|  |
| --- |
| NAVER Cloud Platform  Lab Guide |

DevTrack part1

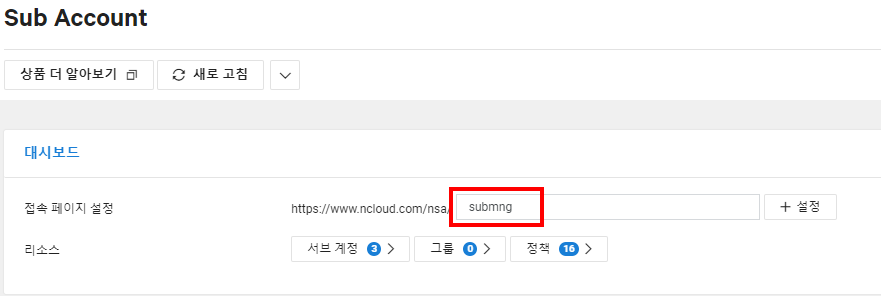
NAVER

**Lab 1 : 레파지토리 생성 (10분)**

1. **리파지토리 관리를 위한 Sub Account만들기**

Management > Sub Account > DashBoard 선택

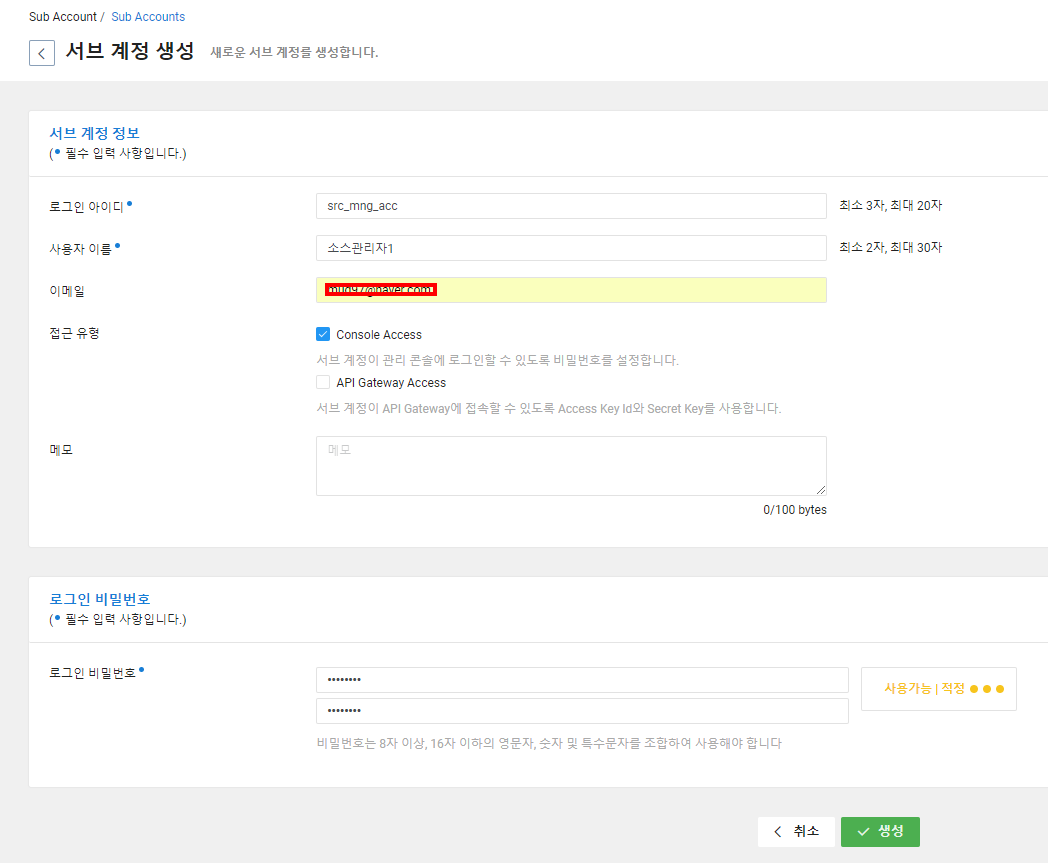
접속 페이지 설정 ( 향 후 sub account 계정은 대시보드에 생성한 url로 로그인 해야 합니다.)



1. **Sub Account 계정 만들기**

Management > Sub Account > Sub Accounts > 서브 계정 생성 선택

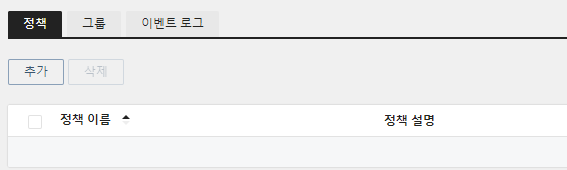
로그인 아이디와, 사용자 이름, 로그인 비밀번호 입력



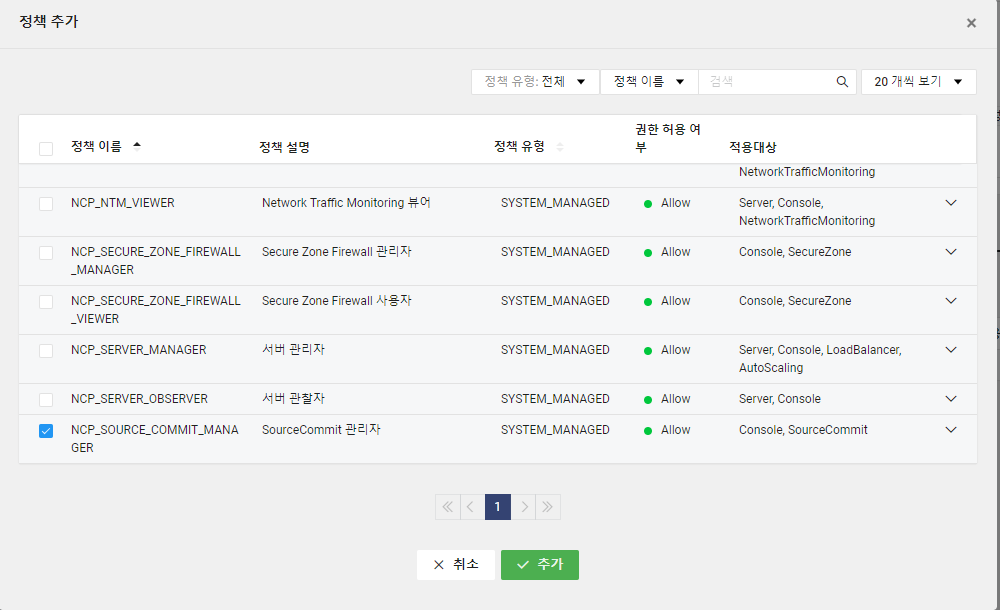
NCP\_SOURCE\_COMMIT\_MANGER 권한을 부여합니다.

서브 계정 상세 > 정책

정책 탭에서 추가 버튼을 누릅니다

.

NCP\_SOURCE\_COMMIT\_MANAGER 권한을 선택합니다.

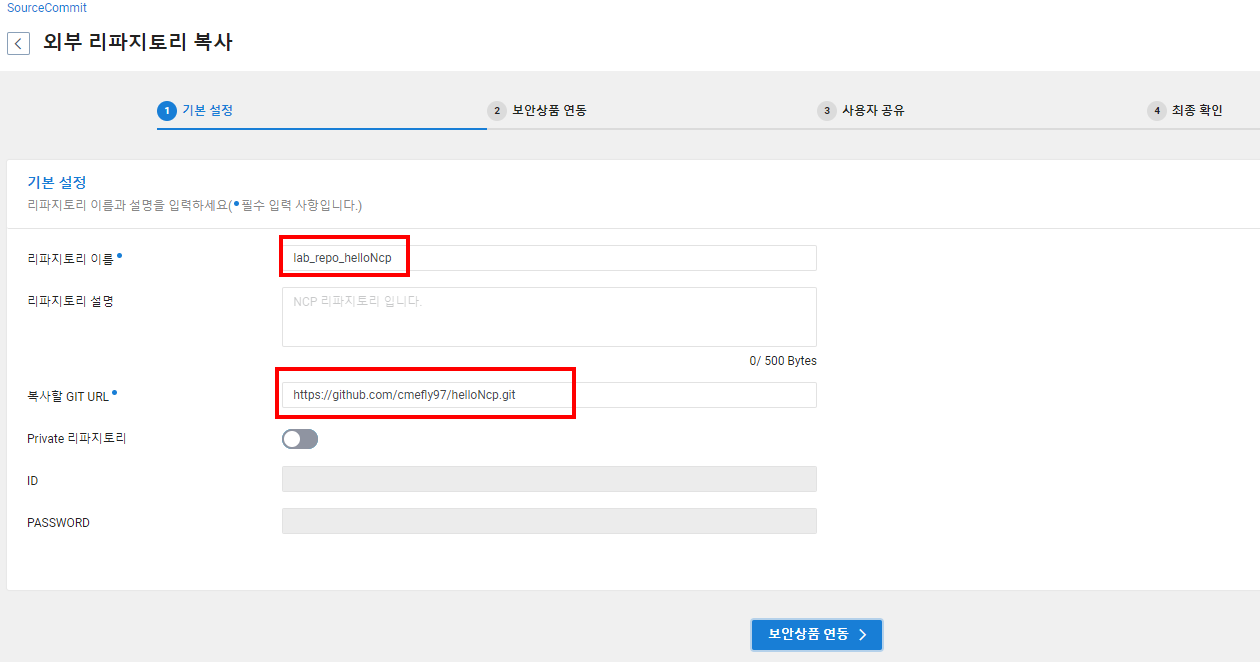


1. **리파지토리 생성**

SourceCommit> 외부 리파지토리 복사

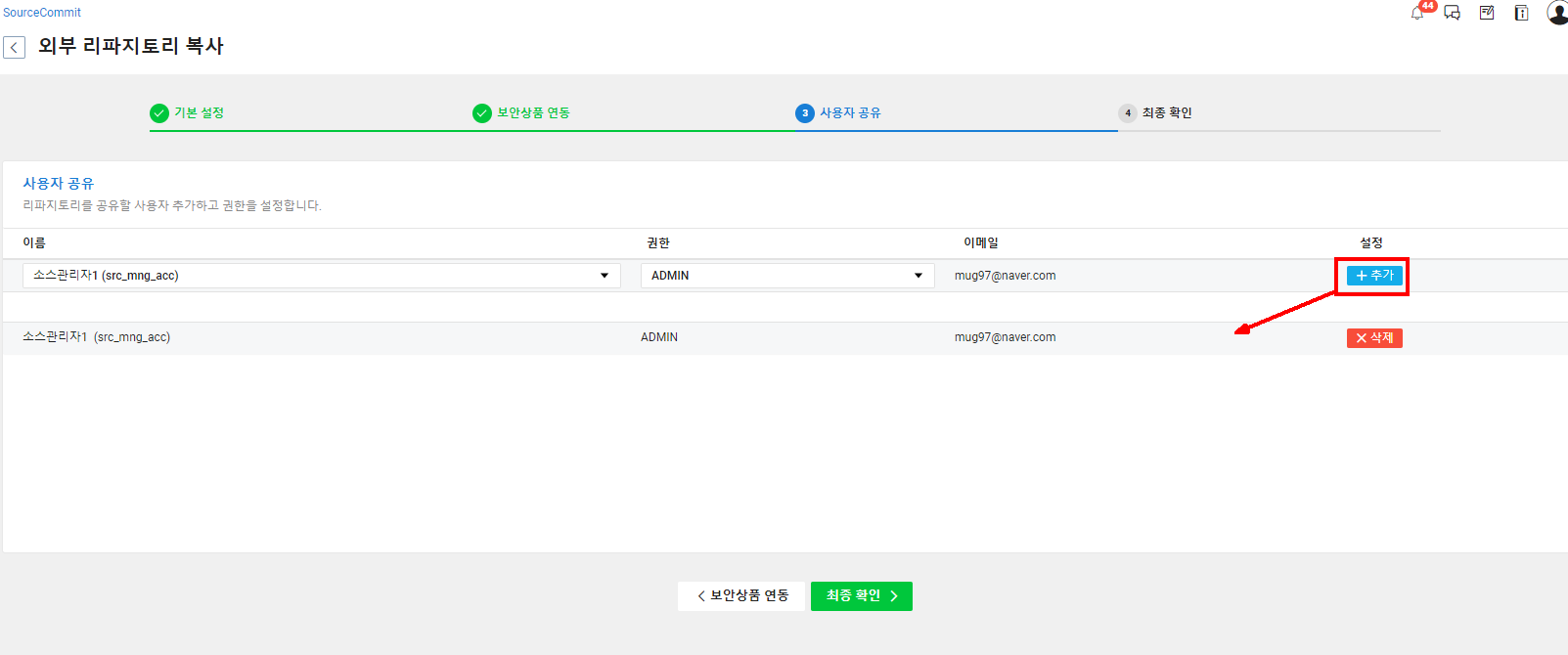
리파지토리 이름 : helloNcp 입력

복사할 GIT URL : https://github.com/NaverCloudPlatform/devtrack\_example\_part1.git

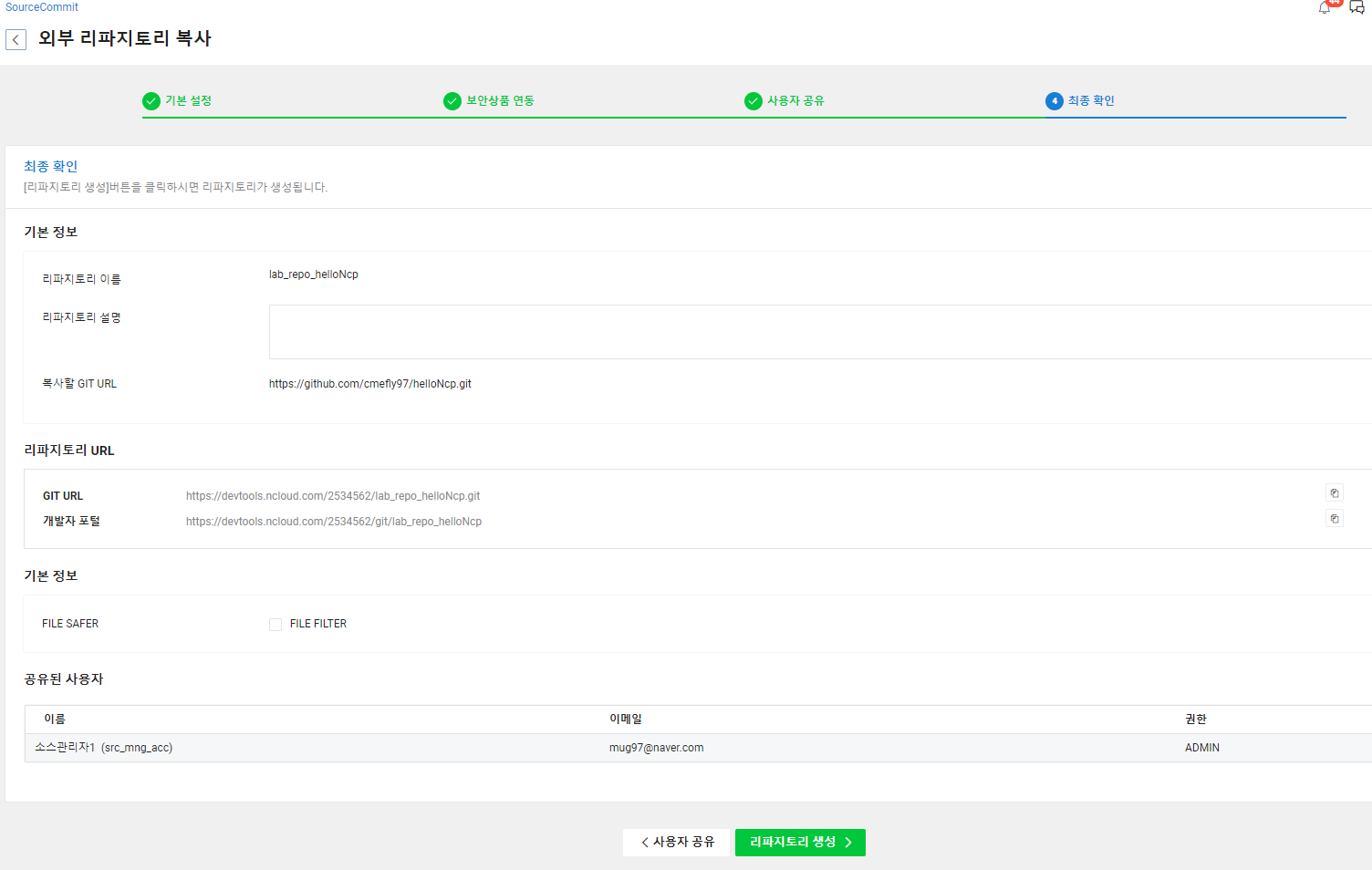


소스관리자에게 리파지토리 ADMIN 권한 추가

사용자 공유 > 소스관리자1 (arc\_mng\_acc) 선택 > ADMIN 권한 선택 > 추가 버튼 클릭합니다.



최종 정보 확인 후 리파지토리 생성 버튼 클릭합니다.



SourceCommit에 생성된 리포지토리 정보를 확인합니다.

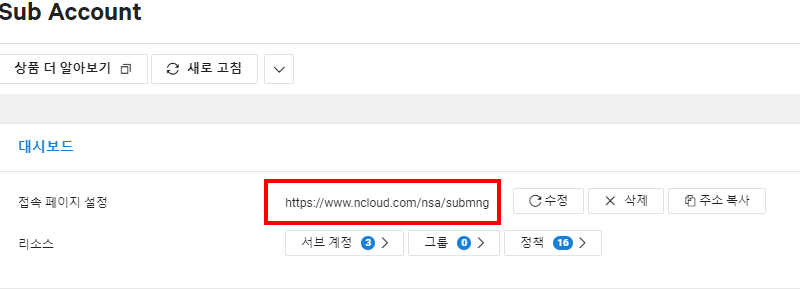
개발자 포털 정보와 Git URL 정보를 확인 합니다.



**Lab 2 : git client 로 소스코드 내려받기/올리기 (15분)**

1. **Sub Account로 콘솔 로그인하기**

먼저 Sub Account 접속 페이지 정보를 확인합니다.

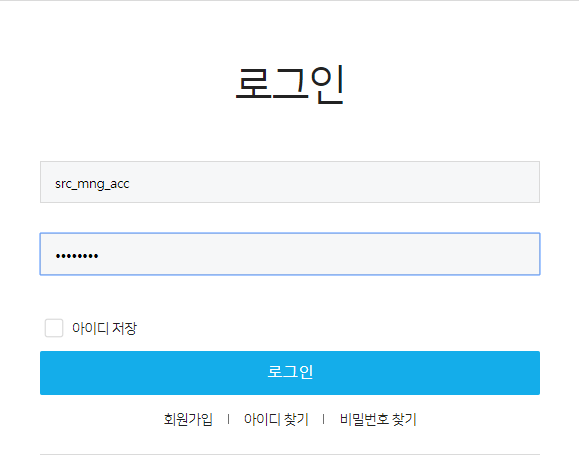


클라우드 콘솔에 로그인 되어 있는 관리자 계정에서 로그아웃하고

리파지토리 관리를 위해 생성한 Sub Account로 콘솔에 로그인합니다.

(Sub Account 대시보드에 등록한 접속 페이지로 로그인)

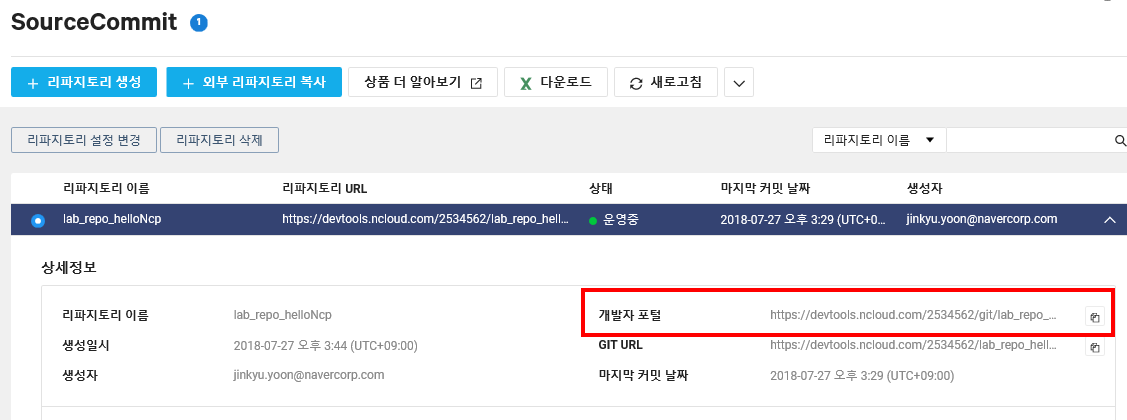
서브계정 아이디 : src\_mng\_acc



1. **개발자 포털 접속**

Sub Account 로콘솔 접속 후 리포지토리 상세보기를 클릭하여

개발자 포털 정보를 확인하고 링크를 클릭합니다.



클릭

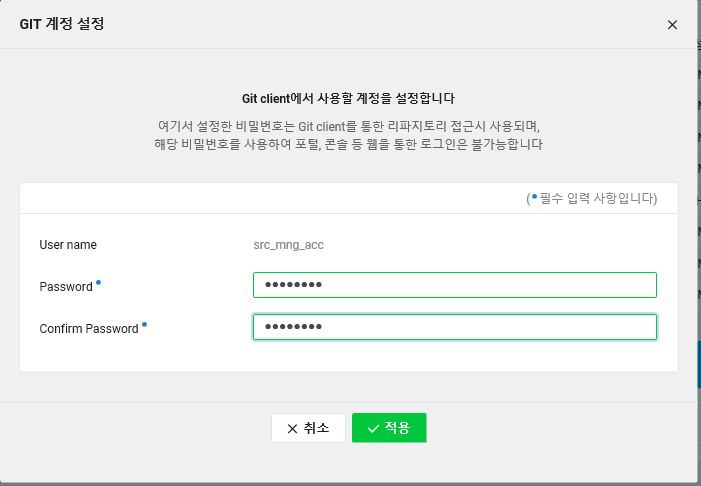
1. **Git 계정 설정**

개발자 포털의 **GIT 계정 설정** 버튼을 클릭하여 리파지토리 접근을 위한 Git 자격 증명을 발급받습니다

* Git 계정은 서브 계정만 설정할 수 있습니다.
* Git 접속 계정의 아이디는 리파지토리 사용자 계정(서브 계정)의 아이디와 동일합니다.



클릭

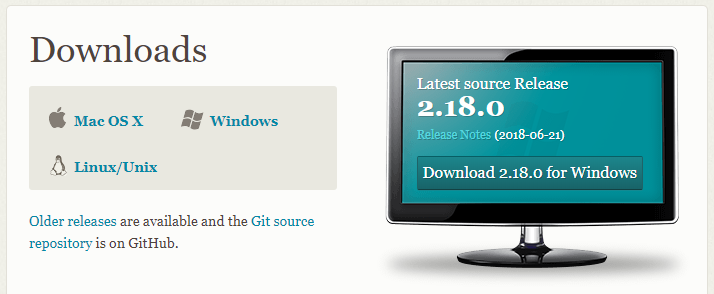


1. **리포지토리 CLONE**

Git client 가 설치되어 있지 않은 사용자 pc에는 git client 프로그램을 먼저 설치합니다.

<https://git-scm.com/downloads>

Git URL에 접속하여 git client 프로그램을 설치합니다.

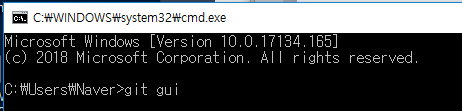


클릭

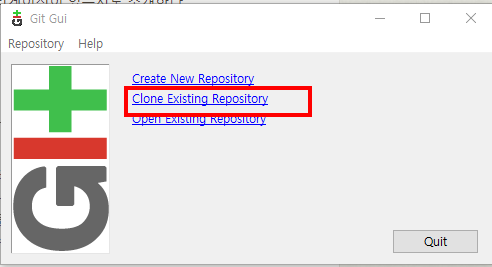
( 모두 Default 설정을 선택하고 설치를 완료합니다. )

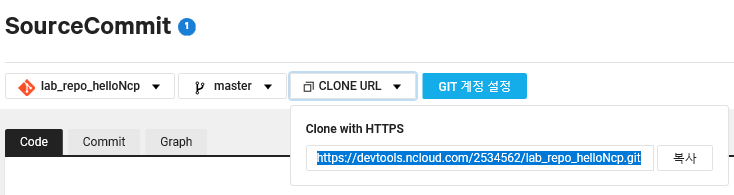
**Git GUI 실행**

커맨드 창을 열고, git gui 입력하여 git Gui를 실행시킵니다.



팝업된 Git GUI 창에서 Clone Existing Repository 를 클릭합니다.





Source Location : 리포지토리 URL을 입력합니다.

**Repository URL에**

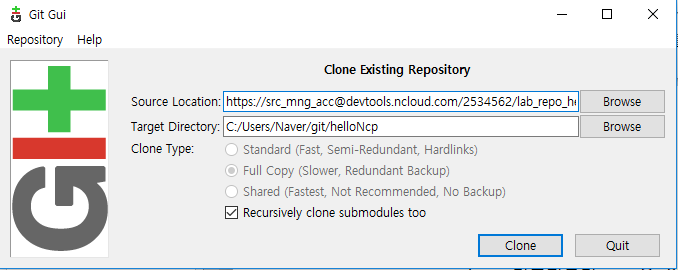
**username 추가**

단, Private repository 이기 때문에 username 정보를 추가해 주어야

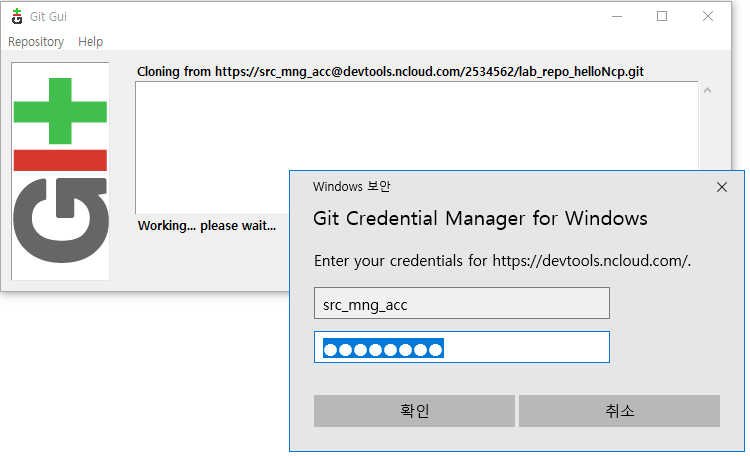
합니다.

https://**src\_mng\_acc@**devtools.ncloud.com/2534562/lab\_repo\_helloNcp.git

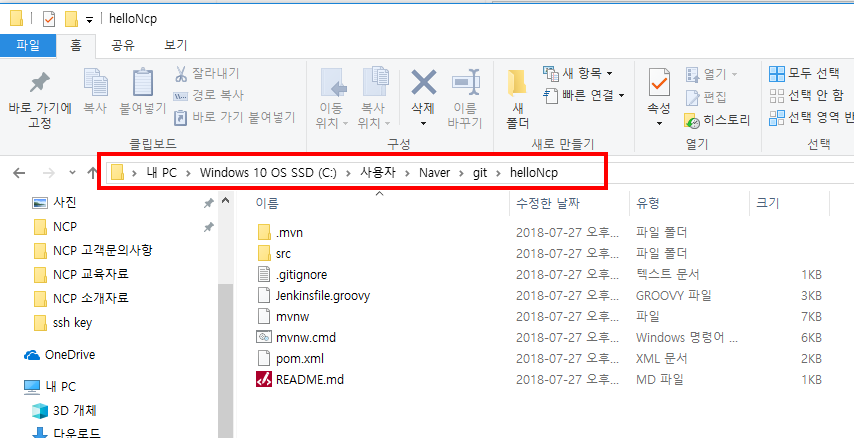
Target Directory : 로컬 PC의 git 리포지토리 경로를 입력합니다.



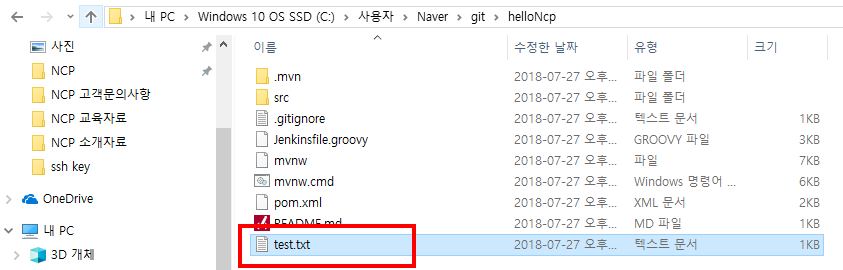
서브계정의 Git 패스워드를 입력합니다.



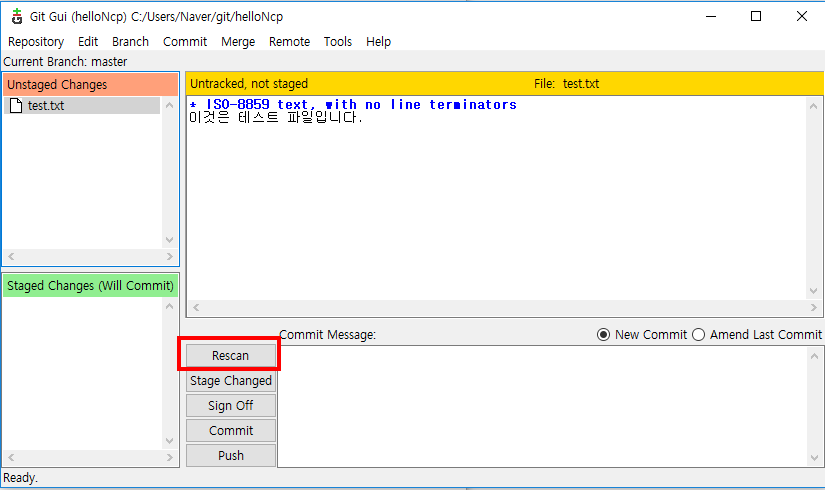
로컬 PC의 리포지토리에 소스가 Checkout 된 것을 확인하실 수 있습니다.



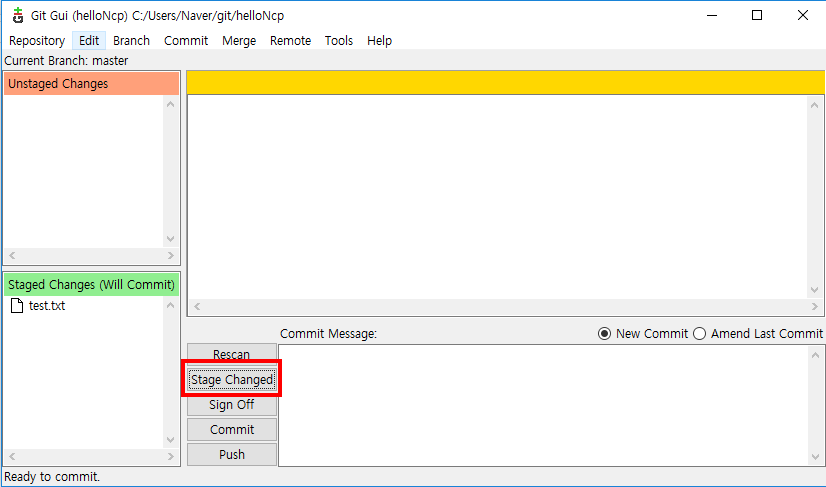
1. **리파지토리 PUSH**
2. Test.txt 파일 생성



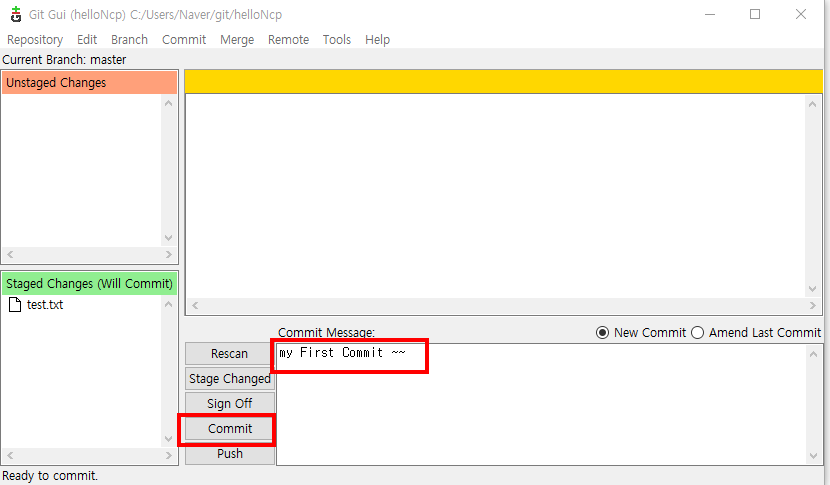
1. Git GUI 에서 Rescan 버튼 클릭



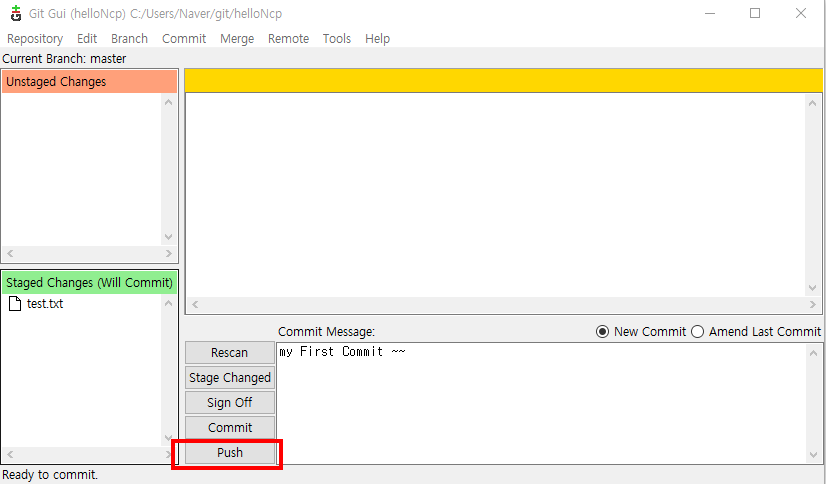
1. Stage Changed 클릭



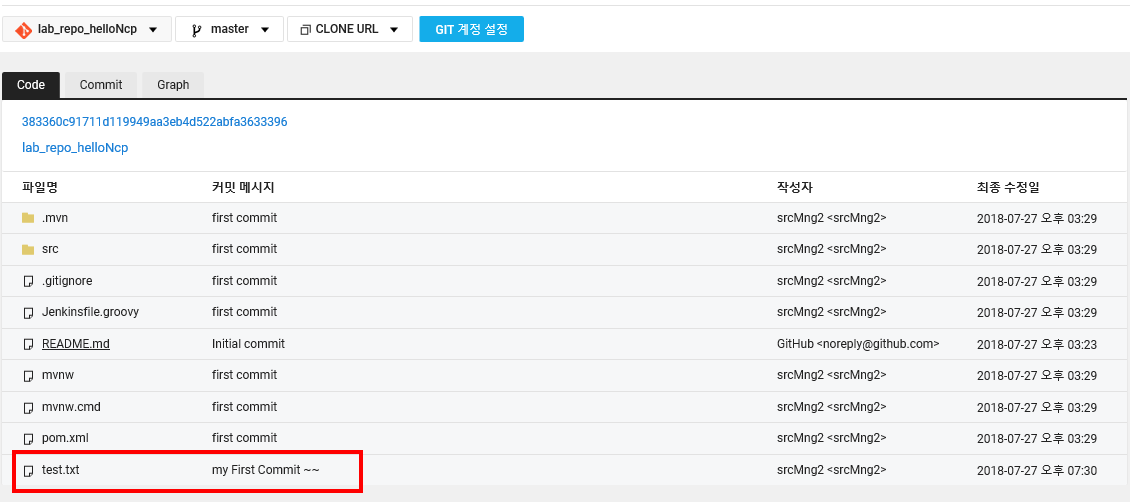
1. Commit 메시지 입력 후 Commit 버튼 클릭

*  

1. Push 하여 서버 Repository에 소스 업로드



1. 개발자 포털 > SourceCommit 에서 업로드된 파일 확인

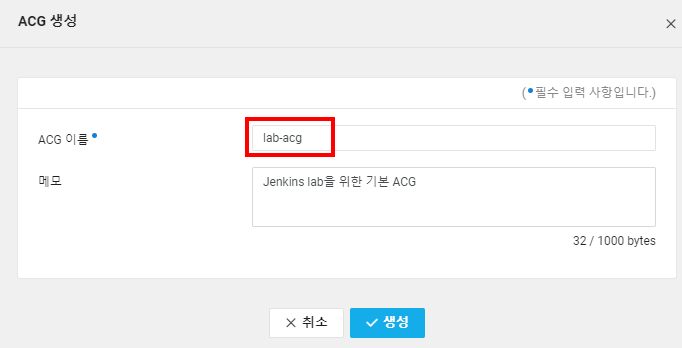


**Lab 3 : Jenkins 서버 설치(15분)**

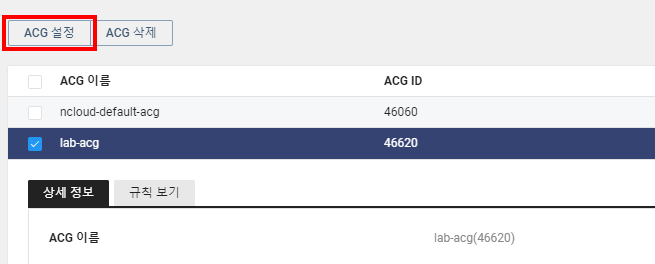
1. **웹서버 ACG 만들기**

All Product > Server > ACG 선택 > +ACG 생성 선택

ACG 이름에 lab-acg 라고 입력



‘lab-acg’ ACG를 선택 후 상단의 ACG 설정 버튼을 클릭합니다.



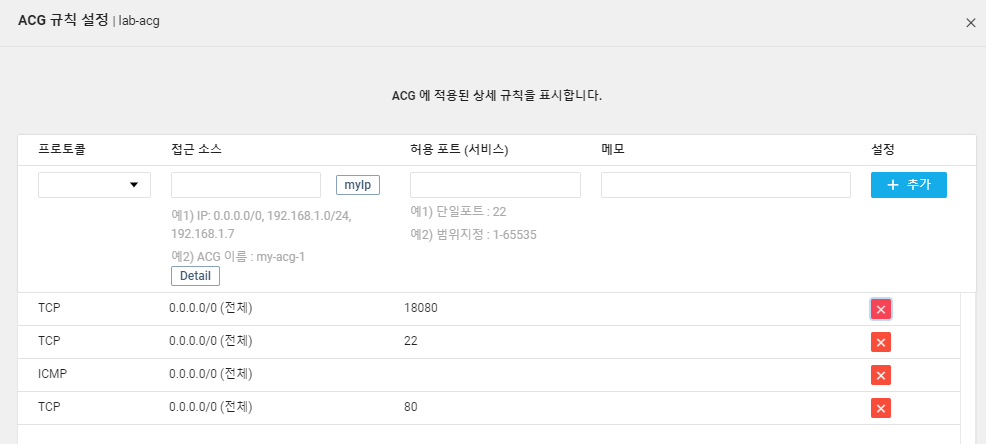
다음 4 개의 룰을 설정합니다.

프로토콜 : ICMP, 접근 소스 : 0.0.0.0/0

프로토콜 : TCP, 접근 소스 : 0.0.0.0/0 허용 포트 (서비스) : 22

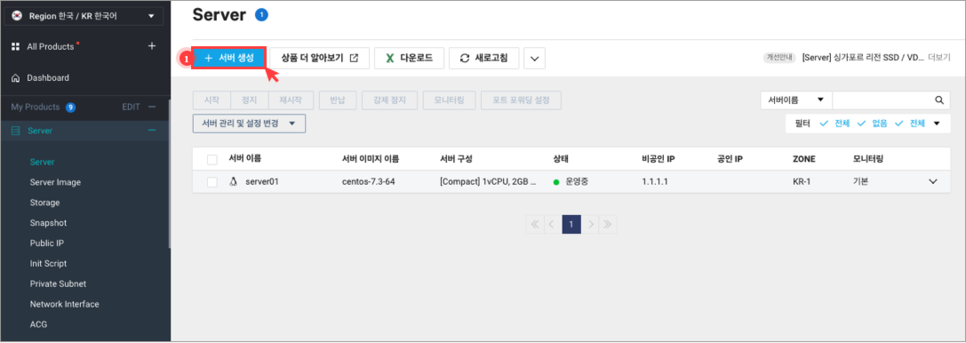
프로토콜 : TCP, 접근 소스 : 0.0.0.0/0 허용 포트 (서비스) : 80

프로토콜 : TCP, 접근 소스 : 0.0.0.0/0 허용 포트 (서비스) : 18080

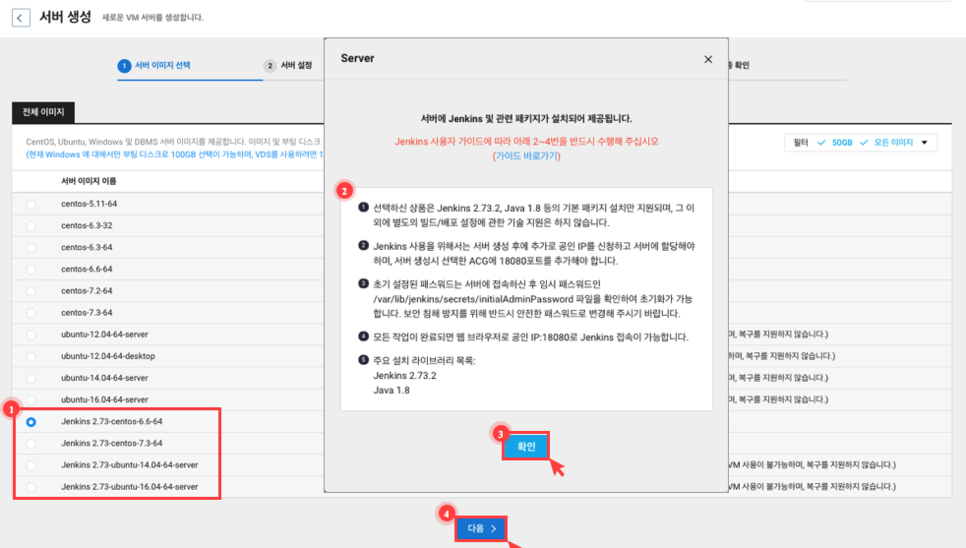


1. **Jenkins 서버 이미지 생성**

네이버 클라우드 플랫폼 콘솔 접속 후 Jenkins 서버 이미지를 생성합니다.



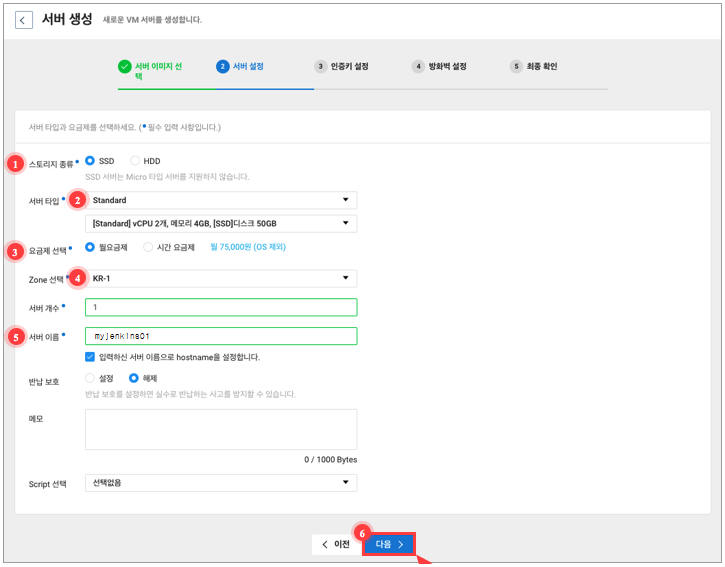
Jenkins 2.73-centos-7.3-64 의 Jenkins 서버 이미지를 선택하여 설치를 진행합니다.



Jenkins 서버의 스토리지, Zone, Hostname 등 서버 기본 정보를 입력합니다.

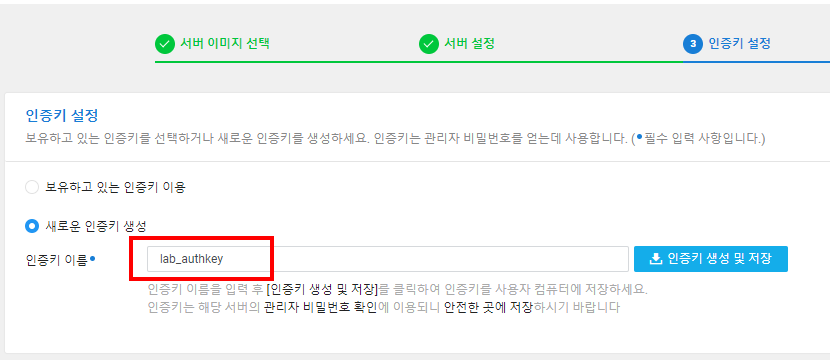
서버 타입은 vCPU 2개, 메모리 4GB, 디스크 50GB 를 선택해 주세요.

서버 개수는 1, 서버 이름은 myjenkins01 입니다.

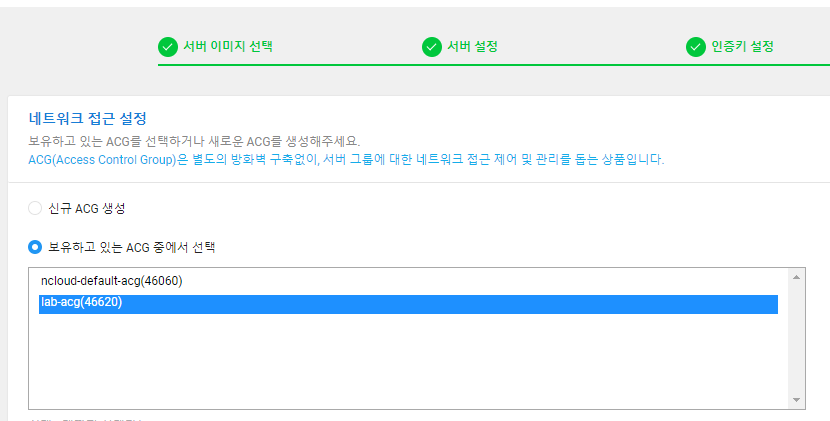


Jenkins 서버 인증키 설정은 추후 원활한 사용을 위해서는 신규 인증 키를 생성하는 것을 권고합니다.

인증키 이름 : lab-authkey

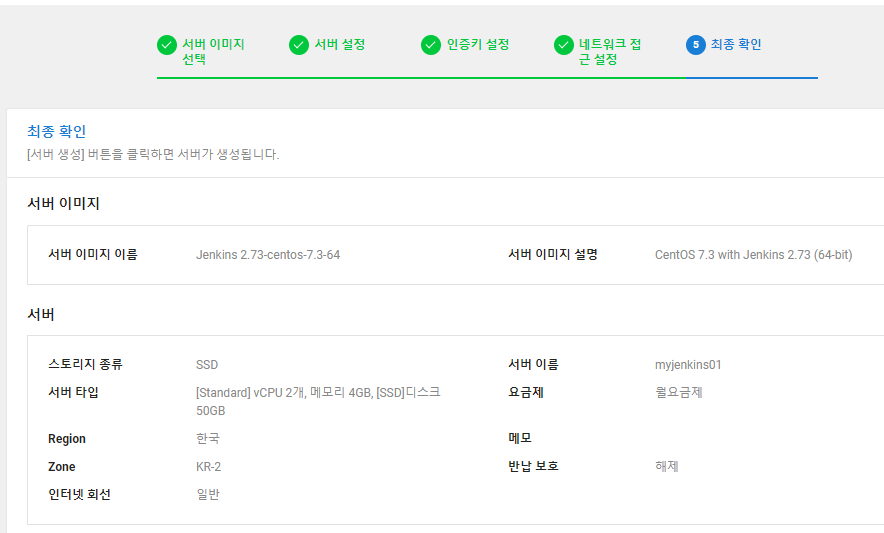


Jenkins ACG 설정은 기존 생성된 ACG중에서 lab-acg를 선택합니다.

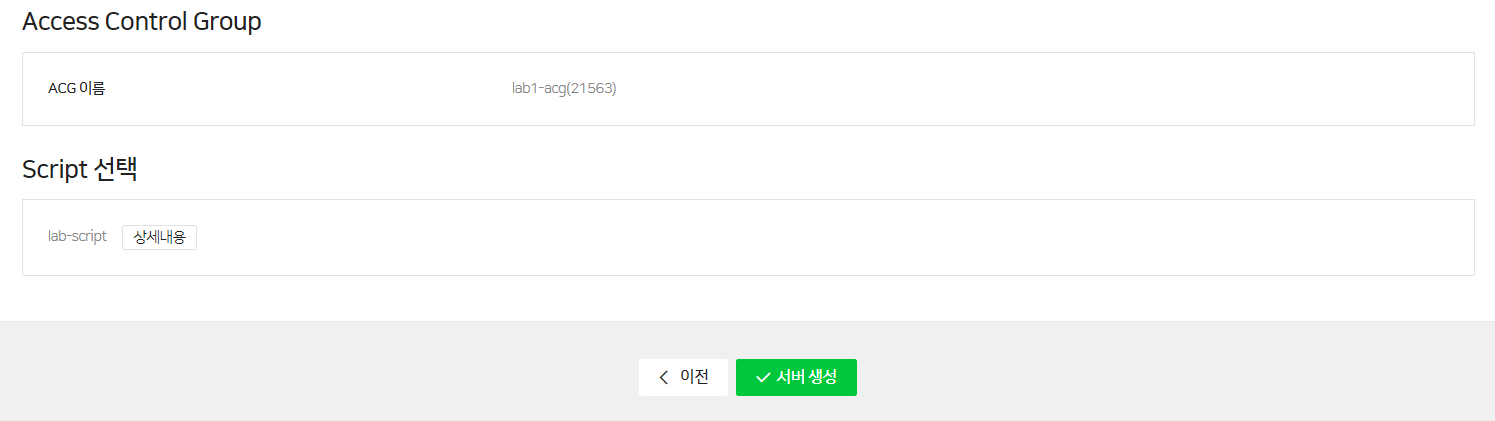


* - 한 번 ACG에 맵핑된 서버는 다른 ACG에 맵핑할 수 없습니다.

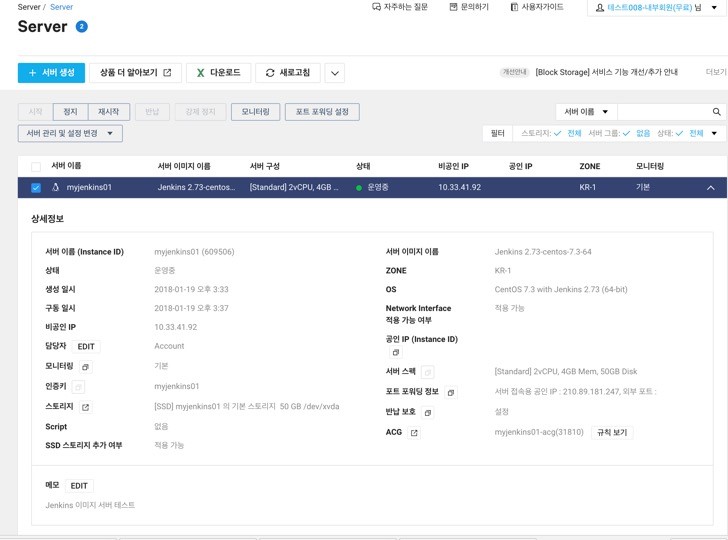
서버 생성을 클릭하면 서버가 만들어집니다



서버 생성을 클릭



- Jenkins 생성 서버 확인은 아래 그림과 같이 생성한 서버의 정보를 확인할 수 있습니다.

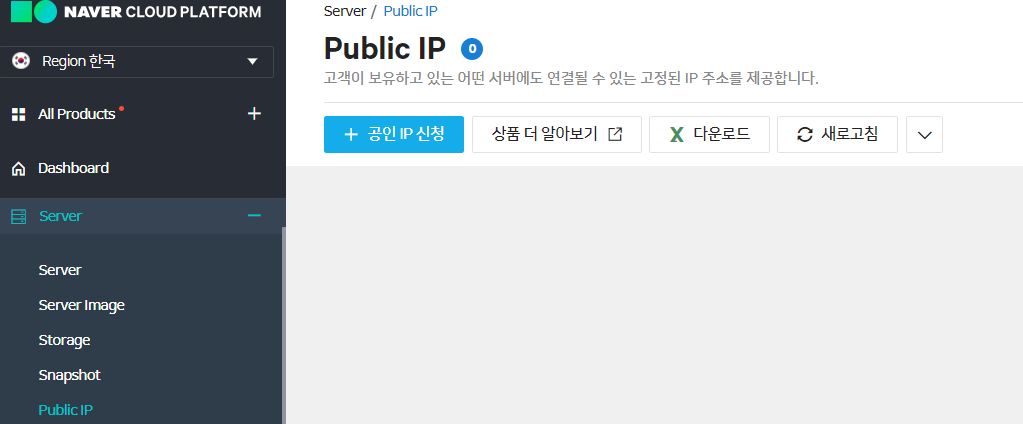


**3. 공인 IP 설정**

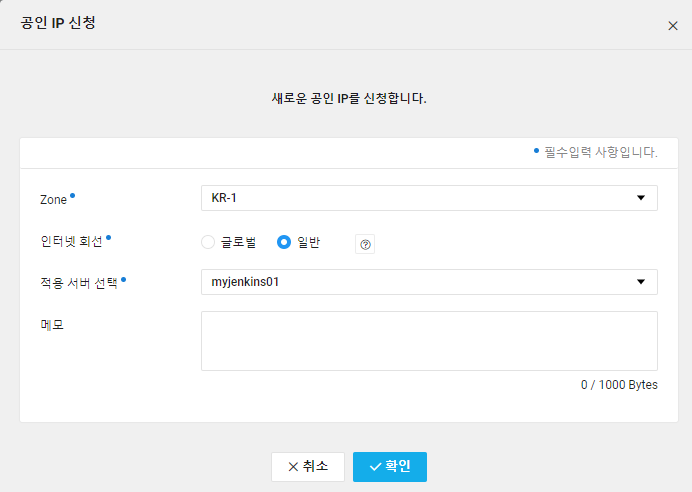
외부 접속을 위한 공인 IP 설정은 아래 그림과 같이 Jenkins 서버 접속을 위해 공인 IP를 신청 및 설정 합니다.

먼저 서버탭에서 좌측 메뉴의 “Public IP”를 선택합니다.

상단의 공인 IP 신청을 선택합니다.



적용 서버 선택에 myjenkins01을 선택합니다.



**4. 포트 포워딩 설정**

포트 포워딩  설정은 아래 그림과 같이 터미널 프로그램 접속을 위한 포트 포워딩 설정을 합니다.

기본적으로 내부 포트는 22번이며, 외부 접속을 위한 포트를 설정합니다.

Server 리스트에서 myjenin01서버를 선택하고 포트포워딩설정 버튼을 클릭합니다.

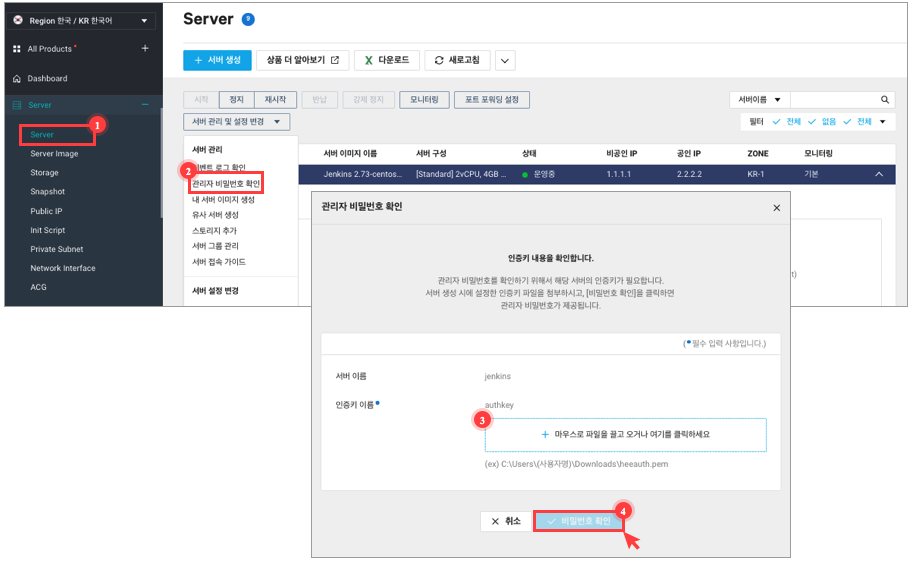


포트번호 10001을 입력하고 추가 버튼을 누릅니다.



**5. 서버 관리자 비밀번호 확인**

서버 관리자 비밀번호는 앞에서 생성한 Key File(.pem)을 불러온 후 해당 비밀번호를 확인합니다.



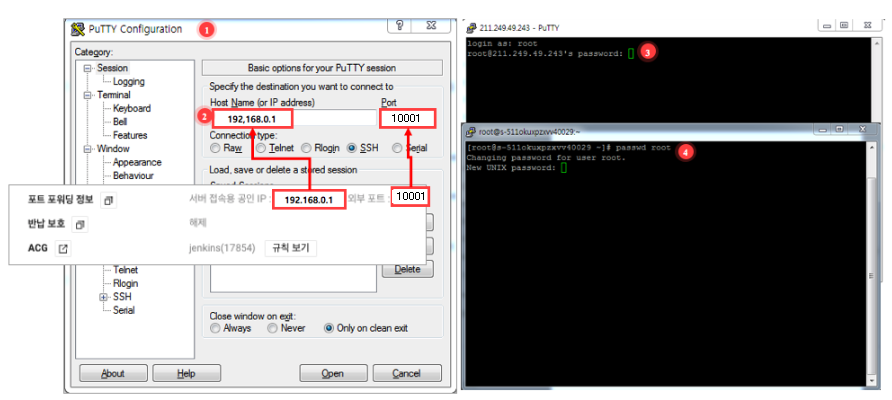
**6. 터미널 접속**

터미널 프로그램(Putty 등)으로 Myjenkins01 서버에 접속해서 Jenkins 프로세스를 관리할 수 있습니다.

터미널 프로그램으로 서버 접속 후 패스워드를 변경합니다.

로그인 아이디 : root

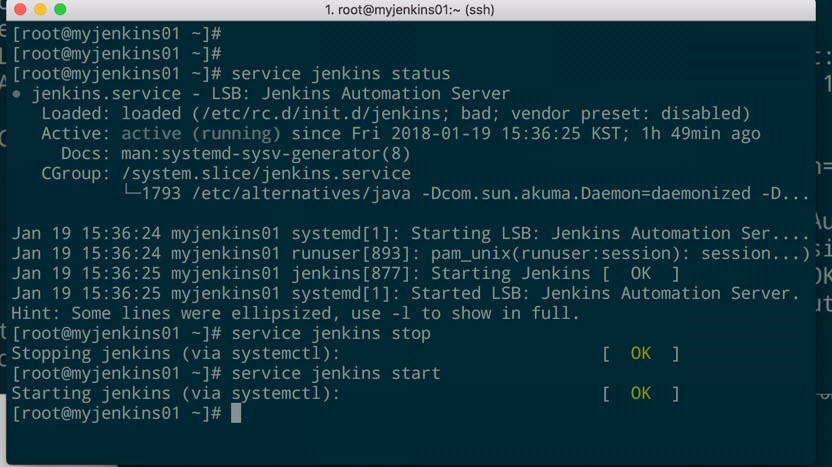
패스워드 : 위에서 확인한 관리자 비밀번호



 'passwd root' 명령어를 입력한 후 새 비밀번호를 입력합니다.

**7. Jenkins 서비스 시작**

 콘솔 프로그램으로 서버 접속은 아래와 같이 ssh로 공인 IP를 설정한 서버로 접속하여, “jenkins status|start|stop” 명령어를 통해 프로세스를 확인할 수 있습니다.

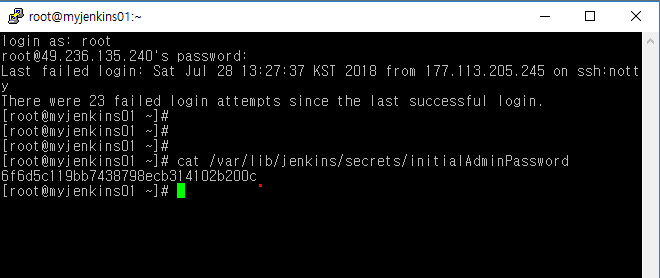
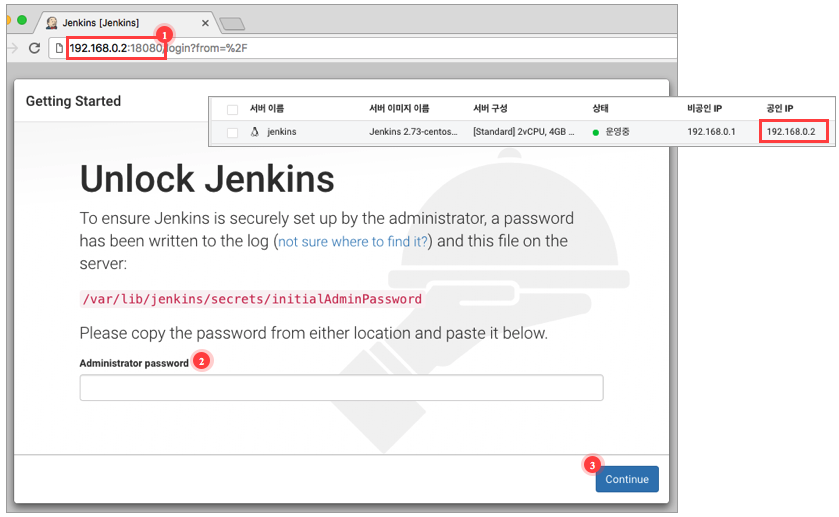


**7. Jenkins 웹 접속**

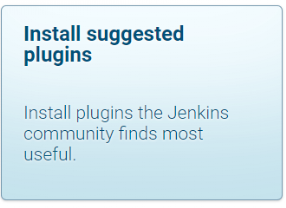
  Jenkins 서버의 공인 IP:Port로 접속하여 Web Server 접속합니다.

   “cat /var/lib/jenkins/secrets/initialAdminPassword” 명령어를 수행 후 초기 패스워를화면에 입력합니다.

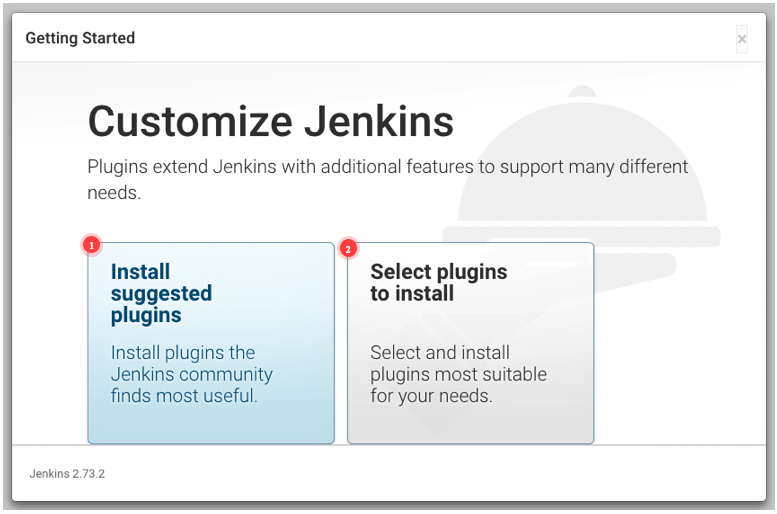
 위의 명령어를 수행하면, Hash 값 형태의 패스워드가 출력되고, 복사한 후 입력하면 됩니다.



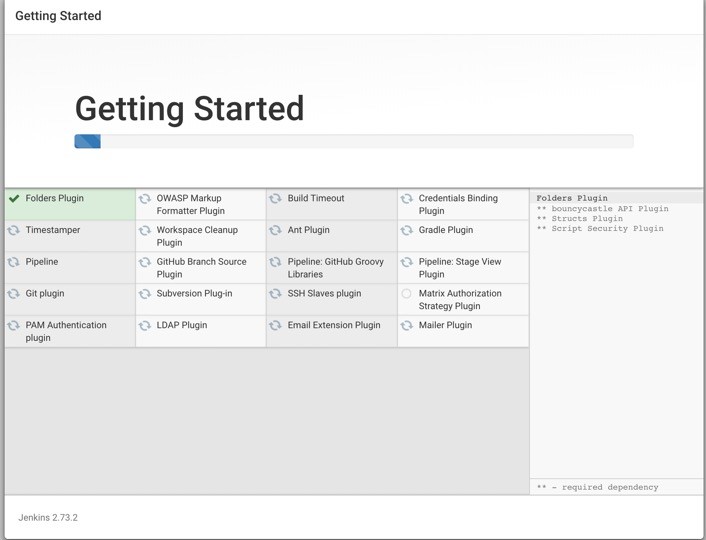
* Jenkins 플러그인 설치는 기본으로 제공되는 플러그인과 사용자가 직접 선택 후 설치를 할 수 있도록 제공해주는 방법을 활용할 수 있습니다.



선택합니다.



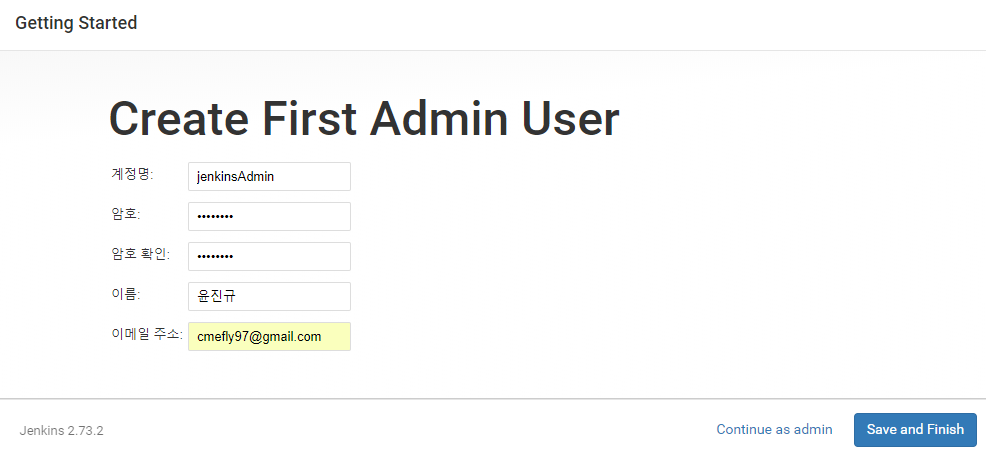
Jenkins 기본 플러그인이설치됩니다.



Jenkins Admin 계정 설정 및 완료 단계에서 계정정보, 비밀번호 등의 입력을 한 후 정보를

저장할 수 있습니다.

Jenkins 접속 계정 정보와 비밀번호를 입력합니다.



작업을 마치면 Jenkins웹 화면에서 웰컴 멘트를 확인하실 수 있습니다.

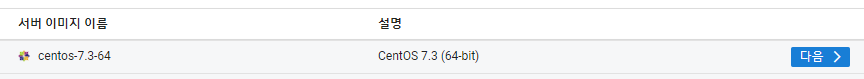


**Lab 4 : Jenkins pipeline 환경설정 1 – web 서버 만들기(5분)**

1. **웹서버 만들기**

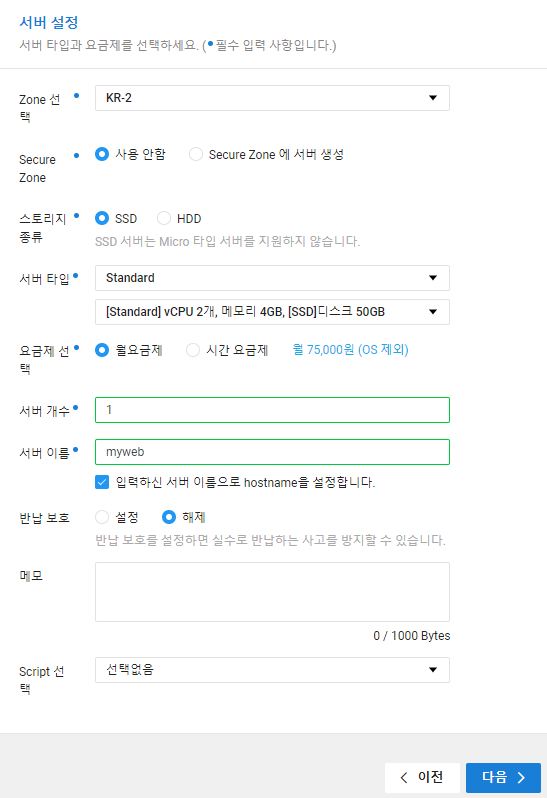
서버탭에서 서버생성을 눌러 서버를 만들 수 있습니다.

이미지는 **Centos-7.3-64**를 선택합니다.



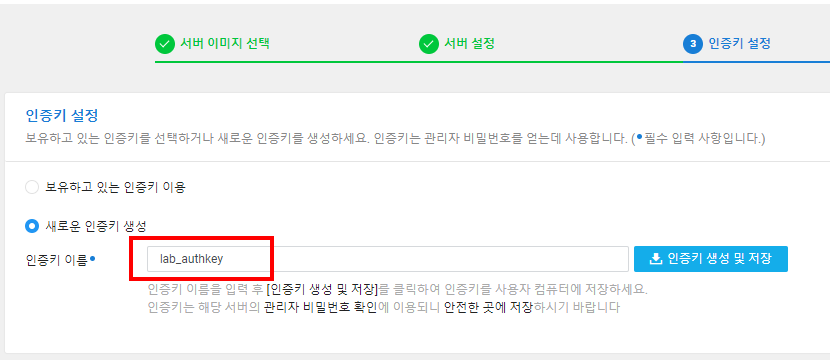
서버 타입은 vCPU 2개, 메모리 4GB, 디스크 50GB 를 선택해 주세요.

서버 개수는 1, 서버 이름은 myweb 입니다

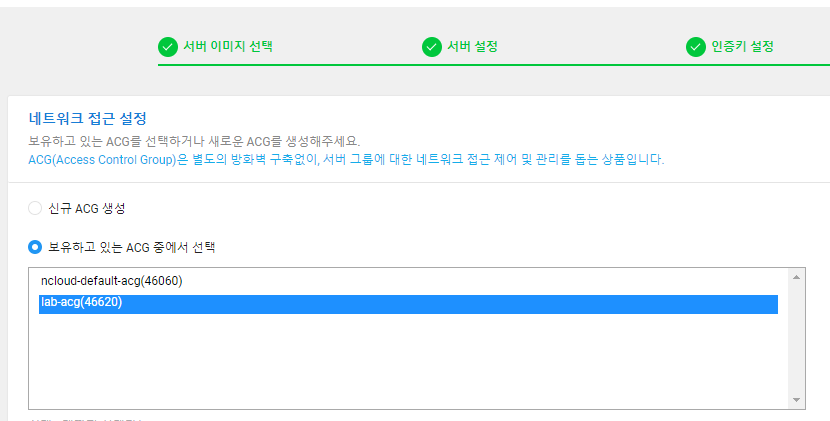


새로운 인증키 생성에서 인증키 이름에 lab\_authkey 을 입력하고 인증키 생성 및 저장을 클릭합니다.

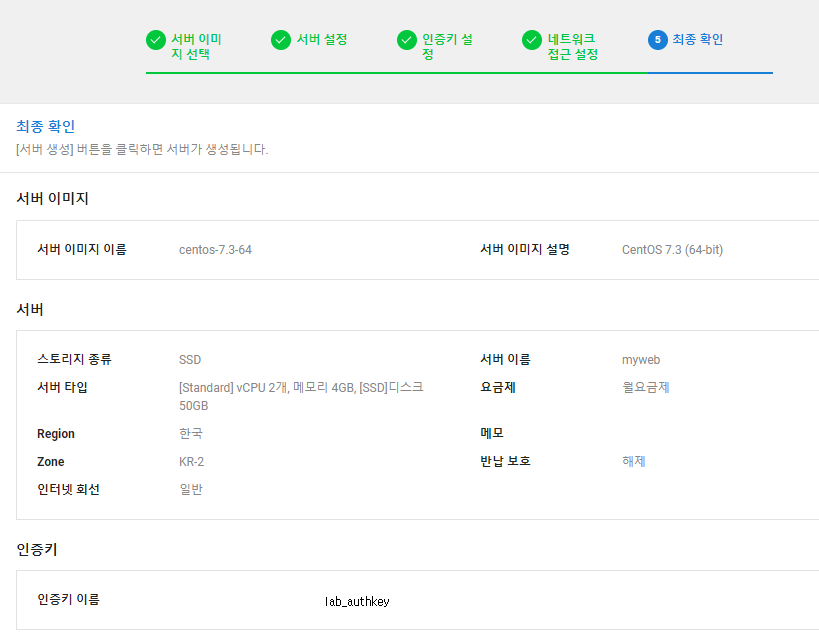
인증키 이름 : lab\_authkey



Jenkins ACG 설정은 기존 생성된 ACG중에서 lab-acg를 선택합니다.



생성을 클릭하면 서버가 만들어집니다.

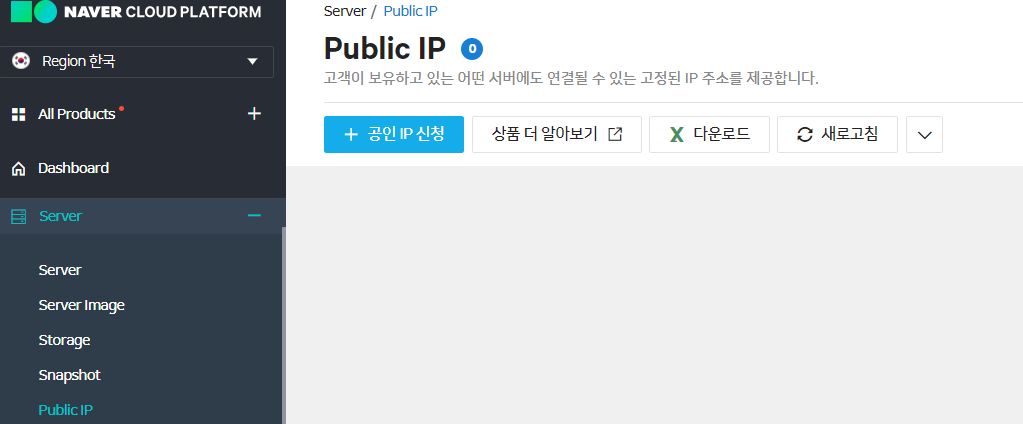


1. **공인 IP 설정**

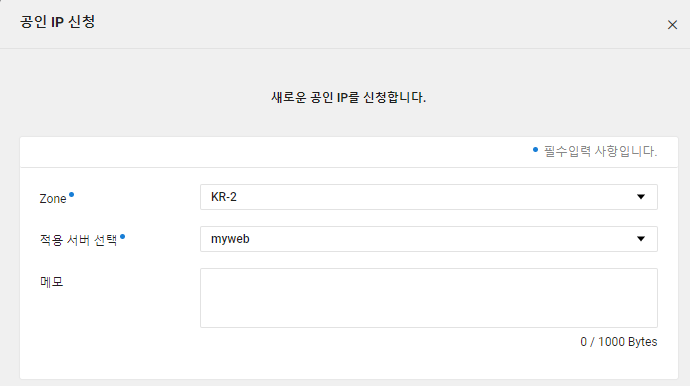
외부 접속을 위한 공인 IP 설정은 아래 그림과 같이 Jenkins 서버 접속을 위해 공인 IP를 신청 및 설정 합니다.

먼저 서버탭에서 좌측 메뉴의 “Public IP”를 선택합니다.

상단의 공인 IP 신청을 선택합니다.



적용 서버 선택에 myweb을 선택합니다.

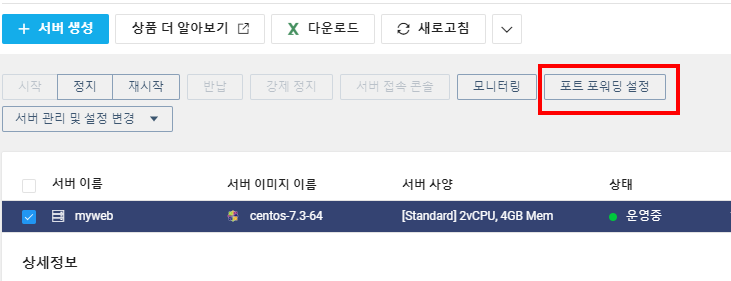


1. **포트 포워딩 설정**

포트 포워딩  설정은 아래 그림과 같이 터미널 프로그램 접속을 위한 포트 포워딩 설정을 합니다.

기본적으로 내부 포트는 22번이며, 외부 접속을 위한 포트를 설정합니다.

Server 리스트에서 myweb서버를 선택하고 포트포워딩설정 버튼을 클릭합니다.

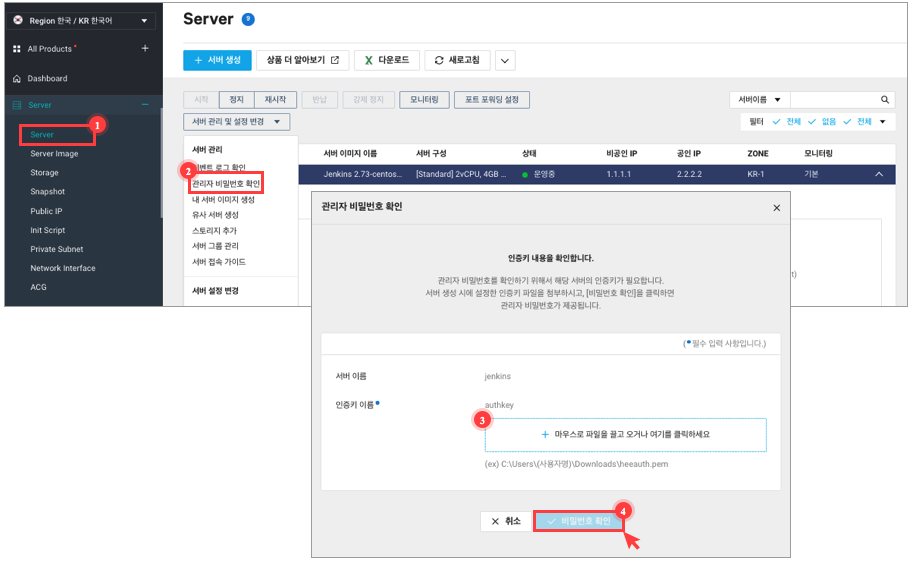


10002을 입력하고 추가 버튼을 누릅니다.



1. **서버 관리자 비밀번호 확인**

서버 관리자 비밀번호는 앞에서 생성한 Key File(.pem)을 불러온 후 해당 비밀번호를 확인합니다.

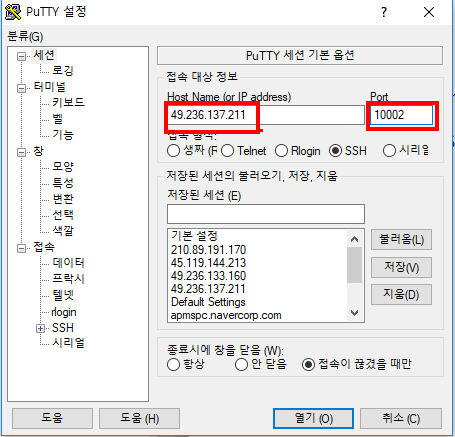


1. **터미널 접속**

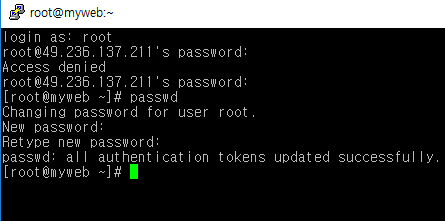
터미널 프로그램(putty 등)으로 서버 접속 후 패스워드를 변경합니다.

로그인 아이디 : root

패스워드 : 위에서 확인한 관리자 비밀번호



passwd 명령어를 입력한 후 새 비밀번호를 입력합니다.



1. **Java JDK 설치**

**yum** 명령어로 java jdk1.8을 설치해 줍니다.

[root@myjenkins01 ~]# yum install java-1.8.0-openjdk-devel.x86\_64

설치 확인

[root@myjenkins01 ~]# rpm -qa java\*jdk-devel

java-1.8.0-openjdk-devel-1.8.0.181-3.b13.el7\_5.x86\_64

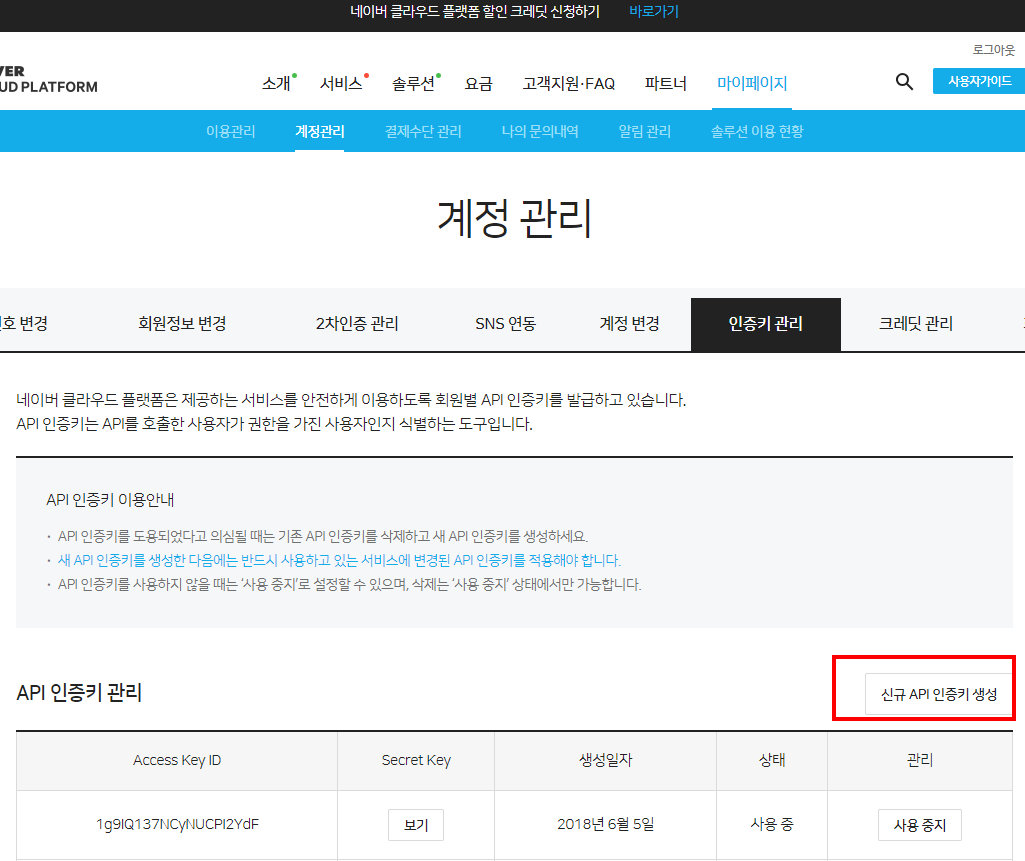
[root@myjenkins01 ~]# javac -version

javac 1.8.0\_181

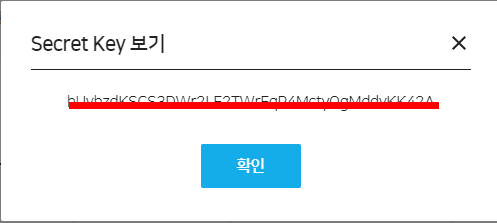
**Lab 5 : Jenkins pipeline 환경설정 2 - ObjectStorage에 폴더 만들기(5분)**

1. **인증키 준비**

nCloud 포털 > 마이페이지 > 계정관리 > 인증키 관리로 이동하여 인증키를 만듭니다.



**클릭**



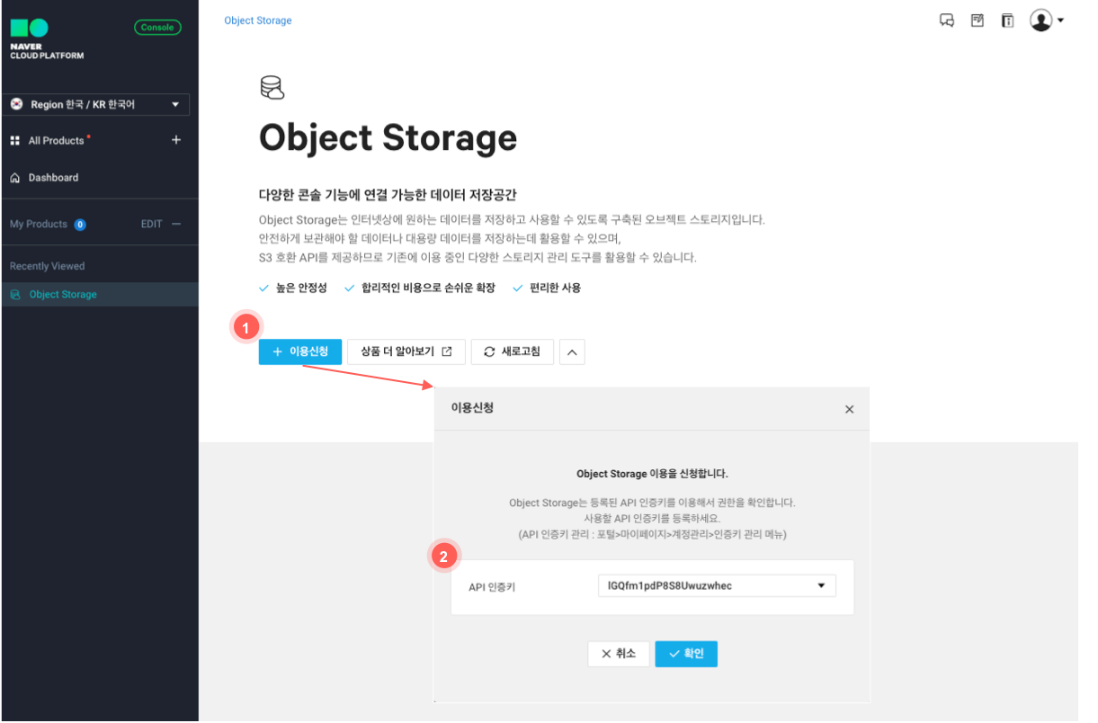
AccessKey ID 를 확인하시고

Secret Key는 보기 버튼을 클릭하면 확인할 수 있습니다.

1. **Object Storage 이용신청**

콘솔 > All Products > Object Storage 로 이동합니다.

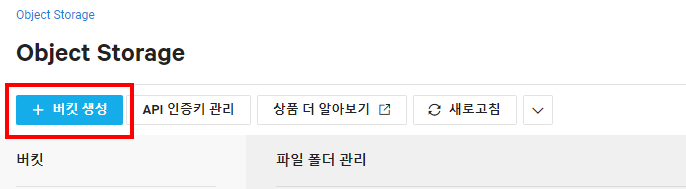
이용신청 버튼을 클릭하여 이용신청합니다



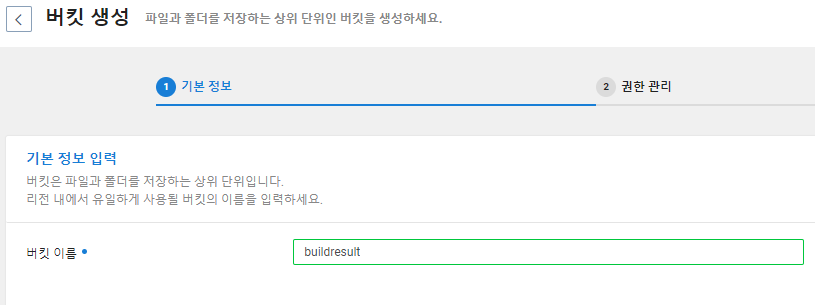
Object Storage에서 사용할 인증키를 선택합니다.

1. **버킷 만들기**

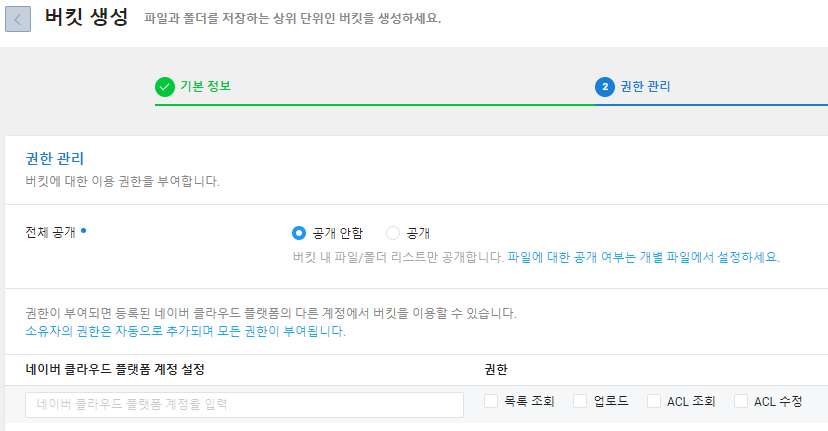
콘솔 > All Products > Object Storage > 버킷 생성 버튼을 선택합니다.



버킷이름을 buildresult 로 입력합니다.

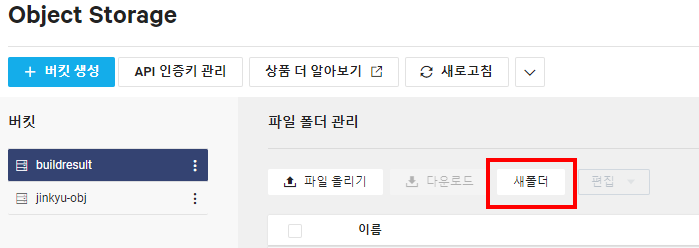


버킷에 대한 권한을 공개안함 default 로 그대로 두고 생성을 완료합니다.

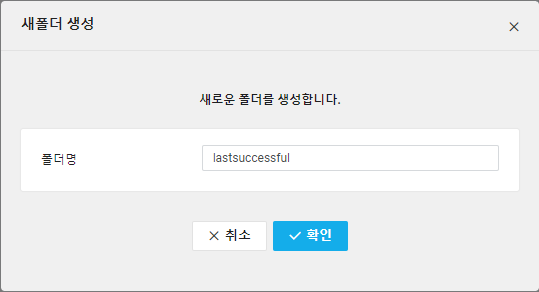
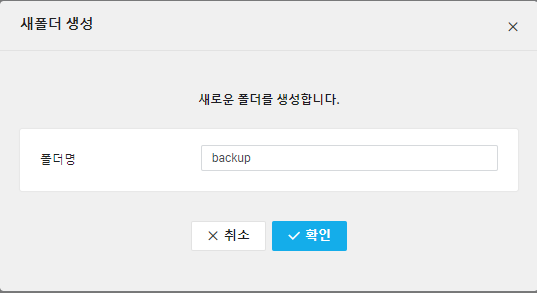


1. **폴더만들기**

Object Storage 화면에서 새폴더를 선택합니다.



backup , lastsuccessful 라는 이름의 두개의 폴더를 생성합니다.

**Lab 6 : Jenkins pipeline 환경설정 3**

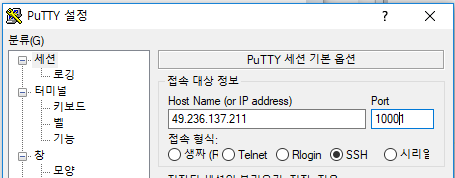
**Jenkins 서버와 Web 서버간 ssh 설정(5분)**

1. **SSH 키 생성하기**

Jenkins 서버에 터미널(putty 등)로 로그인하여 ssh-key 를 생성합니다.

IP : 포트포워드에서 설정한 IP

Port 번호 : 10001



Key 생성 : **ssh-keygen-t rsa** 입력

[root@myjenkins01 ~]# ssh-keygen -t rsa

Passphrase : 빈값 ( SSH 자동로그인을 위해 키인증 패스워드 없음)

Created directory '/root/.ssh'.

Enter passphrase (empty for no passphrase):

Enter same passphrase again:

Your identification has been saved in /root/.ssh/id\_rsa.

Your public key has been saved in /root/.ssh/id\_rsa.pub.

The key fingerprint is:

df:02:23:e7:d7:4c:b1:77:2d:52:c2:dd:48:5f:67:09 root@myjenkins01

The key's randomart image is:

+--[ RSA 2048]----+

| Eo =|

| . o \*o|

| + + o|

| = .|

| . S + o o|

| + + = o o |

| . + + |

| . . |

| |

+-----------------+

1. **SSH public key 전송**

myweb 서버에 public key를 전송합니다.

[root@myjenkins01 ~]# ssh-copy-id root@10.41.2.48 (web서버 사설 ip)

myweb 서버의 패스워드를 입력합니다.

[root@myjenkins01 ~]# ssh-copy-id root@10.41.2.48

/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed

/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys

root@10.41.2.48's password:

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'root@10.41.2.48'"

and check to make sure that only the key(s) you wanted were added.

참고 ) ssh-copy-id username@host 명령어가 실행되지 않는다면

$ cat ~/.ssh/id\_rsa.pub | ssh user@host 'cat >> ~/.ssh/authorized\_keys’ 명령어로 대체할 수 있습니다.

ssh 접속 테스트를 해봅니다.

[root@myjenkins01 ~]# ssh root@10.41.2.48 ls /var/

[root@myjenkins01 ~]# cat "test" > test.text

[root@myjenkins01 ~]# ssh root@10.41.2.48 mkdir /var/www/

[root@myjenkins01 ~]# scp test.text [root@10.41.2.48:/var/www/](mailto:root@10.41.2.48:/var/www/)

[root@myjenkins01 ~]# ssh root@10.41.2.48 ls /var/www/

Jenkins 에서 ssh\_key 사용을 위해 Jenkins 폴더 하위에 복사해 줍니다.

Id\_rsa 파일 접근 권한을 추가합니다.

Known\_hosts 파일 쓰기 권한을 추가합니다.

[root@myjenkins01 ~]# cp -r ~/.ssh/ /var/lib/jenkins/

[root@myjenkins01 ~]# chmod 755 /var/lib/jenkins/.ssh/id\_rsa

[root@myjenkins01 .ssh]# chmod 766 /var/lib/jenkins/.ssh/known\_hosts

**Lab 7 : Jenkins pipeline 환경설정 4**

**Python SDK for S3 사용하여 Object storage에 파일올리기 (10분)**

1. **Jenkins 서버에 BOTO3 SDK설치**

AWS S3에서 제공하는 Python SDK를 이용하여 네이버 클라우드 플랫폼 Object Storage에

파일을 올리거나 내려받을 수 있습니다.

**BOTO3 SDK설치**

먼저 Jenkins 서버에 터미널로 로그인 합니다.

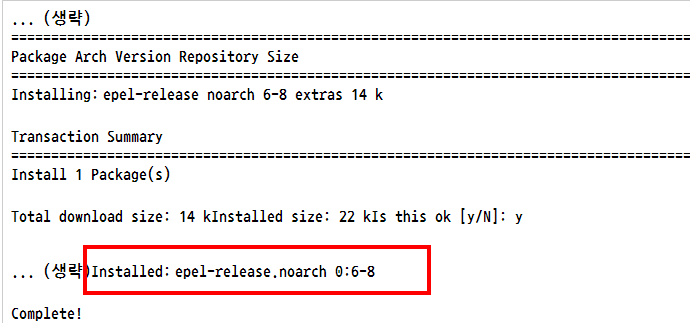
먼저 epel (extra packages for enterprise linux yum)을 설치해야 합니다.

1. epel이 설치되어 있는지 확인

[root@myjenkins01 ~]# yum repolist

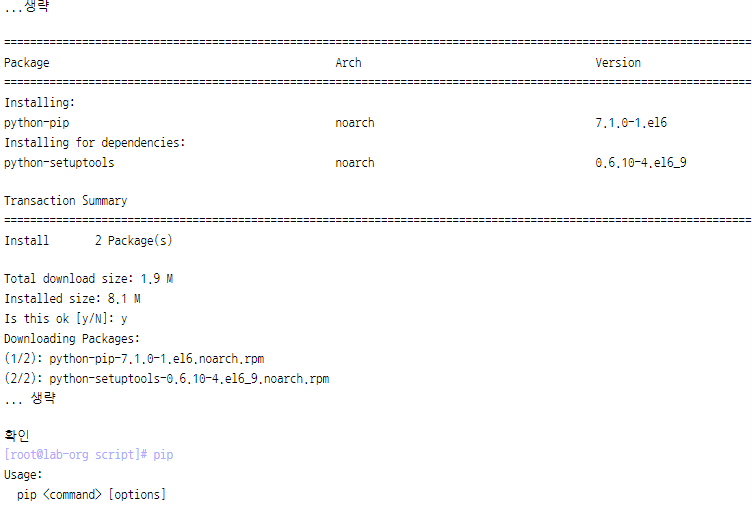
1. 없으면 epel repository를 추가합니다.

[root@myjenkins01 ~]# yum install epel-release



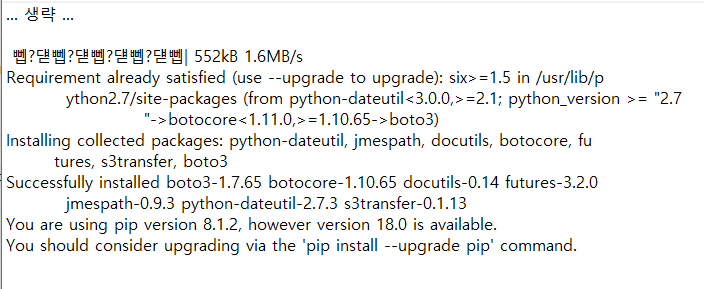
1. Python pip을 설치합니다.

[root@myjenkins01 ~]# yum install python-pip



1. Python SDK for S3 API 설치

[root@myjenkins01 ~]# pip install boto3



1. **Object storage에 빌드결과 파일을 upload/download 하기 위한 파이썬 스크립트 작성**
2. /home/objectstorage/**script**/ , /home/objectstorage/**backup** 폴더를 생성합니다.

Backup 폴더에는 jenkins 프로세스가 쓰고 읽을 수 있게 권한을 추가합니다.

[root@myjenkins01 ~]# mkdir –p /home/devtrack/pyscript/

[root@myjenkins01 ~]# mkdir /home/devtrack/deploy/backup/

[root@myjenkins01 home]# chmod 777 /home/ devtrack/deploy /backup/

1. 빌드 결과물 백업을 위한 backup\_war.py 파일을 생성합니다.

[root@myjenkins01 ~]# vi /home/ devtrack/pyscript/backup\_war.py

|  |
| --- |
| import boto3  import time  now = time.time()  service\_name = 's3'  endpoint\_url = '[https://kr.objectstorage.ncloud.com](https://kr.objectstorage.ncloud.com/)'  region\_name = 'kr-2'  **Access\_key, secret\_key 변경**  access\_key = '1g9IQ137NCyNUCPI2YdF'  secret\_key = 'hUvbzdKSCS3DWr2LF2TWrFqP4MctyOgMddvKK42A'  s3 = boto3.client(service\_name, endpoint\_url=endpoint\_url, aws\_access\_key\_id=access\_key,  aws\_secret\_access\_key=secret\_key)  bucket\_name = 'buildresult'  object\_name = 'lastsuccessful/helloNcp-0.0.1-SNAPSHOT.war'  local\_file\_path = '/home/devtrack/deploy/backup/helloNcp\_1.1.0\_SNAPSHOT.war.' + str(now)  print local\_file\_path  # download file  s3.download\_file(bucket\_name, object\_name, local\_file\_path)  # upload file  object\_name = 'backup/helloNcp-0.0.1-SNAPSHOT.war' + str(now)  s3.upload\_file(local\_file\_path, bucket\_name, object\_name) |

1. 빌드 결과물 업로드를 위한 upload\_war.py 파일을 생성합니다.

[root@myjenkins01 ~]# vi /home/devtrack/pyscript/upload\_war.py

|  |
| --- |
| import boto3  service\_name = 's3'  endpoint\_url = '[https://kr.objectstorage.ncloud.com](https://kr.objectstorage.ncloud.com/)'  region\_name = 'kr-2'  **Access\_key, secret\_key 변경**  access\_key = '1g9IQ137NCyNUCPI2YdF'  secret\_key = 'hUvbzdKSCS3DWr2LF2TWrFqP4MctyOgMddvKK42A'  s3 = boto3.client(service\_name, endpoint\_url=endpoint\_url, aws\_access\_key\_id=access\_key,                    aws\_secret\_access\_key=secret\_key)  bucket\_name = 'buildresult'  # create folder  object\_name = 'lastsuccessful/'  s3.put\_object(Bucket=bucket\_name, Key=object\_name)  # upload file  object\_name = 'lastsuccessful/helloNcp-0.0.1-SNAPSHOT.war'  local\_file\_path = '/var/lib/jenkins/workspace/jenkinpipeline/target/helloNcp-0.0.1-SNAPSHOT.war'  s3.upload\_file(local\_file\_path, bucket\_name, object\_name)  ~ |

**3. Web 서버에서 스프링부트의 war 파일 자동 실행을 위한 스크립트 작성**

myweb 서버에 터미널 로그인 합니다.

1. /var/www/script 폴더를 생성합니다.

[root@myweb ~]# mkdir -p / home/devtrack/script/

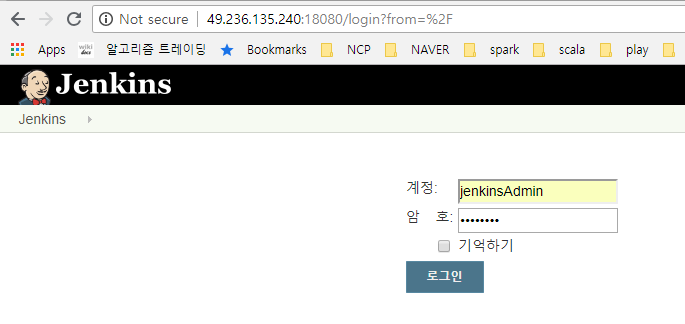
1. 스프링 부트 War 파일을 자동으로 Start|Stop 하기 위한 스크립트

/var/www/script/runNcp.sh 를 생성합니다.

|  |
| --- |
| #!/bin/bash  # chkconfig 2345 20 80  # written by Gavin Kim on 2018.05.18  # 프로세스 명을 명시한다.  readonly PROC\_NAME="helloNcp"  # war 파일  readonly DAEMON="/home/devtrack/deploy/helloNcp-0.0.1-SNAPSHOT.war"  # 프로세스 아이디가 존재할 패스를 설정  readonly PID\_PATH="/home/devtrack/deploy/"  readonly PROC\_PID="${PID\_PATH}${PROC\_NAME}.pid"  # 시작 함수  start()  {      echo "Starting  ${PROC\_NAME}..."      local PID=$(get\_status)      if [ -n "${PID}" ]; then          echo "${PROC\_NAME} is already running"          exit 0      fi      nohup java -jar -XX:MaxPermSize=128m -Xms512m -Xmx1024m "${DAEMON}" > /dev/null 2>&1 &      local PID=${!}      if [ -n ${PID} ]; then          echo " - Starting..."          echo " - Created Process ID in ${PROC\_PID}"          echo ${PID} > ${PROC\_PID}      else          echo " - failed to start."      fi  }  # 중지  stop()  {      echo "Stopping ${PROC\_NAME}..."      local DAEMON\_PID=`cat "${PROC\_PID}"`      if [ "$DAEMON\_PID" -lt 3 ]; then          echo "${PROC\_NAME} was not  running."      else          kill $DAEMON\_PID          rm -f $PROC\_PID          echo " - Shutdown ...."      fi  }  status()  {      local PID=$(get\_status)      if [ -n "${PID}" ]; then          echo "${PROC\_NAME} is running"      else          echo "${PROC\_NAME} is stopped"          # start daemon          #nohup java -jar "${DAEMON}" > /dev/null 2>&1 &      fi  }  get\_status()  {      ps ux | grep ${PROC\_NAME} | grep -v grep | awk '{print $2}'  }  # 케이스 별로 함수를 호출하도록 한다.  case "$1" in      start)          start          sleep 7          ;;      stop)          stop          sleep 5          ;;      status)      status "${PROC\_NAME}"      ;;      \*)      echo "Usage: $0 {start | stop | status }"  esac  exit 0 |

**Lab 8 : Jenkins pipeline 을 통한 자동 빌드/배포 구현 (10분)**

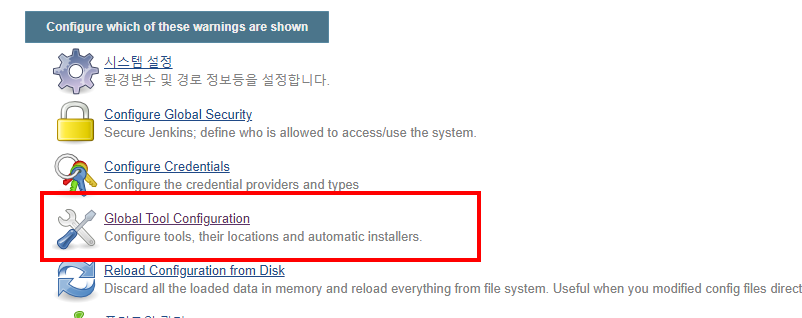
1. **Jenkins web 서버에 로그인합니다.**



1. **Maven 빌드와 java jdk설절을 구성하기 위해 Jenkins 관리를 클릭합니다.**



Global Tool Configuration 을 선택합니다.

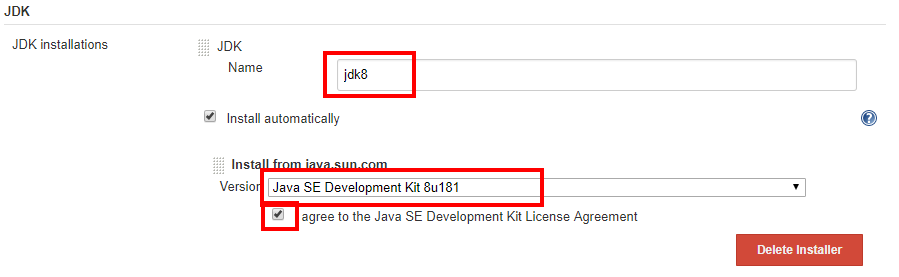


JDK 설정을 추가 합니다.

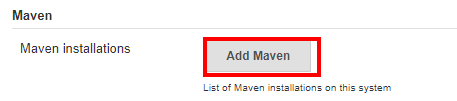


JDK name 을 jdk8 로 입력 , Version을 Java SE Development Kit 8u181로 선택합니다.

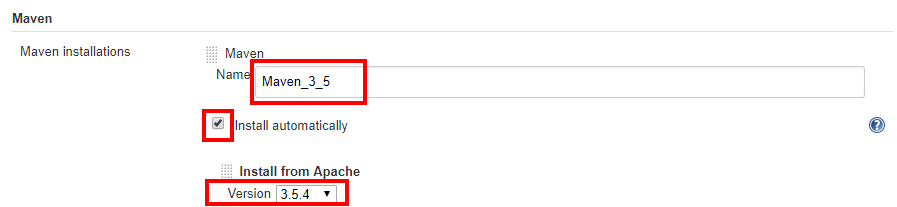
I agree to the Java SE Development Kit License Agreement 를 체크합니다.



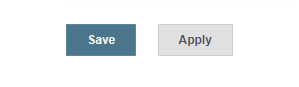
Maven 설정을 추가 합니다.



Name 을 Maven\_3\_5, Version 을 3.5.4 를 선택합니다.

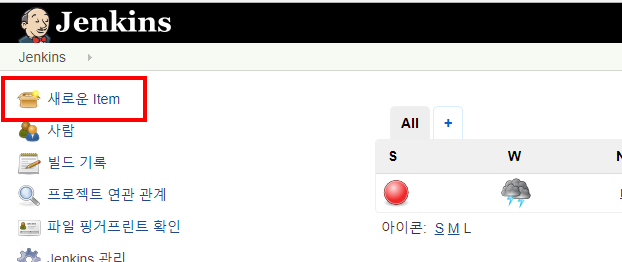


저장 버튼을 눌러 설정을 저장합니다.

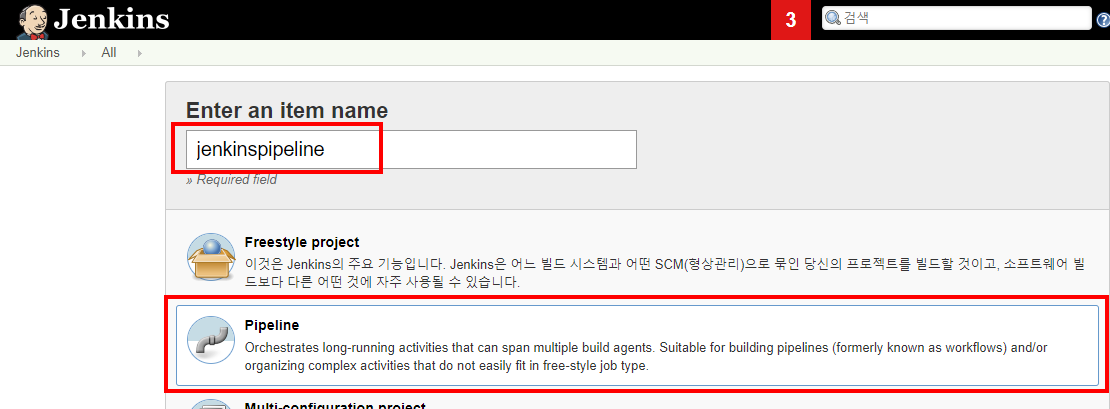


1. **pipeline-project 생성하기**

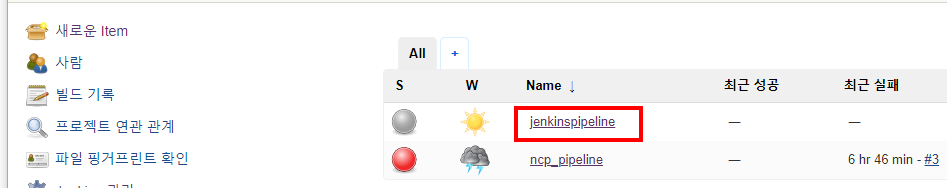
새로운 Item을 클릭합니다.



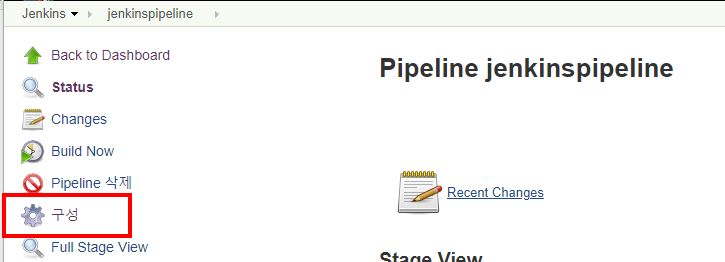
Job 이름을 jenkinspipeline 으로 입력하고 , pipeline-project를 생성합니다.



프로젝트 리스트에서 jenkinspipeline 을 선택합니다.



구성을 클릭합니다.



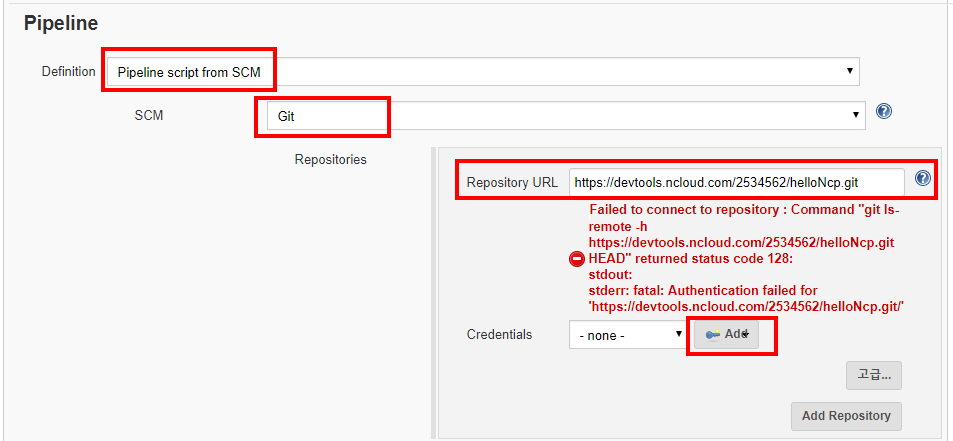
Pipeline 항목에 정보를 입력합니다.

Definition : Pipeline script from SCM

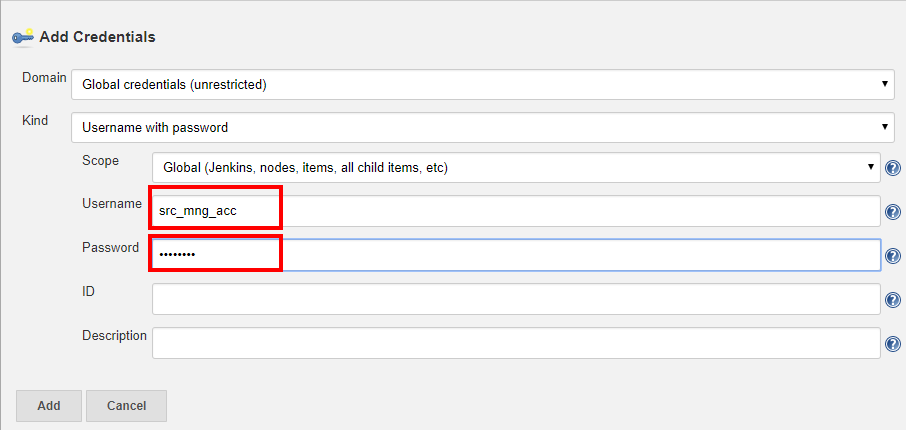
SCM : Git

Repository : SourceCommit 리파지토리 경로 입력

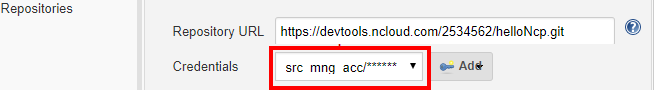
예) https://devtools.ncloud.com/2534562/helloNcp.git



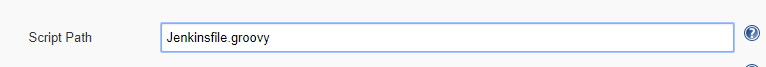
Add 버튼을 눌러 git 리포지토리 username 과 password를 입력합니다.



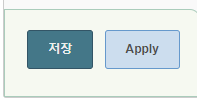
자격 증명 사용자에 acc\_mng\_acc 를 선택합니다.



Script Path를 **Jenkinsfile.groovy 로 입력합니다.**

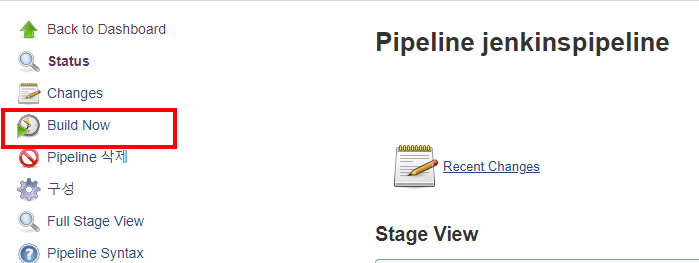


저장을 눌러 정보를 저장합니다.



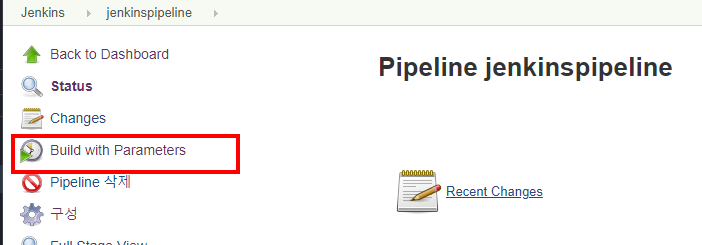
1. **빌드/배포 테스트하기**

빌드를 클릭합니다.



처음 빌드는 기본 라메터값이 없어서 에러가 발생합니다.

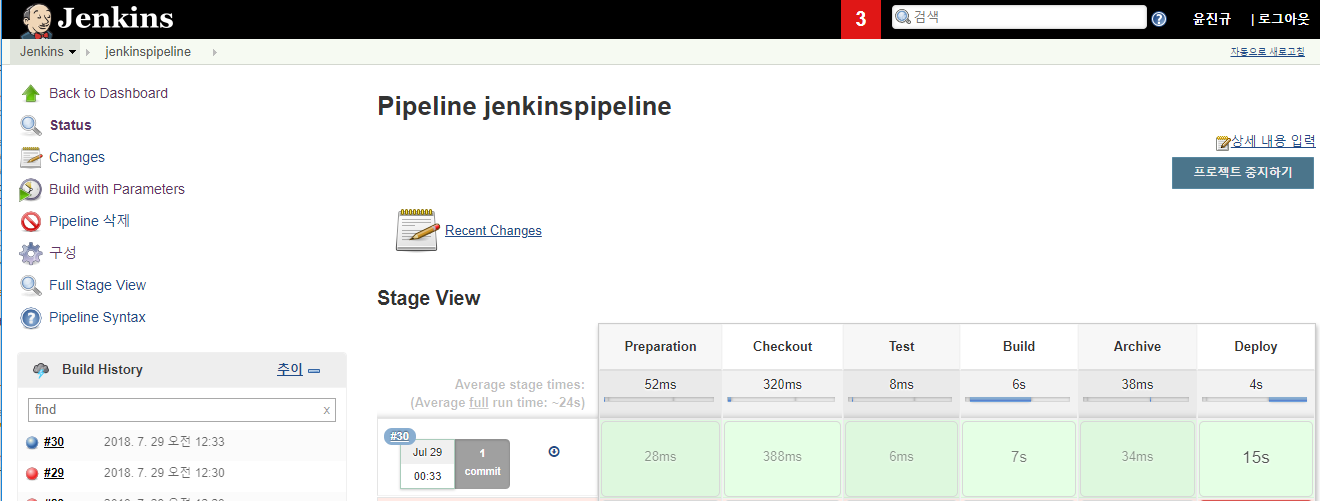
새로고침하면 Build with Parameters 메뉴가 보입니다.



Build With Parameters를 선택하고 빌드하기를 선택합니다.



결과를 확인한다.



웹서버 접속

