

# 7/24 3일차

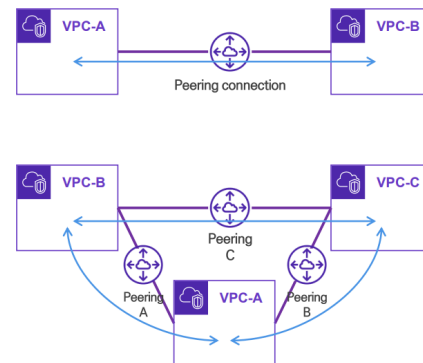
## VPC Peering

### 2. VPC Peering (1/4)

#### VPC Peering 개요

- 서로 격리 되어 있는 **2개 이상의 VPC의 네트워크를 연결**하는 서비스
  - 동일 계정**에 구성된 서로 다른 VPC 연결 가능
  - 다른 계정**에 구성된 서로 다른 VPC 연결 가능
  - 서로 다른 리전**에 구성된 VPC 연결 가능
- VPC Peering 연결 리소스 비용은 **무료**
  - 동일 리전, 가용영역에서 발생시키는 데이터 트래픽은 무료
  - 리전, 가용영역을 교차**하는 데이터 트래픽은 **과금**
- VPC Peering 제약사항
  - 전이적 피어링 구성 불가능**
  - CIDR Block이 겹치는 경우** VPC Peering 구성 불가능

#### VPC Peering 구성



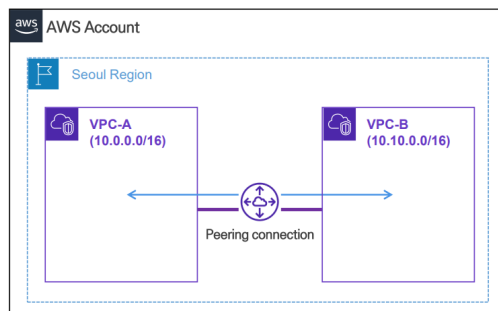
vpc끼리 연결을 private하게 해준다.

private한 vpc들을 인터넷 거치지 않고 통신 가능하게끔

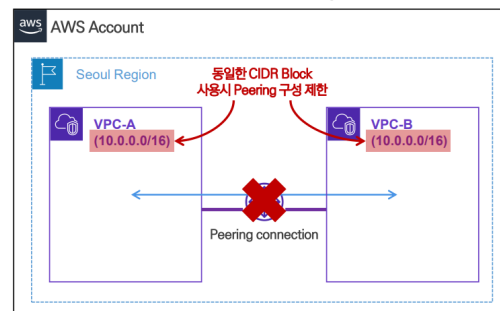
## 주의사항

### 2. VPC Peering (2/4)

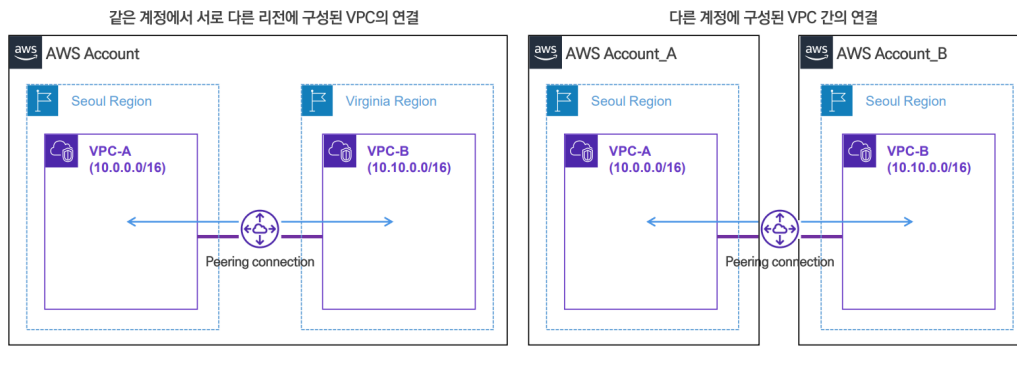
#### 같은 계정에 구성된 서로 다른 VPC의 연결



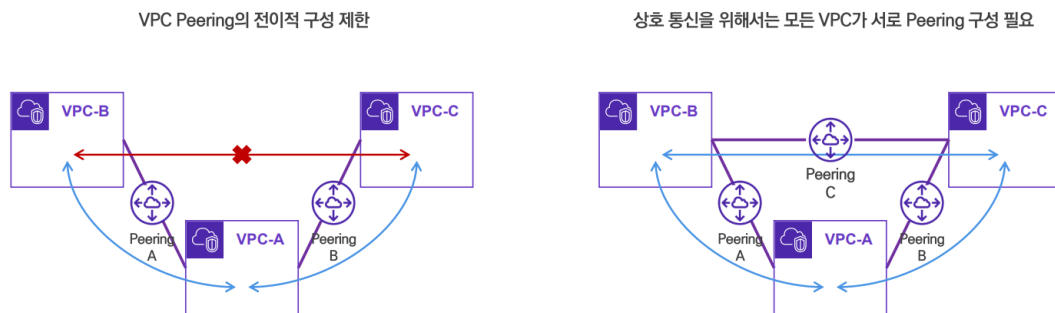
#### CIDR Block이 겹치는 경우 Peering 구성 제한



## 2. VPC Peering (3/4)



## 2. VPC Peering (4/4)



## vpc peering 실습

먼저 sh파일로 vpc와 서브넷 2개 인스턴스 2개 생성

```
sh cloud-wave-workspace/scripts/create_ec2_instance.sh
```

```
#!/bin/bash
```

```
# VPC / SUBNETS INFORMATION
```

```
VPC_NAME="lab-edu-vpc-ap-01"
```

```
PRI_SUB_NAME_01="lab-edu-sub-pri-01"
```

```
PRI_SUB_NAME_02="lab-edu-sub-pri-02"
```

```
NETWORK_EC2_NAME_01="lab-edu-ec2-network-ap-01"
```

```
NETWORK_EC2_NAME_02="lab-edu-ec2-network-ap-02"
```

```
# GET VPC ID
```

```
VPC_ID=$(aws ec2 describe-vpcs --filters "Name=tag:Name,Values=$VPC_NAME" --query
```

```

if [ -z "$VPC_ID" ]; then
    echo "Error: VPC not found."
    exit 1
else
    echo "VPC found: $VPC_ID"
fi

# GET SUBNET ID
SUBNET_ID_PRIVATE_01=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID")
SUBNET_ID_PRIVATE_02=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID")
if [ -z "$SUBNET_ID_PRIVATE_01" ] || [ -z "$SUBNET_ID_PRIVATE_02" ]; then
    echo "Error: One or both subnets not found."
    exit 1
else
    echo "PRIVATE_SUBNET_01 found: $SUBNET_ID_PRIVATE_01"
    echo "PRIVATE_SUBNET_02 found: $SUBNET_ID_PRIVATE_02"
fi

# INSTANCE INFORMATION
AMI_ID="ami-0ff1cd0b5d98708d1"
INSTANCE_TYPE="t3.micro"

# CREATE KEY-PAIR
KEY_NAME="lab-edu-key-network"
KEY_PATH="/home/ec2-user/.ssh/$KEY_NAME.pem"
aws ec2 create-key-pair --key-name $KEY_NAME --query 'KeyMaterial' --output text >
if [ $? -eq 0 ]; then
    chmod 400 $KEY_PATH
    yes | mv $KEY_NAME.pem $KEY_PATH
    echo "Key pair created successfully: $KEY_PATH"
else
    echo "Failed to create key pair: $KEY_NAME"
    exit 1
fi

# CREATE SECURITY-GROUP
SG_NAME="lab-edu-sg-network"
SG_ID=$(aws ec2 create-security-group --group-name $SG_NAME --description "My security group")
if [ -z "$SG_ID" ]; then
    echo "Failed to create Security Group: $SG_ID"
    exit 1
else
    echo "VPC found: $SG_ID"
fi

# Allow ICMP & SSH access
aws ec2 authorize-security-group-ingress --group-id $SG_ID --protocol tcp --port 22
if [ $? -eq 0 ]; then
    echo "Security Group Rule created successfully: SSH"
else
    echo "Failed to create Security Group Rule: SSH"
fi

```

```

    exit 1
fi
aws ec2 authorize-security-group-ingress --group-id $SG_ID --protocol icmp --port
if [ $? -eq 0 ]; then
    echo "Security Group Rule created successfully: ICMP"
else
    echo "Failed to create Security Group Rule: ICMP"
    exit 1
fi

# CREATE INSTANCES
NETWORK_EC2_IP_01=$(aws ec2 run-instances --image-id $AMI_ID --count 1 --instance-
INSTANCE_ID_01=$(aws ec2 describe-instances --filters "Name=private-ip-address,Val
NETWORK_EC2_IP_02=$(aws ec2 run-instances --image-id $AMI_ID --count 1 --instance-
INSTANCE_ID_02=$(aws ec2 describe-instances --filters "Name=private-ip-address,Val
if [ -z "$NETWORK_EC2_IP_01" ] || [ -z "$NETWORK_EC2_IP_02" ]; then
    echo "Error: One or both instance not found."
    exit 1
else
    echo "NETWORK_EC2_01 created successfully: $NETWORK_EC2_IP_01, $INSTANCE_ID_01"
    echo "NETWORK_EC2_02 created successfully: $NETWORK_EC2_IP_02, $INSTANCE_ID_02"
fi

# IAM Role Binding
IAM_ROLE_NAME="lab-edu-role-ec2"
aws ec2 associate-iam-instance-profile --instance-id $INSTANCE_ID_01 --iam-instanc
aws ec2 associate-iam-instance-profile --instance-id $INSTANCE_ID_02 --iam-instanc

CONFIG_PATH="/home/ec2-user/.ssh/config"
cat <<EOF >> $CONFIG_PATH

Host network-01
    HostName $NETWORK_EC2_IP_01
    User ec2-user
    IdentityFile $KEY_PATH

Host network-02
    HostName $NETWORK_EC2_IP_02
    User ec2-user
    IdentityFile $KEY_PATH
EOF
if [ $? -eq 0 ]; then
    echo "Configuration created successfully: $CONFIG_PATH"
else
    echo "Failed to create Configuration: $CONFIG_PATH"
    exit 1
fi

```

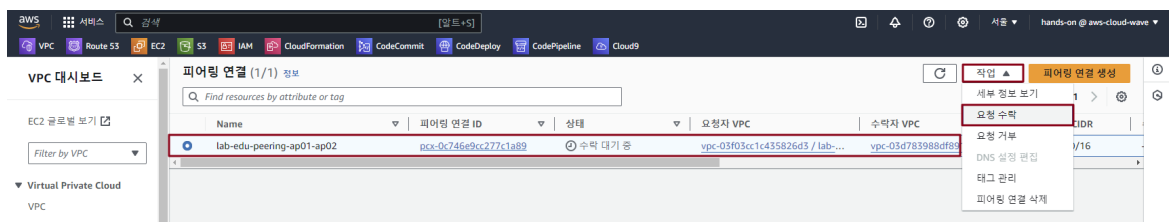
## 서울 리전 ↔ 서울 리전 VPC Peering

### 1. VPC Peering Resource 생성

- VPC 콘솔 메인 화면 → 피어링 연결 리소스 탭 → "피어링 연결 생성" 버튼 클릭
- VPC Peering 생성 정보 입력 → '피어링 연결 생성' 버튼 클릭
  - 이름: lab-edu-peering-ap01-ap02
  - VPC ID(요청자):lab-edu-vpc-ap-01
  - 계정: 내 계정
  - 리전: 현재 리전
  - VPC ID(수락자): lab-edu-vpc-ap-02



- 피어링 연결 리소스 탭으로 이동 → "lab-edu-peering-ap01-ap02" 리소스 선택 → '작업' 버튼 클릭 → '요청 수락' → '요청 수락'



### 2. Routing Table 수정

- VPC 콘솔 메인 화면 → 라우팅 테이블 탭 → "lab-edu-rtb-pri-01" 선택 → '라우팅' 탭 → '라우팅 편집' 버튼 클릭

- 라우팅 테이블 경로 생성 정보 입력
  - '라우팅 추가' 버튼 클릭
  - 대상: 10.10.0.0/16
  - 대상: 피어링 연결 (lab-edu-peering-ap01-ap02)
  - '변경 사항 저장' 버튼 클릭
- VPC 콘솔 메인 화면 → 라우팅 테이블 탭 → "lab-edu-rtb-2nd-pri-01" 선택 → '라우팅' 탭 → '라우팅 편집' 버튼 클릭
- 라우팅 테이블 경로 생성 정보 입력
  - '라우팅 추가' 버튼 클릭
  - 대상: 10.0.0.0/16
  - 대상: 피어링 연결 (lab-edu-peering-ap01-ap02)
  - '변경 사항 저장' 버튼 클릭

### 3. Network 통신 테스트

- EC2 메인 콘솔 화면으로 이동 → 인스턴스 리소스 탭 → 'lab-edu-ec2-network-2nd-ap' 선택 → Private IP 주소 복사
- Cloud9 IDE Terminal 화면으로 이동 → ssh 명령어 실행

```
ssh web-server
```

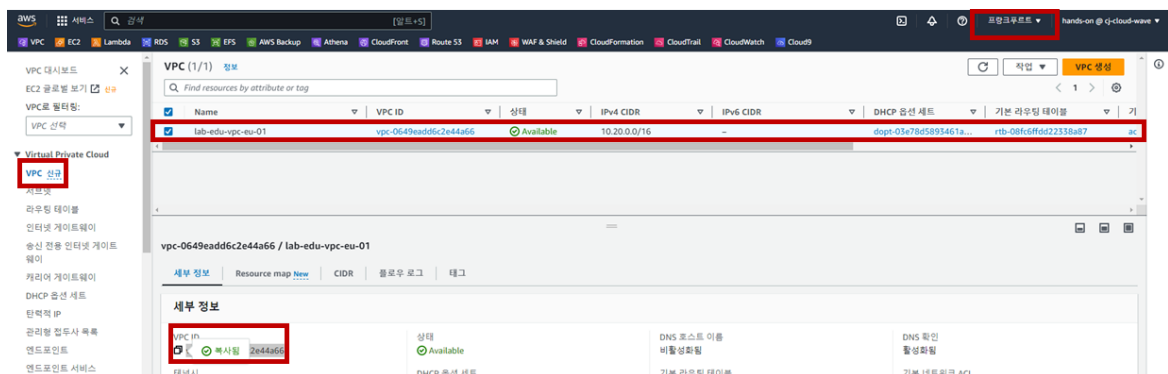
- ICMP 통신 테스트 진행

```
ping {2ND_VPC_NETWORK_SERVER_PRIVATE_IP}
PING 10.10.40.140 (10.10.40.140) 56(84) bytes of data.
64 bytes from 10.10.40.140: icmp_seq=1 ttl=255 time=0.451 ms
64 bytes from 10.10.40.140: icmp_seq=2 ttl=255 time=0.473 ms
64 bytes from 10.10.40.140: icmp_seq=3 ttl=255 time=0.384 ms
64 bytes from 10.10.40.140: icmp_seq=4 ttl=255 time=0.536 ms
```

## 서울 리전 ↔ 프랑크푸르트 리전 VPC Peering

### 1. 프랑크푸르트 리전 VPC ID 확인

- 프랑크푸르트 리전으로 이동 → VPC 콘솔 메인 화면 → VPC 리소스 탭 → "lab-edu-vpc-eu-01" 선택 → 세부화면의 'VPC ID' 복사



### 2. VPC Peering Resource 생성

- VPC 콘솔 메인 화면 → 피어링 연결 리소스 탭 → "피어링 연결 생성" 버튼 클릭
- VPC Peering 생성 정보 입력 → '피어링 연결 생성' 버튼 클릭
  - 이름: lab-edu-peering-ap01-eu01
  - VPC ID(요청자):lab-edu-vpc-ap-01
  - 계정: 내 계정
  - 리전: 다른 리전 → **eu-central-1**
  - VPC ID(수락자): {프랑크푸르트\_VPC\_ID}

### 3. VPC Peering 수락

- 프랑크푸르트 리전으로 이동 → VPC 콘솔 메인 화면 → 피어링 연결 리소스 탭
- 수락 대기 상태의 Peering 리소스 선택 → '작업' → '요청 수락' → '요청 수락'

### 4. Routing Table 수정

- 서울 리전으로 이동 → VPC 콘솔 메인 화면 → 라우팅 테이블 탭 → "lab-edu-rtb-pri-01" 선택 → '라우팅' 탭 → '라우팅 편집' 버튼 클릭
- 라우팅 테이블 경로 생성 정보 입력
  - '라우팅 추가' 버튼 클릭
  - 대상: 10.30.0.0/16
  - 대상: 피어링 연결 (lab-edu-peering-ap01-eu01)
  - '변경 사항 저장' 버튼 클릭
- 프랑크푸르트 리전으로 이동 → VPC 콘솔 메인 화면 → 라우팅 테이블 탭 → "lab-edu-rtb-eu-pri-01" 선택 → '라우팅' 탭 → '라우팅 편집' 버튼 클릭
- 라우팅 테이블 경로 생성 정보 입력
  - '라우팅 추가' 버튼 클릭
  - 대상: 10.0.0.0/16
  - 대상: 피어링 연결 (lab-edu-peering-ap01-eu01)
  - '변경 사항 저장' 버튼 클릭

### 5. Network 통신 테스트

- 프랑크푸르트 리전으로 이동 → EC2 메인 콘솔 화면 → 인스턴스 리소스 탭 → 'lab-edu-ec2-network-eu' 선택 → Private IP 주소 복사
- Cloud9 IDE Terminal 화면으로 이동 → ssh 명령어 실행

```
ssh web-server
```

- ICMP 통신 테스트 진행

```
ping {FRANKFURT_NETWORK_SERVER_PRIVATE_IP}
PING 10.30.40.64 (10.30.40.64) 56(84) bytes of data.
64 bytes from 10.30.40.64: icmp_seq=1 ttl=127 time=225 ms
64 bytes from 10.30.40.64: icmp_seq=2 ttl=127 time=225 ms
64 bytes from 10.30.40.64: icmp_seq=3 ttl=127 time=225 ms
64 bytes from 10.30.40.64: icmp_seq=4 ttl=127 time=225 ms
```

## 전이적 VPC Peering Network 통신 테스트

- 프랑크푸르트 리전에서 앞서 생성했던 피어링 연결을 이용해 프랑크푸르트랑 서울 vpc2 연결

**세부 정보**

라우팅 테이블 ID rtb-01f9770607f38a149	기본 아니요	명시적 서브넷 연결 2 서브넷	옛지 연결 -
VPC vpc-025c845bc53f9b4fd   lab-edu-vpc-eu	소유자 ID 211125410568		

**라우팅 (4)**

대상	대상	상태	전파됨
pl-6ea54007	vpce-02e28a57e76fdd767	✓ 활성화	아니요
10.0.0.0/16	pcx-06e18cfa711016db	✓ 활성화	아니요
10.10.0.0/16	pcx-06e18cfa711016db	✓ 활성화	아니요
10.30.0.0/16	local	✓ 활성화	아니요

- 서울 리전에서 앞서 생성했었던 연결을 이용해 서울 vpc2와 프랑크푸르트 연결

**세부 정보**

라우팅 테이블 ID rtb-0183618836019df90	기본 아니요	명시적 서브넷 연결 subnet-01414b0b83bc5cbb6 / lab-edu-sub-2nd-pri-01	옛지 연결 -
VPC vpc-00ebc0304455579d0   lab-edu-vpc-ap-02	소유자 ID 211125410568		

**라우팅 (5)**

대상	대상	상태	전파됨
pl-78a54011	vpce-06455fea7c2b788f9	✓ 활성화	아니요
0.0.0.0/0	nat-0dc519e9a3903e105	✓ 활성화	아니요
10.0.0.0/16	pcx-0289566e0328ee60a	✓ 활성화	아니요
10.10.0.0/16	local	✓ 활성화	아니요
10.30.0.0/16	pcx-0289566e0328ee60a	✓ 활성화	아니요

즉 vpc1을 매개로 전이적 연결 테스트

### 1. 프랑크푸르트 리전 Network Server 접속

- 프랑크푸르트 리전으로 이동 → EC2 콘솔 메인 화면 → 인스턴스 리소스 탭 → 'lab-edu-ec2-network-us' 선택 → '연결' 버튼 클릭
- 'Session Manager' 탭으로 이동 → '연결' 버튼 클릭

### 2. Network 통신 테스트



```
ping {2ND_VPC_NETWORK_SERVER_PRIVATE_IP}
PING 10.10.40.140 (10.10.40.140) 56(84) bytes of data.
^C
--- 10.10.40.140 ping statistics ---
312 packets transmitted, 0 received, 100% packet loss, time 323451ms
```

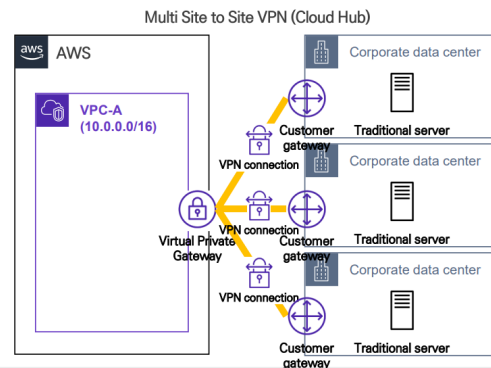
안되는 걸 확인할 수 있다.

## Site to Site VPN

### 3. Site to Site VPN

Site to Site VPN 개요

- AWS에서 제공하는 **완전 관리형 VPN 서비스**
- Virtual Private Gateway와 Custom Gateway를 생성하여 VPN 연결
  - **Virtual Private Gateway** : AWS VPN Device
  - **Custom Gateway** : On-Premise VPN Device
- Site to Site VPN 연결은 **2개의 VPN Tunnel 제공**
- Transit Gateway, Cloud Hub로 Hub & Spoke 형태의 VPN 환경 구성 가능



중간 인터넷 영역을 통신할 때 패킷을 암호화 중간에 스니핑해도 복호화를 못함

상대와 나만이 확인가능 전용선임에도 1:1 연결 같은 느낌

나만이 그 케이블을 점유해서 사용하므로 속도가 빠름

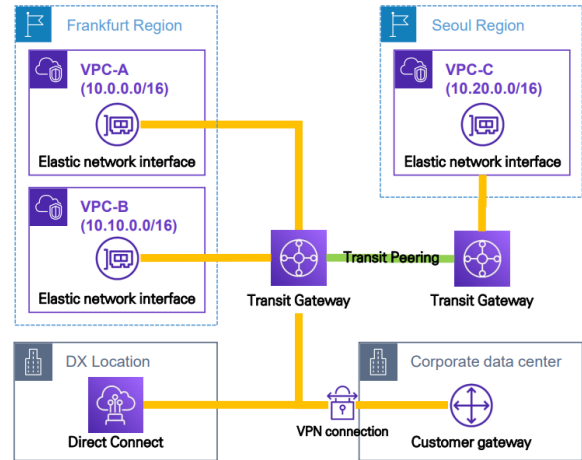
보안성이 굉장히 높다.

## Transit Gateway

#### 4. Transit Gateway (1/3)

Transit gateway 개요

- 서로 다른 **네트워크를 연결**하는데 사용하는 서비스
- **다른 리전**의 VPC와의 연결은 Transit gateway Peering 사용
- 다른 계정과 AWS RAM을 이용해 공유하면 **타 계정**의 네트워크도 연동 가능
- **Site to Site VPN, Direct Connect와 연동**해 온프레미스 네트워크와 연동 가능
- 중앙에서 **Transit Gateway Routing Table** 이용 경로 조절

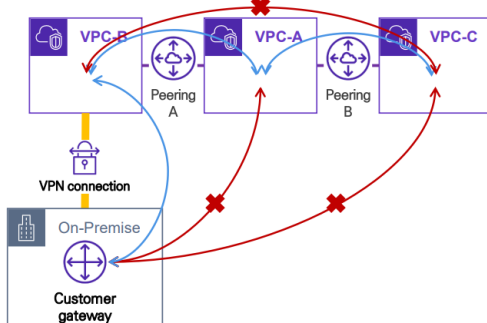


#### • 메쉬업 구조

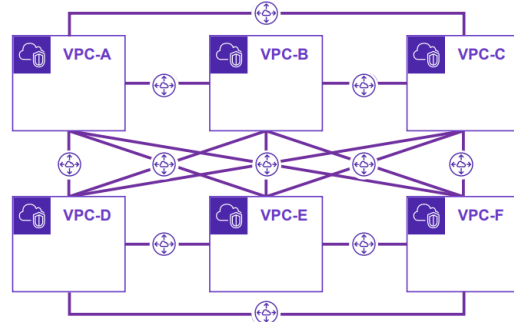
#### 4. Transit Gateway (2/3)

VPC Peering vs Transit Gateway (1) – VPC Peering을 이용한 네트워크 연결은 규모가 큰 네트워크를 다루게 될 수록 운영 관리가 어려움

- VPC Peering은 VPC간 연결 시 전이적 라우팅 제약



- 전체 VPC를 각각 Peering으로 연동할 경우 관리 복잡성 증가

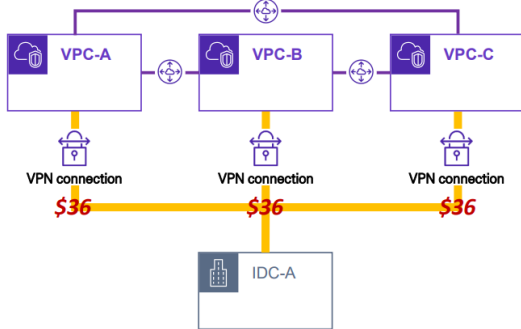


peering은 관리하기 힘들다.

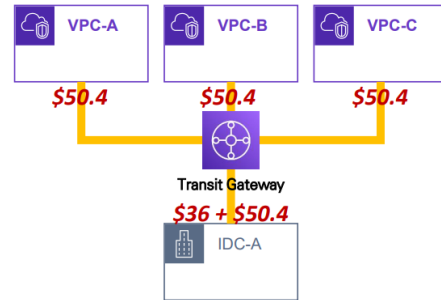
#### 4. Transit Gateway(3/3)

##### VPC Peering vs Transit Gateway (2) – 가격 비교

- VPC Peering & Site to Site VPC : **\$108**



- Transit Gateway & Site to Site VPC : **\$237**



transit gateway가 기본 비용은 더 비싸지만 장기적으로 확장이 많이 되면 더 싸게 먹힌다.

## Transit Gateway 실습

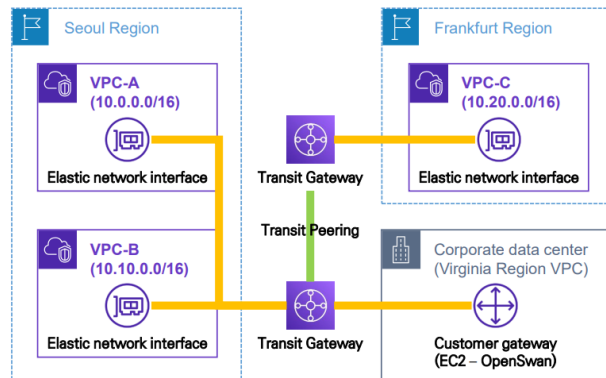
### Hands-on Lab #10 – Transit Gateway & VPN 구성하기

#### 실습 개요

- On-Premise Data Center는 Virginia Region의 VPC로 대체
- 고객사의 물리 VPN 장비는 EC2에 OpenSwan Tool로 구성

#### 실습 리소스

- Region : us-east-1, ap-northeast-2, eu-central-1
- VPC : lab-edu-vpc-ap-01, lab-edu-vpc-eu-01, lab-edu-vpc-us-01 (On-Premise Data Center 가정)
- EC2 : lab-edu-ec2-network-ap-01~02, lab-edu-ec2-network-eu, lab-edu-ec2-network-us, lab-edu-ec2-openswan-us
- Transit Gateway : lab-edu-tgw-ap, lab-edu-tgw-eu
- Site to Site VPN : lab-edu-s2svpg-ap, lab-edu-s2scgw-us



## Lab Environment Configuration

### 1. VPC Peering Resource 삭제

- VPC 콘솔 메인 화면 → 피어링 연결 리소스 탭 → "lab-edu-peering-ap01-ap02" 선택
- '작업' 버튼 클릭 → '피어링 연결 삭제' 버튼 클릭 → '삭제' 입력 → '삭제' 버튼 클릭
- VPC 콘솔 메인 화면 → 피어링 연결 리소스 탭 → "lab-edu-peering-ap01-eu01" 선택
- '작업' 버튼 클릭 → '피어링 연결 삭제' 버튼 클릭 → '삭제' 입력 → '삭제' 버튼 클릭

### 2. Transit Gateway Subnet 생성

- VPC 메인 콘솔 화면 → Subnet 리소스 탭 → "Subnet 생성" 버튼 클릭

- 아래 서브넷 자원 명세서를 참고하여 생성 정보 입력

	Transit Gateway Subnet 01	Transit Gateway Subnet 02
VPC_ID	leb-edu-vpc-ap-01	leb-edu-vpc-ap-01
Subnet_Name	lab-edu-sub-tgw-01	lab-edu-sub-tgw-02
Availability_Zone	ap-northeast-2a	ap-northeast-2c
IPv4 CIDR	10.0.255.224/28	10.0.255.240/28

## 서울 리전 ↔ 서울 리전 Transit Gateway 생성

### 1. Transit Gateway 생성

- VPC 콘솔 메인 화면 → Transit Gateway 리소스 탭 → "Transit Gateway 생성" 버튼 클릭

- Transit Gateway 생성 정보 입력

- 이름: lab-edu-tgw-ap
- Amazon ASN: 64512

'Transit Gateway 생성' 버튼 클릭

![alt text](./img/transit\_gateway\_ap\_ap\_01.png)

### 2. Transit Gateway Attach 생성 (1/2)

- VPC 콘솔 메인 화면 → Transit Gateway Attach 리소스 탭 → "Transit Gateway Attach 생성" 버튼 클릭

- Transit Gateway Attach 생성 정보 입력

- 이름: lab-edu-tgw-att-ap01
- Transit Gateway ID: lab-edu-tgw-ap
- 연결 유형: VPC
- VPC ID: lab-edu-vpc-ap-01
- Subnet ID:
  - ap-northeast-2a: lab-edu-sub-tgw-01
  - ap-northeast-2c: lab-edu-sub-tgw-02
- 'Transit Gateway Attach 생성' 버튼 클릭

### 3. Transit Gateway Attach 생성 (2/2)

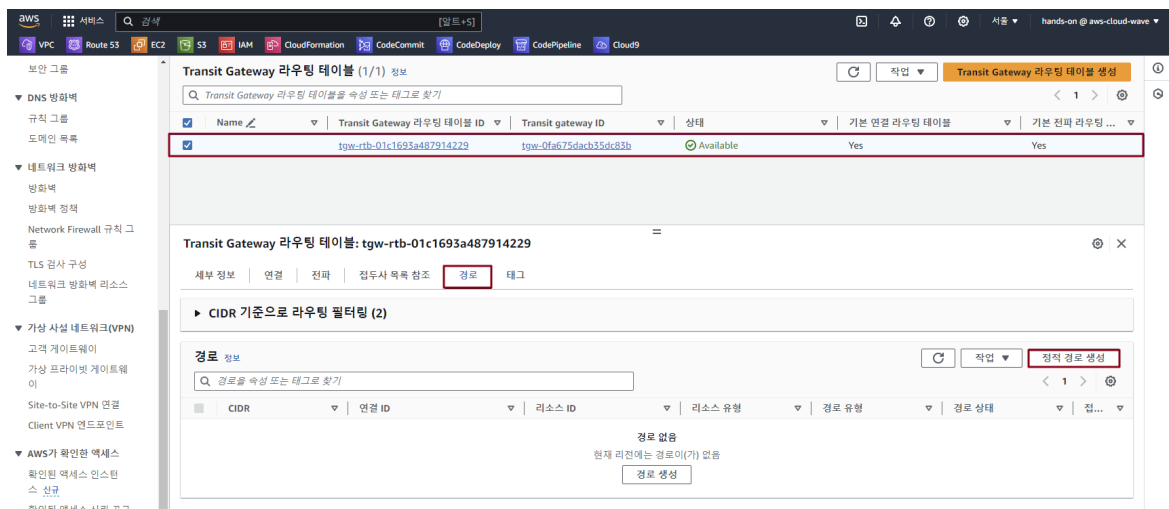
- VPC 콘솔 메인 화면 → Transit Gateway Attach 리소스 탭 → "Transit Gateway Attach 생성" 버튼 클릭

- Transit Gateway Attach 생성 정보 입력

- 이름: lab-edu-tgw-att-ap02
- Transit Gateway ID: lab-edu-tgw-ap
- 연결 유형: VPC
- VPC ID: lab-edu-vpc-ap-02
- Subnet ID:
  - ap-northeast-2a: lab-edu-sub-tgw-01
  - ap-northeast-2c: lab-edu-sub-tgw-02
- 'Transit Gateway Attach 생성' 버튼 클릭

#### 4. Transit Gateway Routing Table 설정

- VPC 콘솔 메인 화면 → Transit Gateway 라우팅 테이블 리소스 탭 → "Routing Table" 선택
- '경로' 탭 → '정적 경로 생성' 버튼 클릭



- 정적 경로 생성 정보 입력 → '정적 경로 생성' 버튼 클릭
  - CIDR: 10.10.0.0/16
  - 연결 선택: lab-edu-tgw-att-ap02

#### 5. VPC Routing Table 수정

- VPC 콘솔 메인 화면 → 라우팅 테이블 탭 → "lab-edu-rtb-pri-01" 선택 → '라우팅' 탭 → '라우팅 편집' 버튼 클릭
- 라우팅 테이블 경로 생성 정보 입력
  - '라우팅 추가' 버튼 클릭
  - 대상: 10.10.0.0/16
  - 대상: Transit Gateway (lab-edu-tgw-att-ap01)
  - '변경 사항 저장' 버튼 클릭
- VPC 콘솔 메인 화면 → 라우팅 테이블 탭 → "lab-edu-rtb-2nd-pri-01" 선택 → '라우팅' 탭 → '라우팅 편집' 버튼 클릭
- 라우팅 테이블 경로 생성 정보 입력
  - '라우팅 추가' 버튼 클릭
  - 대상: 10.0.0.0/16

- 대상: Transit Gateway (lab-edu-tgw-att-ap02)
- '변경 사항 저장' 버튼 클릭

## 6. Network 통신 테스트

- EC2 메인 콘솔 화면으로 이동 → 인스턴스 리소스 탭 → 'lab-edu-ec2-network-2nd-ap' 선택 → Private IP 주소 복사
- Cloud9 IDE Terminal 화면으로 이동 → ssh 명령어 실행

```
ssh web-server
```

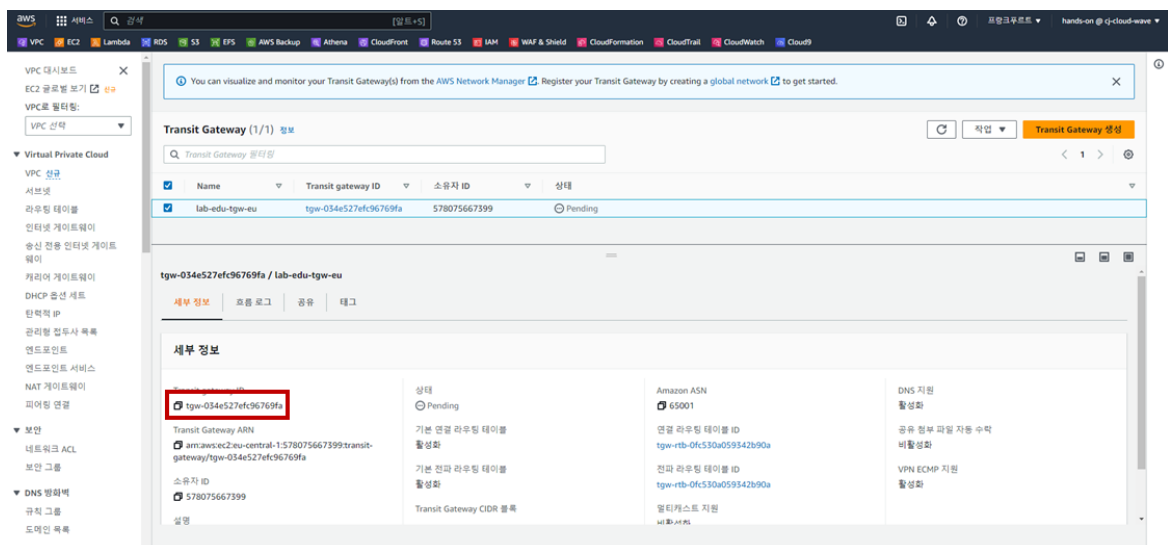
- ICMP 통신 테스트 진행

```
ping {2ND_VPC_NETWORK_SERVER_PRIVATE_IP}
PING 10.10.40.140 (10.10.40.140) 56(84) bytes of data.
64 bytes from 10.10.40.140: icmp_seq=1 ttl=254 time=0.990 ms
64 bytes from 10.10.40.140: icmp_seq=2 ttl=254 time=0.926 ms
64 bytes from 10.10.40.140: icmp_seq=3 ttl=254 time=0.907 ms
64 bytes from 10.10.40.140: icmp_seq=4 ttl=254 time=0.946 ms
```

## 서울 리전 ↔ 프랑크푸르트 리전 Transit Gateway 생성

### 1. 프랑크푸르트 리전에 Transit Gateway 생성

- 프랑크푸르트 리전으로 이동 → VPC 콘솔 메인 화면 → Transit Gateway 리소스 탭 → "Transit Gateway 생성" 버튼 클릭
- Transit Gateway 생성 정보 입력
  - 이름: lab-edu-tgw-eu
  - Amazon ASN: 64513
  - 'Transit Gateway 생성' 버튼 클릭
- Transit Gateway ID 정보 메모장에 저장



### 2. 프랑크푸르트 리전 Transit Gateway Attach 생성 (1/2)

- 프랑크푸르트 리전으로 이동 → VPC 콘솔 메인 화면 → Transit Gateway Attach 리소스 탭 → "Transit Gateway Attach 생성" 버튼 클릭

- Transit Gateway Attach 생성 정보 입력
  - 이름: lab-edu-tgw-att-eu
  - Transit Gateway ID: lab-edu-tgw-eu
  - 연결 유형: VPC
  - VPC ID: lab-edu-vpc-eu-01
  - Subnet ID:
    - ap-northeast-2a: lab-edu-sub-eu-tgw-01
    - ap-northeast-2c: lab-edu-sub-eu-tgw-02

'Transit Gateway Attach 생성' 버튼 클릭

### 3. 서울 리전 Transit Gateway Attach 생성 (2/2)

- 서울 리전으로 이동 → VPC 콘솔 메인 화면 → Transit Gateway Attach 리소스 탭 → "Transit Gateway Attach 생성" 버튼 클릭
- Transit Gateway Attach 생성 정보 입력
  - 이름: lab-edu-tgw-att-peering-eu
  - Transit Gateway ID: lab-edu-tgw-ap
  - 연결 유형: Peering Connection
  - 리전: eu-central-1
  - Transit Gateway(수락자): {FRANKFURT\_REGION\_TRANSIT\_GATEWAY\_ID}
  - 'Transit Gateway Attach 생성' 버튼 클릭

VPC > Transit Gateway Attachment > Transit Gateway Attachment 생성

## Transit Gateway Attachment 생성 정보

Transit Gateway(TGW)는 동일한 AWS 계정 내 또는 AWS 계정 간에 연결(VPC 및 VPN)을 상호 연결하는 네트워크 전송 허브입니다.

**세부 정보**

이름 태그 - 선택 사항  
키가 Name으로 설정되고 값이 지정된 문자열로 설정된 태그를 생성합니다.

Transit gateway ID 정보

연결 유형 정보

**피어링 연결**  
피어링 연결 연결을 선택하고 구성합니다.

**계정 정보**

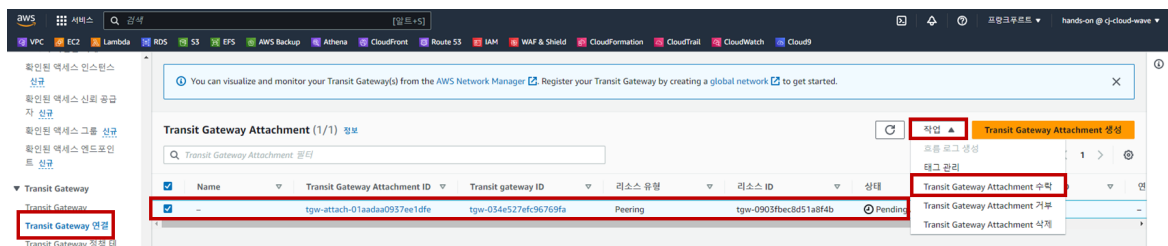
☒ 내 계정  
☐ 다른 계정

**리전 정보**

**Transit Gateway(수락자) 정보**

#### 4. 프랑크푸르트 리전 Transit Gateway Peering 수락

- 프랑크푸르트 리전으로 이동 → VPC 콘솔 메인 화면 → Transit Gateway Attach 리소스 탭
- 'Peering Acceptance' 상태의 Attachment 선택 → 작업 → Transit Gateway Attachment 수락 → 수



#### 5. Transit Gateway Routing Table 설정

- 서울 리전으로 이동 → VPC 콘솔 메인 화면 → Transit Gateway 라우팅 테이블 리소스 탭 → "Routing Table" 선택
- '경로' 탭 → '정적 경로 생성' 버튼 클릭
- 정적 경로 생성 정보 입력 → '정적 경로 생성' 버튼 클릭
  - CIDR: 10.30.0.0/16



- 연결 선택: lab-edu-tgw-att-peering-eu
- 프랑크푸르트 리전으로 이동 → VPC 콘솔 메인 화면 → Transit Gateway 라우팅 테이블 리소스 탭 → "Routing Table" 선택
- '경로' 탭 → '정적 경로 생성' 버튼 클릭
- 정적 경로 생성 정보 입력 → '정적 경로 생성' 버튼 클릭
  - CIDR: 10.0.0.0/16
  - 연결 선택: lab-edu-tgw-att-peering-eu

## 6. VPC Routing Table 수정

- 서울 리전으로 이동 → VPC 콘솔 메인 화면 → 라우팅 테이블 탭 → "lab-edu-rtb-pri-01" 선택 → '라우팅' 탭 → '라우팅 편집' 버튼 클릭
- 라우팅 테이블 경로 생성 정보 입력
  - '라우팅 추가' 버튼 클릭
  - 대상: 10.30.0.0/16
  - 대상: Transit Gateway (lab-edu-tgw-att-ap01)
  - '변경 사항 저장' 버튼 클릭
- 프랑크푸르트 리전으로 이동 → VPC 콘솔 메인 화면 → 라우팅 테이블 탭 → "lab-edu-rtb-eu-pri" 선택 → '라우팅' 탭 → '라우팅 편집' 버튼 클릭
- 라우팅 테이블 경로 생성 정보 입력
  - '라우팅 추가' 버튼 클릭
  - 대상: 10.0.0.0/16
  - 대상: Transit Gateway (lab-edu-tgw-att-eu)
  - '변경 사항 저장' 버튼 클릭

## 7. Network 통신 테스트

- 프랑크푸르트 리전으로 이동 → EC2 메인 콘솔 화면 → 인스턴스 리소스 탭 → 'lab-edu-ec2-network-eu' 선택 → Private IP 주소 복사
- Cloud9 IDE Terminal 화면으로 이동 → ssh 명령어 실행

```
ssh web-server
```

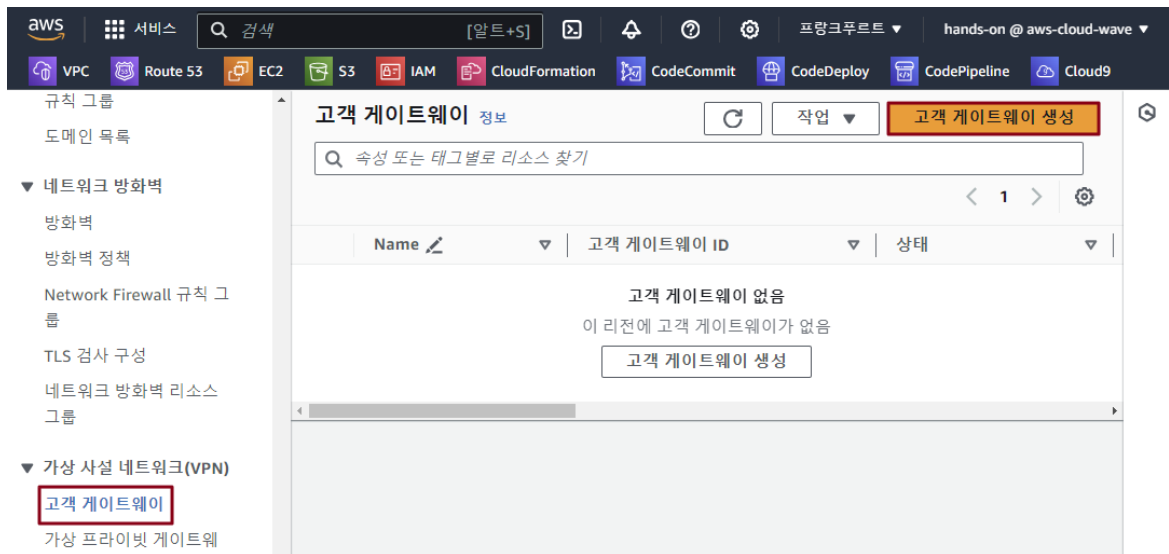
- ICMP 통신 테스트 진행

```
ping {FRANKFURT_REGION_NETWORK_SERVER_PRIVATE_IP}
PING 10.30.40.56 (10.30.40.56) 56(84) bytes of data.
64 bytes from 10.30.40.56: icmp_seq=1 ttl=252 time=241 ms
64 bytes from 10.30.40.56: icmp_seq=2 ttl=252 time=240 ms
64 bytes from 10.30.40.56: icmp_seq=3 ttl=252 time=239 ms
64 bytes from 10.30.40.56: icmp_seq=4 ttl=252 time=239 ms
```

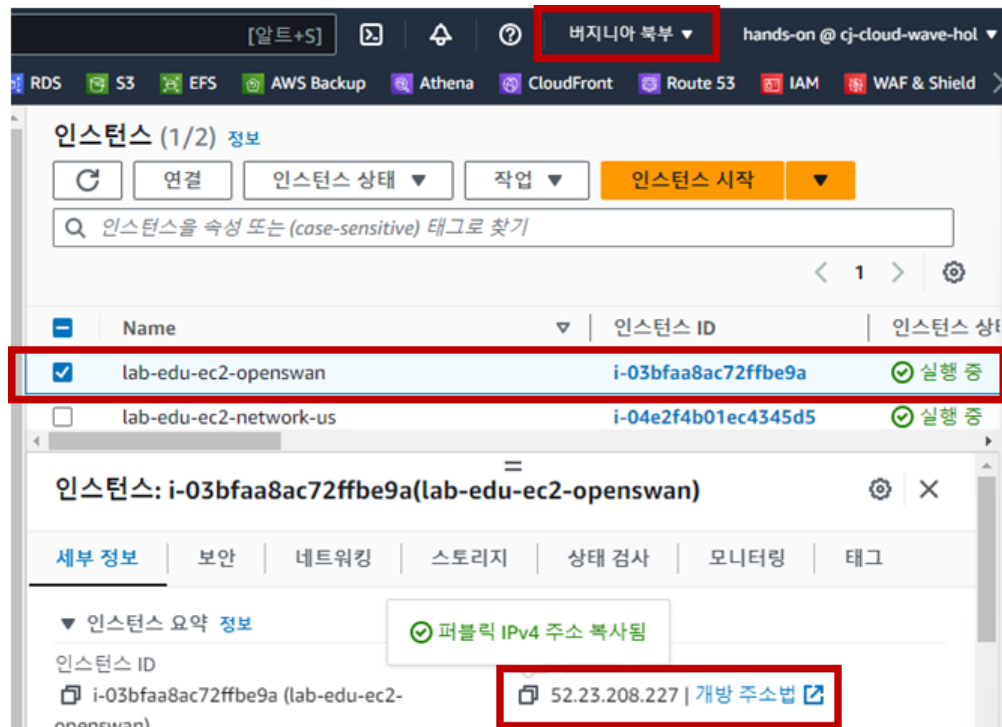
## 서울 리전 ↔ 버지니아 리전 Transit Gateway & Site to Site VPN 연동

### 1. 서울 리전 Custom Gateway 생성

- 서울 리전으로 이동 → VPC 콘솔 메인 화면 → 고객 게이트웨이 리소스 탭 → "고객 게이트웨이 생성" 버튼 클릭



- 버지니아 리전으로 이동 → 인스턴스 콘솔 메인 화면 → 인스턴스 리소스 탭 → 'lab-edu-ec2-openswan-us' 선택 → Public IP 복사



- 고객 게이트웨이 생성 정보 입력
  - 이름: lab-edu-cgw-us
  - BGP ASN: 65100
  - IP Address: {VIRGINIA\_REGION\_OPENSWAN\_SERVER\_PUBLIC\_IP}
  - '고객 게이트웨이 생성' 버튼 클릭

VPC > 고객 게이트웨이 > 고객 게이트웨이 생성

## 고객 게이트웨이 생성 정보

고객 게이트웨이는 AWS에서 생성하는 리소스로, 온프레미스 네트워크의 고객 게이트웨이 디바이스를 나타냅니다.

### 세부 정보

**이름 태그 - 선택 사항**  
이름인 키와 사용자가 지정하는 값을 사용하여 태그를 생성합니다.

customer-gateway-01

값의 길이가 256자 이하여야 합니다.

**BGP ASN 정보**  
고객 게이트웨이 디바이스의 ASN입니다.

65000

값은 1~2147483647의 범위여야 합니다.

**IP 주소 정보**  
고객 게이트웨이 디바이스의 외부 인터페이스에 대한 IP 주소를 지정합니다.

192.0.2.1

**인증서 ARN**  
AWS Certificate Manager(ACM)에 프로비저닝된 프라이빗 인증서의 ARN입니다.

인증서 ARN 선택

**디바이스 - 선택 사항**  
고객 게이트웨이 디바이스의 이름을 입력합니다.

디바이스 이름 입력

### 태그

태그는 사용자가 AWS 리소스에 할당하는 레이블입니다. 각 태그는 키와 서명된 값으로 구성됩니다. 태그를 사용하여 리소스를 검색 및 식별할 수 있습니다.

## 2. 서울 리전 Site to Site VPN 리소 생성

- 서울 리전으로 이동 → VPC 콘솔 메인 화면 → 'Site to Site VPC 연결' 리소스 탭 → "VPN 연결 생성" 버튼 클릭
- 'Site to Site VPC 연결' 생성 정보 입력
  - 이름: lab-edu-s2svpn-ap
  - 대상 게이트웨이 유형: Transit Gateway
  - Transit Gateway: lab-edu-tgw-ap
  - 고객 게이트웨이: 기존
  - 고객 게이트웨이 ID: lab-edu-cgw-us
  - 라우팅 옵션: 정적

VPC > VPN 연결 > VPN 연결 생성

## VPN 연결 생성 정보

site-to-site VPN 연결에 사용하려는 리소스 및 추가 구성 옵션을 선택합니다.

세부 정보

**이름 태그 - 선택 사항**  
'이름'인 키와 사용자가 지정하는 값을 사용하여 태그를 생성합니다.  
  
값의 길이가 256자 이하여야 합니다.

**대상 게이트웨이 유형 정보**  
☐ 가상 프라이빗 게이트웨이  
☒ Transit Gateway  
☐ 연결되지 않음

**Transit Gateway**

**고객 게이트웨이 정보**  
☒ 기존  
☐ 신규

**고객 게이트웨이 ID**

**라우팅 옵션 정보**  
☐ 동적(BGP 필요)  
☒ 정적

- 로컬 IPv4 네트워크 CIDR: 10.30.0.0/16
- 원격 IPv4 네트워크 CIDR: 10.0.0.0/16
- 터널 1, 2 옵션의 사전 공유 키: cloudwave
- 'VPN 연결 생성' 버튼 클릭

**로컬 IPv4 네트워크 CIDR - 선택 사항**  
VPN 터널을 통해 통신할 수 있는 고객 게이트웨이(온프레미스) 측의 IPv4 CIDR 범위입니다. 기본값은 0.0.0.0/0입니다.

**원격 IPv4 네트워크 CIDR - 선택 사항**  
VPN 터널을 통해 통신할 수 있는 AWS 측의 IPv4 CIDR 범위입니다. 기본값은 0.0.0.0/0입니다.

**외부 IP 주소 유형 정보**

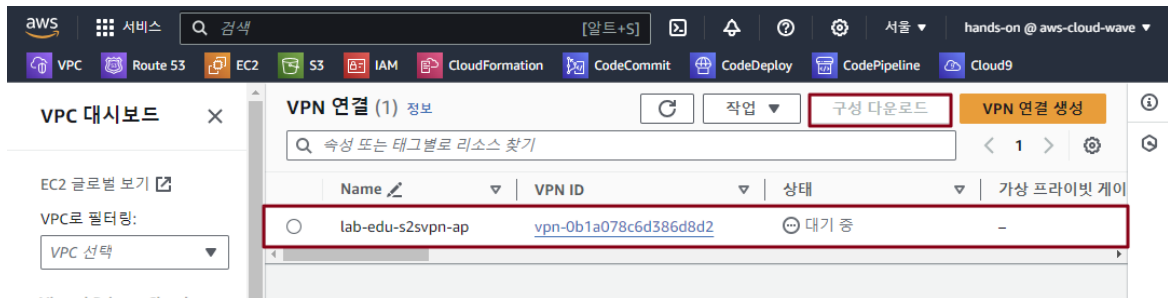
▼ **터널 1 옵션 - 선택 사항 정보**

**터널 1의 내부 IPv4 CIDR**  
  
169.254.0.0/16 범위의 크기 / 30인 IPv4 CIDR 블록.

**터널 1의 사전 공유 키**  
가상 프라이빗 게이트웨이와 고객 게이트웨이 간에 초기 인증을 설정하기 위한 사전 공유 키(PSK)입니다.  
  
사전 공유 키는 8~64자여야 합니다. 유효한 문자는 A~Z, a~z, 0~9, \_ 및 .입니다. 키는 0으로 시작할 수 없습니다.

### 3. 고객 게이트웨이 구성 다운로드

- 서울 리전으로 이동 → VPC 콘솔 메인 화면 → 'Site to Site VPC 연결' 리소스 탭 → "lab-edu-s2svpn-ap" 선택 → '구성 다운로드' 버튼 클릭



- 구성 다운로드 설정 정보 입력
  - 공급 업체: Openswan
  - 플랫폼: Openswan
  - 소프트웨어: Openswan 2.6.38+
  - IKE 버전: ikev1
  - '다운로드' 버튼 클릭

구성 다운로드

고객 게이트웨이를 기반으로 다운로드할 샘플 구성을 선택합니다. 해당 항목은 샘플이며 고급 알고리즘, 인증서 및/또는 IPv6를 사용하려면 수정해야 합니다.

공급 업체  
고객 게이트웨이 디바이스의 제조업체입니다(예: Cisco Systems, Inc.).

Openswan

플랫폼  
고객 게이트웨이 디바이스의 클래스입니다(예: J-Series).

Openswan

소프트웨어  
고객 게이트웨이 디바이스에서 실행 중인 운영 체제입니다(예: ScreenOS).

Openswan 2.6.38+

IKE 버전  
VPN 연결에 사용 중인 IKE 버전입니다.

ikev1

취소

다운로드

### 4. Openswan 서버 설정

- 버지니아 리전으로 이동 → EC2 콘솔 메인 화면 → 인스턴스 리소스 탭 → 'lab-edu-ec2-openswan-us' 선택 → '연결' 버튼 클릭

- 'Session Manager' 탭으로 이동 → '연결' 버튼 클릭
- sysctl.conf 파일 설정

```
sudo su -
```

```
vim /etc/sysctl.conf
```

```
net.ipv4.ip_forward = 1
net.ipv4.conf.default.rp_filter = 0
net.ipv4.conf.default.accept_source_route = 0
```

- 구성 다운로드 파일 열기 → 설정 내용 복사

-----  
IPSEC Tunnel #1  
-----

This configuration assumes that you already have a default openswan installation in place on the Amazon Linux operating system (but may also work with other distros as well)

- 1) Open /etc/sysctl.conf and ensure that its values match the following:  

```
net.ipv4.ip_forward = 1
net.ipv4.conf.default.rp_filter = 0
net.ipv4.conf.default.accept_source_route = 0
```
- 2) Apply the changes in step 1 by executing the command 'sysctl -p'
- 3) Open /etc/ipsec.conf and look for the line below. Ensure that the # in front of the line has been removed, then save and exit the file.  

```
#include /etc/ipsec.d/*.conf
```
- 4) Create a new file at /etc/ipsec.d/aws.conf if doesn't already exist, and then open it. Append the following configuration to the end in the file:  

```
#leftsubnet= is the local network behind your openswan server, and you will need to replace the <LOCAL NETWORK> below with this value (don't include the brackets). If you have multiple subnets, you can use 0.0.0.0/0 instead.
#rightsubnet= is the remote network on the other side of your VPN tunnel that you wish to have connectivity with, and you will need to replace <REMOTE NETWORK> with this value (don't include brackets).
```

```
conn Tunnel1
    authby=secret
    auto=start
    left=%defaultroute
    leftid=53.200.21.2
    right=15.165.247.169
    type=tunnel
    ikelifetime=8h
    keylife=1h
    phase2alg=aes128-sha1;modp1024
    ike=aes128-sha1;modp1024
    auth=esp
    keyingtries=%forever
    keyexchange=ike
    leftsubnet=<LOCAL NETWORK>
    rightsubnet=<REMOTE NETWORK>
    dpddelay=10
    dpdtimeout=30
    dpdaction=restart_by_peer
```

- aws.conf 파일 설정 (구성 다운로드 파일 다운로드 내용 붙여넣기 → 수정)

```
vim /etc/ipsec.d/aws.conf
```

```
conn Tunnel1
    authby=secret
    auto=start
    left=%defaulttroute
    leftid=52.23.208.227
    right=13.125.161.188
    type=tunnel
    ikelifetime=8h
    keylife=1h
    phase2alg=aes128-sha1;modp1024
    ike=aes128-sha1;modp1024
    auth=esp # 삭제!!
    keyingtries=%forever
    keyexchange=ike
    leftsubnet=<LOCAL NETWORK> # <LOCAL NETWORK> 삭제 → 10.30.0.0/16 입력
    rightsubnet=<REMOTE NETWORK> # <REMOTE NETWORK> 삭제 → 10.0.0.0/16 입력
    dpddelay=10
    dpdtimeout=30
    dpdaction=restart_by_peer
```

- 구성 다운로드 파일 열기 → 설정 내용 복사

-----  
IPSEC Tunnel #1  
-----

This configuration assumes that you already have a default openswan installation in place on the Amazon Linux operating system (but may also work with other distros as well)

- 1) Open `/etc/sysctl.conf` and ensure that its values match the following:  
`net.ipv4.ip_forward = 1`  
`net.ipv4.conf.default.rp_filter = 0`  
`net.ipv4.conf.default.accept_source_route = 0`
- 2) Apply the changes in step 1 by executing the command `'sysctl -p'`
- 3) Open `/etc/ipsec.conf` and look for the line below. Ensure that the `#` in front of the line has been removed, then save and exit the file.  
`#include /etc/ipsec.d/*.conf`
- 4) Create a new file at `/etc/ipsec.d/aws.conf` if doesn't already exist, and then open it. Append the following configuration to the end in the file:  
`#leftsubnet=` is the local network behind your openswan server, and you will need to replace the `<LOCAL NETWORK>` below with this value (don't include the brackets). If you have multiple subnets, you can use `0.0.0.0/0` instead.  
`#rightsubnet=` is the remote network on the other side of your VPN tunnel that you wish to have connectivity with, and you will need to replace `<REMOTE NETWORK>` with this value (don't include brackets).

```
conn Tunnel1
    authby=secret
    auto=start
    left=%defaultroute
    leftid=53.200.21.2
    right=15.165.247.169
    type=tunnel
    ikelifetime=8h
    keylife=1h
    phase2alg=aes128-sha1;modp1024
    ike=aes128-sha1;modp1024
    auth=esp
    keyingtries=%forever
    keyexchange=ike
    leftsubnet=<LOCAL NETWORK>
    rightsubnet=<REMOTE NETWORK>
    dpddelay=10
    dpdtimeout=30
    dpdaction=restart_by_peer
```

- 5) Create a new file at `/etc/ipsec.d/aws.secrets` if it doesn't already exist, and append this line to the file (be mindful of the spacing!):  

`53.200.21.2 15.165.247.169: PSK "cloudwave"`

- aws.secrets 파일 설정 (구성 다운로드 파일 다운로드 내용 붙여넣기)

```
vim /etc/ipsec.d/aws.secrets
```

```
53.200.21.2 15.165.247.169: PSK "cloudwave"
```

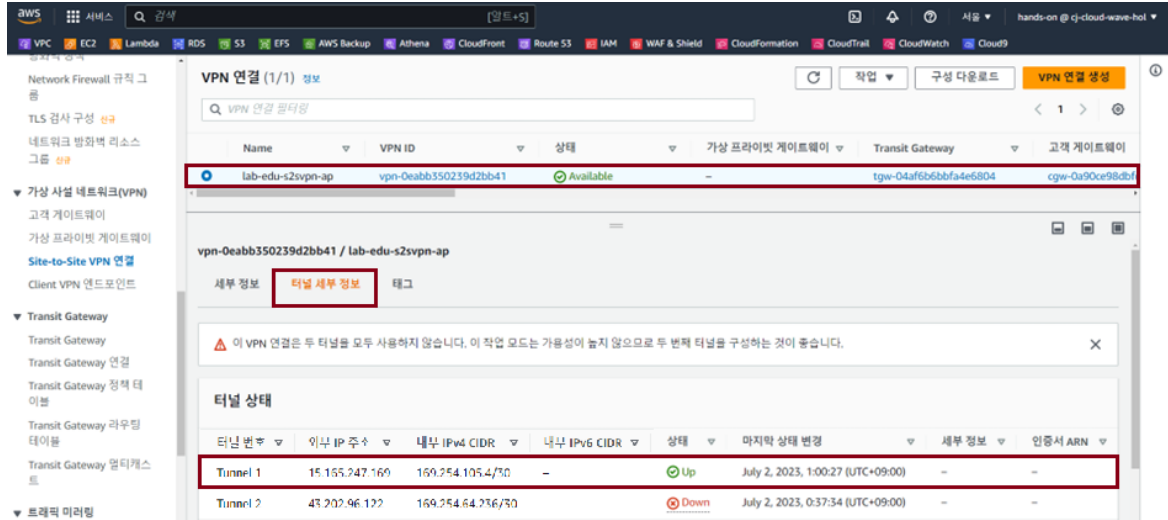
- Openswan 재시작

```
systemctl restart network
```



```
systemctl restart ipsec.service
```

```
systemctl status ipsec.service
```



## 5. Transit Gateway Routing Table 설정

- 서울 리전으로 이동 → VPC 콘솔 메인 화면 → Transit Gateway 라우팅 테이블 리소스 탭 → "Routing Table" 선택
- '경로' 탭 → '정적 경로 생성' 버튼 클릭
- 정적 경로 생성 정보 입력 → '정적 경로 생성' 버튼 클릭
  - CIDR: 10.20.0.0/16
  - 연결 선택: lab-edu-s2svpn-ap

## 6. VPC Routing Table 수정

- 서울 리전으로 이동 → VPC 콘솔 메인 화면 → 라우팅 테이블 탭 → "lab-edu-rtb-pri-01" 선택 → '라우팅' 탭 → '라우팅 편집' 버튼 클릭
- 라우팅 테이블 경로 생성 정보 입력
  - '라우팅 추가' 버튼 클릭
  - 대상: 10.20.0.0/16
  - 대상: Transit Gateway (lab-edu-tgw-att-ap01)
  - '변경 사항 저장' 버튼 클릭
- 버지니아 리전으로 이동 → VPC 콘솔 메인 화면 → 라우팅 테이블 탭 → "lab-edu-rtb-us-pri" 선택 → '라우팅' 탭 → '라우팅 편집' 버튼 클릭
- 라우팅 테이블 경로 생성 정보 입력
  - '라우팅 추가' 버튼 클릭
  - 대상: 10.0.0.0/16
  - 대상: 인스턴스 (lab-edu-ec2-openswan-us)
  - '변경 사항 저장' 버튼 클릭

## 7. Network 통신 테스트

- ```
ssh web-server
```

- ```
ping [VIRGINIA|REGION_NETWORK_SERVER_PRIVATE_IP]
PING 10.20.40.196 (10.20.40.196) 56(84) bytes of data:
64 bytes from 10.20.40.196: icmp_seq=1 ttl=253 time=188 ms
64 bytes from 10.20.40.196: icmp_seq=2 ttl=253 time=187 ms
64 bytes from 10.20.40.196: icmp_seq=3 ttl=253 time=187 ms
64 bytes from 10.20.40.196: icmp_seq=4 ttl=253 time=188 ms
```

## Endpoint 정의

VPC 내부 리소스가 AWS에서 제공하는 다른 서비스에 접근할 때 Internet Gateway, Nat Gateway를 거치지 않고 안전한 내부망을 사용할 수 있도록 서비스하는 리소스

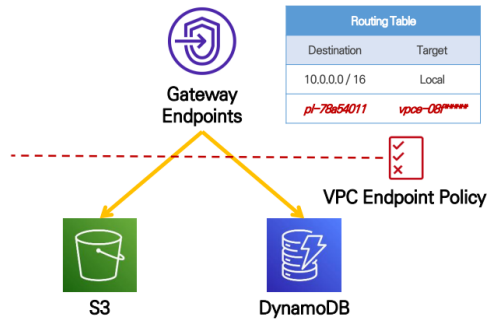
- 

## Endpoint 종류

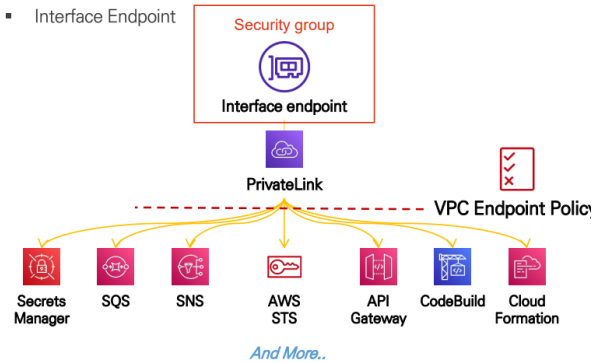
## 5. VPC Endpoint (2/2)

VPC Endpoint는 **Gateway Endpoint**, **Interface Endpoint**로 나뉘고, Endpoint 방식에 따라 접근 가능 서비스 종류 상이

### Gateway Endpoint



### Interface Endpoint



gateway endpoint 무료 interface는 유료

## Endpoint 실습

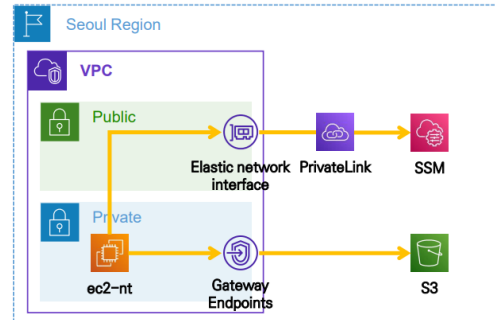
### Hands-on Lab #11 – VPC Endpoint 구성하기

#### 실습 개요

- Private 서브넷에 위치한 EC2를 이용 VPC endpoint 테스트
- NAT Gateway에 대한 라우팅 테이블 정보를 삭제 후 Private 통신 과정 체크
- System Manager Endpoint 이용 인터넷이 안되는 환경 콘솔 접속

#### 실습 리소스

- Region : ap-northeast-2
- VPC : lab-edu-vpc-ap-01
- VPC Endpoint : lab-edu-endpoint-s3
- EC2 : lab-edu-ec2-network-01
- S3 : lab-edu-s3-endpoint-[ACCOUNT\_NUMBER]



## Lab Environment Configuration

### 1. 이미지 파일 저장용 Amazon S3 Bucket 생성

- Cloud9 IDE Terminal 화면으로 이동

```
cd ~/environment
sh cloud-wave-workspace/scripts/upload_images_to_s3.sh
```

### 2. Network 테스트용 EC2 서버 2개 추가 생성

- Cloud9 IDE Terminal 화면에서 이어서 작업 진행

```
cd ~/environment
```

```
sh cloud-wave-workspace/scripts/create_ec2_instance.sh
```

### 3. Network 테스트용 EC2 서버 접속 및 S3 버킷 조회

- Cloud9 IDE Terminal 화면에서 이어서 작업 진행
- Network-01 EC2 접속 → S3 Bucket Object 조회

```
ssh network-01
```

```
aws s3 ls
```

```
aws s3 ls s3://lab-edu-bucket-image-9*****0/
```

- Cloud9 IDE 새로운 Terminal 화면 열기
- Network-02 EC2 접속 → S3 Bucket Object 조회

```
ssh network-02
```

```
aws s3 ls
```

```
aws s3 ls s3://lab-edu-bucket-image-9*****0/
```

### S3 버킷용 VPC Endpoint 생성 (Gateway Type)

#### 1. VPC Endpoint 생성

- VPC 콘솔 메인 화면 → 엔드포인트 리소스 탭 → "엔드포인트 생성" 버튼 클릭
- 엔드포인트 생성 정보 입력
  - 이름: lab-edu-endpoint-s3
  - 서비스 이름 / 소유자 / 유형: com.amazonaws.ap-northeast-2.s3 / amazon / Gateway

aws 서비스 검색 [알트+S] 서울 hands-on @ aws-cloud-wave

VPC > 엔드포인트 > 엔드포인트 생성

### 엔드포인트 생성 정보

VPC 엔드포인트에는 인터페이스 엔드포인트, 게이트웨이 로드 밸런서 엔드포인트 및 게이트웨이 엔드포인트의 세 가지 유형이 있습니다. 인터페이스 엔드포인트와 게이트웨이 로드 밸런서 엔드포인트는 AWS PrivateLink에 의해 구동되며 ENI(탄력적 네트워크 인터페이스)를 서비스로 가는 트래픽의 진입점으로 사용합니다. 인터페이스 엔드포인트는 일반적으로 이러한 서비스에 연결된 퍼블릭 및 프라이빗 DNS 이름을 사용하여 액세스하며 게이트웨이 엔드포인트와 게이트웨이 로드 밸런서 엔드포인트는 서비스로 향하는 트래픽에 대한 라우팅 테이블의 경로에 대한 대상으로 사용됩니다.

#### 엔드포인트 설정

이름 태그 - 선택 사항  
'Name' 키와 귀하가 지정하는 값을 포함하는 태그를 생성합니다.

서비스 범주  
서비스 범주 선택

☒ AWS 서비스  
Amazon에서 제공하는 서비스

☐ PrivateLink Ready 파트너 서비스  
AWS Service Ready 지정된 서비스

☐ AWS Marketplace 서비스  
AWS Marketplace를 통해 구매한 서비스

☐ EC2 인스턴스 연결 엔드포인트  
프라이빗 서브넷의 리소스에 연결할 수 있는 탄력적 네트워크 인터페이스

☐ 다른 엔드포인트 서비스  
서비스 이름별로 공유된 서비스 찾기

#### 서비스 (1/3)

검색

서비스 이름	소유자	유형
<input checked="" type="radio"/> com.amazonaws.ap-northeast-2.s3	amazon	Gateway
<input type="radio"/> com.amazonaws.ap-northeast-2.s3	amazon	Interface
<input type="radio"/> com.amazonaws.ap-northeast-2.s3-outposts	amazon	Interface

- VPC: lab-edu-vpc-ap-01
- 라우팅 테이블: lab-edu-rtb-pri-01, lab-edu-rtb-pri-02

aws 서비스 검색 [알트+S] 서울 hands-on @ aws-cloud-wave

VPC 엔드포인트를 생성할 VPC를 선택

VPC 엔드포인트를 생성할 VPC입니다.

#### 라우팅 테이블 (2/6) 정보

검색

이름	라우팅 테이블 ID	기본
<input type="checkbox"/> -	rtb-00e0bb29c9e64359b	예
<input type="checkbox"/> lab-edu-rtb-db	rtb-09fa61889c42d4c97 (lab-edu-rtb-db)	아니요
<input checked="" type="checkbox"/> lab-edu-rtb-pri-01	rtb-0f4c78984132dbbc4 (lab-edu-rtb-p...	아니요
<input checked="" type="checkbox"/> lab-edu-rtb-pri-02	rtb-0fd350aa223b82e13 (lab-edu-rtb-...	아니요
<input type="checkbox"/> lab-edu-rtb-pub	rtb-086d7008df457794c (lab-edu-rtb-...	아니요
<input type="checkbox"/> lab-edu-rtb-tgw	rtb-0b4c504a39caa5895 (lab-edu-rtb-t...	아니요

- 정책: 전체 액세스

- '엔드포인트 생성' 버튼 클릭

정책 정보

VPC 엔드포인트 정책은 서비스 액세스를 제어합니다.

전체 액세스

Amazon Web Services 계정부터 이 Amazon Web Services 서비스의 모든 리소스까지 자격 증명을 사용하여 VPC 내에 있는 모든 사용자 또는 서비스의 액세스를 허용합니다. IAM 사용자 정책, VPC 엔드포인트 정책 및 Amazon Web Services 서비스별 정책(예: Amazon S3 버킷 정책, 모든 S3 ACL 정책) 등 모든 정책은 성공적인 액세스를 위해 필요한 권한을 부여해야 합니다.

사용자 지정

정책 생성 도구를 사용하여 정책을 생성한 다음 아래에 생성된 정책을 붙여 넣습니다.

1

JSON

1줄, 1열

오류: 0

경고: 0

대괄호 또는 등괄호 안의 코드 블록은 CTRL+Enter 또는 Command+Enter를 사용하여 확장 및 축소할 수 있습니다.

태그

이 리소스에 연결된 태그가 없습니다.

새 태그 추가

50줄(줄) 태그 개 더 추가할 수 있습니다.

취소

엔드포인트 생성

## 2. Network 테스트용 EC2 서버 접속 및 S3 버킷 조회

- Network-02 EC2 접속 → S3 Bucket Object 조회

```
ssh network-02
```

```
aws s3 ls
```

```
aws s3 ls s3://lab-edu-bucket-image-9*****0/
```

## SSM VPC Endpoint 생성 (Interface Type)

### 1. SSM 이용 Network Server 접속 테스트

- EC2 콘솔 메인 화면 → 인스턴스 리소스 탭 → 'lab-edu-ec2-network-ap-01' 선택 → '연결' 버튼 클릭
- 'Session Manager' 탭으로 이동 → '연결' 버튼 클릭
- EC2 콘솔 메인 화면 → 인스턴스 리소스 탭 → 'lab-edu-ec2-network-ap-02' 선택 → '연결' 버튼 클릭
- 'Session Manager' 탭으로 이동 → '연결' 버튼 클릭 → 연결

## 2. SSM VPC Endpoint용 Security Group 생성

- EC2 콘솔 메인 화면 → 보안 그룹 리소스 탭 → "보안 그룹 생성" 버튼 클릭
- 보안그룹 생성 정보 입력
  - 보안그룹 이름: lab-edu-sg-endpoint-interface
  - VPC: lab-edu-vpc-ap-01
  - 인바운드 규칙:
    - 유형: HTTPS / 포트 범위: 443 / 소스: 10.0.0.0/16
    - 유형: UDP / 포트 범위: 53 / 소스: 10.0.0.0/16
  - '보안 그룹 생성' 버튼 클릭

## 3. VPC Endpoint 생성

- VPC 콘솔 메인 화면 → 엔드포인트 리소스 탭 → "엔드포인트 생성" 버튼 클릭
- 엔드포인트 생성 정보 입력 (ssm)
  - 이름: lab-edu-endpoint-ssm
  - 서비스 이름 / 소유자 / 유형: com.amazonaws.ap-northeast-2.ssm / amazon / Interface
  - VPC: lab-edu-vpc-ap-01
  - 서브넷: lab-edu-sub-pri-01, lab-edu-sub-pri-02
  - 보안그룹: lab-edu-sg-endpoint-interface
  - '엔드포인트 생성' 버튼 클릭
- 엔드포인트 생성 정보 입력 (ssmessages)
  - 이름: lab-edu-endpoint-ssmessages
  - 서비스 이름 / 소유자 / 유형: com.amazonaws.ap-northeast-2.ssmessages / amazon / Interface
  - VPC: lab-edu-vpc-ap-01
  - 서브넷: lab-edu-sub-pri-01, lab-edu-sub-pri-02
  - 보안그룹: lab-edu-sg-endpoint-interface
  - '엔드포인트 생성' 버튼 클릭
- 엔드포인트 생성 정보 입력 (ec2messages)
  - 이름: lab-edu-endpoint-ec2messages
  - 서비스 이름 / 소유자 / 유형: com.amazonaws.ap-northeast-2.ec2messages / amazon / Interface
  - VPC: lab-edu-vpc-ap-01
  - 서브넷: lab-edu-sub-pri-01, lab-edu-sub-pri-02
  - 보안그룹: lab-edu-sg-endpoint-interface
  - '엔드포인트 생성' 버튼 클릭

## 4. SSM 이용 Network Server 접속

- EC2 콘솔 메인 화면 → 인스턴스 리소스 탭 → 'lab-edu-ec2-network-ap-02' 선택 → '연결' 버튼 클릭
- 'Session Manager' 탭으로 이동 → '연결' 버튼 클릭 → 연결