

# 시스템프로그래밍(01<sub>분반</sub>)

## 터미널 SSH(Host)로 Guest-OS에 접속하기

2017136063 여승준

SSH란 **S**ecure **S**hell의 약자로, 네트워크 상에서 안전하게 원격지 호스트에 로그인하고 명령을 실행할 수 있는 프로토콜이다. (\* 기본 포트번호는 22번이다.)

ssh를 동작시키려면 원격과 로컬 pc에 ssh-server와 ssh-client 프로그램이 각각 필요하며, ssh의 기본 동작을 간단히 살펴보면 다음과 같다.

1. 원격 컴퓨터에 ssh 서버를 설치한다. 그리고 로컬에 ssh-client 프로그램을 설치한다.  
(Ubuntu: ssh-server 패키지 install / Windows10: terra-term 다운로드)
2. 로컬에서 ssh 키를 생성하면, Public key, Private Key를 각각 생성해준다.
3. 이 중 public Key를 통신하고자 하는 remote(원격)에 등록시켜둔 다음, private Key를 가지고 통신한다.  
(\* 비공개키를 등록시켜둔 다음 공개키를 대조해가며 통신하는 것은 non-sense겠조?)

뒤에서 살펴보겠지만 'terraterrm'이라는 ssh-client 프로그램을 사용했더니 따로 원격 서버에 public key를 등록시키는 절차를 생략 가능했다.

SSH를 이용하여 원격OS에 접속하는 큰 틀에서의 방법은 다음과 같다.

1. 원격, 로컬 각각의 Host-OS IP주소 확인  
(Windows10 - powershell에서 ipconfig 입력하면 VMWare의 ip주소까지 모두 확인할 수 있었다.)
2. (\* 선택) VMWare virtual 네트워크 어댑터가 NAT 방식으로 1개 뿐이기 때문에 bridged 방식의 어댑터를 하나 더 추가해준다.
  - NAT(Network Address Translation) 방식  
: 일반적인 의미는 IP부족 문제의 해결책으로 인해 공인IP 하나에 내부적인 여러 사설 IP를 변환해 쓰는 방식을 뜻한다. 마찬가지로 VMWare에서도 NAT과 유사하게 guestOS가 hostOS에게 내부적으로만 사용할 새로운 IP를 할당해주는 방식을 말한다.
  - Bridged 방식  
: guestOS가 사용중인 사설IP와 같은 네트워크에서 IP를 할당받아 사용하는 방식(비교적 간단).
3. 로컬 ssh-client에서 원격 ssh-server에 접속하여 컴퓨터를 다룬다.

정리하자면 다음과 같다.

**[NAT 방식의 network adapter에 ssh 접속] :**

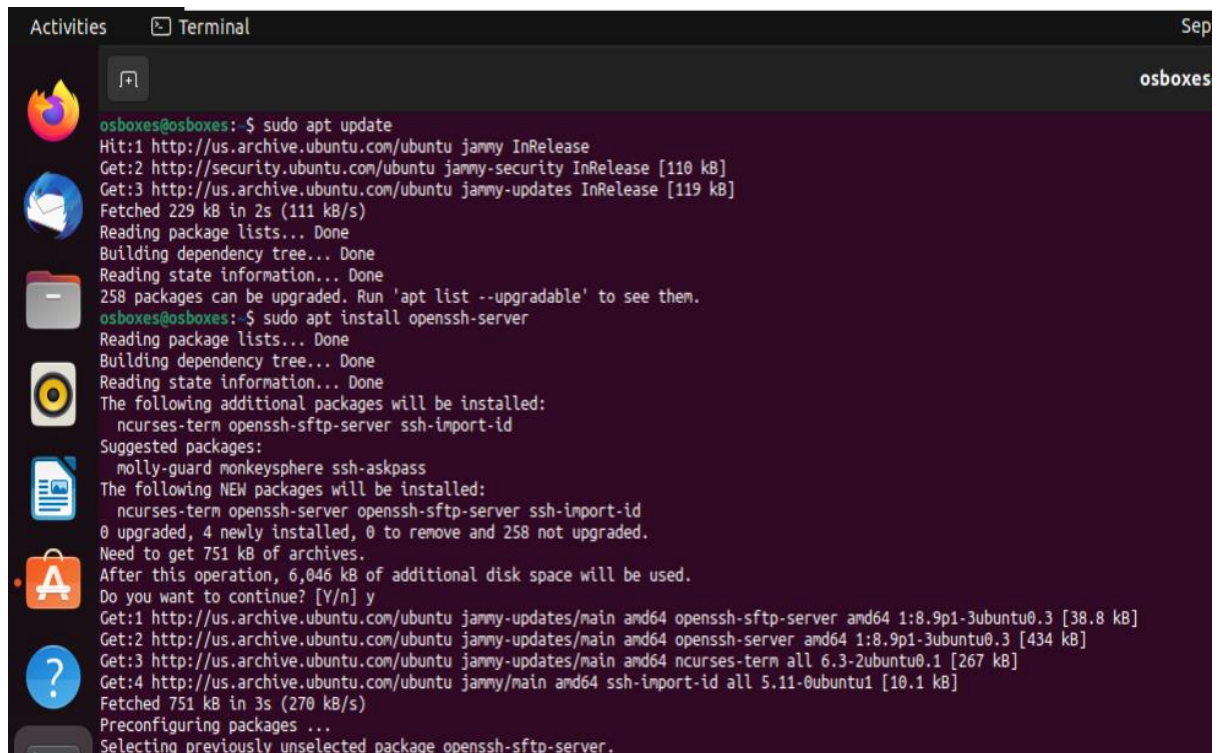
HostOS(원격)에 ssh-server설치, GuestOS(로컬)에 ssh-client 설치 → GuestOS와 HostOS의 IP주소 모두를 메모 → VMWare Network Editor를 설치&실행(\*pro버전의 경우 내장) → NAT setting에서 port-forwarding 설정수정 → 원격 HostOS 실행 → 로컬 ssh-client에서 VMWare NAT setting 해둔대로 접속 & ssh-key생성 과정 & 접속

**[Bridged 방식의 network adapter에 ssh 접속] :**

HostOS(원격)에 ssh-server설치, GuestOS(로컬)에 ssh-client 설치 → GuestOS와 HostOS의 IP주소 모두를 메모 → 로컬 ssh-client에서 앞서 메모해둔 HostOS IP 입력 후 접속

# 상세설명

1. 첫 단계로 Ubuntu에 openssh-server를 설치  
+ GuestOS에서 ssh-client 프로그램 다운로드 (\*terra term)



A terminal window titled 'Terminal' with a dark background. The prompt is 'osboxes@osboxes:~\$'. The user enters 'sudo apt update', followed by 'sudo apt install openssh-server'. The terminal shows the output of these commands, including package lists, dependency trees, and the installation of openssh-server along with other packages like ncurses-term, openssh-sftp-server, and ssh-import-id. The user is prompted to continue with 'y'.

```
osboxes@osboxes:~$ sudo apt update
Hit:1 http://us.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Get:3 http://us.archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Fetched 229 kB in 2s (111 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
258 packages can be upgraded. Run 'apt list --upgradable' to see them.
osboxes@osboxes:~$ sudo apt install openssh-server
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  ncurses-term openssh-sftp-server ssh-import-id
Suggested packages:
  molly-guard monkeysphere ssh-askpass
The following NEW packages will be installed:
  ncurses-term openssh-server openssh-sftp-server ssh-import-id
0 upgraded, 4 newly installed, 0 to remove and 258 not upgraded.
Need to get 751 kB of archives.
After this operation, 6,046 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://us.archive.ubuntu.com/ubuntu jammy-updates/main amd64 openssh-sftp-server amd64 1:8.9p1-3ubuntu0.3 [38.8 kB]
Get:2 http://us.archive.ubuntu.com/ubuntu jammy-updates/main amd64 openssh-server amd64 1:8.9p1-3ubuntu0.3 [434 kB]
Get:3 http://us.archive.ubuntu.com/ubuntu jammy-updates/main amd64 ncurses-term all 6.3-2ubuntu0.1 [267 kB]
Get:4 http://us.archive.ubuntu.com/ubuntu jammy/main amd64 ssh-import-id all 5.11-0ubuntu1 [10.1 kB]
Fetched 751 kB in 3s (270 kB/s)
Preconfiguring packages ...
Selecting previously unselected package openssh-sftp-server.
```

- sudo apt update
- sudo apt install openssh-server



A terminal window showing the output of the command 'service --status-all'. The output lists various services with their status, including 'ssh' which is marked as '[ + ]', indicating it is enabled and running. The prompt is 'osboxes@osboxes:~\$'.

```
osboxes@osboxes:~$ service --status-all
[ + ] acpid
[ - ] alsa-utils
[ - ] anacron
[ + ] apparmor
[ + ] apport
[ + ] avahi-daemon
[ + ] bluetooth
[ - ] console-setup.sh
[ + ] cron
[ + ] cups
[ + ] cups-browsed
[ + ] dbus
[ + ] gdm3
[ - ] grub-common
[ - ] hwclock.sh
[ + ] irqbalance
[ + ] kerneloops
[ - ] keyboard-setup.sh
[ + ] knod
[ + ] open-vm-tools
[ + ] openvpn
[ - ] plymouth
[ + ] plymouth-log
[ + ] procps
[ - ] pulseaudio-enable-autospawn
[ - ] rsync
[ - ] saned
[ - ] speech-dispatcher
[ - ] spice-vdagent
[ + ] ssh
[ + ] udev
[ + ] ufw
[ + ] unattended-upgrades
[ - ] uuidd
[ - ] whoopsie
[ - ] x11-common
osboxes@osboxes:~$ service --status-all | grep 'ssh'
[ + ] ssh
osboxes@osboxes:~$
```

위 명령어 실행 후 시스템에서 ssh가 돌아가고 있음을 확인.

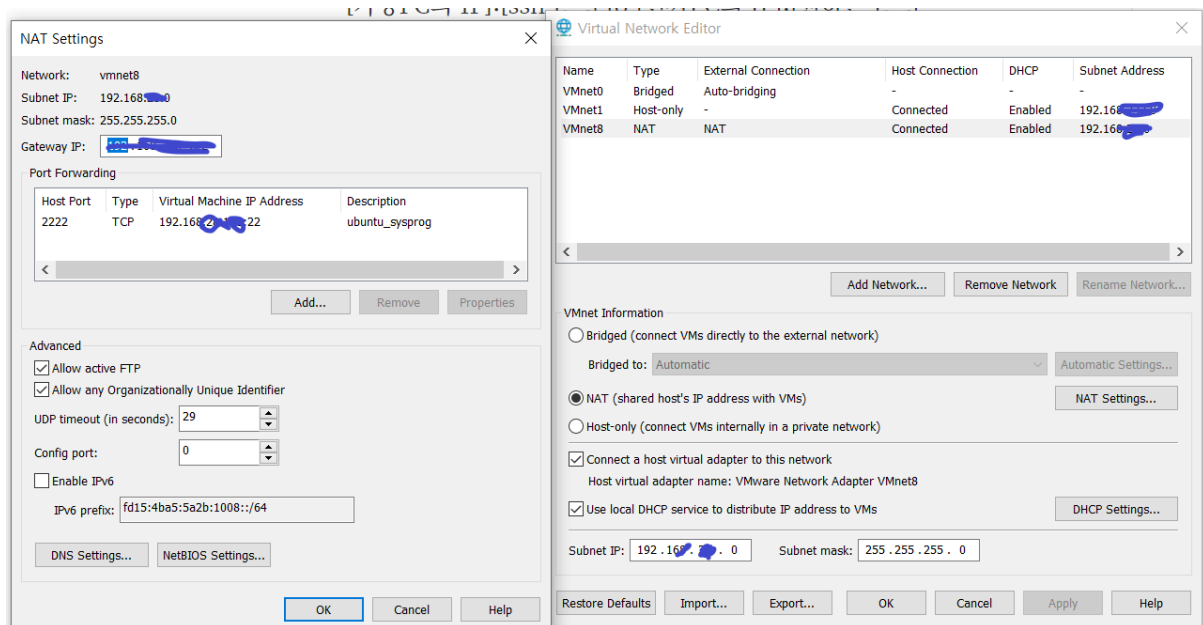
2. VMWare를 실행시켜 원격OS를 부팅한 뒤, Windows10에서 ipconfig 명령어를 치면 guest와 host OS 모두의 IP 주소를 얻을 수 있다. 이렇게 얻은 IP주소를 바탕으로 NAT 방식과 Bridged 방식으로 SSH 접속을 시도해본다.



### 1) NAT 방식

NAT 방식을 진행하려면 NAT setting을 해줘야하기 때문에 VMWare Player 버전을 사용중이라면 인터넷에서 VMWare Network Editor를 설치해야 한다.

(\* 저는 Pro 라이선스라 있기 때문에 건너뛰었습니다.)

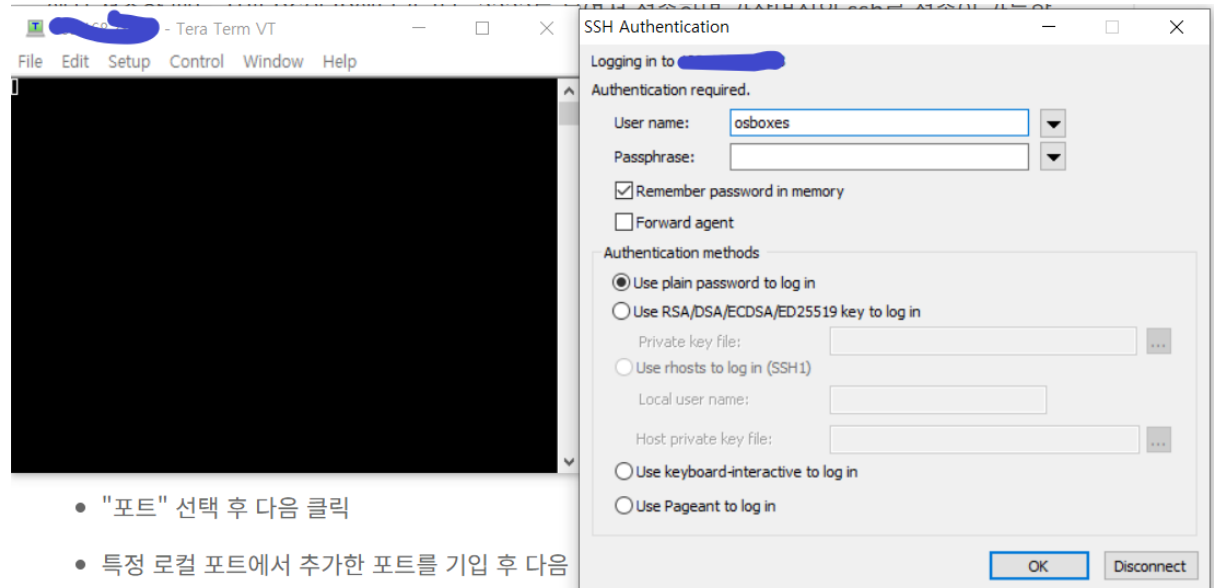


VMWare [Edit] – [Virtual network Editor]를 눌러보면 (우측에) 창이 하나 뜨면서 네트워크 종류를 나열하고 있다.

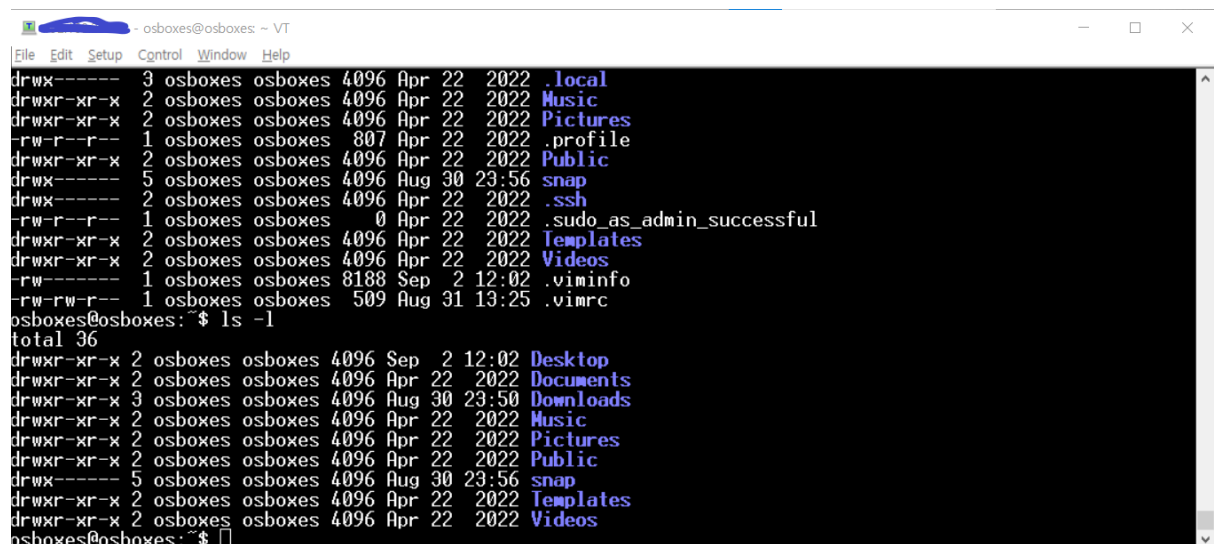
(이에 앞서 Bridged 네트워크 어댑터를 미리 추가해줬기 때문에 NAT, Bridged type이 있는 것을 확인)

이제 VMnet8을 누르고 NAT Settings 버튼을 눌러 관련 setting을 조절해준다.

위 세팅값은 로컬의 2222번 포트를 Virtual OS의 IP의 22번(ssh) 포트로 포워딩 해주겠다는 의미이다. 이렇게 세팅한 후 Host OS를 가동시킨 다음, 로컬의 ssh-client (teraterm) 프로그램을 켜준다.



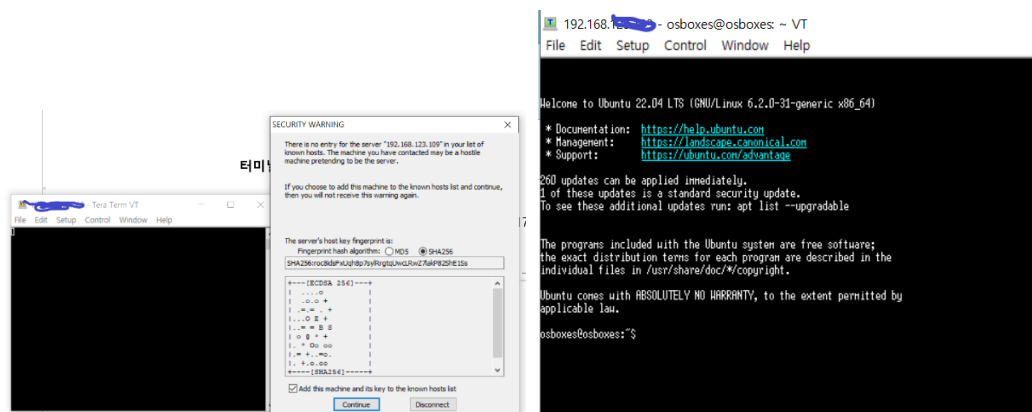
terra term에서 username과 (hostOS 시스템의 사용자 명칭, \*osboxes) passphrase(passwd)를 입력 후 OK 누르면 된다. 이제 왼쪽 검은 창이 원격의 터미널로 작동한다.



다양한 명령어를 실행시켜보고 VMWare에서 GUI로 확인해보면 좋다.

이 경우 그냥 로컬 PC와 마찬가지로 독립적인 IP를 사용하는 것이기 때문에 VMWare 자체적으로 설정해줄 필요가 없다.

따라서 terra-term에 그냥 bridged 방식의 IP주소를 입력해주면 별다른 절차없이 연결된다.



참고 자료:

[가상머신 우분투 SSH 원격 접속하기 \(tistory.com\)](https://tistory.com)

[\[개발환경\] 무료 터미널 테라 텀 \(Tera Term\) 설치 및 사용법 \(tistory.com\)](https://tistory.com)

기타 (과제 수행 중 어려웠던 점)

: ssh의 개념을 정립하고 적용하는 것이 낯설어서 두려움이 앞섰다. github ssh를 사용하려면, 우선 터미널에서 ssh 키를 생성하고 web에 private key를 등록시키는 과정이 있었다. client 프로그램에는 그런 과정이 생략된 듯 보였기 때문에 다소 둘이 서로 다르게 느껴졌다. (terra term에서 new connection을 만들 때 ssh key를 SHA 방식으로 암호화하는 과정이 있긴 했음.)