쿠버네티스(k8s) 설치 및 클러스터 구성하기

1. 선 작업 수행(시스템 패키지 업데이트, k8s 사용자 생성)

k8s 클러스터에 참여하는 모든 노드에서 시스템 패키지 업데이트를 수행합니다.

```
$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo apt-get install linux-image-extra-virtual
$ sudo reboot
```

k8s 클러스터에 참여하는 모든 노드에 k8s 클러스터를 관리할 사용자를 추가합니다.

```
$ sudo useradd -s /bin/bash -m k8s-admin
$ sudo passwd k8s-admin
$ sudo usermod -aG sudo k8s-admin
```

2. 도커 설치

k8s는 k8s서비스 및 어플리케이션을 위해 사용되는 컨테이너 실행을 위해 도커를 필요로 합니다.

k8s 클러스터에 참여하는 모든 노드에 도커를 설치합니다.

'19년 2월 20일기준으로 설치된 버전이 18.06이하인 경우 삭제후 재설치 바랍니다.

- 도커 삭제

```
$ sudo apt-get remove docker docker-engine docker.i
```

- 도커 설치

```
$ sudo apt-get update
$ sudo apt-get install \
    apt-transport-https \
    ca-certificates \
    curl \
    software-properties-common
$ sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
$ sudo add-apt-repository \
    "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
    $(lsb_release -cs) \
    stable"
$ sudo apt-get update
$ sudo apt-cache madison docker-ce
 docker-ce | 5:18.09.3~3-0~ubuntu-xenial | https://download.docker.com/linux/ubuntu
xenial/stable amd64 Packages
 docker-ce | 5:18.09.2~3-0~ubuntu-xenial | https://download.docker.com/linux/ubuntu
xenial/stable amd64 Packages
 docker-ce | 5:18.09.1~3-0~ubuntu-xenial | https://download.docker.com/linux/ubuntu
xenial/stable amd64 Packages
 docker-ce | 5:18.09.0~3-0~ubuntu-xenial | https://download.docker.com/linux/ubuntu
xenial/stable amd64 Packages
 docker-ce | 18.06.3~ce~3-0~ubuntu | https://download.docker.com/linux/ubuntu xenial/stable
amd64 Packages
```

```
 docker-ce | 18.06.2~ce~3-0~ubuntu | https://download.docker.com/linux/ubuntu xenial/stable
amd64 Packages
 docker-ce | 18.06.1~ce~3-0~ubuntu | https://download.docker.com/linux/ubuntu xenial/stable
amd64 Packages
 docker-ce | 18.06.0~ce~3-0~ubuntu | https://download.docker.com/linux/ubuntu xenial/stable
amd64 Packages
 docker-ce | 18.03.1~ce-0~ubuntu | https://download.docker.com/linux/ubuntu xenial/stable
amd64 Packages
 docker-ce | 18.03.0~ce-0~ubuntu | https://download.docker.com/linux/ubuntu xenial/stable
amd64 Packages
 docker-ce | 17.12.1~ce-0~ubuntu | https://download.docker.com/linux/ubuntu xenial/stable
amd64 Packages
 docker-ce | 17.12.0~ce-0~ubuntu | https://download.docker.com/linux/ubuntu xenial/stable
amd64 Packages
 docker-ce | 17.09.1~ce-0~ubuntu | https://download.docker.com/linux/ubuntu xenial/stable
amd64 Packages
 docker-ce | 17.09.0~ce-0~ubuntu | https://download.docker.com/linux/ubuntu xenial/stable
amd64 Packages
 docker-ce | 17.06.2~ce-0~ubuntu | https://download.docker.com/linux/ubuntu xenial/stable
amd64 Packages
 docker-ce | 17.06.1~ce-0~ubuntu | https://download.docker.com/linux/ubuntu xenial/stable
amd64 Packages
 docker-ce | 17.06.0~ce-0~ubuntu | https://download.docker.com/linux/ubuntu xenial/stable
amd64 Packages
 docker-ce | 17.03.3~ce-0~ubuntu-xenial | https://download.docker.com/linux/ubuntu
xenial/stable amd64 Packages
 docker-ce | 17.03.2~ce-0~ubuntu-xenial | https://download.docker.com/linux/ubuntu
xenial/stable amd64 Packages
 docker-ce | 17.03.1~ce-0~ubuntu-xenial | https://download.docker.com/linux/ubuntu
xenial/stable amd64 Packages
 docker-ce | 17.03.0~ce-0~ubuntu-xenial | https://download.docker.com/linux/ubuntu
xenial/stable amd64 Packages
```

```
$ sudo apt-get install docker-ce=18.06.3~ce~3-0~ubuntu # validated 버전 18.06('19 2/20)

$ sudo usermod -aG docker k8s-admin
```

3. k8s 설치 - Master, Worker 노드

k8s 클러스터 구성에 참여하는 노드중 마스터 노드에 k8s 패키지를 설치합니다.

```
$ sudo apt install apt-transport-https
$ sudo curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add
$ sudo add-apt-repository "deb https://apt.kubernetes.io/ kubernetes-$(lsb_release -cs) main"
$ sudo apt update
$ sudo apt install kubelet kubeadm kubectl kubernetes-cni
$ which kubelet
/usr/bin/kubelet
$ which kubeadm
```

Swap을 사용하는 경우 k8s 클러스터가 구동되지 않습니다.

swapoff 명령어를 사용하거나 /etc/fstab 파일에서 swap부분을 주석처리(재부팅 필요)합니다.

```
$ sudo swapoff -a
# 또는
$ sudo vi /etc/fstab
#UUID=8a99a48d-076b-4c85-ac3c-bc950e5ce829 none            swap    sw              0       0
```

4. k8s 초기화 - Master 노드

k8s 클러스터 초기화를 위해 아래 환경 변수를 설정합니다.

```
$ export API_ADDR="192.168.137.2" # Master 서버 외부 IP
$ export DNS_DOMAIN="k8s.local"
$ export POD_NET="10.100.0.0/16" # k8s 클러스터 POD Network CIDR
```

먼저, SDS 사내에서 사용하는 경우 아래와 같이 Proxy 예외 처리를 합니다. (~/.bashrc 파일에 설정 필요)

```
$ export
no_proxy="localhost,127.0.0.0/8,::1,192.168.0.0/16,70.*,10.0.0.0/16,redii.net,*.redii.net,*.s
dsdev.co.kr,10.96.0.0/12,10.10.10.0/24,10.100.0.0/16,10.0.0.0/8"
```

k8s 클러스터를 초기화합니다.

```
$ kubeadm init --pod-network-cidr ${POD_NET} \
--apiserver-advertise-address ${API_ADDR} \
--service-dns-domain "${DNS_DOMAIN}"
[init] Using Kubernetes version: v1.13.4
[preflight] Running pre-flight checks
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your internet
connection
[preflight] You can also perform this action in beforehand using 'kubeadm config images pull'
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-
flags.env"
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Activating the kubelet service
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[certs] Generating "front-proxy-ca" certificate and key
[certs] Generating "front-proxy-client" certificate and key
[certs] Generating "etcd/ca" certificate and key
[certs] Generating "etcd/healthcheck-client" certificate and key
[certs] Generating "apiserver-etcd-client" certificate and key
[certs] Generating "etcd/server" certificate and key
[certs] etcd/server serving cert is signed for DNS names [node1 localhost] and IPs
[192.168.137.2 127.0.0.1 ::1]
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [node1 localhost] and IPs
[192.168.137.2 127.0.0.1 ::1]
[certs] Generating "ca" certificate and key
[certs] Generating "apiserver" certificate and key
[certs] apiserver serving cert is signed for DNS names [node1 kubernetes kubernetes.default
kubernetes.default.svc kubernetes.default.svc.k8s.local] and IPs [10.96.0.1 192.168.137.2]
[certs] Generating "apiserver-kubelet-client" certificate and key
[certs] Generating "sa" key and public key
[kubeconfig] Using kubeconfig folder "/etc/kubernetes"
[kubeconfig] Writing "admin.conf" kubeconfig file
[kubeconfig] Writing "kubelet.conf" kubeconfig file
[kubeconfig] Writing "controller-manager.conf" kubeconfig file
[kubeconfig] Writing "scheduler.conf" kubeconfig file
[control-plane] Using manifest folder "/etc/kubernetes/manifests"
[control-plane] Creating static Pod manifest for "kube-apiserver"
```

```
[control-plane] Creating static Pod manifest for "kube-controller-manager"
[control-plane] Creating static Pod manifest for "kube-scheduler"
[etcd] Creating static Pod manifest for local etcd in "/etc/kubernetes/manifests"
[wait-control-plane] Waiting for the kubelet to boot up the control plane as static Pods from
directory "/etc/kubernetes/manifests". This can take up to 4m0s
[apiclient] All control plane components are healthy after 25.502682 seconds
[uploadconfig] storing the configuration used in ConfigMap "kubeadm-config" in the "kube-
system" Namespace
[kubelet] Creating a ConfigMap "kubelet-config-1.13" in namespace kube-system with the
configuration for the kubelets in the cluster
[patchnode] Uploading the CRI Socket information "/var/run/dockershim.sock" to the Node API
object "node1" as an annotation
[mark-control-plane] Marking the node node1 as control-plane by adding the label "node-
role.kubernetes.io/master=''"
[mark-control-plane] Marking the node node1 as control-plane by adding the taints [node-
role.kubernetes.io/master:NoSchedule]
[bootstrap-token] Using token: lazOgb.8qw5xq3Ol84k3kzj
[bootstrap-token] Configuring bootstrap tokens, cluster-info ConfigMap, RBAC Roles
[bootstraptoken] configured RBAC rules to allow Node Bootstrap tokens to post CSRs in order
for nodes to get long term certificate credentials
[bootstraptoken] configured RBAC rules to allow the csrapprover controller automatically
approve CSRs from a Node Bootstrap Token
[bootstraptoken] configured RBAC rules to allow certificate rotation for all node client
certificates in the cluster
[bootstraptoken] creating the "cluster-info" ConfigMap in the "kube-public" namespace
[addons] Applied essential addon: CoreDNS
[addons] Applied essential addon: kube-proxy

Your Kubernetes master has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  https://kubernetes.io/docs/concepts/cluster-administration/addons/

You can now join any number of machines by running the following on each node
as root:

  kubeadm join 192.168.137.2:6443 --token lazOgb.8qw5xq3Ol84k3kzj --discovery-token-ca-cert-
hash sha256:271ad66bf15c29fb5695c0aa05e462161c00e8c3ae7cd998439ddb61af2c5f80
```

위 출력된 결과중 마지막 라인 **kubeadm join...** 이부분은 Worker 노드에서 k8s 클러스터에 조인할 때 사용됩니다. **token**값에 만기일이 설정되어 있어 만기가 되면 사용할 수 없습니다. TTL 확인은 아래 명령어로 가능합니다.

```
k8s-admin@master:~$ kubeadm token list
TOKEN                     TTL       EXPIRES                     USAGES
DESCRIPTION                                                EXTRA GROUPS
lazOgb.8qw5xq3Ol84k3kzj   23h       2019-03-19T16:28:43+09:00   authentication,signing   The
default bootstrap token generated by 'kubeadm init'.   system:bootstrappers:kubeadm:default-
node-token
```

k8s 클러스터 관리 사용자 계정인 **k8s-admin** 으로 서버에 로그인 합니다.

```
k8s-admin@master:~$
```

k8s Master 노드에 접속하기 위해 설정 정보를 **k8s-admin** 계정에 설정합니다.

```
k8s-admin@master:~$ \rm -rf $HOME/.k8s
k8s-admin@master:~$ mkdir -p $HOME/.k8s
k8s-admin@master:~$ sudo cp -i /etc/kubernetes/admin.conf $HOME/.k8s/config
k8s-admin@master:~$ sudo chown $(id -u):$(id -g) $HOME/.k8s/config
k8s-admin@master:~$ export KUBECONFIG=$HOME/.k8s/config
k8s-admin@master:~$ echo "export KUBECONFIG=$HOME/.k8s/config" | tee -a ~/.bashrc
```

5. Calico Pod Network 배포하기

CNI(Container Network Interface)는 컨테이너와 컨테이너 네트워크 구현체 사이의 표준 API 입니다.

- Introduction to the Container Network Interface (CNI) - https://www.slideshare.net/weaveworks/introduction-to-the-container-network-interface-cni
- Cluster Networking - https://kubernetes.io/docs/concepts/cluster-administration/networking/

CNI 구현체는 ACI, Weave Net, Calico등등 다양하게 지원하고 있으며 이 문서에서는 Calico 프로젝트를 이용해서 구현하는 방법에 대하여 설명합니다.

```
k8s-admin@master:~$ wget https://docs.projectcalico.org/v3.5/getting-
started/kubernetes/installation/hosted/etcd.yaml
k8s-admin@master:~$ wget https://docs.projectcalico.org/v3.5/getting-
started/kubernetes/installation/hosted/calico.yaml
k8s-admin@master:~$ vi calico.yaml
 - name: CALICO_IPV4POOL_CIDR
              value: "10.100.0.0/16" # POD Network CIDR로 변경합니다

k8s-admin@master:~$ kubectl apply -f etcd.yaml
k8s-admin@master:~$ kubectl apply -f calico.yaml
```

정상적으로 Calico Pod 이 실행되는 지 아래 명령어로 확인합니다.

```
k8s-admin@master:~$ watch kubectl get pods --all-namespaces
Every 2.0s: kubectl get pods --all-namespaces
Mon Mar 18 16:52:41 2019

NAMESPACE     NAME                                READY   STATUS    RESTARTS   AGE
kube-system   calico-etcd-s2r95                   1/1     Running   0          80s
kube-system   calico-kube-controllers-5cff4f556c-sl86d   1/1     Running   2          102s
kube-system   calico-node-x4jmd                   1/1     Running   0          102s
kube-system   coredns-86c58d9df4-5c45w            1/1     Running   0          23m
kube-system   coredns-86c58d9df4-pjwdq            1/1     Running   0          23m
kube-system   etcd-node1                          1/1     Running   0          23m
kube-system   kube-apiserver-node1                1/1     Running   0          23m
kube-system   kube-controller-manager-node1       1/1     Running   0          22m
kube-system   kube-proxy-ttzzm                    1/1     Running   0          23m
kube-system   kube-scheduler-node1                1/1     Running   0          23m
```

Calico Pod이 정상적으로 실행된 경우 **STATUS** 값이 Running으로 출력됩니다. 오류가 발생한 경우 **kubectl describe** 명령어를 사용하여 해당 컨테이너의 상태를 확인하여 조치를 취합니다.

```
k8s-admin@master:~$ kubectl get po --all-namespaces
NAMESPACE     NAME                                READY   STATUS    RESTARTS   AGE
kube-system   calico-etcd-s2r95                   1/1     Running   0          2m56s
kube-system   calico-kube-controllers-5cff4f556c-sl86d   1/1     Running   2          3m18s
```

```
kube-system   calico-node-x4jmd                          1/1     Running   0          3m18s
kube-system   coredns-86c58d9df4-5c45w                   1/1     Running   0          25m
kube-system   coredns-86c58d9df4-pjwdq                   1/1     Running   0          25m
kube-system   etcd-node1                                 1/1     Running   0          24m
kube-system   kube-apiserver-node1                       1/1     Running   0          24m
kube-system   kube-controller-manager-node1              1/1     Running   0          24m
kube-system   kube-proxy-ttzzm                           1/1     Running   0          25m
kube-system   kube-scheduler-node1                       1/1     Running   0          24m

k8s-admin@master:~$ kubectl describe pod calico-kube-controllers-5cff4f556c-sl86d -n kube-
system
Events:
  Type     Reason            Age                    From               Message
  ----     ------            ----                   ----               -------
  Warning  FailedScheduling  4m25s (x2 over 4m25s)  default-scheduler  0/1 nodes are
available: 1 node(s) had taints that the pod didn't tolerate.
  Normal   Scheduled         4m3s                   default-scheduler  Successfully assigned
kube-system/calico-kube-controllers-5cff4f556c-sl86d to node1
  Normal   Pulling           4m2s                   kubelet, node1     pulling image
"calico/kube-controllers:v3.5.3"
  Normal   Pulled            3m51s                  kubelet, node1     Successfully pulled
image "calico/kube-controllers:v3.5.3"
  Warning  Unhealthy         3m49s                  kubelet, node1     Readiness probe
errored: rpc error: code = Unknown desc = container not running
(dcf776fa902d1e2bf50f515fcc4ad6cd523cc6ea71d0d8cd9a48c2fba7b62d9c)
  Warning  BackOff           3m47s (x2 over 3m48s)  kubelet, node1     Back-off restarting
failed container
  Normal   Created           3m34s (x3 over 3m50s)  kubelet, node1     Created container
  Normal   Started           3m34s (x3 over 3m50s)  kubelet, node1     Started container
  Normal   Pulled            3m34s (x2 over 3m49s)  kubelet, node1     Container image
"calico/kube-controllers:v3.5.3" already present on machine
```

Master 노드에도 Pod을 배포하려면 아래와 같은 명령어를 실행합니다.

```
k8s-admin@master:~$ kubectl taint nodes --all node-role.kubernetes.io/master-
node/node1 untainted
```

6. k8s 초기화 - Worker 노드

Master노드에서 k8s 클러스터 초기화후 화면에 출력된 **kubeadm join** 명령어를 k8s 클러스터에 조인하려는 Workder 노드에서 실행합니다.

```
k8s-admin@workder1:~$ kubeadm join 192.168.137.2:6443 --token lazOgb.8qw5xq3Ol84k3kzj --
discovery-token-ca-cert-hash
sha256:271ad66bf15c29fb5695c0aa05e462161c00e8c3ae7cd998439ddb61af2c5f80
[preflight] Running pre-flight checks
[discovery] Trying to connect to API Server "192.168.137.2:6443"
[discovery] Created cluster-info discovery client, requesting info from
"https://192.168.137.2:6443"
[discovery] Requesting info from "https://192.168.137.2:6443" again to validate TLS against
the pinned public key
[discovery] Cluster info signature and contents are valid and TLS certificate validates
against pinned roots, will use API Server "192.168.137.2:6443"
[discovery] Successfully established connection with API Server "192.168.137.2:6443"
[join] Reading configuration from the cluster...
[join] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-
```

```
config -oyaml'
[kubelet] Downloading configuration for the kubelet from the "kubelet-config-1.13" ConfigMap
in the kube-system namespace
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-
flags.env"
[kubelet-start] Activating the kubelet service
[tlsbootstrap] Waiting for the kubelet to perform the TLS Bootstrap...
[patchnode] Uploading the CRI Socket information "/var/run/dockershim.sock" to the Node API
object "node2" as an annotation

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the master to see this node join the cluster.
```

Master 노드에서 아래 명령어로 노드 상태를 확인합니다.

```
k8s-admin@master:~$ kubectl get nodes
NAME    STATUS   ROLES    AGE    VERSION
node1   Ready    master   39m    v1.13.3
node2   Ready    <none>   111s   v1.13.3
node3   Ready    <none>   104s   v1.13.3
```

7. Pod 테스트

k8s 클러스터가 정상적으로 구성되었는 지 Busybox Pod을 배포합니다. Busybox 컨테이너는 CLI 역할을 합니다.

```
k8s-admin@master:~$ wget https://raw.githubusercontent.com/infrabricks/kubernetes-
standalone/master/examples/busybox.yml

k8s-admin@master:~$ vi busyboxy.yml
apiVersion: v1
kind: Pod
metadata:
  name: busybox
  namespace: default
spec:
  containers:
  - image: busybox:1.28 # 1.28 버전 사용. 다른 버전 nslookup 오류
    command:
      - sleep
      - "3600"
    imagePullPolicy: IfNotPresent
    name: busybox
  restartPolicy: Always

k8s-admin@master:~$ kubectl apply -f busybox.yml
pod/busybox created
k8s-admin@master:~$ kubectl get pod
NAME                      READY   STATUS        RESTARTS   AGE
busybox                   1/1     Running       0          95s

k8s-admin@master:~$ kubectl exec -it busybox -- nslookup google.com
Server:    10.96.0.10
Address 1: 10.96.0.10 kube-dns.kube-system.svc.k8s.local
```

```
Name:      google.com
Address 1: 2404:6800:4008:c06::8a tk-in-x8a.1e100.net
Address 2: 64.233.188.102 tk-in-f102.1e100.net
Address 3: 64.233.188.101 tk-in-f101.1e100.net
Address 4: 64.233.188.138 tk-in-f138.1e100.net
Address 5: 64.233.188.113 tk-in-f113.1e100.net
Address 6: 64.233.188.139 tk-in-f139.1e100.net
Address 7: 64.233.188.100 tk-in-f100.1e100.net

k8s-admin@master:~$ kubectl exec -it busybox -- nslookup busybox
Server:    10.96.0.10
Address 1: 10.96.0.10 kube-dns.kube-system.svc.k8s.local

Name:      busybox
Address 1: 192.168.166.210 busybox
```

#참고 문서

- https://computingforgeeks.com/how-to-setup-3-node-kubernetes-cluster-on-ubuntu-18-04-with-weave-net-cni/
- https://arisu1000.tistory.com/27828
- https://www.slideshare.net/JamesAhn/k8s-06-sample