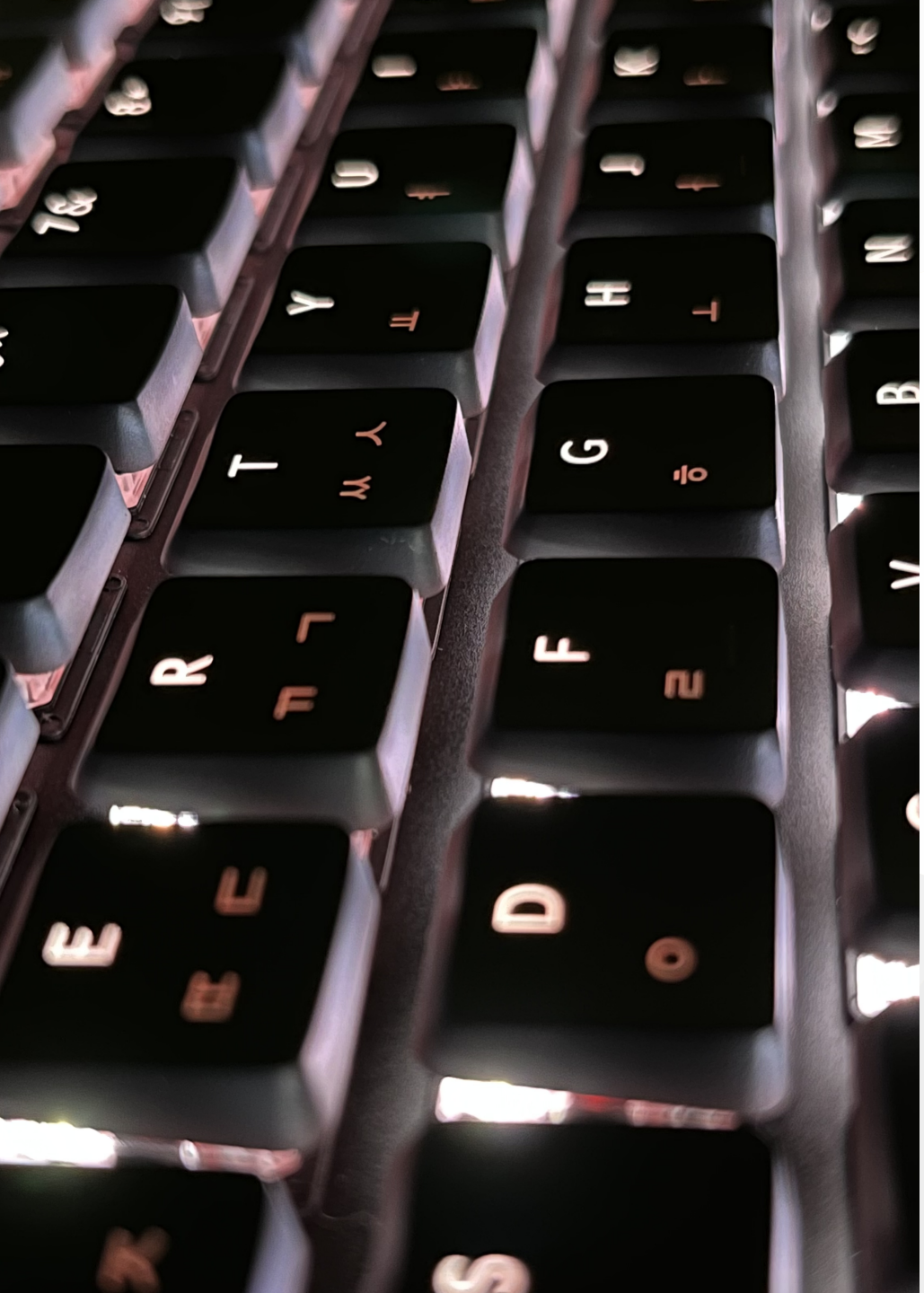

Contents

Minikube 설치하기

Kubernetes 실습을 위해서 **K8s cluster**를 구성합니다.
K8s cluster는 다양한 방법으로 구성할 수 있지만, 우리 실습과정은 **단일노드 cluster**인 Minikube 를 이용합니다.

우리가 앞에서 사용한 VM Instance는 Minikube의 설치조건 을 만족하도록 구성되어 있습니다.

먼저 기존에 Docker 실습에서 사용하던 Instance로 로그인합니다.

```
> ssh -i "mspt3.pem" ubuntu@ec2-00-00-00-00.compute-1.amazonaws.com
Welcome to Ubuntu 22.04.1 LTS (GNU/Linux 5.15.0-1028-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Tue Jan 24 12:28:17 UTC 2023

... 생략 ...

Last login: Tue Jan 24 12:06:05 2023 from 121.165.174.35
```

> **명령어** : `ssh -i "mspt3.pem" ubuntu@[PUBLIC_IPV4_ADDRESS/DNS]`
> [PUBLIC_IPV4_ADDRESS/DNS] 에는 여러분의 VM Instance 정보를 넣으세요.

Docker & Kubernetes - [Hands-on] 06. Environment setup - Kubernetes

minikube 설치파일을 다운로드 하고 설치를 진행합니다.

```
ubuntu@ip-10-0-2-33:~$ curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 73.1M  100 73.1M    0     0   132M      0 --:--:-- --:--:-- --:--:--  132M
ubuntu@ip-10-0-2-33:~$ sudo install minikube-linux-amd64 /usr/local/bin/minikube
```

> 명령어 : `curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64`
> 명령어 : `sudo install minikube-linux-amd64 /usr/local/bin/minikube`

Docker & Kubernetes - [Hands-on] 06. Environment setup - Kubernetes

minikube를 시작하기 전에 먼저 한 가지 패키지(conntrack)를 설치합니다.

```
ubuntu@ip-10-0-2-33:~$ sudo apt-get update
...생략...
ubuntu@ip-10-0-2-33:~$ sudo apt-get install conntrack
...생략...
Unpacking conntrack (1:1.4.6-2build2) ...
Setting up conntrack (1:1.4.6-2build2) ...
Processing triggers for man-db (2.10.2-1) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
```

> 명령어 : `sudo apt-get update`
> 명령어 : `sudo apt-get install conntrack`

Docker & Kubernetes - [Hands-on] 06. Environment setup - Kubernetes

이제 minikube CLI를 이용해서 minikube cluster를 시작합니다.

```
ubuntu@ip-10-0-2-33:~$ minikube start --driver=none --kubernetes-version=v1.23.0
😄  minikube v1.28.0 on Ubuntu 22.04
✨  Using the none driver based on user configuration
👍  Starting control plane node minikube in cluster minikube
🤹  Running on localhost (CPUs=2, Memory=3863MB, Disk=19662MB) ...
ℹ️  OS release is Ubuntu 22.04.1 LTS
🐳  Preparing Kubernetes v1.23.0 on Docker 20.10.23 ...
    ▪ kubelet.resolv-conf=/run/systemd/resolve/resolv.conf
    > kubectl.sha256:  64 B / 64 B [--------------------------] 100.00% ? p/s 0s
    > kubelet.sha256:  64 B / 64 B [--------------------------] 100.00% ? p/s 0s
    > kubeadm.sha256:  64 B / 64 B [--------------------------] 100.00% ? p/s 0s
    > kubectl:  44.42 MiB / 44.42 MiB [----------] 100.00% 431.68 MiB p/s 300ms
    > kubeadm:  43.11 MiB / 43.11 MiB [----------] 100.00% 215.70 MiB p/s 400ms
    > kubelet:  118.73 MiB / 118.73 MiB [---------] 100.00% 156.81 MiB p/s 1.0s
    ▪ Generating certificates and keys ...
    ▪ Booting up control plane ...
    ▪ Configuring RBAC rules ...
🤹  Configuring local host environment ...

❗  The 'none' driver is designed for experts who need to integrate with an existing VM
💡  Most users should use the newer 'docker' driver instead, which does not require root!
📘  For more information, see: https://minikube.sigs.k8s.io/docs/reference/drivers/none/

❗  kubectl and minikube configuration will be stored in /home/ubuntu
❗  To use kubectl or minikube commands as your own user, you may need to relocate them. For example, to overwrite your own settings, run:

    ▪ sudo mv /home/ubuntu/.kube /home/ubuntu/.minikube $HOME
    ▪ sudo chown -R $USER $HOME/.kube $HOME/.minikube
```

```
💡  This can also be done automatically by setting the env var CHANGE_MINIKUBE_NONE_USER=true
🔎  Verifying Kubernetes components...
    ▪ Using image gcr.io/k8s-minikube/storage-provisioner:v5
🌟  Enabled addons: default-storageclass, storage-provisioner
💡  kubectl not found. If you need it, try: 'minikube kubectl -- get pods -A'
🏄  Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
```

> 명령어 : `minikube start --driver=none --kubernetes-version=v1.23.0`

문제없이 시작된 경우 다음과 같이 표시됩니다.

```
ubuntu@ip-10-0-2-33:~/minikube$ minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured
```

> 명령어 : `minikube status`

Minikube 설정하기(Addons)

앞으로 진행할 실습을 위해 몇 가지 minikube addon들을 활성화 하겠습니다.

```
ubuntu@ip-10-0-2-33:~$ minikube addons enable ingress
💡  ingress is an addon maintained by Kubernetes. For any concerns contact minikube on GitHub.
You can view the list of minikube maintainers at: https://github.com/kubernetes/minikube/blob/master/OWNERS
    ▪ Using image k8s.gcr.io/ingress-nginx/kube-webhook-certgen:v1.1.1
    ▪ Using image k8s.gcr.io/ingress-nginx/kube-webhook-certgen:v1.1.1
    ▪ Using image k8s.gcr.io/ingress-nginx/controller:v1.2.1
🔎  Verifying ingress addon...
🌟  The 'ingress' addon is enabled
ubuntu@ip-10-0-2-33:~$ minikube addons enable metrics-server
💡  metrics-server is an addon maintained by Kubernetes. For any concerns contact minikube on GitHub.
You can view the list of minikube maintainers at: https://github.com/kubernetes/minikube/blob/master/OWNERS
    ▪ Using image k8s.gcr.io/metrics-server/metrics-server:v0.6.1
🌟  The 'metrics-server' addon is enabled
```

> 명령어 : `minikube addons enable ingress`
> 명령어 : `minikube addons enable metrics-server`

Kubectl 설치하기 (리눅스에 kubectl 설치 및 설정)

Kubernetes command-line tool인 kubectl 을 다운로드 하고 설치합니다.

```
ubuntu@ip-10-0-2-33:~$ curl -LO https://dl.k8s.io/release/v1.23.0/bin/linux/amd64/kubectl
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   138  100   138    0     0    940      0 --:--:-- --:--:-- --:--:--   945
100 44.4M  100 44.4M    0     0  94.2M      0 --:--:-- --:--:-- --:--:--  572M
ubuntu@ip-10-0-2-33:~$ sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl
```

명령어 : `curl -LO https://dl.k8s.io/release/v1.23.0/bin/linux/amd64/kubectl`
명령어 : `sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl`

아래와 같이 표시되면 정상입니다.

```
ubuntu@ip-10-0-2-33:~$ kubectl version
Client Version: version.Info{Major:"1", Minor:"23", GitVersion:"v1.23.0", GitCommit:"ab69524f795c42094a6630298ff53f3c3ebab7f4", GitTreeState:"clean", BuildDate:"2021-12-07T18:16:20Z", GoVersion:"go1.17.3", Compiler:"gc", Platform:"linux/amd64"}
Server Version: version.Info{Major:"1", Minor:"23", GitVersion:"v1.23.0", GitCommit:"ab69524f795c42094a6630298ff53f3c3ebab7f4", GitTreeState:"clean", BuildDate:"2021-12-07T18:09:57Z", GoVersion:"go1.17.3", Compiler:"gc", Platform:"linux/amd64"}
```

명령어 : `kubectl version`

Docker & Kubernetes - [Hands-on] 06. Environment setup - Kubernetes

편리한 사용을 위해서 선택적 kubectl 구성 및 플러그인 을 진행합니다. (명령어 자동완성, alias 적용)
필수사항은 아니니 필요없는 경우 생략해도 됩니다.

```
ubuntu@ip-10-0-2-33:~$ echo 'source <(kubectl completion bash)' >>~/.bashrc
ubuntu@ip-10-0-2-33:~$ echo 'alias k=kubectl' >>~/.bashrc
ubuntu@ip-10-0-2-33:~$ echo 'complete -o default -F __start_kubectl k' >>~/.bashrc
ubuntu@ip-10-0-2-33:~$ exec bash
```

> 명령어 : `echo 'source <(kubectl completion bash)' >>~/.bashrc`
> 명령어 : `echo 'alias k=kubectl' >>~/.bashrc`
> 명령어 : `echo 'complete -o default -F __start_kubectl k' >>~/.bashrc`
> 명령어 : `exec bash`

이제 명령어 작성 중 `TAB` 키를 눌러 자동완성을 사용하거나, `kubectl` 대신 Alias인 `k` 를 사용할 수 있습니다.

```
ubuntu@ip-10-0-2-33:~$ k version
Client Version: version.Info{Major:"1", Minor:"23", GitVersion:"v1.23.0", GitCommit:"ab69524f795c42094a6630298ff53f3c3ebab7f4", GitTreeState:"clean", BuildDate:"2021-12-07T18:16:20Z", GoVersion:"go1.17.3", Compiler:"gc", Platform:"linux/amd64"}
Server Version: version.Info{Major:"1", Minor:"23", GitVersion:"v1.23.0", GitCommit:"ab69524f795c42094a6630298ff53f3c3ebab7f4", GitTreeState:"clean", BuildDate:"2021-12-07T18:09:57Z", GoVersion:"go1.17.3", Compiler:"gc", Platform:"linux/amd64"}
```

> 명령어 : `k version`

Helm 설치하기 (헬름 설치하기)

Kubernetes 패키지 매니저인 Helm을 설치합니다.

```
ubuntu@ip-10-0-2-33:~$ wget https://get.helm.sh/helm-v3.11.0-linux-amd64.tar.gz
--2023-01-25 14:37:05--  https://get.helm.sh/helm-v3.11.0-linux-amd64.tar.gz
Resolving get.helm.sh (get.helm.sh)... 152.195.19.97, 2606:2800:11f:1cb7:261b:1f9c:2074:3c
Connecting to get.helm.sh (get.helm.sh)|152.195.19.97|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 15023353 (14M) [application/x-tar]
Saving to: ‘helm-v3.11.0-linux-amd64.tar.gz’

helm-v3.11.0-linux-amd64.tar.gz         100%[=============================================================================================>]  14.33M  --.-KB/s    in 0.1s

2023-01-25 14:37:06 (130 MB/s) - ‘helm-v3.11.0-linux-amd64.tar.gz’ saved [15023353/15023353]

ubuntu@ip-10-0-2-33:~$ tar -zxvf helm-v3.11.0-linux-amd64.tar.gz
linux-amd64/
linux-amd64/helm
linux-amd64/LICENSE
linux-amd64/README.md
ubuntu@ip-10-0-2-33:~$ sudo mv linux-amd64/helm /usr/local/bin/helm
```

명령어 : `wget https://get.helm.sh/helm-v3.11.0-linux-amd64.tar.gz`
명령어 : `tar -zxvf helm-v3.11.0-linux-amd64.tar.gz`
명령어 : `sudo mv linux-amd64/helm /usr/local/bin/helm`

설치 후 아래와 같이 확인합니다.

```
ubuntu@ip-10-0-2-33:~$ helm version
version.BuildInfo{Version:"v3.11.0", GitCommit:"472c5736ab01133de504a826bd9ee12cbe4e7904", GitTreeState:"clean", GoVersion:"go1.18.10"}
```

> 명령어 : `helm version`

여기까지 하시면 Kubernetes와 Helm을 공부할 준비는 됐습니다. ˇ◡ˇ