

## **Project: K8s on EC2 & Deploying via AWS ELB** **(EC2에 K8s 설치 및 ELB와 연동)**

---

**Environment(환경): Ubuntu 20.04(EC2)**

List to Create during Hands-on-Lab([오늘 Hands-on-Lab에서 생성할 목록](#))

- 1. VPC
- 2. Subnet([서브넷](#))
- 3. Internet Gateway([인터넷 게이트웨이](#))
- 4. Route Table([라우팅 테이블](#))
- 5. IAM Roles([IAM 역할](#))
  - Master Node Role
  - Worker Node Role
- 6. Running EC2([EC2 생성](#))
  - K8s Cluster Setup([쿠버네티스 클러스터 설정](#))
    - K8s installation([쿠버네티스 설치](#))
      - Hostname([호스트명](#))
      - Cluster Setup([클러스터 설정](#))
      - Kubeadm Reset([Kubeadm 초기화](#))
      - Flannel CNI Installation([Flannel 네트워크 플러그인 설치](#))
      - Attaching the Worker Node([Worker Node 연동](#))
      - Load Balancer Creation via K8s Service([쿠버네티스 서비스를 통한 AWS 로드밸런서 생성](#))
  - 7. Troubleshooting([문제 해결](#))
    - AWS Load Balancer
      - When no Worker Node is added to the ELB([AWS 로드밸런서가 생성됐지만 Worker Node가 연결안되어 있을 때](#))

---

### **1. VPC**

- Create a VPC with the 10.0.0.0/16 CIDR([10.0.0.0/16 대역대를 사용하는 VPC를 생성](#))



VPCs > Create VPC

## Create VPC

A VPC is an isolated portion of the AWS cloud populated by AWS objects, such as Amazon EC2 instances. You must specify a block larger than /16. You can optionally associate an Amazon-provided IPv6 CIDR block with the VPC.

Name tag  i

IPv4 CIDR block\*  i

IPv6 CIDR block i  
 No IPv6 CIDR Block  
 Amazon provided IPv6 CIDR block

Tenancy  i

\* Required

- Add a tag named **kubernetes.io/cluster/kubernetes** with the **owned** value - it will be used by K8s for AWS resources auto-discovery related to the Kubernetes stack, also it will add such a tag itself during creating new resources(쿠버네티스가 AWS 자원을 자동으로 발견할 수 있도록 태그 및 값을 지정)

태그: **kubernetes.io/cluster/kubernetes**

값: **owned**

추가적으로 새로운 리소스를 생성할 때 해당 태그를 생성)

The screenshot shows the AWS VPC Dashboard with a modal dialog titled 'Add/Edit Tags'. The dialog is used to apply tags to a selected VPC. The VPC listed is 'k8s-cluster-vpc' with VPC ID 'vpc-00881455e7f607dd7'. In the 'Add/Edit Tags' dialog, a single tag is being added: 'kubernetes.io/cluster/kubernetes' with the value 'owned'. There are buttons for 'Create Tag', 'Cancel', and 'Save'. The background shows the VPC dashboard with other VPCs listed.

- Enable DNS hostname(DNS 호스트명 활성화)

VPC Dashboard

Filter by VPC: Select a VPC

Virtual Private Cloud

Your VPCs

- Subnets
- Route Tables
- Internet Gateways
- Egress Only Internet Gateways
- DHCP Options Sets
- Elastic IPs
- Endpoints
- Endpoint Services
- NAT Gateways

Create VPC Actions

Name	VPC ID	State	IPv4 CIDR	IPv6 CIDR
k8s	vpc-00881455e7f607dd7	available	10.0.0.0/16	-

VPC: vpc-00881455e7f607dd7

Tags

Add/Edit Tags

Key Name

Name kubernetes.io/cluster/kubernetes

AWS Services: EC2 RDS S3

[VPCs](#) > Edit DNS hostnames

## Edit DNS hostnames

VPC ID vpc-00881455e7f607dd7

DNS hostnames  enable

\* Required

## 2. Subnet

- Create a new subnet in this VPC(VPC에 새로운 서브넷 생성)

[Subnets](#) > Create subnet

### Create subnet

Specify your subnet's IP address block in CIDR format; for example, 10.0.0.0/24. IPv4 block sizes must be between a /16 netmask and /28 netmask.

Name tag	k8s-cluster-net	<small>i</small>
VPC*	vpc-00881455e7f607dd7	<small>i</small>
VPC CIDRs	CIDR	Status
	10.0.0.0/16	associated
Availability Zone	eu-west-3a	<small>i</small>
IPv4 CIDR block*	10.0.0.0/24	<small>i</small>

\* Required

- Enable Public IPs for EC2 instances which will be placed in this subnet(해당 서브넷에 생성될 EC2 인스턴스들에 대해 공인 IP 부여를 허용)

The screenshot shows the AWS VPC Dashboard. On the left sidebar, under 'Subnets', the 'k8s-cluster-' subnet is selected. In the main content area, a context menu is open over the subnet row. The menu options include: Delete subnet, Create flow log, Modify auto-assign IP settings (which is highlighted), Edit IPv6 CIDRs, Edit network ACL association, Edit route table association, Share subnet, and Add/Edit Tags.

The screenshot shows the 'Modify auto-assign IP settings' dialog. It includes fields for 'Subnet ID' (subnet-05253c2474384b21a) and 'Auto-assign IPv4' (checkbox checked). Below the dialog, a note says 'Enable the auto-assign IP address setting to automatically request a public IPv4 or IPv6 address for an instance launch'. A section labeled '\* Required' is present.

- Add the tag(태그 추가)

The screenshot shows the 'Add/Edit Tags' dialog. It contains a table with two rows of tags: 'Name' with value 'k8s-cluster-net' and 'kubernetes.io/cluster/kubernetes' with value 'owned'. There are 'Create Tag' and 'Save' buttons at the bottom.

### 3. Internet Gateway

- Create an IGW to route traffic from the subnet into the Internet(서브넷에서 인터넷 통신을 받기 위해 인터넷 게이트웨이 생성)



Internet gateways > Create internet gateway

## Create internet gateway

An internet gateway is a virtual router that connects a VPC to the internet. To create a new internet gateway specify the name for the gateway below.

Name tag  ?

\* Required

[Cancel](#) [Create](#)

- For IGW, add the tag as well, just in case(인터넷 게이트웨이에도 태그를 추가)

VPC Dashboard

Virtual Private Cloud

Your VPCs

Subnets

Route Tables

Internet Gateways

Egress Only Internet Gateways

DHCP Options Sets

Elastic IPs

Create internet gateway

Add/Edit Tags

Key	Value
Name	k8s-cluster-igw
kubernetes.io/cluster/kubernetes	owned

Internet gateway: igw-0012fd40bf6839ad2

Description Tags Add/Edit Tags

- Attach this IGW to your VPC(1번 단계에서 생성한 VPC에 인터넷 게이트웨이를 추가)

VPC Dashboard

Virtual Private Cloud

Your VPCs

Subnets

Route Tables

Internet Gateways

Egress Only Internet Gateways

DHCP Options Sets

Elastic IPs

Endpoints

Create internet gateway

Attach to VPC

Detach from VPC

Add/Edit Tags

Key	Value
Name	k8s-cluster-igw
kubernetes.io/cluster/kubernetes	owned

AWS Services: EC2, RDS, S3, VPC, Route 53, CloudFormation, CloudFront, CloudWatch

Internet gateways > Attach to VPC

### Attach to VPC

Attach an internet gateway to a VPC to enable communication with the internet. Specify the VPC you would like to attach below.

VPC\*  ?

▶ AWS Command Line Interface command

\* Required Cancel **Attach**

## 4. Route Table

- Create a routing table(라우팅 테이블 생성)

AWS Services: EC2, RDS, S3, VPC, Route 53, CloudFormation, CloudFront, CloudWatch

Route Tables > Create route table

### Create route table

A route table specifies how packets are forwarded between the subnets within your VPC, the internet, and your VPN connection.

Name tag  ?

VPC\*  C ?

\* Required Cancel **Create**

- Add the tag here(태그를 추가)

VPC Dashboard

Virtual Private Cloud

Your VPCs

Subnets

**Route Tables**

Internet Gateways

Egress Only Internet Gateways

DHCP Options Sets

Elastic IPs

Create route table Actions

Route Table ID: rtb-0f886fe39e6cccf7c

Route Table: rtb-0f886fe39e6cccf7c

Summary Routes Subnet Associations

Add/Edit Tags

Key Value

Name k8s-cluster-rtb

kubernetes.io/cluster/kubernetes owned

Create Tag Cancel Save

- Click on the Routes tab, add a new route to the 0.0.0.0/0 network via the IGW we created above(라우팅 테이블 탭으로 가서 IGW를 통해 0.0.0.0/0으로 오는 트래픽 허용)

Destination	Target	Status	Propagated
10.0.0.0/16	local	active	No
0.0.0.0/0	igw-0012fd40bf6839ad2		No

Add route

\* Required

Cancel Save routes

- Attach this table to the subnet & Edit subnet association(생성하신 서브넷에 테이블 추가 및 서브넷 설정 편집)

Destination	Target	Status
10.0.0.0/16	local	active
0.0.0.0/0	igw-0012fd40bf6839ad2	active

- Choose the subnet you created earlier(조금 전에 생성한 서브넷 선택)

Subnet ID	IPv4 CIDR	Current Route Table
subnet-05253c2474384b21a   k8s-cluster-net	10.0.0.0/24	Main

\* Required

Cancel Save

## 5. IAM Role

- To make Kubernetes working with AWS to create two IAM EC2 roles for master and worker(EC2의 K8s가 정상적으로 구동하려면 Master와 Worker로 사용될 EC2에 역할을 부여)

## 5.1. IAM Master Role

- Go to IAM -> Policies, click Create policy, into the JSON and add a new policy description(IAM 서비스 선택 후 정책을 누른 후 정책 생성을 선택합니다. 그리고 JSON을 눌러서 하기애 있는 코드를 복사붙여넣기)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "autoscaling:DescribeAutoScalingGroups",
        "autoscaling:DescribeLaunchConfigurations",
        "autoscaling:DescribeTags",
        "ec2:DescribeInstances",
        "ec2:DescribeRegions",
        "ec2:DescribeRouteTables",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeSubnets",
        "ec2:DescribeVolumes",
        "ec2>CreateSecurityGroup",
        "ec2:CreateTags",
        "ec2:CreateVolume",
        "ec2:ModifyInstanceAttribute",
        "ec2:ModifyVolume",
        "ec2:AttachVolume",
        "ec2:AuthorizeSecurityGroupIngress",
        "ec2:CreateRoute",
        "ec2:DeleteRoute",
        "ec2:DeleteSecurityGroup",
        "ec2:DeleteVolume",
        "ec2:DetachVolume",
        "ec2:RevokeSecurityGroupIngress",
        "ec2:DescribeVpcs",
        "elasticloadbalancing:AddTags",
        "elasticloadbalancing:AttachLoadBalancerToSubnets",
        "elasticloadbalancing:ApplySecurityGroupsToLoadBalancer",
        "elasticloadbalancing>CreateLoadBalancer",
        "elasticloadbalancing>CreateLoadBalancerPolicy",
        "elasticloadbalancing>CreateLoadBalancerListeners",
        "elasticloadbalancing:ConfigureHealthCheck",
        "elasticloadbalancing>DeleteLoadBalancer",
        "elasticloadbalancing>DeleteLoadBalancerListeners",
        "elasticloadbalancing:DescribeLoadBalancers",
        "elasticloadbalancing:DescribeLoadBalancerAttributes",
        "elasticloadbalancing:DetachLoadBalancerFromSubnets",
        "elasticloadbalancing:DeregisterInstancesFromLoadBalancer",
        "elasticloadbalancing:ModifyLoadBalancerAttributes"
      ]
    }
  ]
}
```

```

    "elasticloadbalancing:RegisterInstancesWithLoadBalancer",
    "elasticloadbalancing:SetLoadBalancerPoliciesForBackendServer",
    "elasticloadbalancing:AddTags",
    "elasticloadbalancing>CreateListener",
    "elasticloadbalancing>CreateTargetGroup",
    "elasticloadbalancing>DeleteListener",
    "elasticloadbalancing>DeleteTargetGroup",
    "elasticloadbalancing>DescribeListeners",
    "elasticloadbalancing>DescribeLoadBalancerPolicies",
    "elasticloadbalancing>DescribeTargetGroups",
    "elasticloadbalancing>DescribeTargetHealth",
    "elasticloadbalancing>ModifyListener",
    "elasticloadbalancing>ModifyTargetGroup",
    "elasticloadbalancing>RegisterTargets",
    "elasticloadbalancing>SetLoadBalancerPoliciesOfListener",
    "iam>CreateServiceLinkedRole",
    "kms:DescribeKey"
],
"Resource": [
    "*"
]
}
]
}

```

Cloud Groups   EC2   RDS   S3   VPC   Route 53   CloudFormation   CloudFront   CloudWatch   IAM   1   1

## Create policy

1

A policy defines the AWS permissions that you can assign to a user, group, or role. You can create and edit a policy in the visual editor and using JSON. [Learn more](#)

This policy validation failed and might have errors converting to JSON : The policy must have at least one statement. For more information about the IAM policy grammar, see [AWS Policies](#)

Visual editor   **JSON**   Import manifest

```

1 - {
2     "Version": "2012-10-17",
3     "Statement": [
4         {
5             "Effect": "Allow",
6             "Action": [
7                 "autoscaling:DescribeAutoScalingGroups",
8                 "autoscaling:DescribeLaunchConfigurations",
9                 "autoscaling:DescribeTags",
10                "ec2:DescribeInstances",
11                "ec2:DescribeRegions",
12                "ec2:DescribeRouteTables",
13                "ec2:DescribeSecurityGroups",
14                "ec2:DescribeSubnets",
15                "ec2:DescribeVolumes",
16                "ec2>CreateSecurityGroup",
17                "ec2>CreateTags",
18                "ec2>CreateVolume",
19                "ec2:ModifyInstanceAttribute",
20                "ec2:ModifyVolume",
21                "ec2:AttachVolume",
22                "ec2:AuthorizeSecurityGroupIngress",
23                "ec2>CreateRoute",
24                "ec2>DeleteRoute",
25                "ec2>DeleteSecurityGroup",
26                "ec2:DeleteVolume",
27                "ec2:DetachVolume",
28                "ec2:RevokeSecurityGroupIngress",
29                "ec2:DescribeVpcs",
30                "elasticloadbalancing>AddTags",
31                "elasticloadbalancing>AttachLoadBalancerToSubnets",

```

- Save it(저장)

Create policy

Review policy

Name\* k8s-cluster-iam-master-policy

Description

Summary

Service	Access level	Resource	Request condition
Allow (6 of 189 services) Show remaining 183			
EC2	Limited: List, Write, Tagging	All resources	None
EC2 Auto Scaling	Full: Read Limited: List	All resources	None
ELB	Full: List Limited: Read, Write, Tagging	All resources	None
ELB v2	Limited: Read, Write, Tagging	All resources	None
IAM	Limited: Write	All resources	None
KMS	Limited: Read	All resources	None

\* Required

Cancel Previous Create

- Go to the Roles, create a role using the EC2 type(역할로 들어가서 EC2를 선택)

Create role

Select type of trusted entity

AWS service  
EC2, Lambda and others

Another AWS account  
Belonging to you or 3rd party

Web identity  
Cognito or any OpenID provider

SAML 2.0 federation  
Your corporate directory

Allows AWS services to perform actions on your behalf. Learn more

Choose the service that will use this role

EC2

Lambda

API Gateway AWS Backup AWS Support Amplify AppSync Application Auto Scaling Application Discovery Service Batch CloudFormation CloudHSM CloudTrail CloudWatch Application Comprehend Config Connect DMS Data Lifecycle Manager Data Pipeline DataSync DeepLens Directory Service DynamoDB EC2 - Fleet EC2 Auto Scaling ElastiCache Elastic Beanstalk Elastic Container Service Elastic Transcoder Forecast Glue GuardDuty Inspector IoT IoT Things Graph Kinesis Lex License Manager Machine Learning Macie MediaConvert Migration Hub OpsWorks Personalize RAM Redshift Rekognition RDS SageMaker Security Hub Service Catalog Step Functions Storage Gateway Textract Transfer Trusted Advisor VPC WorkLink

\* Required

Cancel Next: Permissions

- Click on the Permissions, find and attach the policy added above(권한 정책 부분에 위에 만들었던 정책 이름 기입 후 정책을 역할에 추가)



## Create role

1 2 3 4

### Attach permissions policies

Choose one or more policies to attach to your new role.

[Create policy](#)



Filter policies		Policy name	Used as	Description
<input checked="" type="checkbox"/>	▶	k8s-cluster-iam-master-policy	None	

### Set permissions boundary

\* Required

[Cancel](#) [Previous](#) [Next: Tags](#)



## Create role

1 2 3 4

### Review

Provide the required information below and review this role before you create it.

Role name\*

Use alphanumeric and '+=\_@-' characters. Maximum 64 characters.

Role description

Maximum 1000 characters. Use alphanumeric and '+=\_@-' characters.

Trusted entities AWS service: ec2.amazonaws.com

Policies [k8s-cluster-iam-master-policy](#)

Permissions boundary Permissions boundary is not set

No tags were added.

[Cancel](#) [Previous](#) [Create role](#)

## 5.2. IAM Worker Role

- In the same way, create another policy for worker nodes(Master 역할을 만든 것처럼 똑같이 Worker 역할을 만들기)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeInstances",
        "ec2:DescribeRegions",
        "ecr:GetAuthorizationToken",
        "ecr:BatchCheckLayerAvailability",
        "ecr:GetDownloadUrlForLayer",
        "ecr:GetRepositoryPolicy",
        "ecr:DescribeRepositories",
        "ecr>ListImages",
        "ecr:BatchGetImage"
      ],
      "Resource": "*"
    }
  ]
}
```

- Save it as k8s-cluster-iam-worker-policy(k8s-cluster-iam-worker-policy라는 이름으로 저장)



Create policy

/ policy

Name\* k8s-cluster-iam-worker-policy

Use alphanumeric and '+, -, @, \_' characters. Maximum 128 characters.

Description

Maximum 1000 characters. Use alphanumeric and '+, -, @, \_' characters.

Summary

Filter

Service	Access level	Resource	Request condition
Allow (2 of 189 services) Show remaining 187			
EC2	Limited: List	All resources	None
Elastic Container Registry	Full: List Limited: Read	All resources	None

Created

Cancel

Previous

Create

- And create a k8s-cluster-iam-worker-role(그리고 k8s-cluster-iam-worker-role를 생성)

Create role

Review

Provide the required information below and review this role before you create it.

**Role name\*** k8s-cluster-iam-worker-role  
Use alphanumeric and '+'-' characters. Maximum 64 characters.

**Role description** Allows EC2 instances to call AWS services on your behalf.  
Maximum 1000 characters. Use alphanumeric and '+'-' characters.

**Trusted entities** AWS service: ec2.amazonaws.com

**Policies** k8s-aws-cluster-worker-iam-policy

**Permissions boundary** Permissions boundary is not set

No tags were added.

\* Required

Cancel Previous Create role

## 6. Running EC2

- Create an EC2 using t2.medium type (minimal type as Kubernetes master needs to have at least 2 CPU cores and 2 GB of RAM), using your VPC and set k8s-cluster-iam-master-role as the IAM role(쿠버네티스 Master 노드는 최소 2개의 CPU와 2GB의 램이 필요하기 때문에 EC2의 t2.medium 타입을 선택하고 이전에 생성한 VPC 선택 후 IAM 역할을 추가)

Services ▾    EC2    RDS    S3    VPC    Route 53    CloudFormation    CloudFront    CloudWatch    IAM

1. Choose AMI    2. Choose Instance Type    3. Configure Instance    4. Add Storage    5. Add Tags    6. Configure Security Group    7. Review

**Step 3: Configure Instance Details**

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of the latest prices, or use reserved instances to save money over time.

**Number of instances** 1    Launch into Auto Scaling Group

**Purchasing option** Request Spot instances

**Network** vpc-00881455e7f607dd7 | k8s-cluster-vpc    Create new VPC

**Subnet** subnet-05253c2474384b21a | k8s-cluster-net | eu-west-1    Create new subnet  
251 IP Addresses available

**Auto-assign Public IP** Use subnet setting (Enable)

**Placement group** Add instance to placement group

**Capacity Reservation** Open    Create new Capacity Reservation

**IAM role** k8s-cluster-iam-master-role    Create new IAM role

**Shutdown behavior** Stop

**Enable termination protection** Protect against accidental termination

**Monitoring** Enable CloudWatch detailed monitoring  
Additional charges apply.

**Tenancy** Shared - Run a shared hardware instance  
Additional charges will apply for dedicated tenancy.

**T2/T3 Unlimited** Enable  
Additional charges may apply

▼ Network interfaces

- Add tags(태그를 추가)

**Step 5: Add Tags**

A tag consists of a case-sensitive key-value pair. For example, you could define a tag with key = Name and value = Webserver. A copy of a tag can be applied to volumes, instances or both. Tags will be applied to all instances and volumes. [Learn more](#) about tagging your Amazon EC2 resources.

Key	(128 characters maximum)	Value	(256 characters maximum)	Instances	Volumes
kubernetes.io/cluster/kubernetes	owned			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Name		k8s-master-ec2		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
<a href="#">Add another tag</a> (Up to 50 tags maximum)					

- Create a Security Group(보안 그룹을 생성)

**Step 6: Configure Security Group**

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and all HTTP and HTTPS ports. You can create a new security group or select from an existing one below. [Learn more](#) about Amazon EC2 security groups.

Assign a security group:  Create a new security group  Select an existing security group

Security group name: k8s-cluster-sg  
Description: k8s-cluster-sg

Type	Protocol	Port Range	Source
All traffic	All	0 - 65535	Anywhere 0.0.0.0/0

[Add Rule](#)

**Warning**

- While Master is spinning up, create a Worker Node in the same way and change the IAM role only which is *k8s-cluster-iam-worker-role*(Master Node의 생성을 기다리는 동안 똑같이 Worker Node를 생성하지만 IAM 역할을 k8s-cluster-iam-worker-role로 변경)

**Step 3: Configure Instance Details**

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of the lower pricing, or use reserved instances to save money over time.

Number of instances	1	Launch into Auto Scaling Group
Purchasing option	<input type="checkbox"/> Request Spot instances	
Network	vpc-00881455e7f607dd7   k8s-cluster-vpc	<a href="#">Create new VPC</a>
Subnet	subnet-05253c2474384b21a   k8s-cluster-net   eu-west-1	<a href="#">Create new subnet</a> 250 IP Addresses available
Auto-assign Public IP	Use subnet setting (Enable)	
Placement group	<input type="checkbox"/> Add instance to placement group	
Capacity Reservation	Open	<a href="#">Create new Capacity Reservation</a>
IAM role	k8s-cluster-iam-worker-role	<a href="#">Create new IAM role</a>
Shutdown behavior	Stop	
Enable termination protection	<input type="checkbox"/> Protect against accidental termination	
Monitoring	<input type="checkbox"/> Enable CloudWatch detailed monitoring Additional charges apply.	
Tenancy	Shared - Run a shared hardware instance	<small>Additional charges will apply for dedicated tenancy.</small>
T2/T3 Unlimited	<input type="checkbox"/> Enable <small>Additional charges may apply</small>	
▼ Network interfaces		

- Add the tags(태그 추가)

**Step 5: Add Tags**

A tag consists of a case-sensitive key-value pair. For example, you could define a tag with key = Name and value = Webserver.

A copy of a tag can be applied to volumes, instances or both.

Tags will be applied to all instances and volumes. [Learn more](#) about tagging your Amazon EC2 resources.

Key	(128 characters maximum)	Value	(256 characters maximum)	Instances	Volumes
kubernetes.io/cluster/kubernetes		owned		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Name		k8s-worker-ec2		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Add another tag (Up to 50 tags maximum)

- Attach the existing **Security Group**(Master Node를 생성할 때 만들었던 보안 그룹 똑같이 사용)

**Step 6: Configure Security Group**

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server HTTP and HTTPS ports. You can create a new security group or select from an existing one below. [Learn more](#) about Amazon EC2 security groups.

Assign a security group:  Create a new security group  
 Select an existing security group

Security Group ID	Name	Description
sg-067e1a5e6646f282b	default	default VPC security group
sg-017e2417091de32c8	k8s-cluster-sg	k8s-cluster-sg

Inbound rules for sg-017e2417091de32c8 (Selected security groups: sg-017e2417091de32c8)

Type	Protocol	Port Range	Source
All traffic	All	All	0.0.0.0/0
All traffic	All	All	::/0

- Perform the next steps on both EC2. Update packages list and installed packages(하기 단계들을 생성한 2대의 EC2에 동일하게 진행)

```
# apt update && apt -y upgrade
```

- Add Docker and Kubernetes Repositories(도커 및 쿠버네티스 리포를 추가)

```
# apt-get install -y gnupg lsb-release
# mkdir -p /etc/apt/keyrings
# curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o
/etc/apt/keyrings/docker.gpg
# echo \
"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg]
https://download.docker.com/linux/ubuntu \
$(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
# apt-get update
# apt-get install -y docker-ce docker-ce-cli containerd.io docker-compose-plugin
# docker info | grep Cgroup
```

```
# vim /etc/docker/daemon.json(cgroup이 cgroupfs일 때만 실행)
{ "exec-opts": ["native.cgroupdriver=systemd"] }
# systemctl daemon-reload
# systemctl restart docker
# systemctl status docker(docker daemon이 active 상태인지 확인)
# docker info | grep Cgroup
```

```
# sudo curl -fsSLo /usr/share/keyrings/kubernetes-archive-keyring.gpg
https://packages.cloud.google.com/apt/doc/apt-key.gpg
```

```
# echo "deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg]
https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee
/etc/apt/sources.list.d/kubernetes.list
```

- Get it as a fully qualified domain name(FQDN 형식의 도메인명으로 조회)

```
# curl http://169.254.169.254/latest/meta-data/local-hostname
```

- Set the hostname as FQDN(호스트명을 FQDN으로 변경)

```
# hostnamectl set-hostname "FQDN"
```

- Check the hostname and do the same step on the Worker(호스트명 확인 및 호스트명 변경 단계를 Worker에서도 똑같이 진행)

```
# hostname
```

```
# apt-get update
# apt-get install -y kubelet=1.23.0-00 kubeadm=1.23.0-00 kubectl=1.23.0-00
# apt-mark hold kubelet kubeadm kubectl
```

**//Run this command on Master Node only(Master 노드에서만 실행)**

```
# swapoff -a && sed -i '/swap/s/&/#/ ' /etc/fstab
# cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf
br_netfilter
EOF
```

```
# vim /etc/kubernetes/aws.yaml
apiVersion: kubeadm.k8s.io/v1beta2
kind: ClusterConfiguration
networking:
  serviceSubnet: "10.100.0.0/16"
  podSubnet: "10.244.0.0/16"
apiServer:
  extraArgs:
    cloud-provider: "aws"
controllerManager:
  extraArgs:
    cloud-provider: "aws"

# kubeadm init --config /etc/kubernetes/aws.yaml
```

//After the init command, you'll get a kubeadm join command like below(init 명령어 이후 kubeadm join 명령어 확인)//

ex. kubeadm join 10.0.10.238:6443 --token ftknf4.yez7x0g7wxo8vt66 \  
--discovery-token-ca-cert-hash  
sha256:4c2fe2bd01da2a47be7ba1ad6b774f84e3188cff25ccddcadf8fa55e7e688a22

```
# mkdir -p $HOME/.kube  
# cp -i /etc/kubernetes/admin.conf $HOME/.kube/config  
# chown ubuntu:ubuntu $HOME/.kube/config
```

```
# kubectl apply -f  
https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
```

**//Run this command on Worker Node only(Worker 노드에서만 실행)**

```
# vim /etc/kubernetes/node.yaml  
apiVersion: kubeadm.k8s.io/v1beta2  
kind: JoinConfiguration  
discovery:  
  bootstrapToken:  
    token: "ftknf4.yez7x0g7wxo8vt66"  
    apiServerEndpoint: "10.0.10.238:6443"  
    caCertHashes:  
      - "sha256:4c2fe2bd01da2a47be7ba1ad6b774f84e3188cff25ccddcadf8fa55e7e688a22"  
nodeRegistration:  
  name: ip-10-0-10-123.ap-northeast-2.compute.internal(this should be the worker nodes  
FQDN, worker node의 FQDN 적요)  
  kubeletExtraArgs:  
    cloud-provider: aws
```

//fyi. match the colors from kubeadm init command. pink=apiServerEndpoint, orange=token  
and green=caCertHashes(kubeadm init 명령어에서 받은 핑크값이 apiServerEndpoint,  
오렌지값이 token 그리고 그린값이 caCertHashes//

```
# kubeadm join --config /etc/kubernetes/node.yaml
```

**//Back on Master Node(Master Node로 다시 접속)**

```
# kubectl get nodes -o wide
```

```
root@ip-10-0-10-238:/home/ubuntu# kubectl get node -o wide  
NAME           STATUS   ROLES     AGE    VERSION   INTERNAL-IP   EXTERNAL-IP   OS-IMAGE  
ip-10-0-10-123.ap-northeast-2.compute.internal   Ready    <none>    15m    v1.23.0   10.0.10.123  3.39.250.33  Ubuntu 20.04.5 LTS  
ip-10-0-10-238.ap-northeast-2.compute.internal   Ready    control-plane,master  32m    v1.23.0   10.0.10.238  <none>       Ubuntu 20.04.5 LTS
```

- Load Balancer Creation by creating a simple nginx container(nginx 컨테이너를 로드 밸런서로 배포)

```
# vim /home/ubuntu/lb-deploy.yaml
```

```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: hello
spec:
  replicas: 1
  selector:
    matchLabels:
      app: hello
  template:
    metadata:
      labels:
        app: hello
    spec:
      containers:
        - name: hello
          image: nginx

```

```

# vim /home/ubuntu/lb-svc.yaml
apiVersion: v1
kind: Service
metadata:
  name: hello
spec:
  type: LoadBalancer
  selector:
    app: hello
  ports:
    - name: http
      protocol: TCP
      port: 80

```

```

# kubectl create -f /home/ubuntu/lb-deploy.yaml
# kubectl create -f /home/ubuntu/lb-svc.yaml

```

```
# kubectl get deploy
```

```

root@ip-10-0-10-238:/home/ubuntu# kubectl get node -o wide
NAME           STATUS  ROLES      AGE   VERSION  INTERNAL-IP  EXTERNAL-IP  OS-IMAGE
ip-10-0-10-123.ap-northeast-2.compute.internal  Ready   <none>      15m   v1.23.0  10.0.10.123  3.39.250.33  Ubuntu 20.04.5 LTS
ip-10-0-10-238.ap-northeast-2.compute.internal  Ready   control-plane,master  32m   v1.23.0  10.0.10.238  <none>      Ubuntu 20.04.5 LTS

```

```
# kubectl get svc
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)
hello	LoadBalancer	10.100.29.240	ac9257581b2348fe995337f76c16904-93225302.ap-northeast-2.elb.amazonaws.com	80:30839/TCP
kubernetes	ClusterIP	10.100.0.1	<none>	443/TCP

- Check an ELB in AWS Console (ELB가 정상적으로 생성되었는지 AWS 콘솔에서 확인)

The screenshot shows the AWS Load Balancers console. At the top, there's a search bar and a table header with columns: Name, DNS name, State, VPC ID, Availability Zones, Type, and Created At. A single row is selected in the table, corresponding to the load balancer's details below.

**Load balancer: acd9257581b2348fe995337f76c16904**

**Description** tab is selected. Other tabs include Instances, Health check, Listeners, Monitoring, Tags, and Migration.

**Basic Configuration**

Name	acd9257581b2348fe995337f76c16904	Creation time	September 22, 2022 at 1:40:52 PM UTC+9
* DNS name	acd9257581b2348fe995337f76c16904-93225302.ap-northeast-2.elb.amazonaws.com (A Record)	Hosted zone	ZWKZPGT148KDX
Type	Classic (Migrate Now)	Status	1 of 1 instances in service
Schema	internet-facing	VPC	vpc-01595ae19f14cd4aa
Availability Zones	subnet-03cd0697182d73fb2 - ap-northeast-2b, subnet-0b534650399ffd47c - ap-northeast-2d,		

- Check if Worker node is assigned properly to the ELB (Worker node가 ELB에 정상적으로 연동이 되었는지 확인)

The screenshot shows the AWS Load Balancers console with the same load balancer details as the previous screenshot. The Instances tab is now selected.

**Connection Draining:** Disabled (Edit)

**Edit Instances**

Instance ID	Name	Availability Zone	Status	Actions
i-0ccb618628af231e14	worker01	ap-northeast-2a	InService ⓘ	Remove from Load Balancer

7. AWS Load Balancer(TroubleShooting) - no Worker Node added(k8s에서 서비스 배포 시 로드밸런서에 worker node가 연동이 안되었을 때 해결방법)

```
# kubectl edit node "worker-nodes-FQDN"
```

```
29   name: ip-10-0-10-123.ap-northeast-2.compute.internal
30   resourceVersion: "4847"
31   uid: dc28a22e-13e2-4173-814e-190db2aa481b
32 spec:
33   podCIDR: 10.244.1.0/24
34   podCIDRs:
35     - 10.244.1.0/24
36   providerID: aws:///ap-northeast-2a/i-0cb618628af231e14
```

// If you don't see providerID like the image, add one and just change the last part to your worker node's instance ID and save the file(위 사진과 같이 providerID가 생성이 안되어있다면 추가 및 마지막 부분 자신의 worker node 인스턴스 ID로 변경 후 파일 저장)//