



# k8s - 쿠버네티스 아키텍처 (4) - 노드용컴포넌트



클러스터와 워커 노드의 구조를 알아보자

## 노드용 컴포넌트

### kubelet

+ 추가참조 : <https://kubernetes.io/ko/docs/concepts/overview/components/>

Pod에서 컨테이너가 확실히 동작하도록 관리, 헬스체크 진행

PodSpec의 집합을 받아 스펙에 정의된 대로 동작시킴

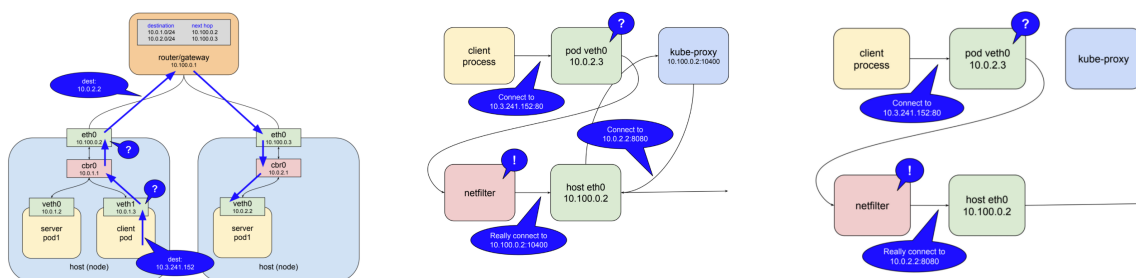
Kubernetes로 만들어진 컨테이너만 관리

### kube-proxy

+ 추가참조 : <https://coffeewhale.com/k8s/network/2019/05/11/k8s-network-02/>

클러스터 내 별도의 가상 네트워크 설정 및 관리

데몬으로 설치가 되어 프로세스가 죽어도 재 시작



1번 그림처럼 Linux Netfilter에 iptables를 이용해 직접적으로 정의되지 않은 ip Server에 접근 가능

kube-proxy가 user space mode로 동작 시, 2번 그림처럼

1. service 요청을 받기 위해 kube-proxy의 포트(10400포트)를 open
2. Netfilter를 이용해 service로 향하는 패킷을 자신에게 라우팅 시키도록 설정

3. server Pod의 IP:Port로 들어오도록 설정 (10.0.2.2:8080)

결론적으로 1번 그림처럼 service IP, 10.3.241.152:80으로 요청을 실제 server pod인 10.0.2.2:8080로 전달하게 됨

## 컨테이너 런타임

+ 추가참조 :

community/container-runtime-interface.md at master · kubernetes/community  
CRI ( Container Runtime Interface) consists of a specifications/requirements (to-be-added), protobuf API, and libraries for container runtimes to integrate with kubelet on a node. The CRI API is currently in Alpha, and the CRI-Docker integration is used by  
<https://github.com/kubernetes/community/blob/master/contributors/devel/sig-node/container-runtime-interface.md>

kubernetes/  
community

Kubernetes community content

1k Contributors 67 Issues 11k Stars 5k Forks

대표적으로 도커가 있고, 런타임 규격에 맞으면 됨

## 애드온

클러스터 안에서 필요한 기능을 실행하는 Pod

네임스페이스는 kube-system

## 네트워킹 애드온

클러스터 내 가상네트워크 구성 시 kube-proxy이외 플러그인 사용

Kubernetes 호스팅 업체에서는 별도의 네트워크 애드온 제공

클러스터 DNS를 갖춘 추가적인 애드온 설치가능,


## DNS 애드온

클러스터 내 DNS레코드 제공, 대표적으로 kube-dns Core DNS이용

## 대시보드 애드온

### 쿠버네티스 대시보드를 배포하고 접속하기

웹 UI(쿠버네티스 대시보드)를 배포하고 접속한다. 대시보드는 웹 기반 쿠버네티스 유저 인터페이스이다. 대시보드를 통해 컨테이너화 된 애플리케이션을 쿠버네티스 클러스터에 배포할 수 있고, 컨테이너화 된 애플리케이션을 트러블슈팅할 수 있다.

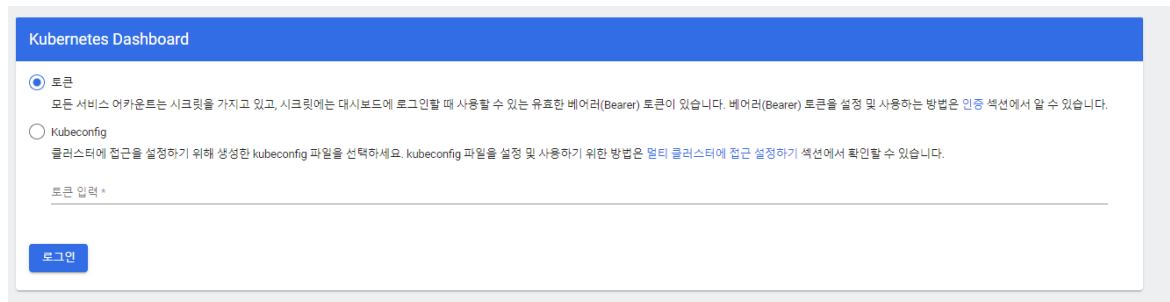
 <https://kubernetes.io/ko/docs/tasks/access-application-cluster/web-ui-dashboard/>

# kubern

```
#대시보드 UI 배포
kubectl apply -f https://raw.githubusercontent.com/kubernetes/dashboard/v2.6.1/aio/deploy/recommended.yaml

#command Proxy
kubectl proxy
```

<http://localhost:8001/api/v1/namespaces/kubernetes-dashboard/services/https:kubernetes-dashboard:/proxy/> 접속 시



The image shows the Kubernetes Dashboard login page. It has a blue header with the text 'Kubernetes Dashboard'. Below the header, there are two radio buttons. The first is '토큰' (Token) and is selected. Below it, there is a paragraph of text explaining that all services are secured with SaaS and that a Bearer token is required. The second radio button is 'Kubeconfig' and is not selected. Below it, there is a paragraph of text explaining that the dashboard can be accessed via a kubeconfig file. At the bottom, there is a text input field labeled '토큰 입력 \*' and a blue button labeled '로그인'.

다음과 같은 인증 및 로그인 접근 권한을 요구합니다.

링크에 접근하니 공식문서와 함께 [대시보드 깃 허브](#) 링크를 찾을 수 있습니다.

우선 서비스 접근이 가능한 어드민 유저를 네임스페이스로 묶어준 뒤 생성합니다.

```
kind: ServiceAccount
metadata:
  name: admin-user
  namespace: kubernetes-dashboard
```

```
PS C:\WorkSpace\infra-k8s-study\nontrust\yamlTemplate\dashboard\user> kubectl apply -f .\dashboard-adminuser.yaml
serviceaccount/admin-user created
```

```
cd C:\WorkSpace\infra-k8s-study\nontrust\yamlTemplate\dashboard\user
ls
kubectl apply -f .\dashboard-adminuser.yaml
```

## 수동

```
apiVersion: rbac.authorization.k8s.io/v1
  kind: ClusterRoleBinding
metadata:
  name: admin-user
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- kind: ServiceAccount
  name: admin-user
  namespace: kubernetes-dashboard
```

계정 생성을 한 뒤 kubernetes-dashboard 네임 스페이스의 admin-user롤을 생성합니다.

```
kubectl -n kubernetes-dashboard create token admin-user
```

[illegible]

이후 생성된 토큰을 토큰에 입력 시 접근 가능합니다.

Kubernetes Dashboard

☒ 토큰

모든 서비스 어카운트는 시크릿을 가지고 있고, 시크릿에는 대시보드에 로그인할 때 사용할 수 있는 유효한 베어리(Bearer) 토큰이 있습니다. 베어리(Bearer) 토큰을 설정 및 사용하는 방법은 [인증](#) 섹션에서 알 수 있습니다.

☐ Kubeconfig

클러스터에 접근을 설정하기 위해 생성한 kubeconfig 파일을 선택하세요. kubeconfig 파일을 설정 및 사용하기 위한 방법은 [멀티 클러스터에 접근 설정하기](#) 섹션에서 확인할 수 있습니다.

토큰 입력 \*

로그인