

# 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.

## 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. 추가 스토리지에 대해 스냅샷 생성

- Services > Compute > Server > Storage 선택
- Inxsvr-org-disk1 선택 후 스토리지 설정 > 스냅샷 생성 선택
- 전체 스토리지 스냅샷 선택
- 스냅샷 이름에 Inxsvr1-disk1-snap1 입력 > 다음 클릭 > 생성 버튼 클릭

(필수 입력 사항입니다.)

원본 스토리지 이름 Inxsvr-disk1

스토리지 크기 10 GB

스냅샷 유형 ☒ 전체 스토리지 스냅샷 ☐ 증분 스냅샷 제공 가능한 제한이 없습니다.

스냅샷 이름 Inxsvr1-disk1-snap1

메모

0 / 1000 Bytes

생성된 스냅샷은 스토리지에 저장되며 저장되는 스토리지에 대해서는 요금이 부과됩니다.

× 취소 다음 >

- Inxsvr-org-disk1 스토리지에 데이터 추가

```
[root@Inxsvr-org ~]# cp -rf /var/log /disk1
```

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

### 2. 증분 스냅샷 생성

- Services > Compute > Server > Storage 선택
- Inxsvr-org-disk1 선택 후 스토리지 설정 > 스냅샷 생성 선택
- 증분 스냅샷 선택
- 스냅샷 이름에 Inxsvr1-disk1-snap2 입력 > 하단의 '다음' 클릭 > '생성' 클릭

(필수 입력 사항입니다.)

원본 스토리지 이름	Inxsvr-disk1
스토리지 크기	10 GB
스냅샷 유형	<input type="radio"/> 전체 스토리지 스냅샷 <input checked="" type="radio"/> 증분 스냅샷
스냅샷 이름	Inxsvr1-disk1-snap2
베이스 스냅샷	Inxsvr1-disk1-snap1    생성 일시 : 2021-06-21 오전 7:46 (UTC+09:00)
메모	<div></div> 0 / 1000 Bytes

• 생성된 스냅샷은 스토리지에 저장되며 저장되는 스토리지에 대해서는 요금이 부과됩니다.

× 취소    다음 >

- Services > Compute > Server > Snapshot 선택
- 생성된 스냅샷의 Depth 비교

<input type="checkbox"/> 스냅샷 이름	스냅샷 유형	Depth (n)	상태	원본 스토리지 이름	크기
<input type="checkbox"/> Inxsvr1-disk1-snap2	증분 스냅샷	1	● 생성됨	Inxsvr-disk1	0 GB
<input checked="" type="checkbox"/> Inxsvr1-disk1-snap1	전체 스토리지 스냅샷	0	● 생성됨	Inxsvr-disk1	10 GB

상세정보

스냅샷 이름 (Instance ID)	Inxsvr1-disk1-snap1(7030974)	생성 일시	2021-06-21 오전 7:46 (UTC+09:00)
----------------------	------------------------------	-------	--------------------------------

### 3. 스토리지 암호화된 서버 생성

- Services > Compute > Server > Server > +서버 생성 선택
- 이미지는 **Centos-7.8-64**를 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 : High-CPU / vCPU 2개, 메모리 4GB, 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

g2

서버 타입

High CPU

[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

1

서버 이름

최소 3글자, 최대 30자 (서버 이름을 작성하지 않으면 자동 생성됩니다.)



입력하신 서버 이름으로 hostname을 설정합니다.

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

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

#### 4. 스토리지 암호화된 스토리지 생성

- Services > Compute > Server > Server > Inxsvr-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@lnxsvr-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

```
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 > Kubernetes Service > 생성하기
  - 클러스터 이름 : k8s-XXX(실습 날짜 기입)
  - 클러스터 버전 : 1.24.10
  - CNI Plugin : cilium
  - VPC : lab1-vpc
  - 가용 zone : KR-2
  - 네트워크 타입 : Public
  - subnet : lab1-vpc-web-subnet
  - LB subnet : lab1-vpc-lb-sub3
  - 최대 노드 수 : 10
  - Audit Log : 미설정

클러스터 설정

클러스터 정보를 입력하세요. (\*필수 입력 사항입니다.)

클러스터 이름

k8s-edu

Kubernetes 버전

1.24.10

CNI Plugin

cilium

VPC

lab2-vpc | 10.0.0.0/16

VPC 생성

가용 Zone

KR-2

네트워크 타입

Public

Subnet

lab2-vpc-web-subnet | KR-2 | 10.0.1.0/24 | Public

Subnet 생성

LB Private 서브넷

lab1-vpc-lb-sub3 | KR-2 | 10.0.5.0/24 | Private

LB Subnet 생성

최대 노드 수

☒ 10
 ☐ 50

Audit Log

☐ 설정
 ☒ 미설정

ACG 설정

Kubernetes Service를 위한 ACG는 자동 생성됩니다. (예: nks-\*)  
 클러스터에 접근을 위한 접근 정보는 클러스터의 생성 완료 후에 확인하실 수 있습니다.

주의사항

• Kubernetes Service 콘솔 이외의 기능으로 서버 자원(VM), VPC, Subnet을 수동으로 정지 또는 삭제하는 경우 클러스터에 오류가 발생할 수 있습니다.  
 • Kubernetes 워커노드 자원을 반납하는 경우, Kubernetes Service 콘솔에서 반납해 주시기 바랍니다.

다음 >

- 모든 정보 기입 후, 하단의 다음 버튼 클릭
- 노드 풀 이름 : default-pool
- 서버 이미지 이름: Ubuntu 20.64
- 서버 타입 : [Standard] vCPU 2EA, Memory 8GB
- 노드 수 : 1
- Subnet : 자동 할당
- Kubernetes label 및 taint는 설정 안함
- 하단의 [추가] 버튼 클릭 후, 하단의 다음 버튼 클릭

노드풀 설정

클러스터 내에서 구성이 모두 동일한 노드 그룹  
(최대 노드 수 이내로 구성해야 합니다.)

노드풀

+

설정

노드풀 이름

서버 이미지 이름

서버 타입

노드 수

Subnet

Kubernetes Label  ?

Taint  ?

클러스터 최대 노드 수 10(0/10)

+ 추가

클러스터 정보

클러스터 크기 vCPU 0EA, Memory 0GB

주의사항

- Subnet IP 생성 수를 초과할 수 없습니다.
- VM 생성 수를 초과할 수 없습니다.

< 이전 다음 >

- 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/2022-edu50/ncp-iam-202305.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 NCLOUD_ACCESS_KEY=<사용자의 Access key>

# export NCLOUD_SECRET_KEY=<사용자의 Secret key>

# export NCLOUD_API_GW=https://nccloud.apigw.ntruss.com
```

사용자 환경 홈 디렉터리의 .nccloud 폴더에 configure 파일

```
# mkdir .nccloud

# vi ~/.nccloud/configure

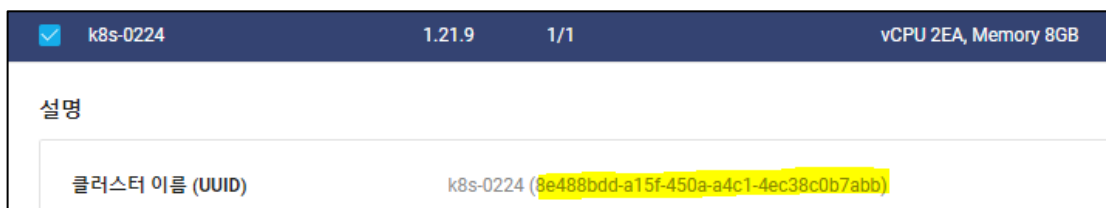
[DEFAULT]

nccloud_access_key_id = ACCESSKEY

nccloud_secret_access_key = SECRETKEY

nccloud_api_url = https://nccloud.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



```
# 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 --kubeconfig kubeconfig.yml
NAME          STATUS    AGE
default       Active    86m
kube-node-lease Active    86m
kube-public   Active    86m
kube-system   Active    86m
```

- 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
```

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

## 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-csqsk                 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: <private-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
```

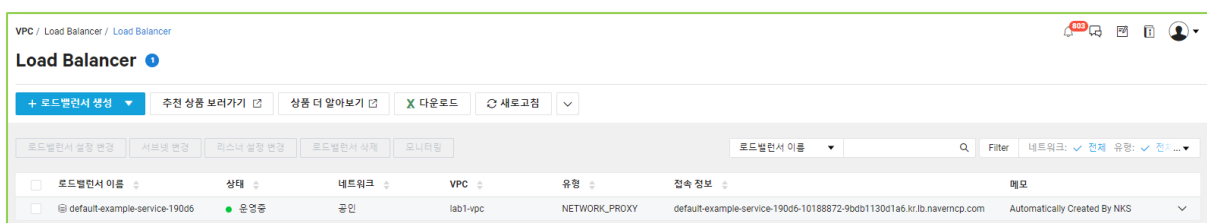
```
# kubectl apply -f create_service.yaml

# 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

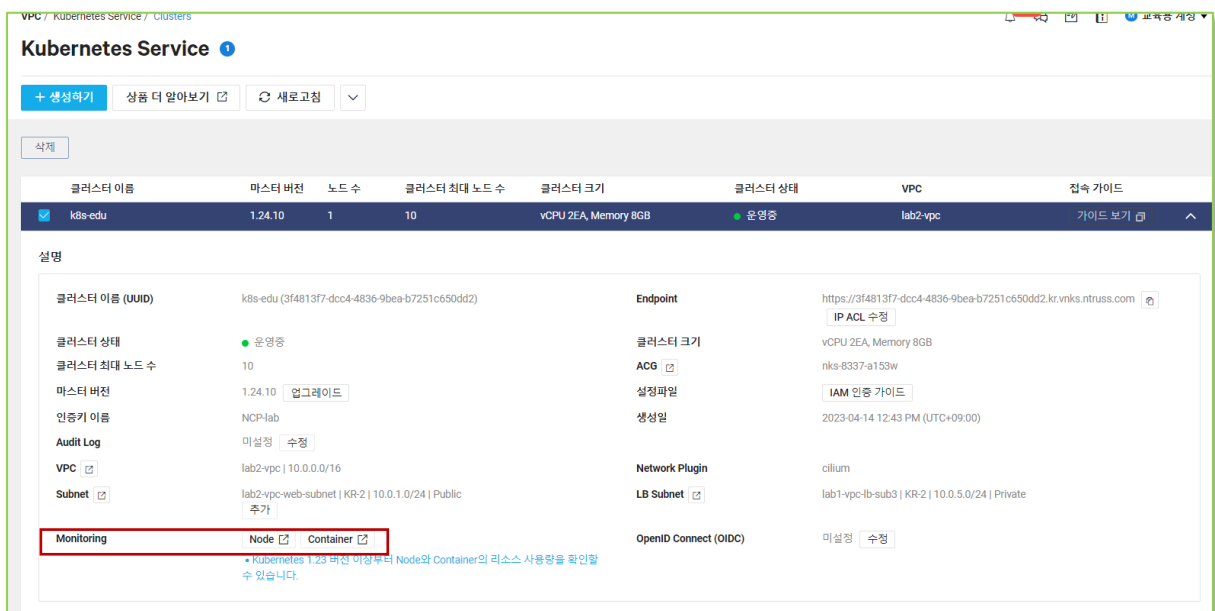


### 2) 브라우저에서 LB 접속정보(URL) 로 접속



## 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

```
[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 > 인증키관리에  
서 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-backup-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

- 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

클러스터 설정  
(필수 입력 사항입니다)

클러스터 이름

클러스터 버전

클러스터 Type

- ☐ Core Hadoop : HDFS(3.1.1), YARN(3.1.1), HIVE(3.1.0), Hue(4.3.0), Ranger(1.2.0)
- ☐ Presto : Presto(0.240), HDFS(3.1.1), YARN(3.1.1), Hue(4.3.0), Zeppelin Notebook(0.8.0), Ranger(1.2.0)
- ☐ HBase : HDFS(3.1.1), YARN(3.1.1), HIVE(3.1.0), HBase(2.0.2), Zookeeper(3.4.6), Ranger(1.2.0)
- ☒ Spark : HDFS(3.1.1), YARN(3.1.1), spark(2.3.2), HIVE(3.1.1), Hue(4.3.0), Zeppelin Notebook(0.8.0), Ranger(1.2.0)

커버로스 인증 구성 ☐ 보안 인증을 위해서 마스터 노드에 KDC(Kerberos Distribute Center)서버를 이중화하여 제공합니다.

VPC  [VPC 생성](#)

KR-2 리전의 VPC환경에서만 클러스터 생성 가능합니다.

클러스터 관리자 계정

클러스터 관리자 계정 비밀번호

ACG 설정 [Cloud Hadoop ACG는 자동 생성 됩니다.  
\(예: cloud-hadoop-acg\)  
hadoop 접근을 위한 ACG 설정은 사용자 가이드 > ACG 사용 가이드를 참고 하세요.](#)

< 이전 다음 >

- Object Storage에 미리 생성한 버킷을 지정
- 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	<a href="#">새로고침</a>
고가용성 지원	<input checked="" type="checkbox"/> 기본적으로 2대의 마스터 노드가 생성됩니다.	
엣지노드 Subnet	demo-subnet   KR-2   Public	<a href="#">Subnet 생성</a>
엣지노드 서버 타입	vCPU 4개, 메모리 8GB, 디스크 50GB	
엣지노드 개수	1 <a href="#">1개로 고정</a>	
마스터노드 Subnet	demo-subnet   KR-2   Public	<a href="#">Subnet 생성</a>
마스터노드 서버타입	vCPU 4개, 메모리 16GB, 디스크 50GB <a href="#">기본적으로 50G크기의 디스크(HDD)가 제공됩니다.</a>	
마스터노드 개수	2 <a href="#">고가용성 지원으로 2대의 마스터 노드가 생성되며 변경은 불가능합니다.</a>	
마스터노드 스토리지 타입	<input checked="" type="radio"/> SSD <input type="radio"/> HDD <a href="#">설치 이후에 스토리지 타입은 변경되지 않습니다.</a>	
마스터노드 스토리지 용량	100 GB, 최소 100GB, 최대 2000GB 까지 10GB단위 선택 가능합니다.	
작업자노드 Subnet	lab1-vc-db-sub1   KR-2   Private	<a href="#">Subnet 생성</a>
	<a href="#">작업자노드 Subnet 은 Private Subnet 만 지원가능합니다.</a>	
작업자노드 서버타입	vCPU 4개, 메모리 32GB, 디스크 50GB <a href="#">기본적으로 50G크기의 디스크(HDD)가 제공됩니다.</a>	
작업자노드 개수	2 <a href="#">작업자 노드는 최소 2개이상 되어야 합니다.(default 2)</a>	
작업자노드 스토리지 타입	<input checked="" type="radio"/> SSD <input type="radio"/> HDD <a href="#">설치 이후에 스토리지 타입은 변경되지 않습니다.</a>	
작업자노드 스토리지 용량	100 GB, 최소 100GB, 최대 2000GB 까지 10GB단위 선택 가능합니다.	
요금제	시간 요금제 <a href="#">요금 안내(Storage 비용은 별도 부과됩니다.)</a>	

### 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-49k70

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		
TCP	cloud-hadoop-acg-49k70(19854)	1-65535	(automatically created, don't delete it) cloud-hadoop-acg-lab-hadoop	
ICMP	211.249.71.0/24		(automatically created, don't delete it) cloud-hadoop-acg-lab-hadoop	
ICMP	211.249.70.0/24		(automatically created, don't delete it) cloud-hadoop-acg-lab-hadoop	
ICMP	211.249.69.128/25		(automatically created, don't delete it) cloud-hadoop-acg-lab-hadoop	
TCP	211.249.71.0/24	8080	(automatically created, don't delete it) cloud-hadoop-acg-lab-hadoop	
TCP	211.249.70.0/24	8080	(automatically created, don't delete it) cloud-hadoop-acg-lab-hadoop	

닫기 적용

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

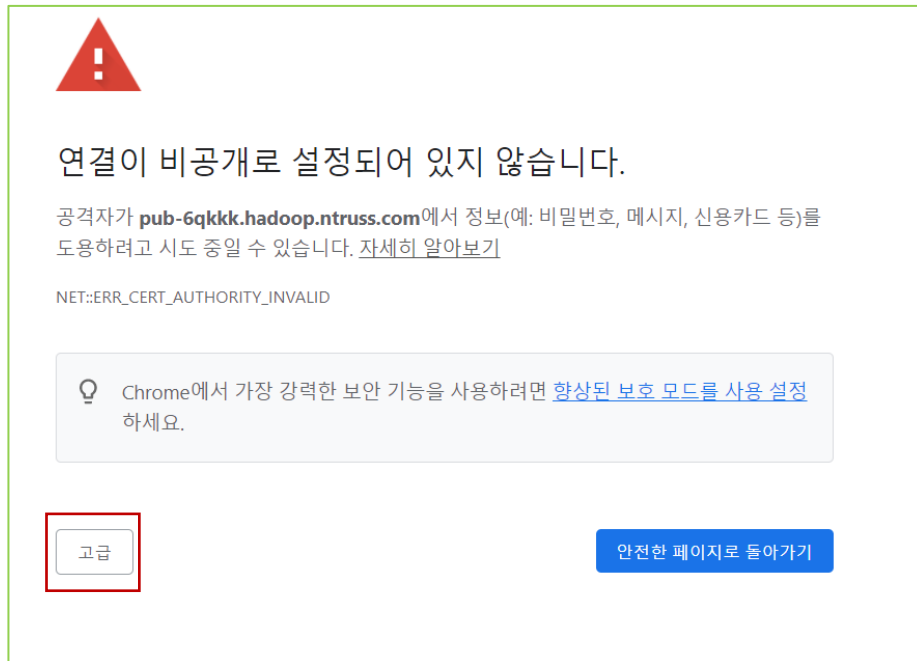
Application web UI 접속을 위한 사전 작업

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.

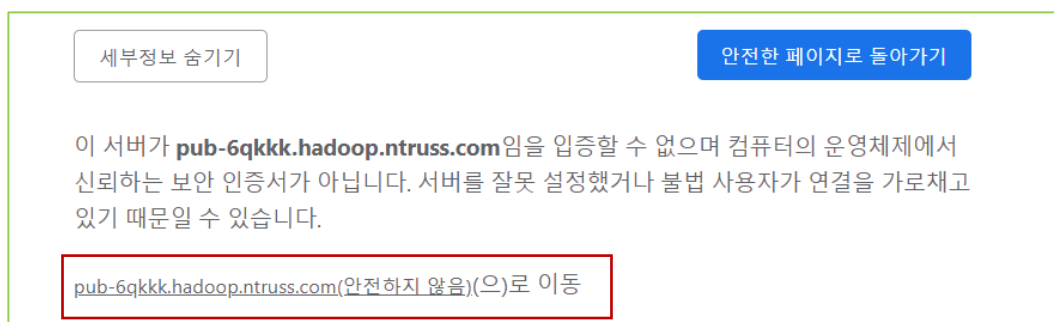
프로토콜	접근소스	허용포트	비고
TCP	도메인	8443	Ambari Web Console 접속용
TCP	도메인	8081	Hue Admin 접속용
TCP	도메인	9996	Zeppelin Notebook 접속용
TCP	도메인	6182	Ranger 접속용

☐ 다시 보지 않음 확인

- 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 : spark2 선택 후 하단의 create 버튼 클릭

- Bank.csv파일을 bank테이블에 로드하는 샘플코드를 복사해서 붙여넣습니다.

```
%spark2.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/2023-06/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) != "\age").map(
  s => Bank(s(0).toInt,
    s(1).replaceAll("\\", ""),
    s(2).replaceAll("\\", ""),
    s(3).replaceAll("\\", ""),
    s(5).replaceAll("\\", "").toInt
  )
).toDF()
bank.registerTempTable("bank")
```

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

Zeppelin Notebook Job

nce-lab

hadoop

```
%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`?
    ^
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.

```
%spark2.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
```

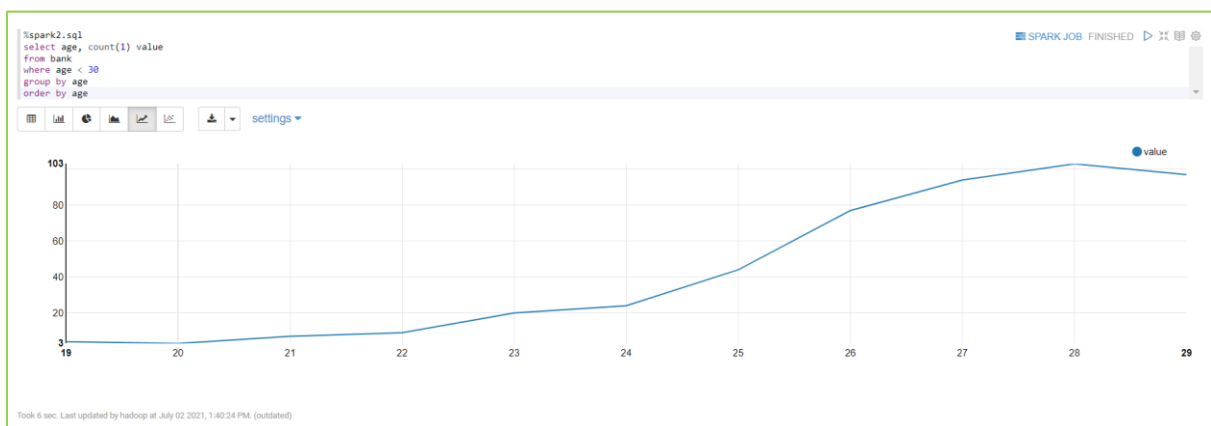
아래와 같이 표 형태로 결과를 확인해 볼 수 있습니다.

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

SPARK JOB FINISHED

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 및 WAS Security Checker 설정

- Web001 접속
- 다음 명령어 실행

```
[root@krweb001 ~]# wget
http://osscloud.com:10080/download/ncp_secuagent.tar.gz
--2020-08-10 17:50:32--
http://osscloud.com:10080/download/ncp_secuagent.tar.gz
Resolving osscloud.com (osscloud.com)... 10.101.86.17
Connecting to osscloud.com (osscloud.com)[10.101.86.17]:10080... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2543027 (2.4M) [application/x-gzip]
Saving to: ?cp_secuagent.tar.gz?

100%[=====] 2,543,027  --.-K/s  in 0.02s

2020-08-10 17:50:32 (104 MB/s) - ?cp_secuagent.tar.gz?saved [2543027/2543027]

[root@krweb001 ~]# tar xvfz ncp_secuagent.tar.gz
ncp_secuagent
[root@krweb001 ~]# ./ncp_secuagent
[Project : linux] => Success
[root@krweb001 ~]# ./ncp_secuagent -p apache
[Project : apache] => Success
[root@krweb001 ~]#
```

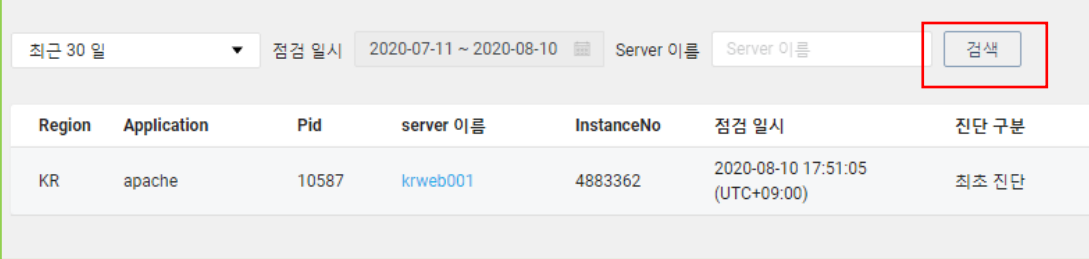
### 점검 내용 확인

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

최근 30 일	점검 일시	2020-07-11 ~ 2020-08-10	Server 이름	Server 이름	검색
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

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



Region	Application	Pid	server 이름	InstanceNo	점검 일시	진단 구분
KR	apache	10587	krweb001	4883362	2020-08-10 17:51:05 (UTC+09:00)	최초 진단

- 우측의 리포트 클릭하여 내용 확인
- 리포트의 내용을 기반으로 보안 강화 작업 수행

## 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 : (생성된 lb 서브넷 활용)

**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="ncs-cm"/>
Network ●	<input type="radio"/> Private IP <input checked="" type="radio"/> Public IP
부하 처리 성능 ② ●	<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 checked="" type="checkbox"/> KR-2 <input type="text" value="lab1-vpc-lb-subnet   10.0.254.0/24"/> <input type="checkbox"/> KR-1 <input type="text" value="- 선택하세요 -"/> <input type="button" value="서브넷 생성"/>

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.naverncp.com 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 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

### Free SSL 인증서 발급

- <https://letsencrypt.org/ko/getting-started/> 를 통해 FreeSSL 인증서 발급
- Let's Encrypt 설치

```
# yum install epel-release

# yum install python2-certbot-apache

# certbot --standalone -d edu0x.ncloudedu.com certonly

Saving debug log to /var/log/letsencrypt/letsencrypt.log
Plugins selected: Authenticator manual, Installer None
Enter email address (used for urgent renewal and security notices)
(Enter 'c' to cancel): 개인 메일 주소
Starting new HTTPS connection (1): acme-v02.api.letsencrypt.org
```

Please read the Terms of Service at <https://letsencrypt.org/documents/LE-SA-v1.2-November-15-2017.pdf>. You must agree in order to register with the ACME server. Do you agree?

-----  
 (Y)es/(N)o: y

-----  
 Would you be willing, once your first certificate is successfully issued, to share your email address with the Electronic Frontier Foundation, a founding partner of the Let's Encrypt project and the non-profit organization that develops Certbot? We'd like to send you email about our work encrypting the web, EFF news, campaigns, and ways to support digital freedom.

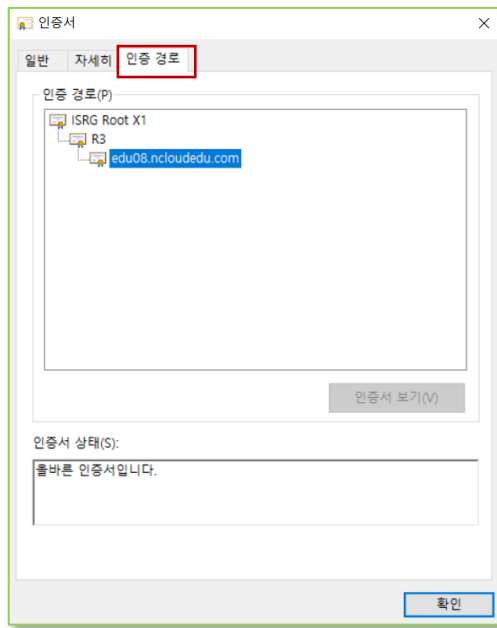
-----  
 (Y)es/(N)o: y

```
# cd /etc/letsencrypt/live/edu0x.ncloudedu.com
# wget http://apps.identrust.com/roots/dstrootcax3.p7c
# openssl pkcs7 -inform der -in dstrootcax3.p7c -out dstrootcax3.pem -print_certs
# cp fullchain.pem fullchain_RootCA.pem
# cat dstrootcax3.pem >> fullchain_RootCA.pem
```

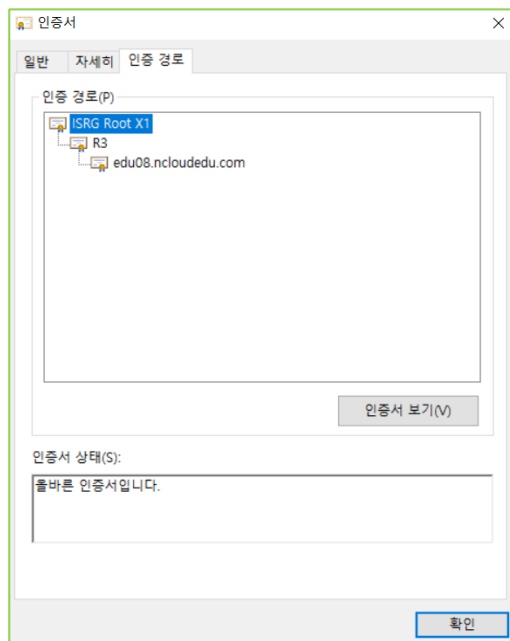
## CM에 인증서 추가

- /etc/letsencrypt/live/edu0x.ncloudedu.com 디렉토리에 있는 파일들을 PC로 다운로드
- `scp root@[서버의공인ip]:/etc/letsencrypt/live/edu0x.ncloudedu.com/cert.pem C:\WUsers\WUSER\Desktop\key`
- cert.pem 파일을 cert.crt 로 파일명 변경
- 인증서 더블 클릭 후 상단의 '인증 경로' 탭 선택

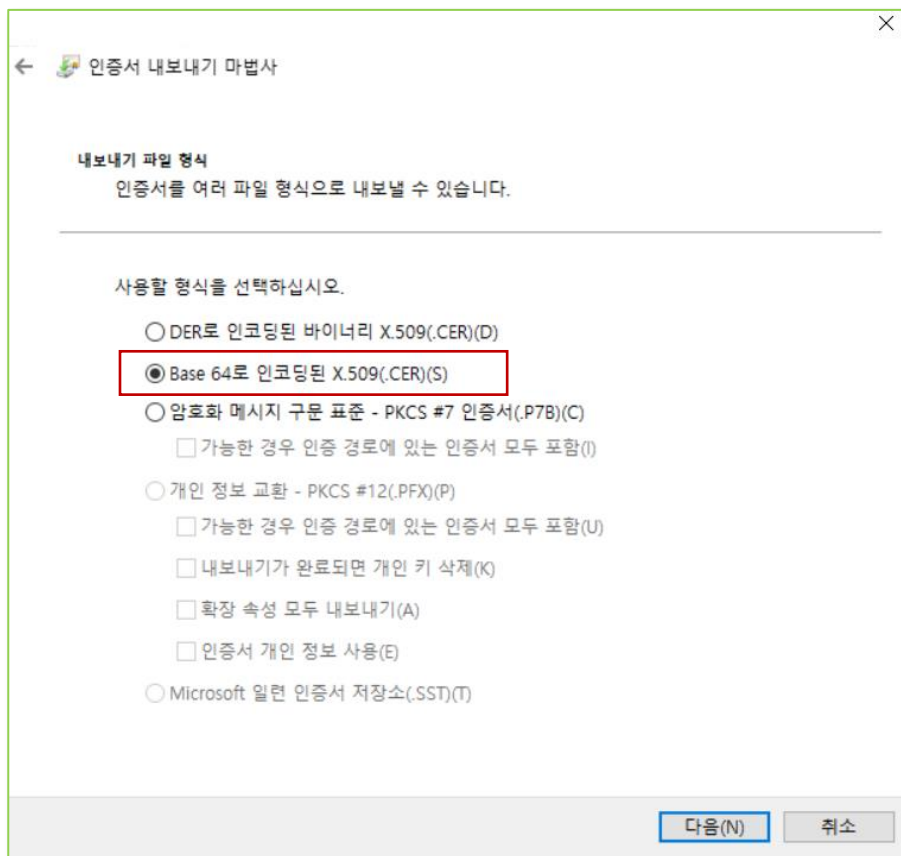




- 상위 인증기관 선택 후 인증서 보기 선택



- 인증서 화면에서 자세히 선택 후 파일에 복사 선택
- Base 64로 인코딩 선택 후 임시 파일에 저장  
(ex : E:\WCA.cer)



- 중간 인증서에 대해서도 동일 작업 수행  
(ex : E:\W\intermediate.cer)
- 만들어진 파일 2개를 .pem파일로 변환 후, web001 서버에 업로드
  - 변환 시 OpenSSL 사용 : x509 -inform PEM -in C:\Users\USER\Desktop\key\CA.cer -out C:\Users\USER\Desktop\key\CA.pem
  - scp E:\W\CA.pem root@[서버의 공인 IP] :/etc/letsencrypt/live/edu0X.ncloudedu.com/
  - scp E:\W\intermediate.pem root@[서버의 공인 IP] :/etc/letsencrypt/live/edu0X.ncloudedu.com/
- Intermediate.pem 인증서안에 CA.pem 인증서의 내용을 추가
  - cat CA.pem >> intermediate.pem
- Security > Certificate Manager > +외부 인증서 등록 선택
- Certificate 이름 : nce
- Private Key 입력

- /etc/letsencrypt/live/edu0x.ncloudedu.com/privkey1.pem 파일을 열어서 내용 전체 복사

```
[root@krweb001 nce.ncloudedu.com]# cat privkey1.pem
-----BEGIN PRIVATE KEY-----
MIIEvgIBADANBgkqhkiG9w0BAQEFAASCBCGwggSkAgEAAoIBAQDZSjY2rz9/JbgR
nuSjTxkM/+uPb4DtCUebo2Wi/yQhn/MWRICsSZtUfTNXXXIbe2vXERS5CqB49kTE
SjUTtct8EiFfuAiXpkUwvfGsbELvp2wnf/z1NsYOOjh83NMHHVcef01VneNln4xn
0/YqqYuR+0lbOPPpOP1AFwoxDkVFK3d6JVfEnxZrZ2UmZ4PQwQ8+U+ITKXF+d/D
NgYQQCBGC33p67Q7rqoWjmna+VaDXxU0aTziebx7mvEevfIKq7pyplbKDRCr8WAB
Yd4Jm3Y2Nght8YCGJFYzuZaTTb8lgo0n5khCUol9a6DnyCMczN3W6c8e8ApdX9e
iUrMa/F1AgMBAAECggEBAINdV1tGJcgzwl1+I1GX71gpnHrx+WuKmmMorLiMXvW
nHrZ9cnDElJJAvInF3/X5QF3xSR9pTicAkT9bAtHfAa9W0noE+HC1s+59dK8/Naw
TDWxLff9rk1Jq37pbfc+KL46TkG1m4P7WUloJgnyHtwBRCqSeCLCKhnhXTU4NII
cN5l88NFyTE6BoA5i/O8xukr9P1hIRLpjRFpzOGRv08aBMT+gAP+QUIZHn9oYQ9X
yIXM/jRJHfeZVL976Mr3wLvCBq+ysPipdr3LZEICdEB7ecc3its4H4M+CdLXrzUu
/4oNfz38PeEL0tnqQtm9ak5VRiTXBa5hOwf8qXQ4dyECgYEA9clMqGO65cv8eKoG
WsyC6/HfHQWlk4Ri3XTn5PXN+Vny9HmAA0nlbg+9yejUq6zdg56w7HGVOseSATDV
xhbH25E8DheKZQ+dp/IYpl7tCve08v4Q2iyH10r4nAVWWC91muK0utK9Bs+ueOCc
grPKLFxA/IgtIfE4mOMn+K/7d/OCgYEA4IHDpxtOk88PsW9mDNtTWlvAR+3YVjqb
k8R+o2Q0AJok7i0KIDi7YcLXHiFN8e65SobrorqycYhckarznX2VUSnLcQrkGeoQ
zCpU83yxQ9gTho4684s1MEncwvTe+LebraPgt0U/wu5sJHvhXmC0vDWDjiAoHmUP
WJeXghYe7NkCgYEAtJNJO6nRxUTH33DjLFB2m4xfJD5i0leB7zwVpxSkWF8qbUza
rQ/HLn1oLXBe1yYwtKOhToWAYuf+r9tGI7vW58zDN4M6DVe0t4/ZZQUQyG8GxUO9
89yljJorHs2ZCz8LA1kt+NgdQmTQxUQYGLqofYDlyeLubbKAp2q0kjQQ560CgYBU
262Nsbpx3at5AQSnAYTNEtDbRXptw3wiQomxgKNmzSv+2l1DBSOYl0AaiJsrUMC
g7ZEjNYtpaB36e5wRc8/4HjsiNXgNZhYxSKXLVDWDGW91QSbnr2xvnAtFV2pSbkw
A3tVnHx83aLkxyJlneAGldYe60W/p8rqP+TKHPs5eQKBgEkgI86L3St7gxinIVjC
3XRkJb968W/MN3XQzv0CT/WJC52qEYNi1bWAQloYgf7IDhvRaDazZx4Yqaj/99ZP
VZBCICmi1/F1OiQthW8IZCkvanaXgS+xf4qfN2OelcpC8lw6MS/aYSGeD8x7PINI
nJRIbq6ZOaLuZdZfz7DDyZi0
-----END PRIVATE KEY-----
[root@krweb001 nce.ncloudedu.com]#
```

- 해당 내용을 복사해서 입력
- Certificate Body 입력
  - /etc/letsencrypt/live/edu0x.ncloudedu.com/cert.pem 파일을 열어서 내용 전체 복사

```
[root@krweb001 nce.ncloudedu.com]# cat cert.pem
-----BEGIN CERTIFICATE-----
MIIFWjCCBEKgAwIBAgISA71+rhcDtmM2iCICK1DEWfBYMA0GCSqGSIb3DQEBCwUA
MEoxCzAJBgNVBAYTAIVTMRYwFAYDVQQKEw1MZXXQncyBFbmNyeXB0MSMwIQYDVQ
```

```

QD
ExpMZXQncyBFbmNyeXB0IEF1dGhvcml0eSBYMTZAEw0yMDA4MTAwODQ3NDVaFw0y
MDExMDgwODQ3NDVaMBwxGjAYBgNVBAMTEW5jZS5uY2xvdWRIZHUuY29tMIIlBjAN
BgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA2Uo2Nq8/fyW4EZ7ko08ZDP/rj2+A
7QIHm6Nlov8kiZ/zFkZQrEmbVH0zV11yG3tr1xEUuQqgePZExEo1E7XLfBlhX7gl
l6ZFML3xrGxC76dsJ3/89TbGDjo4fNzTBx1XHn9NVZ3jSJ+MZ9P2KqmLkftJW9Dz
8Tj9QBcKMq5FRSt3eiVXxJ8Wa2dlJmeD0MEPPIPiEyl3xfnfWzYGEEAgRgt96eu0
O66qFo5p2vIWg18VNGk84nm8e5rxHr35Squ6cqSGyg0Qq/FgAWHeCZt2NjYIbfGA
hiRWGbmWk02/CIMaNJ+ZlQIKJfWug58gjHMzd1unPHvAKXV/XoIkZGvxdQIDAQAB
o4ICZjCCAmIwDgYDVR0PAQH/BAQDAgWgMB0GA1UdJQQWMBQGCCsGAQUFBwMB
Bggr
BgEFBQcDAjAMBgNVHRMBAf8EAjAAMB0GA1UdDgQWBBrkf0NgNfuehYkJPksrcU
FFzbBDAfBgNVHSMEGDAWgBSoSmpjBH3duubRObemRWXv86jsoTBvBggrBgEFBQcB
AQRjMGEwLgYIKwYBBQUHMAAGGIh0dHA6Ly9vY3NwLmludC14My5sZXRzZW5jcnlw
dC5vcmcwLwYIKwYBBQUHMAKGI2h0dHA6Ly9jZXJ0LmludC14My5sZXRzZW5jcnlw
dC5vcmcvMBwGA1UdEQQVMBOCEW5jZS5uY2xvdWRIZHUuY29tMEwGA1UdIARFME
Mw
CAYGZ4EMAQIBMDcGCysGAQQBggt8TAQEBMCgwJgYIKwYBBQUHAhAgEWGmh0dHA6Ly
9j
cHMubGV0c2VuY3J5cHQub3JnMIIIBAYKKwYBBAHWeQIEAgSB9QSB8gDwAHYAsh4F
zluizYogTodm+Su5iiUgZ2va+nDnsklTle+LkF4AAAFz18TKIAAABAMARzBFAiAI
2oCiYt001WchzwTqHeSwqx6LDkqa4tjBKP6xc9Lc+wIhAJ67RpMgEzBG0SiO5hKi
nAR5xoAiPS9aU6hhCoDxbTiWAHYAb1N2rDHwMRnYmQCkURX/dxUcEdkCwQApBo2y
CJo32RMAAAAFz18TK4wAABAMARzBFAiEApx7h7fJy/xOVQPCrP9Jr0rM5t8q5SCAR
zicPbCnSlx8CICyAJ1fv5ajH0w5cAgQhMMRKM69gyJlrBsc7VwLPxQZpMA0GCSqG
SIb3DQEBCwUAA4IBAQAhvno18gkwXpWjxFOdmhGo+2Si/JW0EoEiHvIn2DeXFmdS
5NEB1hK0sbeDnBBJfT88rFwCrSRN/bANsPNmKe0dYNqMvyT2Taebo7cHQBlnA8n
Dweb6zEPqQjOd6evUQp3DTMm1wHjocEJznRy3HhQaOCzVhyMPDpazuplwd02xG3
lrMCwSe8yKxvze5cLglEoAlICyRb6UDDGjIwT2aLNHylzkUWXdziJRLziicbcMK
MXH6ofWYLjK4dJEiRR29CYLSMiMoQ3jclGV+ia0JrsaO126z9B7FVjygfY8ESS+T
IsaJXAq9HvwkN+x9nV4UBxiVnK14sh7GS0sOaO1
-----END CERTIFICATE-----
[root@krweb001 nce.ncloudedu.com]#

```

- 해당 내용을 복사해서 입력
- Certification Chain 에 내용 입력
- Intermediate.pem 파일 내용 입력

```

[root@krweb001 nce.ncloudedu.com]# cat intermediate.pem
-----BEGIN CERTIFICATE-----
MIIEKjCCA3qgAwIBAgIQCGFBQgAAAVOFc2oLheynCDANBgkqhkiG9w0BAQsFADA/
MSQwLgYDVQQKEExtEaWdpdGFsIFNpZ25hdHVyZSBUcnVzdCBDby4xFzAVBgNVBAMT
DkRTVCBSb290IENBIFgzMB4XDTE2MDMxNzE2NDA0Ni0xODIxMDMxNzE2NDA0Nlow

```

SjeELMAkGA1UEBhMCVVMxZjAUBG9NVBAoTDUxldCdzIEVUyY3J5cHQxZAhBgNVBAMT  
 GkxldCdzIEVUyY3J5cHQgQXV0aG9yaXR5IFgzMlIIBjANBgkqhkiG9w0BAQEFAAOOC  
 AQ8AMlIBGKCAQEAAnNMM8Frllke3cl03g7NoYzDq1zUmGSXhvb418XCSL7e4S0EF  
 q6meNQhY7LEqXGiHC6PjdeTm86dicbp5gWaf15Gan/PQeGdxyGkOlZHP/uaZ6WA8  
 SMx+yk13EiSdRxta67nsHjcAHJyse6cF6s5K671B5TaYucv9bTyWaN8jKkKQDIZO  
 Z8h/pZq4UmEUEz9l6YKH9v6Dlb2honzHT+Xhq+w3Brvaw2VFn3EK6BlspkENnWA  
 a6xK8xuQsXgvopZPKiAlKQTGdMDQMc2PMTiVFrqoM7hD8bEfwzB/onkxEz0tNvj  
 /Plzark5McWvxI0NHWWQWM6r6hCm21AvA2H3DkwIDAQABo4IBfTCCAXkwEgYDVR0T  
 AQH/BAgwBgEB/wIBADAObgNVHQ8BAf8EBAMCAAYwfwYIKwYBBQUHAQEEdzBxMDI  
 G  
 CCsGAQUFBzABhiZodHRwOi8vaXNyZy50cnVzdGlkLm9jc3AuaWRlbnRydXN0LmNv  
 bTA7BggrBgEFBQcwAoYvaHR0cDovL2FwcHMuaWRlbnRydXN0LmNvbS9yb290cy9k  
 c3Ryb290Y2F4My5wN2MwHwYDVR0jBBgwFoAUxKexpHsscfrb4UuQdf/EFWCFiRAw  
 VAYDVR0gBE0wSzAIBgZngQwBAgEwPwYLKwYBBAGC3xMBAQEwMDAuBggrBgEFBQc  
 C  
 ARYiaHR0cDovL2Nwcy5yb290LXgxLmxldHNIbmNyeXB0Lm9yZzA8BgNVHR8ENTAz  
 MDGgLG6AthitodHRwOi8vY3JsLmlkZW50cnVzdC5jb20vRFNlUk9PVENBWdNDUkwu  
 Y3JsMB0GA1UdDgQWBBSoSmpjBH3duubRObemRWXv86js0TANBgkqhkiG9w0BAQsF  
 AAOCAQEA3TPXEfNjWDjdGBX7CVW+dla5cEilaUcne8lkCJLxWh9KEik3JHRRHGJo  
 uM2VcGfl96S8TihRzZvoroed6ti6WqEBmtzw3Wodatg+VyOeph4EYpr/1wXKtx8/  
 wAplvJSwtmVi4MFU5aMqrSDE6ea73Mj2tcMyo5jMd6jmeWUHK8so/joWUoHOUGwu  
 X4Po1QYz+3dszkDqMp4fklxBwXR5W10KXzPMTZ+sOPaveyxindmjkW8lGy+QsRIG  
 PfZ+G6Z6h7mjem0Y+iWlkYcV4PIWL1iwBi8saCbGS5jN2p8M+X+Q7UNKEkRob3N6  
 KOqkqm57TH2H3eDJAkSnh6/DNFu0Qg==  
 -----END CERTIFICATE-----cer  
 -----BEGIN CERTIFICATE-----  
 MIIDSjCCAjKgAwIBAgIQRK+wgNajJ7qJMDmGLvhAazANBgkqhkiG9w0BAQUFADA/  
 MSQwlgYDVQQKEExtEaWdpdGFsIFNpZ25hdHVyZSBUCnVzdCBDbY4xZzAVBgNVBAMT  
 DkRTVCBSb290IENBIFgzMB4XDTAwMDkzMdIxMTIxOVVoXDTIxMDkzMDE0MDExNVow  
 PzEkMCIGA1UEChMbRGlnaXRhbCBTaWduYXR1cmUgVHJ1c3QgQ28uMRcwFQYDVQQD  
 D  
 Ew5EU1QgUm9vdCBDQSBYmzCCASlwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggE  
 B  
 AN+v6ZdQCINXtMxiZfaQguzH0yxrMMpb7NnDfcdAwRgUi+DoM3ZJKuM/IUmTrE4O  
 rz5ly2Xu/NMhD2XSKtkyj4zl93ewEnu1lcCJo6m67XMuegwGMoOifooUMM0RoOEg  
 OLI5CjH9UL2AZd+3UWODyOKIYepLYYHsUmu5ouJLGiiFSKOeDNoJjj4XLh7dIN9b  
 xiqKqy69cK3FCxolkHRYxXtqqzTWMIln/5WgTe1QLyNau7Fqckh49ZLOMxt+/yUFw  
 7BZy1SbsOFU5Q9D8/RhcQPGX69Wam40dutolucbY38EVAjqr2m7xPi71XAicPNad  
 aeQQmxkqtilX4+U9m5/wAl0CAwEAaAaNCMEAwDwYDVR0TAQH/BAUwAwEB/zAOBgN  
 V  
 HQ8BAf8EBAMCAQYwHQYDVR0OBBYEFMSnsaR7LHH62+FLkHX/xBVghYkQMA0GCSq  
 G  
 Slb3DQEBBQUAA4IBAQCjGiybFwBcqR7uKGY3Or+Dxz9LwwwmgISbd49IZRNI+DT69  
 ikugdB/OEIKcdBodfpga3csTS7MgROSR6cz8faXbauX+5v3gTt23ADq1cEmv8uXr  
 AvHRAosZy5Q6XkjEGB5YGV8eAlrwdPGxrancWYaLbumR9YbK+rImM6pZW87ipxZz

```
R8srzJmwN0jP41ZL9c8PDHlyh8bwRLtTcm1D9SZlmlJnt1ir/md2cXjbDaJWFBM5
JDGFoqgCWjBH4d1QB7wCCZAA62RjYJsWvljJEubSfZGL+T0yjWW06XyxV3bqxbYo
Ob8VZRzI9neWagqNdwwYkQsEjgfbKbYK7p2CNTUQ
-----END CERTIFICATE-----
[root@krweb001 nce.ncloudedu.com]#
```

#### 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
  - Target Group : nce-cm-lab
- 웹 브라우저로 https 로 접근

## Lab 9

### 프로젝트 생성

- AI-Application > Simple & Easy Notification Service > Project 선택 후 +프로젝트 생성하기 선택
- 서비스 Type : SMS
- 이름 : nce
- 생성하기 선택

프로젝트 생성

**Project 설정**  
(필수 입력 사항입니다.)

서비스 Type ☐ PUSH ☒ SMS ☐ Biz Message

이름

설명

0/128자

• Simple & Easy Notification Service 요금은 SMS/LMS/MMS의 경우 수신 건수, PUSH/Alim Talk의 경우 발송 건수를 합산해 부과합니다.

### SENS 발신번호 등록

- AI-Application > Simple & Easy Notification Service > SMS > Calling Number 선택
- 상단 탭의 발신번호 등록 선택
- 핸드폰 인증 > 본인인증 선택
- 인증 진행

×

### 본인 휴대전화 인증

☒ 아래 약관에 모두 동의합니다.

☒ 인증시 개인정보 이용 [보기](#)
☒ 인증시 고유식별정보 처리 [보기](#)

☒ 통신사 이용약관 [보기](#)
☒ 인증사 이용약관 [보기](#)

☒ 개인정보 수집 [보기](#)

이름

내국인 ▼

남자

여자

생일

년(4자)

월 ▼

일

SKT ▼

휴대전화번호

인증

인증번호

인증비용은 네이버 클라우드 플랫폼에서 부담합니다.

**확인**






## 메시지 전송

- AI-Application > Simple & Easy Notification Service > SMS > Message 선택
- 프로젝트명 : nce
- 프로젝트 명 우측의 'SMS'클릭
- 상단의 '발송하기' 클릭
  - Type : SMS
  - 발송내용 : 일반용
  - 국가코드 : 대한민국
  - 발신번호 : 본인 전화번호 선택
  - 수신번호 : 본인 전화번호 입력 후, 단건추가 클릭
  - 내용에 인사말 입력
  - 발송 설정 : 즉시 발송

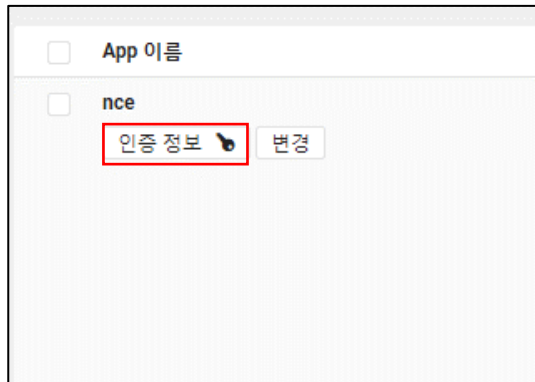


## 단축 URL 테스트

- 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
- Papago Image Translation

 <input type="checkbox"/> CLOVA	<input checked="" type="checkbox"/> CLOVA Speech Recognition (CSR) <input type="checkbox"/> CLOVA Face Recognition (CFR) <input checked="" type="checkbox"/> CLOVA Voice - Premium <small>CLOVA Voice - Premium은 기본료 월 90,000원 상품이며, 사용량에 따라 추가 요금이 발생합니다.</small> <input type="checkbox"/> CLOVA Sentiment <input type="checkbox"/> CLOVA Summary	<a href="#">서비스 설명/요금안내</a> <a href="#">서비스 설명/요금안내</a> <a href="#">서비스 설명/요금안내</a> <a href="#">서비스 설명/요금안내</a> <a href="#">서비스 설명/요금안내</a>	<a href="#">개발 가이드</a> <a href="#">개발 가이드</a> <a href="#">개발 가이드</a> <a href="#">개발 가이드</a> <a href="#">개발 가이드</a>
 <input checked="" type="checkbox"/> Papago Translation	<input checked="" type="checkbox"/> Papago Image Translation(Text) <input checked="" type="checkbox"/> Papago Image Translation(Image) <input type="checkbox"/> Papago Text Translation <input type="checkbox"/> Papago Website Translation <input type="checkbox"/> Papago Doc Translation <input type="checkbox"/> Papago Language Detection	<a href="#">서비스 설명/요금안내</a> <a href="#">서비스 설명/요금안내</a> <a href="#">서비스 설명/요금안내</a> <a href="#">서비스 설명/요금안내</a> <a href="#">서비스 설명/요금안내</a>	<a href="#">개발 가이드</a> <a href="#">개발 가이드</a> <div>           Papago Translation 은 신규 콘솔에서 이용할 수 있습니다.  <a href="#">Papago Translation 콘솔 바로가기</a> </div>
 <input type="checkbox"/> Machine Learning	<input type="checkbox"/> Ncloud Pose Estimation <input type="checkbox"/> Object Detection	<a href="#">서비스 설명/요금안내</a> <a href="#">서비스 설명/요금안내</a>	<a href="#">개발 가이드</a> <a href="#">개발 가이드</a>
 <input type="checkbox"/> Maps	<input type="checkbox"/> Web Dynamic Map <input type="checkbox"/> Mobile Dynamic Map <input type="checkbox"/> Static Map <input type="checkbox"/> Directions 5 <input type="checkbox"/> Directions 15 <input type="checkbox"/> Geocoding <input type="checkbox"/> Reverse Geocoding	<a href="#">서비스 설명/요금안내</a> <a href="#">서비스 설명/요금안내</a> <a href="#">서비스 설명/요금안내</a> <a href="#">서비스 설명/요금안내</a> <a href="#">서비스 설명/요금안내</a> <a href="#">서비스 설명/요금안내</a> <a href="#">서비스 설명/요금안내</a>	<a href="#">개발 가이드</a> <a href="#">개발 가이드</a> <a href="#">개발 가이드</a> <a href="#">개발 가이드</a> <a href="#">개발 가이드</a> <a href="#">개발 가이드</a> <a href="#">개발 가이드</a>
 <input checked="" type="checkbox"/> NAVER	<input checked="" type="checkbox"/> CAPTCHA (Image) <input checked="" type="checkbox"/> CAPTCHA (Audio) <input checked="" type="checkbox"/> Korean Name Romanizer <input checked="" type="checkbox"/> nShortURL <input checked="" type="checkbox"/> Search Trend	<a href="#">서비스 설명/요금안내</a> <a href="#">서비스 설명/요금안내</a> <a href="#">서비스 설명/요금안내</a> <a href="#">서비스 설명/요금안내</a> <a href="#">서비스 설명/요금안내</a>	<a href="#">개발 가이드</a> <a href="#">개발 가이드</a> <a href="#">개발 가이드</a> <a href="#">개발 가이드</a> <a href="#">개발 가이드</a>

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



App 이름

nce

인증 정보 변경

- 다음 정보 기록



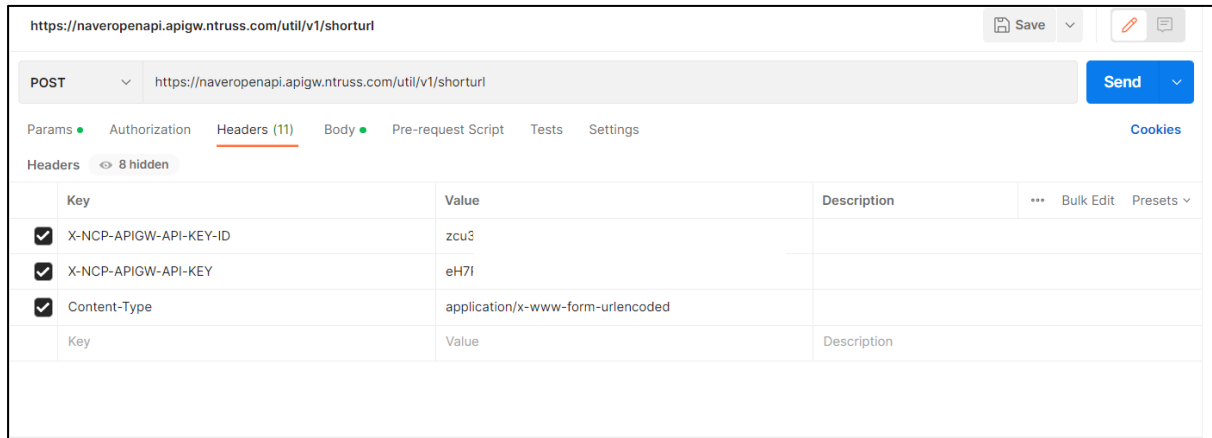
Application key

Application 이름 nce

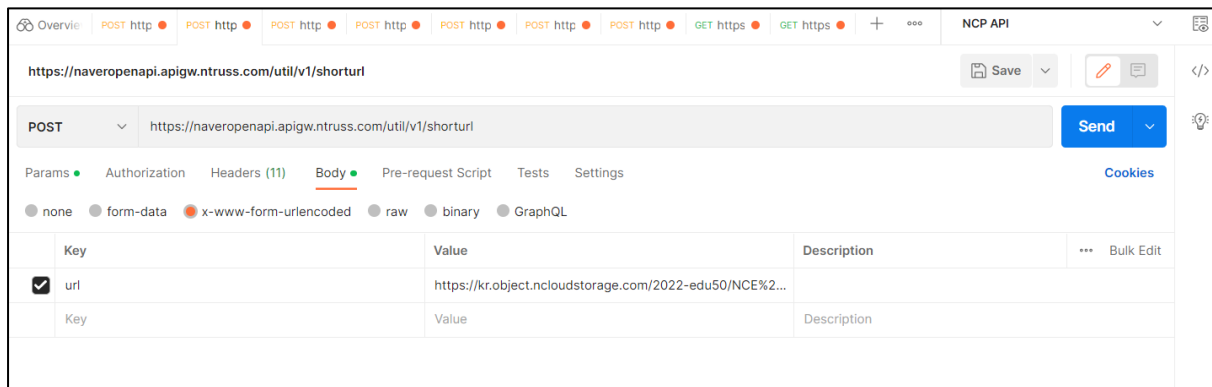
Client ID (X-NCP-APIGW-API-KEY-ID) hdc828m72j

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



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



- 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.

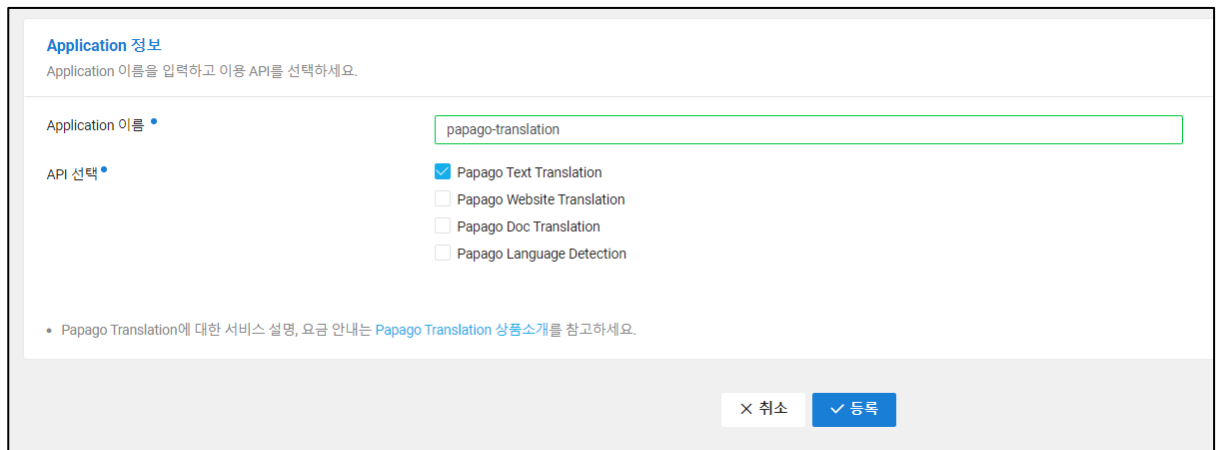
### Clova Speech Recognition API 호출

- 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

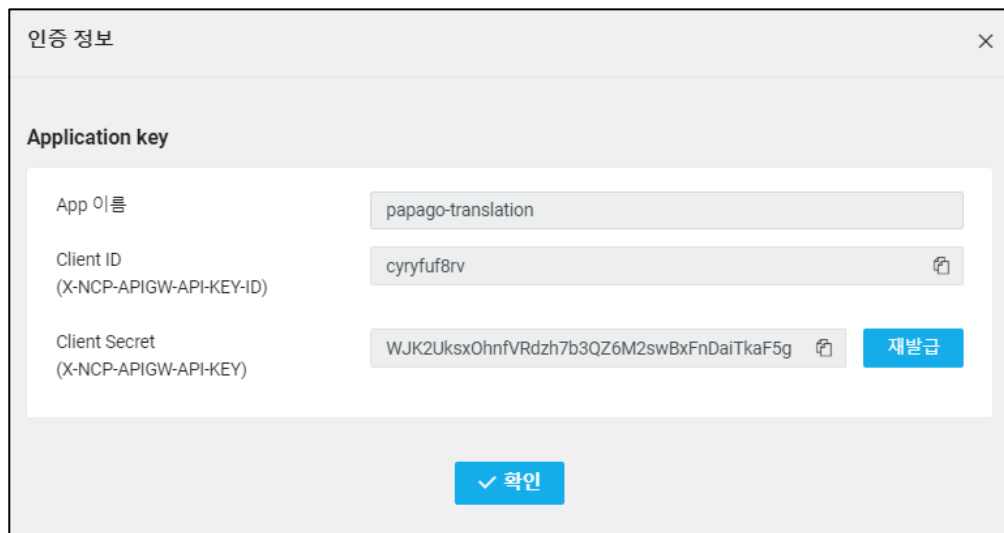
## Lab 11

### 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=안녕하세요" ₩
'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 방식으로 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

### 트리거 생성

- Cloud function > +트리거 생성 클릭
- 트리거 종류 : Basic
- 트리거 이름 : lab-hello
- 외부 연결 주소 생성
- Product : 새로만들기 > lab-hello
- API : 새로만들기 > lab-hello
- Stage : 새로만들기 > lab-hello
- 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 :  

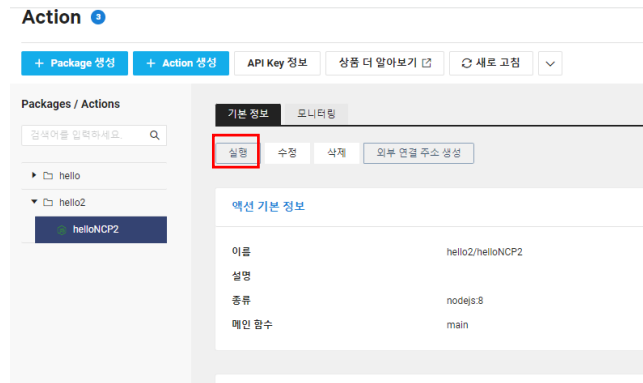
```
function main(params) {  
    return {payload: 'Hello, ' + params.name + ' from ' + params.place + '?'};  
}
```
- VPC :lab1-vpc 선택
- Subnet :lab1-vpc 내 private subnet 선택
- 옵션 설정 : Default선택
- 디폴트 파라미터:  

```
{"name":"stranger","place":"somewhere"}
```
- 생성 버튼 클릭

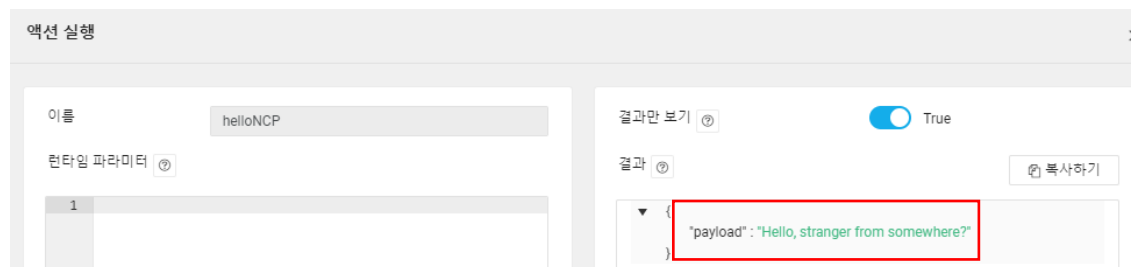
## 액션 단일 동작

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





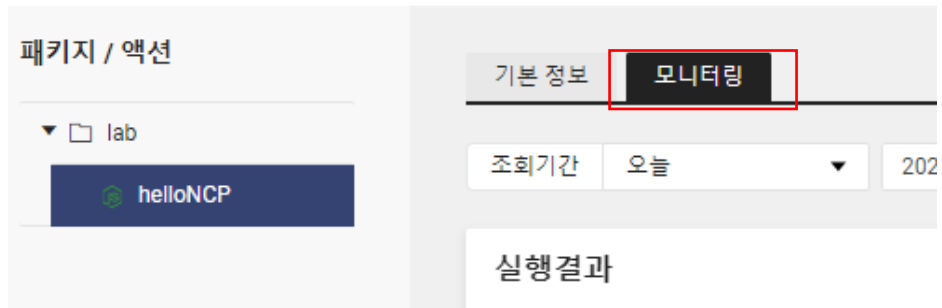
- 결과만 보기를 True로 변경
- 실행 버튼 클릭
- 아래와 같이 Hello, stranger from somewhere? 가 출력되는지 확인



### 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

이름	lab/helloNCP	ID	5c50d337c209460190d337c209660101
시작시간	2020-07-14 17:20:21 (UTC+09:00)	종료시간	2020-07-14 17:20:21 (UTC+09:00)
실행시간	1ms	메모리	256MB
상태	SUCCESS		

**결과**  
액션의 실행 결과가 보입니다.

```
{'payload':'Hello, NCP from GangNam?'}
```

### 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
- 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/cI3Uiwc3ga{"activationId":"7b988bbc668c4720988bbc668c672021"} [root@cidc-org ~]#
```

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

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

- Manamgement 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 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 권  
한 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
AAAAB3NzaC1vc2EAAAADAQABAAQCAOC08o27ez6EJQmVkcmy62+INDyGopIVWLUAWht/xgGLLTpmBDpYqmei
9Lq668p7i+ruwz7AYtkhUcqeKLI0ZkiDiZEoc4fTXotsj0TZCGMlonC18E4dyUh2ZnLMenio2F2L+4b0LJH/1zS/+IR6BsUx
isubWIPVbCbXWtWtiMailb4V74/nBvm1B+tEQIRri93AgPmulR82BOJBZoZOpI3QWeUYEZtixcKbQtQ6Zxon6xz+MVAf+
Z+a7xmASclBj8qw13pYLWqTaH31ge3WMulqD8fUIT7mcg3E4kvznrKGH07Nd5WcViygMfHQ0aSGwhOuFo9FCz72M
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 (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
```

```
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 확인
- 생성 완료 후, gitlab 빌드를 클릭 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
- 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 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

- 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
```

`$ vi /var/lib/tomcat/webapps/ROOT/WEB-INF/web.xml`를 통해 아래 정보가 `/description` 하단에 있는 지 확인

```
<servlet>
    <servlet-name>HelloNCP</servlet-name>
    <servlet-class>HelloServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>HelloNCP</servlet-name>
    <url-pattern>/hello</url-pattern>
</servlet-mapping>
```

- 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)
 

```
$ wget https://me2.do/xiXnMsHP -O /root/init.sh && chmod 755 init.sh && sed -i -e 's/₩r$//' init.sh && bash init.sh
```
- 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 톡캣 재실행
- 웹 브라우저에서 <web003 서버 IP>:8080/hello 로 접속하여 Hello, 이름! 출력 여부 확인