
  #    - name: copy-portal-skins
  #      mountPath: /srv/var/lib/lemonldap-ng/portal/skins

  extraVolumeMounts: []
  ## Additional volumeMounts to the controller main container.
  #  - name: copy-portal-skins
  #   mountPath: /var/lib/lemonldap-ng/portal/skins

  extraVolumes: []
  ## Additional volumes to the controller pod.
  #  - name: copy-portal-skins
  #    emptyDir: {}

  extraInitContainers: []
  ## Containers, which are run before the app containers are started.
  # - name: init-myservice
  #   image: busybox
  #   command: ['sh', '-c', 'until nslookup myservice; do echo waiting
for myservice; sleep 2; done;']

  admissionWebhooks:
    annotations: {}
    enabled: true
    failurePolicy: Fail
    # timeoutSeconds: 10
    port: 8443
    certificate: "/usr/local/certificates/cert"
    key: "/usr/local/certificates/key"
    namespaceSelector: {}
    objectSelector: {}

    service:
      annotations: {}
      # clusterIP: ""
      externalIPs: []
      # loadBalancerIP: ""
      loadBalancerSourceRanges: []
      servicePort: 443
      type: ClusterIP
```

```
    patch:
      enabled: true
      image:
        repository: docker.io/jettech/kube-webhook-certgen
        tag: v1.5.1
        pullPolicy: IfNotPresent
      ## Provide a priority class name to the webhook patching job
      ##
      priorityClassName: ""
      podAnnotations: {}
      nodeSelector: {}
      tolerations: []
      runAsUser: 2000

  metrics:
    port: 10254
    # if this port is changed, change healthz-port: in extraArgs:
accordingly
    enabled: true

    service:
      annotations:
        prometheus.io/scrape: "true"
        prometheus.io/port: "10254"

      # clusterIP: ""

      ## List of IP addresses at which the stats-exporter service is
available
      ## Ref:
https://kubernetes.io/docs/user-guide/services/#external-ips
      ##
      externalIPs: []

      # loadBalancerIP: ""
      loadBalancerSourceRanges: []
      servicePort: 10254
      type: ClusterIP
      # externalTrafficPolicy: ""
      # nodePort: ""

    serviceMonitor:
      enabled: false
      additionalLabels: {}
      namespace: ""
      namespaceSelector: {}
      # Default: scrape .Release.Namespace only
      # To scrape all, use the following:
      # namespaceSelector:
      #   any: true
      scrapeInterval: 30s
      # honorLabels: true
```

```
      targetLabels: []
      metricRelabelings: []

    prometheusRule:
      enabled: false
      additionalLabels: {}
      # namespace: ""
      rules: []
        # # These are just examples rules, please adapt them to your
needs
        # - alert: NGINXConfigFailed
        #   expr:
count(nginx_ingress_controller_config_last_reload_successful == 0) > 0
        #   for: 1s
        #   labels:
        #     severity: critical
        #   annotations:
        #     description: bad ingress config - nginx config test
failed
        #     summary: uninstall the latest ingress changes to allow
config reloads to resume
        # - alert: NGINXCertificateExpiry
        #   expr:
(avg(nginx_ingress_controller_ssl_expire_time_seconds) by (host) -
time()) < 604800
        #   for: 1s
        #   labels:
        #     severity: critical
        #   annotations:
        #     description: ssl certificate(s) will expire in less then
a week
        #     summary: renew expiring certificates to avoid downtime
        # - alert: NGINXTooMany500s
        #   expr: 100 * ( sum(
nginx_ingress_controller_requests{status=~"5.+"} ) /
sum(nginx_ingress_controller_requests) ) > 5
        #   for: 1m
        #   labels:
        #     severity: warning
        #   annotations:
        #     description: Too many 5XXs
        #     summary: More than 5% of all requests returned 5XX, this
requires your attention
        # - alert: NGINXTooMany400s
        #   expr: 100 * ( sum(
nginx_ingress_controller_requests{status=~"4.+"} ) /
sum(nginx_ingress_controller_requests) ) > 5
        #   for: 1m
        #   labels:
        #     severity: warning
        #   annotations:
        #     description: Too many 4XXs
        #     summary: More than 5% of all requests returned 4XX, this
```

```
requires your attention

  ## Improve connection draining when ingress controller pod is
deleted using a lifecycle hook:
  ## With this new hook, we increased the default
terminationGracePeriodSeconds from 30 seconds
  ## to 300, allowing the draining of connections up to five minutes.
  ## If the active connections end before that, the pod will terminate
gracefully at that time.
  ## To effectively take advantage of this feature, the Configmap
feature
  ## worker-shutdown-timeout new value is 240s instead of 10s.
  ##
  lifecycle:
    preStop:
      exec:
        command:
          - /wait-shutdown

  priorityClassName: ""

## Rollback limit
##
revisionHistoryLimit: 10

## Default 404 backend
##
defaultBackend:
  ##
  enabled: true

  name: defaultbackend
  image:
    repository: k8s.gcr.io/defaultbackend-amd64
    tag: "1.5"
    pullPolicy: IfNotPresent
    # nobody user -> uid 65534
    runAsUser: 65534
    runAsNonRoot: true
    readOnlyRootFilesystem: true
    allowPrivilegeEscalation: false

  extraArgs: {}

  serviceAccount:
    create: true
    name: ""
    automountServiceAccountToken: true
  ## Additional environment variables to set for defaultBackend pods
  extraEnvs: []

  port: 8080
```

```
  ## Readiness and liveness probes for default backend
  ## Ref:
https://kubernetes.io/docs/tasks/configure-pod-container/configure-li
veness-readiness-probes/
  ##
  livenessProbe:
    failureThreshold: 3
    initialDelaySeconds: 30
    periodSeconds: 10
    successThreshold: 1
    timeoutSeconds: 5
  readinessProbe:
    failureThreshold: 6
    initialDelaySeconds: 0
    periodSeconds: 5
    successThreshold: 1
    timeoutSeconds: 5

  ## Node tolerations for server scheduling to nodes with taints
  ## Ref:
https://kubernetes.io/docs/concepts/configuration/assign-pod-node/
  ##
  tolerations: []
  #  - key: "key"
  #    operator: "Equal|Exists"
  #    value: "value"
  #    effect: "NoSchedule|PreferNoSchedule|NoExecute(1.6 only)"

  affinity: {}

  ## Security Context policies for controller pods
  ## See
https://kubernetes.io/docs/tasks/administer-cluster/sysctl-cluster/
for
  ## notes on enabling and using sysctls
  ##
  podSecurityContext: {}

  # labels to add to the pod container metadata
  podLabels: {}
  #  key: value

  ## Node labels for default backend pod assignment
  ## Ref: https://kubernetes.io/docs/user-guide/node-selection/
  ##
  nodeSelector: {}

  ## Annotations to be added to default backend pods
  ##
  podAnnotations: {}

  replicaCount: 1
```

```
  minAvailable: 1

  resources: {}
  # limits:
  #   cpu: 10m
  #   memory: 20Mi
  # requests:
  #   cpu: 10m
  #   memory: 20Mi

  extraVolumeMounts: []
  ## Additional volumeMounts to the default backend container.
  #  - name: copy-portal-skins
  #   mountPath: /var/lib/lemonldap-ng/portal/skins

  extraVolumes: []
  ## Additional volumes to the default backend pod.
  #  - name: copy-portal-skins
  #    emptyDir: {}

  autoscaling:
    enabled: false
    minReplicas: 1
    maxReplicas: 2
    targetCPUUtilizationPercentage: 50
    targetMemoryUtilizationPercentage: 50

  service:
    annotations: {}

    # clusterIP: ""

    ## List of IP addresses at which the default backend service is
available
    ## Ref:
https://kubernetes.io/docs/user-guide/services/#external-ips
    ##
    externalIPs: []

    # loadBalancerIP: ""
    loadBalancerSourceRanges: []
    servicePort: 80
    type: ClusterIP

  priorityClassName: ""

## Enable RBAC as per
https://github.com/kubernetes/ingress/tree/master/examples/rbac/nginx
and https://github.com/kubernetes/ingress/issues/266
rbac:
  create: true
  scope: false
```

```
# If true, create & use Pod Security Policy resources
# https://kubernetes.io/docs/concepts/policy/pod-security-policy/
podSecurityPolicy:
  enabled: false

serviceAccount:
  create: true
  name: ""
  automountServiceAccountToken: true

## Optional array of imagePullSecrets containing private registry
credentials
## Ref:
https://kubernetes.io/docs/tasks/configure-pod-container/pull-image-p
rivate-registry/
imagePullSecrets: []
# - name: secretName

# TCP service key:value pairs
# Ref:
https://github.com/kubernetes/contrib/tree/master/ingress/controllers
/nginx/examples/tcp
##
tcp: {}
#  8080: "default/example-tcp-svc:9000"

# UDP service key:value pairs
# Ref:
https://github.com/kubernetes/contrib/tree/master/ingress/controllers
/nginx/examples/udp
##
udp: {}
#  53: "kube-system/kube-dns:53"

# A base64ed Diffie-Hellman parameter
# This can be generated with: openssl dhparam 4096 2> /dev/null |
base64
# Ref:
```

```
https://github.com/krmichel/ingress-nginx/blob/master/docs/examples/c
ustomization/ssl-dh-param
dhParam:
```

helm3 명령어를 통해 배포합니다.

PRIVATE PUBLIC

1. 클러스터에 환경에 맞는 values.yaml 을 다운 받습니다. Private 클러스터인 경우, Internal 방식의 NLB를 사용합니다. Helm을 이용하여 nginx를 설치합니다.

   values.yaml 파일의 image.repository 주소를 주석처리하여 사용합니다.

```
helm3 upgrade -i private-ingress-nginx ingress-nginx/ingress-nginx \
--version 3.27.0 \
--namespace kube-system \
-f values-zcp.yaml \
--set controller.ingressClass=private-nginx \
--set controller.replicaCount=2 \
--set
controller.service.annotations."service\.beta\.kubernetes\.io/aws-loa
d-balancer-type"=nlb \
--set
controller.service.annotations."service\.beta\.kubernetes\.io/aws-loa
d-balancer-internal"=true \
--set
controller.service.annotations."service\.beta\.kubernetes\.io/aws-loa
d-balancer-subnets"="{SUBNET-ID}\,{SUBNET-ID}"
```

   예제)

```
helm3 install private-ingress-nginx ingress-nginx/ingress-nginx
--version 3.27.0 \
--namespace kube-system \
-f values-eks-private.yaml --set
controller.service.annotations."service\.beta\.kubernetes\.io/aws-loa
d-balancer-subnets"="subnet-003ff1e484789081d\,subnet-052ad8b1a8dfac7
7e"
```

Nginx가 잘 배포되었는지 확인합니다.

```
$ kubectl get po,ing,svc -n kube-system | grep nginx
```

혹시 image를 가져오지 못하는 오류가 발생한다면 아래 수정을 통해 이미지를 직접 변경합니다.

```
$ kubectl edit ds private-ingress-nginx -n kube-system
```

```
image: k8s.gcr.io/ingress-nginx/controller:v0.45.0
```

배포를 확인합니다.

ALB 구성(AWS ALB Controller)

아래 링크를 참조하여 구성합니다.

링크: EKS에서의 ALB사용