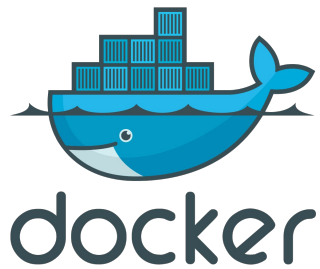
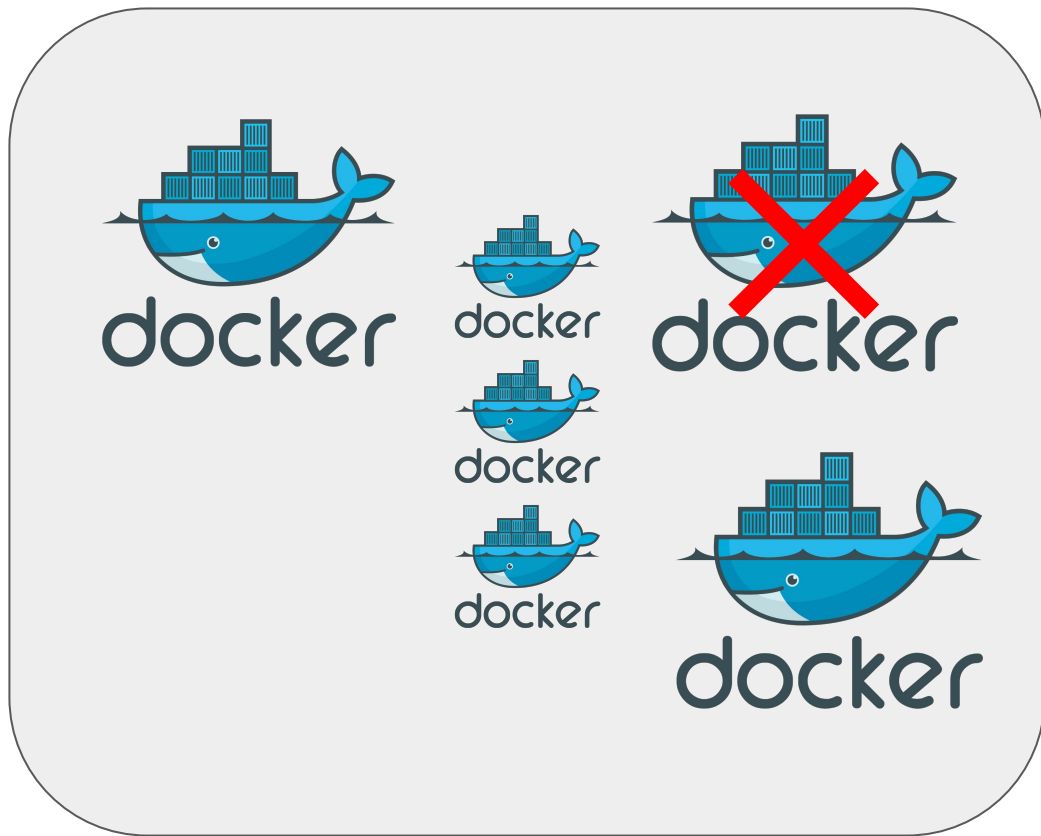
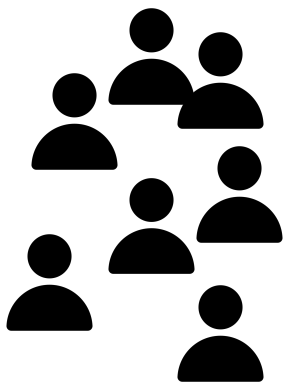


클라우드컴퓨팅

Kubernetes



kubernetes

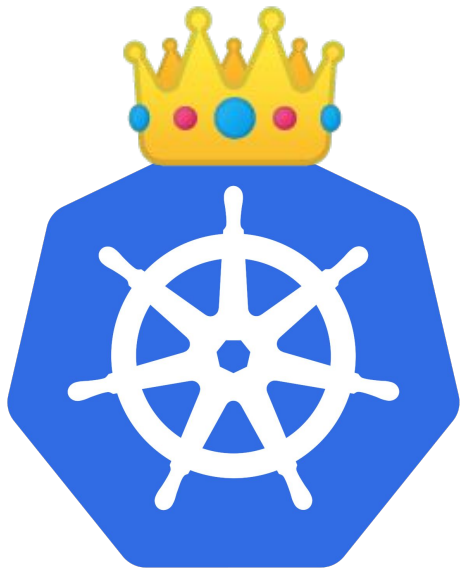




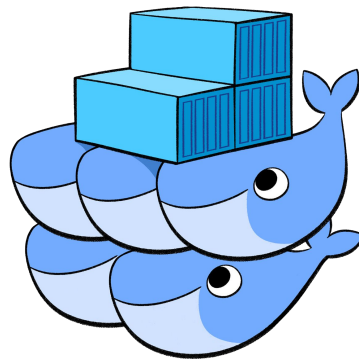
kubernetes

**Container
Orchestration**

Why Kubernetes?



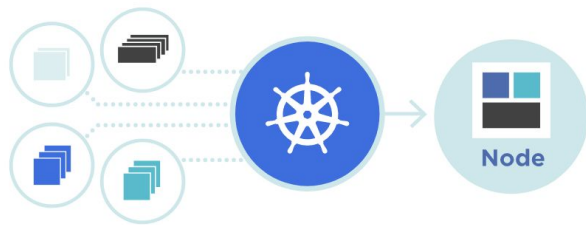
De facto!





K8s라고도 알려진 **쿠버네티스**는 컨테이너화된 애플리케이션을 자동으로 배포, 스케일링 및 관리해주는 오픈소스 시스템입니다.

애플리케이션을 구성하는 컨테이너들의 쉬운 관리 및 발견을 위해서 컨테이너들을 논리적인 단위로 그룹화합니다. 쿠버네티스는 [Google에서 15년간 프로덕션 워크로드 운영한 경험](#)을 토대로 구축되었으며, 커뮤니티에서 제공한 최상의 아이디어와 방법들이 결합되어 있습니다.

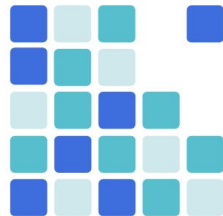


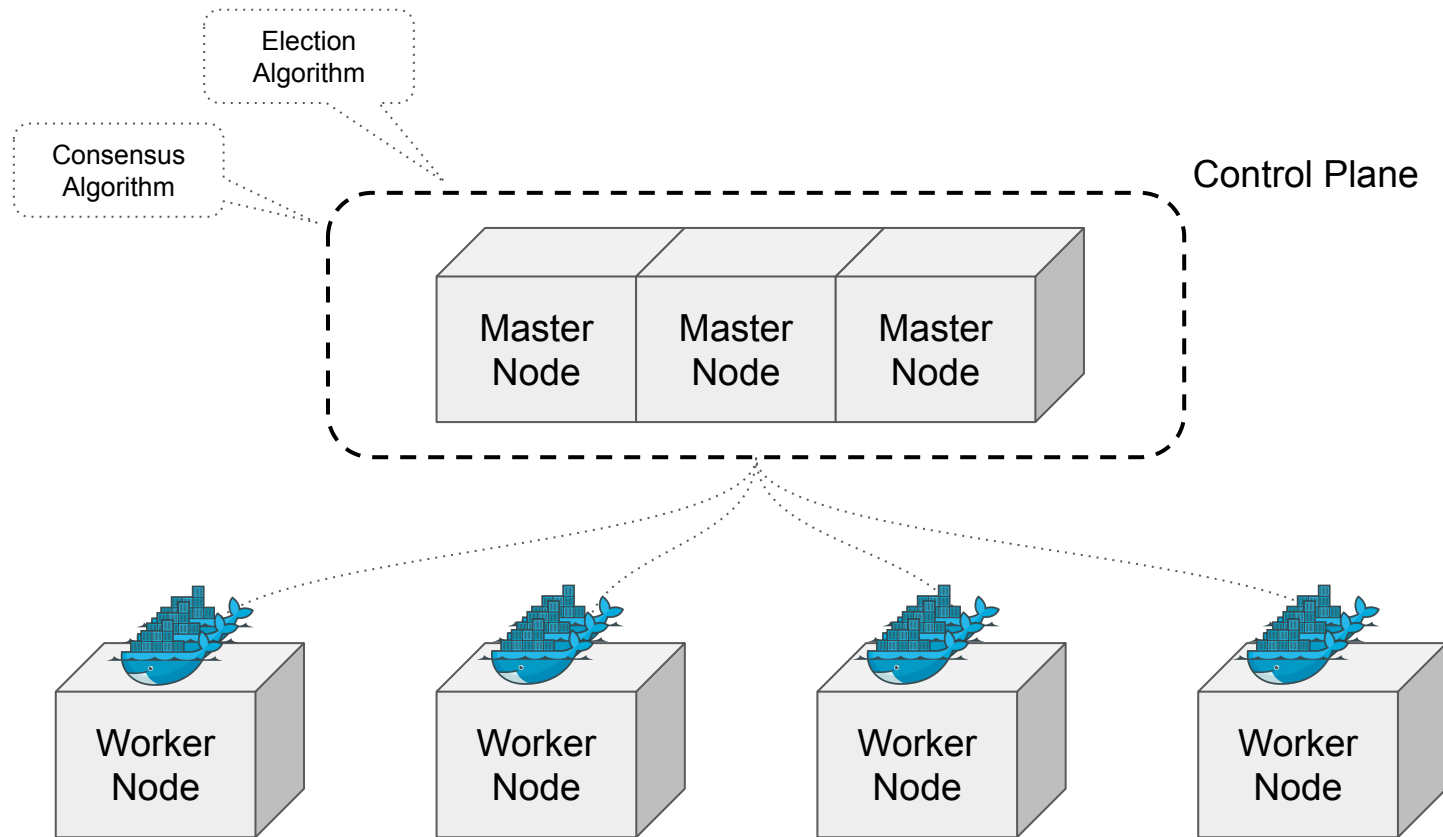
행성 규모 확장성

Google이 일주일에도 수십억 개의 컨테이너들을 운영하게 해준 원칙들에 따라 디자인되었기 때문에, 쿠버네티스는 운영팀의 규모를 늘리지 않고도 확장될 수 있습니다.

무한한 유연성

지역적인 테스트든지 글로벌 기업 운영이든지 상관없이, 쿠버네티스의 유연성은 사용자의 복잡한 니즈를 모두 수용하기 때문에 사용자의 애플리케이션들을 끊임없고 쉽게 전달할 수 있습니다.





쿠버네티스 사용해보기

The screenshot shows the 'Kubernetes Playground Terminal' interface on the Katacoda platform. The browser address bar shows the URL: `katacoda.com/lloodse/courses/kubernetes/kubernetes-02-playground`. The page header includes the O'Reilly logo, 'KATACODA OVERVIEW & SOLUTIONS', and a 'YOUR PROFILE' link with a 'LOG OUT' button.

The main content area is titled 'Kubernetes Playground Terminal' and indicates 'Step 1 of 2'. It provides instructions for starting an application:

- Some helpful quick wins!
- Start an instant Application
- Start using Kubernetes with `kubectl`.
- Launch containers with `kubectl run my-nginx --image=nginx`.
- Expose the deployment inside of the cluster
`kubectl expose deployment my-nginx --port 80`.
- Now you have one type `deployment` (created by `kubectl run ...`) and one type `service` (created by `kubectl expose ...`). This maps in Kubernetes to the following components:
`kubectl get all`
- Access a service by `localhost`
- Proxy the traffic into your local host by using the `port-forward` command:
`kubectl port-forward svc/my-nginx 80:80`
- Connect to the Nginx web server locally through `curl localhost:80`
- Stop the proxying background process: `kill %1`
- How to write the right spec?

At the bottom, there are two links:

- Take a look at the [Kubernetes reference docs](#)
- Use `kubectl explain pod.spec.containers`

On the right side, there are two terminal windows. 'Terminal Host1' shows the setup of the 'kubernetes-lab' environment, including cloning the repository, setting up the controlplane, and starting the kubelet. 'Terminal Host2' shows the setup of the 'node01' environment, including cloning the repository and setting up the node.

Katacoda


<https://www.katacoda.com/login>

Welcome! Connect to start

 Sign Up with Github

 Sign Up with LinkedIn

 Sign Up with Twitter

 Sign Up with Google

...or sign up using your email

Email Address

Password

SIGN UP FOR FREE

Already have an account? [Log in here](#)

Katacoda

<https://www.katacoda.com/courses/kubernetes/playground>

누르고 조금
기다려야함!



Launch Cluster

```
launch.sh ↵
```

This will create a two node Kubernetes cluster.

Health Check



```
kubectl cluster-info ↵
```

Interested in writing your own Kubernetes scenarios and demos? Visit www.katacoda.com/teach

CONTINUE

Terminal Host 1 +

Your Interactive Bash Terminal.

```
controlplane $
```

Terminal Host 2

Your Interactive Bash Terminal.

```
node01 $
```

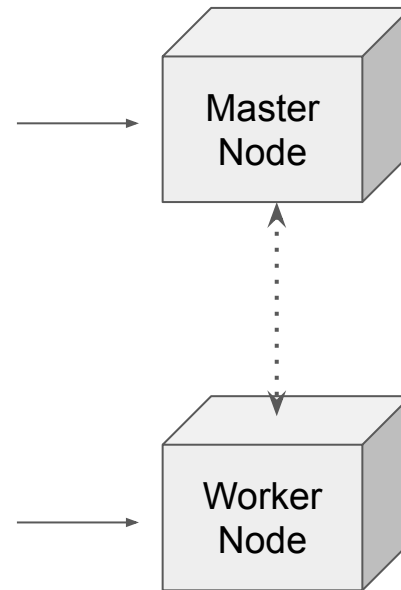
Katacoda

```
Terminal Host 1 + [X] [G]
Your Interactive Bash Terminal.

controlplane $

Terminal Host 2
Your Interactive Bash Terminal.

node01 $
```



Katacoda

```
Terminal Host 1 +
controlplane $ kubectl get nodes
NAME           STATUS    ROLES    AGE   VERSION
controlplane   Ready    master   5m22s v1.18.0
node01         Ready    <none>   4m54s v1.18.0
controlplane $
```

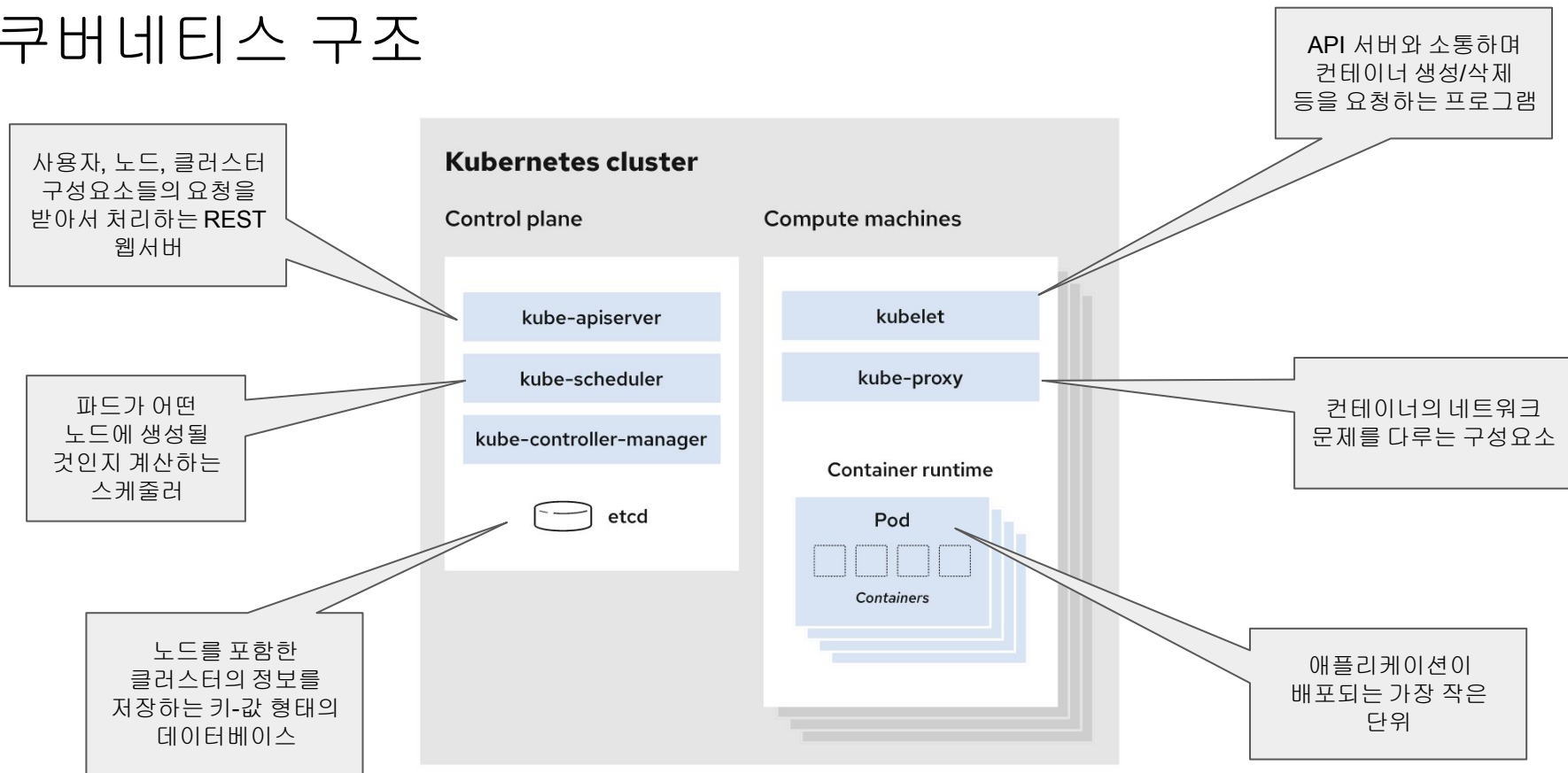
docker: 도커 엔진과 소통하기 위한 명령어 (e.g., *docker ps*)

kubectl: 쿠버네티스 클러스터와 소통하기 위한 명령어

```
Terminal Host 1 +
controlplane $ kubectl get nodes
NAME           STATUS    ROLES    AGE   VERSION
controlplane   Ready     master   5m22s v1.18.0
node01         Ready     <none>    4m54s v1.18.0
controlplane $
```

kubectl get {resource}	자원을 나열
kubectl describe {resource}	자원에 대한 상세한 정보 보여주기
kubectl create {resource}	자원 생성 (ex. namespace, service, etc)
kubectl delete {resource}	자원 삭제

쿠버네티스 구조



쿠버네티스 구조

1. `kubectl` : 새로운 Pod(Container) 생성 요청
2. `apiserver` : 요청에 대한 정보를 `etcd`에 저장
3. `kube-controller-manager` : 요청은 들어왔지만 아직 생성되지 않은 Pod 발견
4. `kube-scheduler` : 어떤 노드에 생성해야 하는지 판단
5. `kubelet` : Pod(Container) 생성 요청 받음
6. `kubelet` : 도커에게 명령 보냄

Katacoda

nginx 이미지를 사용하여 'first-pod'
라는 이름을 가진 파드를 실행한다.

```
Terminal Host 1 +
controlplane $ kubectl run first-pod --image=nginx
pod/first-pod created
controlplane $ kubectl get pods
NAME          READY   STATUS             RESTARTS   AGE
first-pod     0/1     ContainerCreating   0           6s
controlplane $ kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
first-pod     1/1     Running   0           21s
```

Katacoda

```
controlplane $ kubectl describe pods first-pod
```

```
...
```

```
...
```

```
Events:
```

Type	Reason	Age	From	Message
----	-----	----	----	-----
Normal	Scheduled	11m	default-scheduler	Successfully assigned default/first-pod to node01
Normal	Pulling	11m	kubelet, node01	Pulling image "nginx"
Normal	Pulled	11m	kubelet, node01	Successfully pulled image "nginx"
Normal	Created	11m	kubelet, node01	Created container first-pod
Normal	Started	11m	kubelet, node01	Started container first-pod

파드가 생성될 위치는
node01 노드로 결정

node01의 kubelet이
docker run nginx

```
controlplane $ kubectl delete pods first-pod
pod "first-pod" deleted
```

```
controlplane $ kubectl get pods
```

```
No resources found in default namespace.
```

👉 **'kubectl'**은 마스터
노드에서만!

Katacoda

```
controlplane $ kubectf create deployment nginx --image=nginx  
deployment.apps/nginx created
```

```
controlplane $ kubectf get deployments.apps
```

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
nginx	1/1	1	1	15s

```
controlplane $ kubectf get pods
```

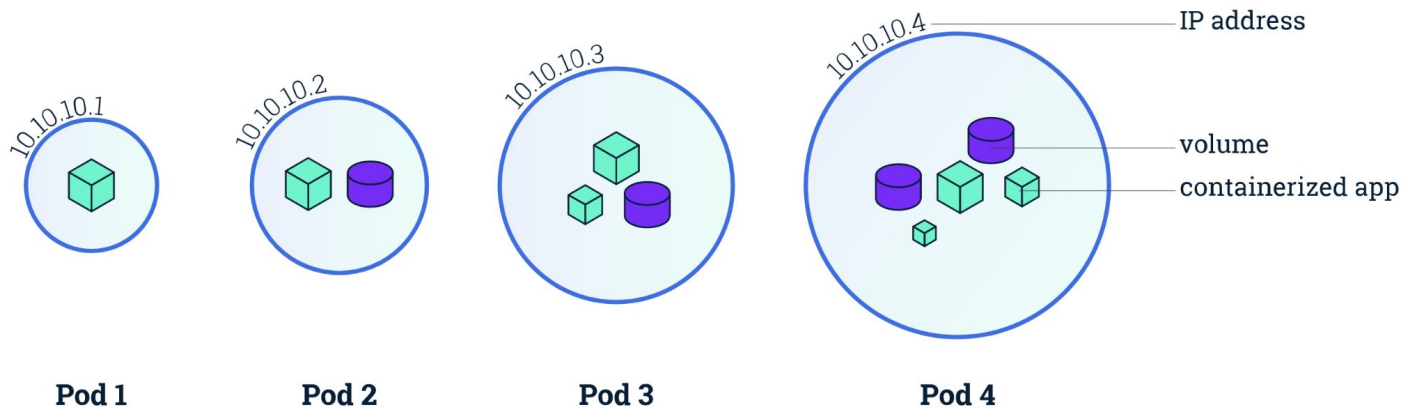
NAME	READY	STATUS	RESTARTS	AGE
nginx-f89759699-cxf47	1/1	Running	0	20s

Pod

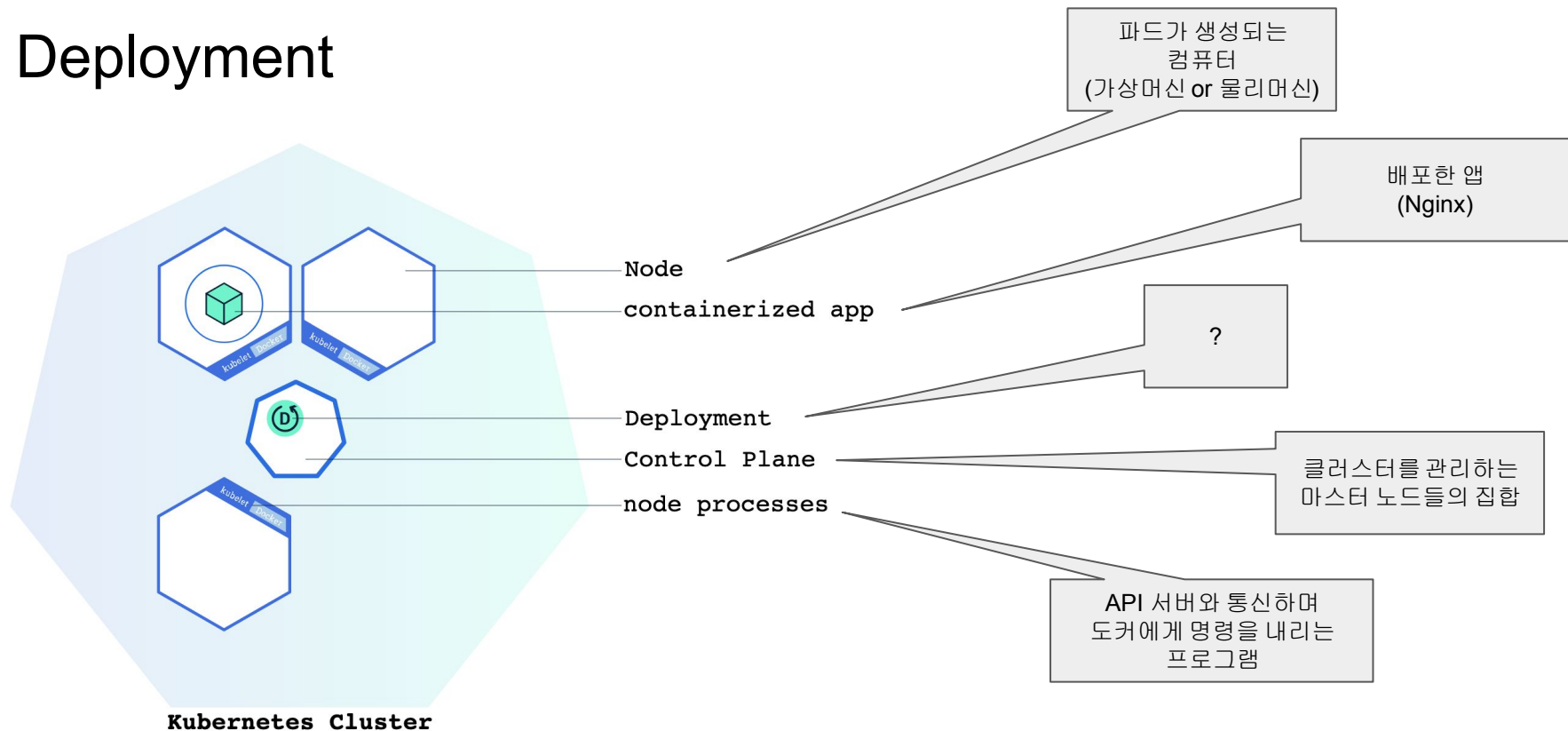
“쿠버네티스에서 생성하고 관리할 수 있는 배포 가능한 가장 작은 컴퓨팅 단위”

보통 1개의 파드에는 1개의 컨테이너

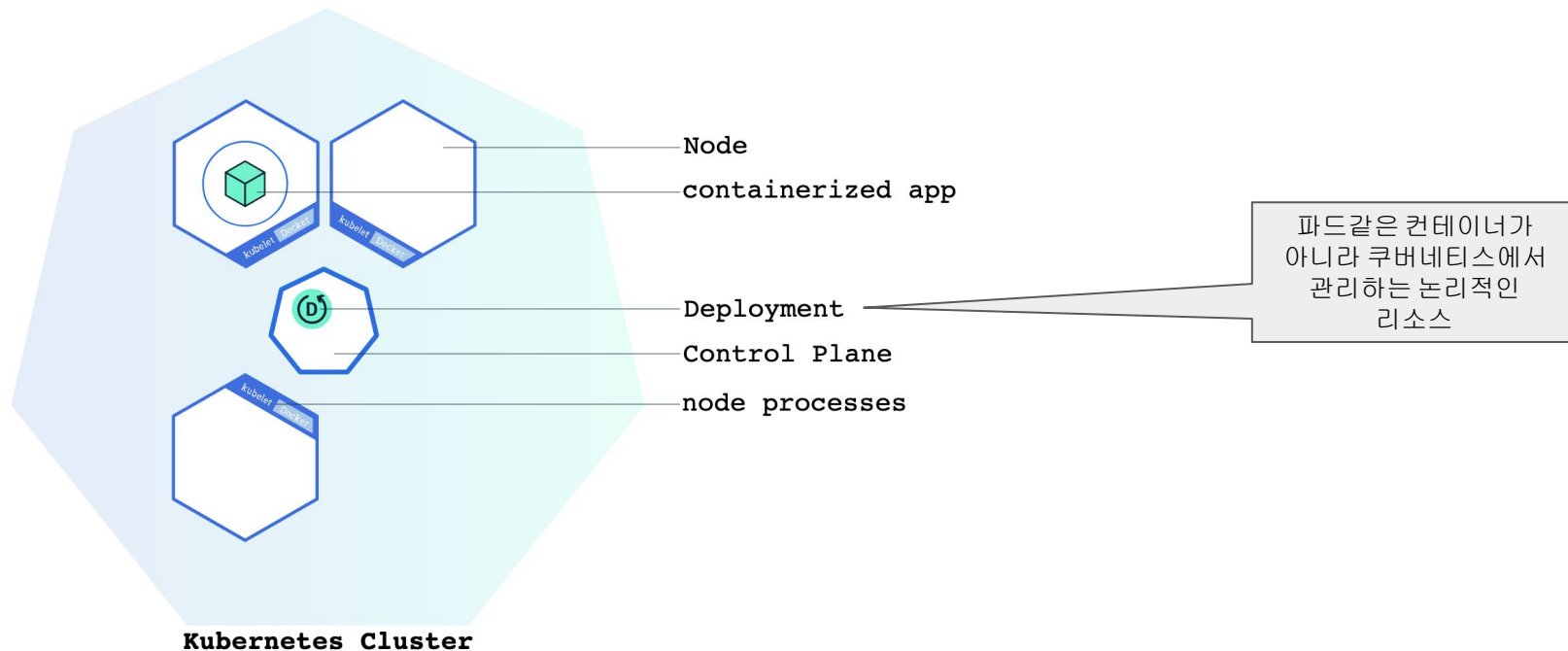
파드 안에 있는 여러 컨테이너는 스토리지와 네트워크를 공유할 수 있다



Deployment



Deployment



Deployment

“kubectl run”으로 파드를 실행시킬 수 있는데
굳이?

똑같은 파드를 100개 배포할 실행할 때는?
만약 노드1에 있는 파드에 장애가 일어나면?

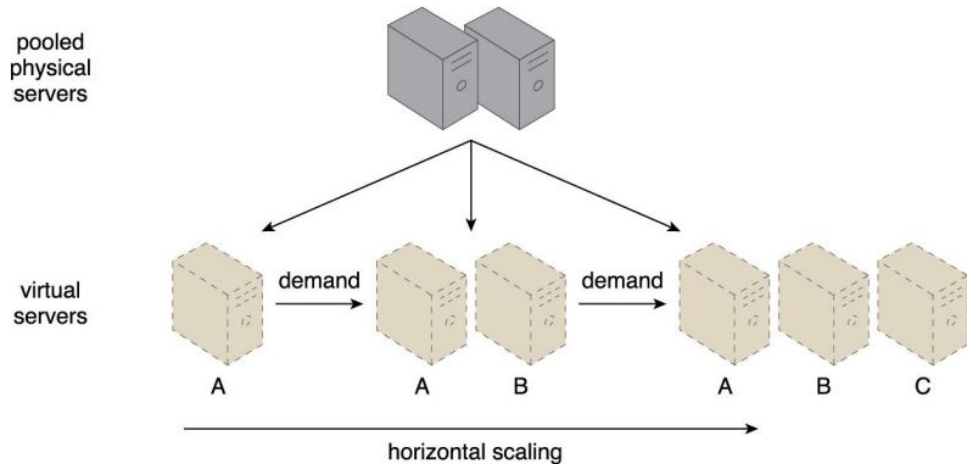


Figure 3.4 An IT resource (Virtual Server A) is scaled out by adding more of the same IT resources (Virtual Servers B and C)

Deployment

curl 10.244.1.5

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
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>

<p>For online documentation and support please refer to
<a href="http://nginx.org">nginx.org</a>. <br>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

```
controlplane $ kubectl scale deployment nginx --replicas=3
deployment.apps/nginx scaled
```

```
controlplane $ kubectl get deployments.apps nginx
```

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
nginx	3/3	3	3	2m15s

```
controlplane $ kubectl get pods -o wide
```

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE	NOMINATED NODE	READINESS GATES
nginx-f89759699-6hgvm	1/1	Running	0	22s	10.244.1.5	node01	<none>	<none>
nginx-f89759699-6jtmk	1/1	Running	0	2m26s	10.244.1.3	node01	<none>	<none>
nginx-f89759699-stj9k	1/1	Running	0	22s	10.244.1.4	node01	<none>	<none>

워커 노드가 1개이기
때문에 node01에만
생성

Deployment

```
controlplane $ kubectl delete pods nginx-f89759699-6hgvm
pod "nginx-f89759699-6hgvm" deleted
```

```
controlplane $ kubectl get pods -o wide
```

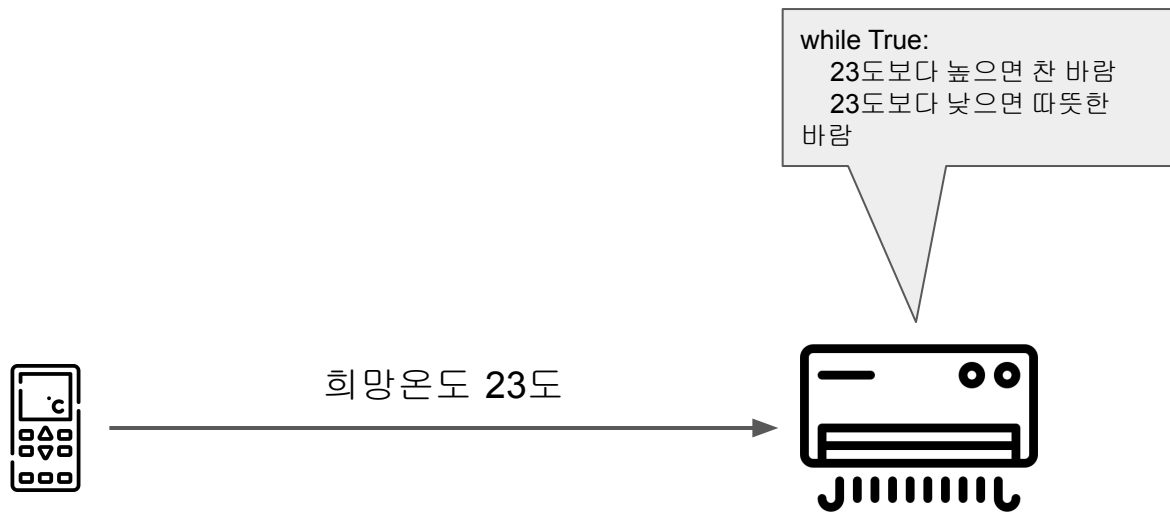
NAME	READY	STATUS	RESTARTS	AGE	IP	NODE	NOMINATED NODE	READINESS GATES
nginx-f89759699-6jtmk	1/1	Running	0	16m	10.244.1.3	node01	<none>	<none>
nginx-f89759699-9h85r	1/1	Running	0	5m	10.244.1.6	node01	<none>	<none>
nginx-f89759699-stj9k	1/1	Running	0	14m	10.244.1.4	node01	<none>	<none>

새로운 파드 생성



어떻게 파드에 문제가 생긴 것을 알 수 있을까?
어떻게 정확히 3개를 맞춰서 실행할까?

Controller



Controller

```
controlplane $ kubectl get deployments.apps
```

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
nginx	0/1	1	0	21s

```
controlplane $ kubectl describe deployments.apps nginx
```

```
Name:          nginx
```

```
...
```

```
Replicas:      3 desired | 3 updated | 3 total | 3 available | 0 unavailable
StrategyType:  RollingUpdate
```

```
...
```

```
Events:
```

Type	Reason	Age	From	Message
Normal	ScalingReplicaSet	98s	deployment-controller	Scaled up replica set nginx-f89759699 to 1
Normal	ScalingReplicaSet	18s	deployment-controller	Scaled up replica set nginx-f89759699 to 3

Controller

```
controlplane $ kubectf get replicaset.apps nginx-f89759699
...
Controlled By:      Deployment/nginx
Replicas:           3 current / 3 desired
Pods Status:        3 Running / 0 Waiting / 0 Succeeded / 0 Failed
...
```

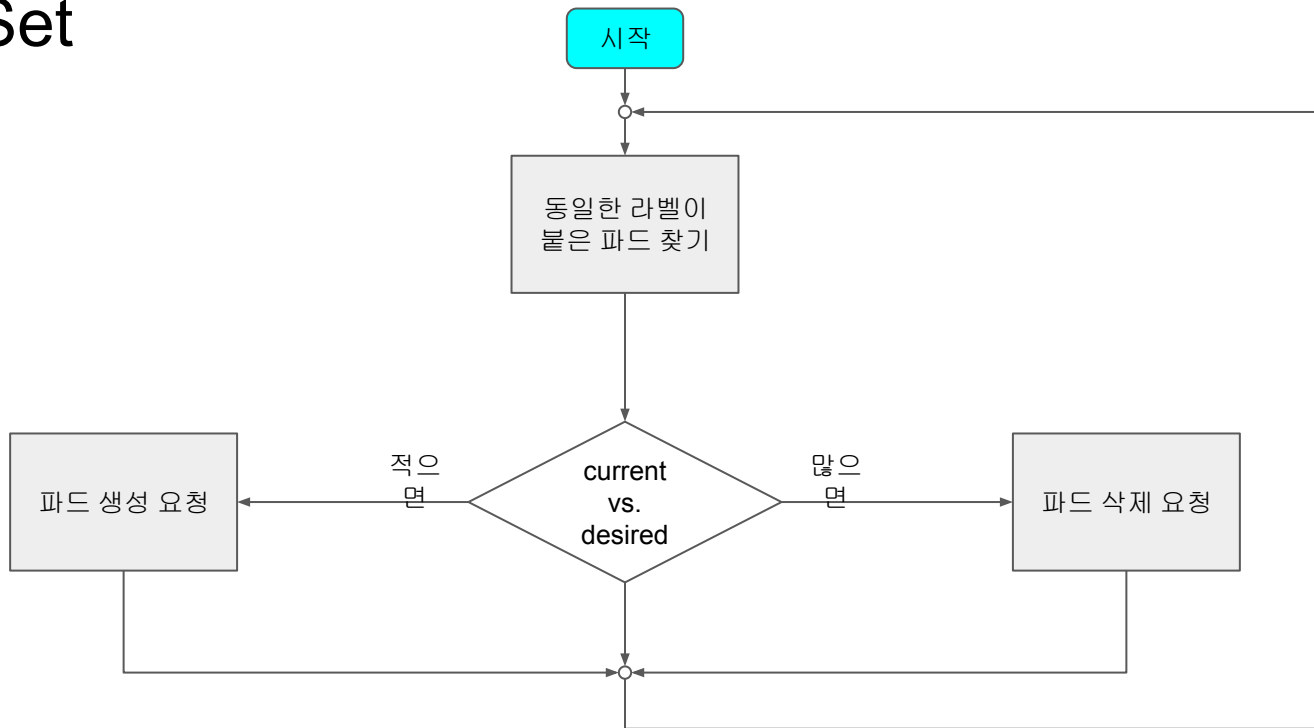
ReplicaSet

Replication Controller의 발전된 형태

일종의 컨트롤러

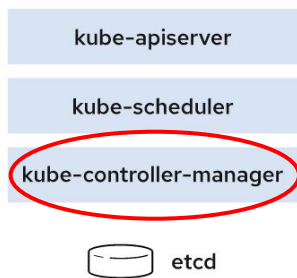
연결되어 있는 Deployment의 Pod를 책임진다

ReplicaSet

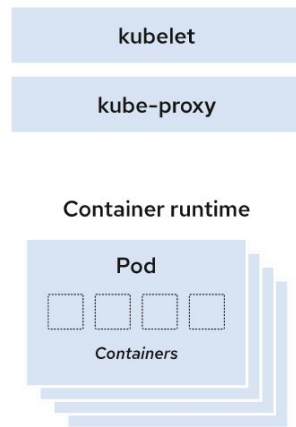


Kubernetes cluster

Control plane



Compute machines



All controllers: attachdetach, bootstrapsigner, cloud-node-lifecycle, clusterrole-aggregation, cronjob, csrapproving, csrcleaner, csrsigning, daemonset, deployment, disruption, endpoint, endpointslice, endpointslicemirroring, ephemeral-volume, garbagecollector, horizontalpodautoscaling, job, namespace, nodeipam, nodelifecycle, persistentvolume-binder, persistentvolume-expander, podgc, pv-protection, pvc-protection, replicaset, replicationcontroller, resourcequota, root-ca-cert-publisher, route, service, serviceaccount, serviceaccount-token, statefulset, tokencleaner, ttl, ttl-after-finished

+ 커스텀 컨트롤러 추가 가능

오토스케일링이 가능한 앱을 배포하고 싶다면

Deployment

ReplicaSet

Pod

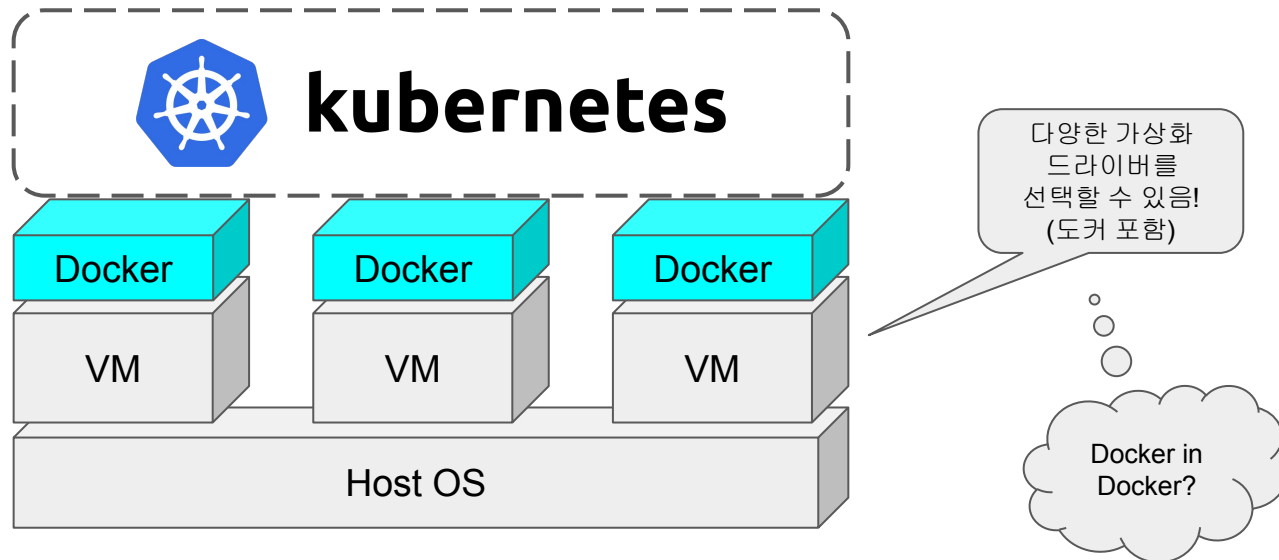
Container

Docker

Minikube

로컬 환경에 쿠버네티스를 제공하는 프로그램 (개발용 & 교육용)

Minikube



<https://minikube.sigs.k8s.io/docs/start/>

```
minikube start --nodes=2
```

😊 Darwin 11.6 의 minikube v1.23.0

👍 minikube 클러스터의 minikube 컨트롤 플레인 노드를 시작하는 중

```
🔥 Creating docker container (CPUs=2, Memory=1986MB) ...
```

■ 인증서 및 키를 생성하는 중 ...

■ RBAC 규칙을 구성하는 중 ...

Kubernetes 구성 요소를 확인 ...

👉 애드온 활성화 : `storage-provisioner, default-storageclass`

베이스 이미지를 다운받는 중 ...

네트워크 옵션을 찾았습니다

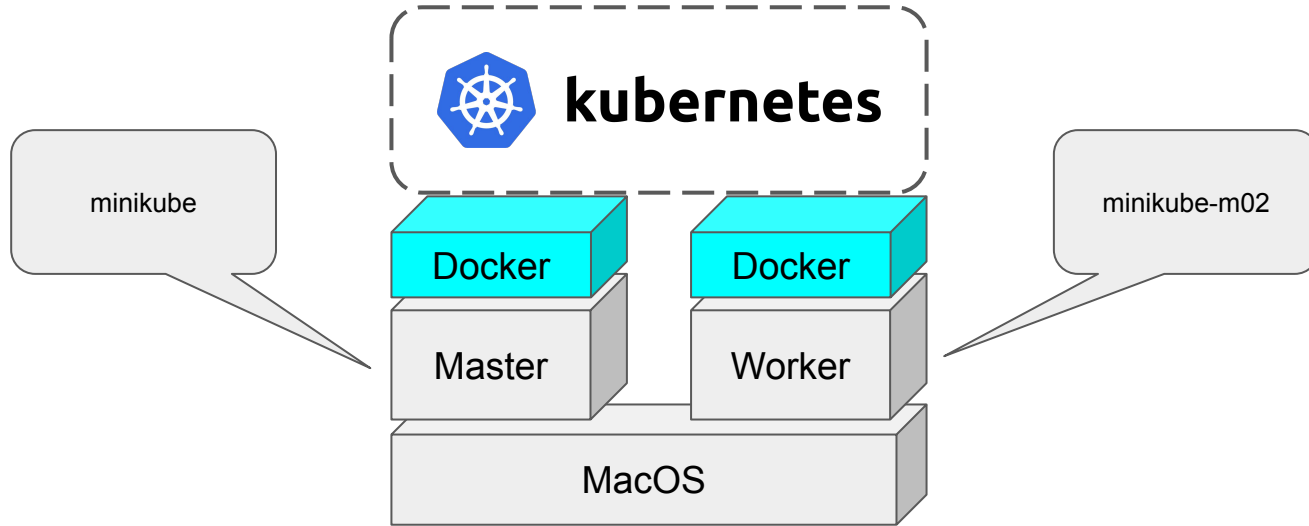
쿠버네티스 v1.22.1 을 Docker 20.10.8 런타임으로 설치하는 중

Kubernetes 구성 요소를 확인

끝났습니다! kubectl이 "minikube" 클러스터와 "default" 네임스페이스를 기본적으로 사용하도록 구성되었습니다.

```
sejin@sejin-MacBookPro:~  
[~] 59s * minikube  
[~] > kubectl get nodes  
NAME          STATUS    ROLES          AGE   VERSION  
minikube      Ready    control-plane,master  84s   v1.22.1  
minikube-m02  Ready    <none>         59s   v1.22.1  
[~] * minikube  
[~] > kubectl get pods -A  
NAMESPACE     NAME                                READY   STATUS    RESTARTS   AGE  
kube-system   coredns-78fcd69978-49zzf          1/1     Running   0           75s  
kube-system   etcd-minikube                     1/1     Running   0           87s  
kube-system   kindnet-57bbn                     1/1     Running   0           66s  
kube-system   kindnet-mrh2v                     1/1     Running   0           76s  
kube-system   kube-apiserver-minikube           1/1     Running   0           87s  
kube-system   kube-controller-manager-minikube  1/1     Running   0           90s  
kube-system   kube-proxy-2ghqb                  1/1     Running   0           76s  
kube-system   kube-proxy-62b27                  1/1     Running   0           66s  
kube-system   kube-scheduler-minikube           1/1     Running   0           87s  
kube-system   storage-provisioner               1/1     Running   0           86s  
[~]
```


Minikube



로컬 환경에서 쿠버네티스 클러스터 준비
끝!

minikube ssh

minikube ssh --node={노드 이름}

minikube status

minikube delete

쿠버네티스 다루는 방법

1. HTTP 요청
2. kubectl CLI
3. YAML 템플릿
4. 쿠버네티스 클라이언트 라이브러리

쿠버네티스 다루는 방법

1. HTTP 요청
2. kubectl CLI
3. YAML 템플릿
4. 쿠버네티스 클라이언트 라이브러리

YAML

```
! scaffold.yaml ●
! scaffold.yaml > [ ] profiles > name
1  apiVersion: scaffold/v2beta4
2  kind: Config
3  build:
4    tagPolicy:
5      sha256: {}
6    artifacts:
7      - context: .
8        imag
9  deploy:
10    kubect
11    mani
12    - ku
13  profiles
14  - name: 1234
15    build:
16      googleCloudBuild: {}
17
```

JSON 형식과 비슷한 파일 포맷 (Dockerfile)

YAML 파일로 쿠버네티스 다루기

실제로
쿠버네티스에
적용시키지는
않는다

```
kubect1 run nginx --image=nginx --dry-run=client -o yaml > nginx-pod.yaml
```

이 템플릿을 **yaml**
형식으로 저장

YAML 파일로 쿠버네티스 다루기

오브젝트를
유일하게
구분짓는
데이터

```
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: nginx
  name: nginx
spec:
  containers:
  - image: nginx
    name: nginx
    resources: {}
  dnsPolicy: ClusterFirst
  restartPolicy: Always
status: {}
```

오브젝트의
의도한 상태

YAML 파일로 쿠버네티스 다루기

1. nginx-deployment 라는 이름의 Deployment 템플릿을 만들고
확인하기
2. 만든 템플릿을 적용하기

YAML 파일로 쿠버네티스 다루기

★ `kubectl apply -f {template-name}.yaml` ★

Deployment로 만들어진 Pod들은 클러스터 내에서만 접근 가능
Pod의 IP로 직접 요청을 보내야 함



외부에서 내 서버로 어떻게 접근하지?

서비스

```
kubect1 expose deployment nginx --port 80 --dry=run=client -o yaml > service.yaml
```

서비스

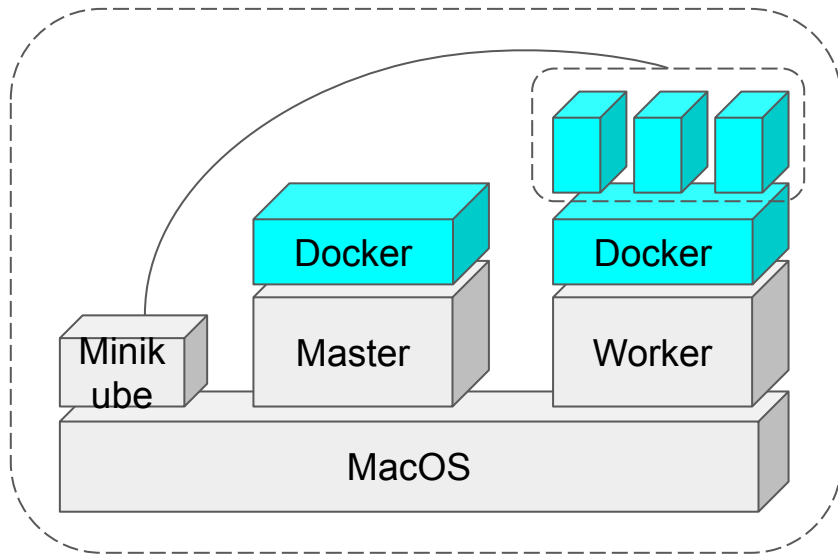
```
apiVersion: v1
kind: Service
metadata:
  creationTimestamp: null
  labels:
    app: nginx
  name: nginx
spec:
  ports:
    - port: 80
      protocol: TCP
      targetPort: 80
  selector:
    app: nginx
status:
  loadBalancer: {}
```

서비스

```
minikube service nginx
```

쿠버네티스 자체 기능은 아니지만, **Minikube**가 서비스에 접근하기 쉽게 만들어줌

서비스



내 앱을 외부에서 사용하려면

Ingress

IngressController

Service

Deployment

ReplicaSet

Pod

Container

Docker

서비스를 노출시키는 방식

ClusterIP(default): 클러스터 내부 IP에 노출

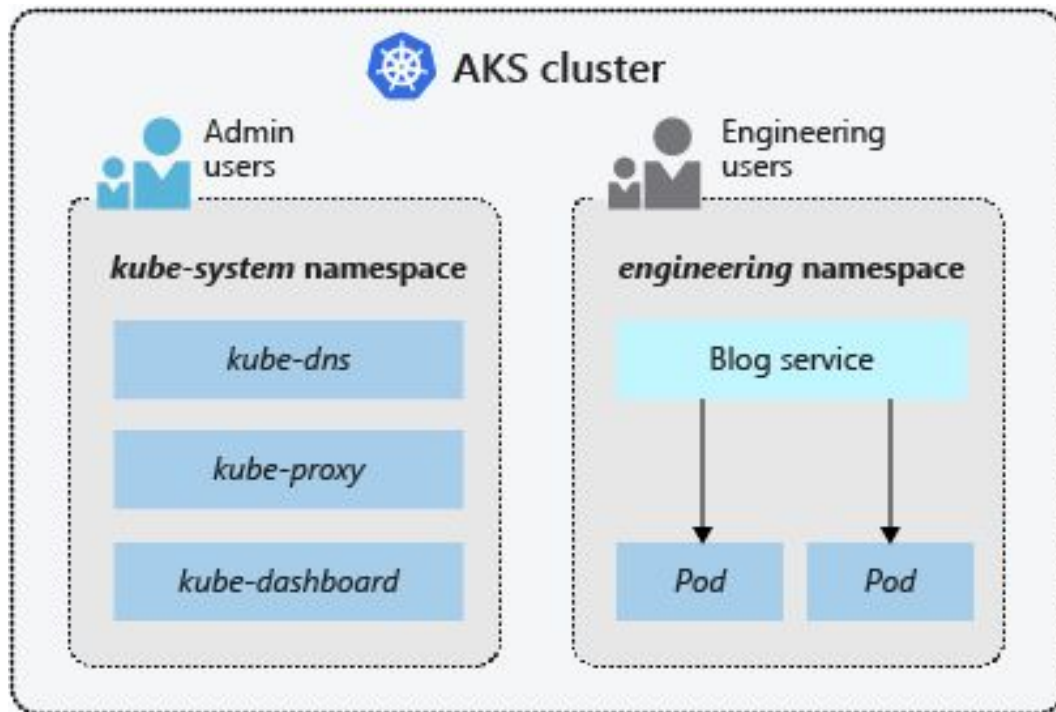
NodePort: 포트 포워딩을 사용

LoadBalancer: 외부 로드 밸런서 사용

네임스페이스

```
kubectl get pod -A
```

네임스페이스



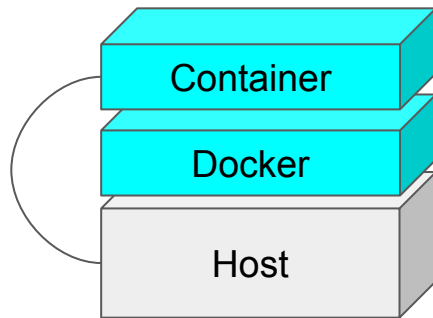
네임스페이스

```
kubectl create namespace {name}  
kubectl run test --image=nginx
```

Volume

도커에서의 볼륨

```
docker run ~~~ -v {src}:{dst} ~~~
```



Volume

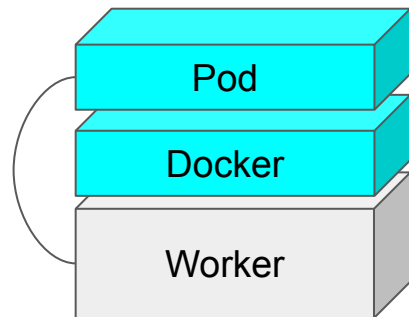
```
apiVersion: v1
kind: Pod
metadata:
  name: test-pod
spec:
  containers:
  - image: nginx
    name: test-container
    volumeMounts:
    - mountPath: /dst
      name: test-volume
  volumes:
  - name: test-volume
    hostPath:
      path: /src
```


Volume

```
kubectl apply -f test-volume.yaml  
minikube ssh --node=minikube-m02  
    sudo touch /src/hello  
        docker ps  
docker exec {docker_name} ls /dst
```

Volume

① 호스트와 마운트된 Pod
생성



② 워커 노드로 진입 (ssh)

④ /dst/hello 파일 확인

③ /src/hello 파일 생성

과제 1

요구사항을 만족하는 YAML 템플릿 만들기

목표: 아래 요구사항을 만족하는 YAML 템플릿 작성하기

- Deployment 이름: nginx-deployment
- Deployment 이미지: nginx
- Pod replication 개수: 2
- Service 이름: nginx-service
- nginx-deployment와 nginx-service 연결하기

힌트: 템플릿과 템플릿 사이에 “---” 문자를 넣으면 하나의 템플릿으로 합칠 수 있다.

과제 1

요구사항을 만족하는 YAML 템플릿 만들기

YAML 파일 제출 (파일 포맷: 학번_1.yaml)

확인 방법: ***kubect1 apply -f 2020021465_1.yaml*** 실행 후에 ***minikube service nginx-service*** 명령어로 확인

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

🍏 ~/Developer/k8s-practice

> minikube service list

NAMESPACE	NAME	TARGET PORT	URL
default	kubernetes	No node port	
default	nginx-service	No node port	
kube-system	kube-dns	No node port	

🍏 ~/Developer/k8s-practice

> minikube service nginx-service

NAMESPACE	NAME	TARGET PORT	URL
default	nginx-service		No node port

🐞 service default/nginx-service has no node port

🚧 nginx-service 서비스의 터널을 시작하는 중

NAMESPACE	NAME	TARGET PORT	URL
default	nginx-service		http://127.0.0.1:55348

🔥 Opening service default/nginx-service in default browser...

! Because you are using a Docker driver on darwin, the terminal needs to be open to run it.

과제 2

요구사항을 만족하는 YAML 템플릿 만들기

목표: 아래 요구사항을 만족하는 YAML 템플릿 작성하기

- Pod 이름: printenv
- image: ubuntu
- namespace: my-ns
- Pod 생성시 환경 변수를 출력한다. (printenv 명령어 사용)

힌트: Pod의 spec > containers 내용 중에는 image, name 이외에도 사용할 수 있는 것들이 있다.

<https://kubernetes.io/docs/reference/kubernetes-api/workload-resources/pod-v1/#entrypoint>

과제 2

요구사항을 만족하는 YAML 템플릿 만들기

YAML 파일 제출 (파일 포맷: 학번_2.yaml)

확인 방법: **kubect1 apply -f 2020021465_2.yaml**

실행 후에 **kubect1 logs -n my-ns printenv** 명령어로
출력내용 확인 (그림에서는 네임스페이스 빠짐)

```
~/Developer/k8s-practice .....  
> k get pods  
NAME      READY   STATUS    RESTARTS   AGE  
printenv  0/1     Completed  0           5s  
  
~/Developer/k8s-practice .....  
> k logs printenv  
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin  
HOSTNAME=printenv  
KUBERNETES_PORT_443_TCP_ADDR=10.96.0.1  
KUBERNETES_SERVICE_HOST=10.96.0.1  
KUBERNETES_SERVICE_PORT=443  
KUBERNETES_SERVICE_PORT_HTTPS=443  
KUBERNETES_PORT=tcp://10.96.0.1:443  
KUBERNETES_PORT_443_TCP=tcp://10.96.0.1:443  
KUBERNETES_PORT_443_TCP_PROTO=tcp  
KUBERNETES_PORT_443_TCP_PORT=443  
HOME=/root
```

과제 제출

제출 마감: 12월 5일 23:59PM

쿠버네티스 구조

Pod, Deployment, Controller,
ReplicaSet, Service, Namespace

Volume

YAML

<https://subicura.com/k8s/prepare/yaml.html>

조교 김세진

sejjj120@korea.ac.kr