

Kubernetes(k8s)

- 배의 조타수란 그리스어에서 유래
- 2014년 구글에서 개발
- Kubernetes== k8s
- 컨테이너 orchestration(오케스트레이션)
 - 관현적 편곡, 통합, 결집, 편성, 조직화, 조정
 - 컨테이너와 호스트의 수가 증가함에 따라 여러 컨테이너의 배포 프로세스를 최적화 시킴



K8s 기능

- 컨테이너가 수천 개, 수만 개, 수십억 개가 된다면 관리가 복잡하고, 어려워짐
- 컨테이너를 통합·관리해야 하는 필요에 의해 등장한 기술
- 쿠버네티스를 사용하는 조직은 리눅스 기반의 컨테이너화된 애플리케이션들을 실행하는 호스트 그룹을 묶어 클러스터링을 구성해 사용하는 방식을 채용하고 있음
- 쿠버네티스는 이렇게 구성된 클러스터를 관리
- 또한 클러스터 내 애플리케이션 컨테이너들을 자동으로 배치하고 스케일링하는 등 여러 운영 작업을 자동화함
- 구글 ~ 2020년 기준 G메일, 구글 드라이브 등 애플리케이션을 구동하기 위해 쿠버네티스를 활용, 쿠버네티스로 약 30억 개의 컨테이너를 운영 중
- 국내 대표 스타트업 10곳 중 7곳은 이미 수백~수천 개의 컨테이너를 운영 중

Docker & K8S

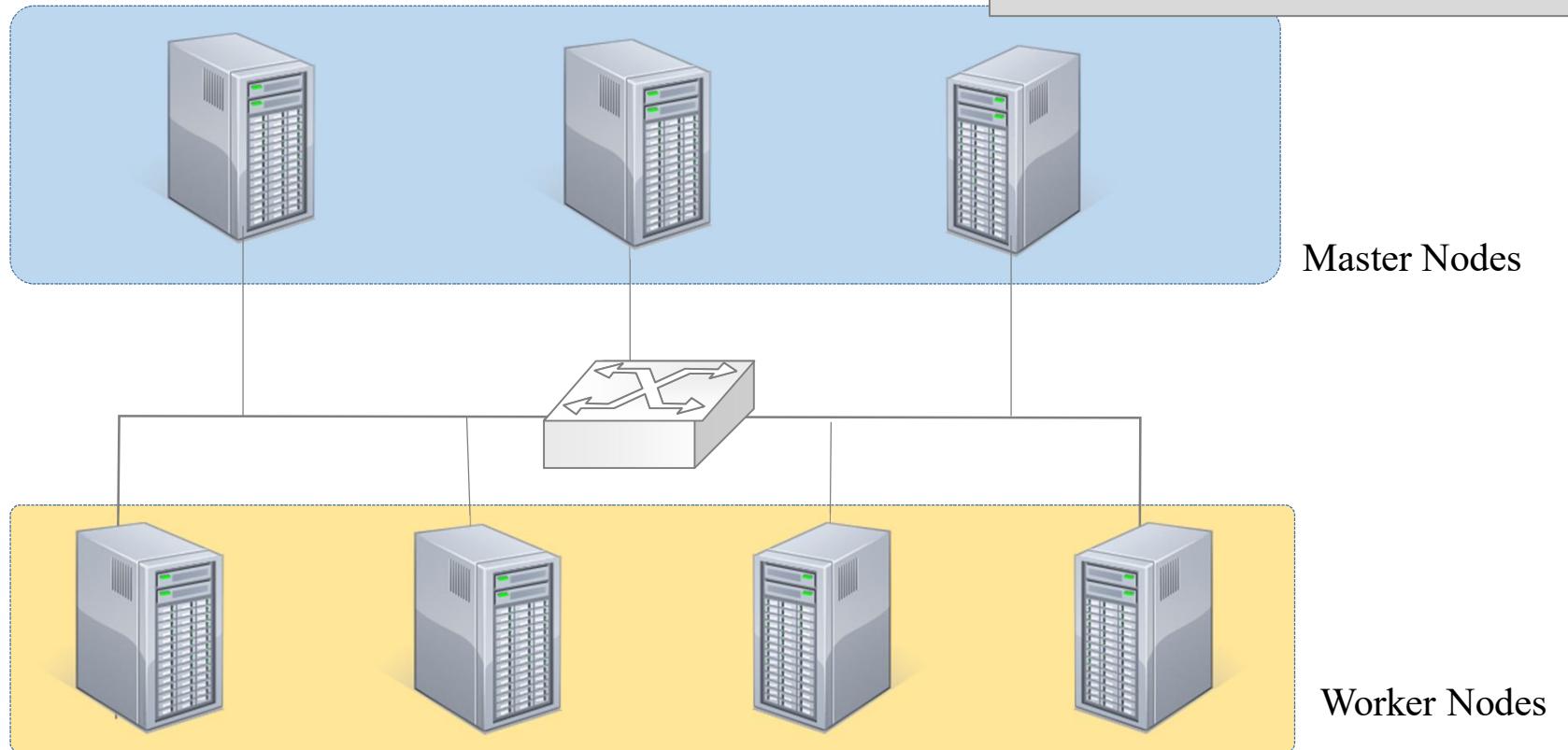
- 컨테이너 오케스트레이션 : 컨테이너의 배포, 관리, 확장, 네트워킹을 자동화

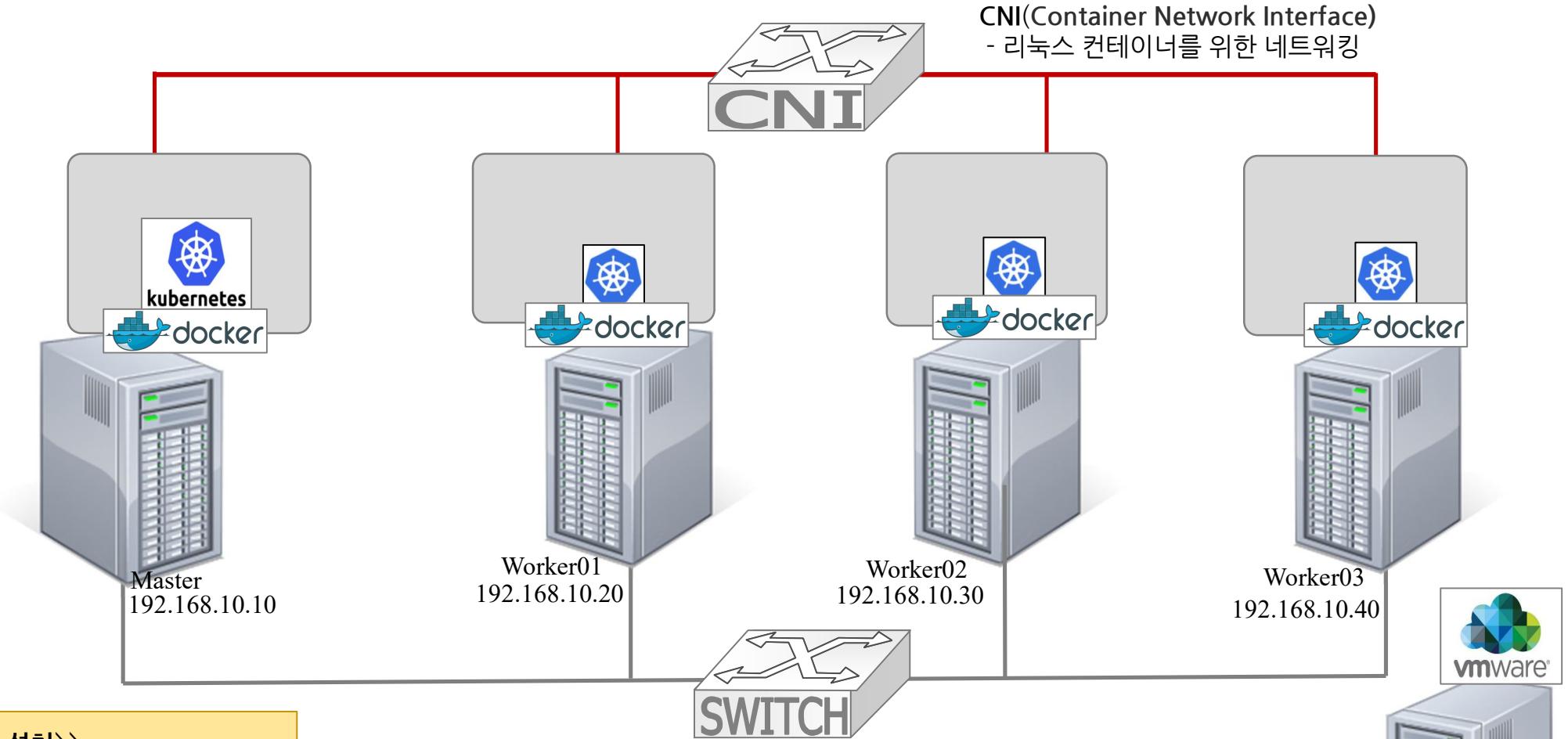
Session Layer (Layer 5)	Orchestration Scheduling	Kubernetes(K8S) , Docker Swarm
Middle Layer (Layer 3~4)	Container Engine	Docker , Rocket, LxC, LxD
	Operating System	Ubuntu , CentOS, Fedora, RHEL
Hardware Layer (Layer 1~2)	Virtual Infrastructure	Openstack, vSphere, Azure
	Physical Infrastructure	Computer, Network, Storage

K8S 클러스터 전체구조

- K8S 클러스터는 크기 두 종류의 서버로 구성
 - Master Nodes : Cluster 관리 노드들
 - Worker Nodes : Container를 실행시키는 노드들

- Leader master : 1대
- Standby master : 2대
- 안정적으로 운영하기 위해 5대 구성도 가능





<<공통 설치>>

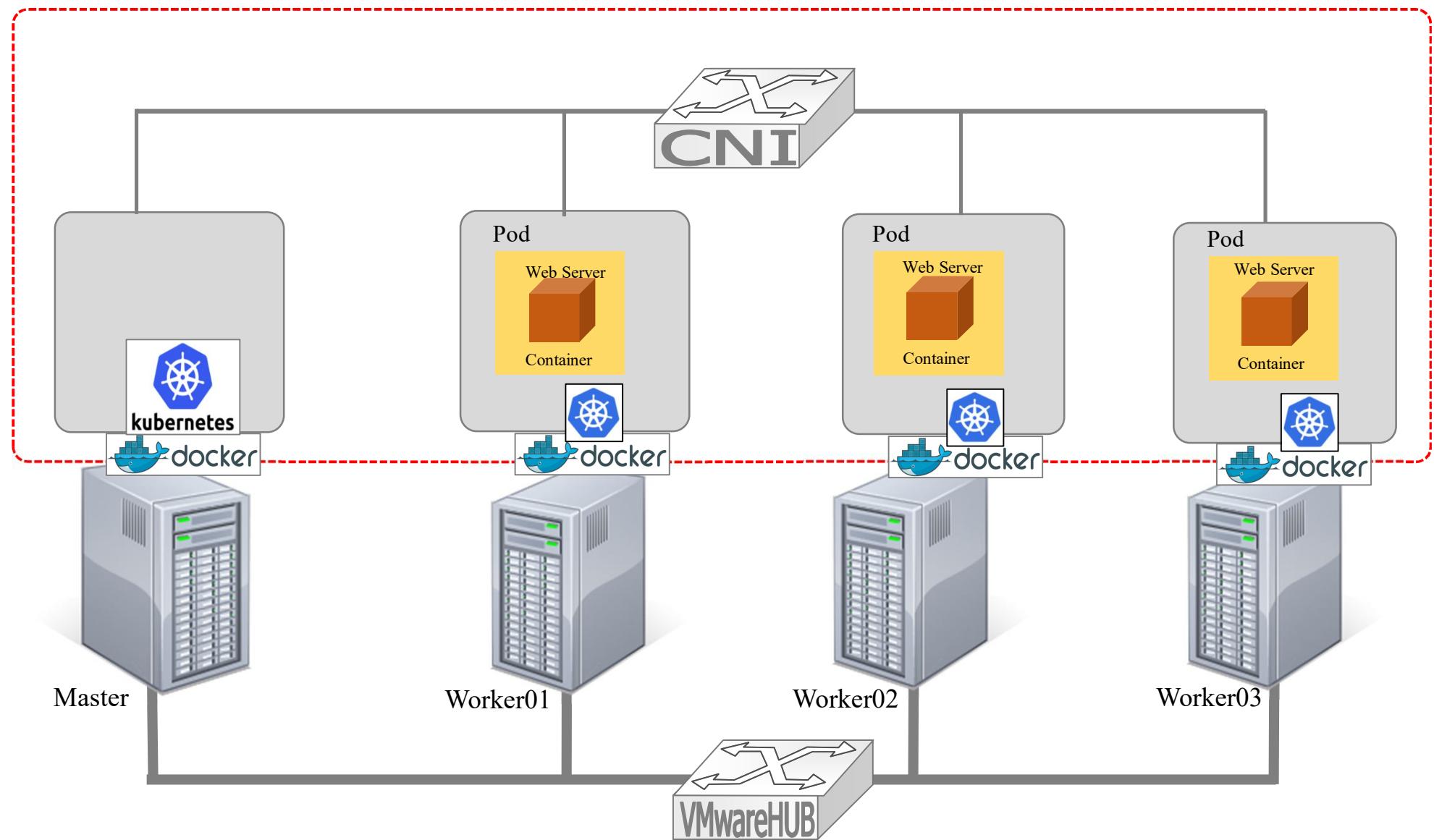
- ① Ubuntu 설치
- ② Docker 설치
- ③ K8s 설치
- Kubeadm, kubect, kubelet

<<개별 설치>>

- ④ [Master] Control plane component 구성
- ⑤ [Master] CNI 구성
- ⑥ [Worker] Worker node 구성 (Master와 Worker node Join)

Windows OS





실습 1.

- kubectl get nodes
- kubectl get nodes -o wide

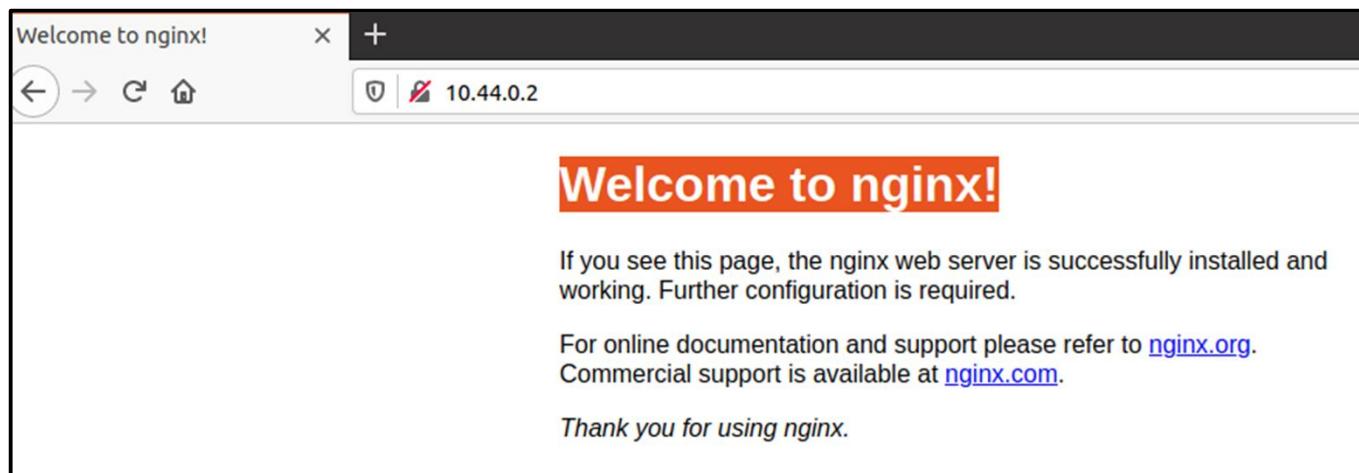
```
root@master:~#  
root@master:~# kubectl get nodes  
NAME      STATUS    ROLES          AGE     VERSION  
master    Ready     control-plane  175d    v1.28.14  
worker01  Ready     <none>        175d    v1.28.14  
worker02  Ready     <none>        175d    v1.28.14  
worker03  Ready     <none>        175d    v1.28.14  
root@master:~#  
root@master:~# kubectl get nodes -o wide  
NAME      STATUS    ROLES          AGE     VERSION   INTERNAL-IP   EXTERNAL-IP   OS-IMAGE           KERNEL-VERSION   CONTAINER-RUNTIME  
master    Ready     control-plane  175d    v1.28.14  192.168.10.100  <none>       Ubuntu 20.04 LTS  5.4.0-26-generic  containerd://1.7.22  
worker01  Ready     <none>        175d    v1.28.14  192.168.10.10   <none>       Ubuntu 20.04 LTS  5.4.0-26-generic  containerd://1.7.22  
worker02  Ready     <none>        175d    v1.28.14  192.168.10.20   <none>       Ubuntu 20.04 LTS  5.4.0-26-generic  containerd://1.7.22  
worker03  Ready     <none>        175d    v1.28.14  192.168.10.30   <none>       Ubuntu 20.04 LTS  5.4.0-26-generic  containerd://1.7.22  
root@master:~#
```

실습 2.

- kubectl run web --image=nginx:1.14 --port 80
- kubectl get pods
- kubectl get pods -o wide

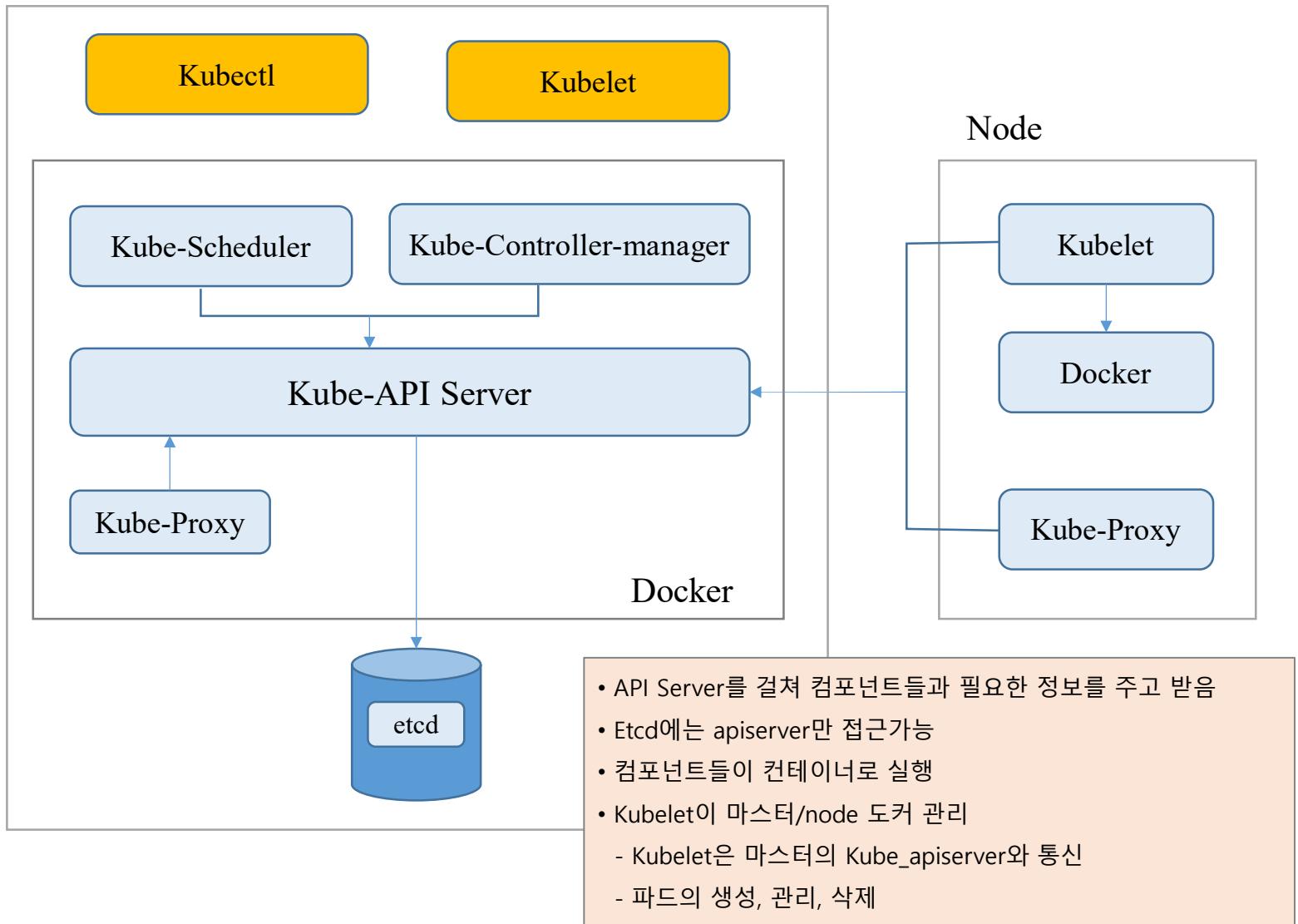
```
root@master:~# kubectl run web --image=nginx:1.14 --port 80
pod/web created
root@master:~# kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
deploy-nginx-bd76d5967-4s25b   1/1     Running   1 (30m ago)   104d
deploy-nginx-bd76d5967-54d7k   1/1     Running   1 (30m ago)   104d
deploy-nginx-bd76d5967-5kqz5   1/1     Running   1 (30m ago)   104d
web            1/1     Running   0           11s
root@master:~#
root@master:~# kubectl get pods -o wide
NAME          READY   STATUS    RESTARTS   AGE      IP           NODE   NOMINATED NODE   READINESS GATES
deploy-nginx-bd76d5967-4s25b   1/1     Running   1 (30m ago)   104d   10.47.0.2   worker03 <none>        <none>
deploy-nginx-bd76d5967-54d7k   1/1     Running   1 (31m ago)   104d   10.36.0.2   worker02 <none>        <none>
deploy-nginx-bd76d5967-5kqz5   1/1     Running   1 (31m ago)   104d   10.44.0.1   worker01 <none>        <none>
web            1/1     Running   0           29s    10.44.0.2   worker01 <none>        <none>
```

```
web          1/1     Running   0      29s  10.44.0.2  worker01  <none>
root@master:~# curl 10.44.0.2
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
body {
    width: 35em;
    margin: 0 auto;
    font-family: Tahoma, Verdana, Arial, sans-serif;
}
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>
```

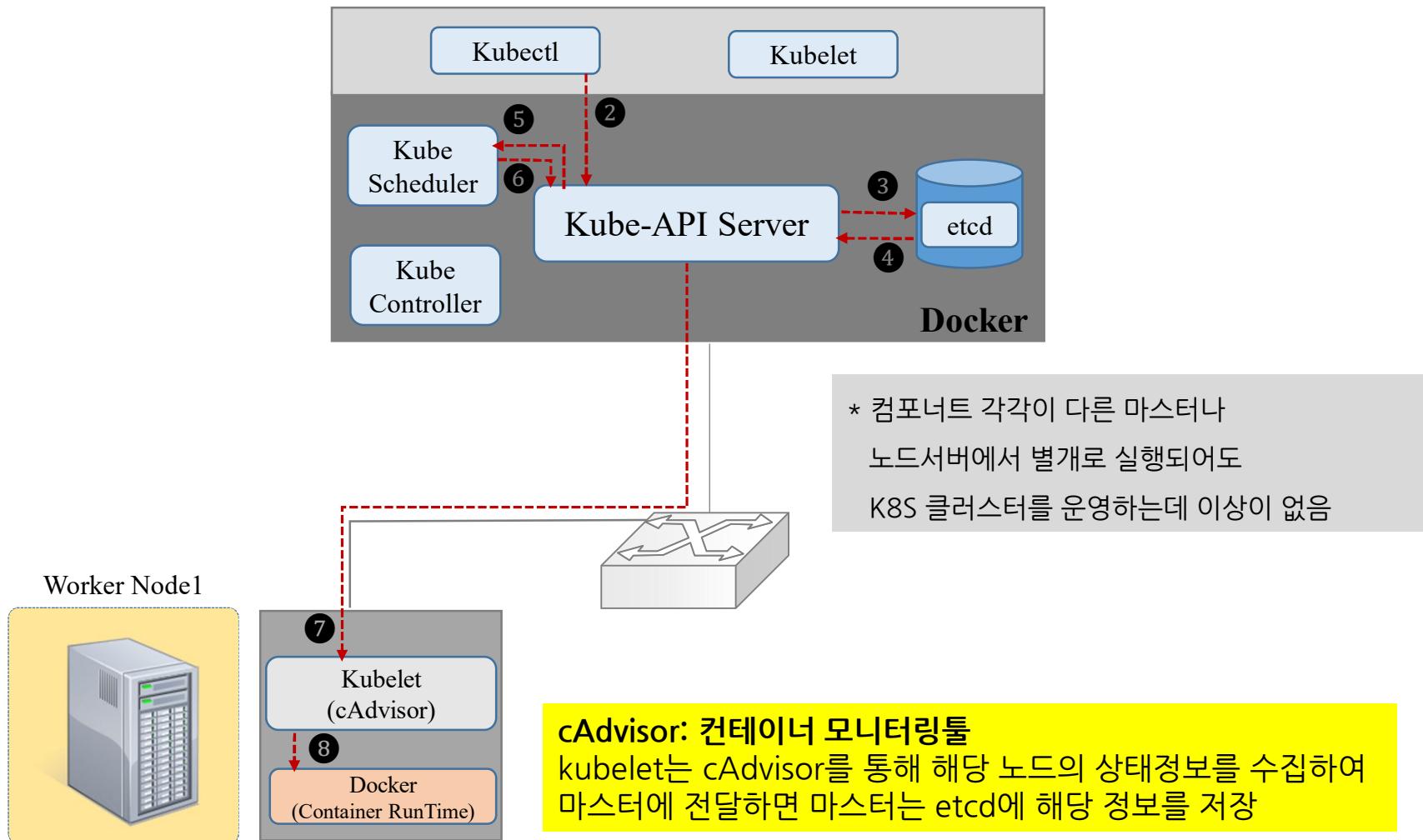


K8S Architecture

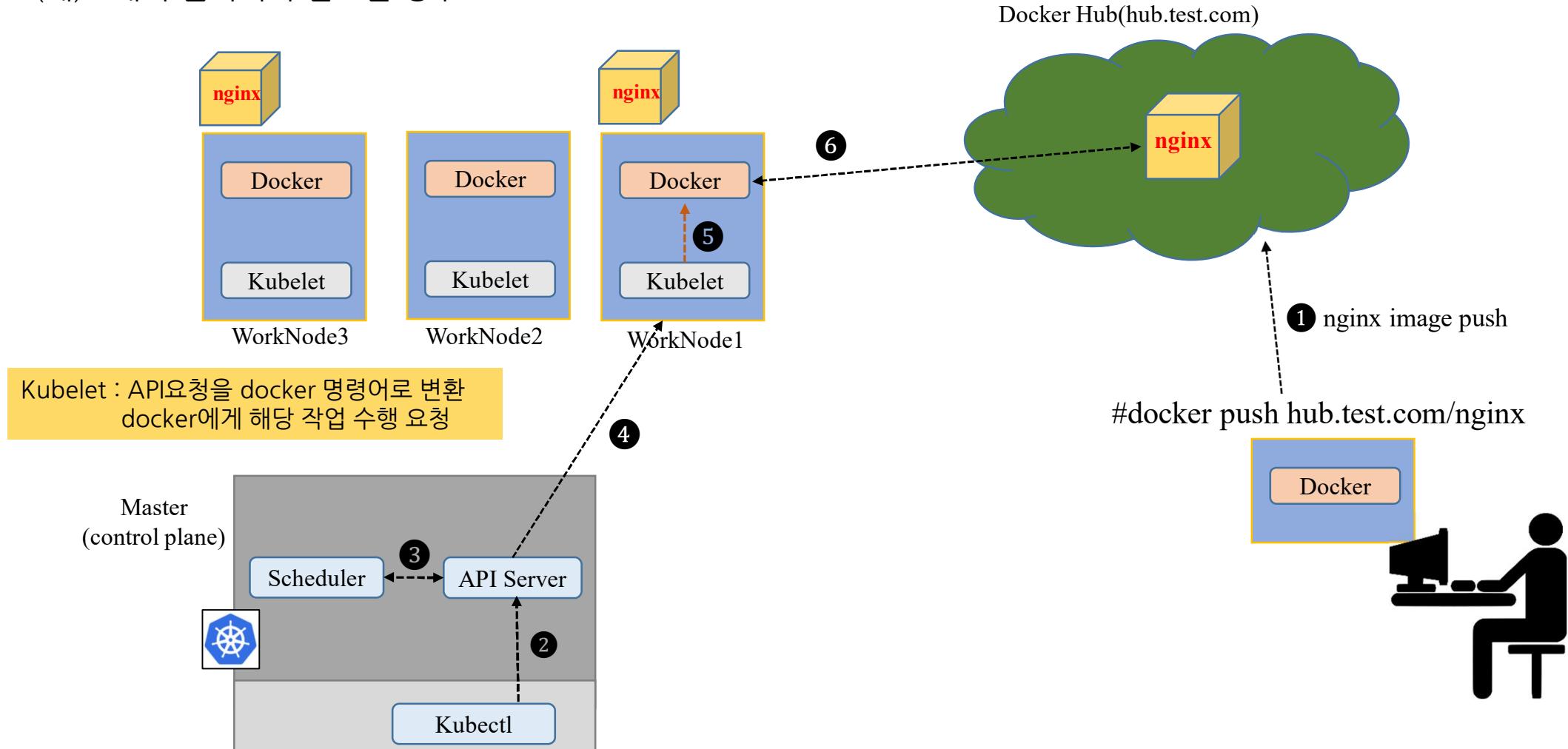
Master



❶ `#kubectl create deploy web --image=hub.test.com/nginx`



(예) 2대의 웹서버가 필요한 경우



K8S 주요 컴포넌트

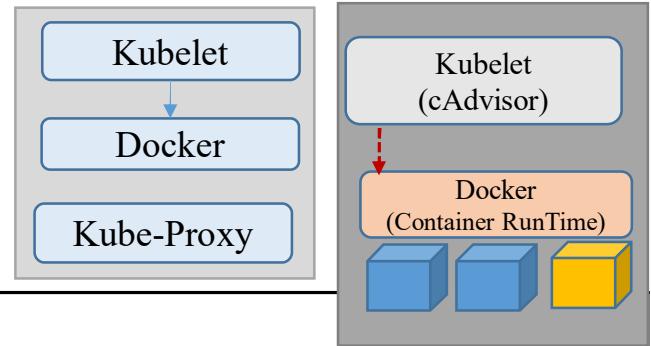
- Master 용 컴포넌트
- Node 용 컴포넌트
- Addon용 컴포넌트 (필수는 아님)

Master(Control Plane) Components

etcd	<p>Key-value 타입의 저장소</p> <p>K8S에서 필요한 모든 데이터를 저장하는 데이터베이스 역할</p> <p>Etcd에 있는 데이터를 백업할 것을 권장</p>	<ul style="list-style-type: none">- 노드들의 자원 상태 정보저장- 컨테이너들의 동작 상태 저장- 다운받은 이미지 상태를 저장- K8s 상태정보 저장
kube-apiserver	<p>K8s cluster API를 사용하도록 하는 컴포넌트</p> <p>클러스터로 온 요청이 유효한지 검사</p> <p>서버 여러 대에 여러 개의 apiserver를 실행해 사용할 수 있음</p>	<ul style="list-style-type: none">- 명령어 문법 점검- 요청을 실행할 권한 여부 검사
kube-scheduler	자원 할당이 가능한 worker 노드 선택	
kube-controller-manager	<p>파드들을 관리하는 컨트롤러</p> <p>Pod를 관찰하며 개수를 보장</p>	
cloud-controller-manager	K8s 컨트롤러들을 클라우드 서비스와 연결해 관리	

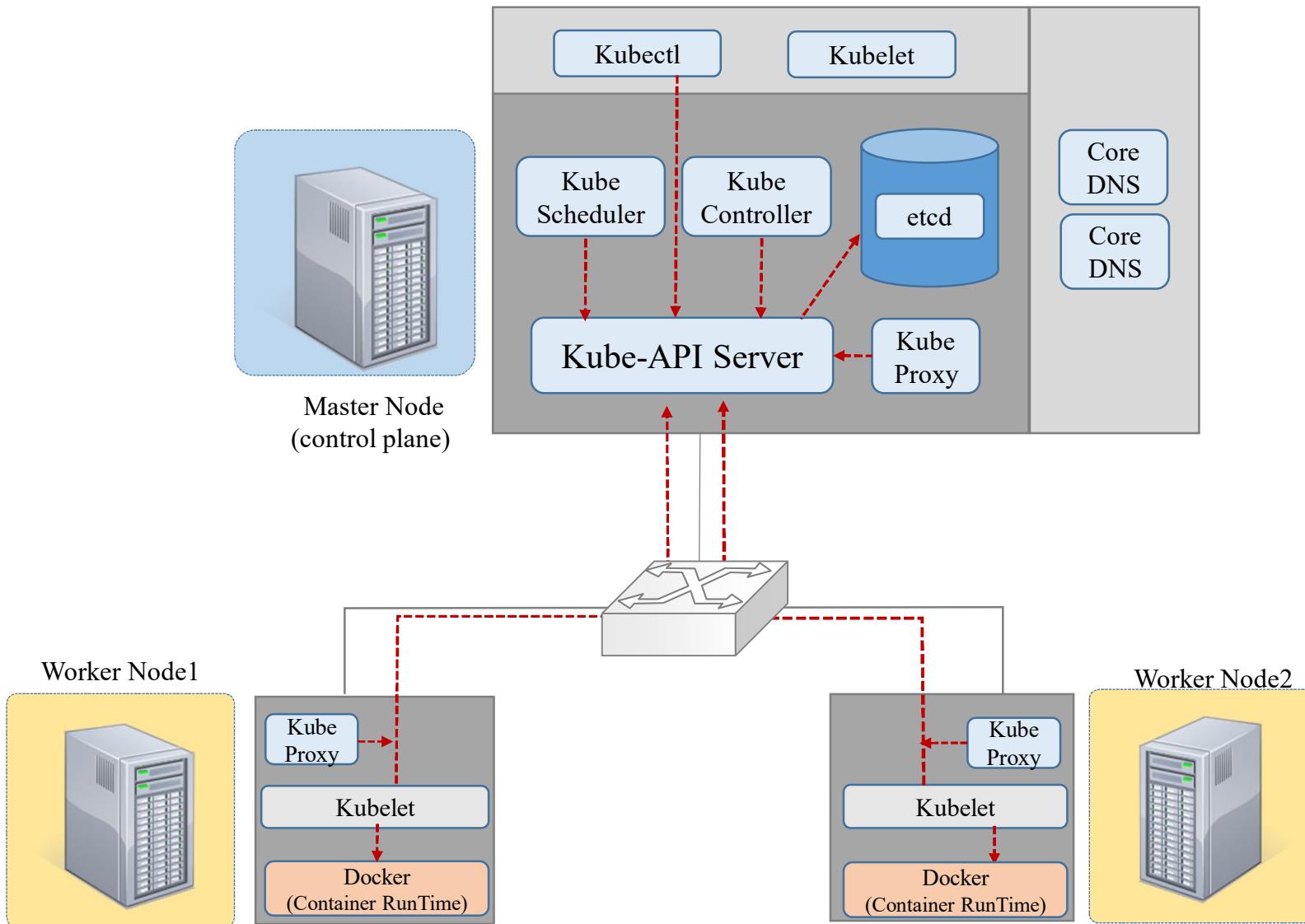
Worker Node Components

* 노드용 컴포넌트는 K8S 실행 환경을 관리

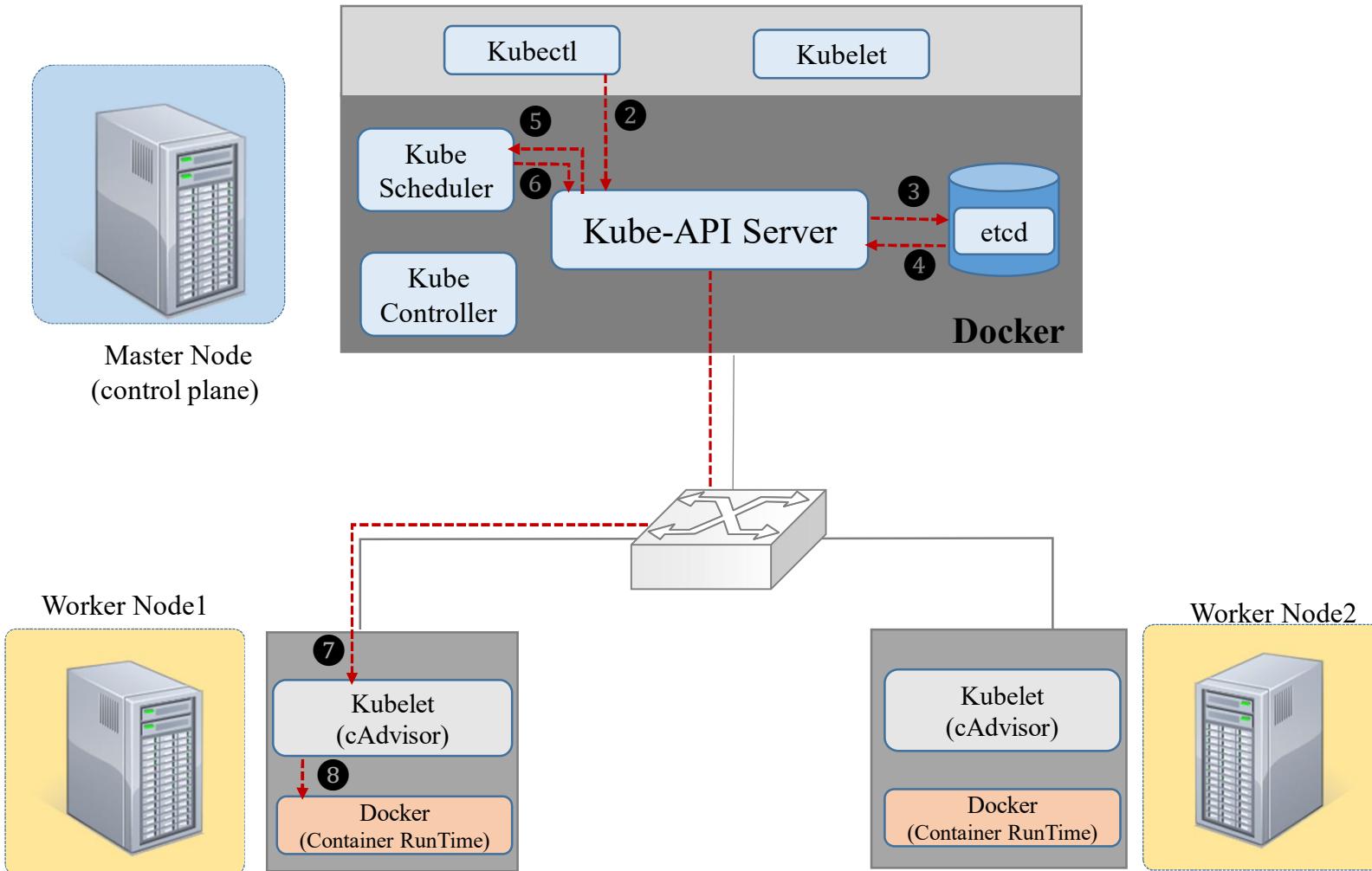


클러스터 안 모든 노드에서 실행되는 k8s agent	
kubelet	<p>Pod container들의 실행을 직접 관리</p> <p>Container가 정상적으로 실행되는지 health check (cAdvisor)</p> <p>노드 안에 있는 컨테이너라도 K8S가 만들지 않는 컨테이너는 관리하지 않음</p>
Kube-proxy	<p>클러스터 안의 별도의 가상 네트워크를 설정하고 관리</p> <p>가상 네트워크 동작을 관리하는 컴포넌트</p> <p>호스트의 네트워크 규칙을 관리하거나 연결을 전달 할 수 있음</p>
Container runtime	<p>실제로 컨테이너를 실행(Container를 실행하는 엔진)</p> <p>Docker, containerd, runc 등</p>

#kubectl create deploy web --image=hub.test.com/nginx



① `#kubectl create deploy web --image=hub.test.com/nginx`

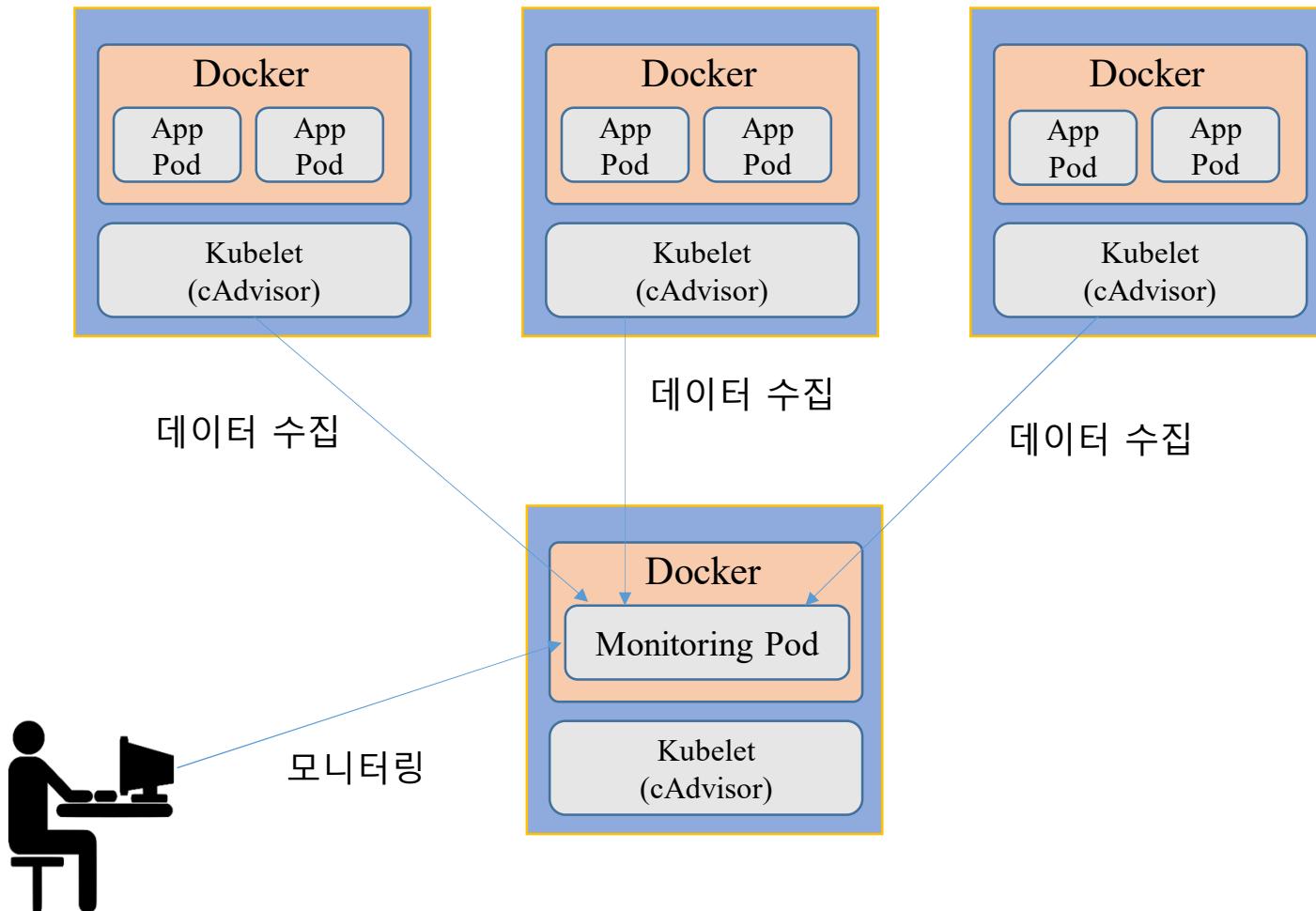


Addons Components

* Addon은 클러스터 안에서 필요한 기능을 실행하는 파드

Networking Addon	가상 네트워크를 구성 시 사용 (네트워크 플러그인) CNI(container Network Interface)
DNS Addon	클러스터 안에서 동작하는 DNS 서버 Kube-DNS와 CoreDNS
Dashboard Addon	웹UI로 K8s를 관리 시 사용
Container resource monitoring	클러스터 안에서 실행중인 컨테이너의 상태를 모니터링 CPU와 메모리 사용량 정보 Kubelet 안에 포함된 cAdvisor라는 컨테이너 모니터링 도구 사용
Clustering logging	클러스터 안 개별 컨테이너의 로그와 k8s 구성요소의 로그들을 중앙화한 로그 수집 중앙 저장 파드로 로그를 수집

* cAdvisor 동작원리 Container resource monitoring



* 클러스터 로깅 동작원리

Clustering logging

