

## 실습 목표

1. 리눅스 (가상머신) 에서 spring\_9jwt(BE) 와 routerAndjwt(FE) 프로젝트를 실행한다.

- github 에 올릴 때 등 보안이 필요한 주요 환경변수는 .env 파일로 저장한다.

- 데이터베이스 접속은 다음 2가지 방법을 테스트 한다.

- 1 윈도우 오라클 데이터 베이스로 connect 하기

- 2 docker 컨테이너로 실행하는 오라클 데이터 베이스로 connect 하기

2. 리눅스 (가상머신)에서 BE,FE 프로젝트를 docker 컨테이너로 실행해 본다.

- docker 이미지를 만들기 위해 Dockerfile 로 만들고 docker 명령들을 실행한다.

- ↳ 프로젝트 직접 빌드 후에 Dockerfile 로 이미지 생성

- 데이터베이스 접속은 위 1, 2 번 모두 가능하다.

3. AWS 클라우드에서 실행되도록 한다.

- 로컬 환경이 아니므로 환경 변수 값, FE 의 요청 주소, cors 등을 수정하여 도커 이미지를 만든다.

- ↳ Dockerfile 은 프로젝트 빌드와 이미지 생성 작업을 멀티스테이지로 작성한다.

- docker hub 를 이용하여 만들어진 이미지를 AWS 에서 가져갈 수 있도록 한다.

- 리눅스 (가상머신) 에서 만든 이미지를 클라우드에서 도커 컨테이너로 실행한다.

- 데이터베이스는 AWS 클라우드의 RDS 서비스를 사용한다.

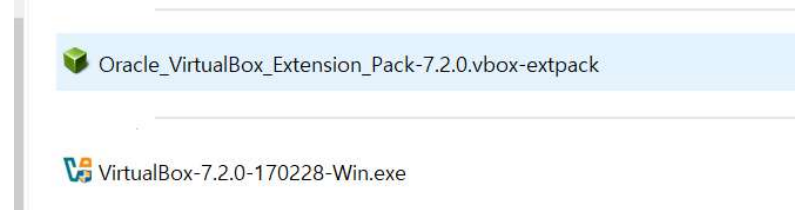
4. 리눅스 (가상머신) 에서 수정된 코드를 github main 브랜치에 push 하면 자동으로 이미지가 변경되고 컨테이너가 재실행 되는 작업이 AWS 클라우드에서 동작하도록 한다.

 최종 3,4 번을 이해하기 위해서 알아야할 내용이 많습니다

## 가상화 프로그램 설치

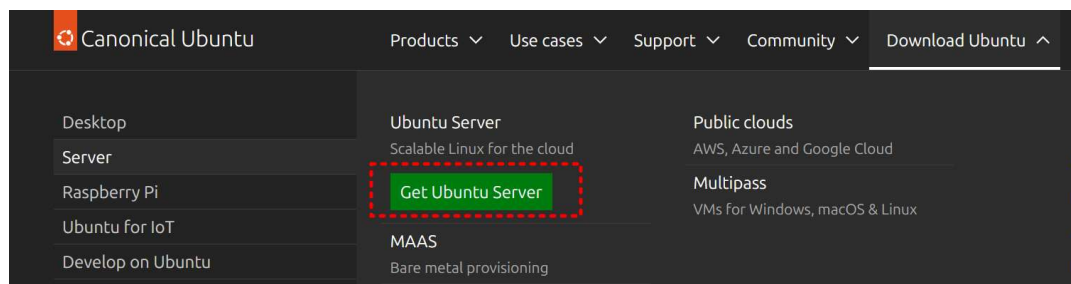
<https://www.virtualbox.org/wiki/Downloads>

👍 VirtualBox is a general-purpose full virtualization software for x86\_64 hardware.....

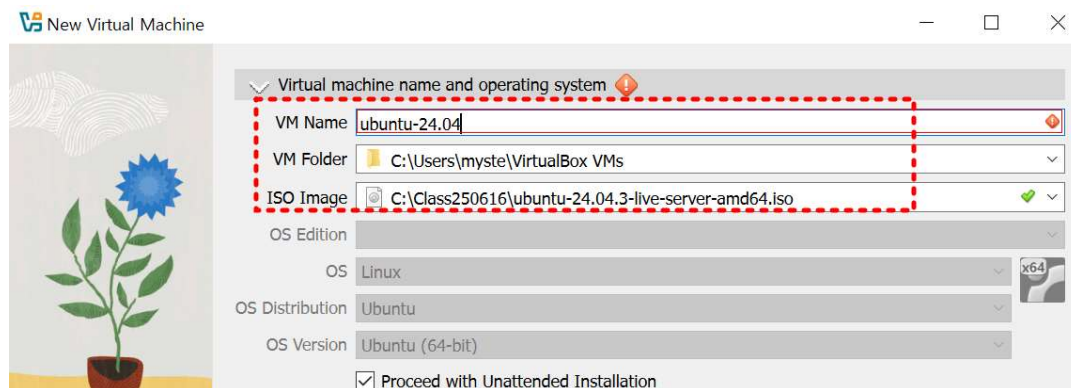


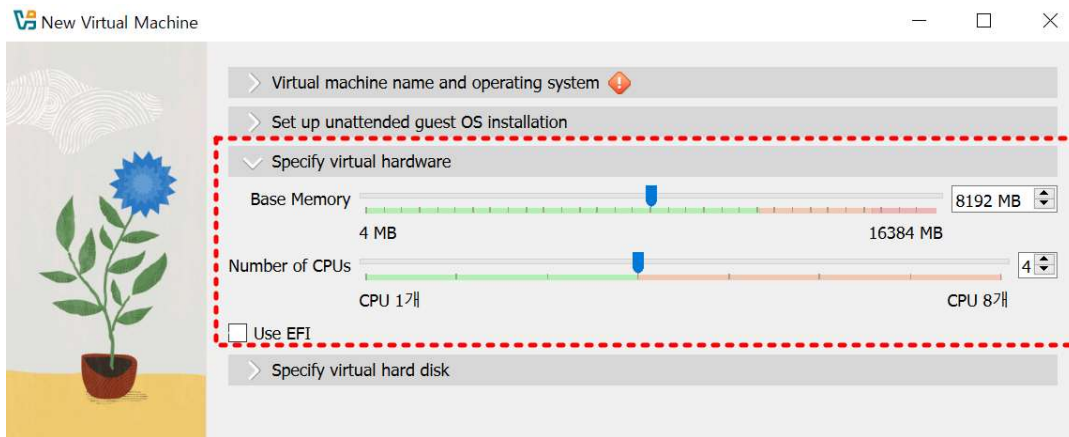
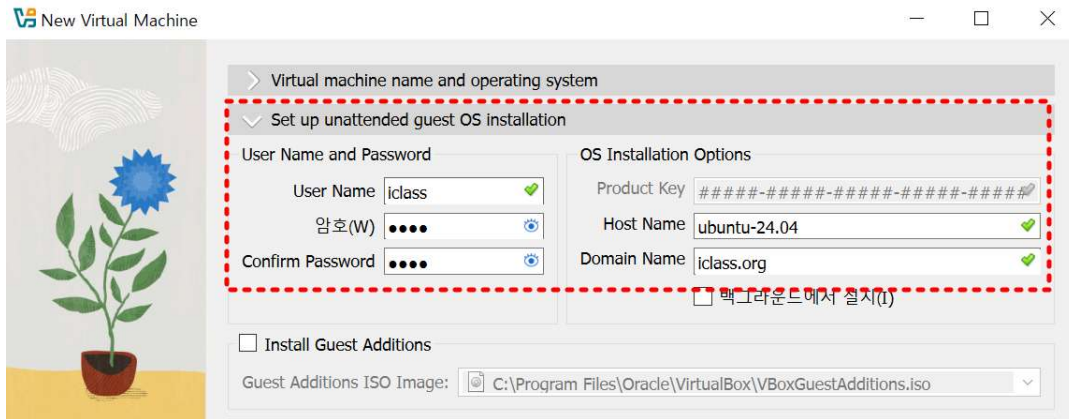
## 우분투 리눅스 다운로드

<https://ubuntu.com/download/>

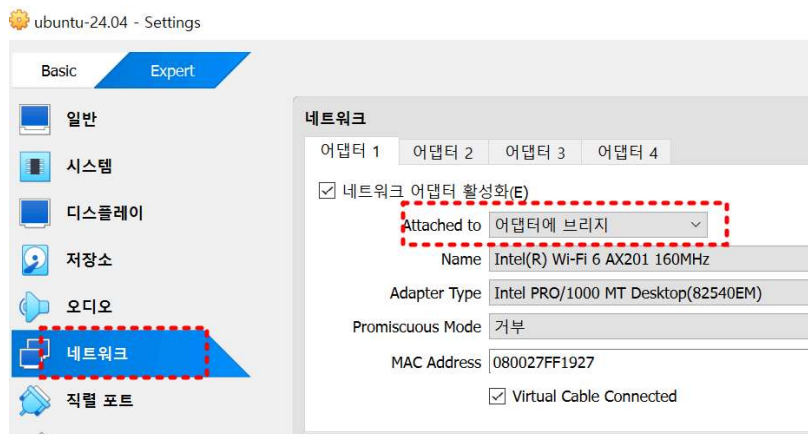


## 새 가상 머신 만들기





설치 후 재시동 → 머신 네트워크 설정



→ 이 설정은 다음 네트워크 관련 작업 후 변경할 예정임.

## 필요한 프로그램 설치 1 : 네트워크 명령어

```
$ sudo apt update
$ sudo apt install net-tools
```

## 고정 IP 설정하기

네트워크 인터페이스 이름 확인

```
$ ip addr show
```

예시:

```
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic enp0s3
```

설정 파일 열기

```
$ sudo nano /etc/netplan/01-netcfg.yaml
```

— 파일 내용 —

```
network:
  version: 2
  ethernets:
    enp0s3:      # 위의 이름을 작성
      dhcp4: no
      addresses:
        - 10.100.0.108/24
      routes:
        - to: default
          via: 10.100.0.1      # 게이트웨이 주소
      nameservers:
        addresses:
          - 8.8.8.8
          - 1.1.1.1
```

적용하기

```
$ sudo netplan apply
```

확인하기

```
$ ip route show      또는 ip addr show  이름
$ ifconfig
```

→ 고정 IP 설정 후 shutdown

## VirtualBox 네트워크 설정

The screenshot shows the 'Host-Only Network' tab in the VirtualBox Network settings. The 'Name' field is set to 'VirtualBox Host-Only Ethernet Adapter'. Under the 'Adapter (A)' section, 'DHCP Server (D)' is selected. The 'Manually set IP address (M)' radio button is selected, and the 'IPv4 Address (I)' is set to '10.100.0.108' and the 'IPv4 Subnet Mask (M)' is set to '255.255.255.0'. These fields are highlighted with a red dashed box.

The screenshot shows the 'NAT Network' tab in the VirtualBox Network settings. The 'Name' field is set to 'NatNetwork' and the 'IPv4 Subnet' is set to '10.100.0.0/24'. Under the 'General (G)' section, the 'Name (A)' is set to 'NatNetwork' and the 'IPv4 Subnet (A)' is set to '10.100.0.0/24'. The 'DHCP Enable (D)' checkbox is checked, and the 'IPv6 Enable (E)' checkbox is unchecked. These fields are highlighted with a red dashed box.

호스트 전용 네트워크		NAT 네트워크	클라우드 네트워크
이름	IPv4 접두사	IPv6 접두사	DHCP 서버
NatNetwork	10.100.0.0/24		사용 안 함

일반 옵션(G)		포트 포워딩(P)			
IPv4(4)	IPv6(6)				
이름	프로토콜	호스트 IP	호스트 포트	게스트 IP	게스트 포트
ssh	TCP	127.0.0.1	22	10.100.0.108	22

👉 호스트 포트는 22번 아니고 다른 예약되지 않은 포트 번호 사용 가능하나, VS CODE 가 임의로 사용하는 포트와 충돌 가능성이 있으므로 동일하게 예약된 포트 번호 사용합니다.

## VM 머신 네트워크 설정 변경

일반

시스템

디스플레이

저장소

오디오

네트워크

직렬 포트

네트워크

어댑터 1 어댑터 2 어댑터 3 어댑터 4

☒ 네트워크 어댑터 활성화(E)

Attached to

NAT 네트워크

Name

NatNetwork

Adapter Type

Intel PRO/1000 MT Desktop (82540EM)

Promiscuous Mode

거부

MAC Address

080027FF1927

☒ Virtual Cable Connected

→ 머신 시작 후 테스트

```
$ ping [호스트 IP]
```

필요한 프로그램 설치 2 : ssh

```
$ sudo apt install openssh-server
```

# 상태 확인

```
$ systemctl status ssh
```

# 실행 중이 아니면 시작

```
$ sudo systemctl start ssh
```

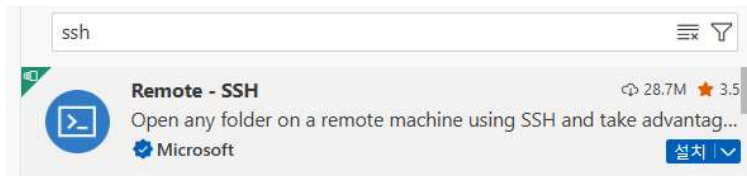
# 부팅 시 자동 시작

```
$ sudo systemctl enable ssh
```

## 원격 접속 SSH 클라이언트 프로그램

### 1. VS Code 활용

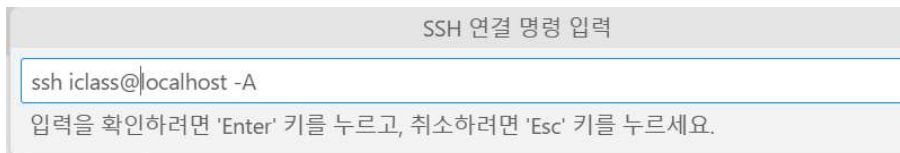
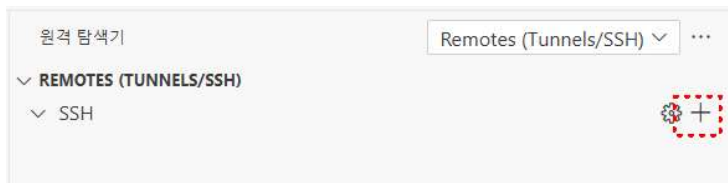
✓ 확장 프로그램 설치



→ 실행



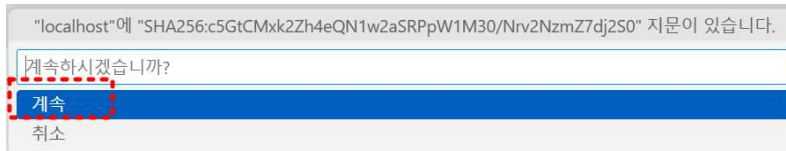
✓ 새로운 원격접속 추가



[구성 열기]



[현재 창에서 열기]



## 1. 전용 프로그램 Mobaterm

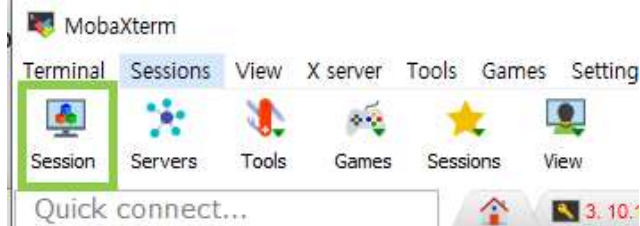
☺ 앞으로 리눅스 GUI 화면에서 작업하지 않습니다. 호스트 윈도우즈 컴퓨터에서 가상머신 리눅스로 원격접속하여 명령을 각 작업에 필요한 명령을 실행합니다.

Xshell 또는 Mobaterm , putty 등 많은 프로그램이 있으며 vs code , 인텔리제이 에서 원격 접속 확장 설치하며 사용할 수도 있습니다. (도커 및 프로그래밍 작업시 편의성)

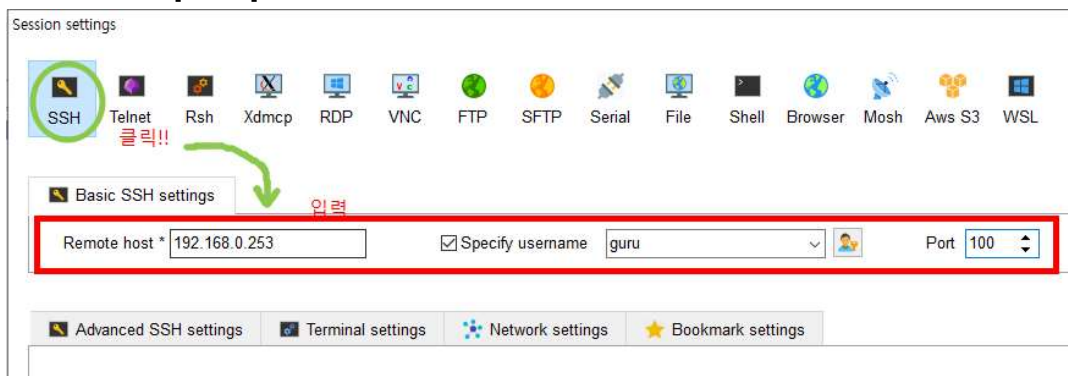
### 설치하기

- 다운로드 : MobaXterm\_Installer\_v25.1.zip 압축 풀고 설치
- 실행 : 접속 설정

[화면 1]

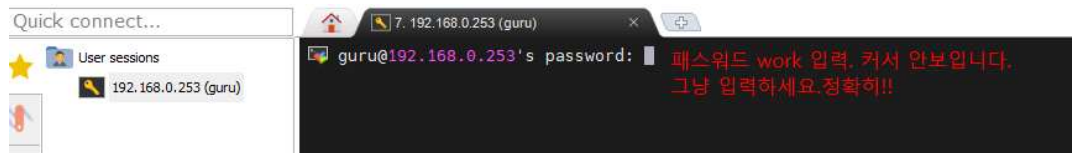


[화면 2]



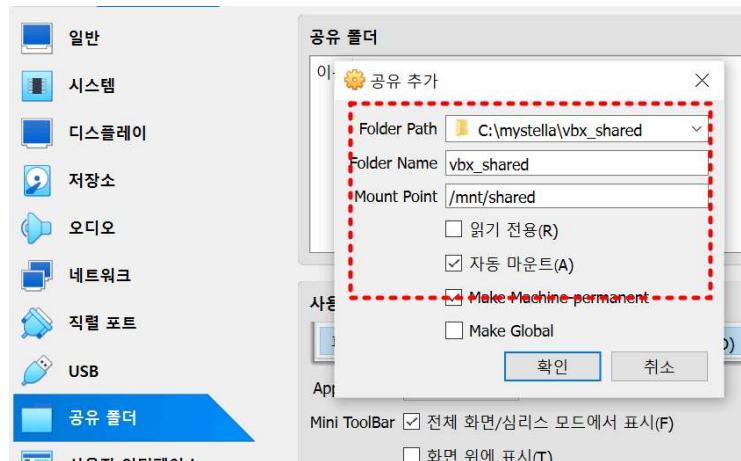
[화면 3]





## 호스트 컴퓨터와 공유 폴더 설정

[머신 폴더] 에 추가하기



👉 Folder Path 경로는 임의로 합니다.

```
$ sudo apt update
$ sudo apt install build-essential dkms linux-headers-$(uname -r) -y
$ sudo apt install virtualbox-guest-utils -y
```

→ 가상 머신 재시작

```
$ sudo reboot
```

권한 추가

```
$ sudo usermod -aG vboxsf $USER
```

수동 마운트

```
$ sudo mount -t vboxsf shared /mnt
```

확인

```
$ /mnt/shared
```

## 자바 개발 환경

# OpenJDK 21 설치 (LTS)

```
sudo apt install -y openjdk-21-jdk
```

# 확인

```
java -version
```

# JAVA\_HOME 환경 변수 설정(현재 세션에서만 유효)

```
export JAVA_HOME=/usr/lib/jvm/java-21-openjdk-amd64
```

```
export PATH=$JAVA_HOME/bin:$PATH
```

```
echo $JAVA_HOME
```

# JAVA\_HOME 환경 변수 설정(계속 유지하려면 홈디렉토리 설정 파일에 추가)

```
vi ~/.bashrc
```

➡ 파일이 열리면 위 2개 export 구문을 맨 아랫 줄에 추가/저장/터미널새로열기

# maven , gradle

```
sudo apt install -y maven gradle
```

# Node.js LTS (20.x) 설치

```
curl -fsSL https://deb.nodesource.com/setup_20.x | sudo -E bash -
```

```
sudo apt install -y nodejs
```

# 확인

```
node -v
```

```
npm -v
```

## 도커 설치하기

- 가상화\_도커 문서 참고하기

## 오라클

docker 컨테이너로 실행 :

이미지 [container-registry.oracle.com/database/express:21.3.0-xe](https://container-registry.oracle.com/database/express:21.3.0-xe) 또는

```
docker run -d \  
  --name oracle21xe \  
  -p 1521:1521 -p 5500:5500 \  
  -e ORACLE_PASSWORD=oracle \  
  --restart unless-stopped \  
  -e ORACLE_ALLOW_REMOTE=true \  
  -v oracle_data:/opt/oracle/oradata \  
  gvenzl/oracle-xe:21-slim
```

# 필요시 컨테이너 실행 중단

```
docker stop oracle21xe
```

# 필요시 컨테이너 삭제

```
docker rm oracle21xe
```

# 컨테이너 내부에서 직접 셸 명령어 실행하기

```
docker exec -it oracle21xe bash
```

```
sqlplus sys/oracle@XEPDB1 as sysdba
```

👉 윈도우(호스트) 오라클 계정에서 xepdb1 을 사용하려면

ALTER SESSION SET CONTAINER = XEPDB1; 으로 세션을 변경하고 c## 없이 계정만들기

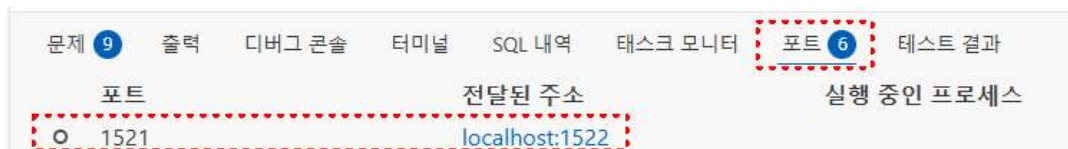
-- XEPDB1에서 사용자 생성 및 권한 부여 (slim 버전은 바로 XEPDB1 사용)

```
CREATE USER idev IDENTIFIED BY 1234  
  DEFAULT TABLESPACE USERS  
  TEMPORARY TABLESPACE TEMP;
```

```
GRANT CREATE SESSION TO idev ;  
GRANT CREATE TABLE, CREATE SEQUENCE, CREATE VIEW, CREATE PROCEDURE  
TO idev ;
```

GRANT UNLIMITED TABLESPACE TO idev ; -- 개발용, 테이블스페이스 제한 해제

🔥 퀴즈 : 아래와 같이 vscode 에서 설정한 포트 포워딩 의미는?



## Git 설치

```
sudo apt update  
sudo apt install git
```

```
git --version
```

```
git config --global user.name "kimssohee"  
git config --global user.email "koreaait.sec2020@gmail.com"
```

### ✅ 인증(로그인)

#### 방법 A: HTTPS + Personal Access Token (추천)

GitHub는 비밀번호 대신 **토큰 인증**을 사용합니다.

1. GitHub 웹사이트에서 Personal Access Token 생성
2. 토큰을 복사해두세요.
3. GitHub 저장소를 클론하거나 push할 때, 사용자명 입력 후 **비밀번호 대신 토큰**을 입력합니다.

```
bash  
git clone https://github.com/yourusername/yourrepo.git
```

#### 방법 B: SSH 키 인증 - 가상머신 ubuntu 와 호스트(윈도우)

1. SSH 키 생성:

```
ssh-keygen -t ed25519 -C "koreaait.sec2020@gmail.com"
```

2. 개인키/공개키 등록: (확인)

```
ls ~/.ssh  
# 공개키  
cat ~/.ssh/id_ed25519.pub
```

3. 공개키 값 설정

GitHub 접속 → SSH and GPG keys 설정 페이지

“New SSH key” 클릭 → 복사한 공개키를 붙여넣기

4. 연결 테스트:(GitHub 서버의 SSH 접속 주소)

```
ssh -T git@github.com
```

## 가상머신 우분투 인증방법 변경

위와 같이 만든 공개키와 개인키를 우분투 접속 할 때에도 사용하기  
단, 동일한 키를 사용할 수 있지만 github 보안을 위해 새로 만들어서 사용합니다.

🔑 개인키 관리는 매우 중요함.

## 가상 머신 작업

```
# 키 생성 : 키 위치 및 파일명 /home/iclass/.ssh/id_ed25519_ubuntu  
ssh-keygen -t ed25519 -C "myvbox-ubuntu"
```

```
# 공개 키 확인  
cat id_ed25519_ubuntu.pub
```

```
# 공개키 문자열을 복사해서 아래 파일에 추가하기(공개키 여러개 일때)  
vi ~/.ssh/authorized_keys
```

## 호스트 컴퓨터 작업

- 1) 윈도우 호스트 머신에 개인키 복사 하여 저장해 놓기  
개인키는 같은 파일명에 .pub 확장자가 없는 것.  
폴더 : C:\Users\Class01\.ssh

- 2) 윈도우 호스트 머신 ssh 설정 파일 변경하기

config 설정 파일에 아래 추가

```
IdentityFile C:/Users/Class01/.ssh/id_ed25519_ubuntu
```