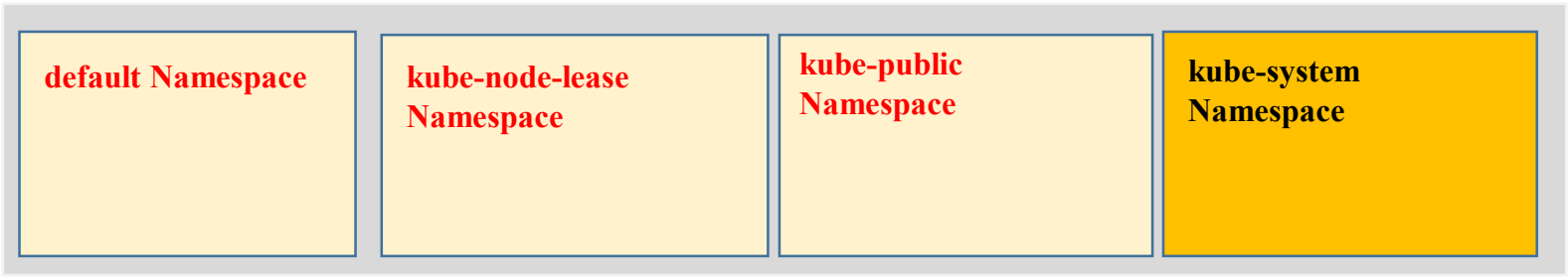
NameSpace

NameSpace

- 클러스터 내에서 리소스를 논리적으로 분리하는 방법 또는 단위
- namespace는 고유한 이름을 가짐
- 해당 네임스페이스 내에서만 오브젝트를 생성하고 관리 가능
- Kubernetes Object 2가지 유형
  - Namespace에 종속되는지 여부에 따라 크게 두 가지로 나뉨
  - Namespaced object : 특정 Namespace 안에 소속되어 존재하는 리소스
  - Cluster-scoped object ：네임스페이스와 상관없이 전체 클러스터에서 공유됨

kubectl get namespaces

```
root@master:/k8s# kubectl get namespaces
NAME                STATUS   AGE
calico-apiserver    Active   14d
calico-system       Active   14d
default             Active   14d
kube-node-lease     Active   14d
kube-public         Active   14d
kube-system         Active   14d
tigera-operator     Active   14d
root@master:/k8s#
```



NameSpace 종류

항목	설명	
default	별도로 Namespace를 지정하지 않으면 자동으로 사용 일반 애플리케이션(Pod, Service 등)이 주로 배치됨	Namespaced object
kube-system	Kubernetes 핵심 시스템 컴포넌트	Cluster-scoped object
kube-node-lease	노드 상태(heartbeat) 관리 전용 각 노드가 자신의 Lease 객체를 갱신 대규모 클러스터에서 성능 최적화 목적 관리자가 직접 다룰 일은 거의 없음	
kube-public	모든 사용자(인증 없이도) 읽기 가능 클러스터 공용 정보 저장 용도 실제 사용 빈도는 매우 낮음	

kube-system

- Kubernetes가 설치될 때 자동으로 생성되는 기본 네임스페이스 중 하나
- 클러스터의 제어·운영·관리 기능을 담당하는 핵심 컴포넌트들이 배치되는 공간
- 사용자가 직접 애플리케이션을 배포하는 공간이 아니라, 클러스터 내부 시스템용 공간

kubectl get pod –n kube-system

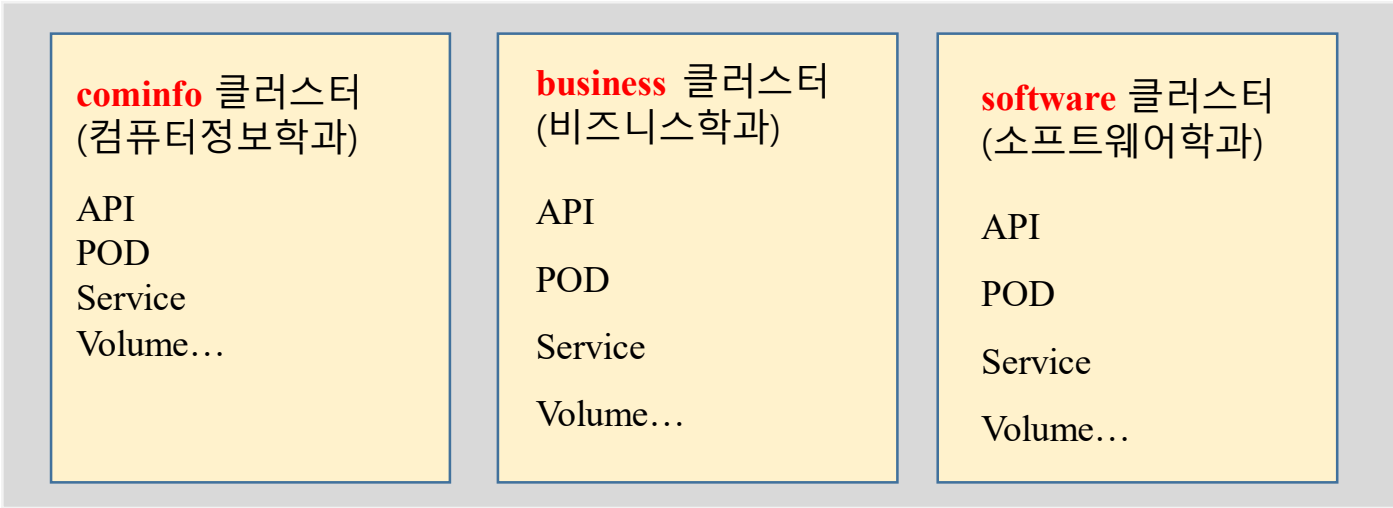
```
root@master:/K8S# kubectl get pods -n kube-system
NAME                              READY   STATUS    RESTARTS       AGE
coredns-7c65d6cfc9-wc8ts          1/1     Running   3 (8h ago)     66d
coredns-7c65d6cfc9-x68p2          1/1     Running   3 (8h ago)     66d
etcd-master                       1/1     Running   17 (8h ago)    66d
kube-apiserver-master             1/1     Running   21 (8h ago)    66d
kube-controller-manager-master    1/1     Running   26 (8h ago)    66d
kube-proxy-jkvlf                  1/1     Running   14 (8h ago)    66d
kube-proxy-p7xtm                  1/1     Running   2 (8h ago)     66d
kube-proxy-qhtlv                  1/1     Running   2 (8h ago)     66d
kube-proxy-vmx9w                  1/1     Running   3 (8h ago)     66d
kube-scheduler-master             1/1     Running   28 (8h ago)    66d
```

kubectl get pod –n kube-system –o wide

```
root@master:/K8S# kubectl get pods -n kube-system -o wide
NAME                              READY   STATUS    RESTARTS      AGE   IP               NODE       NOMINATED NODE   READINESS G
ATES
coredns-7c65d6cfc9-wc8ts          1/1     Running   3 (8h ago)    66d   192.168.219.85   master     <none>           <none>
coredns-7c65d6cfc9-x68p2          1/1     Running   3 (8h ago)    66d   192.168.219.86   master     <none>           <none>
etcd-master                       1/1     Running   17 (8h ago)   66d   192.168.10.10    master     <none>           <none>
kube-apiserver-master             1/1     Running   21 (8h ago)   66d   192.168.10.10    master     <none>           <none>
kube-controller-manager-master    1/1     Running   26 (8h ago)   66d   192.168.10.10    master     <none>           <none>
kube-proxy-jkvlf                  1/1     Running   14 (8h ago)   66d   192.168.10.10    master     <none>           <none>
kube-proxy-p7xtm                  1/1     Running   2 (8h ago)    66d   192.168.10.40    worker03   <none>           <none>
kube-proxy-qhtlv                  1/1     Running   2 (8h ago)    66d   192.168.10.30    worker02   <none>           <none>
kube-proxy-vmx9w                  1/1     Running   3 (8h ago)    66d   192.168.10.20    worker01   <none>           <none>
kube-scheduler-master             1/1     Running   28 (8h ago)   66d   192.168.10.10    master     <none>           <none>
```

NameSpace

- K8S API 종류 중 하나
- K8S cluster 하나를 여러 개 논리적인 단위로 나눠 사용하는 것
- K8S Cluster 하나를 여러 개 팀이나 사용자가 함께 공유 할 수 있음



물리적 K8S 클러스터

현재 생성된 NameSpace 확인

kubectl get namespaces

```
root@master:/k8s# kubectl get namespaces
NAME                STATUS   AGE
calico-apiserver    Active   65d
calico-system       Active   65d
default             Active   65d
kube-node-lease     Active   65d
kube-public         Active   65d
kube-system         Active   65d
tigera-operator     Active   65d
```

default Namespace	**kube-node-lease Namespace**	**kube-public Namespace**	**kube-system Namespace**	**calico-system Namespace**

❶ NameSpace 생성

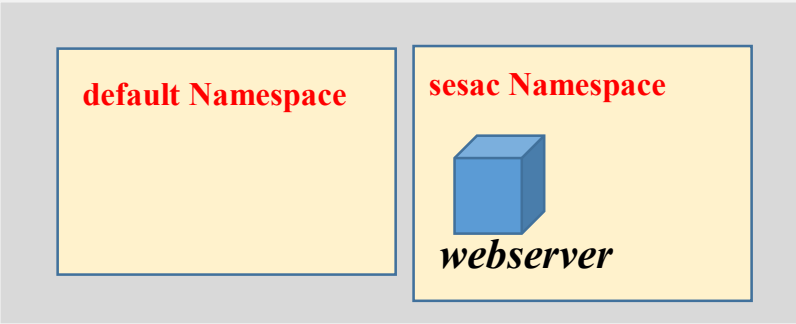
kubectl create namespace **sesac**

```
root@master:/k8s# kubectl create namespace sesac
namespace/sesac created
root@master:/k8s# kubectl get namespaces
NAME                STATUS   AGE
calico-apiserver    Active   65d
calico-system       Active   65d
default             Active   65d
kube-node-lease     Active   65d
kube-public         Active   65d
kube-system         Active   65d
sesac               Active   5s
tigera-operator     Active   65d
```

❷ 지정된 NameSpace에 pod 생성

kubectl run webserver –image=nginx:1.14 --port 80 --namespace **sesac**
kubectl get pods --namespace sesac

```
root@master:/k8s# kubectl run webserver --image=nginx:1.14 --port 80 --namespace sesac
pod/webserver created
root@master:/k8s# kubectl get pod
No resources found in default namespace.
root@master:/k8s# kubectl get pod --namespace sesac
NAME        READY   STATUS    RESTARTS   AGE
webserver   1/1     Running   0          27s
root@master:/k8s#
```



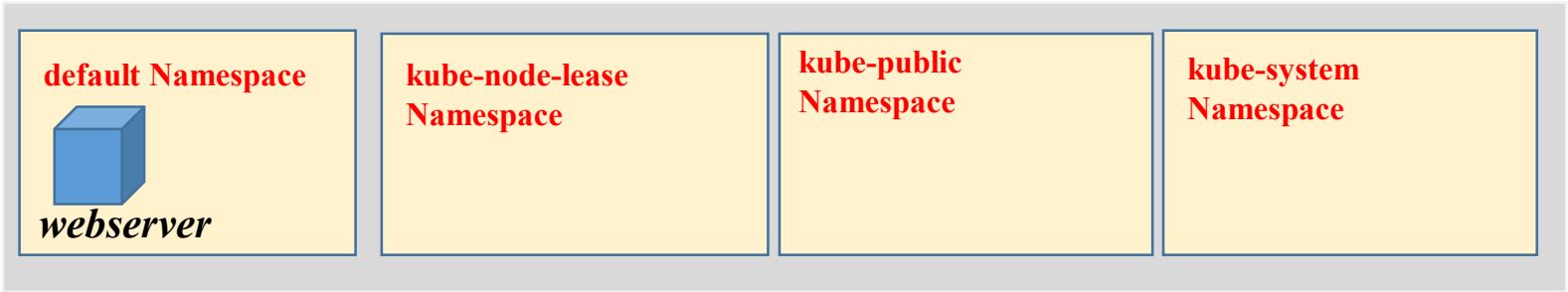
기본 namespace

kubectl run web --image=nginx:1.14 --port 80

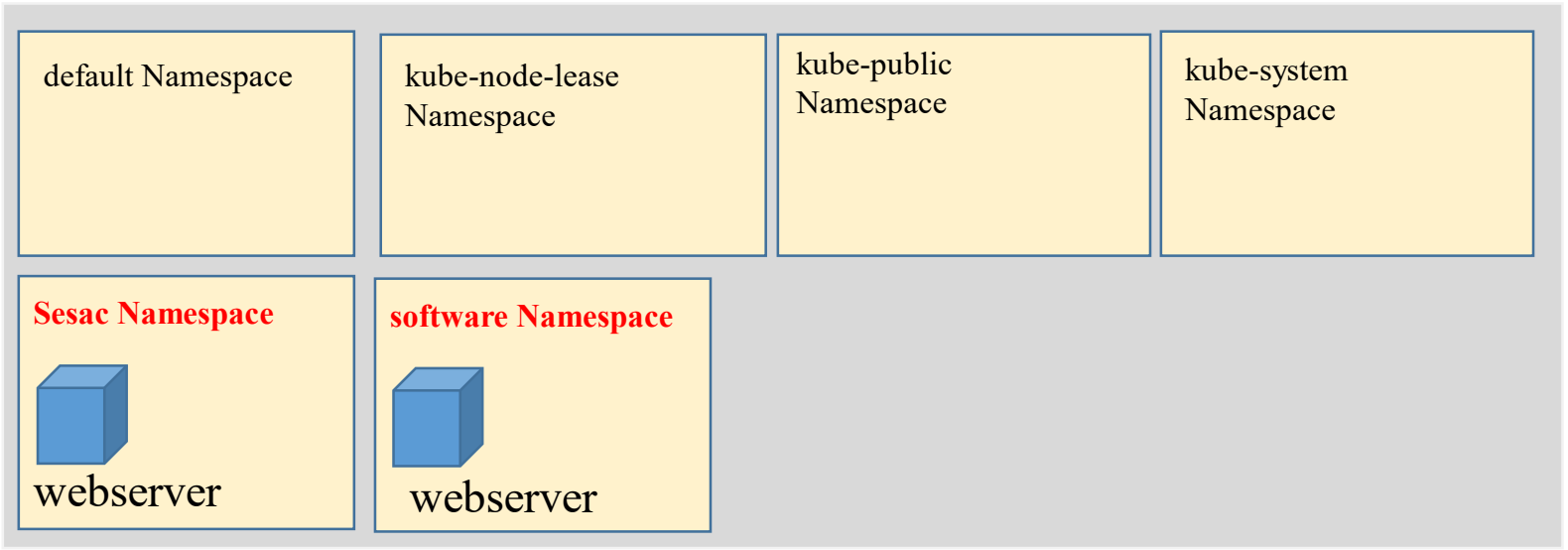
```
controlplane $ kubectl run web --image=nginx:1.14 --port 80
pod/web created
controlplane $ kubectl get pods
NAME   READY   STATUS    RESTARTS   AGE
web    1/1     Running   0          10s
```

```
controlplane $ kubectl get pod
No resources found in default namespace.
controlplane $
```

kubectl get pod



실습 1. Namespace 생성 후 지정된 cluster에 nginx server 생성 배치



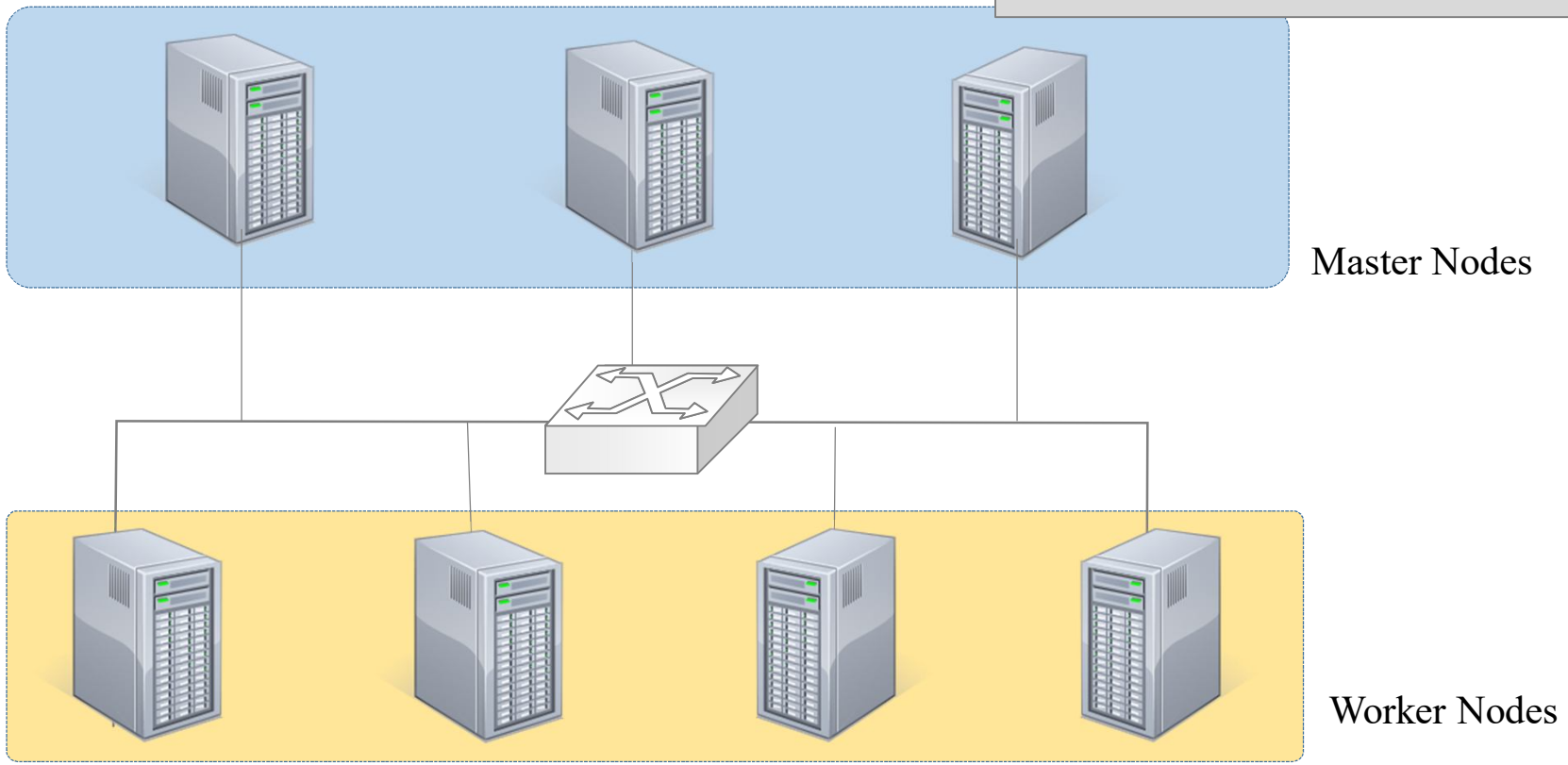
- software Namespace 생성
- 생성된 software namespace에 nginx:1.14 Webserver pod 배치

K8S Architecture

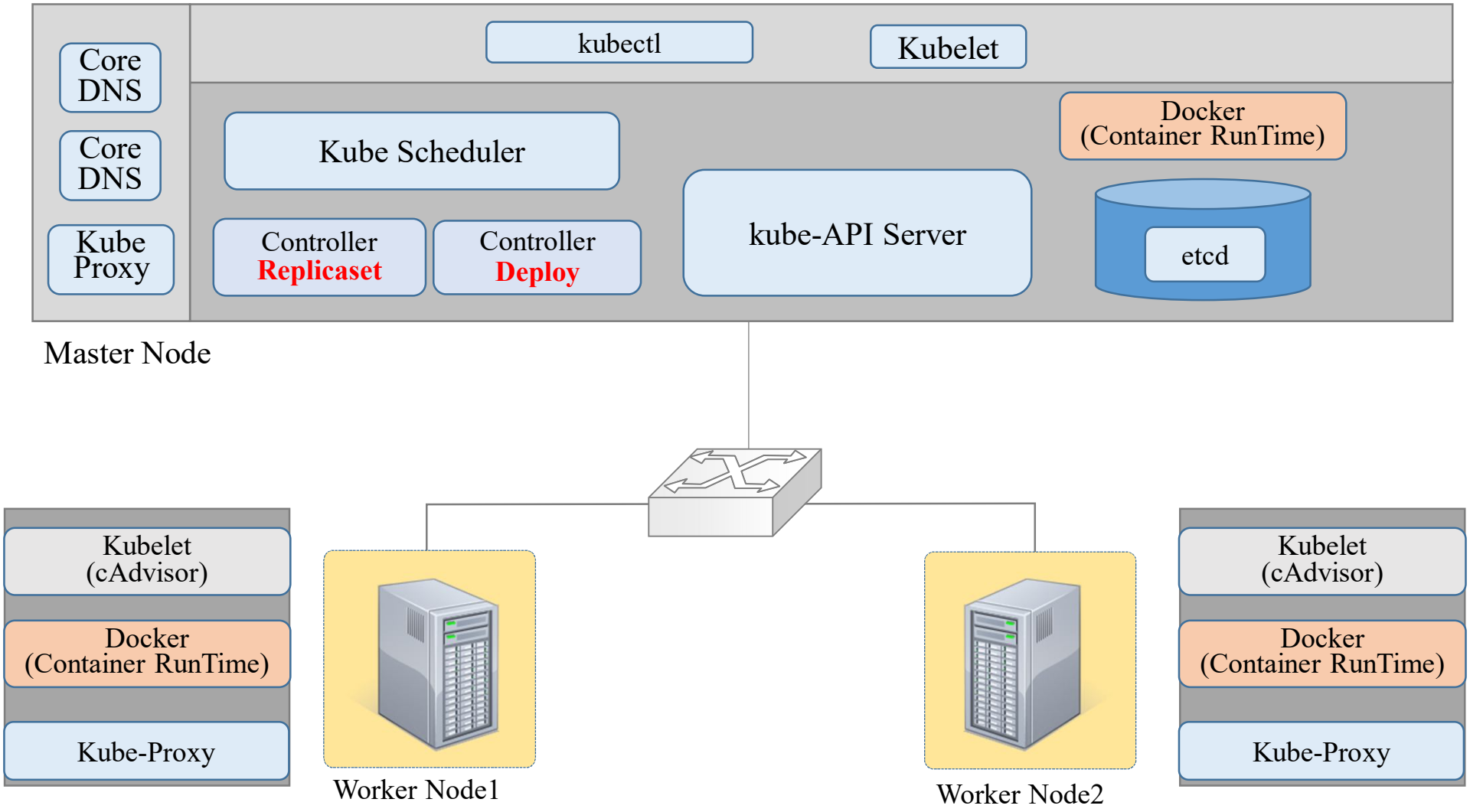
K8S 클러스터 전체구조

- 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

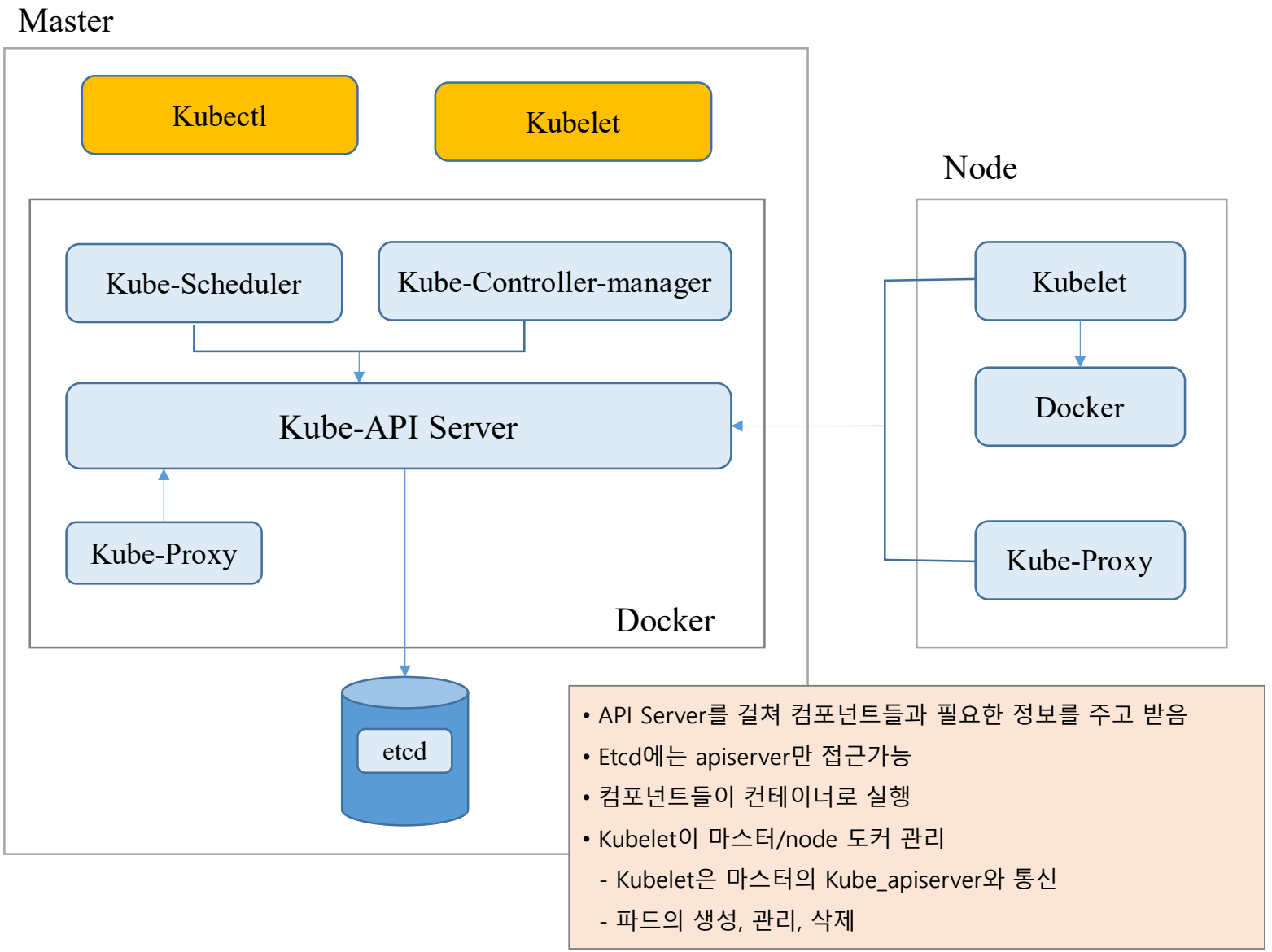


kube-system의 주요 컴포넌트

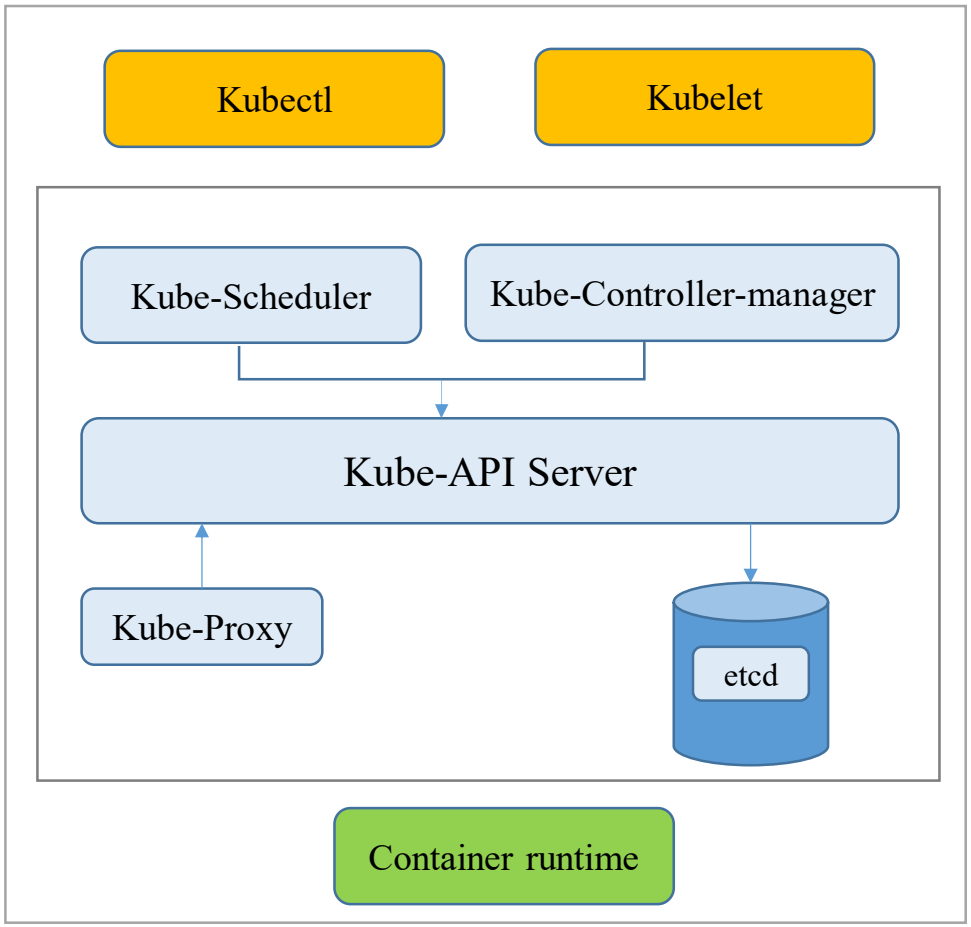


kube-system의 주요 컴포넌트

CoreDNS	내부 DNS 서비스 Pod 이름을 기반으로 IP 변환
kube-proxy	노드 내에서 서비스 트래픽 iptables/ipvs 라우팅 담당 Service를 이용하여 Pod로 패킷 전달 기능
etcd	Kubernetes의 모든 상태가 저장되는 분산 Key-Value DB
kube-apiserver, scheduler, controller-manager	Control Plane에서 실행되는 Kubernetes 핵심 컴포넌트들
CNI 플러그인	Pod 간 네트워크 연결 관리


Master
Kubectl
Kubelet
Kube-Scheduler
Kube-Controller-manager
Kube-API Server
Kube-Proxy
Docker
etcd
Node
Kubelet
Docker
Kube-Proxy
• API Server를 걸쳐 컴포넌트들과 필요한 정보를 주고 받음
• Etcd에는 apiserver만 접근가능
• 컴포넌트들이 컨테이너로 실행
• Kubelet이 마스터/node 도커 관리
- Kubelet은 마스터의 Kube_apiserver와 통신
- 파드의 생성, 관리, 삭제

Master Components



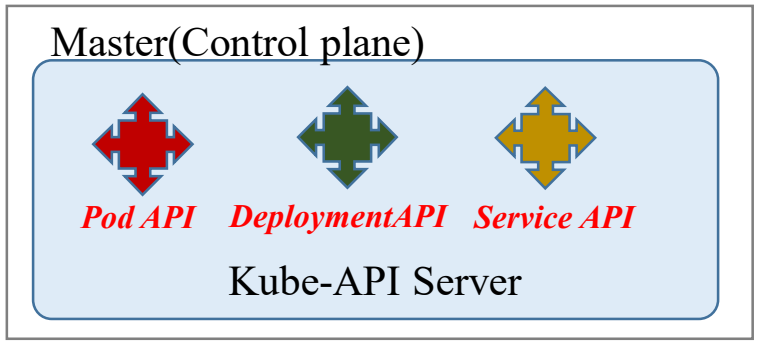
#kubectl get pod –n kube-system

```
root@master:/k8s# kubectl get pods -n kube-system
NAME                                 READY   STATUS
coredns-7c65d6cfc9-wc8ts             1/1     Running
coredns-7c65d6cfc9-x68p2             1/1     Running
etcd-master                          1/1     Running
kube-apiserver-master                1/1     Running
kube-controller-manager-master       1/1     Running
kube-proxy-jkvlf                     1/1     Running
kube-proxy-p7xtm                     1/1     Running
kube-proxy-qhtlv                     1/1     Running
kube-proxy-vmx9w                     1/1     Running
kube-scheduler-master                1/1     Running
```

*K8S 클러스터의 핵심 시스템 컴포넌트들이 정상 동작 중인지 확인 명령어

API Version

- K8S Object 정의 시 API 버전 필요, Update 된 API가 있다면 새로운 API로 생성 (Alpha → Beta → Stable)



#kubectl api-resources

NAME	SHORTNAMES	APIVERSION	NAMESPACED	KIND
bindings		v1	true	Binding
componentstatuses	cs	v1	false	ComponentStatus
configmaps	cm	v1	true	ConfigMap
endpoints	ep	v1	true	Endpoints
events	ev	v1	true	Event
limitranges	limits	v1	true	LimitRange
namespaces	ns	v1	false	Namespace
nodes	no	v1	false	Node
persistentvolumeclaims	pvc	v1	true	PersistentVolumeClaim
persistentvolumes	pv	v1	false	PersistentVolume
pods	po	v1	true	Pod
podtemplates		v1	true	PodTemplate
replicationcontrollers	rc	v1	true	ReplicationController
resourcequotas	quota	v1	true	ResourceQuota
secrets		v1	true	Secret
serviceaccounts	sa	v1	true	ServiceAccount
services	svc	v1	true	Service

K8S Object API Version 확인

#kubectl explain pod

#kubectl explain deployment

Object API	Version
Pod	v1
Deployment	Apps/v1
ReplicaSet	Apps/v1
service	v1

```
controlplane $ kubectl explain pod
KIND:     Pod
VERSION:  v1
```

```
controlplane $ kubectl explain deployment
KIND:     Deployment
VERSION:  apps/v1
```

kubectl api-resources

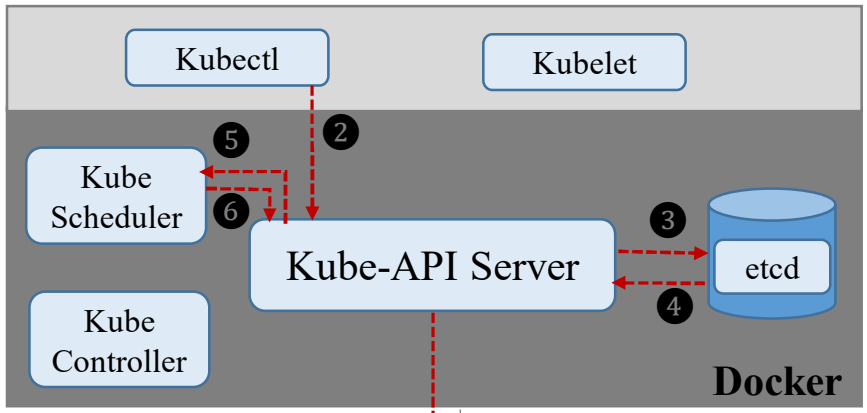
```
root@masternode:~# kubectl api-resources
NAME                              SHORTNAMES   APIVERSION   NAMESPACED   KIND
bindings                                       v1           true         Binding
componentstatuses                 cs           v1           false        ComponentStatus
configmaps                        cm           v1           true         ConfigMap
endpoints                         ep           v1           true         Endpoints
events                            ev           v1           true         Event
limitranges                       limits       v1           true         LimitRange
namespaces                        ns           v1           false        Namespace
nodes                             no           v1           false        Node
```

kubectl api-resources | grep deploy

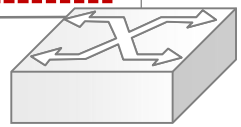
K8S 주요 컴포넌트

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

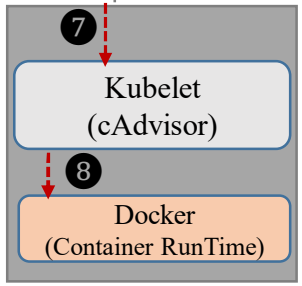
❶ *#kubectl create deploy web --image=hub.test.com/nginx*



* 컴포넌트 각각이 다른 마스터나
노드서버에서 별개로 실행되어도
K8S 클러스터를 운영하는데 이상이 없음



Worker Node1



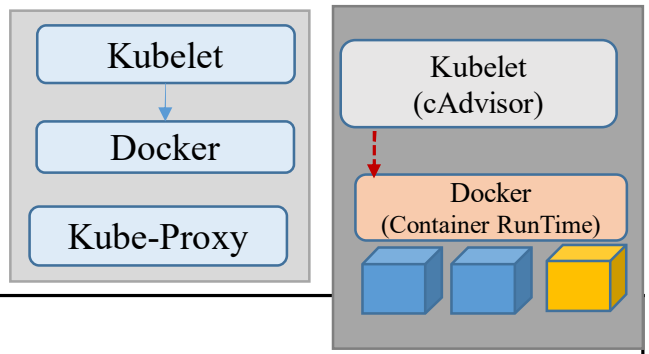
cAdvisor: 컨테이너 모니터링툴
kubelet는 cAdvisor를 통해 해당 노드의 상태정보를 수집하여
마스터에 전달하면 마스터는 etcd에 해당 정보를 저장

Master(Control Plane) Components

etcd	Key-value 타입의 저장소 K8S에서 필요한 모든 데이터를 저장하는 데이타베이스 역할 Etcd에 있는 데이터를 백업할 것을 권장	- 노드들의 자원 상태 정보저장 - 컨테이너들의 동작 상태 저장 - 다운받은 이미지 상태를 저장 - K8s 상태정보 저장
kube-apiserver	K8s cluster API를 사용하도록 하는 컴포넌트 클러스터로 온 요청이 유효한지 검사 서버 여러 대에 여러 개의 apiserver를 실행해 사용할 수 있음	- 명령어 문법 점검 - 요청을 실행할 권한 여부 검사
kube-scheduler	자원 할당이 가능한 worker 노드 선택	
kube-controller-manager	파드들을 관리하는 컨트롤러 Pod를 관찰하며 개수를 보장	
cloud-controller-manager	K8s 컨트롤러들을 클라우드 서비스와 연결해 관리	

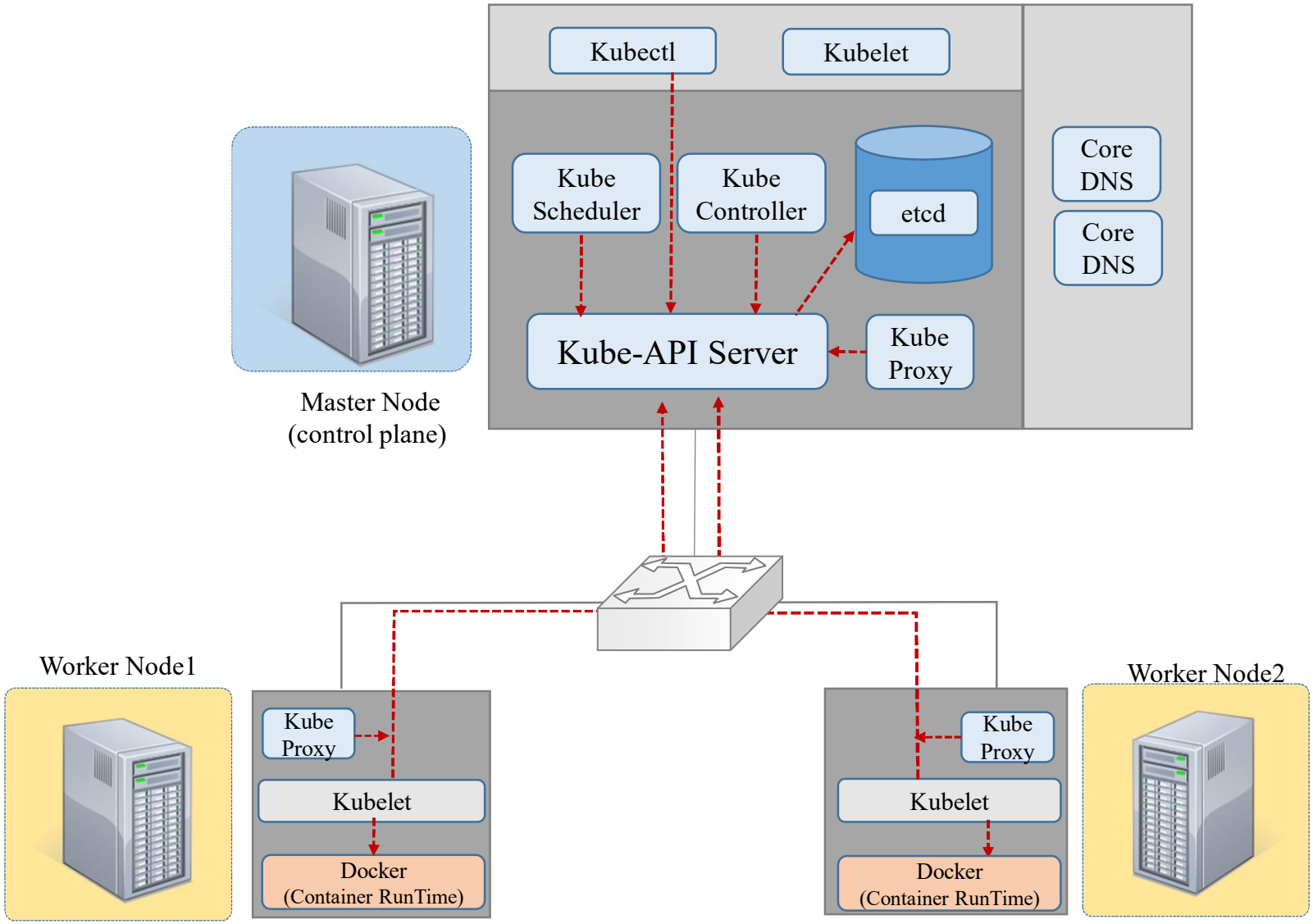
Worker Node Components

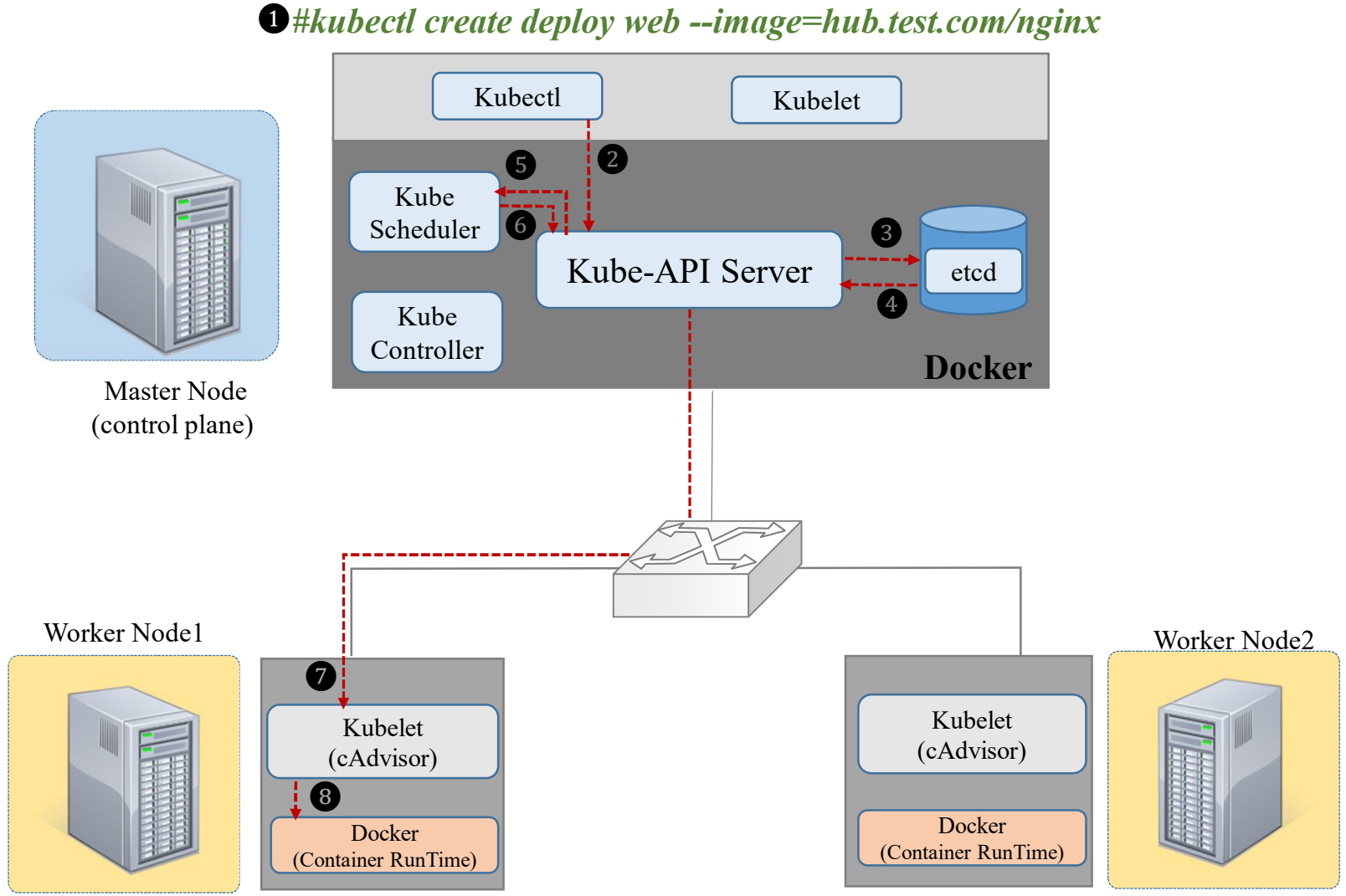
* 노드용 컴포넌트는 K8S 실행 환경을 관리



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

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




❶#kubectl create deploy web --image=hub.test.com/nginx
Kubectl
Kubelet
Kube Scheduler
Kube-API Server
etcd
Kube Controller
Docker
Master Node
(control plane)
Worker Node1
Worker Node2
Kubelet
(cAdvisor)
Docker
(Container RunTime)
Kubelet
(cAdvisor)
Docker
(Container RunTime)

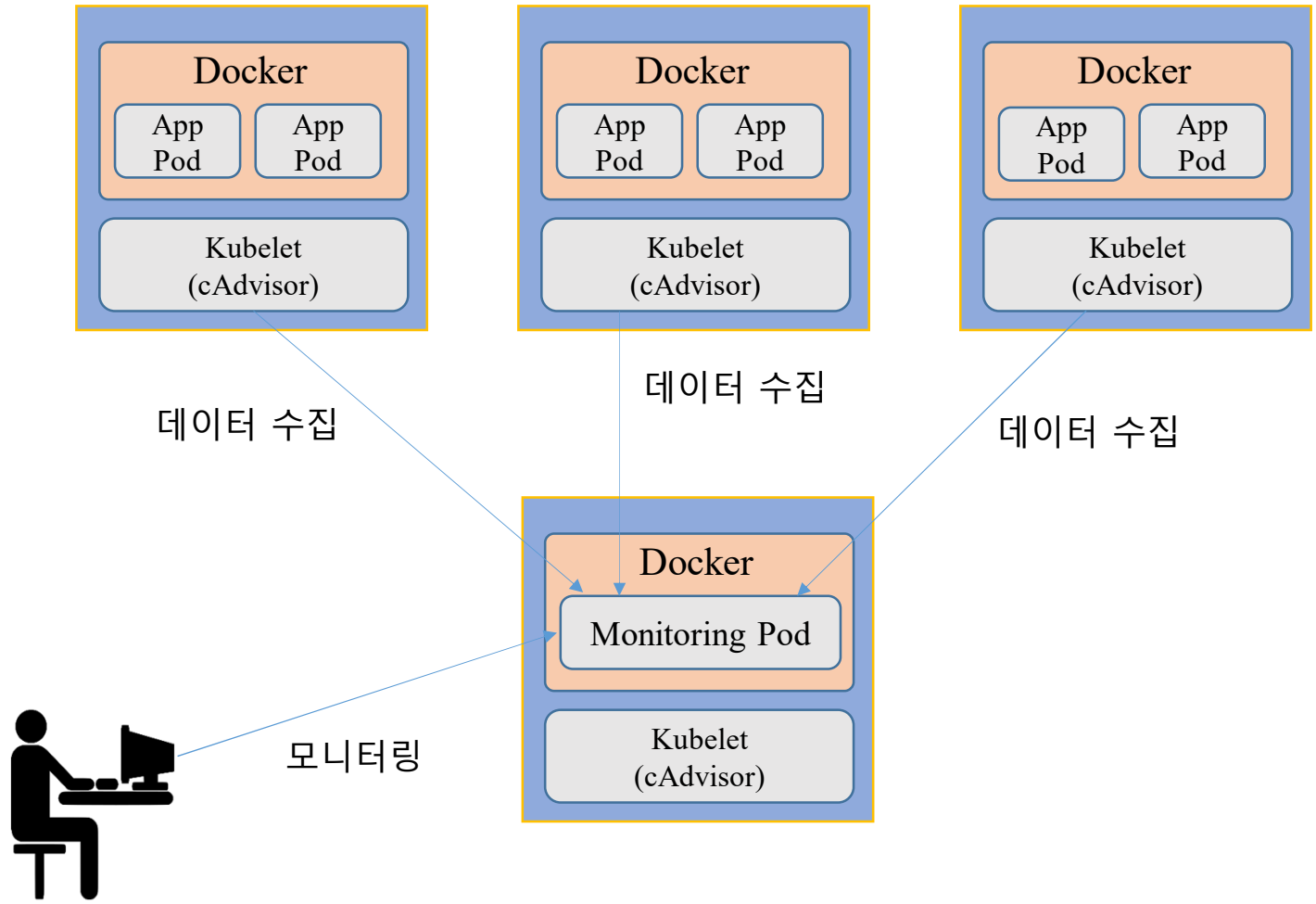
Addons Components

* Addon은 클러스터 안에서 필요한 기능을 실행하는 파드

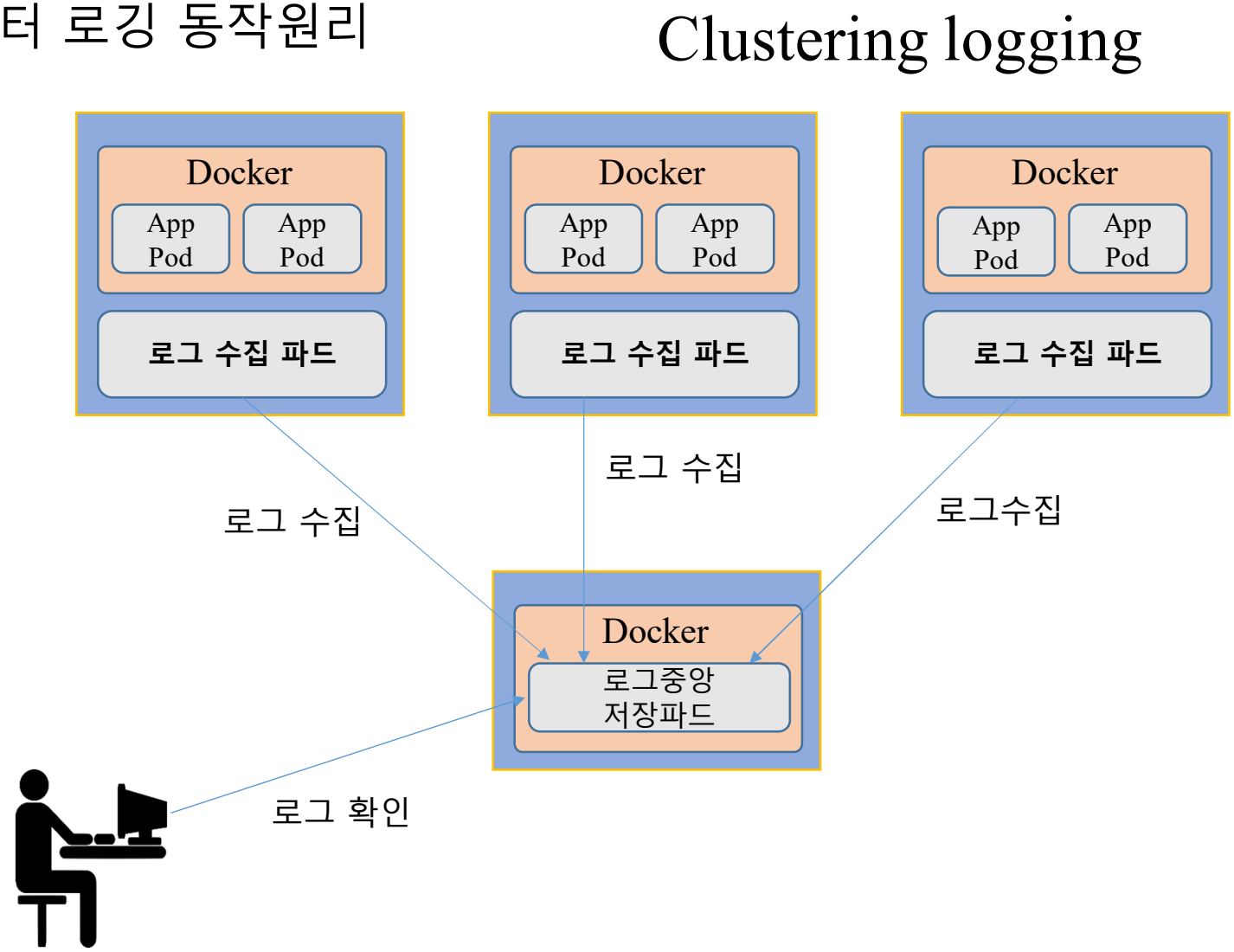
Networking Addon	가상 네트워크를 구성 시 사용 (네트워크 플러그인) CNI(container Network Interface)
DNS Addon	클러스터 안에서 동작하는 DNS 서버 Kube-DNS와 CoreDNS
Dashboard Addon	웹UI로 K8s를 관리 시 사용
Container resource monitoring	클러스터 안에서 실행중인 컨테이너의 상태를 모니터링 CPU와 메모리 사용량 정보 Kubelet 안에 포함된 cAdvisor라는 컨테이너 모니터링 도구 사용
Clustering logging	클러스터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

* cAdvisor 동작원리

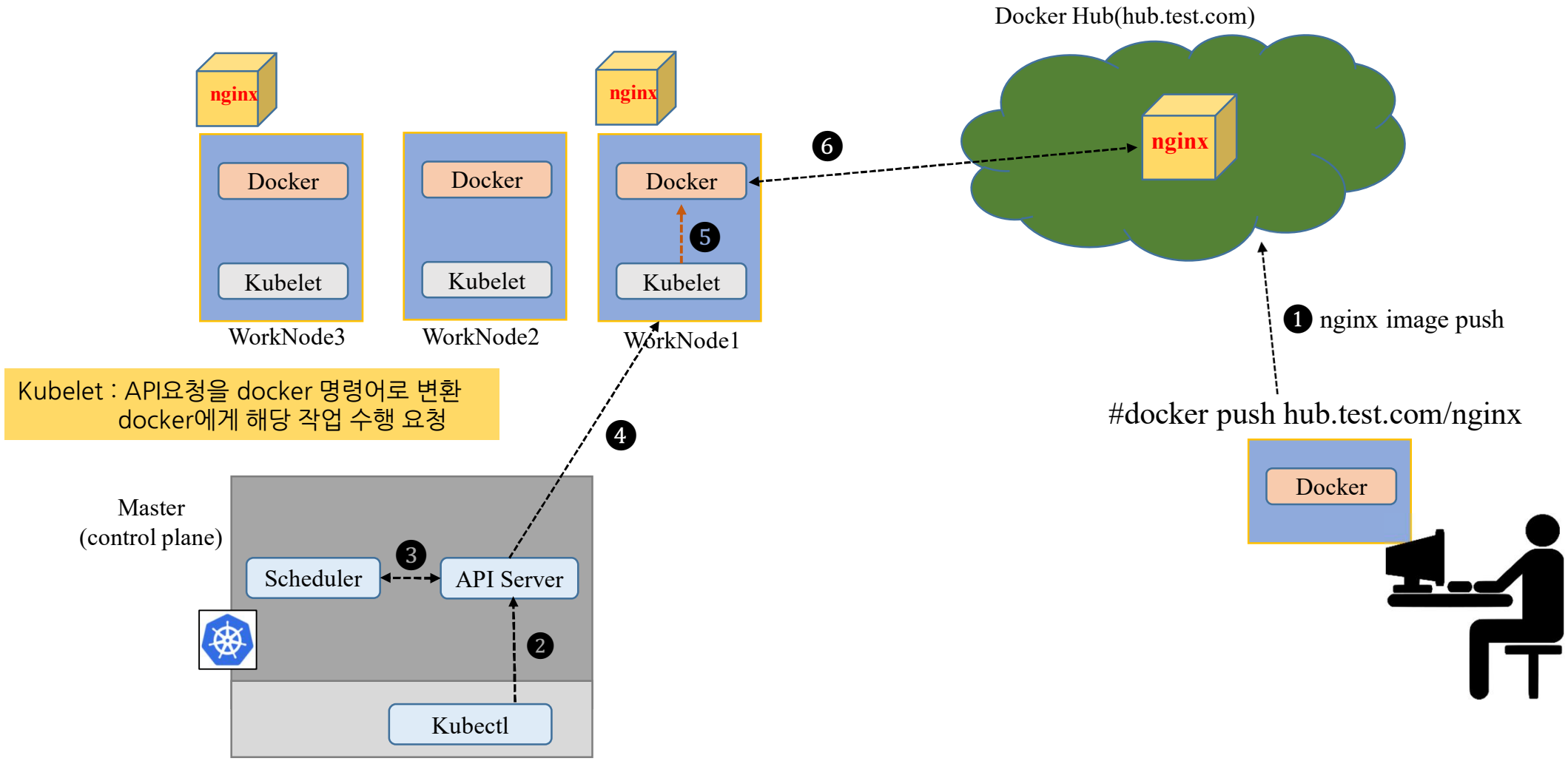
Container resource monitoring



* 클러스터 로깅 동작원리


Clustering logging
Docker
App Pod
App Pod
로그 수집 파드
Docker
App Pod
App Pod
로그 수집 파드
Docker
App Pod
App Pod
로그 수집 파드
로그 수집
로그 수집
로그수집
Docker
로그중앙 저장파드
로그 확인

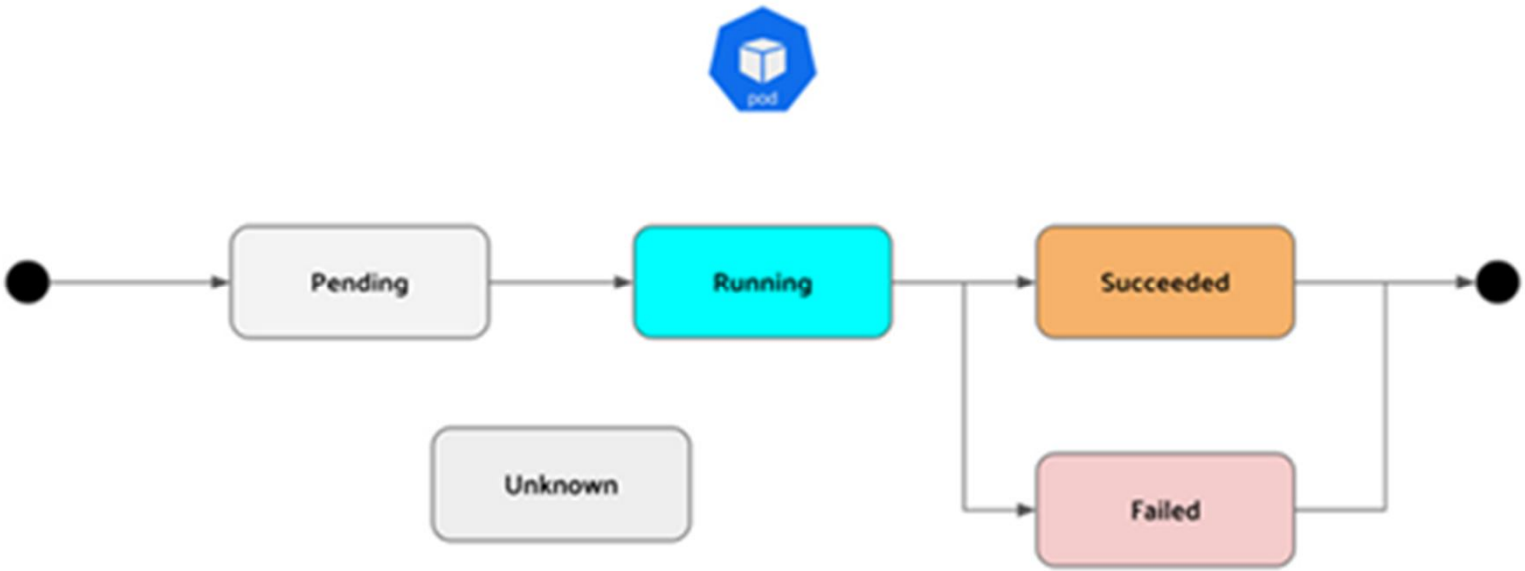
(예) 2대의 웹서버가 필요한 경우



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

K8S Lifecycle/State/Condition

Pod 생명주기(lifecycle)



- kubectl logs <pod-name>
- kubectl describe pod <pod-name>

	설 명
Pending	• K8S 시스템에 pod를 생성하는 중임을 뜻함 • Container 이미지를 다운로드한 후 전체 container를 실행하는 도중 • Pod안에 전체 container가 실행 될 때까지 시간이 걸림
Running	• Pod 안에 모든 container가 실행 중인 상태 • 1개 이상의 container가 실행 중이거나 시작 또는 재시작 상태 임
Succeeded	• Pod 안 모든 container가 정상 종료 상태(exit 0)
Failed	• Pod 안 모든 container가 정상적으로 실행 종료되지 않은 container가 있는 상태 • Container 종료코드가 0이 아니면 비정상종료 또는 시스템이 직접 Container를 종료한 것
Unknown	• Pod의 상태를 알 수 없는 상태 (Pod가 있는 노드와 통신 할 수 없을 때)

- Pod 생성부터 삭제까지의 과정에서 생명주기(lifecycle)가 있음

```
[node1 ~]$ kubectl get pods -o wide
NAME       READY   STATUS    RESTARTS   AGE   IP          NODE    NOMINATED NODE   READINESS GATES
multiple   2/2     Running   0          19s   10.5.1.3    node2   <none>           <none>
single     1/1     Running   0          11m   10.5.2.2    node3   <none>           <none>
```

kubectl describe pods *nginx-pod-live*

```
[node1 ~]$ kubectl describe pods nginx-pod-live
Name:         nginx-pod-live
Namespace:    default
Priority:     0
Node:         node3/192.168.0.11
Start Time:   Sun, 01 May 2022 04:19:01 +0000
Labels:       <none>
Annotations:  <none>
Status:       Running
IP:           10.5.2.3
IPs:
  IP:  10.5.2.3
Containers:
```

Pod 상태(state)

상태	의미
CrashLoopBackOff	컨테이너가 반복적으로 죽고 재시작
ImagePullBackOff	이미지 다운로드 실패
ErrImagePull	이미지 이름/레지스트리 오류
ContainerCreating	볼륨 마운트, 네트워크 설정 중
Terminating	Pod 삭제 중
Init:Error	Init Container 실패

Pod Conditions

- Pod의 현재 상태 정보를 나타냄 (Type과 Status로 구분)

```
Conditions:
  Type              Status
  Initialized       False
  Ready             False
  ContainersReady   False
  PodScheduled      True
Volumes:
  default-token-hg45s:
    Type:        Secret (a volume populated by a Secret)
    SecretName:  default-token-hg45s
    Optional:    false
QoS Class:       BestEffort
```

Initialized	모든 초기화 컨테이너가 성공적으로 시작 완료
Ready	Pod는 요청을 실행 할 수 있음 연결된 모든 서비스의 로드밸런싱 Pool에 추가되어야 한다는 뜻
ContainersReady	Pod 안 모든 컨테이너가 준비상태
PodScheduled	Pod가 하나의 노드로 스케쥴을 완료 했음
UnSchedulable	스케쥴러가 자원의 부족이나 다른 제약 등으로 지금 당장 Pod를 스케쥴 할 수 없음