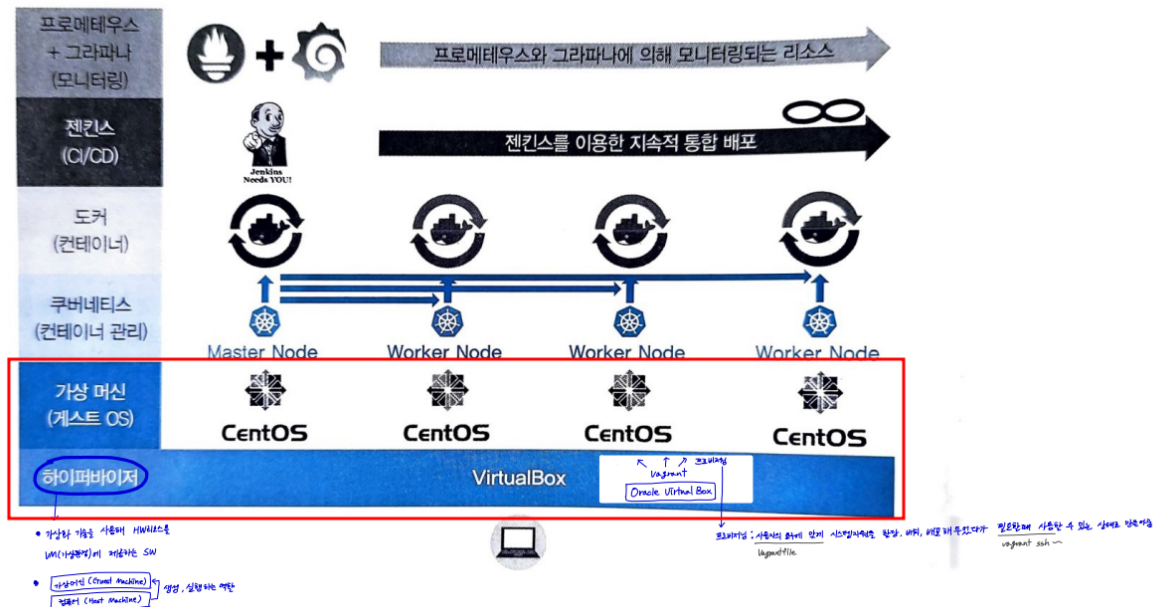


<https://techblog.woowahan.com/2562/>

▼ 그림 1-8 실습용 컨테이너 인프라 환경 구성



## 2.1 테스트 환경을 자동으로 구성하는 도구

### 2.1.1 버추얼박스 설치하기

### 2.1.2 베이그런트 설치하기

### 2.1.3 베이그런트 구성하고 테스트하기

## 자주 사용하는 베이그런트 명령

1. 프로비저닝에 필요한 기본 코드 생성
2. c:/HashiCorp/Vagrantfile을 열고 config.vm.box = "base" 존재 확인
3. vagrant up으로 프로비저닝 진행 → 에러
4. 에러가 발생하지 않게 설치할 운영체제 이미지 선택
5. c:/HashiCorp/Vagrantfile 수정
6. 다시 vagrant up
7. 버추얼박스를 실행해 VM이 제대로 생성되었는지 확인
8. vagrant ssh를 실행해 CentOS에 접속
9. CentOS의 실행시간, os 종류를 확인
10. 가상머신 삭제 exit, vagrant destroy -f

## 2.2 베이그런트로 테스트 환경 구축하기

### 2.2.1 가상 머신에 필요한 설정 자동으로 구성하기

- ### 1. 코드 입력

### 2.2.2 가상머신에 추가 패키지 설치하기

1. 코드 입력
2. 코드 실행

### 2.2.3 가상 머신 추가로 구성하기

1. 코드 입력
2. 코드 실행

## 2.3 터미널 프로그램으로 가상 머신 접속하기

### 2.3.1 푸티 설치하기

### 2.3.2 슈퍼푸티 설치하기

### 2.3.3 슈퍼푸티로 다수의 가상머신 접속하기

## 학습목표

일관성 있는 결과를 얻으려면 프로젝트 환경이 가능한 한 일정하게 생성되고 유지되어야 합니다.

그래서 여기서는 **코드형 인프라로 인프라 환경을 일정하게 유지하고 구성합니다.**


### **코드형 인프라 (IaC, Infrastructure as Code)란?**

하드웨어를 설정하고, 운영체제를 설치하고, 네트워크를 구성하고, 개발 환경을 구축하는 것  
즉, 코드로 인프라를 소프트웨어처럼 다룰 수 있음

이렇게 설치된 환경은 사용자가 모두 동일한 환경에서 테스트할 수 있고, 문제가 발생했을 때 몇번의 명령 실행만으로 환경을 다시 새것처럼 구성할 수 있다는 장점이 있다!


## 2.1 테스트 환경을 자동으로 구성하는 도구

### 2.1.1 버추얼박스 설치하기

 **버추얼박스의 역할** : 베이그런트가 인프라를 생성하는 데 기반이 되는 환경을 제공

Virtual Box Download > <https://www.virtualbox.org/wiki/Downloads>

### 2.1.2 베이그런트 설치하기

 **베이그런트의 역할** : 사용자의 요구에 맞게 시스템 자원을 할당, 배치, 배포해두었다가 필요할 때 시스템을 사용할 수 있는 상태로 만들어 줌 (프로비저닝, provisioning)

프로비저닝을 하면 필요할 때 환경을 매우 쉽고 간단하게 구현할 수 있다.

Vagrant Download > <https://developer.hashicorp.com/vagrant/downloads>

### 2.1.3 베이그런트 구성하고 테스트하기

#### 목표

- 테스트 환경을 구성하기 전에, 설치된 도구가 정상적으로 작동하는지 확인해보자!

**1** 프로비저닝을 위한 코드 작성

**2** 이를 베이그런트에서 불러오기

**3** 버추얼박스에 운영체제 설치

#### 자주 사용하는 베이그런트 명령



베이그런트에는 프로비저닝에 필요한 다양한 명령이 있다.

`vagrant init` - 프로비저닝을 위한 기초 파일을 생성  
`vagrant up` - Vagrantfile을 읽어 들여 프로비저닝을 진행  
`vagrant halt` - 베이그런트에서 다루는 VM 종료  
`vagrant destroy` - 베이그런트에서 관리하는 VM 삭제  
`vagrant ssh` - 베이그런트에서 관리하는 VM에 ssh로 접속  
`vagrant provision` - 베이그런트에서 관리하는 VM에 변경된 설정 적용

## 1. 프로비저닝에 필요한 기본 코드 생성

```
C:\Users\User>cd c:/HashiCorp

c:\HashiCorp>vagrant init
A `Vagrantfile` has been placed in this directory. You are now
ready to `vagrant up` your first virtual environment! Please read
the comments in the Vagrantfile as well as documentation on
`vagrantup.com` for more information on using Vagrant.
```

## 2. c:/HashiCorp/Vagrantfile을 열고 config.vm.box = "base" 존재 확인

```
File Edit Selection View Go Run Terminal Help
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

Vagrantfile x
C: > HashiCorp > Vagrantfile
1 # -*- mode: ruby -*-
2 # vi: set ft=ruby :
3
4 # All Vagrant configuration is done below. The "2" in Vagrant.configure
5 # configures the configuration version (we support older styles for
6 # backwards compatibility). Please don't change it unless you know what
7 # you're doing.
8 Vagrant.configure("2") do |config|
9   # The most common configuration options are documented and commented below.
10  # For a complete reference, please see the online documentation at
11  # https://docs.vagrantup.com.
12
13  # Every Vagrant development environment requires a box. You can search for
14  # boxes at https://vagrantcloud.com/search.
15  config.vm.box = "base"
16
17  # Disable automatic box update checking. If you disable this, then
18  # boxes will only be checked for updates when the user runs
19  # `vagrant box outdated`. This is not recommended.
20  # config.vm.box_check_update = false
```

## 3. vagrant up으로 프로비저닝 진행 → 에러

```
c:\HashiCorp>vagrant up

Bringing machine 'default' up with 'virtualbox' provider...
==> default: Box 'base' could not be found. Attempting to find and install...
default: Box Provider: virtualbox
default: Box Version: >= 0
==> default: Box file was not detected as metadata. Adding it directly...
==> default: Adding box 'base' (v0) for provider: virtualbox
default: Downloading: base
default:
An error occurred while downloading the remote file. The error
message, if any, is reproduced below. Please fix this error and try
```

```
again.
```

```
Couldn't open file c:/HashiCorp/base
```

에러 이유 - 설치하려는 이미지가 'base'로 명시되어 있으나 베이그런트가 해당 이미지를 찾지 못해서 발생

#### 4. 에러가 발생하지 않게 설치할 운영체제 이미지 선택

저자가 올려놓은 쿠버네티스 실습에 필요한 설정 변경 사항 > <https://app.vagrantup.com/sysnet4ad>

#### 5. c:/HashiCorp/Vagrantfile 수정

```
config.vm.box = "base" 를  
config.vm.box = "sysnet4admin/CentOS-k8s" 로 변경한다.
```

#### 6. 다시 vagrant up

▼ cmd에 vagrant up 명령어를 입력한다.

```
c:\HashiCorp>vagrant up  
Bringing machine 'default' up with 'virtualbox' provider...  
==> default: Box 'sysnet4admin/CentOS-k8s' could not be found. Attempting to find and install...  
default: Box Provider: virtualbox  
default: Box Version: >= 0  
==> default: Loading metadata for box 'sysnet4admin/CentOS-k8s'  
default: URL: https://vagrantcloud.com/sysnet4admin/CentOS-k8s  
==> default: Adding box 'sysnet4admin/CentOS-k8s' (v0.7.4) for provider: virtualbox  
default: Downloading: https://vagrantcloud.com/sysnet4admin/boxes/CentOS-k8s/versions/0.7.4/providers/virtualbox.box  
default:  
==> default: Successfully added box 'sysnet4admin/CentOS-k8s' (v0.7.4) for 'virtualbox'!  
==> default: Importing base box 'sysnet4admin/CentOS-k8s'...  
==> default: Matching MAC address for NAT networking...  
==> default: Checking if box 'sysnet4admin/CentOS-k8s' version '0.7.4' is up to date...  
==> default: Setting the name of the VM: HashiCorp_default_1686399533031_72528  
Vagrant is currently configured to create VirtualBox synced folders with  
the 'SharedFoldersEnableSymlinksCreate' option enabled. If the Vagrant  
guest is not trusted, you may want to disable this option. For more  
information on this option, please refer to the VirtualBox manual:  
  
https://www.virtualbox.org/manual/ch04.html#sharedfolders  
  
This option can be disabled globally with an environment variable:  
  
VAGRANT_DISABLE_VBOXSYMLINKCREATE=1  
  
or on a per folder basis within the Vagrantfile:  
  
config.vm.synced_folder '/host/path', '/guest/path', SharedFoldersEnableSymlinksCreate: false  
==> default: Clearing any previously set network interfaces...  
==> default: Preparing network interfaces based on configuration...  
default: Adapter 1: nat  
==> default: Forwarding ports...  
default: 22 (guest) => 2222 (host) (adapter 1)  
==> default: Booting VM...
```

#### 7. 버추얼박스를 실행해 VM이 제대로 생성되었는지 확인



## 8. vagrant ssh를 실행해 CentOS에 접속

```
c:\HashiCorp>vagrant ssh
[vagrant@k8s ~]$
```

## 9. CentOS의 실행시간, os 종류를 확인

```
[vagrant@k8s ~]$ uptime
21:22:37 up 3 min, 1 user, load average: 0.03, 0.08, 0.05

[vagrant@k8s ~]$ cat /etc/redhat-release
CentOS Linux release 7.8.2003 (Core)
```

## 10. 가상머신 삭제 exit, vagrant destroy -f

```
[vagrant@k8s ~]$ exit
logout
Connection to 127.0.0.1 closed.

c:\HashiCorp>vagrant destroy -f
==> default: Forcing shutdown of VM...
==> default: Destroying VM and associated drives...
```

## 2.2 베이그런트로 테스트 환경 구축하기

### 목표

- Vagrantfile을 수정해 원하는 구성이 자동으로 CentOS에 입력되도록 해보자!

### 2.2.1 가상 머신에 필요한 설정 자동으로 구성하기

#### 1. 코드 입력

##### ▼ 코드

```
# -*- mode: ruby -*-
# vi: set ft=ruby :
Vagrant.configure("2") do |config|
  config.vm.define "m-k8s" do |cfg|
    cfg.vm.box = "sysnet4admin/CentOS-k8s"
    cfg.vm.provider "virtualbox" do |vb|
      vb.name = "m-k8s(github_SysNet4Admin)"
      vb.cpus = 2
      vb.memory = 2048
      vb.customize ["modifyvm", :id, "--groups", "/k8s-SM(github_SysNet4Admin)"]
    end
    cfg.vm.host_name = "m-k8s"
    cfg.vm.network "private_network", ip: "192.168.1.10"
    cfg.vm.network "forwarded_port", guest: 22, host: 60010, auto_correct: true, id: "ssh"
    cfg.vm.synced_folder "../data", "/vagrant", disabled: true
  end
end
```

The screenshot shows the Vagrantfile code with several handwritten annotations in Korean:

- API 버전**: Points to the `config` block.
- 가상 머신 설정 시작!**: Points to the `config.vm.define` block.
- Virtual box 설정 시작!**: Points to the `cfg.vm.provider` block.
- 이름과 메모리**: Points to `vb.name` and `vb.memory`.
- 가상 머신에 설정한 VM의 이름, CPU, 메모리, 디스크 크기**: Points to the `vb` block.
- 호스트 이름 설정**: Points to `cfg.vm.host_name`.
- 호스트 IP 설정**: Points to `cfg.vm.network "private_network"`.
- 호스트 포트 설정**: Points to `cfg.vm.network "forwarded_port"`.
- 호스트 디렉토리 설정**: Points to `cfg.vm.synced_folder`.
- 호스트 그룹 설정**: Points to `vb.customize`.
- 호스트 그룹 설정**: Points to `vb.customize`.
- 호스트 그룹 설정**: Points to `vb.customize`.

Additional notes at the bottom right:

- SSH 통신은 호스트 60010에서 게스트 22로 연결되도록 구성
- 포트 번호를 대비한 설정.
- 포트 번호의 모든 자동 변경

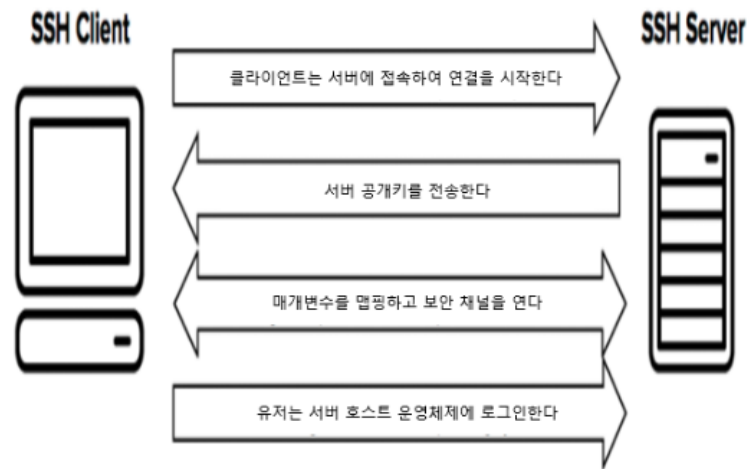
### 질문!

#### ▼ ssh 통신이란 뭘까?

##### SSH(Secure Shell)

- 원격 컴퓨터와 안전하게 통신하기 위한 프로토콜 및 암호화 기술
- SSH는 네트워크를 통해 안전한 원격 접속 및 데이터 통신을 제공
- SSH를 사용하면 클라이언트 컴퓨터(로컬 컴퓨터)와 서버 컴퓨터(원격 컴퓨터) 간에 암호화된 연결 설정 가능 (데이터 전송 과정에서 제3자가 데이터를 엿볼 수 없도록 보안을 제공)

- 동작 방식



- SSH 키는 공개키와 비공개키로 이루어짐
- 비공개키는 로컬 머신(SSH Client)에 위치
- 공개키는 리모트 머신(SSH Server)에 위치
- SSH 접속을 시도하면 SSH Client가 로컬 머신의 비공개키와 원격 머신의 비공개키를 비교해서 둘이 일치하는지 확인

#### ▼ 왜 호스트와 가상 머신 사이에 디렉터리 동기화가 이뤄지지 않아야해?

- 호스트와 가상 머신간에 디렉터리를 동기화 할 때, 호스트의 특정 디렉터리에 있는 파일이 가상 머신에 공유되므로 보안에 취약할 수 있기 때문이다.
- 생각해보니까 내 노트북에 있는 내용을 내가 만든 CentOS에 굳이 동기화 시킬 이유가 없다!! okok 이해했음

## 2. 코드 실행

```
vagrant up

생성된 가상머신(CentOS) 접속
vagrant ssh

CentOS에서 IP설정이 제대로 설정되었는지 확인
ip addr show eth1

CentOS 접속 종료
exit
```

### 2.2.2 가상머신에 추가 패키지 설치하기

#### 1. 코드 입력

##### ▼ 코드

```
# -*- mode: ruby -*-
# vi: set ft=ruby :
Vagrant.configure("2") do |config|
  config.vm.define "m-k8s" do |cfg|
    cfg.vm.box = "sysnet4admin/CentOS-k8s"
    cfg.vm.provider "virtualbox" do |vb|
```

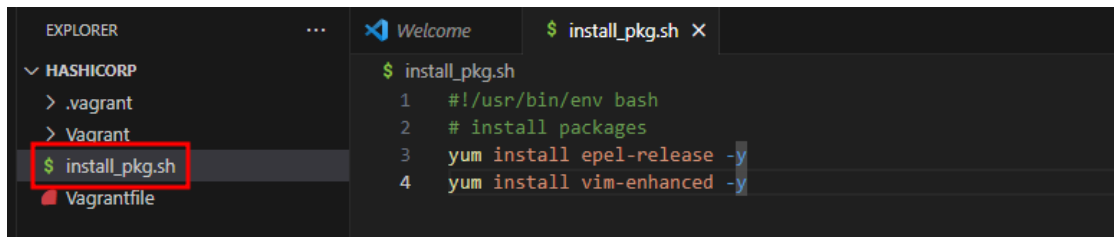
```

vb.name = "m-k8s(github_SysNet4Admin)"
vb.cpus = 2
vb.memory = 2048
vb.customize ["modifyvm", :id, "--groups", "/k8s-SM(github_SysNet4Admin)"]
end
cfg.vm.host_name = "m-k8s"
cfg.vm.network "private_network", ip: "192.168.1.10"
cfg.vm.network "forwarded_port", guest: 22, host: 60010, auto_correct: true, id: "ssh"
cfg.vm.synced_folder "../data", "/vagrant", disabled: true
cfg.vm.provision "shell", path: "install_pkg.sh" # add provisioning script
end
end

```

추가 코드 - `cfg.vm.provision "shell", path: "install_pkg.sh" # add provisioning script`

vm.provision "shell" 구문으로 경로(path)에 있는 install\_pkg.sh를 게스트(CentOS)내부에서 호출해 실행되도록 합니다.



#### ▼ 코드

```

#!/usr/bin/env bash
# install packages
yum install epel-release -y
yum install vim-enhanced -y

```

Vagrantfile에서 호출한 install\_pkg.sh로 입력해 둔 배시 셸 파일을 실행해

- EPEL(Extra Packages for Enterprise Linux) 저장소
- 코드 하이라이트를 위한 Vim

의 추가 기능을 설치한다.

## 2. 코드 실행

```

추가한 프로비전 구문 실행
vagrant provision

CentOS에 접속
vagrant ssh

EPEL저장소가 구성됐는지 확인
yum repolist

문법 하이라이트가 적용되었는지 확인
vi .bashrc

vi 프로그램 종료
:q

가상머신 종료
exit

가상머신 삭제
vagrant destroy -f

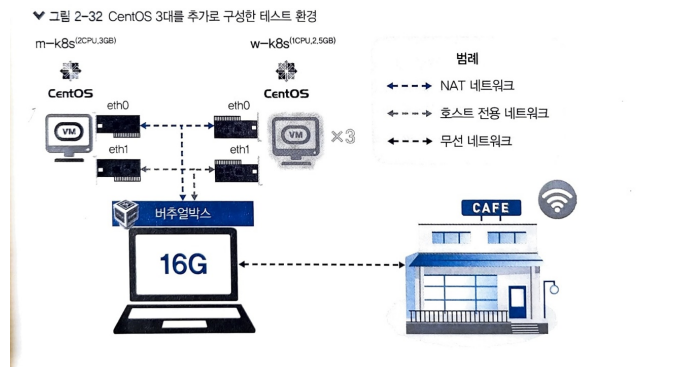
```



## 2.2.3 가상 머신 추가로 구성하기

### 목표

- 베이그런트로 os를 자동으로 설치하고 구성하면 편하지만 단순히 os 1개를 위해서 베이그런트를 쓰는 것은 아니다.
- 기존에 설치한 VM외에 **VM 3대를 추가로 설치하자!**
- 기존 VM과 추가한 VM간에 **네트워크 통신이 원활하게 작동하는지 확인하자!**



## 1. 코드 입력

### ▼ Vagrantfile

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

Vagrant.configure("2") do |config|
  config.vm.define "m-k8s" do |cfg|
    cfg.vm.box = "sysnet4admin/CentOS-k8s"
    cfg.vm.provider "virtualbox" do |vb|
      vb.name = "m-k8s(github_SysNet4Admin)"
      vb.cpus = 2
      vb.memory = 2048
      vb.customize ["modifyvm", :id, "--groups", "/k8s-SM(github_SysNet4Admin)"]
    end
    cfg.vm.host_name = "m-k8s"
    cfg.vm.network "private_network", ip: "192.168.1.10"
    cfg.vm.network "forwarded_port", guest: 22, host: 60010, auto_correct: true, id: "ssh"
    cfg.vm.synced_folder "../data", "/vagrant", disabled: true
    cfg.vm.provision "shell", path: "install_pkg.sh" # add provisioning script
    cfg.vm.provision "file", source: "ping_2_nds.sh", destination: "ping_2_nds.sh"
    cfg.vm.provision "shell", path: "config.sh"
  end

  #####
  # Added Nodes #
  #####

  (1..3).each do |i| # 1부터 3까지의 인자를 반복해 i로 입력
    config.vm.define "w#{i}-k8s" do |cfg| # {i} 값이 1, 2, 3으로 차례대로 치환됨
      cfg.vm.box = "sysnet4admin/CentOS-k8s"
      cfg.vm.provider "virtualbox" do |vb|
        vb.name = "w#{i}-k8s(github_SysNet4Admin)"
        vb.cpus = 1
        vb.memory = 1020 # 메모리를 1GB 사용하도록 변경
        vb.customize ["modifyvm", :id, "--groups", "/k8s-SM(github_SysNet4Admin)"]
      end
      cfg.vm.host_name = "w#{i}-k8s"
      cfg.vm.network "private_network", ip: "192.168.1.10#{i}"
      cfg.vm.network "forwarded_port", guest: 22, host: "6010#{i}", auto_correct: true, id: "ssh"
      cfg.vm.synced_folder "../data", "/vagrant", disabled: true
      cfg.vm.provision "shell", path: "install_pkg.sh"
    end
  end
end
```

```
end
end
end
```

#### ▼ install\_pkg.sh

```
#!/usr/bin/env bash
# install packages
yum install epel-release -y
yum install vim-enhanced -y
```

#### ▼ ping\_3\_nds.sh

```
# ping 3 times per nodes
ping 192.168.1.101 -c 3
ping 192.168.1.102 -c 3
ping 192.168.1.103 -c 3
```

추가로 설치한 CentOS 3대로 ping을 보내 네트워크가 제대로 작동하는지 확인하는 명령

#### ▼ config.sh

```
#!/usr/bin/env bash
# modify permission
chmod 744 ./ping_2_nds.sh
```

## 2. 코드 실행

```
4대의 CentOS 설치 및 구성
vagrant up

CentOS 접속. 설치된 VM이 여러대라서 가상 머신의 이름을 입력해야함
vagrant ssh

가장 먼저 설치된 가상 머신 m-k8s에 접속
vagrant ssh m-k8s

ping_2_nds.sh 파일을 실행해 3대의 CentOS와 통신하는데 문제없는지 확인
./ping_2_nds.sh

확인했으니 가상머신 접속 종료
exit
```

## 2.3 터미널 프로그램으로 가상 머신 접속하기

### 🚩 목표

- 여러대의 VM에 한번에 접속하도록 구성하기!

### 2.3.1 푸티 설치하기

푸티 다운로드 > <https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html>

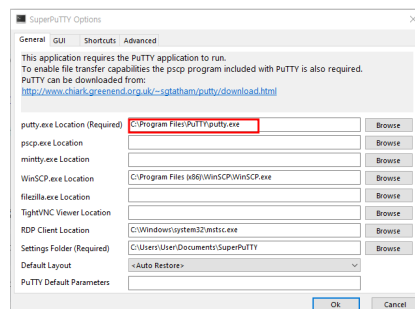
- 푸티로 여러 대 가상 머신에 접근할 수 있지만, 한 번에 한 대씩만 접근 가능

## 2.3.2 슈퍼푸티 설치하기

- 푸티를 단독으로 사용하면 창을 여러 개 띄워야 하므로 명령내리기 매우 번거롭다.
- 슈퍼푸티를 사용하면? ⇒ 푸티의 제약사항이 해결됨!

슈퍼푸티 다운로드 > <https://github.com/jimradford/superputty/releases>

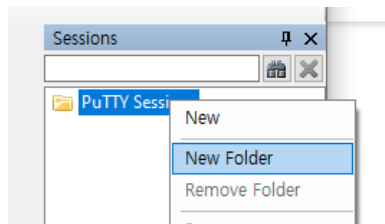
- SuperPuttySetup-1.5.0.0.msi 를 다운로드해서 실행하자
- 슈퍼푸티는 푸티를 통해 실행되므로 푸티의 위치를 지정해야 한다. 위치를 지정하자



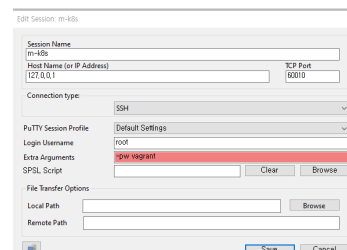
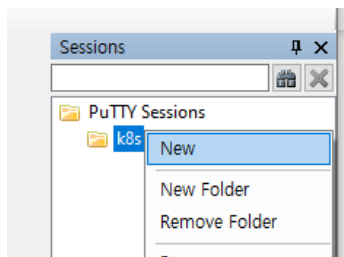
- 슈퍼푸티의 정상실행을 확인하자

## 2.3.3 슈퍼푸티로 다수의 가상머신 접속하기

- 슈퍼푸티로 가상머신 4대(m-k8s, w1-k8s, w2-k8s, w3-k8s)에 접속하자
- 반복적으로 사용할 가상머신의 접속 정보부터 슈퍼 푸티에 구성하자
- 새로운 세션 디렉토리 생성 (k8s)



- m-k8s의 접속 정보 입력

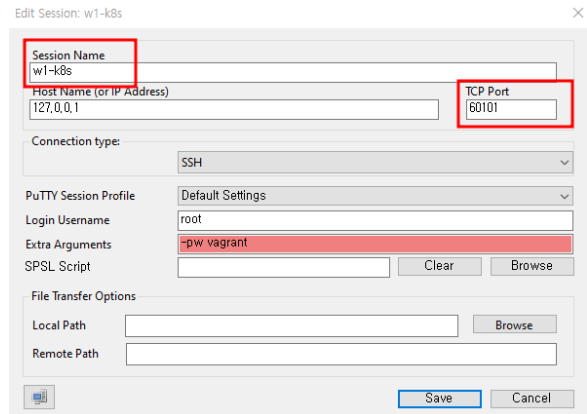
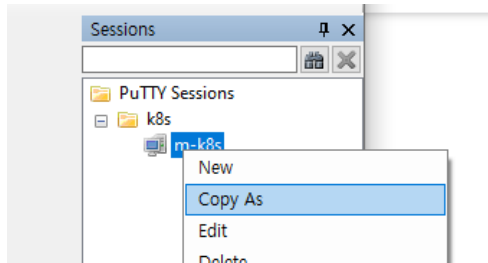


- ▼ 127.0.0.1로 접속하는 이유?

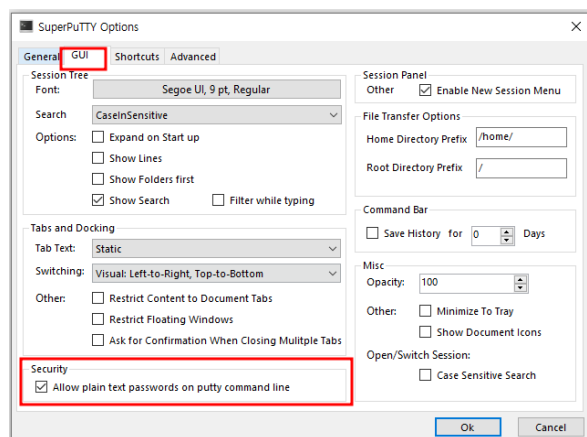
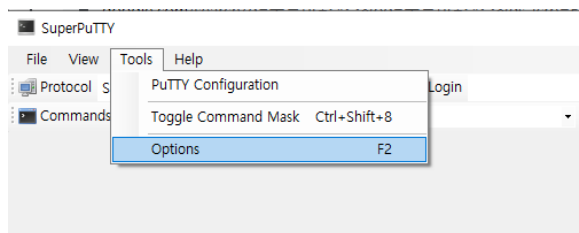
현재 가상 머신들은 192.168.1.0/24 영역대에 있어서 대부분의 경우 모두 접속할 수 있다

하지만 현업에서는 데이터 통신과 관리 네트워크를 분리해 사용하는데, 이와 비슷하게 관리 네트워크를 분리한 것을 보면 된다.

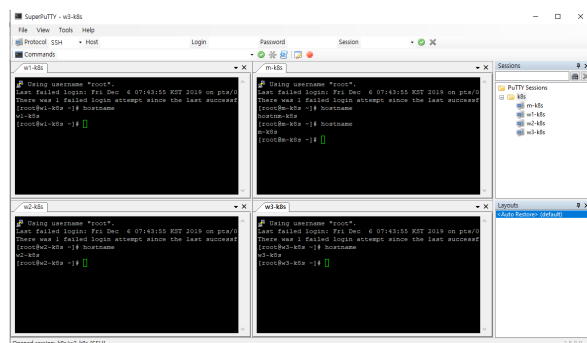
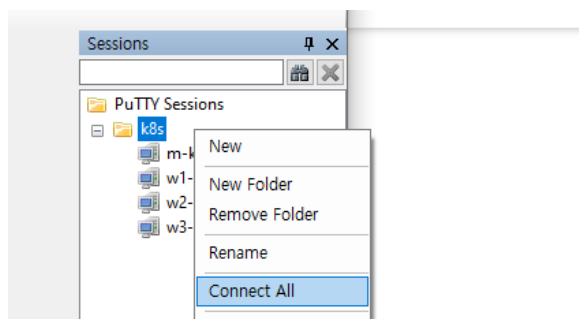
- w1-k8s, w2-k8s, w3-k8s 접속 정보 추가로 생성



- 평문으로 접속하기 위해 슈퍼푸티의 보안 설정 변경



- 모든 가상 머신에 한번에 접속



- vagrant destroy -f

