

네이버 클라우드 플랫폼

Expert 과정

Lab Guide

대외비

Cloud Tech Froniter | 2023.

문서정보

본 문서는 네이버클라우드플랫폼 공인 교육 - Expert진행을 위한 실습 가이드입니다. 본 실습 가이드는 이론 교육과 더불어 실습 교육 시 활용됩니다.

Copyright © 2018 Naver Business Platform, Inc. All rights reserved

Lab 1

내 서버 이미지 공유하기

- Services > Compute > Server > Server image 로 이동
- web001-image 선택 후 상단의 공유 권한 설정
- 강사가 지정한 수강생의 ID 를 입력 후 우측의 '권한 추가' 클릭 > 하단의 '적용' 클릭

- 서버 이미지의 상태가 '생성됨(공유됨)'으로 변경됨을 확인

Lab 2

1. 스토리지 암호화된 서버 생성

- Services > Compute > Server > Server > +서버 생성 선택
- 신규콘솔화면으로 접속하여야지만 스토리지 암호화 설정 가능
- 신규콘솔화면에서 NCP서버 이미지 선택 후 OS타입으로 'CentOS'선택
- 이미지는 **Centos-7.8-64(XEN)**을 선택합니다.

서버 이미지 이름	설명
 centos-7.3-64	CentOS 7.3 (64-bit) (커널 업데이트 시 서버의 정상적인 사용이 불가능할 수 있으며 이에 따른 복구는 지원하지 않습니다.)
 centos-7.8-64	CentOS 7.8 (64-bit) (커널 업데이트 시 서버의 정상적인 사용이 불가능할 수 있으며 이에 따른 복구는 지원하지 않습니다.)
 ubuntu-16.04-64-server	Ubuntu Server 16.04 (64-bit) (커널 업데이트 시 서버의 정상적인 사용이 불가능할 수 있으며 이에 따른 복구는 지원하지 않습니다.)

- VPC : lab-vpc1
- Subnet : lab1-vpc-web-subnet
- 스토리지 종류 : SSD
- 서버세대 : G2
- 서버타입 : Standard / vCPU 2개, 메모리 8GB, 디스크 50GB 를 선택합니다.
- 스토리지 암호화 적용을 선택합니다.

스토리지 암호화 적용



암호화 기본 스토리지(OS)가 적용된 서버에는 암호화된 추가 스토리지만 연결할 수 있습니다.
마찬가지로, 암호화 되지 않은 기본 스토리지가 적용된 서버는 암호화 적용되지 않은 추가 스토리지만 연결 가능합니다.

- 서버 개수는 1, 서버 이름은 Inxsvr-en 입니다.
- Network Interface의 IP는 10.0.1.181 을 입력 > 우측의 '추가'버튼 클릭
- 공인IP : 새로운 공인IP 할당

스토리지 종류 ☒ SSD ☐ HDD

서버 세대

서버 타입
[High CPU] vCPU 2개, 메모리 4GB, [SSD]디스크 50GB [g2] C2-g2-s50

스토리지 암호화 적용 ☒
암호화 기본 스토리지(OS)가 적용된 서버에는 암호화된 추가 스토리지만 연결할 수 있습니다.
마찬가지로, 암호화 되지 않은 기본 스토리지가 적용된 서버는 암호화 적용되지 않은 추가 스토리지만 연결 가능합니다.

요금제 선택 ☒ 월요금제 ☐ 시간 요금제 월 72,000원 (OS 제외)

서버 개수

서버 이름
☒ 입력하신 서버 이름으로 hostname을 설정합니다.

- Init script는 미선택,
- 인증키는 lab1에서 만든 인증키를 사용합니다.
- 네트워크 접근 설정에서 eth0 NIC에 lab1-web-acg 를 할당합니다.

✓ 서버 이미지 선택 ✓ 서버 설정 ✓ 인증키 설정 4 네트워크 접근 설정 5 최종 확인

네트워크 접근 설정
보유하고 있는 ACG를 선택하거나 새로운 ACG를 생성해주세요. (*필수 입력 사항입니다.)
ACG(Access Control Group)은 별도의 방화벽 구축없이, 서버 그룹에 대한 네트워크 접근 제어 및 관리를 돕는 상품입니다.

디바이스	ACG
eth0	lab1-acg x

최대 3개까지 선택가능

설정 시 주의사항

- '신규 Network Interface'로 지정해서 생성된 ACG만 설정 가능합니다.
- 기존에 생성된 'Network Interface'를 사용하는 경우 해당 Network Interface의 ACG를 사용하게 됩니다.

- 서버 생성 마지막 확인페이지에서 '서버생성' 클릭

2. 스토리지 암호화된 스토리지 생성

- Services > Compute > Server > Server > Inxsrv-en 선택

- Inxsvr-en 서버를 선택한 후 상단의 '정지' 버튼 클릭
 - 정지가 완료 > 상단 메뉴의 서버 관리 및 설정 변경을 선택 > "스토리지 생성" 선택
 - 스토리지 종류 : HDD
 - 스토리지 이름 : Inxsvr-en-disk1
 - 크기 : 10GB
 - 스토리지 반납 보호 : 해제
 - 하단의 '추가'버튼 클릭 > '확인'버튼 클릭
 - 스토리지 추가가 완료되면 Services > Compute > Server 메뉴로 이동하여 Inxsvr-en 서버 선택 후, 상단의 '시작' 버튼 클릭하여 서버 재부팅
 - Inxsvr-en 서버가 운영중인 상태로 전환되면, 서버에 접속하여 다음과 같이 작업 진행
 - 공인IP로 접속
 - Inxsvr-en서버 선택 후 서버 관리 및 설정 변경 > 관리자비밀번호 확인
 - 마운트 하기 위한 폴더를 생성
- ```
mkdir /disk1
```
- 생성하여 Attach한 디스크의 파티션 생성을 진행

```
[root@Inxsvr-org ~]# fdisk /dev/xvdb
```

```
Welcome to fdisk (util-linux 2.23.2).
```

```
Changes will remain in memory only, until you decide to write them.
```

```
Be careful before using the write command.
```

```
Device does not contain a recognized partition table
```

```
Building a new DOS disklabel with disk identifier 0xd7dfcbf5.
```

```
Command (m for help): n
```

```
Partition type:
```

```
 p primary (0 primary, 0 extended, 4 free)
```

```
 e extended
```

```

Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-20971519, default 2048):
Using default value 2048
Last sector, +sectors or +size{K,M,G} (2048-20971519, default 20971519):
Using default value 20971519
Partition 1 of type Linux and of size 10 GiB is set

Command (m for help): w

The partition table has been altered!

Calling ioctl() to re-read partition table.

Syncing disks.

[root@lnxsvr-org ~]#

```

생성한 /dev/xvdb1 파티션에 대해 포맷을 진행, 포맷이 안되는 것 확인

```

[root@lnxsvr-org ~]# mkfs.ext4 /dev/xvdb1

mke2fs 1.39 (29-Oct-2015)
/dev/sdb2 is apparently in use by the system; will not make a filesystem here!

[root@lnxsvr-org ~]#

```

가비지 정보 업데이트

```

[root@lnxsvr-org ~]# dmsetup status

[root@lnxsvr-org ~]# dmsetup remove_all

[root@lnxsvr-org ~]# dmsetup status

```

포맷 재시도

```
[root@lnxsvr-org ~]# mkfs.ext4 /dev/xvdb1
mke2fs 1.42.9 (28-Dec-2013)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
655360 inodes, 2621184 blocks
131059 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=2151677952
80 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Superblock backups stored on blocks:
 32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632

Allocating group tables: done
Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done

[root@lnxsvr-org ~]#
```

마운트 작업 진행하고 데이터를 생성

```
[root@lnxsvr-org ~]# mount /dev/xvdb1 /disk1
[root@lnxsvr-org ~]# df
```

| Filesystem | 1K-blocks | Used    | Available | Use% | Mounted on     |
|------------|-----------|---------|-----------|------|----------------|
| /dev/xvda3 | 50305028  | 1957200 | 48347828  | 4%   | /              |
| devtmpfs   | 1926388   | 0       | 1926388   | 0%   | /dev           |
| tmpfs      | 1809260   | 0       | 1809260   | 0%   | /dev/shm       |
| tmpfs      | 1809260   | 8536    | 1800724   | 1%   | /run           |
| tmpfs      | 1809260   | 0       | 1809260   | 0%   | /sys/fs/cgroup |
| tmpfs      | 361852    | 0       | 361852    | 0%   | /run/user/0    |
| /dev/xvdb1 | 10189076  | 36888   | 9611568   | 1%   | /disk1         |

[root@lnxsvr-org ~]# cp -rf /etc/\* /disk1/



## Lab 3

### 1. 서버에 추가 IP 설정

- Services > server > network interface 선택
- web001 서버에 할당되어있는 네트워크인터페이스 선택 후 상단의 secondary IP 버튼 클릭
- Secondary IP 입력란에 '10.0.1.251' 입력 후 우측에 추가 버튼 클릭
- 하단의 설정 버튼 클릭
- web001에 로그인 후 vi /etc/sysconfig/network-scripts/ifcfg-eth0:1파일 생성
- 파일 내용 - 아래 링크에서 복사가능
  - <https://kr.object.ncloudstorage.com/11-edu-yangju/DEVICE.txt>

```
DEVICE=eth0:1
BOOTPROTO=STATIC
IPADDR=10.0.1.251
NETMASK=255.255.255.0
ONBOOT=yes
```

- 인터페이스 활성화

```
[root@target-linux network-scripts]# ifup eth0:1
[root@target-linux network-scripts]#
```

## Lab4

### 1. NKS 클러스터 생성

- Services > Containers > Ncloud Kubernetes Service > 생성하기
  - 클러스터 이름 : k8s-XXX(실습 날짜 기입)
  - 하이퍼바이저 : KVM
  - 클러스터 버전 : 1.27.9
  - CNI Plugin : cilium
  - VPC : lab1-vpc
  - 가용 zone : KR-2
  - 네트워크 타입 : Public
  - subnet : lab1-vpc-web-subnet
  - LB Private subnet : (LB서브넷 생성 클릭)
    - ◆ 상단의 Subnet생성 클릭
    - ◆ Subnet 이름 : lb-private-subnet
    - ◆ VPC : lab1-vpc
    - ◆ IP주소범위 : 10.0.6.0/24
    - ◆ 가용존 : KR-2
    - ◆ Network ACL : lab1-vpc-private-nacl 선택
    - ◆ Internet Gateway사용여부 : N
    - ◆ 용도 : 로드밸런서 선택 후 하단의 '생성' 버튼 클릭
  - 위 서브넷이 생성이 완료되면 다시 쿠버네티스 생성 탭으로 이동
  - LB Private Subnet : lb-private-subnet
  - LB Public subnet : lab1-vpc-lb-sub3
  - Audit Log : 미설정
  - 반납보호 : 미설정

## 클러스터 설정

클러스터 정보를 입력하세요. (\*필수 입력 사항입니다.)

|                |                                                                                                                                                                                                                                                                               |                                  |                                                               |
|----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------|---------------------------------------------------------------|
| 클러스터 이름        | <input type="text" value="k8w-20240419"/>                                                                                                                                                                                                                                     |                                  |                                                               |
| 하이퍼바이저         | <input checked="" type="radio"/> KVM <input type="radio"/> XEN                                                                                                                                                                                                                |                                  |                                                               |
| Kubernetes 버전  | <input type="text" value="1.27.9"/>                                                                                                                                                                                                                                           | <input type="button" value="↺"/> |                                                               |
| CNI Plugin     | <input type="text" value="cilium"/>                                                                                                                                                                                                                                           | <input type="button" value="↺"/> |                                                               |
| VPC            | <input type="text" value="lab1-vpc   10.0.0.0/16"/>                                                                                                                                                                                                                           | <input type="button" value="↺"/> | <input type="button" value="VPC 생성"/> <a href="#">↗</a>       |
|                | <ul style="list-style-type: none"> <li>Kubernetes 클러스터 생성을 위해서는 IP 대역(10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16) 내에서 /17~/26 범위의 Subnet, 로드 밸런서 전용 Subnet이 필요합니다.</li> <li>Docker Bridge 대역의 충돌을 방지하기 위해 172.17.0.0/16 범위 내의 Subnet, 로드 밸런서 전용 Subnet은 선택할 수 없습니다.</li> </ul> |                                  |                                                               |
| 가용 Zone        | <input type="text" value="KR-2"/>                                                                                                                                                                                                                                             | <input type="button" value="↺"/> |                                                               |
| 네트워크 타입        | <input type="text" value="Public"/>                                                                                                                                                                                                                                           | <input type="button" value="↺"/> |                                                               |
| Subnet         | <input type="text" value="lab1-vpc-web-subnet-202307   KR-2   10.0.8.0/24   P..."/>                                                                                                                                                                                           | <input type="button" value="↺"/> | <input type="button" value="Subnet 생성"/> <a href="#">↗</a>    |
| LB Private 서브넷 | <input type="text" value="lb-private-subnet   KR-2   10.0.6.0/24   Private"/>                                                                                                                                                                                                 | <input type="button" value="↺"/> | <input type="button" value="LB Subnet 생성"/> <a href="#">↗</a> |
| LB Public 서브넷  | <input type="text" value="lb-subnet   KR-2   10.0.255.0/24   Public"/>                                                                                                                                                                                                        | <input type="button" value="↺"/> | <input type="button" value="LB Subnet 생성"/> <a href="#">↗</a> |
| Audit Log      | <input type="radio"/> 설정 <input checked="" type="radio"/> 미설정                                                                                                                                                                                                                 |                                  |                                                               |
| 반납보호           | <input type="radio"/> 설정 <input checked="" type="radio"/> 미설정                                                                                                                                                                                                                 |                                  |                                                               |
|                | 반납 보호를 설정하면 실수로 반납하는 사고를 방지할 수 있습니다.                                                                                                                                                                                                                                          |                                  |                                                               |
| ACG 설정         | Ncloud Kubernetes Service를 위한 ACG는 자동 생성됩니다.(예: nks-*)<br>클러스터에 접근을 위한 접속 정보는 클러스터의 생성 완료 후에 확인 하실 수 있습니다.                                                                                                                                                                    |                                  |                                                               |
| 주의사항           | <ul style="list-style-type: none"> <li>Ncloud Kubernetes Service 콘솔 이외의 기능으로 서버 자원(VM), VPC, Subnet을 수동으로 정지 또는 삭제하는 경우 클러스터에 오류가 발생할 수 있습니다.</li> <li>Kubernetes 워커노드 자원을 반납하는 경우, Ncloud Kubernetes Service 콘솔에서 반납해 주시기 바랍니다.</li> </ul>                                   |                                  |                                                               |

- 모든 정보 기입 후, 하단의 다음 버튼 클릭
- 노드 풀 이름 : default-pool
- 서버 이미지 이름: Ubuntu 20.64
- 서버 타입 : [Standard] s2-g3
- Storage Size : 100
- 노드 수 : 1
- Subnet : 자동 할당
- Node IAM Role, Kubernetes label 및 taint는 설정 안함
- 하단의 [추가] 버튼 클릭 후, 하단의 다음 버튼 클릭
- 로그인 키 설정 : (이전 실습에서 생성한 인증키 선택)
- 하단의 [다음] 클릭 후 [생성하기] 버튼 클릭

## 2. Kubectl config 설정

- 스크립트 확인 : <https://kr.object.ncloudstorage.com/nce-edu/ncp-iam.txt>
- Kubernetes Service는 ncp-iam-authenticator를 통해 IAM 인증을 제공합니다. IAM 인증을 통해 kubectl 명령을 사용하려면 ncp-iam-authenticator를 설치하고 이를 인증에 사용하도록 kubectl 설정 파일을 수정해야 합니다
- 다음과 같이 web001 서버에서 ncp-iam-authenticator 설치

- Ncp-iam-authenticator 바이너리 파일을 홈디렉토리에 다운로드

```
cd ~

curl -o ncp-iam-authenticator https://kr.object.ncloudstorage.com/nks-download/ncp-iam-authenticator/v1.0.0/linux/amd64/ncp-iam-authenticator
```

- 바이너리에 실행권한 추가

```
chmod +x ./ncp-iam-authenticator
```

- \$HOME/bin/ncp-iam-authenticator를 생성하고 \$PATH에 추가

```
mkdir -p /root/bin && cp ./ncp-iam-authenticator /root/bin/ncp-iam-authenticator && export PATH=$PATH:/root/bin
```

- Shell Profile에 PATH를 추가 후 명령어가 잘 작동하는지 확인

```
echo 'export PATH=$PATH:$HOME/bin' >> ~/.bash_profile

ncp-iam-authenticator help
```

- 다음과 같이 IAM 인증을 위해 kubeconfig를 생성합니다.
- Kubeconfig 생성 시 ncp-iam-authenticator 를 통해 진행해야 합니다. 이 때 ncp-iam-authenticator를 사용하기 위해서 먼저 API 인증키값을 설정
- 홈페이지 > 마이페이지에서 API 인증키값을 확인 후, 아래 명령어에 맞게 넣어 실행
- 아래 예시는 ENV 설정을 통해 셋팅

```
export NCCLOUD_ACCESS_KEY=〈사용자의 Access key〉

export NCCLOUD_SECRET_KEY=〈사용자의 Secret key〉
```

```
export NCLOUD_API_GW=https://ncloud.apigw.ntruss.com
```

- 사용자 환경 홈 디렉터리의 .ncloud 폴더에 configure 파일

```
mkdir .ncloud

vi ~/.ncloud/configure

[DEFAULT]

ncloud_access_key_id = ACCESSKEY

ncloud_secret_access_key = SECRETKEY

ncloud_api_url = https://ncloud.apigw.ntruss.com
```

- ncp-iam-authenticator create-kubeconfig 명령을 사용하여 kubeconfig를 생성
- Cluster UUID는 Services > Kubernetes Service 에서 생성한 Cluster를 클릭하면 클러스터 이름 옆에서 확인 가능합니다

| <input checked="" type="checkbox"/> k8s-0224    | 1.21.9 | 1/1 | vCPU 2EA, Memory 8GB |
|-------------------------------------------------|--------|-----|----------------------|
| 설명                                              |        |     |                      |
| 클러스터 이름 (UUID)                                  |        |     |                      |
| k8s-0224 (8e488bdd-a15f-450a-a4c1-4ec38c0b7abb) |        |     |                      |

```
ncp-iam-authenticator create-kubeconfig --region KR --clusterUuid <cluster-uuid> > kubeconfig.yml
```

- Kubeconfig 파일이 생성되면 kubectl 명령어를 테스트합니다

```
kubectl get namespaces --kubeconfig kubeconfig.yml
```

```
[root@lnxsvr1 ~]# kubectl get namespaces
NAME STATUS AGE
default Active 99m
kube-node-lease Active 99m
kube-public Active 99m
kube-system Active 99m
```

- Kubeconfig 파일이 지정이 번거로울 경우, 아래와 같이 bash\_profile 에 alias로 명시합니다

```
vi ~/.bash_profile

alias kubectl='kubectl --kubeconfig="/root/kubeconfig.yml"' -> 파일 맨 밑에 alias 내용 추가

source ~/.bash_profile

kubectl get namespaces
```

## Container 에 올라간 이미지를 이용하여 Pod 생성

- 1) Container Registry 의 Access/Secret Key 를 저장한 Secret 오브젝트 생성

```
kubectl create secret docker-registry regcred ₩

--docker-server=<private-registry-end-point> --docker-username=<access-key-id> ₩

--docker-password=<secret-key> --docker-email=<your-email>

kubectl get secret
```

```
[root@lnxsvr1 ~]# kubectl get secret
NAME TYPE DATA AGE
default-token-csqs kubernetes.io/service-account-token 3 173m
regcred kubernetes.io/dockerconfigjson 1 56s
```

- 2) Pod 생성

```
cd lab_source

cd lab3
```

- create\_only\_pod.yaml 파일 생성 및 배포
- image: registry-name 값 변경

```
vi create_only_pod.yaml

apiVersion: v1

kind: Pod

metadata:
```

```

name: apache-pod

namespace: default

spec:

 containers:

 - name: apache-pod

 image: <prviate-endpoint> /image_apache:1.0

 imagePullSecrets:

 - name: regcred

```

```

kubectl create -f create_only_pod.yaml

kubectl get pods -o wide

```

```

[root@lnxsvr1 lab3]# kubectl get pods -o wide
NAME READY STATUS RESTARTS AGE IP NODE NOMINATED NODE READINESS GATES
apache-pod 1/1 Running 0 14s 198.18.0.122 default-pool-w-10e5 <none> <none>

```

### 3) Deployment 오브젝트로 Pod 생성

- create\_deployment.yaml 파일 수정
- image: registry-name 값 변경

```
vi create_deployment.yaml
```

```

apiVersion: apps/v1

kind: Deployment

metadata:

 name: apache-deployment

spec:

 replicas: 3

 selector:

```

```

matchLabels:

 app: apache

template:

 metadata:

 labels:

 app: apache

 spec:

 containers:

 - name: apache

 image: <private-endpoint>/image_apache:1.0

 ports:

 - containerPort: 80

 imagePullSecrets:

 - name: regcred

```

```
kubectl apply -f create_deployment.yaml
```

```
kubectl get pods
```

```
[root@lnxsvr1 lab3]# kubectl get pods
```

| NAME                               | READY | STATUS  | RESTARTS | AGE |
|------------------------------------|-------|---------|----------|-----|
| apache-deployment-68fb8cc664-65bmg | 1/1   | Running | 0        | 7s  |
| apache-deployment-68fb8cc664-n69zd | 1/1   | Running | 0        | 7s  |
| apache-deployment-68fb8cc664-zsphv | 1/1   | Running | 0        | 7s  |
| apache-pod                         | 1/1   | Running | 0        | 96s |

#### 4) Deployment 로 생성한 Pod 에 Service 연결

- create\_service.yaml 파일 생성

```
vi create_service.yaml
```



```
kind: Service

apiVersion: v1

metadata:
 name: example-service

spec:
 ports:
 - port: 80
 targetPort: 80

 selector:
 app: apache

 type: LoadBalancer
```

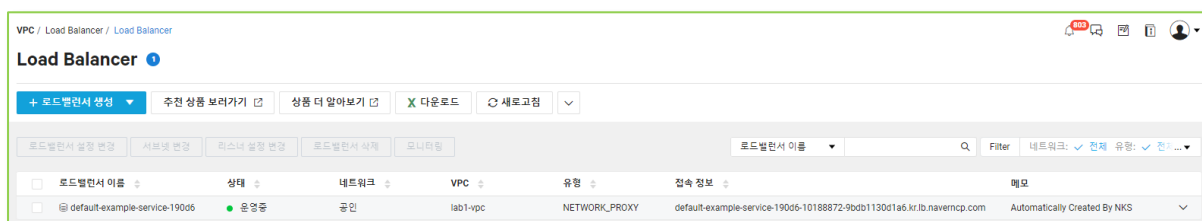
```
kubectl apply -f create_service.yaml
```

```
kubectl get service
```

```
[root@lnxsvr1 lab3]# kubectl get service
NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE
example-service LoadBalancer 198.19.241.70 <pending> 80:30943/TCP 6s
kubernetes ClusterIP 198.19.128.1 <none> 443/TCP 178m
```

## 로드밸런서 확인 및 서비스 접속 테스트

### 1) Services > Load Balancer > 생성된 LB 확인



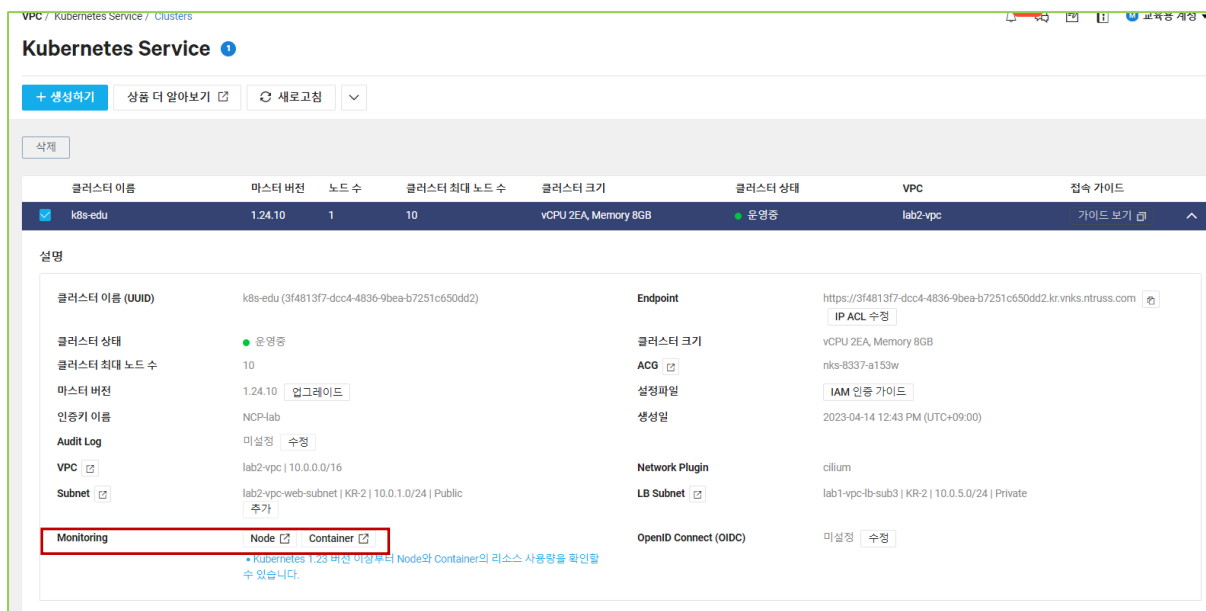
| 로드밸런서 이름                      | 상태  | 네트워크 | VPC      | 유형            | 접속 정보                                                                 | 메모                           |
|-------------------------------|-----|------|----------|---------------|-----------------------------------------------------------------------|------------------------------|
| default-example-service-19006 | 운행중 | 공인   | lab1-vpc | NETWORK_PROXY | default-example-service-19006-10188872-9b0d1130d1a6.kr.lb.navercp.com | Automatically Created By NKS |

### 2) 브라우저에서 LB 접속정보(URL) 로 접속



## 쿠버네티스 클러스터 모니터링

- Services > Container service > Kubernetes 로 이동
- 운영중인 Kubernetes 클러스터 클릭



- 그래파나와 연동되어 워커노드 및 컨테이너 현황 모니터링 가능

## Lab 5

### 1. Object Storage 사용을 위한 fuse 구성

- Services > Storage > Object Storage선택 > +버킷 생성 선택
- 버킷 이름에 '네이버클라우드플랫폼 아이디-fuse' 버킷 생성
  - 설정관리 및 암호화 설정, 권한관리는 디폴트값으로 두고 생성 진행
- Web001 에 접속하여 패키지 설치
  - 패키지 설치 명령어 : <https://kr.object.ncloudstorage.com/ai-edu/s3fuse.txt>

```
[root@target-linux cli_linux]# yum install automake fuse-devel gcc-c++ git libcurl-devel libxml2-devel make openssl-devel -y
[root@target-linux cli_linux]# git clone https://github.com/s3fs-fuse/s3fs-fuse.git
[root@target-linux cli_linux]# cd s3fs-fuse
[root@target-linux cli_linux]# ./autogen.sh
[root@target-linux cli_linux]# ./configure
[root@target-linux cli_linux]# make
[root@target-linux cli_linux]# make install
```

- 마운트 포인트 생성

```
[root@target-linux cli_linux]# mkdir -p /objectstorage/아이디
```

- 환경 설정
  - 각자의 access key와 secret key값은 ncloud.com > 마이페이지 > 인증키관리에서 확인 가능

```
echo ACCESS_KEY_ID:SECRET_ACCESS_KEY > /etc/passwd-s3fs
chmod 600 /etc/passwd-s3fs
```

- 마운트

```
s3fs 오브젝트스토리지버킷명 /objectstorage/아이디 -o
url=https://kr.object.ncloudstorage.com
```

- 마운트 확인

```
[root@lab2-vpc-lnxsvr1 s3fs-fuse]# df -h
```

| Filesystem | Size | Used | Avail | Use% | Mounted on           |
|------------|------|------|-------|------|----------------------|
| s3fs       | 16E  | 0    | 16E   | 0%   | /objectstorage/edu50 |

## 백업 서비스 신청하기

- Services > Storage > Backup 선택
- Backup > Resource 메뉴로 이동 상단의 '리소스 생성'클릭
- 리소스 이름 : nce-backup
- 존 : KR-2
- 서버 : web002
- 에이전트 유형 : Data
- 아이디 : root(백업서비스 수행을 위해선 root 권한 필요)
- 비밀번호 : root 계정 비밀번호 입력 후 하단의 '다음'버튼 클릭
- 마지막 최종 페이지 확인 후, 하단의 '리소스 생성' 클릭
- Backup > Storage 클릭 > '저장소 생성' 클릭
- 저장소 이름 : nce-backup
- 존 : KR-2 선택 후 하단의 '다음'버튼 클릭 > 하단의 '생성' 클릭
- Backup > Policy 클릭 > 상단의 '정책 생성' 클릭
- 정책 이름 : nce-policy
- 보관 기간 : 7일
- 존 : KR-2 하단의 '다음' 클릭 후 '생성' 클릭
- Backup > Job 클릭 후 '작업생성' 클릭
- 작업이름 : nce-job
- 리소스 : nce-backup
- 백업 대상 유형 : Data
- 백업 대상 경로 : root 클릭
- 정책 : nce-policy 선택 후 하단의 '다음' 클릭 > '생성' 클릭
- Backup > Schedule 클릭 > 상단의 '일정 생성' 클릭
- 일정 이름 : nce-bakcup-schedule
- 작업 : nce-job
- 백업 방식 : 전체

- 백업 주기 : 일간
- 시작 시간 : AM 10:00 클릭 후 하단의 '다음' 버튼 클릭 > 하단의 '생성' 클릭

## Lab 6

### Cloud Hadoop 구성

#### 1. (사전 작업) Hadoop cloud Object storage 구성

- Services > Object Storage > Bucket Management > + 버킷생성 선택
- 기본정보> “버킷이름” 에 labhadoop아이디 입력 후 하단의 ‘다음’버튼 클릭

VPC / Object Storage / Bucket Management

< 버킷 생성 파일과 폴더를 저장하는 상위 단위인 버킷을 생성하세요.

1 기본 정보 2 권한 관리 3 잠금 관리 4 최종 확인

**기본 정보 입력**

버킷은 파일과 폴더를 저장하는 상위 단위입니다.  
리전 내에서 유일하게 사용될 버킷의 이름을 입력하세요.  
Object Storage 요금은 저장된 데이터 양, API 요청수, 네트워크 전송 요금을 합산하여 부과합니다.

버킷 이름

다음 >

- 잠금 설정, 암호화 설정은 디폴트 값으로 두고 ‘다음’버튼 클릭
- 권한 관리도 디폴트 값으로 유지 후 하단의 ‘다음’버튼 클릭
- 마지막 확인페이지에서 “버킷생성” 클릭 하면 Object storage 생성 완료

#### 2. Hadoop 클러스터 설정

- Services > Big data & Analytics > Cloud Hadoop > + 클러스터 생성 선택
- “클러스터 이름” 에 lab 이라고 입력
- 클러스터 버전은 Cloud Hadoop 2.0
- 클러스터 타입은 Core Hadoop with Spark 선택
- 클러스터 add-on은 선택 안함
- Data catalog 서비스 카탈로그 사용 : 사용 안함
- 커버로스 인증 구성은 비활성화
- VPC : lab1-vpc

- 클러스터 관리자 계정 : hadoop
- 클러스터 관리자 계정 비밀번호 : ncpNCP!@#123 입력 후 하단의 '다음'버튼 클릭

- Object Storage에 미리 생성한 버킷을 지정
- Bootstrap script : 선택 해제
- 엣지 노드,마스터 노드는 Public Subnet에 배치 - lab1-vpc-web-subnet 선택
- 작업자 노드는 Private Subnet 배치 - lab1-vpc-redis-subnet 선택
- 나머지는 Default 설정을 유지합니다.
- 하단의 다음 버튼을 클릭합니다.
- 인증키 설정 부분은 '보유하고 있는 인증키 이용'을 선택 후, lab1에서 서버 생성 시 다운로드 받았던 인증키를 선택합니다.
- 하단의 다음 버튼을 클릭합니다.
- 마지막 최종 확인 후, 생성 버튼을 클릭합니다.



스토리지 & 서버 설정

(필수 입력 사항입니다)

Object Storage 버킷

labhadoop-edu50

새로고침

고가용성 지원

☒ 기본적으로 2대의 마스터 노드가 생성됩니다.

옛지노드 Subnet

demo-subnet | KR-2 | Public

Subnet 생성

옛지노드 서버 타입

vCPU 4개, 메모리 8GB, 디스크 50GB

옛지노드 개수

1

1개로 고정

마스터노드 Subnet

demo-subnet | KR-2 | Public

Subnet 생성

마스터노드 서버타입

vCPU 4개, 메모리 16GB, 디스크 50GB

기본적으로 50G크기의 디스크(HDD)가 제공됩니다.

마스터노드 개수

2

고가용성 지원으로 2대의 마스터 노드가 생성되며 변경은 불가능합니다.

마스터노드 스토리지 타입

☒ SSD ☐ HDD

설치 이후에 스토리지 타입은 변경되지 않습니다.

마스터노드 스토리지 용량

100

GB, 최소 100GB, 최대 2000GB 까지 10GB단위 선택 가능합니다.

작업자노드 Subnet

lab1-vc-db-sub1 | KR-2 | Private

Subnet 생성

작업자노드 Subnet 은 Private Subnet 만 지원가능합니다.

작업자노드 서버타입

vCPU 4개, 메모리 32GB, 디스크 50GB

기본적으로 50G크기의 디스크(HDD)가 제공됩니다.

작업자노드 개수

2

작업자 노드는 최소 2개이상 되어야 합니다.(default 2)

작업자노드 스토리지 타입

☒ SSD ☐ HDD

설치 이후에 스토리지 타입은 변경되지 않습니다.

작업자노드 스토리지 용량

100

GB, 최소 100GB, 최대 2000GB 까지 10GB단위 선택 가능합니다.

요금제

시간 요금제

요금 안내(Storage 비용은 별도 부과됩니다.)

### 3. Zeppelin Notebook 접속

- Zeppelin notebook에 접속하기 위해서 cloud Hadoop ACG를 업데이트합니다.
- Services > Server > ACG 항목에서 cloud-hadoop-acg-\*로 시작하는 acg를 클릭후, myip에 대해서 9996포트를 허용해줍니다.

ACG 규칙 설정 | cloud-hadoop-acg-49k7o

ACG 에 적용된 상세 규칙을 표시합니다.

**Inbound** Outbound

프로토콜: TCP | 접근 소스: | 허용 포트: | 메모: | 설정: + 추가

예1) IP: 0.0.0.0/0, 192.168.1.0/24, 192.168.1.7  
예2) ACG 이름: my-acg-1  
[Detail](#)

| 프로토콜 | 접근 소스                         | 허용 포트   | 메모                                                                   | 설정                       |
|------|-------------------------------|---------|----------------------------------------------------------------------|--------------------------|
| TCP  | 182.216                       | 9996    |                                                                      | <input type="checkbox"/> |
| TCP  | cloud-hadoop-acg-49k7o(19854) | 1-65535 | (automatically created, don't delete it) cloud-hadoop-acg-lab-hadoop | <input type="checkbox"/> |
| ICMP | 211.249.71.0/24               |         | (automatically created, don't delete it) cloud-hadoop-acg-lab-hadoop | <input type="checkbox"/> |
| ICMP | 211.249.70.0/24               |         | (automatically created, don't delete it) cloud-hadoop-acg-lab-hadoop | <input type="checkbox"/> |
| ICMP | 211.249.69.128/25             |         | (automatically created, don't delete it) cloud-hadoop-acg-lab-hadoop | <input type="checkbox"/> |
| TCP  | 211.249.71.0/24               | 8080    | (automatically created, don't delete it) cloud-hadoop-acg-lab-hadoop | <input type="checkbox"/> |
| TCP  | 211.249.70.0/24               | 8080    | (automatically created, don't delete it) cloud-hadoop-acg-lab-hadoop | <input type="checkbox"/> |

- lab-hadoop선택 후, 상단의 'application별 보기' 클릭 후 zeppelin notebook 접속 용 클릭

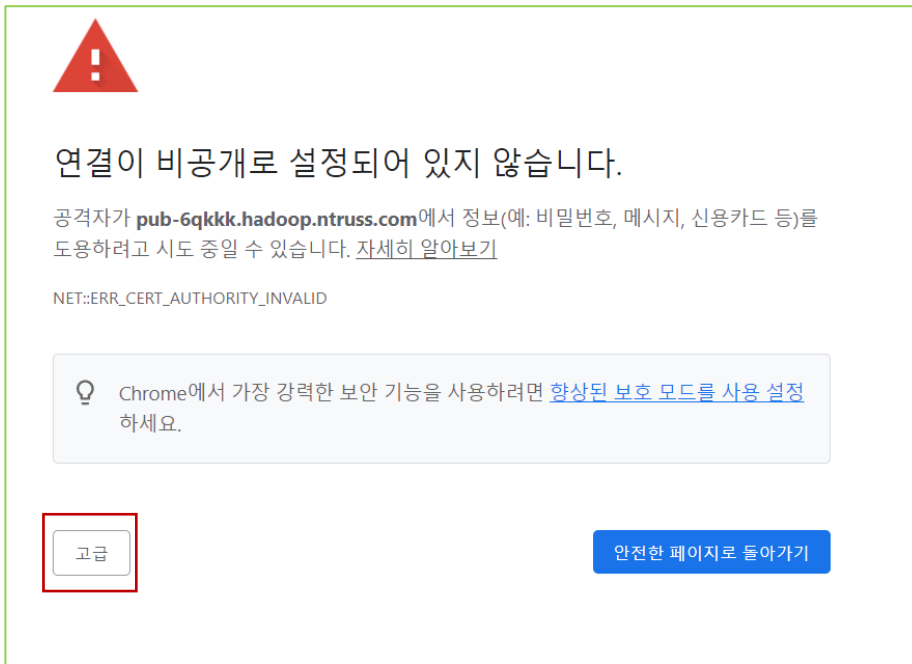
Application web UI 접속을 위한 사전 작업

Application web UI 접속을 위해서는 사전에 아래 모든 규칙이 해당 클러스터의 ACG에 필수로 설정되어 있어야 합니다.  
ACG 설정 변경은 콘솔 > 클러스터 상세 정보 > ACG에서 가능합니다.

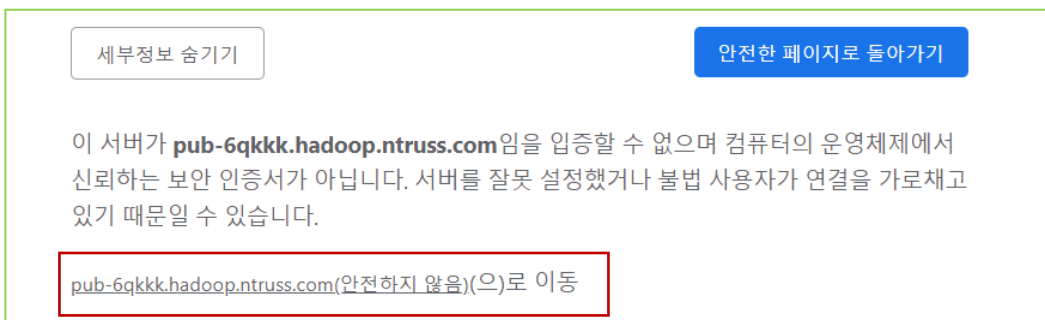
| 프로토콜 | 접근소스 | 허용포트 | 비고                                     |
|------|------|------|----------------------------------------|
| TCP  | 도메인  | 8443 | <a href="#">Ambari Web Console 접속용</a> |
| TCP  | 도메인  | 8081 | <a href="#">Hue Admin 접속용</a>          |
| TCP  | 도메인  | 9996 | <a href="#">Zeppelin Notebook 접속용</a>  |
| TCP  | 도메인  | 6182 | <a href="#">Ranger 접속용</a>             |

☐ 다시 보기 않음

- 아래와 같이 페이지가 노출되면, 고급 버튼을 클릭합니다.



- \*\* (안전하지 않음) 링크를 클릭합니다.



- 아래와 같이 zeppelin화면이 노출되면 정상 접속이 된 것입니다.
- Zeppelin 메인 화면 우측 상단의 로그인 버튼을 클릭한 후 클러스터 생성 시 기입한 클러스터 관리자 계정 정보로 접속을 합니다.



- ✓ Username : hadoop

✓ Password : ncpNCP!@#123

#### 4. Zeppelin notebook 생성 및 데이터 확인

- Zeppelin 상단의 노트북 클릭 후, 상단의 '+create new note'를 클릭합니다.
  - Note name : nce-lab 입력
  - Default interpreter : spark 선택 후 하단의 create 버튼 클릭
- Bank.csv파일을 bank테이블에 로드하는 샘플코드를 복사해서 붙여넣습니다.
  - 링크 : <https://kr.object.ncloudstorage.com/ai-edu/spark.txt>

```
%spark.spark

import org.apache.commons.io.IOUtils

import java.net.URL

import java.nio.charset.Charset

import spark.implicits._

// Zeppelin creates and injects sc (SparkContext) and sqlContext (HiveContext or SqlContext)
// So you don't need create them manually

// load bank data

val bankText = sc.parallelize(

 IOUtils.toString(

 new URL("https://kr.object.ncloudstorage.com/nce-edu/bank.csv"),

 Charset.forName("utf8")).split("\n"))

case class Bank(age: Integer, job: String, marital: String, education: String, balance: Integer)
```

```
val bank = bankText.map(s => s.split(";")).filter(s => s(0) != "W"ageW").map(
 s => Bank(s(0).toInt,
 s(1).replaceAll("W", ""),
 s(2).replaceAll("W", ""),
 s(3).replaceAll("W", ""),
 s(5).replaceAll("W", "").toInt
)
).toDF()

bank.registerTempTable("bank")
```

- 우측 상단의 ▶ 버튼을 클릭하여 실행합니다.

```
%spark2.spark
import org.apache.commons.io.IOUtils
import java.net.URL
import java.nio.charset.Charset

// Zeppelin creates and injects sc (SparkContext) and sqlContext (HiveContext or SqlContext)
// So you don't need create them manually

// load bank data
val bankText = sc.parallelize(
 IOUtils.toString(
 new URL("https://s3.amazonaws.com/apache-zeppelin/tutorial/bank/bank.csv"),
 Charset.forName("utf8")).split("\n"))

case class Bank(age: Integer, job: String, marital: String, education: String, balance: Integer)

val bank = bankText.map(s => s.split(";")).filter(s => s(0) != "W"ageW").map(
 s => Bank(s(0).toInt,
 s(1).replaceAll("W", ""),
 s(2).replaceAll("W", ""),
 s(3).replaceAll("W", ""),
 s(5).replaceAll("W", "").toInt
)
).toDF()

bank.registerTempTable("bank")
```

실행 후, 우측 상단의 status가 finished로 변경되면 성공적으로 데이터가 입력된 것입니다.

```

%spark2.spark
import org.apache.commons.io.IOUtils
import java.net.URL
import java.nio.charset.Charset

// Zeppelin creates and injects sc (SparkContext) and sqlContext (HiveContext or SqlContext)
// So you don't need create them manually

// load bank data
val bankText = sc.parallelize(
 IOUtils.toString(
 new URL("https://s3.amazonaws.com/apache-zeppelin/tutorial/bank/bank.csv"),
 Charset.forName("utf8")).split("\n")
)

case class Bank(age: Integer, job: String, marital: String, education: String, balance: Integer)

val bank = bankText.map(s => s.split(";")).filter(s => s(0) != "\"age\"").map(
 s => Bank(s(0).toInt,
 s(1).replaceAll("\"", ""),
 s(2).replaceAll("\"", ""),
 s(3).replaceAll("\"", ""),
 s(5).replaceAll("\"", "").toInt
)
).toDF()
bank.registerTempTable("bank")

<console>:35: error: value toDF is not a member of org.apache.spark.rdd.RDD[Bank]
possible cause: maybe a semicolon is missing before `value toDF`?
).toDF()
 ^
warning: there was one deprecation warning; re-run with -deprecation for details
import sqlContext.implicits._
import org.apache.commons.io.IOUtils
import java.net.URL
import java.nio.charset.Charset
bankText: org.apache.spark.rdd.RDD[String] = ParallelCollectionRDD[0] at parallelize at <console>:24
defined class Bank
bank: org.apache.spark.sql.DataFrame = [age: int, job: string ... 3 more fields]

```

이번에는 데이터를 조회해보고, 그래프로 결과를 확인해보겠습니다.

- 링크 : [https://kr.object.ncloudstorage.com/ai-edu/spark\\_sql.txt](https://kr.object.ncloudstorage.com/ai-edu/spark_sql.txt)

```
%spark.sql
```

```
select age, count(1) value
```

```
from bank
```

```
where age < 30
```

```
group by age
```

```
order by age
```

```

%spark2.sql
select age, count(1) value
from bank
where age < 30
group by age
order by age

```

아래와 같이 표 형태로 결과를 확인해 볼 수 있습니다.

SPARK JOB FINISHED

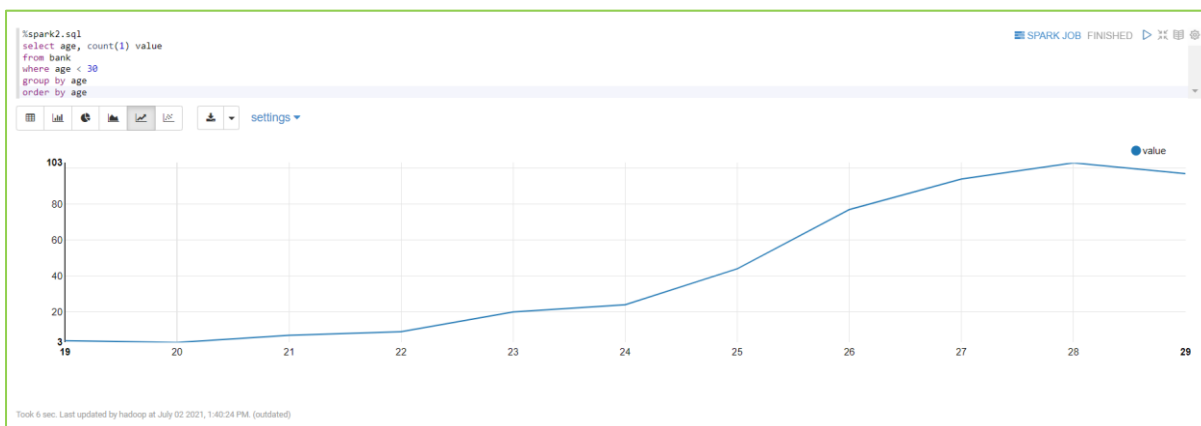
```

%spark2.sql
select age, count(1) value
from bank
where age < 30
group by age
order by age

```

| age | value |
|-----|-------|
| 19  | 4     |
| 20  | 3     |
| 21  | 7     |
| 22  | 9     |
| 23  | 20    |
| 24  | 24    |
| 25  | 44    |
| 26  | 77    |

도표 클릭 시, 아래와 같이 그래프 형태로도 데이터 결과를 확인할 수 있습니다.



## Lab 7(Demo)

### OS Security Checker 설정

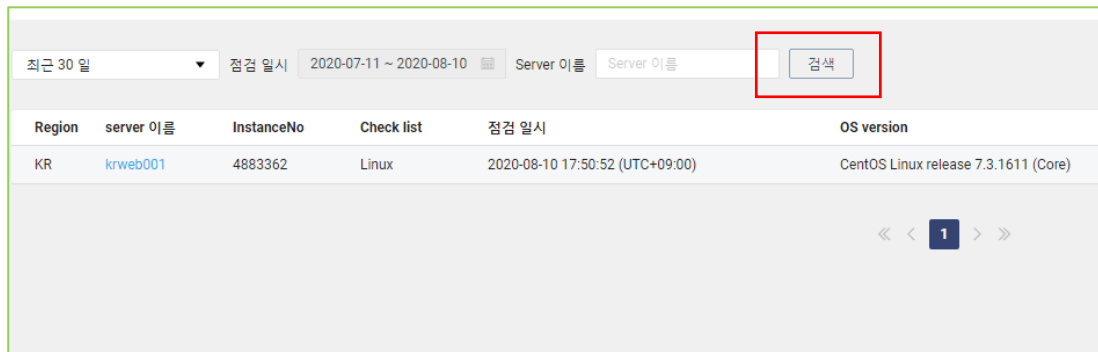
- Web001 접속
- 다음 명령어 실행

```
[root@krweb001 ~]# wget http://oss.ncloud.com/download/sscAgent
[root@krweb001 ~]# chmod 755 sscAgent
[root@krweb001 ~]# ./sscAgent
```

- 점검 항목 중 + OS - Linux (KISA) 선택

### 점검 내용 확인

- Security > System Security Checker > OS Security Checker 선택 후 검색 버튼 클릭



| Region | server 이름 | InstanceNo | Check list | 점검 일시                           | OS version                           |
|--------|-----------|------------|------------|---------------------------------|--------------------------------------|
| KR     | krweb001  | 4883362    | Linux      | 2020-08-10 17:50:52 (UTC+09:00) | CentOS Linux release 7.3.1611 (Core) |

- 우측의 리포트 클릭하여 내용 확인



## Lab 8(Demo)

### 인증서 발급용 로드밸런서 생성

- Target Group 생성
  - 이름 : nce-cm-lab
  - Target 유형 : VPC Server
  - VPC : lab1-vpc
  - 프로토콜 : HTTP
  - 포트 : 80

#### Target Group 생성

생성할 Target Group의 이름을 입력하고 Target 유형과 포함될 Target의 VPC를 선택해주세요  
프로토콜에 따라 연결 가능한 로드밸런서 유형이 다릅니다. (●필수 입력 사항입니다.)

|                   |                        |
|-------------------|------------------------|
| Target Group 이름 ● | nce-cm-lab             |
| Target 유형 ●       | VPC Server             |
| VPC ●             | lab1-vpc (10.0.0.0/16) |
| 프로토콜 ⓘ ●          | HTTP                   |
| 포트 ●              | 80                     |
| 메모                |                        |

- 헬스체크 프로토콜 : HTTP
- 헬스체크 포트 : 80
- URL Path : /
- HTTP Method : HEAD
- 나머지 값은 디폴트 값으로 설정

### Health Check 설정

Target에 대한 Health Check를 위한 정보를 입력하세요.  
Health Check에 실패한 서버는 로드밸런싱 대상에서 제외됩니다.(필수 입력 사항입니다.)

|                     |      |
|---------------------|------|
| 프로토콜                | HTTP |
| 포트                  | 80   |
| URL Path            | /    |
| HTTP Method         | HEAD |
| Health Check 주기 (초) | 30   |
| 정상 임계값              | 2    |
| 실패 임계값              | 2    |

- 포함시킬 서버 : web001
- Networking > Load Balancer > +로드 밸런서 생성 > 어플리케이션 로드밸런서 생성 선택
  - 로드밸런서 이름 : nce-cm
  - Network : public IP
  - 부차 처리 성능 : small
  - 대상 vpc : lab1-vpc
  - 서브넷 : KR-2 선택 후 이전 실습에서 생성한 lb 서브넷 선택
  - 공인IP : KR-2 / 공인IP신청 클릭

**로드밸런서 생성**

생성할 로드밸런서 이름을 입력하고 로드밸런서가 활성화될 VPC 및 서브넷을 선택해주세요.

로드밸런서를 생성하시면 로드밸런서 이용 시간과 트래픽 사용량에 따라 요금이 부과됩니다. (필수 입력 사항입니다.)

|          |                                                                                                 |                                                                                               |
|----------|-------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------|
| 유형       | Application                                                                                     |                                                                                               |
| 로드밸런서 이름 | <input type="text" value="nce-cm"/>                                                             |                                                                                               |
| Network  | <input type="radio"/> Private IP <input checked="" type="radio"/> Public IP                     |                                                                                               |
| 부하 처리 성능 | <input checked="" type="radio"/> Small <input type="radio"/> Medium <input type="radio"/> Large |                                                                                               |
| 대상 VPC   | <input type="text" value="lab1-vpc (10.0.0.0/16)"/>                                             |                                                                                               |
| 서브넷 선택   | <input type="checkbox"/> KR-1 <input checked="" type="checkbox"/> KR-2                          | <input type="text" value="- 선택하세요 -"/> <input type="text" value="lb-subnet   10.0.255.0/24"/> |
| 공인 IP    | <input type="checkbox"/> KR-1 <input checked="" type="checkbox"/> KR-2                          | <input type="text" value="공인 IP 신청"/> <input type="text" value="공인 IP 신청"/>                   |

각 서비스 Zone에 로드밸런서를 배치할 전용 서브넷을 생성하셔야 합니다. 전용 서브넷에 서버 인스턴스를 위치시키면 로드밸런싱이 동작하지 않습니다. 각각의 로드밸런서마다 서브넷을 생성할 필요는 없으나 가급적 C클래스(255.255.255.0) 이상을 권고합니다.

- 리스너 설정 : HTTP, 80포트 추가 후 하단의 '다음' 버튼 클릭
- Target Group : nce-cm-lab 선택
- 생성 후 접속정보 확인

|           |                                                                    |
|-----------|--------------------------------------------------------------------|
| 접속 정보     | ncs-cm-7057057-6b67a2af43b5.kr.lb.navernccp.com<br>175.106.100.235 |
| 부하 처리 성능  | Small                                                              |
| 엑세스 로그 수집 | 비활성                                                                |

**DNS에 존 레코드 추가**

- Networking > Global DNS > 상단의 도메인 추가
- 이름 : 교육용계정명.ncloudedu.com 선택
- 상단의 레코드 추가
  - 레코드명에 공란

- 레코드 타입 : A
- TTL : 15분
- 레코드값 : 로드밸런서 공인 IP
- 레코드 추가 선택
  
- 레코드명에 www
- 레코드 타입 : A
- TTL : 15분
- 레코드값 : 로드밸런서 공인 IP
  
- 레코드명에 server1
- 레코드값 : web001서버 공인 IP
- 레코드 타입 : A
- TTL : 15분
- 추가 및 설정 적용

| <input type="checkbox"/> 호스트 ▾   | 도메인                 | 레코드 타입 ▾ | 레코드 값                                                              | TTL ▾ |
|----------------------------------|---------------------|----------|--------------------------------------------------------------------|-------|
| <input type="checkbox"/> @       | edu76.ncloudedu.com | SOA      | ns1-1.ns-ncloud.com. ns1-2.ns-ncloud.com. 6 21600 1800 1209600 300 | 300   |
| <input type="checkbox"/> @       | edu76.ncloudedu.com | NS       | ns1-1.ns-ncloud.com<br>ns1-2.ns-ncloud.com                         | 86400 |
| <input type="checkbox"/> server1 | edu76.ncloudedu.com | A        | 175.106.98.12                                                      | 300   |
| <input type="checkbox"/> @       | edu76.ncloudedu.com | A        | 175.106.100.235                                                    | 300   |
| <input type="checkbox"/> www     | edu76.ncloudedu.com | A        | 175.106.100.235                                                    | 300   |

### Certificate Manager를 활용하여 인증서 발급

- Certificate Manager 클릭 > Certificate List클릭
- 상단의 '+인증서 발급'클릭
- Cloud Basic - Domain Validation Certificate 선택 후 하단의 '발급 시작' 클릭
- Certificate 이름 : nce-cm
- 도메인 이름 : (보유한 도메인 주소 입력)

- 하단의 '다음'버튼 클릭
- 검증 방식 : DNS 검증 클릭
- 하단의 '다음' 버튼 클릭
- 하단의 '다음' 버튼 클릭
- 검증이 완료되면 하단의 '확인'버튼 클릭
- Certificate List > 인증서 상세 정보 보기에서 아래 정보를 Global DNS에 도메인을 등록하고 레코드를 추가하여 진행

검증 상태

검증 방식: 도메인 검증

도메인

edu.nclocloudedu.com

검증 상태

진행중

DNS 구성에 아래 CNAME 레코드를 추가합니다.

| Record Name                                                                                         | Record Type | Record Value                                                                                             |
|-----------------------------------------------------------------------------------------------------|-------------|----------------------------------------------------------------------------------------------------------|
| <div><div></div><div>_2acea2d81c7f4b729afcba7b8784a5a9.edu.nclocloudedu.com.</div><div></div></div> | CNAME       | <div><div></div><div>_4ddb14f919b64fd992c869bedb835fe3.ncp-cm-validation.nccloud.</div><div></div></div> |

- Record Name: DNS 레코드 이름
- Record Type: CNAME
- Record Value: DNS 레코드 값에 입력되는 검증 값입니다.
- DNS 검증이 완료되면, 인증서 상태가 '정상'으로 변경

| 이름                   | 인증서 유형     | 도메인                  | 추가 도메인 | 신청일                             | 인증 시작일                          | 인증 종료일                          | 발급 기관              | 사용 여부 | 상태 ① | 개인키 등록 여부 |
|----------------------|------------|----------------------|--------|---------------------------------|---------------------------------|---------------------------------|--------------------|-------|------|-----------|
| nce-cm               | Cloud B... | edu.nclocloudedu.com | -      | 2023-12-28 17:50:25 (UTC+09:00) | 2023-12-29 09:00:00 (UTC+09:00) | 2025-01-27 20:59:59 (UTC+09:00) | NAVER Secure Ce... | 미사용   | ● 정상 | 등록됨       |
| 인증서 상태: 정상           |            |                      |        |                                 |                                 |                                 |                    |       |      |           |
| 검증 상태                |            |                      |        |                                 |                                 |                                 |                    |       |      |           |
| 검증 방식: 도메인 검증        |            |                      |        |                                 |                                 |                                 |                    |       |      |           |
| 도메인                  |            | 검증 상태                |        |                                 |                                 |                                 |                    |       |      |           |
| edu.nclocloudedu.com |            | 성공                   |        |                                 |                                 |                                 |                    |       |      |           |

## 로드밸런서에 인증서 적용

- Networking > Load Balancer > nce-cm 선택 후 리스너 설정 변경 선택
- 리스너 추가
  - 프로토콜 : HTTPS
  - 포트 : 443
  - SSL Certificate 선택 : nce
  - TLS 최소지원 버전 : TLS v1.0
  - Cipher Suite 설정 : 디폴트값 유지



- Target Group : nce-cm-lab 선택 후 하단의 '확인'버튼클릭
- 웹 브라우저로 https 로 접근

## Lab 9.

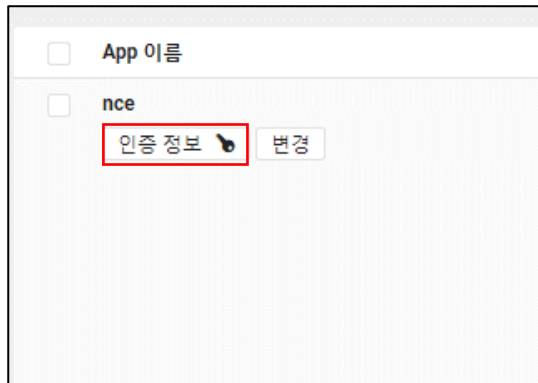
### 단축 URL 테스트

- Services > AI-NAVER API > AI NAVER API 선택 후 +Application 등록 선택
- Application 이름 : nce
- Service 선택
- Clova : Clova Voice(Premium), Clova Speech Recognition
- NAVER : All

**Service 선택**  
Application에서 이용할 Service 를 선택하세요

|                                                                                                                      |                                                                    |                                                     |             |
|----------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------|-----------------------------------------------------|-------------|
| <br><input type="checkbox"/> CLOVA  | <input checked="" type="checkbox"/> CLOVA Speech Recognition (CSR) | 서비스 설명/요금안내                                         | 개발 가이드      |
|                                                                                                                      | <input checked="" type="checkbox"/> CLOVA Voice - Premium          | 서비스 설명/요금안내                                         | 개발 가이드      |
|                                                                                                                      | CLOVA Voice - Premium은 기본료 월 90,000원 상품이며, 사용량에 따라 추가 요금이 발생합니다.   |                                                     |             |
|                                                                                                                      | <input type="checkbox"/> CLOVA Sentiment                           | 서비스 설명/요금안내                                         | 개발 가이드      |
|                                                                                                                      | <input type="checkbox"/> CLOVA Summary                             | 서비스 설명/요금안내                                         | 개발 가이드      |
| <br><input type="checkbox"/> Maps | <input type="checkbox"/> Web Dynamic Map                           | 서비스 설명/요금안내                                         | 개발 가이드      |
|                                                                                                                      | <input type="checkbox"/> Mobile Dynamic Map                        | 서비스 설명/요금안내                                         | 개발 가이드      |
|                                                                                                                      | <input type="checkbox"/> Static Map                                | 서비스 설명/요금안내                                         | 개발 가이드      |
|                                                                                                                      | <input type="checkbox"/> Directions 5                              | 서비스 설명/요금안내                                         | 개발 가이드      |
|                                                                                                                      | <input type="checkbox"/> Directions 15                             | 서비스 설명/요금안내                                         | 개발 가이드      |
|                                                                                                                      | <input type="checkbox"/> Geocoding                                 | 서비스 설명/요금안내                                         | 개발 가이드      |
|                                                                                                                      | <input type="checkbox"/> Reverse Geocoding                         | 서비스 설명/요금안내                                         | 개발 가이드      |
|                                                                                                                      | <input checked="" type="checkbox"/> NAVER                          | <input checked="" type="checkbox"/> CAPTCHA (Image) | 서비스 설명/요금안내 |
| <input checked="" type="checkbox"/> CAPTCHA (Audio)                                                                  | 서비스 설명/요금안내                                                        | 개발 가이드                                              |             |
| <input checked="" type="checkbox"/> Korean Name Romanizer                                                            | 서비스 설명/요금안내                                                        | 개발 가이드                                              |             |
| <input checked="" type="checkbox"/> nShortURL                                                                        | 서비스 설명/요금안내                                                        | 개발 가이드                                              |             |
| <input checked="" type="checkbox"/> Search Trend                                                                     | 서비스 설명/요금안내                                                        | 개발 가이드                                              |             |

- 서비스 환경 등록
  - Web 서비스 URL : <http://ai.edu.com> 입력 후 '추가'버튼 클릭
- 하단의 '등록' 버튼 클릭
- 인증정보 클릭



App 이름

nce

인증 정보 변경

- 다음 정보 기록



Application key

Application 이름 nce

Client ID  
(X-NCP-APIGW-API-KEY-ID) hdc828m72j

Client Secret  
(X-NCP-APIGW-API-KEY) 1lkxZiQKqmD4liKgVz8UZIPoVczzG4CgiGtlxmMi 재발급

- Postman 설치
  - <https://www.postman.com/downloads/> 로 이동하여 사용중인 PC OS 에 맞는 postman 다운로드 후 설치
  - postman 실행
    - ◆ POST 선택 후 호출 URL <https://naveropenapi.apigw.ntruss.com/util/v1/shorturl> 입력
    - ◆ Header 값
      - X-NCP-APIGW-API-KEY-ID : Client ID
      - X-NCP-APIGW-API-KEY : Client Secret
      - Content-Type : application/x-www-form-urlencoded



https://naveropenapi.apigw.ntruss.com/util/v1/shorturl

POST https://naveropenapi.apigw.ntruss.com/util/v1/shorturl

Params Authorization Headers (11) Body Pre-request Script Tests Settings

Headers 8 hidden

| Key                                                        | Value                             | Description |
|------------------------------------------------------------|-----------------------------------|-------------|
| <input checked="" type="checkbox"/> X-NCP-APIGW-API-KEY-ID | zcu3                              |             |
| <input checked="" type="checkbox"/> X-NCP-APIGW-API-KEY    | eH7I                              |             |
| <input checked="" type="checkbox"/> Content-Type           | application/x-www-form-urlencoded |             |
| Key                                                        | Value                             | Description |

- Body 선택 > x-www-form-urlencoded 선택 후 아래 정보 기입
  - url : (줄이고자하는 URL 기입)

Overview POST http POST http POST http POST http POST http POST http GET https GET https + ... NCP API

https://naveropenapi.apigw.ntruss.com/util/v1/shorturl

POST https://naveropenapi.apigw.ntruss.com/util/v1/shorturl

Params Authorization Headers (11) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL

| Key                                     | Value                                                   | Description |
|-----------------------------------------|---------------------------------------------------------|-------------|
| <input checked="" type="checkbox"/> url | https://kr.object.ncloudstorage.com/2022-edu50/NCE%2... |             |
| Key                                     | Value                                                   | Description |

- 우측의 send 버튼을 클릭하여 shorturl 확인

## Lab 10

### CLOVA voice API 활용

- Postman 실행
- Post 선택, 호출 URL : <https://naveropenapi.apigw.ntruss.com/tts-premium/v1/tts>
- Header 설정
  - X-NCP-APIGW-API-KEY-ID : Client ID
  - X-NCP-APIGW-API-KEY : Client Secret
  - Content-Type : application/x-www-form-urlencoded
- Body 선택 > raw 선택 후 아래 정보 기입
  - speaker=nara&speed=0&format=mp3&text=안녕하세요. 저는 클라우드 봇입니다.

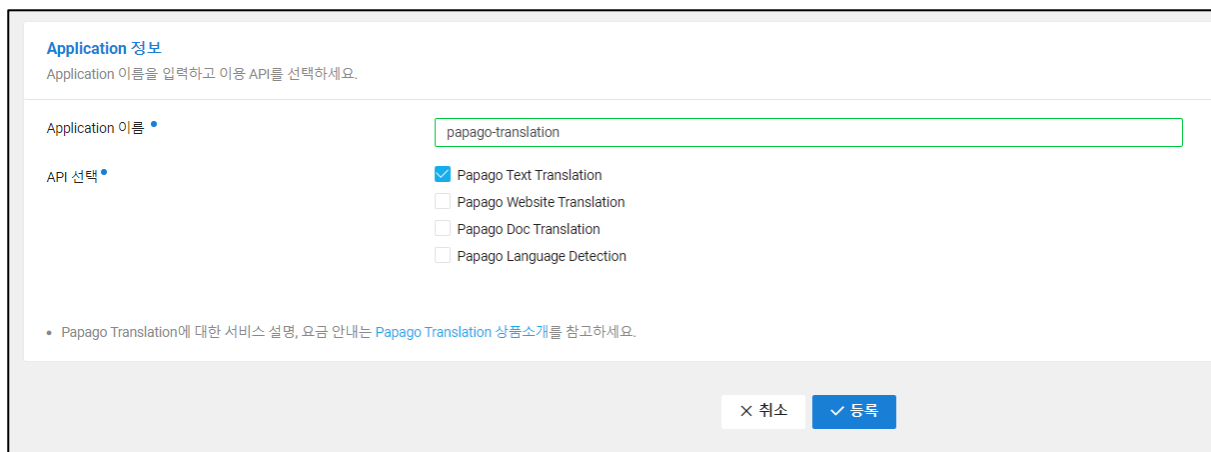
### Clova Speech Recognition API 호출

- Postman 실행
- Post 선택, 호출 URL : <https://naveropenapi.apigw.ntruss.com/recog/v1/stt>
- Parameter 설정
  - lang : Kor
- Header 설정
  - X-NCP-APIGW-API-KEY-ID : Client ID
  - X-NCP-APIGW-API-KEY : Client Secret
  - Content-Type : application/octet-stream
- Body 선택 > binary 선택 후 녹음 파일(mp3)선택
  - 녹음 파일은 60 초 이하

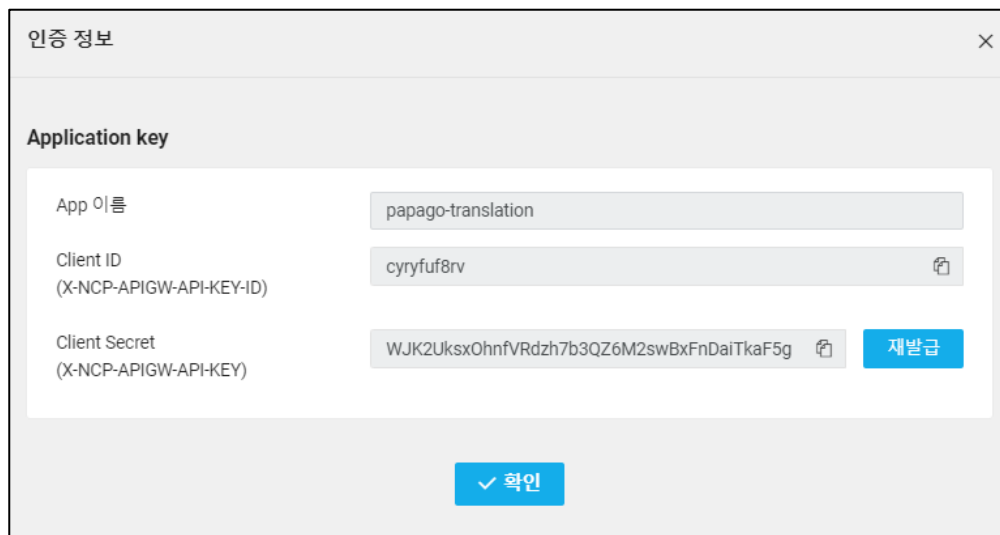
## Lab 11

### Curl을 이용한 Papago Translation 구현

- Services > AI services > Papago translation 선택
- Application 등록 버튼 클릭
  - Application 이름 : papago-translation
  - Papago Text Translation 선택 후 하단의 '다음'버튼 클릭



- Papago-translation 선택 후, 상단의 '인증정보' 클릭



- ClientID, Client Secret 키 값 확인
- Curl을 이용하여 다음과 같이 명령어 수행
- Web001 서버에 접속
- 다음 명령어 수행

```
curl -i -X POST ₩
-H "Content-Type:application/x-www-form-urlencoded" ₩
-H "X-NCP-APIGW-API-KEY-ID:{client ID}" ₩
-H "X-NCP-APIGW-API-KEY:{Secret}" ₩
-d "source=ko" ₩
-d "target=en" ₩
-d "text=오늘 날씨가 좋네요" ₩
'https://naveropenapi.apigw.ntruss.com/nmt/v1/translation'
```

- Json 형식으로 요청 가능하며 이 경우 보다 간결한 요청을 보낼 수 있다.

```
curl -i -X POST ₩
-H "X-NCP-APIGW-API-KEY-ID:{앱 등록 시 발급받은 Client ID}" ₩
-H "X-NCP-APIGW-API-KEY:{앱 등록 시 발급 받은 Client Secret}" ₩
-H "Content-Type:application/json" ₩
-d ₩
'{
 "source": "{원본 언어 코드}",
 "target": "{번역 결과 언어 코드}",
 "text": "{번역할 text}"
}' ₩
'https://naveropenapi.apigw.ntruss.com/nmt/v1/translation'
```

- 응답은 어떤 형식이던 Json 형식으로 응답

```
{"message":{"@type":"response","@service":"naverservice.nmt.proxy","@version":"1.0.0","result":{"srcLangType":"ko","tarLangType":"en","translatedText":"Hi."}}}
```

## Lab 12

액션을 만들고 트리거를 통해 액션을 실행시키는 방식을 알아봅니다.

액션은 독자적인 특정 액션을 실행시킬 수 있지만, 트리거에 파라미터를 넣어 액션을 호출할 시, 다른 방식으로도 실행이 가능합니다.

액션을 단독으로 실행시킬 때와 트리거에 파라미터를 넣어 액션을 실행시킬 때, 결과 값이 어떻게 달라지는지에 집중해서 확인해보고, 외부와 통신할 수 있는 URL 주소를 통해서도 트리거를 작동시키는 것까지 함께 살펴봅니다. 마지막으로 액션에 코드가 아닌 여러 코드파일로 이루어진 압축파일을 이용하여 네이버 클라우드 플랫폼 내의 다른 상품과 연동하여 사용하는 방법에 대해서도 실습해봅니다.

### 테스트 환경 설정

- Action은 Private Subnet 서버에서 실행할 수 있습니다.
- 테스트 용이므로 lab1-vpc-redis-subnet 을 임시로 이용합니다
- Route Table 설정에 외부 통신은 NAT를 통하도록 설정합니다.

### 내 패키지 생성

- Cloud Functions 탭 > Action 선택, +패키지 생성 클릭
- 패키지 생성
- 패키지 이름 : hello

**Package 생성**

패키지는 단일 또는 다수의 액션을 분류, 관리하는 목적의 단위입니다.  
패키지에 포함된 액션에서 공통으로 필요한 값은 디폴트 파라미터를 통해 적용할 수 있습니다.  
( • 필수 입력 사항입니다.)

패키지 이름 •

패키지 설명

0 / 3000 bytes

디폴트 파라미터 Ⓞ

| 1                             |
|-------------------------------|
| <input type="text" value=""/> |

0 / 3000 bytes

× 취소 ✓ 생성

### 트리거 생성

- Cloud function > +트리거 생성 클릭
- 트리거 종류 : Basic
- 트리거 이름 : lab-hello
- 외부 연결 주소 생성
- Product : '+생성'버튼 클릭 > Product 이름에 'lab-hello' 입력 후 우측의 '저장'버튼 클릭
- API : '+생성'버튼 클릭 > API 이름에 'lab-hello' 입력 후 우측의 '저장'버튼 클릭
- Stage : '+생성'버튼 클릭 > Stage이름에 'lab-hello' 입력
- 인증 : none 선택
- 저장하고 액션 연결하기 버튼 클릭 후 저장 버튼 클릭

### 액션 생성

- +액션 생성 클릭
- 트리거 종류: Basic 선택
- 이름: lab-hello
- '추가'버튼 클릭 > 하단의 '다음'버튼 클릭
- 패키지 : hello 선택
- 타입 : 일반 액션
- 이름 : helloNCP
- 소스코드 언어 : nodejs:8
- 타입 : 코드
- 코드 :

```
function main(params) {
 return {payload: 'Hello, ' + params.name + ' from ' + params.place + '?'};
}
```

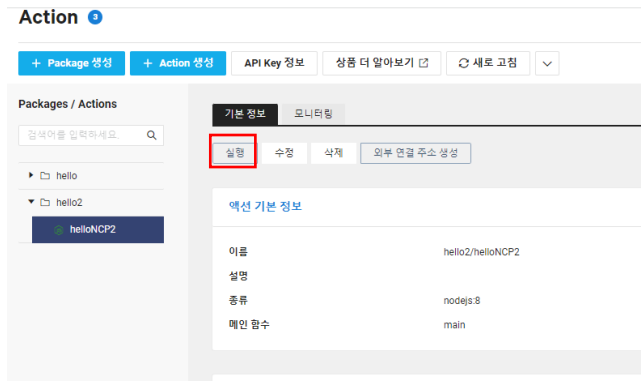
- 디폴트 파라미터:  

```
{"name":"stranger","place":"somewhere"}
```
- VPC :lab1-vpc 선택
- Subnet :lab1-vpc 내 private subnet 선택

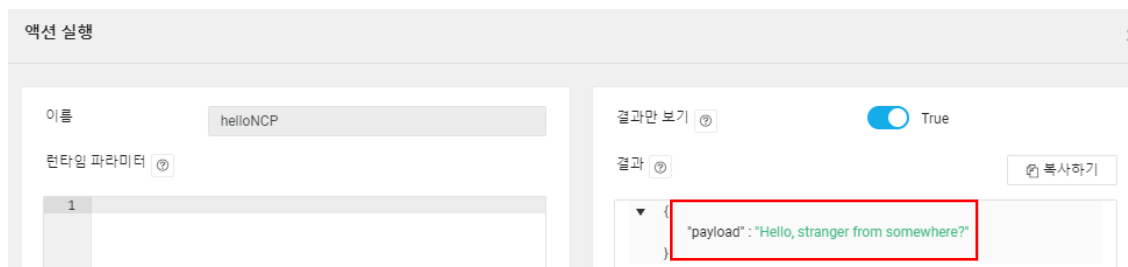
- 옵션 설정 : Default선택
- 하단의 '생성' 버튼 클릭

### 액션 단일 동작

- Cloud Functions 탭 > Action > hello 패키지 > helloNCP액션 선택
- '실행' 버튼 클릭



- 결과만 보기를 True로 변경
- 실행 버튼 클릭
- 아래와 같이 Hello, stranger from somewhere? 가 출력되는지 확인



### 트리거로 액션 동작

- Cloud Functions 탭 > Trigger > Basic > lab-hello 트리거 선택
- 트리거 실행 버튼 클릭
- 런타임 파라미터 :

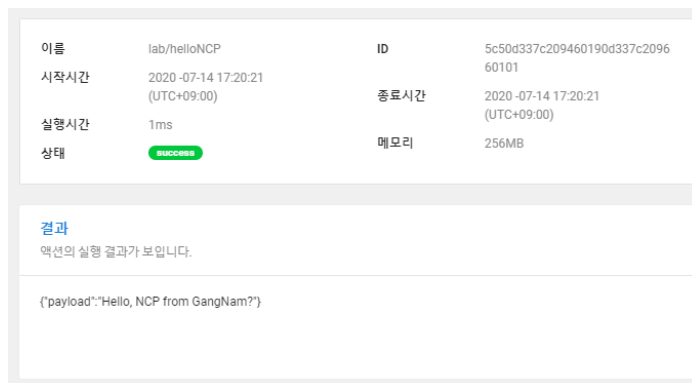
```
{
 "name" : "NCP",
 "place" : "GangNam"
}
```

}

- 결과만 보기를 True로 변경
- 실행 버튼 클릭
- Activation ID 확인
- Cloud Functions 탭 > Action > hello 패키지 > helloNCP 액션 선택
- 모니터링 탭 선택



- Activation ID 결과값의 자세히 보기 클릭
- 아래와 같이 Hello, NCP from GangNam? 이 출력되는지 확인



## 외부 URL을 통해 호출

- Cloud function > Action > hello 패키지 > helloNCP클릭 > 외부 연결 주소 생성 버튼 클릭
  - Product : lab-hello
  - API : lab-hello
  - Stage : lab-hello
  - 인증 : None 선택 후, 하단의 완료 버튼 클릭



외부 연결 주소 생성

API Gateway 서비스를 이용하여 외부 연결 주소를 설정합니다. 더 자세한 상품 설명은 [API Gateway 사용자 가이드](#)를 참고해 주세요. (필수 입력 사항입니다.)

Product

lab-hello

+ 생성

API

lab-hello

+ 생성

Stage

lab-hello

+ 생성

인증

☐ IAM
☒ NONE

X 취소

✓ 완료

- 서버에서 외부 연결 호출 URL로 POST 요청을 전송

- 명령어 :

```
curl -X POST <URL주소> -H "Content-Type:application/json" -d '{"name":"NCP","place":"Seoul"}'
```

```

[root@cidc-org ~]# curl -X POST https://9flyj7yr2k.apigw.ntruss.com/labtrigger/labtrigger/cI3Uwc3ga
{"activationId":"7b988bbc668c4720988bbc668c672021"} [root@cidc-org ~]#

```

- HelloNCP 액션 > 모니터링 탭에서 ActionId 선택 후 자세히 보기 클릭 > 결과로 {"payload":"Hello, NCP from Seoul?"} 가 노출되었는지 확인

## Lab 13

소스 저장소인 SourceCommit 리파지토리 생성법과 사용 방법에 대해 알아본 후, 서버에서 SourceCommit 리파지토리에 원격으로 연결하여 소스를 저장/업데이트 할 수 있는 방법에 대해 알아봅니다.

SourceCommit 을 이용하는 계정은 Sub Account라는 가정하에, Sub Account를 먼저 생성하고 특정 계정에 특정 권한을 부여하는 방법등에 대해 먼저 실습을 진행합니다.

### Web001,web003 서버에 아래 스크립트 반영 필요

```
yum install -y tomcat
systemctl enable tomcat
systemctl start tomcat
yum install -y java-11-openjdk-devel
mkdir -p /var/lib/tomcat/webapps/ROOT/WEB-INF/classes
```

### 서브계정 생성

- Management and Governance> Sub Account 클릭
- Dashboard 에서 Sub Account를 위한 접속 페이지 생성
- Sub Accounts 에서 +서브 계정 생성 클릭
- 로그인 아이디 : student
- 사용자 이름 : student
- 콘솔 접근 및 API 접근 허용
- 로그인 비밀번호 직접 입력 > ncloud<오늘날짜>! Ex) 7월 20일인 경우, ncloud0720!
- 비밀번호 재설정 알림 : 체크 해제
- 생성이 완료된 student 계정을 클릭

| <input type="checkbox"/>            | 로그인 아이디 | 사용자 이름  | Console 접근 | API 접근 | 상태     |
|-------------------------------------|---------|---------|------------|--------|--------|
| <input checked="" type="checkbox"/> | student | student | ✓          | ✓      | ● 사용 중 |
| <input type="checkbox"/>            | dev01   | dev01   | ✓          |        | ● 사용 중 |

- 하단 정책 탭에서 개별 권한 추가 버튼 클릭
- NCP\_SOURCECOMMIT\_MANAGER, NCP\_SOURCEBUILD\_MANAGER, NCP\_VPC\_SOURCEDEPLOY\_ADMIN, NCP\_VPC\_SOURCEPIPELINE\_MANAGER 권한 선택 후 추가 버튼 클릭

### SourceCommit 리파지토리 생성

- Developer Tools > SourceCommit 클릭
- + 리파지토리 생성 버튼 클릭
- 리파지토리 이름 : lab-repo
- 나머지 설정 값은 디폴트로 유지
- 하단의 다음 버튼 클릭
- File safer 연동 안함 > 다음 버튼 클릭
- 하단의 생성 버튼 클릭

#### Sub Account 접속 및 HTTPS 접근용 Git Client 설정

- 다른 브라우저를 하나 더 띄워, Sub Account 접속 페이지로 들어간 후, student 계정으로 로그인
- lab-repo 리파지토리 선택 후, GIT 계정/GIT SSH 설정 버튼 클릭
- Git Client 패스워드를 'ncp!@#123' 으로 설정 후 적용 버튼 클릭

#### Git Client SSH 접근용 자격증명 발급

- Web001 서버에 접속
- gitlab 이란 이름의 디렉토리 생성  

```
$ mkdir ~/gitlab
```

```
$ cd ~/gitlab
```
- ssh-keygen 명령어 입력  

```
$ ssh-keygen
```

```
Enter file in which to save the key (/root/.ssh/id_rsa): id_rsa_lab
```

```
Enter passphrase (empty for no passphrase): ncp!@#123
```

```
Enter same passphrase again: ncp!@#123
```
- 생성된 public rsa key 결과값을 복사  

```
$ cat /root/gitlab/id_rsa_lab.pub
```

```
<결과값/ ssh-rsa ~>
```
- Root계정의 SourceCommit > Git 계정/Git SSH 설정 클릭 > GIT SSH 설정에 해당 값 입력 후 등록 클릭

## SSH 퍼블릭 키 등록

```
ssh-rsa
AAAAB3NzaC1vc2EAAAADAQABAAQCAQ08o27ez6EJQmVkcmy62+INDyGopIvWLUAWht/xqGLLTPmBDpYgmej
9Lg668p7i+ruwz7AYtkhUcqeKLI0ZkiDIEoc4fTXotsj0TZCGMlonC18E4dyUh2ZnLMenlo2F2L+4b0IJH/1zS/+IR6BsUx
IsuBWiPVbCbXWtWtiMailb4V74/nBvm1B+tEQIRri93AgPmujR82BOJBjYoZOpI3QWeUYEZtixcKbQtQ6Zxon6xz+MVAf+
Z+a7xmASclBj8qw13pYLWqTaH31ge3WMuiqD8fUIT7mcg3E4kvznrKGH07Nd5WcViygMfHQ0aSgwhOuFo9FCz72M
W4UCiWiY5 mignon.kim@navercorp.com
```

✓ 등록

- Web001 서버에서 ssh key 를 추가 후 저장

```
$ eval $(ssh-agent)
```

```
$ ssh-add ~/gitlab/id_rsa_lab
```

```
Enter passphrase for /root/gitlab/id_rsa_lab:ncp!@#123
```

```
$ ssh-add -l
```

- ssh config 파일을 생성하여, private key 파일 경로 저장

```
$ mkdir ~/.ssh
```

```
$ vi ~/.ssh/config
```

```
Host devtools.ncloud.com
```

```
User <SSH 키>
```

```
IdentityFile ~/gitlab/id_rsa_lab
```

## 등록된 SSH 퍼블릭 키

| SSH 키       | 업로드 날짜                          | 상태            |
|-------------|---------------------------------|---------------|
| 입력할 SSH 키 🔍 | 2020-12-15 20:24<br>(UTC+09:00) | 활성화 비활성화 ✕ 삭제 |

## 서버에 로컬 리파지토리 생성 후 원격 리파지토리 업데이트

- web001 서버에 접속
- gitlab 아래 sourcecommit 디렉터리 생성
 

```
$ mkdir ~/gitlab/sourcecommit
```

```
$ cd ~/gitlab/sourcecommit
```
- 로컬 리파지토리 생성 및 사용자 정보 설정

```
$ git init
```

```
$ git config --global user.name "student"
```

```
$ git config --global user.email "사용자 이메일"
```

```
$ touch readme.txt
```

- 로컬 리파지토리에 readme.txt 파일 추가 후 커밋

```
$ git add readme.txt
```

```
$ git commit -m "First Commit"
```

```
$ git status
```

- 원격 저장소(Sourcecommit) 등록 후 확인

```
$ git remote add origin <리파지토리 URL>
```

```
$ git remote -v
```

- 원격 저장소의 master 브랜치를 가져온 후, 원격 저장소에 업데이트

```
$ git pull --rebase origin master
```

```
$ git push origin master
```

- SourceCommit에 들어가 read.txt 파일이 추가되었는지 확인

- 만약 fatal error 발생 시 아래 명령어대로 수행

- git remote set-url {remote git 주소 복사}

## Lab 14

앞서 로컬 리포지토리에서 SourceCommit 리파지토리와 어떻게 연동하여 소스 업데이트를 할 수 있는지 살펴보았습니다. 이번 Lab에서는 업데이트한 소스를 네이버 클라우드 상에서 어떻게 빌드할 수 있는지 알아봅니다.

빌드하기 전, 빌드 결과물을 저장할 수 있는 Object Storage 생성부터 시작합니다.

### Object Storage 생성

- 다시 마스터 계정으로 돌아와 Storage > Object Storage 에서 + 버킷 생성 클릭
- 버킷 이름 : gitlab<생년월일>
- 나머지는 default선택으로 두고 버킷 생성
- 버킷을 선택 후, 새폴더 버튼 클릭
- 폴더명 : sourcebuild

### 빌드용 파일 업로드

- HelloServlet.java 파일을 만들어 원격저장소에 push

```
$ vi ~/gitlab/sourcecommit/HelloServlet.java
```

- 링크 : <https://kr.object.ncloudstorage.com/11-edu-yangju/import%20java.txt>

```
import java.io.*;

import javax.servlet.*;

import javax.servlet.http.*;

public class HelloServlet extends HttpServlet {

 public void doGet(HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException {

 PrintWriter out = res.getWriter();

 out.println("Hello, NCP!");

 }

}
```

```
$ git add HelloServlet.java
```

```
$ git commit -m "HelloServlet.java added"
```

```
$ git push origin master
```

### 빌드 프로젝트 생성

- Dev Tools > SourceBuild > + 빌드 프로젝트 생성 클릭
- 빌드 프로젝트 이름 : gitlab
- 빌드 대상 : Sourcecommit
- 빌드 대상 리파지토리 : lab-repo
- 브랜치 : master
- 빌드 환경 이미지 : SourceBuild에서 관리되는 이미지
- 운영 체제 : ubuntu 16.04
- 빌드 런타임 : java
- 빌드 런타임 버전 : java8 버전
- 컴퓨팅 유형 : 2vCpu 4GB 메모리
- 타임 아웃 : 60분
- 하단의 '다음'버튼 클릭
- 빌드 명령어 :

#### 1. 빌드 전 명령어 :

```
apt-get -y update
apt-get install -y tomcat7
```

#### 2. 빌드 명령어

```
javac -classpath /usr/share/tomcat7/lib/servlet-api.jar HelloServlet.java
```

- 하단의 다음 버튼 클릭
- 빌드 결과물 : 결과물 저장

- 빌드 결과물 경로 : ./HelloServlet.class
- 업로드 할 Object Storage : 앞에서 생성한 버킷 선택
- Object Storage 폴더 경로 : sourcebuild
- 저장될 파일 이름 : HelloServlet
- 나머지는 Default 설정 그대로 남겨두고 하단의 '다음'버튼 클릭
- 로그 상품 연동, 보안 상품 연동은 디폴트값 유지하고 하단의 '다음'버튼 클릭
- '생성'버튼 클릭
- 생성 완료 후, gitlab 프로젝트 선택 후 상단의 '빌드로 이동'클릭
- 우측 상단의 '빌드 시작하기' 버튼 클릭



작업 결과가 Success 인지 확인



## Lab 15

소스 코드 빌드가 완료되면, 실제로 서버에 배포해야 합니다. 이번 Lab에서는 SourceDeploy를 통해 쉽고 편하게 빌드 이미지를 배포하는 방법에 대해 실습해봅니다. 마지막으로 SourcePipeline을 통해 빌드와 배포 자동화를 어떻게 할 수 있는지에 대해 알아봅니다.

### Web001 Agent 설치

- ncloud.com 메인 > 마이페이지 > 계정 관리 > 인증키 관리 > 신규 API 인증키 생성 (없을 경우)  
Access key ID와 Secret Key ID를 메모장에 저장 (Access Key의 상태가 사용 중 이어야 함)
- accesskey와 secretkey 을 개인 API 인증키로 치환하여, 아래 명령어 수행  

```
$ echo $'NCP_ACCESS_KEY=accesskey\nNCP_SECRET_KEY=secretkey' > /opt/NCP_AUTH_KEY
```

```
$ wget https://sourcedeploy-agent.apigw.ntruss.com/agent/vpc/download/install
```

```
$ chmod 755 install
```

```
$./install
```

```
$ rm -rf install
```

```
$ service sourcedeploy start
```

```
$ service sourcedeploy status
```

### 배포 프로젝트 생성

- Dev Tools > SourceDeploy > + 배포 프로젝트 생성 클릭
- 프로젝트 이름 : gitlab
- dev stage : 설정 선택
- 배포 타겟: 서버
- 적용 서버 : web001 선택 하단의 다음 버튼 클릭
- 배포 프로젝트 생성 버튼 클릭
- gitlab 프로젝트를 선택 후, 배포 시나리오 부분에서 '생성' 버튼 클릭

배포 Stage +

dev

배포 환경 ?

설정 변경 삭제

배포 방법 기본 Region 한국

배포 타겟 Server 확인

배포 시나리오 ?

생성 설정 변경 삭제

- 배포 시나리오 이름 : test
- 배포 전략 : 기본
- 배포 과정 : 순차배포
- 배포 파일 위치 : Source Build
- 빌드 프로젝트 선택 : gitlab 선택 후 하단의 '다음'버튼 클릭
- 소스 파일 배포 경로 : /.
- 배포 경로 : /var/lib/tomcat/webapps/ROOT/WEB-INF/classes 입력 후 추가 버튼 클릭

배포 전 실행

실행 계정 실행 명령 명령 추가

명령어를 수행할 서버 계정입니다. 서버에서 실행할 명령어입니다. + 추가

데이터가 없습니다.

파일 배포

소스 파일 경로 배포 경로 파일 배포 추가

/ /var/lib/tomcat/webapps/ROOT/WEB-INF/classes + 추가

- 하단의 다음 클릭 후, 배포 시나리오 생성 버튼 클릭

## 배포 시나리오 실행

- gitlab 프로젝트 선택 후, 상단의 배포로 이동 버튼 클릭



- test 시나리오를 클릭 후, 배포 시작하기 클릭
- 배포가 끝나고 시나리오의 상태가 배포 완료 인지 확인
- Web001 서버의 /var/lib/tomcat/webapps/ROOT/WEB-INF/classes 디렉터리에 HelloServlet 클래스 파일이 배포된 것을 확인

### 웹페이지 접속

- 서버에서 web.xml 파일을 열고, </description> 밑에 url 정보 추가  

```
$ cd /var/lib/tomcat/webapps/ROOT/WEB-INF
```

```
$ wget https://kr.object.ncloudstorage.com/nce/web.xml
```
- tomcat 서비스 재시작  

```
$ systemctl restart tomcat
```
- Services > VPC > NetworkACL에 가서 lab1-vpc-web-nacl 선택 후 상단의 'Rule설정' 클릭
- 우선순위 : 30 / 프로토콜 : TCP / 접근소스 : 0.0.0.0 / 포트: 8080 / 허용여부 : 허용 추가후 하단의 '적용' 버튼 클릭
- Services > Server > ACG로 이동 lab1-web-acg 선택 후 상단의 'ACG 설정' 클릭
- 프로토콜 : TCP / 접근 소스 : 0.0.0.0 / 허용 포트 : 1-65535 기입 후 우측의 '추가' 버튼 클릭 후 하단의 '적용' 버튼 클릭
- 웹 브라우저를 열고 <web001 서버IP>:8080/hello 페이지에 접속하여 Hello, NCP! 문구가 보이는지 확인

### 파이프라인 생성

- Web003 서버에 Hello, 이름! 를 출력하기 위한 사전 작업을 진행
  - Web003 서버에 Public IP가 부여되어 있지 않다면 Public IP 부여

- Web003 서버에서 아래 명령어 실행 (deploy를 위한 사전 작업 및 web.xml 파일 다운로드)

```
$ vi /root/init.sh
```

→ <https://kr.object.ncloudstorage.com/academy2024/bin.txt> 내용을 복사해서 init.sh 파일에 붙여넣기

```
$ chmod 755 init.sh && sed -i -e 's/₩r$//' init.sh && bash init.sh
```

→ 위 스크립트를 실행하여 access key와 secret key 입력

- Web001 서버에서 HelloServlet.java 출력 문구 수정 후 리파지토리에 변경내용 업데이트

```
$ vi ~/gitlab/sourcecommit/HelloServlet.java
```

문구 수정 >> Hello, 수강생 성함!

```
$ cd ~/gitlab/sourcecommit
```

```
$ git add HelloServlet.java
```

```
$ git commit -m "HelloServlet.java Revised"
```

```
$ git push origin master
```

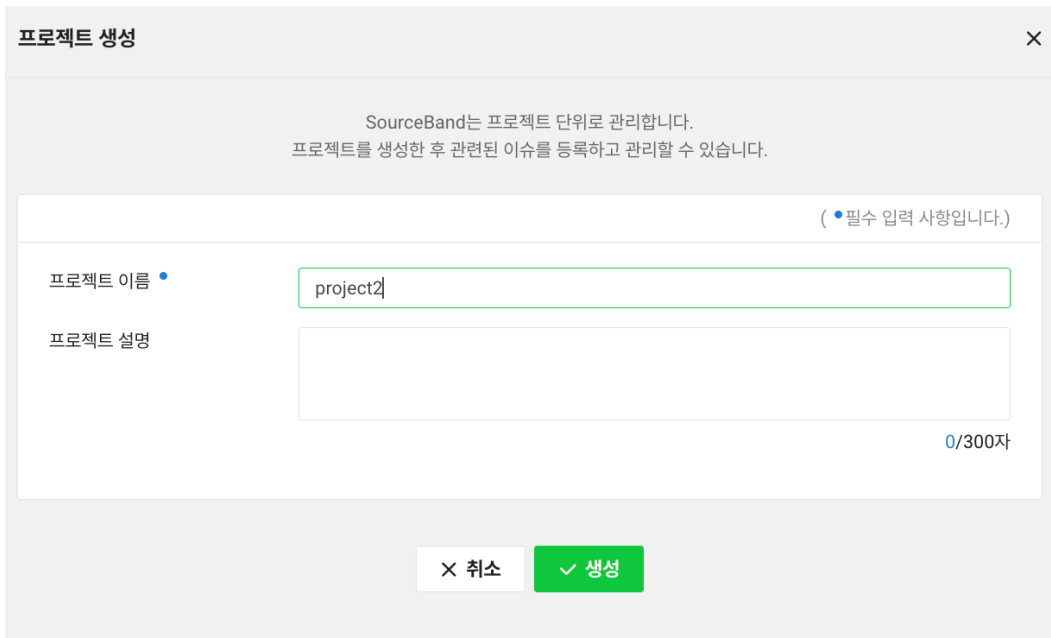
- SourceCommit > lab-repo 에 들어가 Code 및 Commit 내역 업데이트 확인
- SourceDeploy > gitlab 선택 > 배포환경 > 설정 변경 클릭 > 적용 서버를 web003 로 변경 > 하단의 '저장'클릭
- Dev Tools > SourcePipeline > + 파이프라인 생성
- 파이프라인 이름 : gitlab 입력 후 '다음'클릭
- 작업추가 클릭
- 작업 이름 : lab-build
- 타입 : SourceBuild
- 프로젝트 : gitlab
- 하단의 '확인'버튼 클릭
- 상단의 '작업추가' 클릭
- 이름 : lab-deploy
- 타입 : SourceDeploy
- 프로젝트 : gitlab

- 스테이지 : dev
- 시나리오 : test 하단의 '확인'버튼 클릭
- lab-build 작업의 + 버튼을 클릭, 선행작업 없음 선택 후 확인 클릭
- lab-deploy 작업의 + 버튼을 클릭, 선행작업 lab-build 선택 후 확인 클릭
- 하단의 '다음' 클릭 후 파이프라인 생성 클릭
- 파이프라인 'gitlab' 선택 후 상단의 '파이프라인으로 이동'클릭
- 상단의 파이프라인 실행하기 클릭
- Web003 서버에서 톡넷 재실행
- 웹 브라우저에서 <web003 서버 IP>:8080/hello 로 접속하여 Hello, 이름! 출력 여부 확인

## Lab 16

Sourceband 를 활용한 업무 협업

- 1) Services > Developer tools > Sourceband 클릭
- 2) 상품 이용 신청
- 3) Organization > '조직 등록' 클릭
- 4) 서비스 플랜 - 'Basic' 선택 후 하단의 '다음' 버튼 클릭
- 5) 조직명 - academy 입력
- 6) 참여 중인 프로젝트 추가 > '+추가' 버튼 클릭
  - 프로젝트 이름 : project2 입력 후 하단의 '생성'버튼 클릭



- 7) 'Board'로 이동하면 '+신규생성' 클릭 > '이슈 추가' 클릭
  - 이슈제목 : 코드 최종 리뷰
  - 타입 : Issue

- 소속칼럼 : To-do
- 추가할 담당자 : student
- DevOps 작업 > + 작업 추가
- SourceDeploy 선택 후 '검색' 버튼 클릭
- 이전 과정에서 생성한 작업 ID 선택 후 하단의 '확인' 버튼 클릭

DevOps 작업 추가

SourceBuild

SourceDeploy

SourcePipeline

프로젝트

ALL

실행 기간

시작 날짜 - 종료 날짜

검색

| <input type="checkbox"/> 작업 ID             | 작업 구분               | 시작 시간               | 완료 시간               | 요청자                    |
|--------------------------------------------|---------------------|---------------------|---------------------|------------------------|
| <input checked="" type="checkbox"/> 162647 | gitlab > dev > test | 2024-03-14 14:31:13 | 2024-03-14 14:31:34 | edu_50@navercorp.co... |
| <input type="checkbox"/> 162645            | gitlab > dev > test | 2024-03-14 14:26:39 | 2024-03-14 14:26:56 | edu_50@navercorp.com   |
| <input type="checkbox"/> 162643            | gitlab > dev > test | 2024-03-14 14:19:52 | 2024-03-14 14:20:13 | edu_50@navercorp.com   |
| <input type="checkbox"/> 157731            | gitlab > dev > test | 2024-02-21 11:44:38 | 2024-02-21 11:44:55 | edu_50@navercorp.com   |

<

<<

<

1

>

>>

>

× 취소

✓ 확인

- 입력 내용 확인 후 '저장' 버튼 클릭

이슈 추가

크게 보기

이슈 제목

코드 최종 리뷰

이슈 설명

0/5000 자

세부 정보

타입

Issue

상위 이슈 그룹

이슈를 할당할 상위 이슈 그룹 선택

소속 칼럼

To Do

태그

태그를 선택해 주십시오.

담당자 1

추가할 담당자를 검색해 주십시오.

aaa

DevOps 작업 1

+ 작업 추가

gitlab > dev > test (162647)

2024-03-14 14:31:13

edu\_50@navercorp.com

취소

저장

## 8) 이슈에 대한 댓글 등록

- 코드 최종 리뷰 Issue 클릭
- 댓글 창에 'student 님, 빠르게 코드 리뷰 부탁드립니다'로 입력 후 '등록' 버튼 클릭

## 9) 이슈에 대한 상태 변경

- 리뷰가 완료되면, 'To do' 칼럼에서 'Done'칼럼으로 이동
- (이슈 박스를 드래그해서 Done 칼럼으로 이동)