K8s 실습
(Kubernetes)

2019. 11.

K8s Clustering의 설치
부분만 공개 합니다.

안 종 석
james@jslab.kr
JS Lab

목차

I. K8s Clustering

II. K8s Operations

목차

I. K8s Clustering

II. K8s Operations

I. K8s Clustering

1. K8s 인프라 개요
2. K8s 설치 개요
3. Host for MicroK8s
4. Host for Kubeadm
5. Host for Rancher

I. K8s Clustering 1. K8s 인프라 개요

1) K8s Architecture 개요
2) 계층화 인프라
3) IaaS(Infrastructure as a Service) 계층
4) Cloud Native 계층
5) MSA 고속처리 HCI 구성
6) 성능 확장 인프라
7) 컨테이너 기반 보안 강화 플랫폼 구성
8) 스토리지 구성
9) DMZ 영역 구성
10) L7 LB(Load Balancer) 구성

I. K8s Clustering 2. 설치 개요

1) K8s 구축 솔루션
2) Hypervisor
3) Check Security Policy

2. 설치 개요

1) Kubernetes 구축 솔루션

- **Kubernetes 구축 솔루션**

1. **Self-managed solution:**
   - **Minikube:** Single node, Windows and MacOS with virtualization
   - **MicroK8s:** It was just for a Single node, Linux (Now for multimode)
   - **K3s:** lightweight Kubernetes (512 MB RAM, 200 MB disk), SQLite (Instead of Etcd)
   - **Kind:** Kubernetes-in-Docker (Windows, Mac, and Linux)
   - **Desktop Docker:** Docker for Mac/Windows now ships with a bundled Kubernetes offering
   - **K3d:** K3s-in-Docker (similar to Kind).
   - **Kubeadm:** The official CNCF tool for provisioning Kubernetes clusters
   - **Kubespray:** Composition of Ansible playbooks, inventory, provisioning tools
2. **Enterprise solution:**
   - **OpenShift:** Red Hat (IBM), hybrid cloud, enterprise container platform
   - **Rancher**: Rancher, multi-cluster orchestration platform
3. **Cloud-based solution:**
   - **Amazon EKS:** Amazon Elastic Container Service (EKS) for Kubernetes
   - **Google Cloud:** Kubernetes Engine
   - **MS Azure:** Azure Kubernetes Service (AKS)
   - **IBM Cloud:** IBM Cloud Kubernetes service

References:

- **실습용 Host Image: Ubuntu Desktop 16.04, CentOS 7, Ubuntu Server 18.04**

2. 설치 개요

2) Hypervisor

- **Type 1 Hypervisor 솔루션**

1. Type 1 Hypervisor:
   - Windows Hyper-V 2019
   - vSphere 6.7
   - XenServer 7.6
   - KVM

Feature	Windows Hyper-V 2019	vSphere 6.7	XenServer 7.6	KVM
RAM/Host	24TB	12 TB	5TB	12TB
RAM/VM	12 TB for generation 2; 1 TB for generation 1	6 TB	1.5TB	6 TB
CPUs/VM	240 for generation 2; 64 for generation 1;	128	32	240
VM Disk	64 TB for VHDX format; 2040 GB for VHD format	62TB	2TB	10TB
VM Live Migration	Yes	Yes	Yes	Yes
VM Replication supports	Yes	Yes	Yes	Yes
Overcommit resources	No	Yes	No	Yes
Disk I/O Throttling	Yes	Yes	Yes	Yes
Hot plug of virtual resources	Yes	Yes	Yes	Yes

References:

2. 설치 개요

2) Hypervisor

- **Type 2 Hypervisor 솔루션**

	Oracle	VMware				
	VirtualBox 5.2.22	Player 15	Workstation 15	Fusion 11	Fusion Pro 11	ESXi 6.7
Hypervisor Type	2	2	2	2	2	1
Licensing	Free	Free	From 250 $	80 $	160 $	Free/Paid (from 495$)
Virtualization	Hardware + Software	Hardware	Hardware	Hardware	Hardware	Hardware
Host OS	Linux, Windows, Solaris, macOS, FreeBSD	Linux, Windows	Linux, Windows	macOS	macOS	---
Guest OS	Linux, Windows, Solaris, FreeBSD, macOS	Linux, Windows, Solaris, FreeBSD	Linux, Windows, Solaris, FreeBSD	Linux, Windows, Solaris, FreeBSD, macOS	Linux, Windows, Solaris, FreeBSD, macOS	Linux, Windows, Solaris, FreeBSD
Shared Folders	Yes	Yes	Yes	Yes	Yes	No
Seamless mode/Unity	Yes	Yes	Yes	Yes	Yes	No
VM snapshots	Yes	No	Yes	Yes	Yes	Yes
USB for VMs	With Extension Pack	Out of the box	Out of the box	Out of the box	Out of the box	Out of the box
3D graphics in VMs	DirectX 9, OpenGL 3.0	DirectX 10, OpenGL 3.3	DirectX 10, OpenGL 3.3	DirectX 10, OpenGL 3.3	DirectX 10, OpenGL 3.3	DirectX 10, OpenGL 3.3
Max.VM video memory	128 MB	2 GB	2 GB	2 GB	2 GB	2 GB
Virtual Disk Format	VDI, VMDK, VHD, HDD*	VMDK	VMDK	VMDK	VMDK	VMDK
Linked clones support	Yes	No	Yes	No	Yes	No*
Shared storage support	iSCSI, NFS, SMB (CIFS)	No*	No*	No*	No*	iSCSI, NFS, Fibre Channel
VM Live Migration	Yes (Teleportation)	No	No	No	No	Yes (vMotion)
Centralized Management	PhpVirtualBox	No	No	No	No	vCenter
VM Encryption	Yes, with Ext. Pack	Yes (limited)	Yes	Yes (limited)	Yes	Yes
Memory Ballooning	Yes	-	Yes	Yes	Yes	Yes
Clustering	No	No	No	No	No	Yes

References:

- **VMware Workstation Player 15 설치 시연**
- **VirtualBox 6.0 설치 시연**

2. 설치 개요

3) Check Security Policies

- **Unblock Security Policies**

1. External Resources for Opensource Software:
   - apt
   - apt-get
   - snap
   - yum
   - dnf
   - Docker hub
   - Helm

References:

- nano /etc/apt/apt.conf
- vi /etc/yum.conf

I. K8s Clustering 3. Host for MicroK8s

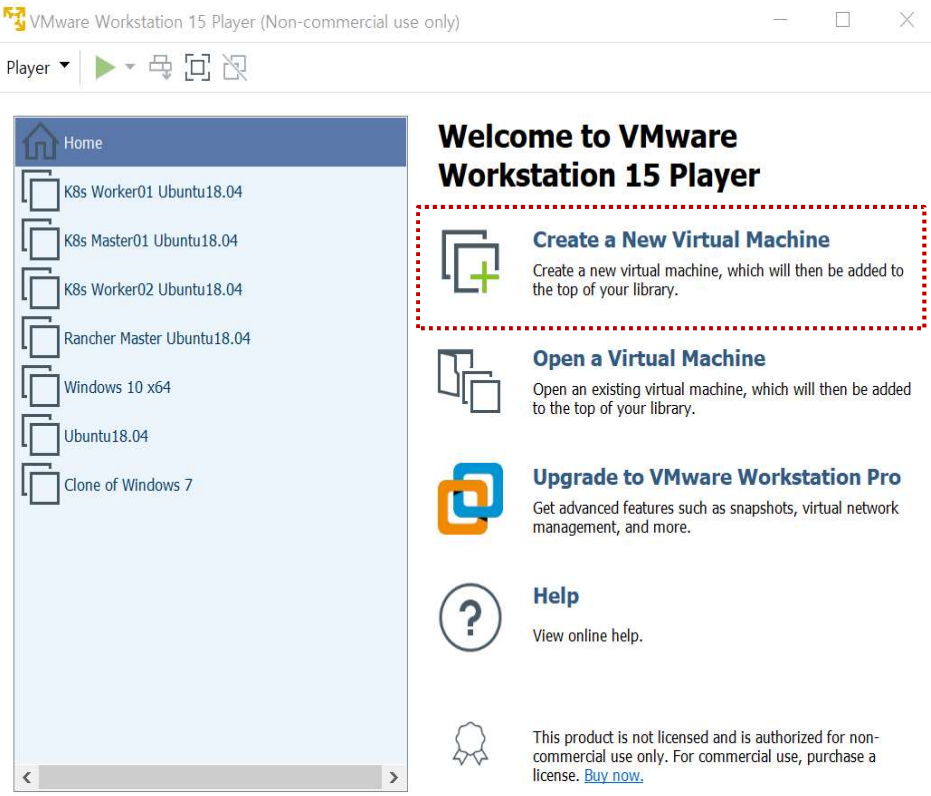
1) Ubuntu Desktop 16.04 설치
2) MicroK8s @ Ubuntu Desktop
3) MicroK8s Dashboard
4) Multi-node MicroK8s

3. Host for MicroK8s

1) Ubuntu Desktop 16.04 설치

- **Ubuntu Desktop 16.04 OS 설치**

1. Create a New Virtual Machine



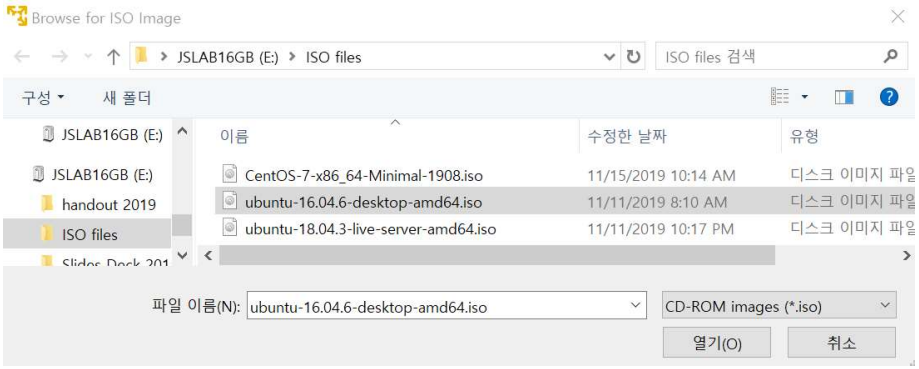
References:

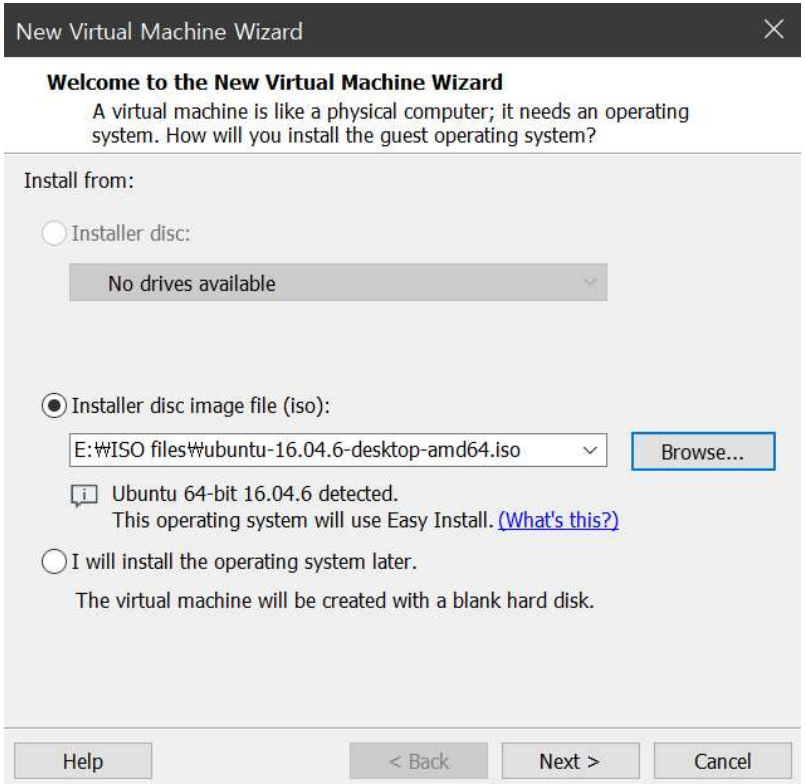
3. Host for MicroK8s

1) Ubuntu Desktop 16.04 설치

- **Ubuntu Desktop 16.04 OS 설치**

1. **Installer Disk Images**





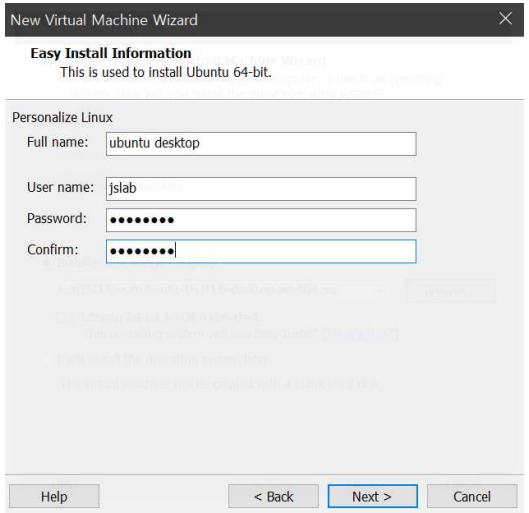
References:

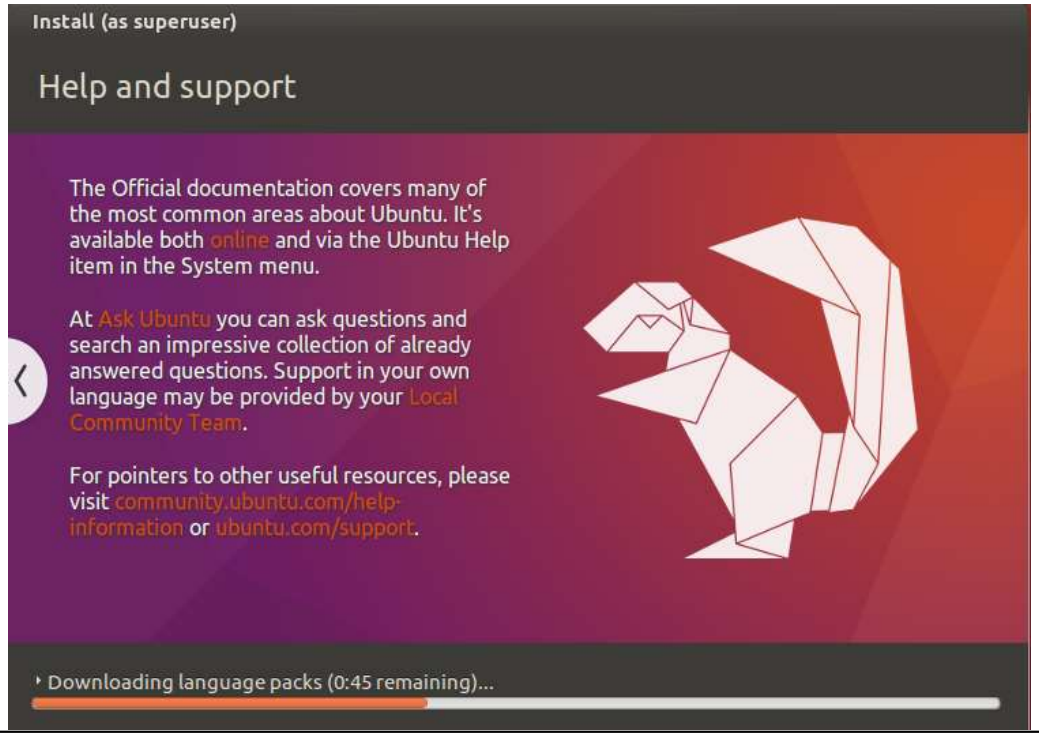
3. Host for MicroK8s

1) Ubuntu Desktop 16.04 설치

- **Ubuntu Desktop 16.04 OS 설치**

1. VM 전원 켜기
2. 계정 설정
3. 시연 따라하기





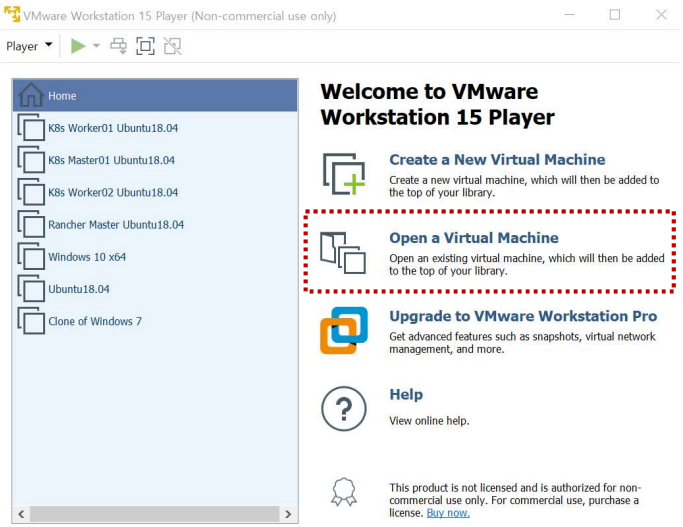
References:

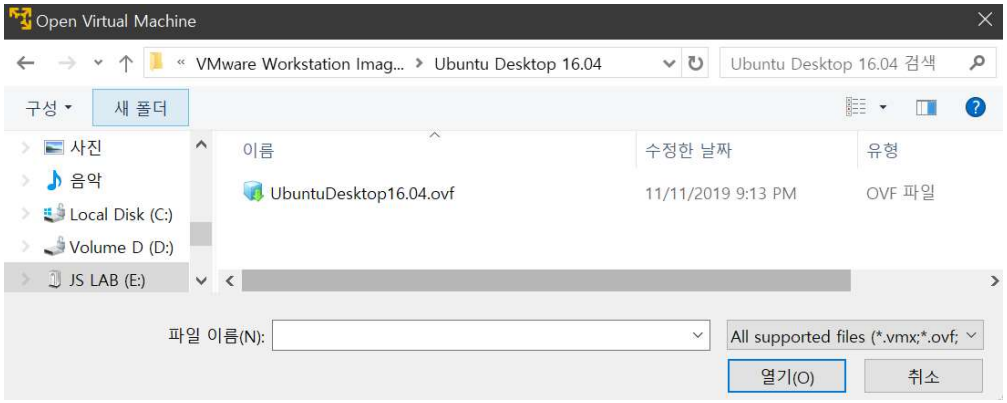
3. Host for MicroK8s

1) Ubuntu Desktop 16.04 설치

- **Ubuntu Desktop 16.04 OS 설치**

1. ISO file 사용
2. VMware Workstation Player 사용
3. 계정ID/암호: jslab / jslab123
4. 설치 후 외부접속 불가능시 “Open a Virtual Machine”





References:

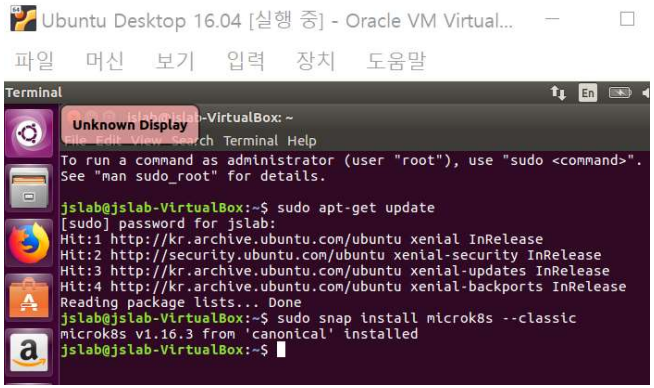
3. Host for MicroK8s

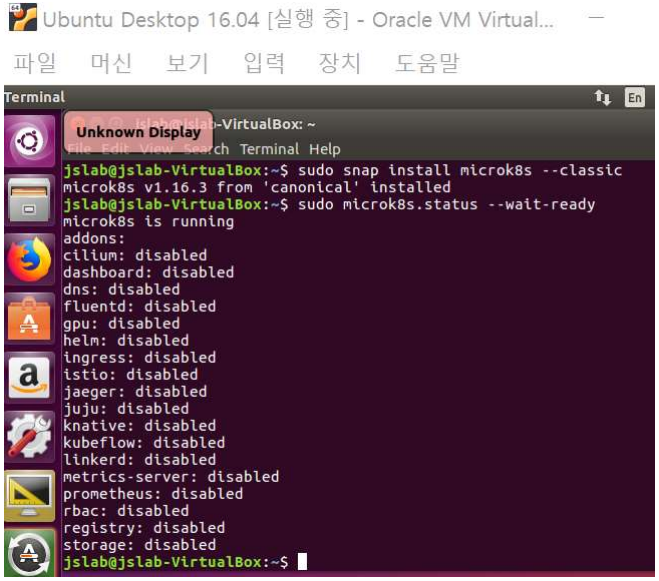
2) MicroK8s @ Ubuntu Desktop

- **Deploying MicroK8s**

1. `sudo snap install microk8s --classic`
   - `sudo snap install microk8s --classic --channel=1.15/stable`
   - `snap install microk8s --classic --channel=1.16/stable`
   - `snap install microk8s --classic --channel=1.16/beta`
2. `sudo microk8s.status --wait-ready`

계정 / 암호
(jslab / jslab123)





References:

- https://microk8s.io/ , https://microk8s.io/docs/setting-snap-channel
  https://tutorials.ubuntu.com/tutorial/install-a-local-kubernetes-with-microk8s#0
- **MicroK8s Basics:** api-server, controller-manager,scheduler, kubelet, cni, kube-proxy

3. Host for MicroK8s

2) MicroK8s @ Ubuntu Desktop

- **Enable addons**

1. **microk8s.disable dashboard dns**

   - **dns:** Deploy DNS. This addon may be required by others, thus we recommend you always enable it.
   - **dashboard:** Deploy kubernetes dashboard as well as grafana and influxdb.
   - **storage**: Create a default storage class. This storage class makes use of the hostpath-provisioner pointing to a directory on the host.
   - **ingress:** Create an ingress controller.
   - **gpu:** Expose GPU(s) to MicroK8s by enabling the nvidia-docker runtime and nvidia-device-plugin-daemonset. Requires NVIDIA drivers to be already installed on the host system.
   - **istio:** Deploy the core Istio services. You can use the microk8s.istioctl command to manage your deployments.
   - **registry:** Deploy a docker private registry and expose it on localhost:32000. The storage addon will be enabled as part of this addon.
   - **jaeger:** Deploy the Jaeger Operator in the “simplest” configuration.
   - **knative:** Adds the Knative middleware to your cluster.
   - **linkerd:** Deploys the linkerd service mesh.
   - **prometheus:** Deploys the Prometheus Operator.
   - **metallb:** Deploys the MetalLB Loadbalancer.

References:

- https://microk8s.io/docs/addons
- https://microk8s.io/docs/configuring-services

3. Host for MicroK8s

2) MicroK8s @ Ubuntu Desktop

- **Accessing grafana dashboards**

1. `microk8s.kubectl get all --all-namespaces`
2. `microk8s.kubectl cluster-info`
   - Kubernetes master: https://127.0.0.1:16443
   - Heapster: https://127.0.0.1:16443/api/v1/namespaces/kube-system/services/heapster/proxy
   - CoreDNS: https://127.0.0.1:16443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy
   - Grafana: https://127.0.0.1:16443/api/v1/namespaces/kube-system/services/monitoring-grafana/proxy
   - InfluxDB: https://127.0.0.1:16443/api/v1/namespaces/kube-system/services/monitoring-influxdb:http/proxy
3. https://127.0.0.1:16443/api/v1/namespaces/kube-system/services/monitoring-grafana/proxy

References:

- https://microk8s.io/ , https://microk8s.io/docs/clustering ,
- https://virtualizationreview.com/articles/2019/01/28/microk8s-how-to-install-and-use-kubernetes.aspx

3. Host for MicroK8s

2) MicroK8s @ Ubuntu Desktop

- **Host your service in Kubernetes**

1. `microk8s.kubectl create deployment microbot --image=dontrebootme/microbot:v1`
2. `microk8s.kubectl scale deployment microbot --replicas=2`
3. `microk8s.kubectl expose deployment microbot --type=NodePort --port=80 --name=microbot-service`
4. `microk8s.kubectl get all --all-namespaces`
5. `http://localhost:32648`

```
> microk8s.kubectl get all --all-namespaces
NAMESPACE     NAME                                                READY   STATUS    RESTARTS   AGE
default       pod/microbot-7dd47b8fd6-4rzt6                       1/1     Running   0          46s
default       pod/microbot-7dd47b8fd6-xr49r                       1/1     Running   0          39s
kube-system   pod/coredns-f7867546d-zb9t5                         1/1     Running   0          10m
kube-system   pod/heapster-v1.5.2-844b564688-5bpzs                4/4     Running   0          8m22s
kube-system   pod/kubernetes-dashboard-7d75c474bb-jcglw           1/1     Running   0          10m
kube-system   pod/monitoring-influxdb-grafana-v4-6b6954958c-nc6bq 2/2     Running   0          10m

NAMESPACE     NAME                         TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)                  AGE
default       service/kubernetes           ClusterIP   10.152.183.1     <none>        443/TCP                  10m
default       service/microbot-service     NodePort    10.152.183.69    <none>        80:32648/TCP             25s
kube-system   service/heapster             ClusterIP   10.152.183.7     <none>        80/TCP                   10m
kube-system   service/kube-dns             ClusterIP   10.152.183.10    <none>        53/UDP,53/TCP,9153/TCP   10m
kube-system   service/kubernetes-dashboard ClusterIP   10.152.183.64    <none>        443/TCP                  10m
kube-system   service/monitoring-grafana   ClusterIP   10.152.183.3     <none>        80/TCP                   10m
kube-system   service/monitoring-influxdb  ClusterIP   10.152.183.203   <none>        8083/TCP,8086/TCP        10m

NAMESPACE     NAME                                        READY   UP-TO-DATE   AVAILABLE   AGE
default       deployment.apps/microbot                    2/2     2            2           46s
kube-system   deployment.apps/coredns                     1/1     1            1           10m
kube-system   deployment.apps/heapster-v1.5.2             1/1     1            1           10m
kube-system   deployment.apps/kubernetes-dashboard        1/1     1            1           10m
kube-system   deployment.apps/monitoring-influxdb-grafana-v4 1/1  1            1           10m

NAMESPACE     NAME                                                  DESIRED   CURRENT   READY   AGE
default       replicaset.apps/microbot-7dd47b8fd6                   2         2         2       46s
kube-system   replicaset.apps/coredns-f7867546d                     1         1         1       10m
kube-system   replicaset.apps/heapster-v1.5.2-6b794f77c8            0         0         0       10m
kube-system   replicaset.apps/heapster-v1.5.2-6f5d55456             0         0         0       8m42s
kube-system   replicaset.apps/heapster-v1.5.2-844b564688            1         1         1       8m22s
kube-system   replicaset.apps/kubernetes-dashboard-7d75c474bb       1         1         1       10m
kube-system   replicaset.apps/monitoring-influxdb-grafana-v4-6b6954958c 1     1         1       10m
```

References:

3. Host for MicroK8s

2) MicroK8s @ Ubuntu Desktop

- **Host your service in Kubernetes**

```
1. sudo microk8s.kubectl get nodes
2. sudo microk8s.kubectl get services
3. sudo microk8s.kubectl create deployment kubernetes-bootcamp --image=gcr.io/google-
   samples/kubernetes-bootcamp:v1            # Deploy an app
4. sudo microk8s.kubectl get pods
5. sudo microk8s.enable dns storage
6. microk8s.enable dashboard registry istio [...]
7. snap info microk8s
```

```
jslab@jslab-VirtualBox:~$ sudo microk8s.kubectl get nodes
NAME              STATUS   ROLES    AGE     VERSION
jslab-virtualbox  Ready    <none>   4m14s   v1.16.3
jslab@jslab-VirtualBox:~$ sudo microk8s.kubectl get services
NAME         TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.152.183.1   <none>        443/TCP   4m26s
jslab@jslab-VirtualBox:~$ sudo microk8s.kubectl create deployment kubernetes-bootcamp --image=gcr.io/
google-samples/kubernetes-bootcamp:v1
deployment.apps/kubernetes-bootcamp created
jslab@jslab-VirtualBox:~$ sudo microk8s.kubectl get pod
NAME                                  READY   STATUS    RESTARTS   AGE
kubernetes-bootcamp-69fbc6f4cf-bqddx  1/1     Running   0          34s
```

References:

3. Host for MicroK8s

2) MicroK8s @ Ubuntu Desktop

- **Integrated commands**

1. **microk8s.status:** Provides an overview of the MicroK8s state (running / not running) as well as the set of enabled addons
2. **microk8s.enable:** Enables an addon
3. **microk8s.disable:** Disables an addon
4. **microk8s.kubectl:** Interact with kubernetes
5. **microk8s.config:** Shows the kubernetes config file
6. **microk8s.istioctl:** Interact with the istio services; needs the istio addon to be enabled
7. **microk8s.inspect:** Performs a quick inspection of the MicroK8s installation
8. **microk8s.reset:** Resets the infrastructure to a clean state
9. **microk8s.stop:** Stops all kubernetes services
10. **microk8s.start:** Starts MicroK8s after it is being stopped
11. **microk8s.stop**

References:

3. Host for MicroK8s

2) MicroK8s @ Ubuntu Desktop

- **snap info microk8s**

1. `snap info microk8s`
2. `sudo microk8s.kubectl get nodes`
3. `sudo microk8s.kubectl get services`

```
jslab@ubuntu:~$ snap info microk8s

name:      microk8s
summary:   Kubernetes for workstations and appliances
publisher: Canonical✓
contact:   https://github.com/ubuntu/microk8s
license:   unset
description: |
  MicroK8s is a small, fast, secure, single node Kubernetes that installs on just
  about any Linux box. Use it for offline development, prototyping, testing, or
  use it on a VM as a small, cheap, reliable k8s for CI/CD. It's also a great k8s
  for appliances - develop your IoT apps for k8s and deploy them to MicroK8s on
  your boxes.
commands:
  - microk8s.add-node
  - microk8s.cilium
  - microk8s.config
  - microk8s.ctr
  - microk8s.disable
  - microk8s.enable
  - microk8s.helm
  - microk8s.inspect
  - microk8s.istioctl
  - microk8s.join
  - microk8s.kubectl
  - microk8s.leave
  - microk8s.linkerd
  - microk8s.remove-node
  - microk8s.reset
  - microk8s.start
  - microk8s.status
  - microk8s.stop
services:
  microk8s.daemon-apiserver:          simple, enabled, active
  microk8s.daemon-apiserver-kicker:   simple, enabled, active
  microk8s.daemon-cluster-agent:      simple, enabled, active
  microk8s.daemon-containerd:         simple, enabled, active
  microk8s.daemon-controller-manager: simple, enabled, active
  microk8s.daemon-etcd:               simple, enabled, active
  microk8s.daemon-flanneld:           simple, enabled, active
  microk8s.daemon-kubelet:            simple, enabled, active
  microk8s.daemon-proxy:              simple, enabled, active
  microk8s.daemon-scheduler:          simple, enabled, active
snap-id:      EaXqgt1lyCaxKaQCU349mlodBkDCXRcg
tracking:     stable
refresh-date: 4 days ago, at 03:45 PST
installed:    v1.16.2 (993) 187MB classic
```

References:

3. Host for MicroK8s

3) MicroK8s Dashboard

- **MicroK8s Dashboard with Ubuntu Desktop**

1. `sudo microk8s.enable dashboard`
2. `sudo -i`

```
jslab@ubuntu:~$ sudo microk8s.enable dashboard
Applying manifest
serviceaccount/kubernetes-dashboard unchanged
service/kubernetes-dashboard unchanged
secret/kubernetes-dashboard-certs unchanged
secret/kubernetes-dashboard-csrf unchanged
secret/kubernetes-dashboard-key-holder unchanged
configmap/kubernetes-dashboard-settings unchanged
role.rbac.authorization.k8s.io/kubernetes-dashboard unchanged
clusterrole.rbac.authorization.k8s.io/kubernetes-dashboard unchanged
rolebinding.rbac.authorization.k8s.io/kubernetes-dashboard unchanged
clusterrolebinding.rbac.authorization.k8s.io/kubernetes-dashboard unchanged
deployment.apps/kubernetes-dashboard unchanged
service/dashboard-metrics-scraper unchanged
deployment.apps/dashboard-metrics-scraper unchanged
service/monitoring-grafana unchanged
service/monitoring-influxdb unchanged
service/heapster unchanged
deployment.apps/monitoring-influxdb-grafana-v4 unchanged
serviceaccount/heapster unchanged
clusterrolebinding.rbac.authorization.k8s.io/heapster unchanged
configmap/heapster-config unchanged
configmap/eventer-config unchanged
deployment.apps/heapster-v1.5.2 unchanged

If RBAC is not enabled access the dashboard using the default token retrieved with:

token=$(microk8s.kubectl -n kube-system get secret | grep default-token | cut -d " " -f1)
microk8s.kubectl -n kube-system describe secret $token

In an RBAC enabled setup (microk8s.enable RBAC) you need to create a user with restricted
permissions as shown in:
https://github.com/kubernetes/dashboard/blob/master/docs/user/access-control/creating-sample-user.md

jslab@ubuntu:~$
```

References:

3. Host for MicroK8s

3) MicroK8s Dashboard

- **MicroK8s Dashboard with Ubuntu Desktop**

1. `sudo token=$(microk8s.kubectl -n kube-system get secret | grep default-token | cut -d " " -f1)`
2. `microk8s.kubectl -n kube-system describe secret $token`

```
jslab@ubuntu:~$ token=$(microk8s.kubectl -n kube-system get secret | grep default-token | cut -d " " -f1)
jslab@ubuntu:~$ microk8s.kubectl -n kube-system describe secret $token
Name:         default-token-k9dzd
Namespace:    kube-system
Labels:       <none>
Annotations:  kubernetes.io/service-account.name: default
              kubernetes.io/service-account.uid: 1fd3b08c-cf2a-483d-bbc7-5d8b614eb3af

Type:  kubernetes.io/service-account-token

Data
====
ca.crt:     1103 bytes
namespace:  11 bytes
token:
eyJhbGciOiJSUzI1NiIsImtpZCI6InNSRGxTQ0ZMQ2ZkalVCNzR2cWpBYW90QThyZS1fRk1rZ3BKS0JfcnB1T0EifQ.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiwia3ViZ
XJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2UiOiJrdWJlLXN5c3RlbSIsImt1YmVybmV0ZXMuaW8vc2Vydmljc2WFjY291bnQvc2VjcmV0Lm5hbWUiOiJkZWZhdWx0LXRva2VuL
Ws5ZHpkIiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9zZXJ2aWNlLWFjY291bnQubmFtZSI6ImRlZmF1bHQiLCJrdWJlcm5ldGVzLmlvL3NlcnZpY2VhY2NvdW50L3NlcnZpY2UtY
WNjb3VudC51aWQiOiIxZmQzYjA4Yy1jZjJhLTQ4M2QtYmJjNy01ZDhiNjE0ZWIzYWYiLCJzdWIiOiJzeXN0ZW06c2VydmljZWFjY291bnQ6a3ViZS1zeXN0ZW06ZGVmYXVsdCJ9.e_3JWodE
hVbSjaPv-
8LUSZSw3vTNGw9gwZ45B7CKEMauDzhBP2TIT1rJBiINz_RIG5BL94DvQNIO5XiHCHskh3CSZp34T7tQcj1EsIUn1_0NSrvkswfpzXuWjJ9YyEPfRLtNtEevtoMyNybtBH6Cq1dwv0RbzS9B0
wxkPeG5q2GHECQcHXvf9bTeMeR0tGSea2m_X5cfz31W49ErzXWhNIC-
roxQLFr8ZSTMgZ8RotcyswPlHZ0u9RP0PZKGe2LV6o0RSGOoyZNLyhe1FmTg_Y0__KzHKS7cnOq6x2JBNqLTfS5ER7a6m7OEJ2IgvjcGZOSaQO6qn9UD0h1OVmqSTw
jslab@ubuntu:~$
```

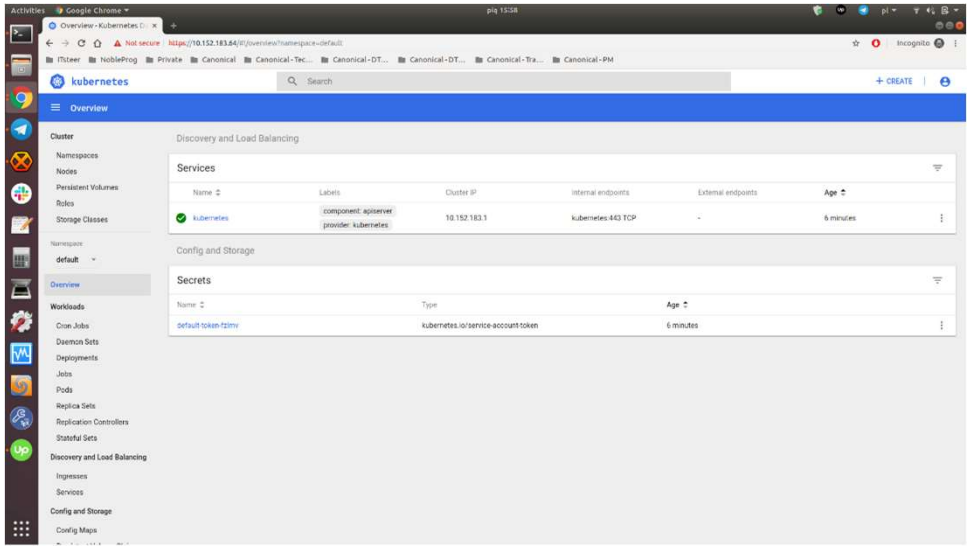
References:

3. Host for MicroK8s

3) MicroK8s Dashboard

- **MicroK8s Dashboard with Ubuntu Desktop**

1. `sudo microk8s.enable rbac`
2. `sudo microk8s.kubectl proxy --accept-hosts=.* --address=0.0.0.0`
3. `http://127.0.0.1:8001/api/v1/namespaces/kube-system/services/https:kubernetes-dashboard:/proxy/`



References:

- https://virtualizationreview.com/articles/2019/01/30/microk8s-part-2-how-to-monitor-and-manage-kubernetes.aspx?m=1

3. Host for MicroK8s

4) Multi-node MicroK8s

- **Multi-node MicroK8s**

1. `microk8s.add-node`
2. `microk8s.kubectl get no`
3. `microk8s.remove-node 10.22.254.79`
4. `microk8s.leave`

```
microk8s.add-node
Join node with: microk8s.join ip-172-31-20-243:25000/DDOkUupkmaBezNnMheTBqFYHLWINGDbf

If the node you are adding is not reachable through the default
interface you can use one of the following:

 microk8s.join 10.1.84.0:25000/DDOkUupkmaBezNnMheTBqFYHLWINGDbf
 microk8s.join 10.22.254.77:25000/DDOkUupkmaBezNnMheTBqFYHLWINGDbf
```

References:

- https://microk8s.io/docs/clustering

3. Host for MicroK8s

5) Multi-node MicroK8s

- **쿠버네티스의 스케줄링**

1. kubectl apply -f https://k8s.io/examples/controllers/daemonset.yaml
   - running a node monitoring daemon on every node
   - Prometheus Node Exporter
   - Sysdig Agent
   - Collectd
   - Dynatrace OneAgent
   - AppDynamics Agent
   - Datadog agent
   - New Relic agent
   - Ganglia gmond
   - Instana Agent.

```yaml
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: fluentd-elasticsearch
  namespace: kube-system
  labels:
    k8s-app: fluentd-logging
spec:
  selector:
    matchLabels:
      name: fluentd-elasticsearch
  template:
    metadata:
      labels:
        name: fluentd-elasticsearch
    spec:
      tolerations:
      - key: node-role.kubernetes.io/master
        effect: NoSchedule
      containers:
      - name: fluentd-elasticsearch
        image: quay.io/fluentd_elasticsearch/fluentd:v2.5.2
        resources:
          limits:
            memory: 200Mi
          requests:
            cpu: 100m
            memory: 200Mi
        volumeMounts:
        - name: varlog
          mountPath: /var/log
        - name: varlibdockercontainers
          mountPath: /var/lib/docker/containers
          readOnly: true
      terminationGracePeriodSeconds: 30
      volumes:
      - name: varlog
        hostPath:
          path: /var/log
      - name: varlibdockercontainers
        hostPath:
          path: /var/lib/docker/containers
```

A DaemonSet ensures that all (or some) Nodes run a copy of a Pod.

3. Host for MicroK8s

5) Multi-node MicroK8s

- **쿠버네티스의 스케줄링**

1. 노드 라벨 설정 명령어 (예): `kubectl label nodes <NODE_NAME> disktype=ssd`

```yaml
- matchExpressions:
  - key: kubernetes.io/e2e-az-name
    operator: In
    values:
    - e2e-az2
    - e2e-az2
  - key: kubernetes.io/region
    operator: In
    values:
    - ap-northeast-1
    - ap-northeast-2
```

matchExpression에 여러 개 조건이 정의될 시, 모든 조건을 만족해야 Pod 생성

```yaml
nodeSelectorTerms:
- matchExpressions:
  - key: kubernetes.io/e2e-az-name
    operator: In
    values:
    - e2e-az2
    - e2e-az2
- matchExpressions:
  - key: disktype
    operator: In
    values:
    - ssd
    - hdd
```

2개 이상의 macthExpressions 정의 시, 하나의 matchExpressions 만족해도 됨

```yaml
# Container Spec
spec:
  affinity:
    nodeAffinity:
      preferredDuringSchedulingIgnoredDuringExecution:
      - weight: 50
        preference:
          matchExpressions:
          - key: disktype
            operator: In
            values:
            - ssd
  containers:
    - name: kubernetes-dev
```

weight 값은 1에서 100까지의 값을 사용할 수 있으며, 이 값은 할당 가능한 노드들을 최종적으로 선정한 뒤, 최종적으로 Pod를 할당할각 노드에 대해 점수를 매길 때 사용

```yaml
# Container Spec
spec:
  nodeName: ip-10-43-0-30.ap-northeast-2.compute.internal
  containers:
    - name: kubernetes-dev
      image: alicek106/pycharm-remote-builder:latest
```

특정 노드 선택

```yaml
# Container Spec
spec:
  nodeSelector:
    disktype: ssd
  containers:
    - name: kubernetes-dev
      image: alicek106/pycharm-remote-builder:latest
```

라벨 'ssd'를 가진 노드에 Pod가 스케줄링 (노드 이름에 종속적이지는 않도록 nodeName 대신 nodeSelector를 사용)

```yaml
# Container Spec
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: kubernetes.io/e2e-az-name
            operator: In
            values:
            - e2e-az1
            - e2e-az2
  containers:
    - name: kubernetes-dev
```

키가 kubernetes.io/e2e-az-name 이고 값이 e2e-az1, e2e-az2 인 라벨을 가지는 노드에 Pod를 할당

http://blog.naver.com/PostView.nhn?blogId=alice_k106&logNo=221511412970&parentCategoryNo=&categoryNo=20&viewDate=&isShowPopularPosts=false&from=postView

I. K8s Clustering 4. Host for Kubeadm

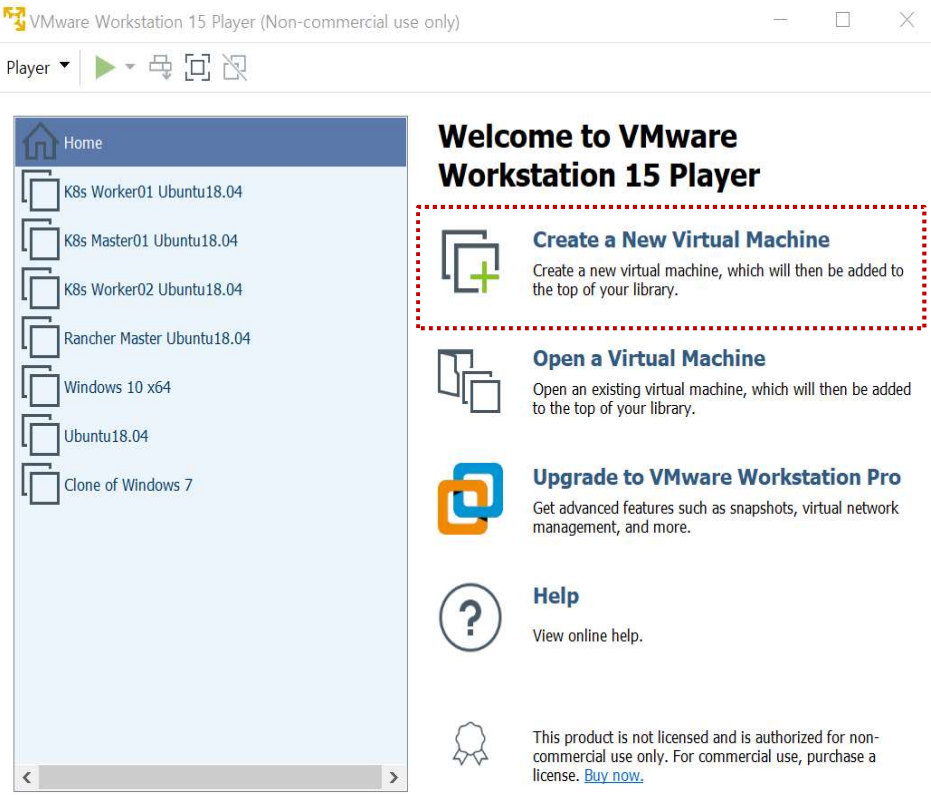
1) CentOS 7 설치
2) master / worker 호스트 노드 설정
3) Staic IP 설정 (CentOS 7)
4) Docker Installation (CentOS 7)
5) K8s Installation (CentOS 7)
   - K8s 설치 준비
   - 쿠버네티스 YUM 리포지토리 설정
   - 마스터 컴포넌트 설치
   - 워커노드 컴포넌트 설치
6) K8s Installation (Ubuntu Server)

4. Host for Kubeadm

1) CentOS 7 설치

- **CentOS 7 설치**

1. Create a New Virtual Machine



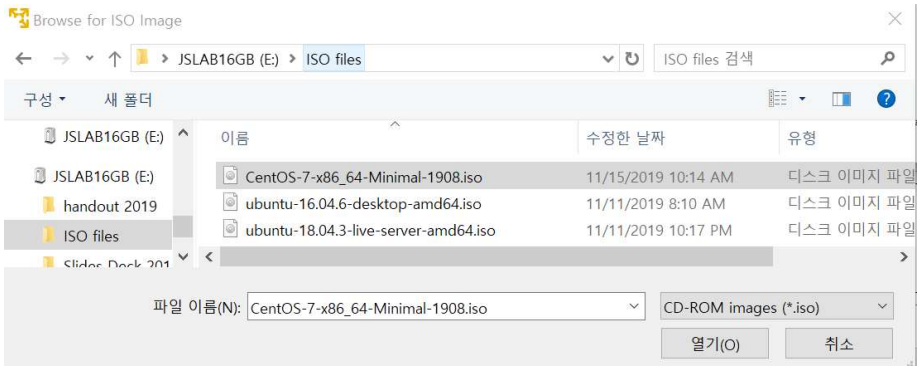
References:

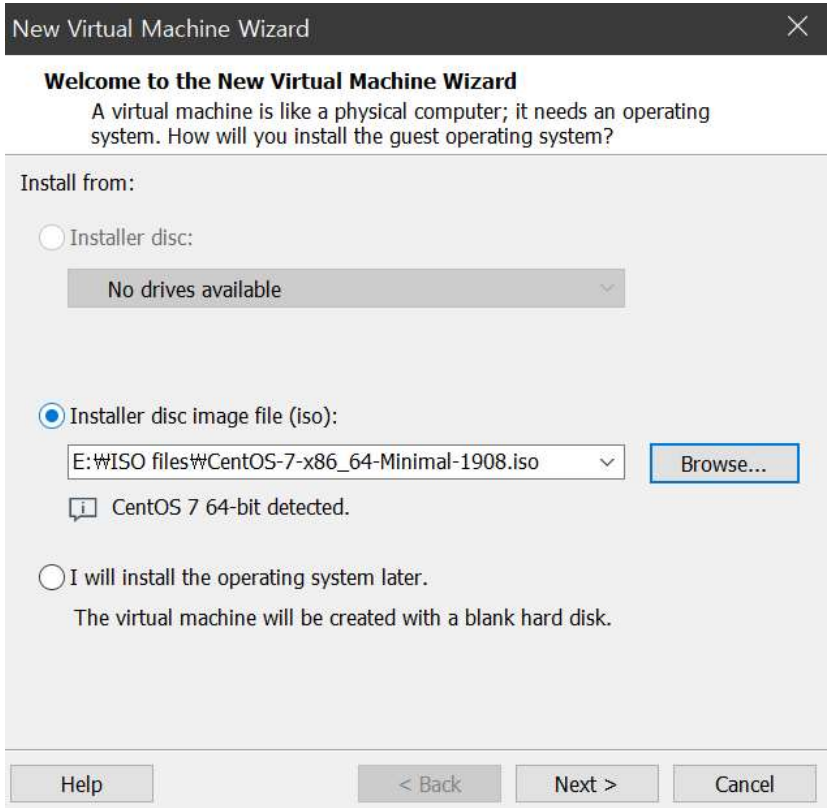
4. Host for Kubeadm

1) CentOS 7 설치

- **CentOS 7 설치**

1. **Installer Disk Images**





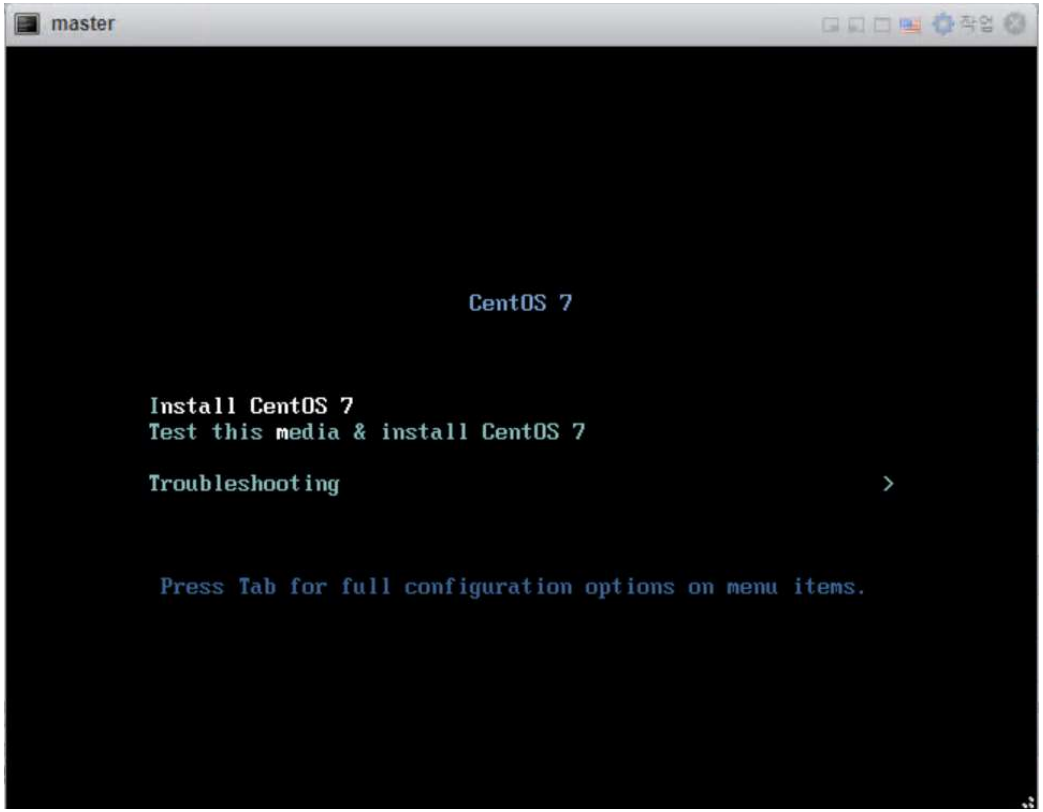
References:

4. Host for Kubeadm

1) CentOS 7 설치

- **CentOS 7 설치**

1. VM 전원 켜기
2. Install CentOS 7
3. 시연 따라하기



References:

4. Host for Kubeadm

2) master / worker 호스트 노드 설정

- **master / worker 호스트 노드 설정 (CentOS 7)**

1. ifconfig
2. # 각 호스트 테이블 동일 설정 계획
   - 192.168.0.xxa master01 # 172.16.88.142
   - 192.168.0.xxb master02
   - 192.168.0.xxc master03
   - 192.168.0.yya worker01 # 172.16.88.143
   - 192.168.0.yyb worker02
   - 192.168.0.yyc worker03
3. sudo hostnamectl set-hostname master # @ master node, 'nmtui'에서 설정 가능
4. sudo hostnamectl set-hostname worker # @ worker node, 'nmtui'에서 설정 가능
5. su - # 각 호스트 이름 변경 확인
6. nano /etc/hosts # 각 호스트의 hosts 파일 내용 동일 설정
   - 192.168.0.xxa master
   - 192.168.0.yya worker

계정 / 암호
(root / jslab123)

	Host Name	IP Address	Memo
Master	master		
Worker 01	worker01		
Worker 02	worker02		
Worker 03	woeker03		

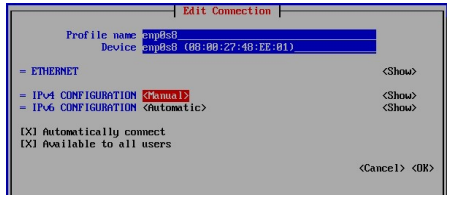
References:
- Multi-console views (Putty/Superputty)
- sudo su (root 권한)
- CentOS 7 다운로드 주소: https://www.centos.org/download/

4. Host for Kubeadm

3) Staic IP 설정 (CentOS 7)

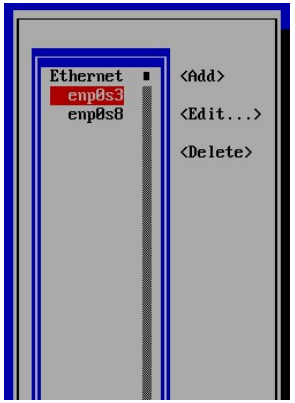
- **CentOS 7 Staic IP 설정 1 of 2**

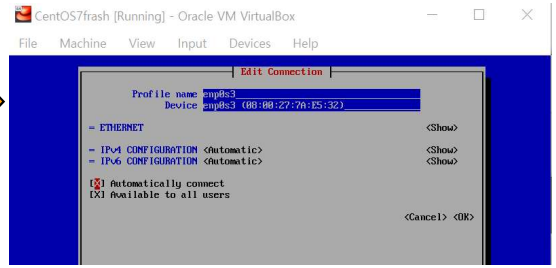
```
1. ip add                                  # 설정한 IP 주소 확인 @ Terminal
2. nmtui                                   # IP 주소 설정 (Tab 키 사용 이동)
3. Automatically Active                    # Automatically Active Interfaces
4. Connection as manual                    # Connection as manual
5. systemctl disable firewalld
6. sudo swapoff -a                         # 각 호스트 확인 manual setting
7. sudo vi /etc/fstab                      # Comment out the swap line ('esc' + ':x')
```



Connection as manual

nmtui 명령어 수행 화면





Activate a connection

References:

- CentOS 7 master / worker01 /worker02 VM 이미지 복사 시에는 MAC reset 고려
- ls -l /sys/class/net , ip addr show NAME
- file named ifcfg-NAME in the directory /etc/sysconfig/network-scripts/

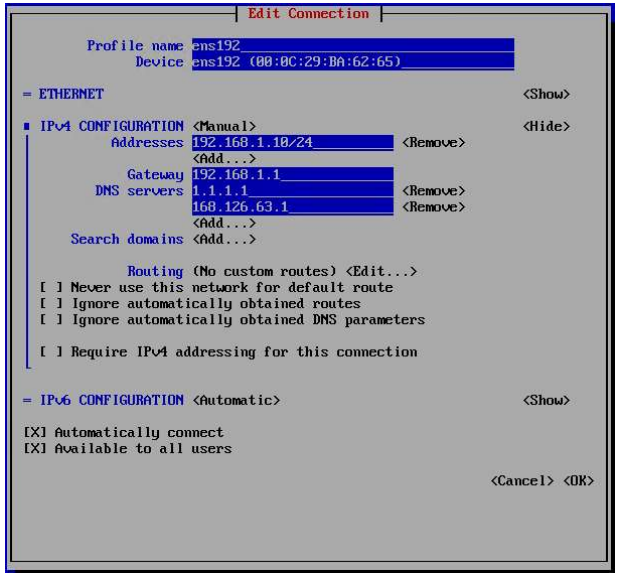
4. Host for Kubeadm

3) Staic IP 설정 (CentOS 7)

- **CentOS 7 Staic IP 설정 (2 of 2)**

1. **nmtui** # IP 주소 설정 192.168.xx.1x (Tab 키 사용 이동)
2. **Edit a connection** # 설정한 IP 주소 확인 @ Terminal, not SSH
3. **Set system hostname** # 설정한 IP 주소 확인 @ Terminal, not SSH
4. **변경 후 Deactivate - Activate a Connection**

nmtui 명령어 수행 화면



Activate a connection

IP 주소 설정

```
[root@ansible ~]# ip add
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default
qlen 1000
    link/ether 00:0c:29:92:60:cc brd ff:ff:ff:ff:ff:ff
    inet 192.168.186.135/24 brd 192.168.186.255 scope global noprefixroute dynamic ens33
       valid_lft 1150sec preferred_lft 1150sec
    inet6 fe80::9df1:2eb5:6bbd:40f/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
[root@ansible ~]#
```

References:

- echo "nameserver 1.1.1.1">> /etc/resolv.conf # 선택
- vi /etc/resolv.conf # dns 주소 1.1.1.1 추가 확인

4. Host for Kubeadm

4) Docker Installation (CentOS 7)

- **Docker Installation (CentOS 7)**

```
1. sudo setenforce 0        # Change SELinux mode to permissive
2. sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
3. sudo yum update -y && yum install -y yum-utils device-mapper-persistent-data lvm2
4. sudo yum-config-manager --add-repo
   https://download.docker.com/linux/centos/docker-ce.repo
5. sudo yum install -y docker-ce
6. sudo systemctl start docker && sudo systemctl enable docker
7. docker version
```

```
[root@localhost ~]# docker version
Client: Docker Engine - Community
 Version:           19.03.5
 API version:       1.40
 Go version:        go1.12.12
 Git commit:        633a0ea
 Built:             Wed Nov 13 07:25:41 2019
 OS/Arch:           linux/amd64
 Experimental:      false
```

```
Server: Docker Engine - Community
 Engine:
  Version:          19.03.5
  API version:      1.40 (minimum version 1.12)
  Go version:       go1.12.12
  Git commit:       633a0ea
  Built:            Wed Nov 13 07:24:18 2019
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          1.2.10
  GitCommit:        b34a5c8af56e510852c35414db4c1f4fa6172339
 runc:
  Version:          1.0.0-rc8+dev
  GitCommit:        3e425f80a8c931f88e6d94a8c831b9d5aa481657
 docker-init:
  Version:          0.18.0
  GitCommit:        fec3683
```

References:

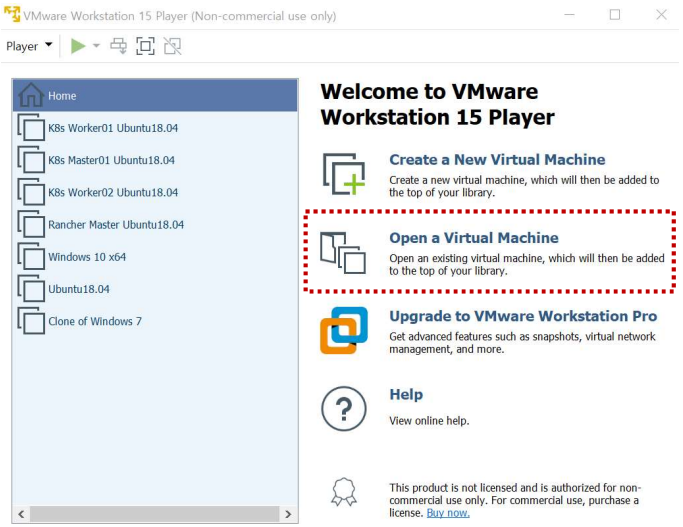
- **sudo apt install docker.lo** **# @ Ubuntu**

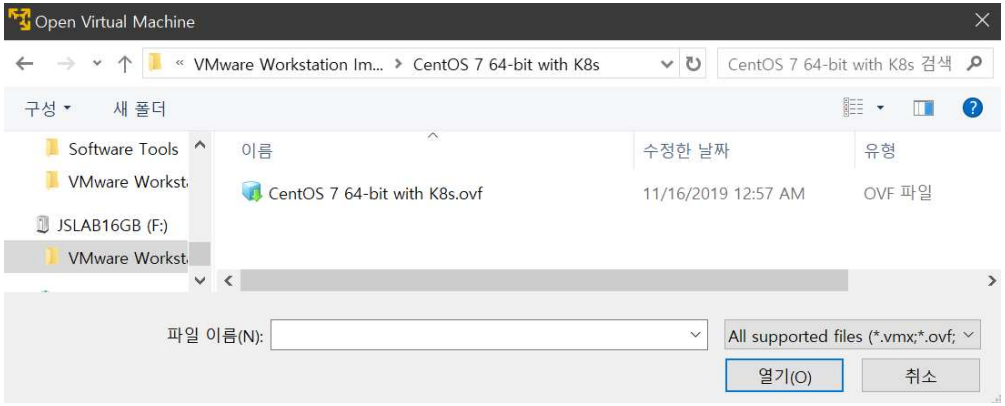
4. Host for Kubeadm

1) CentOS 7 설치

- **CentOS 7 설치**

1. 설치 후 외부접속 불가능시 “Open a Virtual Machine”
2. 클러스터링 Host 수에 따른 반복 과정
3. Host 이름 변경
4. 계정ID/암호: root / jslab123





References:

4. Host for Kubeadm

5) K8s Installation (CentOS 7)

- **K8s 설치 준비**

```
cat <<EOF >  /etc/sysctl.d/k8s.conf
net.ipv4.ip_forward = 1
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
EOF
sudo sysctl --system
systemctl stop firewalld
systemctl disable firewalld
sudo swapoff -a                          # 각 호스트 확인 manual setting
sudo vi /etc/fstab                       # Comment out the swap line ('esc' + ':x')
reboot
```

References:

- curl -fsSL https://get.docker.com/ | sh # @ General for New
- sudo apt install docker.lo # @ Ubuntu

4. Host for Kubeadm

5) K8s Installation (CentOS 7)

- **쿠버네티스 YUM 레포지토리 설정**

```
cat <<EOF > /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-x86_64
enabled=1
gpgcheck=1
repo_gpgcheck=1
gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg
https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg
exclude=kube*
EOF
yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes
systemctl enable kubelet && systemctl start kubelet
```

References:

- curl -fsSL https://get.docker.com/ | sh # @ General for New
- sudo apt install docker.lo # @ Ubuntu

4. Host for Kubeadm

5) K8s Installation (CentOS 7)

- **마스터 컴포넌트 설치**

```
1. kubeadm init --pod-network-cidr=10.244.0.0/16
2. mkdir -p $HOME/.kube
3. sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
4. sudo chown $(id -u):$(id -g) $HOME/.kube/config
5. export KUBECONFIG=/etc/kubernetes/admin.conf
6. kubectl apply -f
   https://raw.githubusercontent.com/coreos/flannel/bc79dd1505b0c8681ece4de4c0d86c5cd2643275/Documentation/kube-flannel.yml
```

References:

- **예: kubeadm init --ignore-preflight-errors=all --pod-network-cidr 172.31.0.0/16 --service-cidr 10.128.0.0/12 --service-dns-domain "cluster.local" --apiserver-advertise-address 192.168.0.60 --token-ttl 0**
  - ✓ 설치시 생성되는 토큰(bootstrap token)은 24시간 이후에 자동 삭제 되며, 이후에 노드 추가 등의 작업을 위해서 'kubeadm token create' 으로 새로운 토큰을 생성해 주어야 한다. '--token-ttl 0' 옵션은 토큰이 만료되지 않도록 설정

4. Host for Kubeadm

5) K8s Installation (CentOS 7)

- **워커노드 컴포넌트 설치**

```
1. kubeadm join 172.16.1.100:6443 --token yrc47a.55b25p2dhe14pzd1 --discovery-token-
   ca-cert-hash
   sha256:2a7a31510b9a0b0da1cf71c2c29627b40711cdd84be12944a713ce2af2d5d148
2.  kubectl get nodes
```

References:

- https://javacan.tistory.com/entry/k8s-install-in-centos7

4. Host for Kubeadm

6) K8s Installation (Ubuntu Server)

- **K8s Installation (Ubuntu Server)**

```
1. sudo apt-get update && sudo apt-get install -y apt-transport-https curl
2. curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -
3. cat <<EOF | sudo tee /etc/apt/sources.list.d/kubernetes.list
4. deb https://apt.kubernetes.io/ kubernetes-xenial main
5. EOF
6. sudo apt-get update
7. sudo apt-get install -y kubelet kubeadm kubectl
8. sudo apt-mark hold kubelet kubeadm kubectl
9. sudo kubeadm init --pod-network-cidr=172.16.0.0/16 --apiserver-advertise-
   address=172.16.88.xxx
10. mkdir -p $HOME/.kube
11. sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
12. sudo chown $(id -u):$(id -g) $HOME/.kube/config
13. kubectl apply -f
   https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-
   flannel.yml
```

Setup at master and worker

Setup at master

Admin Key

References:

- https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/
- kubectl apply -f https://docs.projectcalico.org/v2.6/getting-started/kubernetes/installation/hosted/kubeadm/1.6/calico.yaml

4. Host for Kubeadm

6) K8s Installation (Ubuntu Server)

- **K8s 설치**

```
1. sudo su
2. apt-get update && apt-get install -y apt-transport-https curl
3. curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key add -
4. echo deb http://apt.kubernetes.io/ kubernetes-xenial main >
   /etc/apt/sources.list.d/kubernetes.list
5. apt-get update
6. apt-get install -y kubeadm=1.12.3-00 kubelet=1.12.3-00
7. swapoff -a
8. sed -i '/swap.img/s/^/#/' /etc/fstab
9. kubeadm init --apiserver-advertise-address=10.0.0.10 --pod-network-
   cidr=192.168.0.0/16 --kubernetes-version=v1.12.0
10. ...
11. kubeadm join 10.0.0.10:6443 --token x7tk20.4hp9x2x43g46ara5 --discovery-token-ca-
   cert-hash sha256:cab2cc0a4912164f45f502ad31f5d038974cf98ed10a6064d6632a07097fad79
```

References:

- https://honggg0801.tistory.com/43

4. Host for Kubeadm

6) K8s Installation (Ubuntu Server)

- **Add worker node @ K8s Cluster (1 of 2)**

```
1. kubeadm token list
2. kubeadm token generate
3. kubeadm join --discovery-token-unsafe-skip-ca-verification --
   token=iss2ag.sobja7vsz9ymslhz 172.16.88.146:6443
4. kubeadm join 172.16.88.142:6443 --token iss2ag.sobja7vsz9ymslhz \
5.     --discovery-token-ca-cert-hash
   sha256:9bdcf63becea3d191c41825fdd9c2d34d540c70dd67d35a5fddb44572dd9838
6. kubectl get nodes
7. kubectl get svc
8. kubectl get svc --all-namespaces
9. kubectl get pods
10. kubectl get pods --all-namespaces
```

References:

- apt install curl / kubectl get nodes / kubectl get pods --all-namespaces
- https://waspro.tistory.com/516

4. Host for Kubeadm

6) K8s Installation (Ubuntu Server)

- **Add worker node @ K8s Cluster (2 of 2)**

```
1. kubeadm token list
2. kubeadm token generate
3. kubeadm j
   token=iss
4. kubeadm j
5.     --dis
   sha256:9b
6. kubectl g
7. kubectl g
8. kubectl g
9. kubectl g
10. kubectl g
```

```
root@master:/home/jslab# kubectl get nodes
NAME     STATUS   ROLES    AGE   VERSION
master   Ready    master   57m   v1.16.2
worker   Ready    <none>   56m   v1.16.2
root@master:/home/jslab# kubectl get svc
NAME                  TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)         AGE
kubernetes            ClusterIP   10.96.0.1       <none>        443/TCP         57m
root@master:/home/jslab# kubectl get svc --all-namespaces
NAMESPACE             NAME                        TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)                  AGE
default               kubernetes                  ClusterIP   10.96.0.1       <none>        443/TCP                  57m
kube-system           kube-dns                    ClusterIP   10.96.0.10      <none>        53/UDP,53/TCP,9153/TCP   57m
root@master:/home/jslab# kubectl get pods
NAME                     READY   STATUS             RESTARTS   AGE
root@master:/home/jslab# kubectl get pods --all-namespaces
NAMESPACE             NAME                                       READY   STATUS            RESTARTS   AGE
kube-system           coredns-5644d7b6d9-g48v8                   1/1     Running           0          57m
kube-system           coredns-5644d7b6d9-rcn4f                   1/1     Running           0          57m
kube-system           etcd-master                                1/1     Running           0          56m
kube-system           kube-apiserver-master                      1/1     Running           0          56m
kube-system           kube-controller-manager-master             1/1     Running           0          56m
kube-system           kube-flannel-ds-amd64-fmlzr                1/1     Running           0          56m
kube-system           kube-flannel-ds-amd64-zvzrw                1/1     Running           0          56m
kube-system           kube-proxy-4sl4j                           1/1     Running           0          57m
kube-system           kube-proxy-p4wnf                           1/1     Running           0          57m
kube-system           kube-scheduler-master                      1/1     Running           0          56m
root@master:/home/jslab#
```

References:

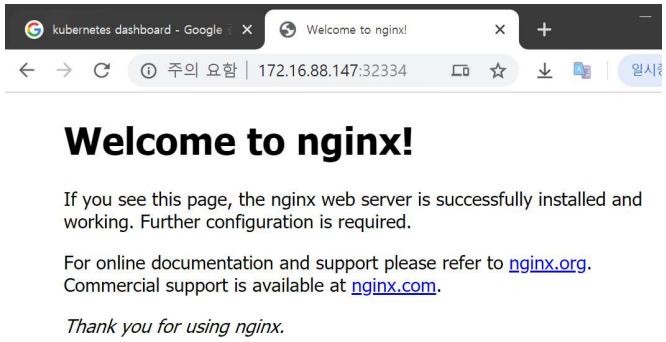
- **apt install curl / kubectl get nodes / kubectl get pods --all-namespaces**
- https://waspro.tistory.com/516

4. Host for Kubeadm

6) K8s Installation (Ubuntu Server)

Deploy App (nginx) (1 of 2)

1. `kubectl create deployment nginx --image=nginx`
   - `deployment “nginx" created`
2. `kubectl create service nodeport nginx --tcp=80:80`
3. `kubectl get pod`
4. `kubectl get svc`
5. type the URL http://172.16.88.147:32334 # worker Node IP




```
root@master:/home/jslab# kubectl create deployment nginx --image=nginx
deployment.apps/nginx created
root@master:/home/jslab# kubectl create service nodeport nginx --tcp=80:80
service/nginx created
root@master:/home/jslab# kubectl get pod
NAME                      READY   STATUS             RESTARTS   AGE
alpine-b4dc77bc6-nr5ln    0/1     CrashLoopBackOff   3          84s
apache-6b5cf8bc7-142h9    0/1     ImagePullBackOff   0          4m14s
nginx-86c57db685-mqc89    1/1     Running            0          20s
root@master:/home/jslab# kubectl get svc
NAME                   TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)         AGE
alpine                 NodePort    10.97.116.192   <none>        80:31934/TCP    82s
apache                 NodePort    10.110.170.16   <none>        80:32468/TCP    4m5s
kubernetes             ClusterIP   10.96.0.1       <none>        443/TCP         38m
kubernetes-dashboard   NodePort    10.103.66.84    <none>        443:31900/TCP   21m
nginx                  NodePort    10.97.75.211    <none>        80:32334/TCP    26s
```

References:

- kubectl get deployments
- kubectl describe deployment nginx
- https://medium.com/@Alibaba_Cloud/how-to-install-and-deploy-kubernetes-on-ubuntu-16-04-6769fd1646db

4. Host for Kubeadm

6) K8s Installation (Ubuntu Server)

- **Deploy the nginx container (2 of 2)**

1. `kubectl get deployments`
2. `kubectl describe deployment`

```
root@master:/home/jslab# kubectl get deployments
NAME     READY   UP-TO-DATE   AVAILABLE   AGE
nginx    1/1     1            1           26m
root@master:/home/jslab# kubectl describe deployment nginx
Name:                   nginx
Namespace:              default
CreationTimestamp:      Mon, 11 Nov 2019 20:22:02 +0000
Labels:                 app=nginx
Annotations:            deployment.kubernetes.io/revision: 1
Selector:               app=nginx
Replicas:               1 desired | 1 updated | 1 total | 1 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=nginx
  Containers:
   nginx:
    Image:        nginx
    Port:         <none>
    Host Port:    <none>
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets:  <none>
NewReplicaSet:   nginx-86c57db685 (1/1 replicas created)
Events:
  Type    Reason             Age   From                   Message
  ----    ------             ----  ----                   -------
  Normal  ScalingReplicaSet  26m   deployment-controller  Scaled up replica set nginx-86c57db685 to 1
```

References:

- https://medium.com/@Alibaba_Cloud/how-to-install-and-deploy-kubernetes-on-ubuntu-16-04-6769fd1646db

4. Host for Kubeadm

6) K8s Installation (Ubuntu Server)

- **Start Minikube**

```
grep -E --color 'vmx|svm' /proc/cpuinfo
curl -Lo minikube
https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64 \
  && chmod +x minikube
sudo mkdir -p /usr/local/bin/
sudo install minikube /usr/local/bin/
minikube start
minikube delete
```

References:

- https://kubernetes.io/ko/docs/concepts/overview/what-is-kubernetes/
- https://kubernetes.io/docs/tasks/tools/install-minikube/

4. Host for Kubeadm

6) K8s Installation (Ubuntu Server)

- **Start Minikube**

```
1. sudo apt-get update
2. sudo apt-get install -y apt-transport-https
3. sudo apt-get install -y virtualbox virtualbox-ext-pack
4. curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -
5. sudo touch /etc/apt/sources.list.d/kubernetes.list
6. echo "deb http://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee -a
   /etc/apt/sources.list.d/kubernetes.list
7. sudo apt-get update
8. sudo apt-get install -y kubectl
9. curl -Lo minikube
   https://storage.googleapis.com/minikube/releases/v0.28.2/minikube-linux-amd64
10. chmod +x minikube && sudo mv minikube /usr/local/bin/
11. minikube start
12. $ kubectl api-versions
```

References:

- https://kubernetes.io/ko/docs/concepts/overview/what-is-kubernetes/
- https://matthewpalmer.net/kubernetes-app-developer/articles/install-kubernetes-ubuntu-tutorial.html

4. Host for Kubeadm

6) K8s Installation (Ubuntu Server)

- **Start Minikube**

```
1. ip link or ifconfig
2. sudo cat /sys/class/dmi/id/product_uuid
3. sudo update-alternatives --set iptables /usr/sbin/iptables-legacy
4. sudo update-alternatives --set ip6tables /usr/sbin/ip6tables-legacy
5. sudo update-alternatives --set arptables /usr/sbin/arptables-legacy
6. sudo update-alternatives --set ebtables /usr/sbin/ebtables-legacy
```

Protocol	Direction	Port Range	Purpose	Used By
TCP	Inbound	6443*	Kubernetes API server	All
TCP	Inbound	2379-2380	etcd server client API	kube-apiserver, etcd
TCP	Inbound	10250	Kubelet API	Self, Control plane
TCP	Inbound	10251	kube-scheduler	Self
TCP	Inbound	10252	kube-controller-manager	Self

Protocol	Direction	Port Range	Purpose	Used By
TCP	Inbound	10250	Kubelet API	Self, Control plane
TCP	Inbound	30000-32767	NodePort Services**	All

References:

- https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/
- **sudo netstat –tulpn | grep LISTEN**

4. Host for Kubeadm

6) K8s Installation (Ubuntu Server)

- **Helm Package Manager**

1. cd /tmp
2. curl https://raw.githubusercontent.com/kubernetes/helm/master/scripts/get > install-helm.sh
3. chmod u+x install-helm.sh
4. ./install-helm.sh
5. kubectl -n kube-system create serviceaccount tiller
6. kubectl create clusterrolebinding tiller --clusterrole cluster-admin --serviceaccount=kube-system:tiller
7. helm init --service-account tiller
8. helm install stable/kubernetes-dashboard --name dashboard-demo
9. kubectl apply -f https://raw.githubusercontent.com/kubernetes/dashboard/v1.10.1/src/deploy/recommended/kubernetes-dashboard.yaml
10. kubectl apply -f https://raw.githubusercontent.com/kubernetes/dashboard/v2.0.0-beta4/aio/deploy/recommended.yaml
11. kubectl proxy --port=8080 --address='0.0.0.0' --disable-filter=true &
12. http://localhost:8001/api/v1/namespaces/default/services/https:kubernetes-dashboard:/proxy/
13. http://localhost:8001/api/v1/namespaces/kubernetes-dashboard/services/https:kubernetes-dashboard:/proxy/
14. https://eksworkshop.com/dashboard/dashboard/

References:

- https://snapcraft.io/install/helm/ubuntu
- https://www.digitalocean.com/community/tutorials/how-to-install-software-on-kubernetes-clusters-with-the-helm-package-manager

I. K8s Clustering 5. Host for Rancher

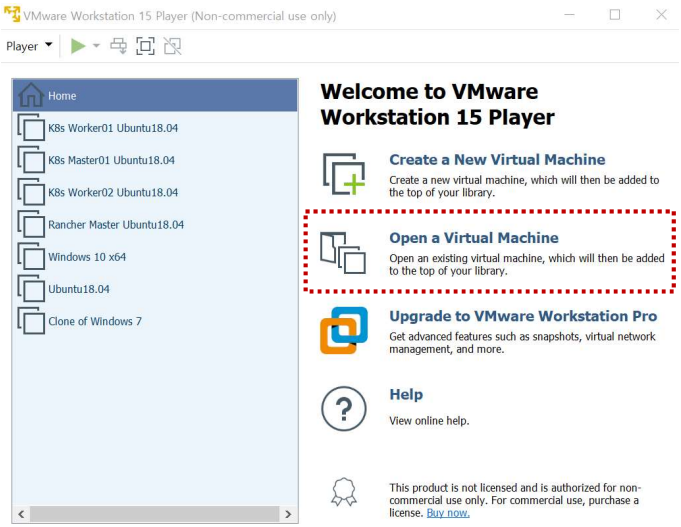
1) Ubuntu Server 18.04 설치
2) master / worker 호스트 노드 설정
3) Static IP 설정
4) Docker Installation @ Multi Hosts
5) Rancher 설치
6) 참고
   - K8s 설치
   - Deploy App (nginx)
   - Helm Package Manager
   - Minikube

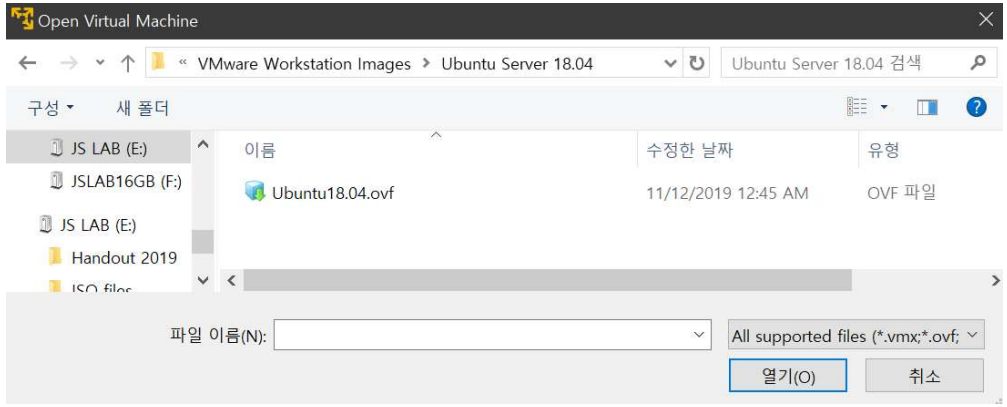
5. Host for Rancher

1) Ubuntu Server 18.04 설치

- **Ubuntu Server 18.04 설치**

1. ISO file 사용
2. VMware Workstation Player 사용
3. 계정ID/암호: root / jslab123
4. 설치 후 외부접속 불가능시 “Open a Virtual Machine”





References:

5. Host for Rancher

2) master / worker 호스트 노드 설정 (Ubuntu Server 18.04)

- **master / worker 호스트 노드 설정 (Ubuntu Server 18.04)**

```
1. ifconfig
2. # 각 호스트 테이블 동일 설정 계획
   ■ 192.168.0.xxa master01                    # 172.16.88.142
   ■ 192.168.0.xxb master02
   ■ 192.168.0.xxc master03
   ■ 192.168.0.yya worker01                    # 172.16.88.143
   ■ 192.168.0.yyb worker02
   ■ 192.168.0.yyc worker03
3. sudo hostnamectl set-hostname master  # @ master node
4. sudo hostnamectl set-hostname worker  # @ worker node
5. su -                                  # 각 호스트 이름 변경 확인
6. nano /etc/hosts                       # 각 호스트의 hosts 파일 내용 동일 설정
   ■ 192.168.0.xxa master
   ■ 192.168.0.yya worker
7. sudo swapoff -a                       # 각 호스트 확인 manual setting
8. sudo nano /etc/fstab                  # Comment out the swap line ('cntl x' + 'y')
```

계정 / 암호
(jslab / jslab123)

References:

- Multi-console views (Putty/Superputty)
- sudo su (root 권한)
- https://medium.com/@Alibaba_Cloud/how-to-install-and-deploy-kubernetes-on-ubuntu-16-04-6769fd1646db

5. Host for Rancher

3) Static IP 설정

Staic IP 설정 (Ubuntu Server 18.04)

```
1. ip a
2. ping aaa.bbb.ccc.255 -b                    # Check reply from gateway IP address
3. sudo nano /etc/netplan/50-cloud-init.yaml  # 16.04의 /etc/network/interfaces 대체
   network:
       ethernets:
           enp0s31f6:
               addresses: [192.168.0.xxa/24]
               gateway4: 192.168.0.2
               nameservers:
                 addresses: [168.126.63.1,8.8.8.8]
                 #  search: [lesstif.com]
                 #            dhcp4: yes
               dhcp4: no
                 #  optional: true
       version: 2
4. sudo netplan apply
```

References:

5. Host for Rancher

4) Docker Installation @ Multi Hosts

- **컨테이너 환경 설치 (Docker)**

```
1. sudo apt-get update
2. sudo apt install docker.io
3. sudo systemctl enable docker
```

References:

5. Host for Rancher

5) Rancher 설치

- **Rancher 설치**

```
1. $ sudo docker run -d --restart=unless-stopped -p 80:80 -p 443:443 rancher/rancher
   docker run -d --restart=unless-stopped \
     -p 80:80 -p 443:443 \
     -v /host/certs:/container/certs \
     -e SSL_CERT_DIR="/container/certs" \
     rancher/rancher:latest
```

References:

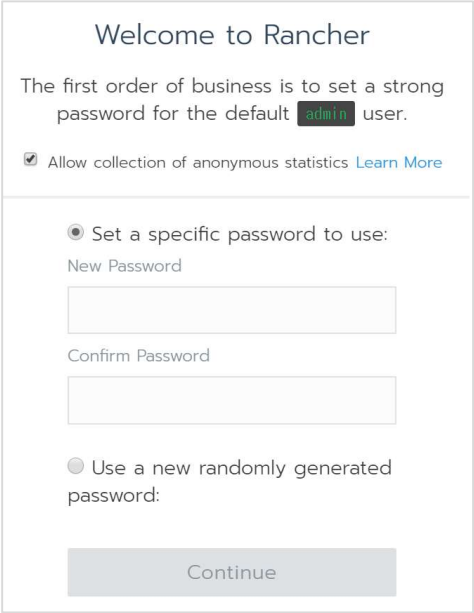
- https://rancher.com/docs/rancher/v2.x/en/installation/single-node/

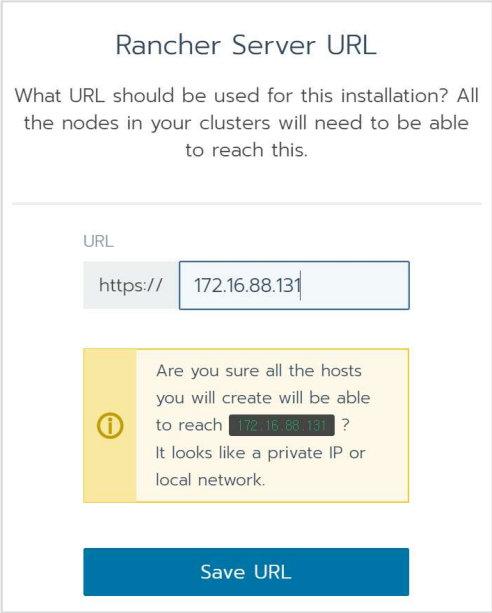
5. Host for Rancher

5) Rancher 설치

- **Dashboard**

1. **http://172.16.88.131:80 @ Web Browser**
2. **admin password 'jslab123'**
3. **Save URL**





References:

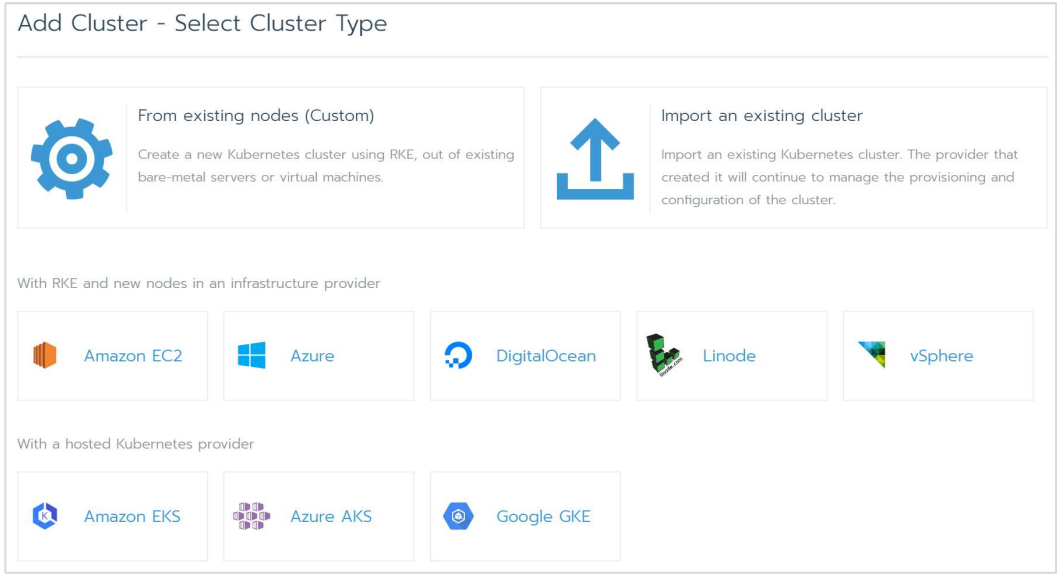
- https://rancher.com/docs/rancher/v2.x/en/installation/single-node/

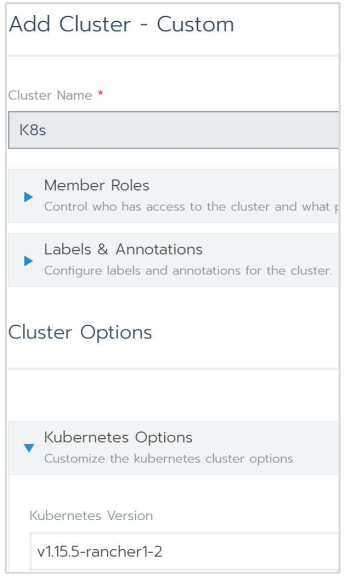
5. Host for Rancher

5) Rancher 설치

- **Rancher Web**

1. **Add cluster**
2. **Save URL**
3. **From existing nodes**
4. **Cluster Name**





References:

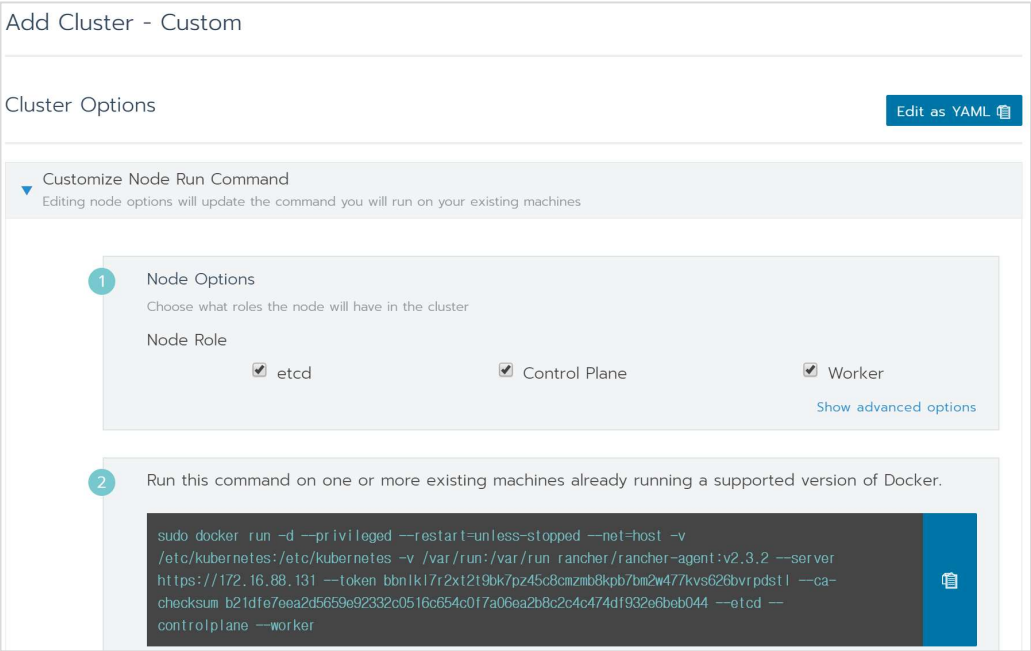
- https://rancher.com/docs/rancher/v2.x/en/installation/single-node/

5. Host for Rancher

5) Rancher 설치

- **Rancher Web**

1. **Cluster Options**
2. **Copy / Paste key**



References:

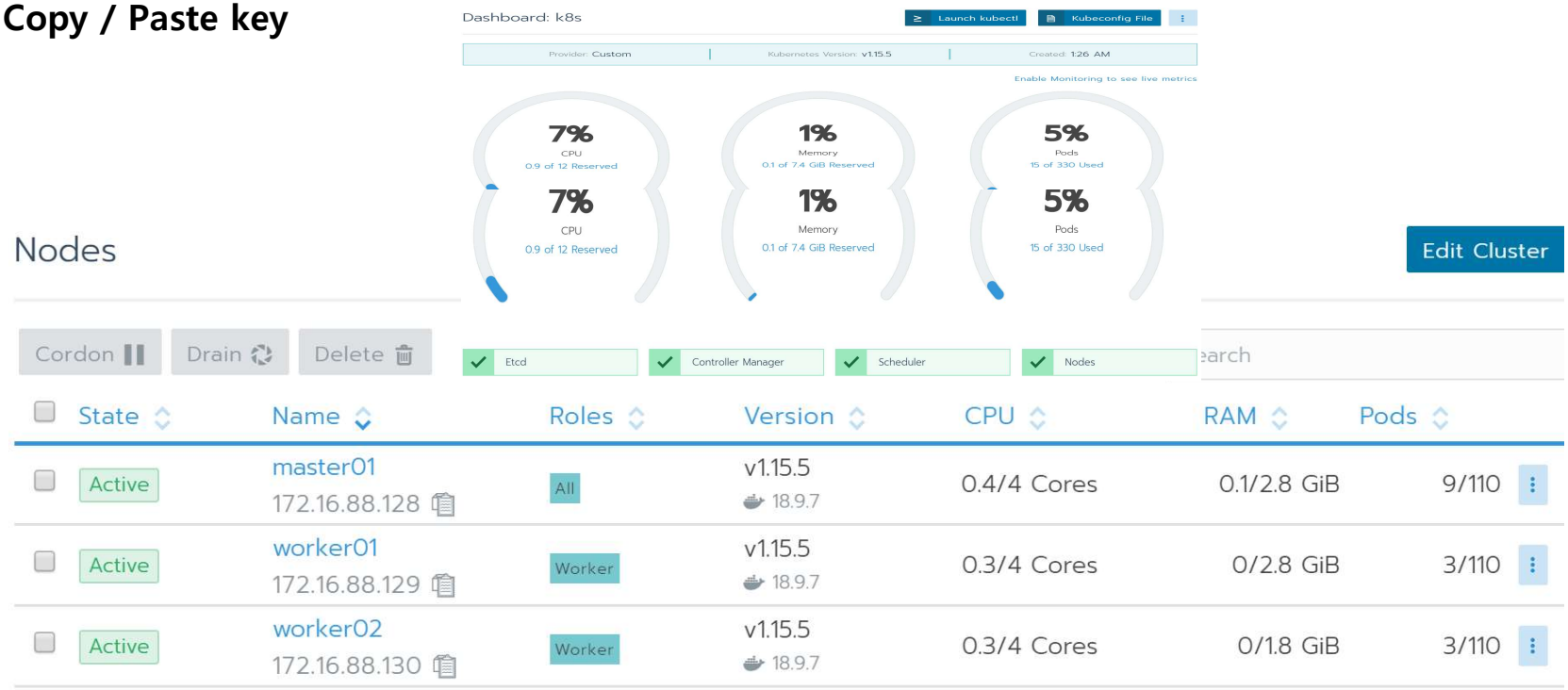
- https://rancher.com/docs/rancher/v2.x/en/installation/single-node/

5. Host for Rancher

5) Rancher 설치

- **Rancher Web**

1. **Cluster Options**
2. **Copy / Paste key**



References:

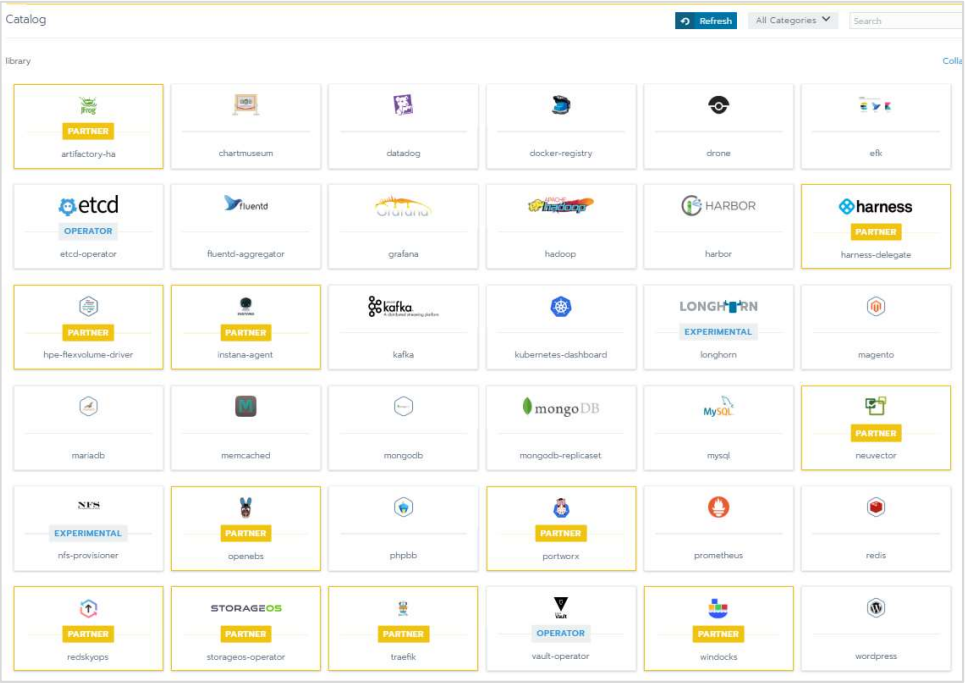
- https://rancher.com/docs/rancher/v2.x/en/installation/single-node/

5. Host for Rancher

5) Rancher 설치

- **Rancher Web**

1. **Apps @ k8s system**
2. **K8s Dashboard**
3. **Wordpress**



References:

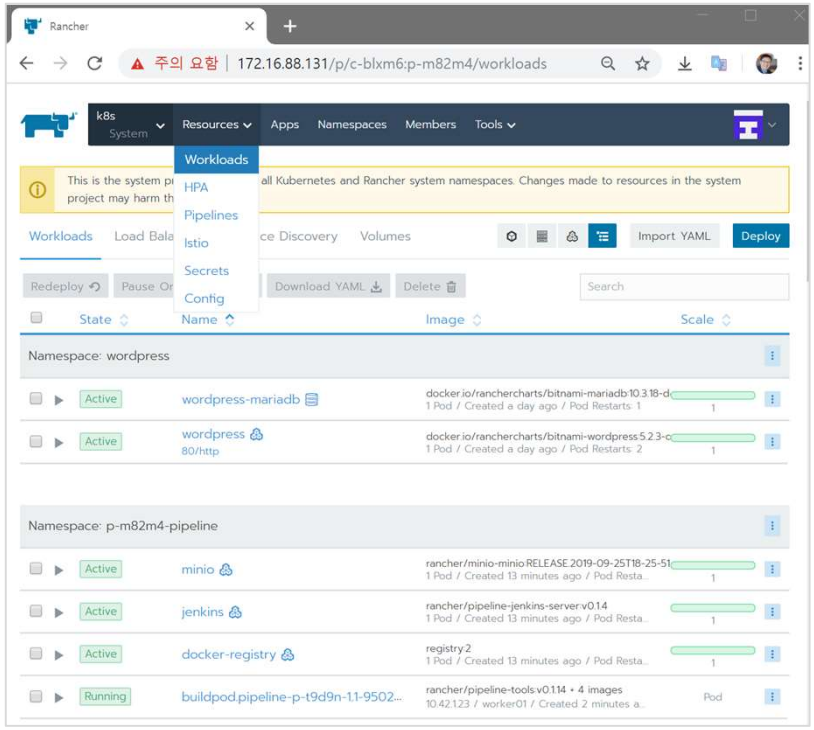
- https://rancher.com/docs/rancher/v2.x/en/installation/single-node/

5. Host for Rancher

5) Rancher 설치

- **Workloads**

1. **Workloads @ K8s system Resources**



References:

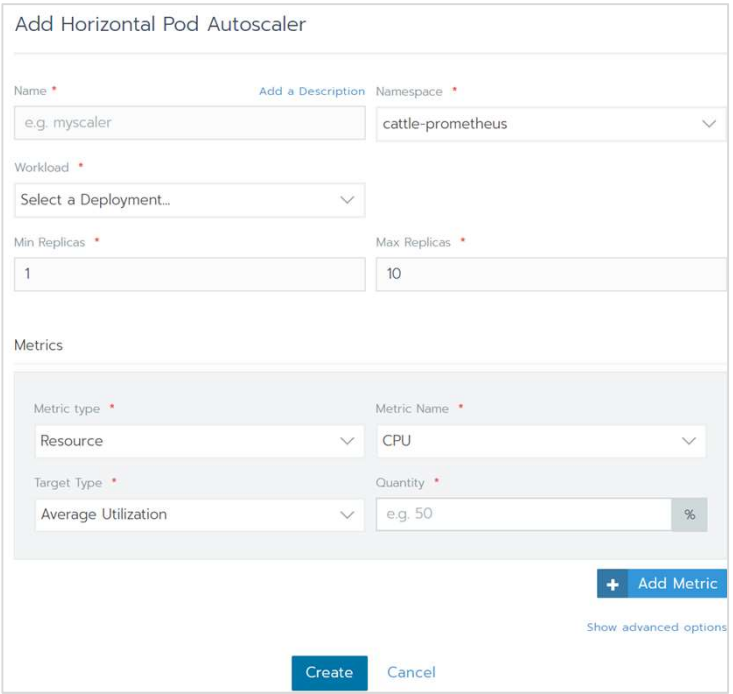
- https://rancher.com/docs/rancher/v2.x/en/installation/single-node/

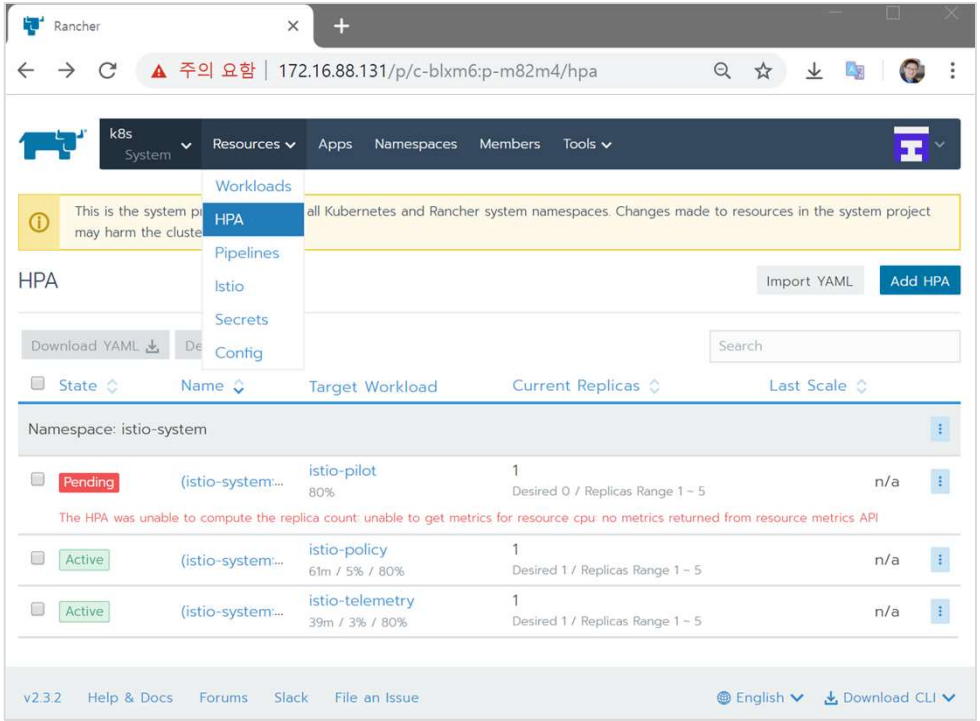
5. Host for Rancher

5) Rancher 설치

- **HPA** (Horizontal Pod Autoscaler)

1. HPA @ K8s system Resources





References:

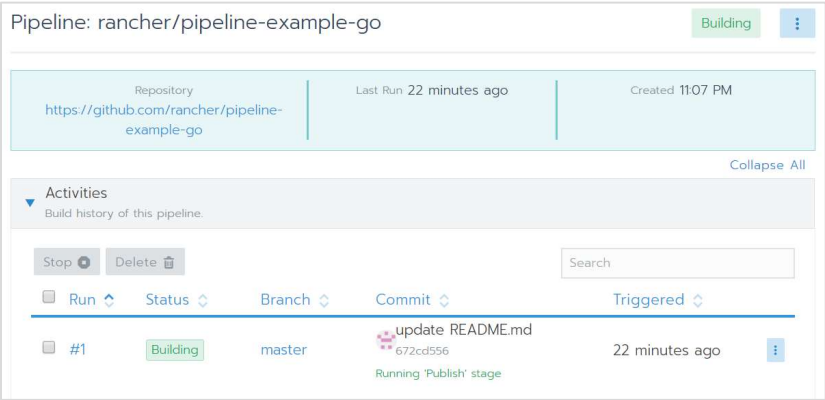
- https://rancher.com/docs/rancher/v2.x/en/installation/single-node/

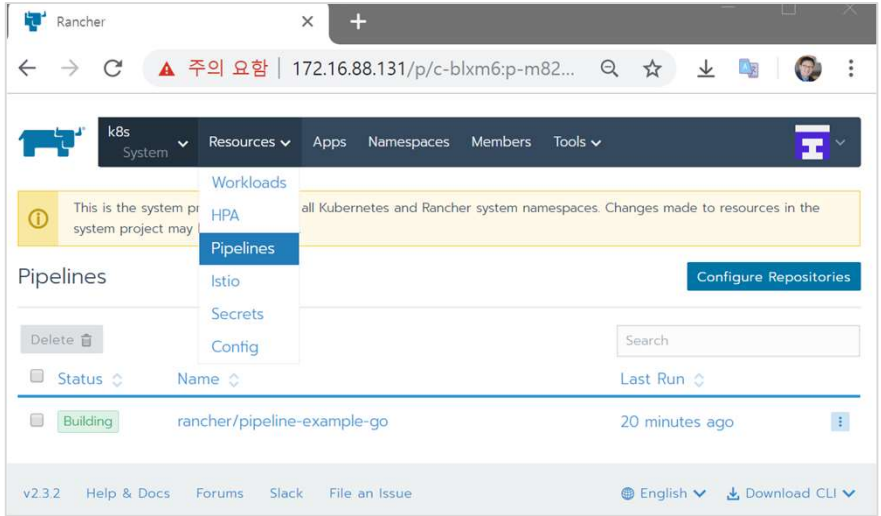
5. Host for Rancher

5) Rancher 설치

- **Pipeline**

1. **Pipeline @ K8s system Resources**





References:

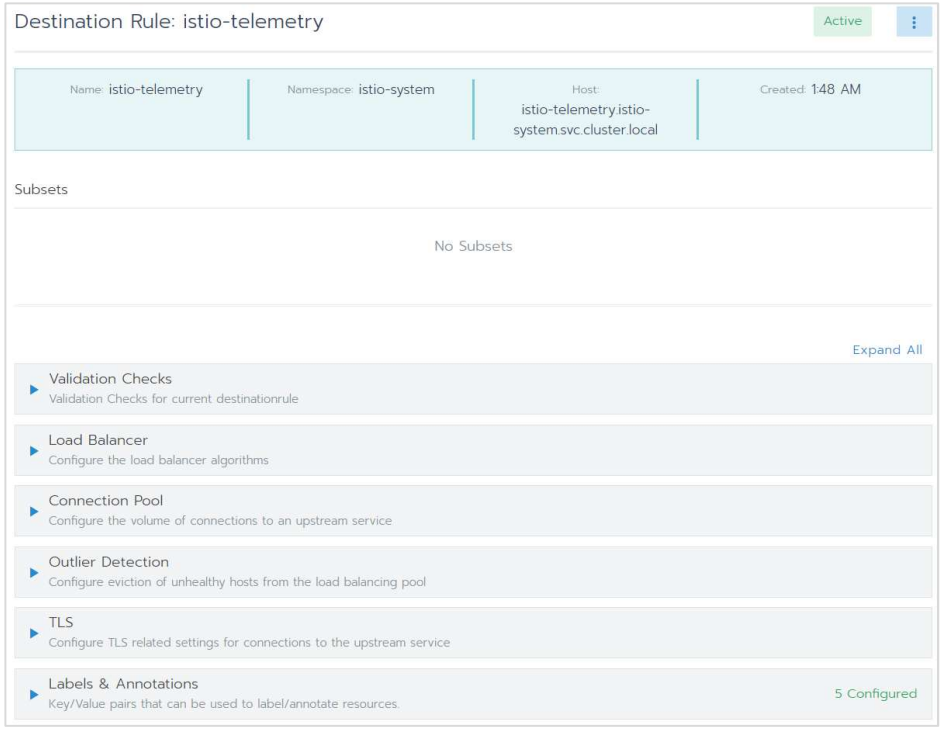
- https://rancher.com/docs/rancher/v2.x/en/installation/single-node/

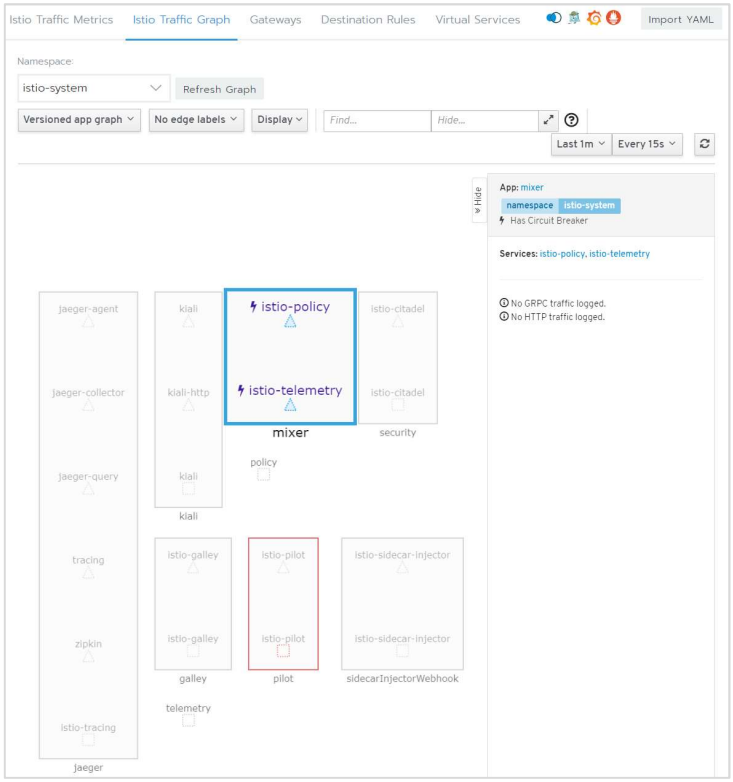
5. Host for Rancher

5) Rancher 설치

- **Istio**

1. Istio @ K8s system Resources





References:

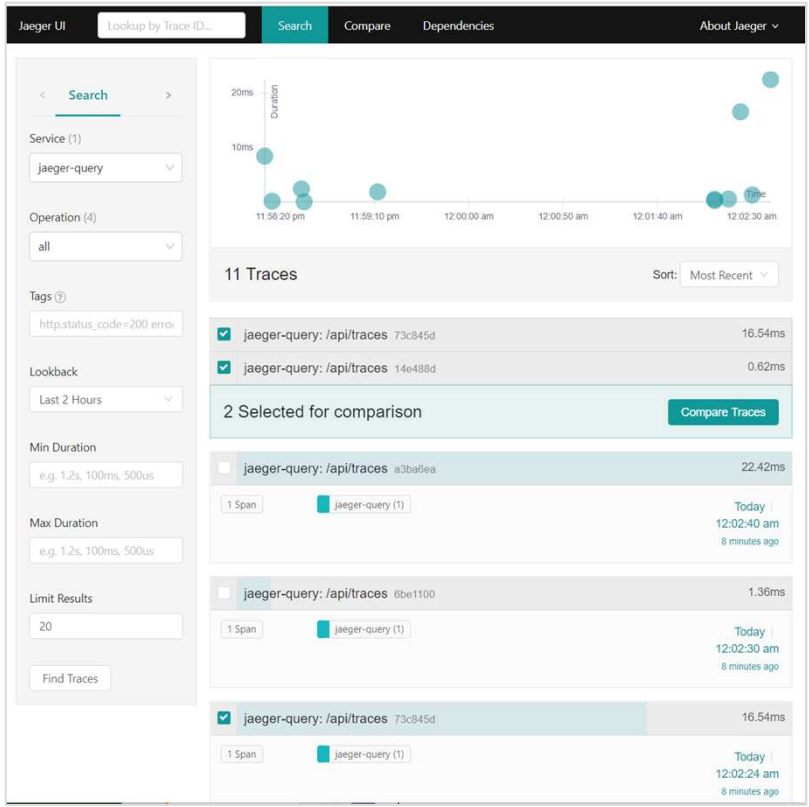
- https://rancher.com/docs/rancher/v2.x/en/installation/single-node/

5. Host for Rancher

5) Rancher 설치

- **Jaeger**

1. **Jaeger @ Istio**



References:

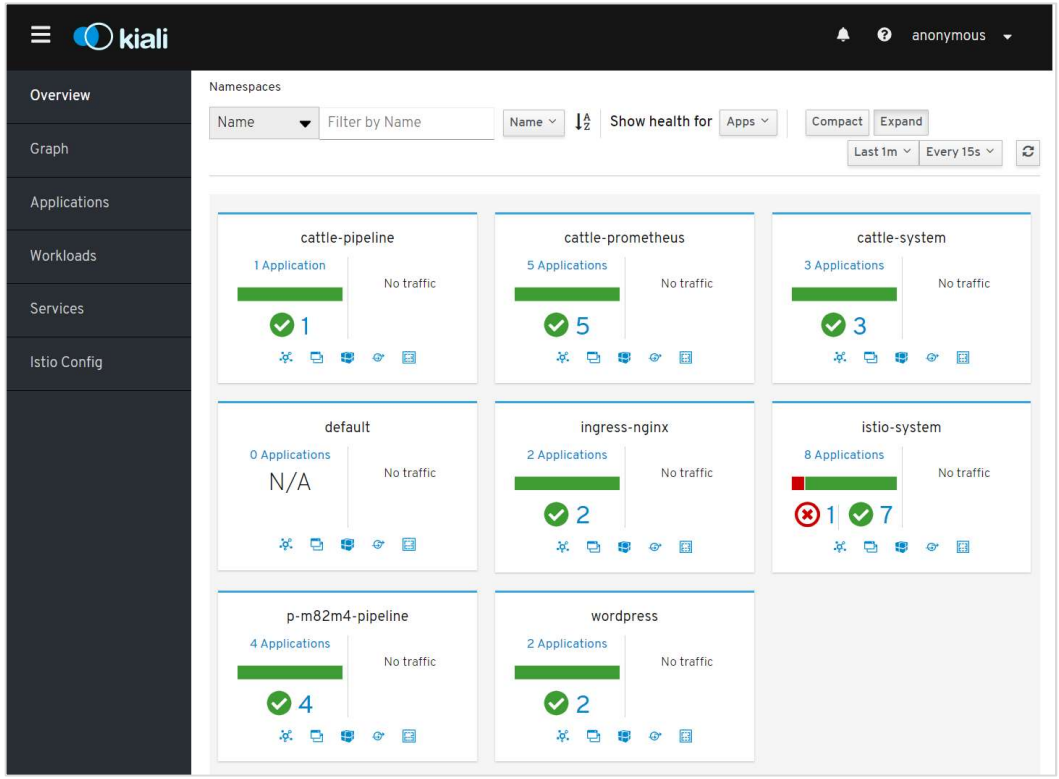
- https://rancher.com/docs/rancher/v2.x/en/installation/single-node/

5. Host for Rancher

5) Rancher 설치

- **kiali**

1. **kiali @ Istio**



References:

- https://rancher.com/docs/rancher/v2.x/en/installation/single-node/

목차

I. K8s Clustering

II. K8s Operations

II. K8s Operations

1. Launch A Single Node Cluster
2. Getting Started With Kubeadm
3. Deploy Containers Using Kubectl
4. Deploy Containers Using YAML
5. Deploy Guestbook Web App Example
6. Networking Introduction
7. Create Ingress Routing
8. Liveness and Readiness Healthchecks
9. Getting Started With CRI-O and Kubeadm
10. Running Stateful Services on Kubernetes
11. Use Kubernetes To Manage Secrets And Passwords
12. Deploy Docker Compose Files with Kompose
13. Deploying from source onto Kubernetes

Q
&
A