

# Advanced Kubernetes

**v0.2.1**

2022-01-21

written by A.K.A whatwant ([whatwant@whatwant.com](mailto:whatwant@whatwant.com))

# 두번째 뵙겠습니다 ?!

▷ Camera 켜고, Mic 켜고 ~ 우리 수다 떨어요!

- 수업 중에도 말 많이 하자구요!!! ^^

▷ 출석 체크도 한 번 해보시면 어떠세요?!

- <https://modulabs.co.kr/>

- 모두연 홈페이지 → 로그인 → 마이페이지 → 참여한 랩·풀잎 → 자세히 보기 → 내 풀잎스쿨 출석 확인하기

**2<sup>nd</sup>**  
**Week**

# Agenda

- ▷ [10m] make-friendship
- ▷ [40m] kubernetes Install
- ▷ [10m] break
- ▷ [40m] Flip - Pods & Namespaces
- ▷ [20m] Wrap-Up
- ▷ [10m] break
- ▷ [20m] Hands-On
- ▷ [10m] Q&A
- ▷ [20m] Quiz

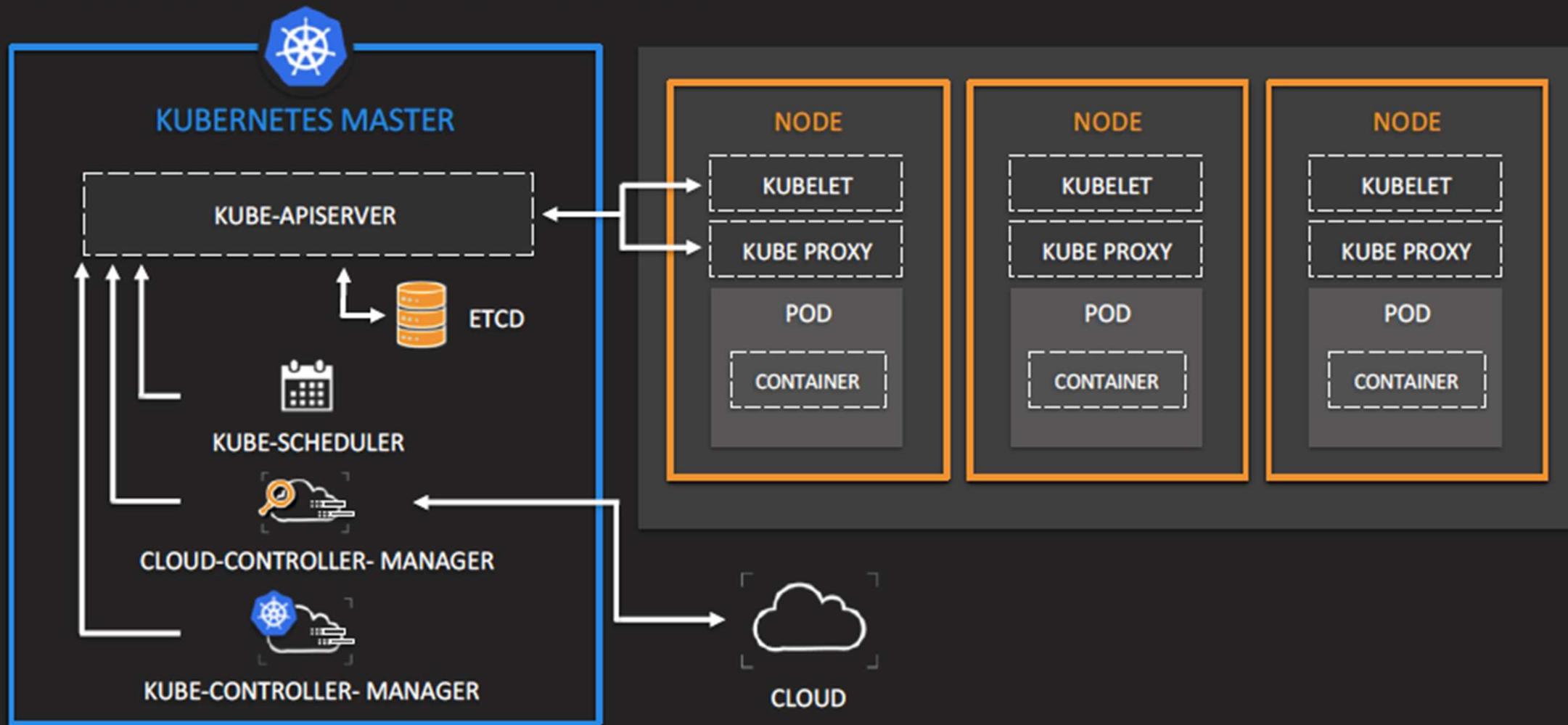
# 잡담 & 지난 수업 관련 이야기

and `2<sup>nd</sup> 자기 소개`



# **Kubernetes Architecture**

# BASIC KUBERNETES ARCHITECTURE



※ 참고 : <https://techoopla.wordpress.com/2018/05/24/notes-on-kubernetes-api-primitives-and-cluster-architecture/>



# Kubernetes Install

# Keywords

▷ IaC (Infrastructure as Code, 코드형 인프라)

▷ Vagrant

- Vagrantfile
- Vagrant Cloud (Hub)

▷ Kubespray

▷ SSH-Key

▷ VirtualBox

- Host
- Guest

▷ Node

- Master
- Worker

▷ Ansible

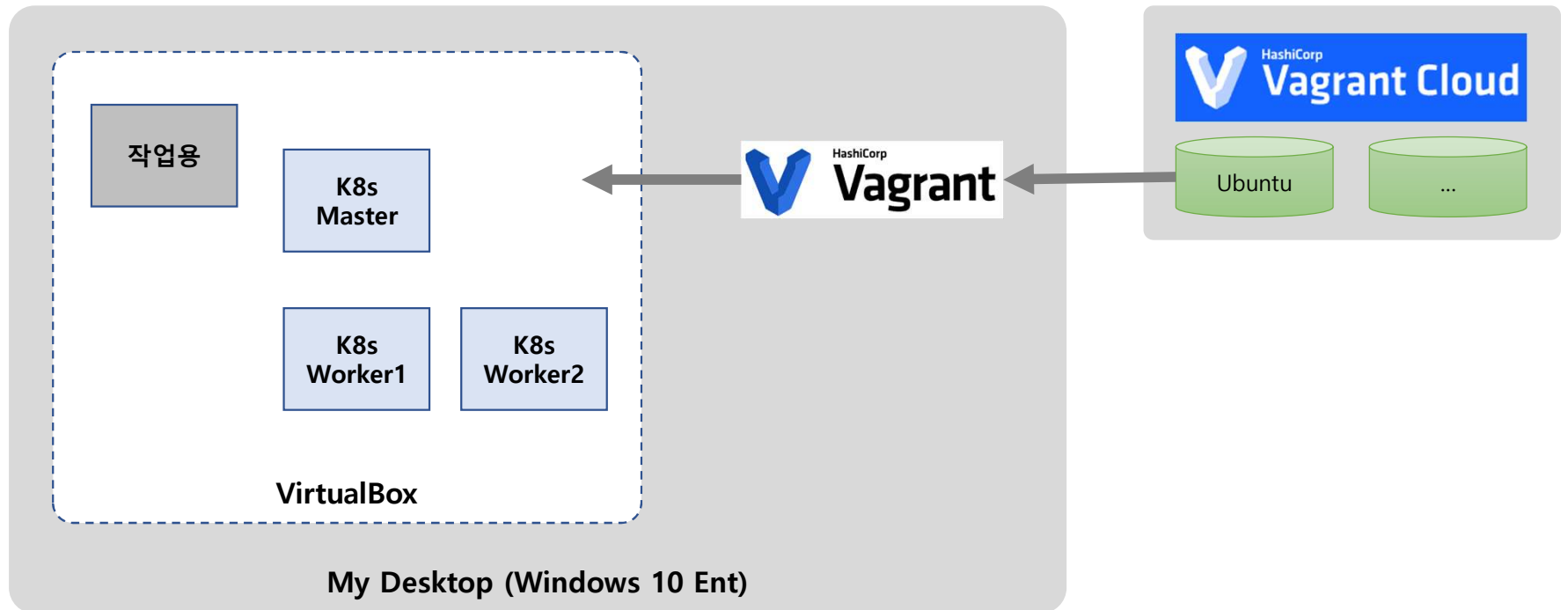
▷ kubectl

# Sketch ...

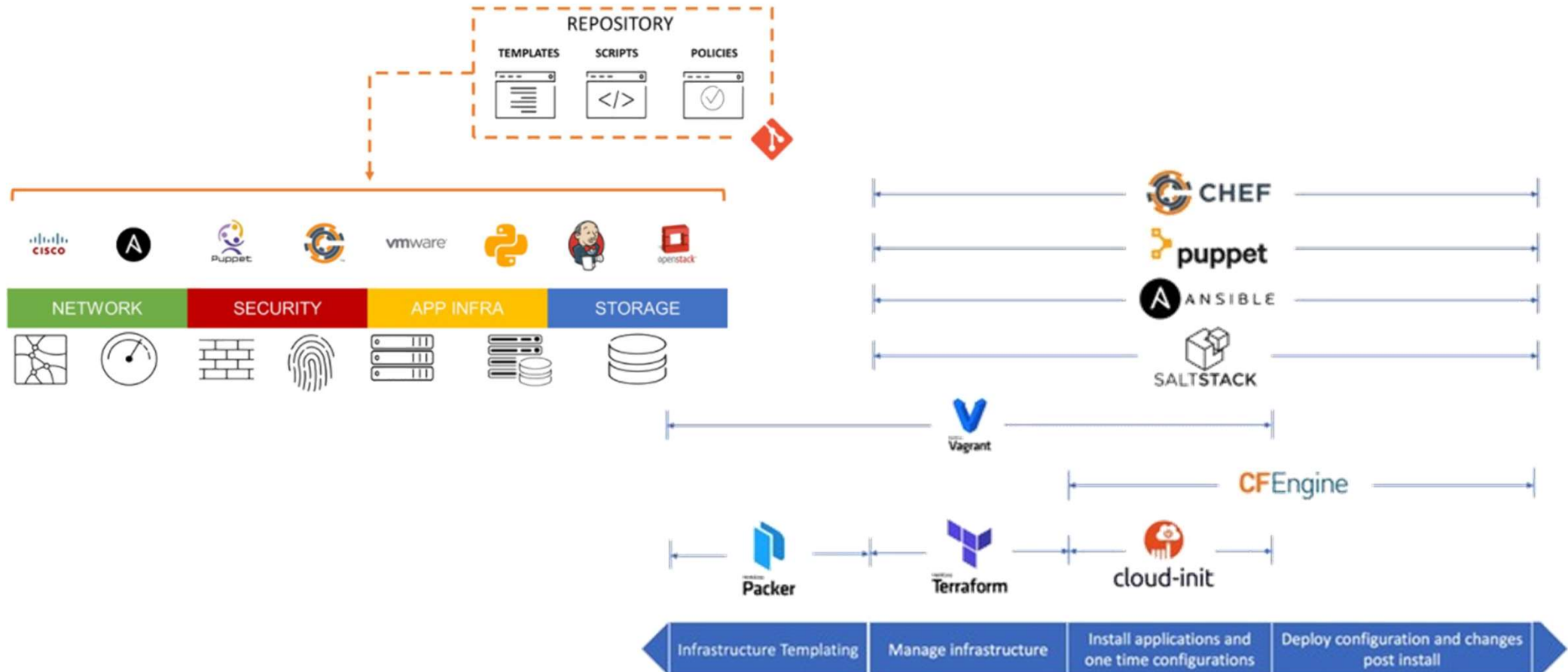
Vagrant를 이용해서 Kubernetes 구성을 위한 VM을 만들고  
그렇게 만들어진 VM에서 Master 및 Worker들을 설치하는 과정을 진행해보자.



공유기



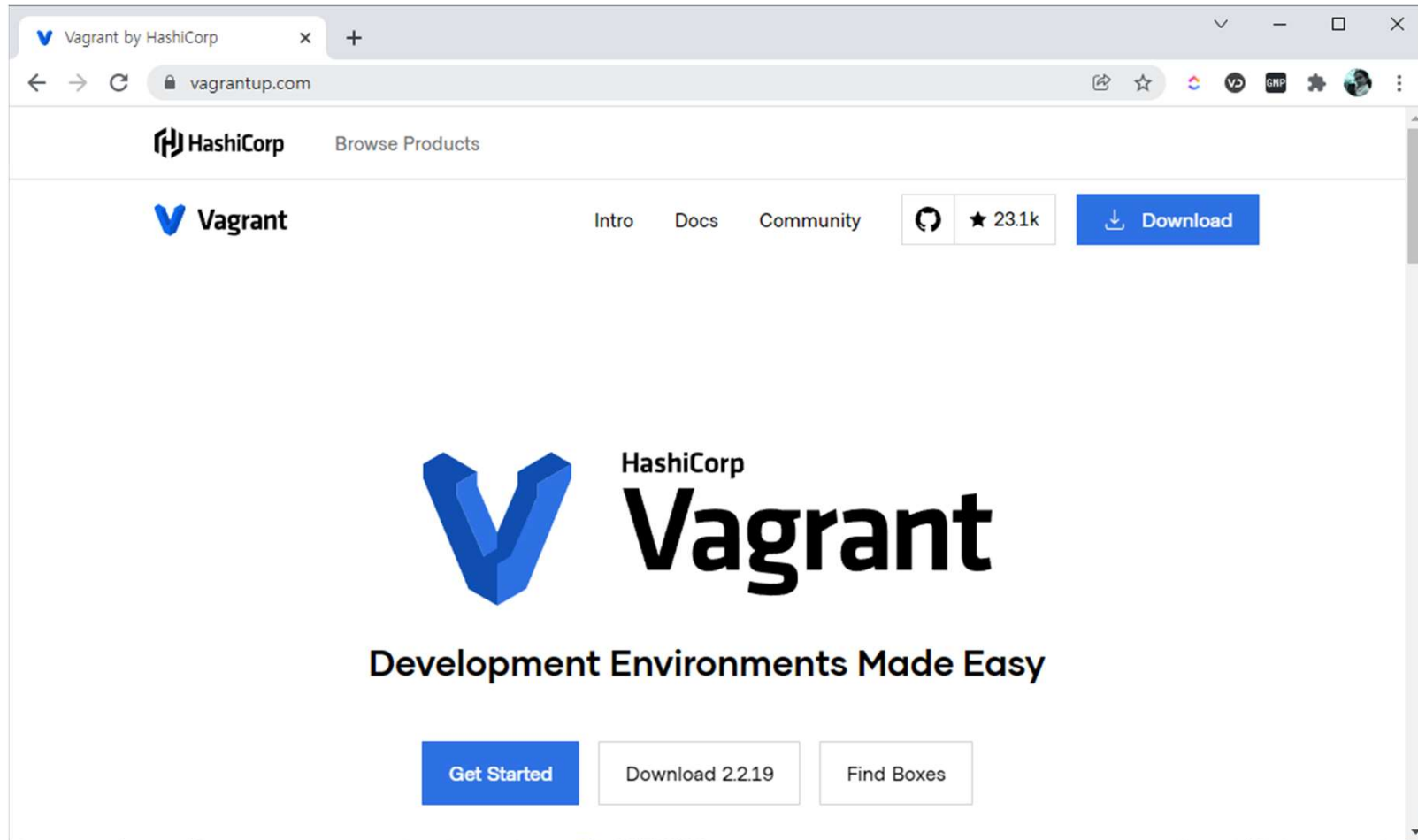
# IaC (Infrastructure as Code)



※ 참고 : <https://judo0179.tistory.com/120>

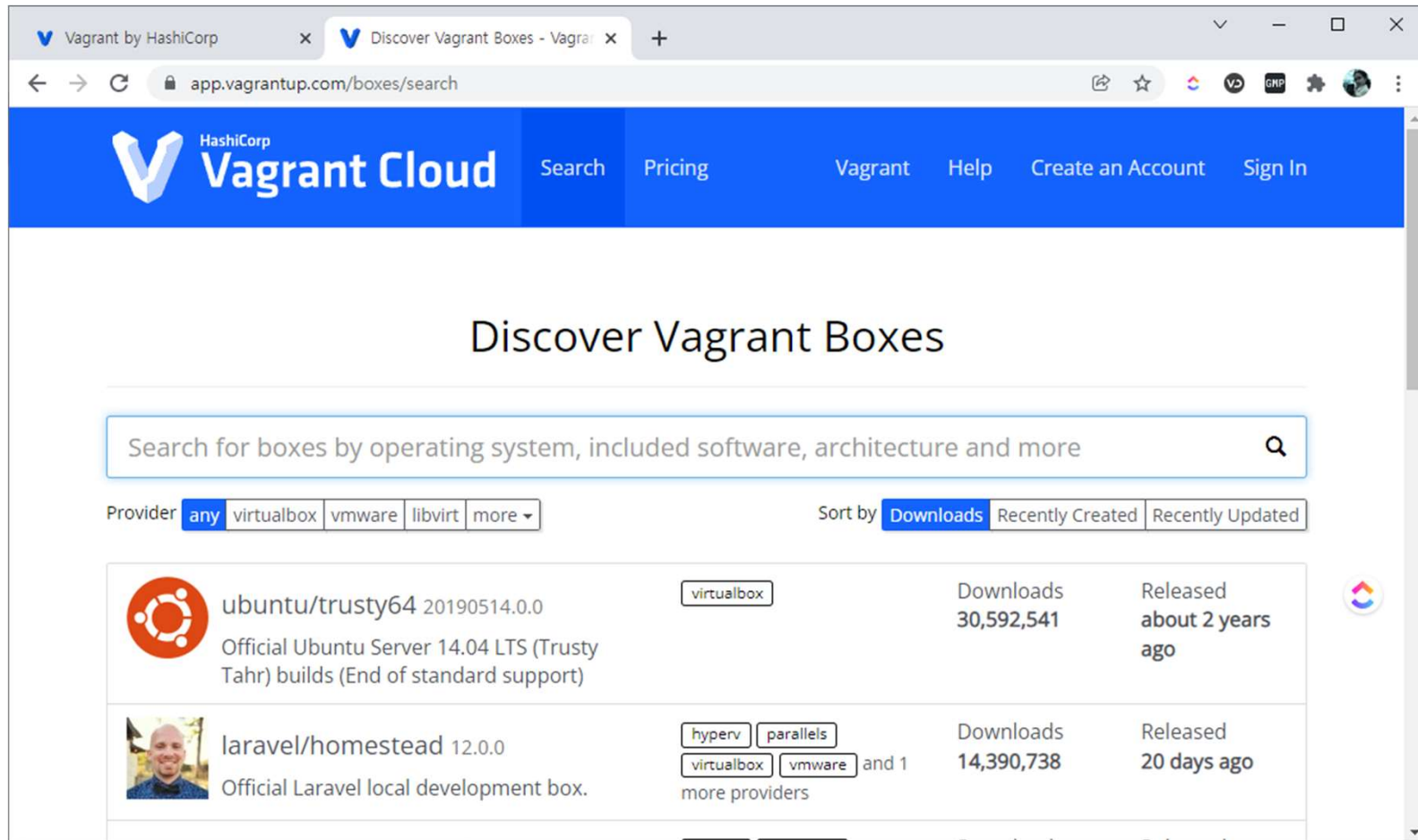
※ 참고 : <https://www.comworld.co.kr/news/articleView.html?idxno=50114>

# Vagrant



※ 참고 : <https://www.vagrantup.com/>

# Vagrant Cloud (Hub)



The screenshot shows the Vagrant Cloud website interface. The header is blue with the HashiCorp Vagrant Cloud logo and navigation links: Search, Pricing, Vagrant, Help, Create an Account, and Sign In. The main heading is "Discover Vagrant Boxes". Below this is a search bar with the placeholder text "Search for boxes by operating system, included software, architecture and more". Under the search bar, there are filters for "Provider" (any, virtualbox, vmware, libvirt, more) and "Sort by" (Downloads, Recently Created, Recently Updated). The search results are displayed in a table-like format with two visible entries:

Box Name	Version	Provider	Downloads	Released
ubuntu/trusty64	20190514.0.0	virtualbox	30,592,541	about 2 years ago
laravel/homestead	12.0.0	hyperv, parallels, virtualbox, vmware and 1 more providers	14,390,738	Released 20 days ago

※ 참고 : <https://app.vagrantup.com/boxes/search>

# Network (공유기)

Kubernetes 환경을 꾸밀 때, IP 고정 할당 해주기 위해서 집에서 사용하고 있는 공유기 환경에 대해서 파악하고 있어야 한다.

DHCP 주소 예약 목록

사용함	호스트명	IP 주소	MAC 주소		
<input checked="" type="checkbox"/>	HP-PC	192.168.100.100	28:00:00:00:00:14		
<input checked="" type="checkbox"/>	master	192.168.100.117	b4:7e:44:00:00:19f		
<input checked="" type="checkbox"/>	master-stg	192.168.100.111	08:00:27:00:00:04		
<input checked="" type="checkbox"/>	worker1	192.168.100.112	08:00:27:00:00:f2		
<input checked="" type="checkbox"/>	worker2	192.168.100.113	08:00:27:00:00:05		

DHCP 클라이언트 목록

호스트명	IP 주소	MAC 주소	만료시간
LAPTOP-D19USFQ8	192.168.100.103	7*00:00:00:00:00:13	6 일 22 시 32 분
jk7744-park03	192.168.100.104	60:00:00:00:00:99	6 일 21 시 31 분
jaegyui-Galaxy-Note10-5G	192.168.100.105	6e:53:00:00:00:00	6 일 22 시 39 분
SUPER-PC	192.168.100.101	*00:00:00:00:00:05	6 일 17 시 58 분
Galaxy-Note8	192.168.100.106	50:00:00:00:00:b5	20 시 34 분
cgllic-airmonitor-b1	192.168.100.110	90:97:00:00:00:00	6 일 3 시 49 분
jaegyui-Galaxy-Tab-S5e	192.168.100.107	*00:00:00:00:00:db	4 일 6 시 19 분
Parkui-iPhone	192.168.100.108	80:00:00:00:00:cf	6 일 13 시 56 분

설정저장

설정취소

WIRELESS

Copyright © 2013 D-Link Corporation. All rights reserved.

# Vagrantfile

Kubernetes Master 1대, Worker 2대를 설치할 VM을 확보하기 위한 Vagrantfile 이다.

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

N = 2

Vagrant.configure("2") do |config|

  config.vm.define "w-k8s-master" do |cfg|

    cfg.vm.box = "whatwant/Ubuntu-20.04-Server"
    cfg.vm.box_version = "0.2.0"

    cfg.vm.hostname = "master"
    cfg.vm.network "public_network", ip: "192.168.100.200"

    cfg.vm.provider "virtualbox" do |vb|
      vb.gui = false
      vb.cpus = "2"
      vb.memory = "2048"
    end

    cfg.vm.provision "shell", inline: <<-SHELL
      apt-get update
      apt-get upgrade -y
    SHELL
  end

end
```

```
(1..N).each do |i|

  config.vm.define "w-k8s-worker#{i}" do |cfg|

    cfg.vm.box = "whatwant/Ubuntu-20.04-Server"
    cfg.vm.box_version = "0.2.0"

    cfg.vm.hostname = "worker#{i}"
    cfg.vm.network "public_network", ip: "192.168.100.20#{i}"

    cfg.vm.provider "virtualbox" do |vb|
      vb.gui = false
      vb.cpus = "1"
      vb.memory = "1280"
    end

    cfg.vm.provision "shell", inline: <<-SHELL
      apt-get update
      apt-get upgrade -y
    SHELL
  end
end
end
```

※ 참고 : <https://www.whatwant.com/entry/Kubernetes-Vagrant-VirtualBox-Kubespray>



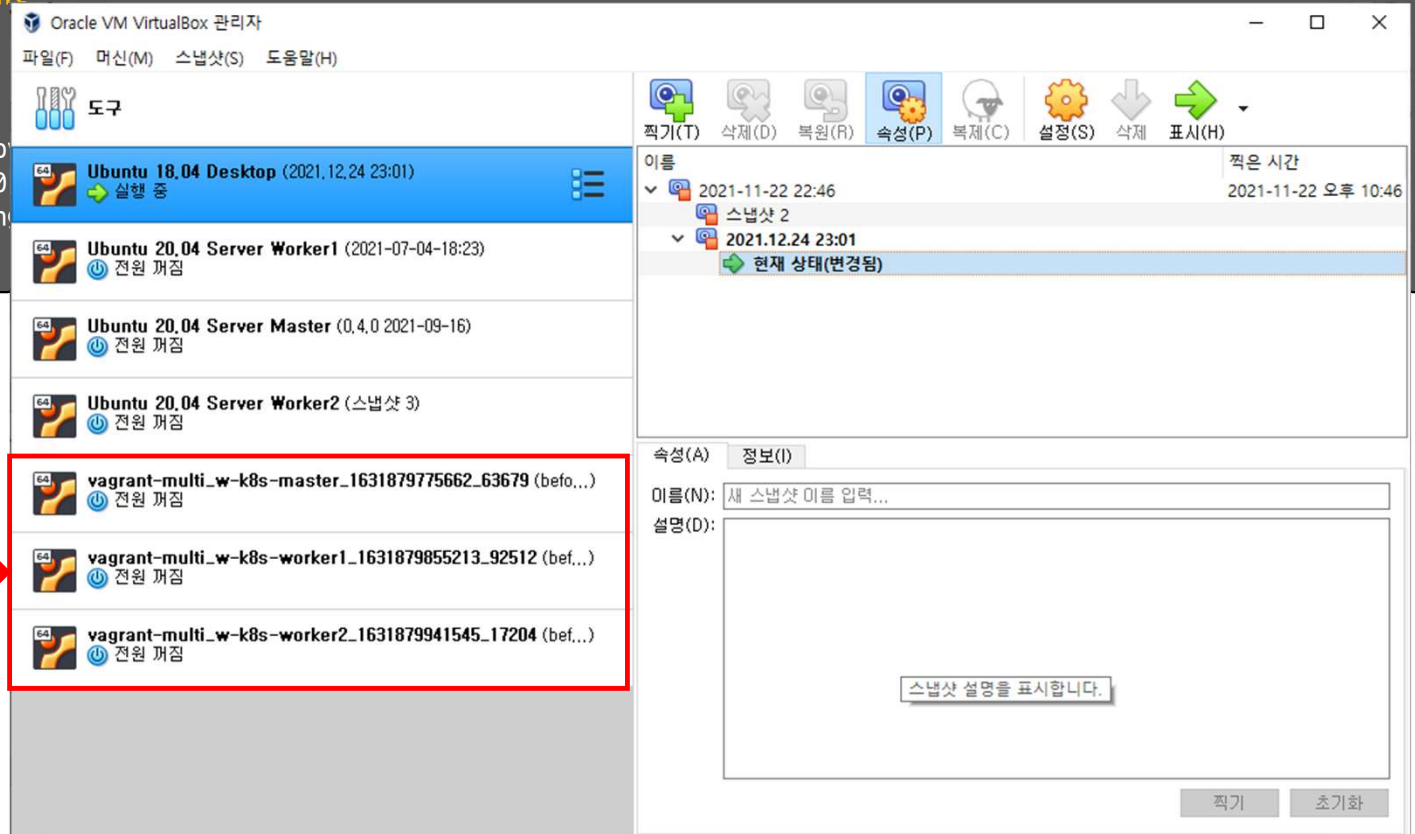
# Create VMs

개인적인 취향인데, Host(Windows)에 설치되어 있는 Git-Bash 환경에서 진행했다.

```
host > git clone https://github.com/whatwant-school/advanced-kubernetes.git
host > cd advanced-kubernetes/02-week/vagrant
```

```
host > vagrant up
```

```
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Importing base box 'whatwant/Ubuntu-20.04'...
==> default: Matching MAC address for NAT networking...
...
```



※ 참고 : <https://www.whatwant.com/entry/Kubernetes-Vagrant-VirtualBox-Kubespary>

# Master setup - SSH

K8s master에서 worker1 / worker2 모두 접근이 가능해야 설치가 진행되므로 이를 위한 환경 설정을 해보자  
우선 master에 접속을 하자. 기본 패스워드는 `vagrant`이다.

```
host > ssh vagrant@192.168.100.200
```

Kubespray를 이용해서 K8s 설치를 할 것이기에, master node에서 이를 위한 기본 패키지들이 설치되어야 한다.

```
master > sudo apt install ansible python3-argcomplete
```

master에서 K8s의 모든 node들에 접속할 수 있도록, SSH 키를 생성하고 이를 각 node들에 넣어두자

```
master > ssh-keygen
```

```
master > ssh-copy-id 192.168.100.200
```

```
master > ssh-copy-id 192.168.100.201
```

```
master > ssh-copy-id 192.168.100.202
```

※ 참고 : <https://www.whatwant.com/entry/Kubernetes-Vagrant-VirtualBox-Kubespray>

# Master setup - Ansible

ansible에서 모든 node들에 접속이 잘 되는지 확인해보자

```
master > sudo nano /etc/ansible/hosts
```

```
[all]
master ansible_ssh_host=192.168.100.200 ip=192.168.100.200
ansible_user=vagrant
worker1 ansible_ssh_host=192.168.100.201 ip=192.168.100.201
ansible_user=vagrant
worker2 ansible_ssh_host=192.168.100.202 ip=192.168.100.202
ansible_user=vagrant

[k8s_master]
master

[k8s_worker]
worker1
worker2
```

```
master > ansible all -m ping
```

※ 참고 : <https://www.whatwant.com/entry/Kubernetes-Vagrant-VirtualBox-Kubespray>

# Master setup - node configuration

K8s node로 사용하기 위해서는 서버의 설정들을 일부 맞춰줘야 한다.  
일단, swap을 꺼보자.

```
master/worker > sudo swapoff -a
```

```
master/worker > sudo nano /etc/fstab
```

```
# /swap.img    none    swap    sw    0    0
```

Network 부분에서 ip\_forward를 가능하도록 해주고,  
hostname으로 각 node 접근이 가능하도록 잡아주자.

```
master/worker > sudo sh -c 'echo 1 > /proc/sys/net/ipv4/ip_forward'
```

```
master/worker > sudo nano /etc/hosts
```

```
192.168.100.200 master
192.168.100.201 worker1
192.168.100.202 worker2
```

K8s 모든 node에 동일하게 적용되어야 한다.  
즉, master / worker1 / worker2 모두 반영해야 한다.

그리고, 이 시점에서 모두 **reboot** 하자.

※ 참고 : <https://www.whatwant.com/entry/Kubernetes-Vagrant-VirtualBox-Kubespray>

# Master setup - Kubespray install

K8s 설치하는 여러 방법 中 Kubespray를 선택했다.

```
master > cd /srv/install
master > git clone https://github.com/kubernetes-sigs/kubespray.git
master > cd kubespray
master > git checkout v2.16.0
master > sudo pip install -r requirements.txt
master > cp -rfp inventory/sample inventory/mycluster
master > declare -a IPS=(192.168.100.200 192.168.100.201 192.168.100.202)
master > CONFIG_FILE=inventory/mycluster/hosts.yaml python contrib/inventory_builder/inventory.py ${IPS[@]}
master > nano ./inventory/mycluster/hosts.yaml
```

※ 참고 : <https://www.whatwant.com/entry/Kubernetes-Vagrant-VirtualBox-Kubespray>

# Master setup - Kubespray configuration 1/2

우리 K8s node 환경에 맞도록 설정을 잡아준다

```
master > nano ./inventory/mycluster/hosts.yaml
```

```
all:
  hosts:
    master:
      ansible_host: 192.168.100.200
      ip: 192.168.100.200
      access_ip: 192.168.100.200
    worker1:
      ansible_host: 192.168.100.201
      ip: 192.168.100.201
      access_ip: 192.168.100.201
    worker2:
      ansible_host: 192.168.100.202
      ip: 192.168.100.202
      access_ip: 192.168.100.202
```

```
children:
  kube_control_plane:
    hosts:
      master:
  kube_node:
    hosts:
      worker1:
      worker2:
  etcd:
    hosts:
      master:
  k8s_cluster:
    children:
      kube_control_plane:
      kube_node:
  calico_rr:
    hosts: {}
```

※ 참고 : <https://www.whatwant.com/entry/Kubernetes-Vagrant-VirtualBox-Kubespray>

# Master setup - Kubespray configuration 2/2

같이 설치할 addon은 다음과 같이 true 설정하면 된다

```
master > nano ./inventory/mycluster/group_vars/k8s_cluster/addons.yml
```

```
...  
dashboard_enabled: true  
...  
helm_enabled: true  
...  
metrics_server_enabled: true  
...
```

proxy mode도 iptables로 설정하자

```
master > nano ./inventory/mycluster/group_vars/k8s_cluster/k8s-cluster.yml
```

```
# Kube-proxy proxyMode configuration.  
# Can be ipvs, iptables  
kube_proxy_mode: iptables
```

※ 참고 : <https://www.whatwant.com/entry/Kubernetes-Vagrant-VirtualBox-Kubespray>

# Master setup - Kubespray execute

준비는 끝났다. 이제 K8s 설치를 진행해보자

```
master > ansible-playbook -i inventory/mycluster/hosts.yaml --become --become-user=root cluster.yml
```

※ 참고 : <https://www.whatwant.com/entry/Kubernetes-Vagrant-VirtualBox-Kubespray>



# Master setup - kubectl

Kubernetes를 다루기 위한 대표적인 UI는 'kubectl'이다.

앞에서 진행한 대로 하면 'kubectl'도 같이 master node에 설치가 되기는 하지만, 'root' 계정으로만 사용 가능한 상태이다.

그래서 기본 사용자 계정에서도 'kubectl'을 사용할 수 있게 하려면 다음과 같은 과정을 거쳐야 한다.

```
master > mkdir -p $HOME/.kube  
  
master > sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config  
  
master > sudo chown $(id -u):$(id -g) $HOME/.kube/config  
  
master > echo "source <(kubectl completion zsh)" >> ~/.zshrc  
  
master > source ~/.zshrc
```

※ 참고 : <https://www.whatwant.com/entry/Kubernetes-Vagrant-VirtualBox-Kubespray>

# Master setup - check

이제 모두 끝났다 ???  
잘 된 것인지 확인해보자.

```
master > kubectl cluster-info
```

```
Kubernetes control plane is running at https://127.0.0.1:6443
```

```
To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
```

```
master > kubectl get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
node1	Ready	control-plane,master	14m	v1.20.7
node2	Ready	<none>	13m	v1.20.7
node3	Ready	<none>	13m	v1.20.7

※ 참고 : <https://www.whatwant.com/entry/Kubernetes-Vagrant-VirtualBox-Kubespray>

# Remote Workspace - 1/3

K8s가 설치되지 않은 다른 PC에서 'kubectl'을 사용하기

일단, 설치된 K8s version을 다시 한 번 확인하자. 같은 버전의 'kubectl' 설치를 위해 ...

```
remote > ssh vagrant@192.168.100.200
```

```
master > kubectl get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
master	Ready	control-plane,master	103d	v1.20.7
worker1	Ready	<none>	103d	v1.20.7
worker2	Ready	<none>	103d	v1.20.7

```
master > exit
```

'kubectl' 설치

```
remote > curl -LO "https://dl.k8s.io/release/v1.20.7/bin/linux/amd64/kubectl"
```

```
remote > sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl
```

```
remote > kubectl version
```

```
Client Version: version.Info{Major:"1", Minor:"20", GitVersion:"v1.20.7", GitCommit:"132a687512d7fb058d0f5890f07d4121b3f0a2e2", GitTreeState:"clean",  
BuildDate:"2021-05-12T12:40:09Z", GoVersion:"go1.15.12", Compiler:"gc", Platform:"linux/amd64"}  
The connection to the server localhost:8080 was refused - did you specify the right host or port?
```

※ 참고 : <https://kubernetes.io/ko/docs/tasks/tools/install-kubectl-linux/>

# Remote Workspace - 2/3

앞에서 설치한 우리의 K8s 시스템과 연결하자. (인증 정보 복사해오기)

```
remote > mkdir ~/.kube
```

```
remote > scp vagrant@192.168.100.200:/home/vagrant/.kube/config ~/.kube/
```

```
The authenticity of host '192.168.100.200 (192.168.100.200)' can't be established.  
ECDSA key fingerprint is SHA256:RX1VW+w652onI+Tz8gPnJRm7SM1urzscf8iHQeJFF0o.  
Are you sure you want to continue connecting (yes/no)? yes  
Warning: Permanently added '192.168.100.200' (ECDSA) to the list of known hosts.  
config
```

확인해보면, 아직도 뭔가 이상하다 ??

```
remote > kubectl version
```

```
Client Version: version.Info{Major:"1", Minor:"20", GitVersion:"v1.20.7", GitCommit:"132a687512d7fb058d0f5890f07d4121b3f0a2e2", GitTreeState:"clean",  
BuildDate:"2021-05-12T12:40:09Z", GoVersion:"go1.15.12", Compiler:"gc", Platform:"linux/amd64"}  
The connection to the server 127.0.0.1:6443 was refused - did you specify the right host or port?
```

우리가 설치한 K8s master node를 찾지 못하는 것이다.

※ 참고 : <https://kubernetes.io/ko/docs/tasks/tools/install-kubectl-linux/>

# Remote Workspace - 3/3

K8s master node IP를 제대로 넣어주면 된다.

```
remote > nano ~/.kube/config
```

```
apiVersion: v1
clusters:
- cluster:
  certificate-authority-data:...
  server: https://192.168.100.200:6443
  name: cluster.local
...
```

Docker도 설치해 놓자 ! ! !

이제는 정상적으로 잘 나온다.

```
remote > kubectl version
```

```
Client Version: version.Info{Major:"1", Minor:"20", GitVersion:"v1.20.7", GitCommit:"132a687512d7fb058d0f5890f07d4121b3f0a2e2", GitTreeState:"clean",
BuildDate:"2021-05-12T12:40:09Z", GoVersion:"go1.15.12", Compiler:"gc", Platform:"linux/amd64"}
Server Version: version.Info{Major:"1", Minor:"20", GitVersion:"v1.20.7", GitCommit:"132a687512d7fb058d0f5890f07d4121b3f0a2e2", GitTreeState:"clean",
BuildDate:"2021-05-12T12:32:49Z", GoVersion:"go1.15.12", Compiler:"gc", Platform:"linux/amd64"}> nano ~/.kube/config
```

```
remote > kubectl get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
master	Ready	control-plane,master	103d	v1.20.7
worker1	Ready	<none>	103d	v1.20.7
worker2	Ready	<none>	103d	v1.20.7

※ 참고 : <https://kubernetes.io/ko/docs/tasks/tools/install-kubectllinux/>



# **Tip #1**

**Enable shell  
autocompletion**

# Autocompletion

case #1 - bash

```
remote > sudo apt install bash-completion

remote > source /usr/share/bash-completion/bash_completion

(remote shell)

remote > echo 'source <(kubectl completion bash)' >> ~/.bashrc

remote > kubectl completion bash >/etc/bash_completion.d/kubectl

remote > source ~/.bashrc
```

case #2 - zsh

```
remote > echo 'source <(kubectl completion zsh) ' >> ~/.zshrc

remote > source ~/.zshrc
```

※ 참고 : <https://kubernetes.io/ko/docs/tasks/tools/install-kubectl-linux/>





**Break**



# **Flip Learning**

## **(Pods & Namespaces)**

**- 이민준님**



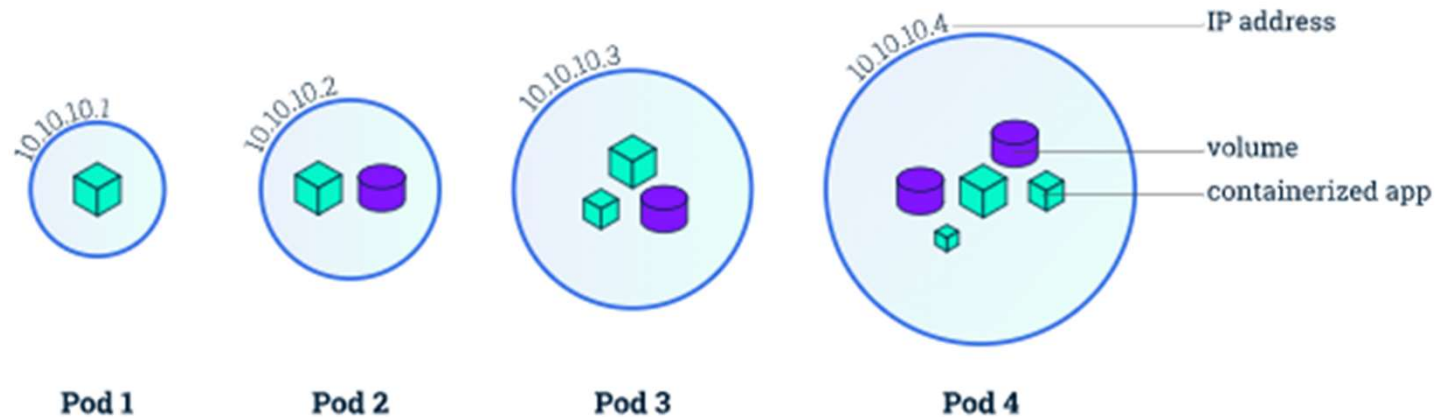
# Kubernetes

## Pod

# Pod is ... #1

**Pod**는 Kubernetes에서 생성하고 관리할 수 있는 배포 가능한 가장 작은 컴퓨팅 단위이다.

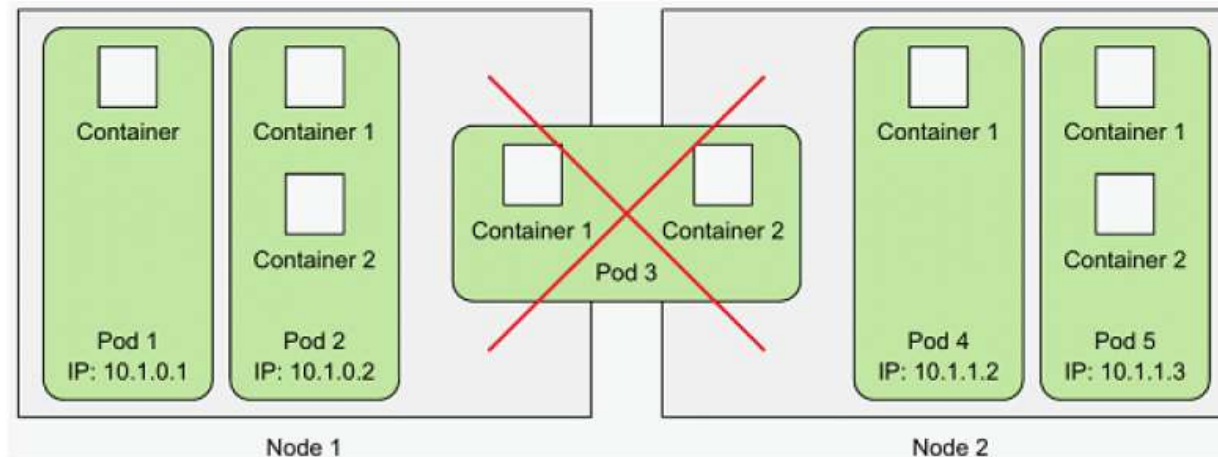
**Pod**는 하나 이상의 컨테이너 그룹이다.



※ 참고 : <https://kubernetes.io/ko/docs/tutorials/kubernetes-basics/explore/explore-intro/>

## Pod is ... #2

- Pod는 함께 배치된 Container 그룹을 의미
- Container는 단일 프로세스를 실행하는 것을 목적으로 설계
- 따라서, 여러 Container를 묶고 하나의 단위로 관리할 수 있는 상위 구조가 필요 → Pod
- Kubernetes는 Pod 단위로 배포하고 운영



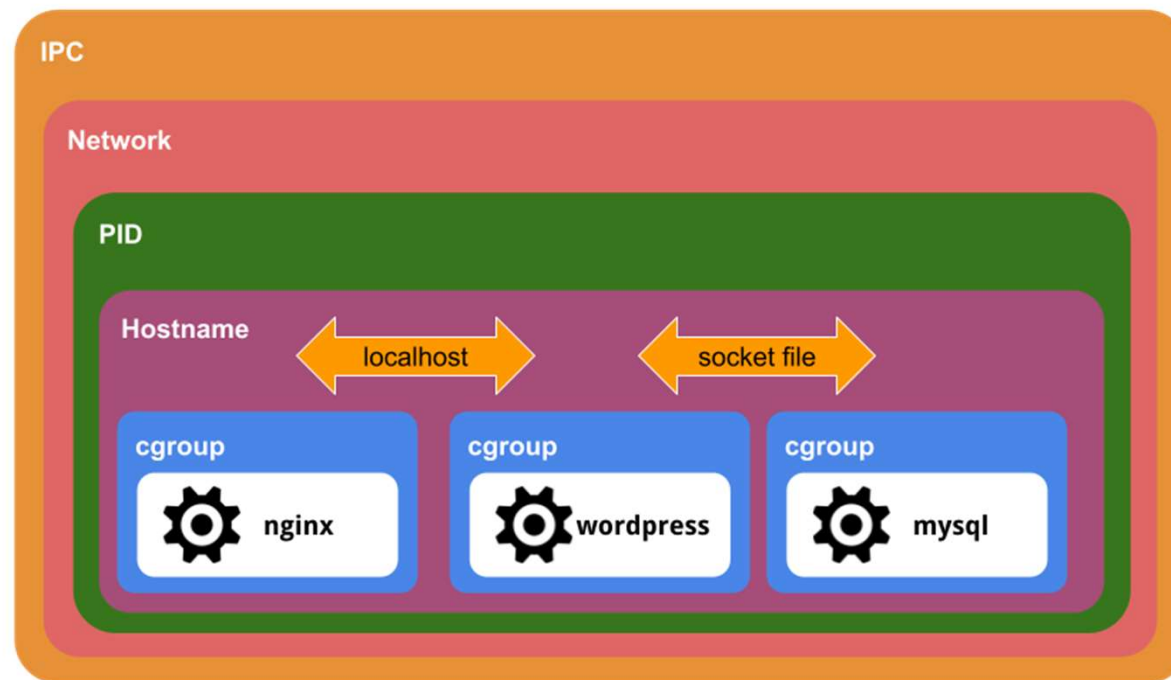
▲ 그림 3.1 파드 안에 있는 모든 컨테이너는 같은 노드에서 실행된다. 절대로 두 노드에 걸쳐 배포되지 않는다.

※ 참고 : <https://livebook.manning.com/book/kubernetes-in-action/chapter-3/10>



## Pod is ... #3

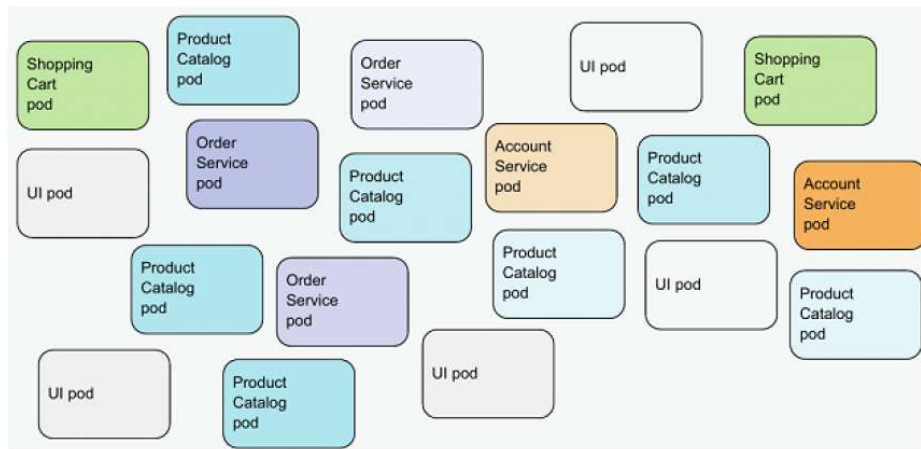
- Pod 1개 안에서 여러 개의 Container가 실행되는 것은 단순히 각 프로세스가 동일한 머신 위에서 실행하는 것과 유사
- 이들 프로세스는 localhost(127.0.0.1)로 네트워크 통신을 할 수 있으며, 볼륨에 있는 파일을 공유 할 수 있다.
- 또는 IPC(Inter Process Communication)를 이용하거나 HUP, TERM 시그널을 보낼 수도 있다.



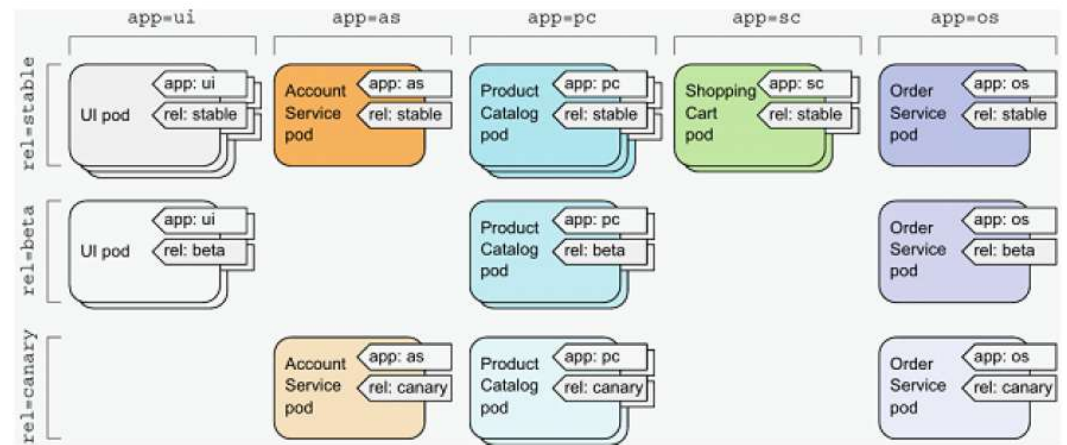
※ 참고 : <https://www.joinc.co.kr/w/man/12/kubernetes/Pod>

# label

- Label은 Kubernetes 리소스를 분류할 수 있는 기능
- 각 오브젝트는 하나 이상의 레이블을 가질 수 있으며 label은 Key-Value Pair로 이루어짐
- Kubernetes 명령어에서 동일한 label을 가진 오브젝트를 선택할 수 있음



▲ 그림 3.6 마이크로서비스 아키텍처 안에 있는 분류되지 않는 파드



▲ 그림 3.7. 파드 레이블을 이용해 마이크로서비스 아키텍처 안에 파드를 조직화했다.



# **K8s : Pod Hands-On**

# Scenario

실습 시나리오는 다음과 같다

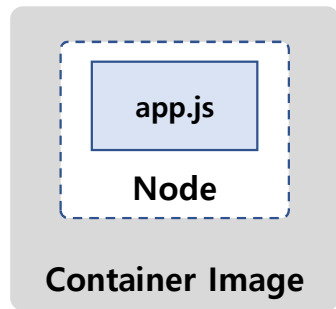
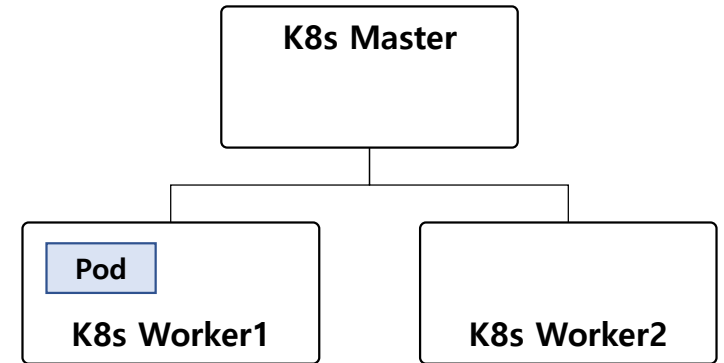


Image Build



Image Push



Pod Create

※ 참고 : <https://github.com/luksa/kubernetes-in-action/tree/master/Chapter02/kubia>

# Image Build / Push

간단한 Node App을 하나 만들어 보자

app.js

```
const http = require('http');
const os = require('os');

console.log("node-web server starting...");

var handler = function(request, response) {
  console.log("Received request from " +
    request.connection.remoteAddress);
  response.writeHead(200);
  response.end("You've hit " + os.hostname() + "\n");
};

var www = http.createServer(handler);
www.listen(8080);
```

Dockerfile

```
FROM node:latest
ADD app.js /app.js
ENTRYPOINT ["node", "app.js"]
```

```
remote > git clone https://github.com/whatwant-school/advanced-kubernetes.git
remote > cd advanced-kubernetes/02-week/Pod

remote > docker build -t node-web:1.0 .

remote > docker tag node-web:1.0 <user-id>/node-web:1.0

remote > docker push <user-id>/node-web:1.0
```

※ 참고 : <https://github.com/luksa/kubernetes-in-action/tree/master/Chapter02/kubia>

# Pod Create by YAML

앞에서 만든 container를 실행하기 위한 Pod를 생성해보자.

pod-node-web.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: node-web
  labels:
    creation_method: manual
    env: stage
spec:
  containers:
  - image: whatwant/node-web:1.0
    name: node-web
    ports:
    - containerPort: 8080
      protocol: TCP
```

```
remote > git clone https://github.com/whatwant-school/advanced-kubernetes.git
```

```
remote > cd advanced-kubernetes/02-week/Pod
```

```
remote > kubectl create -f pod-node-web.yaml
```

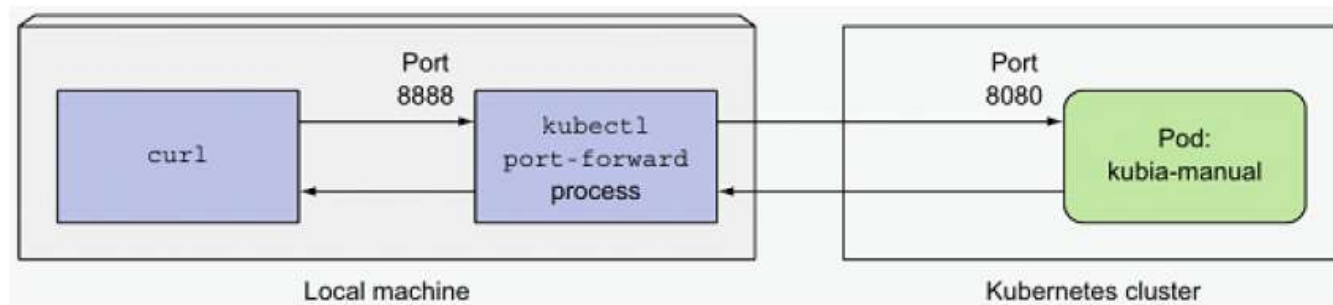
```
remote > kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
node-web	1/1	Running	0	71s

※ 참고 : <https://github.com/luksa/kubernetes-in-action/tree/master/Chapter02/kubia>

# port-forward

그럼 이제, 우리의 Pod에서 실행되고 있는 Node App에 접속을 해보자.  
어떻게 해야 할까?!



▲ 그림 3.5 curl을 kubectl port-forward와 함께 사용할 때 일어나는 일을 간략하게 설명한다.

```
remote > kubectl port-forward node-web 8080:8080 &
```

```
[1] 8466
Forwarding from 127.0.0.1:8080 -> 8080
Forwarding from [::1]:8080 -> 8080
```

```
remote > curl http://localhost:8080
```

```
Handling connection for 8080
You've hit node-web
```

※ 참고 : <https://github.com/luksa/kubernetes-in-action/tree/master/Chapter02/kubia>



# Pod Delete

앞에서 만든 port-forward 프로세스를 삭제해보자.

```
remote > ps -ef | grep kubectl
```

```
chani      8466   3572  0 15:35 pts/1    00:00:00 kubectl port-forward node-web 8080:8080
chani      8728   3572  0 15:47 pts/1    00:00:00 grep --color=auto --exclude-dir=.bzz --exclude-dir=CVS --exclude-dir=.git --exclude-dir=.hg --exclude-dir=.svn --exclude-dir=.idea --exclude-dir=.tox kubectl
```

```
remote > kill -9 8466
```

```
[1]  + 8466 killed      kubectl port-forward node-web 8080:8080
```

이제는 Pod도 삭제해보자.

```
remote > kubectl get pods -o wide
```

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE	NOMINATED NODE	READINESS GATES
node-web	1/1	Running	0	28m	10.233.103.67	worker2	<none>	<none>

```
remote > kubectl delete pods node-web
```

```
pod "node-web" deleted
```

```
remote > kubectl get pods -o wide
```

```
No resources found in default namespace.
```

# Pod Create by CLI

YAML 파일 없이도 그냥 Pod를 생성할 수 있다.

```
remote > kubectl run node-web-command --image whatwant/node-web:1.0 --port=8080
```

```
pod/node-web-command created
```

```
remote > kubectl get pods -o wide
```

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE	NOMINATED	NODE	READINESS	GATES
node-web-command	1/1	Running	0	4s	10.233.103.68	worker2	<none>		<none>	

port-forward 해서 확인도 해보고, 삭제도 해보자

```
remote > kubectl port-forward node-web-command 8080:8080 &
```

```
remote > curl http://localhost:8080
```

```
remote > ps -ef | grep kubectl
```

```
remote > kill -9 11766
```

```
remote > kubectl delete pods node-web-command
```

```
pod "node-web-command" deleted
```

```
remote > kubectl get pods -o wide
```

```
No resources found in default namespace.
```

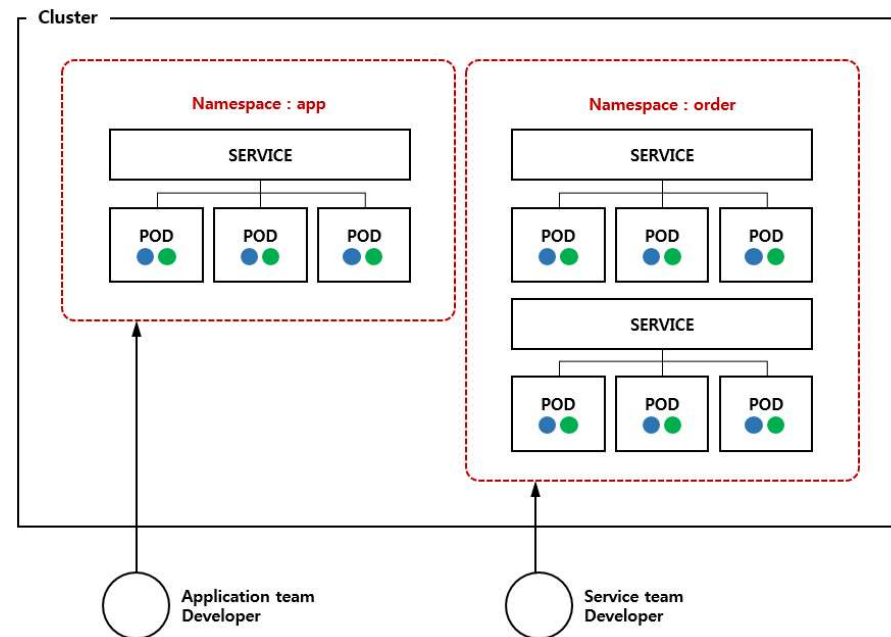


# **Kubernetes**

# **Namespace**

# Namespace is ...

- 네임스페이스는 클러스터 자원을 (리소스 쿼터를 통해) 여러 사용자 사이에서 나누는 방법
  - . 여러 개의 팀이나, 프로젝트에 걸쳐서 많은 사용자가 있는 환경에서 사용
  - . 네임스페이스는 이름의 범위를 제공
  - . 네임스페이스는 서로 중첩될 수 없으며, 각 쿠버네티스 리소스는 하나의 네임스페이스에만 존재
  - . Pod, ReplicaSet, Deployment, Service 등을 묶어 놓을 수 있는 하나의 가상 공간 또는 그룹



※ 참고 : <https://wiki.webnori.com/display/kubernetes/Namespace>



# **K8s : Namespace Hands-On**

# Namespace

이미 많은 namespace가 있다. 확인해보자.

```
remote > kubectl get namespaces
```

NAME	STATUS	AGE
default	Active	103d
ingress-nginx	Active	103d
kube-node-lease	Active	103d
kube-public	Active	103d
kube-system	Active	103d
test-ingress	Active	102d

특별하게 지정하지 않으면 기본적으로 사용되는 공간

```
remote > kubectl get pods --namespace default
```

No resources found in default namespace.

```
remote > kubectl get pods --namespace kube-system
```

NAME	READY	STATUS	RESTARTS	AGE
calico-kube-controllers-7c5b64bf96-cdj9r	1/1	Running	28	103d
calico-node-29cnd	1/1	Running	20	103d
calico-node-4lbgw	1/1	Running	16	103d
calico-node-srq8m	1/1	Running	20	103d
coredns-657959df74-g2ljk	1/1	Running	10	103d
coredns-657959df74-lvd9h	1/1	Running	16	103d
dns-autoscaler-b5c786945-gdzh	1/1	Running	16	103d
kube-apiserver-master	1/1	Running	17	103d
kube-controller-manager-master	1/1	Running	17	103d
kube-proxy-7jl29	1/1	Running	16	103d
kube-proxy-pp5sv	1/1	Running	13	103d
kube-proxy-spvnh	1/1	Running	12	103d
kube-scheduler-master	1/1	Running	17	103d
kubernetes-dashboard-7ddc76ff5f-h4cp8	1/1	Running	15	103d
kubernetes-metrics-scraper-64db6db887-59bhm	1/1	Running	10	103d
metrics-server-988c74c86-x6g9v	2/2	Running	32	103d
nginx-proxy-worker1	1/1	Running	14	103d
nginx-proxy-worker2	1/1	Running	13	103d
node-localdns-ctbfh	1/1	Running	13	103d
node-localdns-lch58	1/1	Running	16	103d
node-localdns-r5p5l	1/1	Running	12	103d



# Namespace Create by YAML

YAML 파일을 이용해서 namespace를 생성해보자

namespace-whatwant.yaml

```
apiVersion: v1
kind: Namespace
metadata:
  name: whatwant
```

```
remote > git clone https://github.com/whatwant-school/advanced-kubernetes.git
```

```
remote > cd advanced-kubernetes/02-week/Namespac
```

```
remote > kubectl create -f ./namespace-whatwant.yaml
```

```
namespace/whatwant created
```

```
remote > kubectl get namespaces
```

NAME	STATUS	AGE
default	Active	104d
ingress-nginx	Active	103d
kube-node-lease	Active	104d
kube-public	Active	104d
kube-system	Active	104d
test-ingress	Active	102d
whatwant	Active	21s

# Namespace Usage

namespace를 지정해서 Pod를 실행하는 예제를 살펴보자

```
remote > cd advanced-kubernetes/02-week/Pod
```

```
remote > kubectl create -f ./namespace-whatwant.yaml --namespace whatwant
```

```
namespace/whatwant created
```

```
remote > kubectl get pods --namespace whatwant
```

```
No resources found in whatwant namespace.
```

```
remote > kubectl create -f ./pod-node-web.yaml --namespace whatwant
```

```
pod/node-web created
```

```
remote > kubectl get pods
```

```
No resources found in default namespace.
```

```
remote > kubectl get pods --namespace whatwant
```

NAME	READY	STATUS	RESTARTS	AGE
node-web	1/1	Running	0	12s

# Namespace Delete

Pod 삭제 하고, Namespace까지 삭제 해보자

```
remote > kubectl delete pods node-web
```

```
Error from server (NotFound): pods "node-web" not found
```

```
remote > kubectl delete pods node-web --namespace whatwant
```

```
pod "node-web" deleted
```

```
remote > kubectl delete namespace whatwant
```

```
namespace "whatwant" deleted
```

```
remote > kubectl get namespaces
```

NAME	STATUS	AGE
default	Active	104d
ingress-nginx	Active	103d
kube-node-lease	Active	104d
kube-public	Active	104d
kube-system	Active	104d
test-ingress	Active	102d

# Namespace Create by CLI

resource 생성할 때 YAML 이용하는 것이 좋지만, namespace는 CLI로 생성해도 편잡을 정도로 simple 하다.

```
remote > kubectl create namespace whatwant
```

```
namespace/whatwant created
```

```
remote > kubectl get namespaces
```

NAME	STATUS	AGE
default	Active	104d
ingress-nginx	Active	103d
kube-node-lease	Active	104d
kube-public	Active	104d
kube-system	Active	104d
test-ingress	Active	102d
whatwant	Active	2s

```
remote > kubectl delete namespace whatwant
```

```
namespace "whatwant" deleted
```



**Tip #2**

**Cheat Sheet**

# cheat sheet

```
> kubectl cluster-info
```

```
> kubectl get nodes
```

```
> kubectl get namespaces
```

```
> kubectl create -f <yaml file>
```

```
> kubectl apply -f <yaml file>
```

```
> kubectl run <name> --image <image>
```

```
> kubectl get pods
```

```
> kubectl logs <pod name>
```

```
> kubectl describe pod <pod name>
```

```
> kubectl delete pod <pod name>
```

```
> kubectl get pods -w
```

```
> kubectl get pods -o wide
```

```
> kubectl get pods -n <namespace>
```

```
> kubectl get pods -l <label>
```





**Problem**

# Container Image missing - 1/2

Image를 찾지 못하는 경우 어떻게 해야 할까?

pod-error.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-error
  labels:
    env: trouble
spec:
  containers:
  - image: whatwant/err-pod:1.0
    name: pod-error
```

Q1. 'kubectl create' vs. 'kubectl apply' ?

Q2. 에러에 대한 상세 정보를 확인하려면 ?

Q3. Pod 삭제 직접 해보기 !

```
remote > cd advanced-kubernetes/02-week/Pod
```

```
remote > kubectl apply -f ./pod-error.yaml
```

```
pod/pod-error created
```

```
remote > kubectl get pods -o wide
```

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE	NOMINATED NODE	READINESS GATES
pod-error	0/1	ImagePullBackOff	0	58s	10.233.103.75	worker2	<none>	<none>

# Container Image missing - 2/2

```
remote > kubectl describe pod pod-error
```

```
Name:          pod-error
Namespace:     default
Priority:      0
Node:         worker2/192.168.100.202
Start Time:   Fri, 31 Dec 2021 20:59:11 +0900
Labels:       env=trouble
Annotations:  cni.projectcalico.org/podIP: 10.233.103.75/32
              cni.projectcalico.org/podIPs: 10.233.103.75/32

Status:       Pending
IP:           10.233.103.75
IPs:
  IP: 10.233.103.75
Containers:
  pod-error:
    Container ID:
    Image:         whatwant/err-pod:1.0
    Image ID:
    Port:         <none>
    Host Port:    <none>
    State:        Waiting
      Reason:     ImagePullBackOff
    Ready:        False
    Restart Count: 0
    Environment:  <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from
      default-token-mdnvq (ro)
Conditions:
  Type          Status
  Initialized    True
  Ready          False
  ContainersReady False
  PodScheduled   True
```

## Volumes:

default-token-mdnvq:

Type: Secret (a volume populated by a Secret)

SecretName: default-token-mdnvq

Optional: false

QoS Class: BestEffort

Node-Selectors: <none>

Tolerations: node.kubernetes.io/not-ready:NoExecute op=Exists for 300s  
node.kubernetes.io/unreachable:NoExecute op=Exists for 300s

## Events:

Type	Reason	Age	From	Message
Normal	Scheduled	7m32s	default-scheduler	Successfully assigned default/pod-error to worker2
Normal	Pulling	5m54s (x4 over 7m31s)	kubelet	Pulling image "whatwant/err-pod:1.0"
Warning	Failed	5m51s (x4 over 7m29s)	kubelet	Failed to pull image "whatwant/err-pod:1.0": rpc error: code = Unknown desc = Error response from daemon: pull access denied for whatwant/err-pod, repository does not exist or may require 'docker login': denied: requested access to the resource is denied
Warning	Failed	5m51s (x4 over 7m29s)	kubelet	Error: ErrImagePull
Warning	Failed	5m29s (x7 over 7m28s)	kubelet	Error: ImagePullBackOff
Normal	BackOff	2m29s (x20 over 7m28s)	kubelet	Back-off pulling image "whatwant/err-pod:1.0"



# 보충 수업

데이터센터 모놀리스에서  
클라우드 쿠버네티스로

# KCD Korea 2021

## Kubernetes Community Days Korea 2021

- <https://community.cncf.io/events/details/cncf-kcd-korea-presents-kubernetes-community-days-korea/>
- [https://www.youtube.com/playlist?list=PLj6h78yzYM2OO9\\_EWXS13LxAe-Bkn0xXt](https://www.youtube.com/playlist?list=PLj6h78yzYM2OO9_EWXS13LxAe-Bkn0xXt)



성민 이  
Netflix

02:00 오후 — 02:40 오후

### Hermes | 데이터센터 모놀리스에서 클라우드 쿠버네티스로: 클라우드 네이티브로의 성공적인 이전을 위한 전략 | Netflix

잘 짜여진 모놀리스는 효율적이고 직관적이며 운용하기 편리한 합리적인 아키텍처입니다. 대부분의 작은 규모의 회사는 모놀리스로 서비스를 시작하지만 이 중 대부분은 서비스의 규모와 인력이 점차 늘어남에 따라 효율적인 아키텍처를 장기적으로 유지하는데 실패합니다. 오히려 모놀리스라는 이름의 수년간 쌓아온 프랑켄슈타인에 세번째 팔과 다섯번째 다리를 용접하고 있는 자신을 문득 깨닫게 되는 일도 드물지 않습니다. 뒤늦게 개발도 운용도 정상적으로 진행 할 수 없는 하향 나선의 상황에 직면에 있음을 깨닫게 되어도, 이미 하루에도 수만명이 이용하는 서비스를 근본적으로 손 대는 것 또한 좀처럼 엄두가 나지 않습니다. 우리는 수년간 쌓아 온 모놀리스 서비스를 해체하고 클라우드의 쿠버네티스로 옮길 수 있을까요? 어떻게하면 서비스를 사용하는 수 많은 유저들이 알아차리지 못하는 사이에 우리의 서비스를 안전하게 이전 할 수 있을까요? 클라우드 네이티브로의 이전은 과연 우리들의 Velocity에 큰 영향을 줄까요? 이 세션을 통해 데이터센터의 모놀리스를 클라우드의 쿠버네티스로 성공적으로 이전하기 위한 전략과 클라우드 네이티브 서비스의 장점, 그리고 주의점들을 짚어보려 합니다.

[발표자료] : [https://drive.google.com/file/d/1phjKwPQp7fSqyDeHRQFAbGn5D0\\_H\\_U9G/view](https://drive.google.com/file/d/1phjKwPQp7fSqyDeHRQFAbGn5D0_H_U9G/view)

[강의영상] : [https://www.youtube.com/watch?v=lyrg4x-A\\_Jk&list=PLj6h78yzYM2OO9\\_EWXS13LxAe-Bkn0xXt&index=5](https://www.youtube.com/watch?v=lyrg4x-A_Jk&list=PLj6h78yzYM2OO9_EWXS13LxAe-Bkn0xXt&index=5)



<https://kahoot.it/>

**[ Score ]**

이혜정 (3)

김상호 (2)

이민준 (1)



# 자습 (복습)

▷ Lab Setting

▷ K8s Pod

- Pod 심화: <https://speakerdeck.com/devinjeon/containerbuteo-dasi-salpyeoboneun-kubernetes-pod-dongjag-weonri>

▷ K8s Namespace