

Kubernetes (k8s)

1 쿠버네티스란

2 쿠버네티스 구성 요소

3 간단한 실습

4 참고자료



Part 1
쿠버네티스란?

키잡이나 파일럿을 뜻하는 그리스어에서 유래

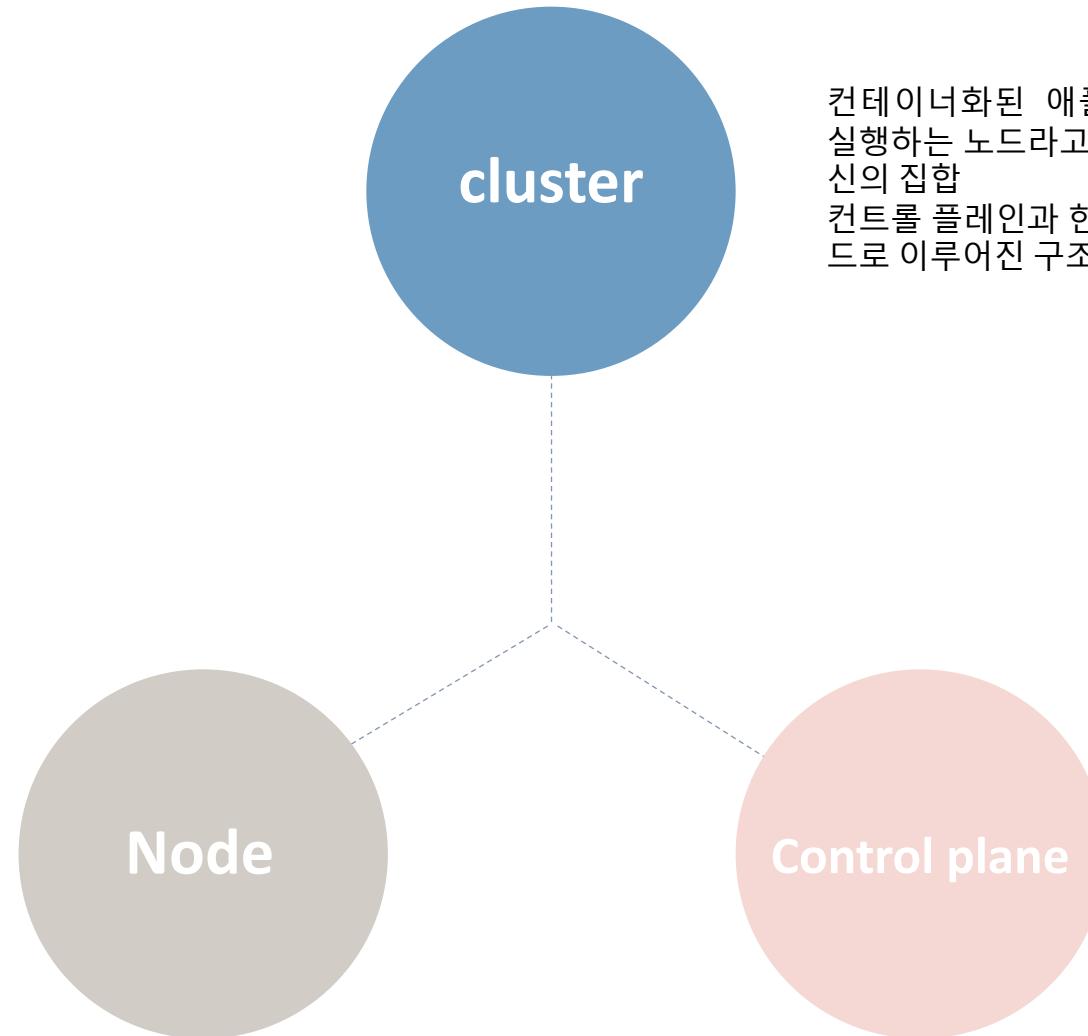
쿠버네티스란? (k8s)

컨테이너화된 워크로드와 서비스를 관리하기 위한 이식성이 있고 확장 가능한 오픈소스 플랫폼



Part 2

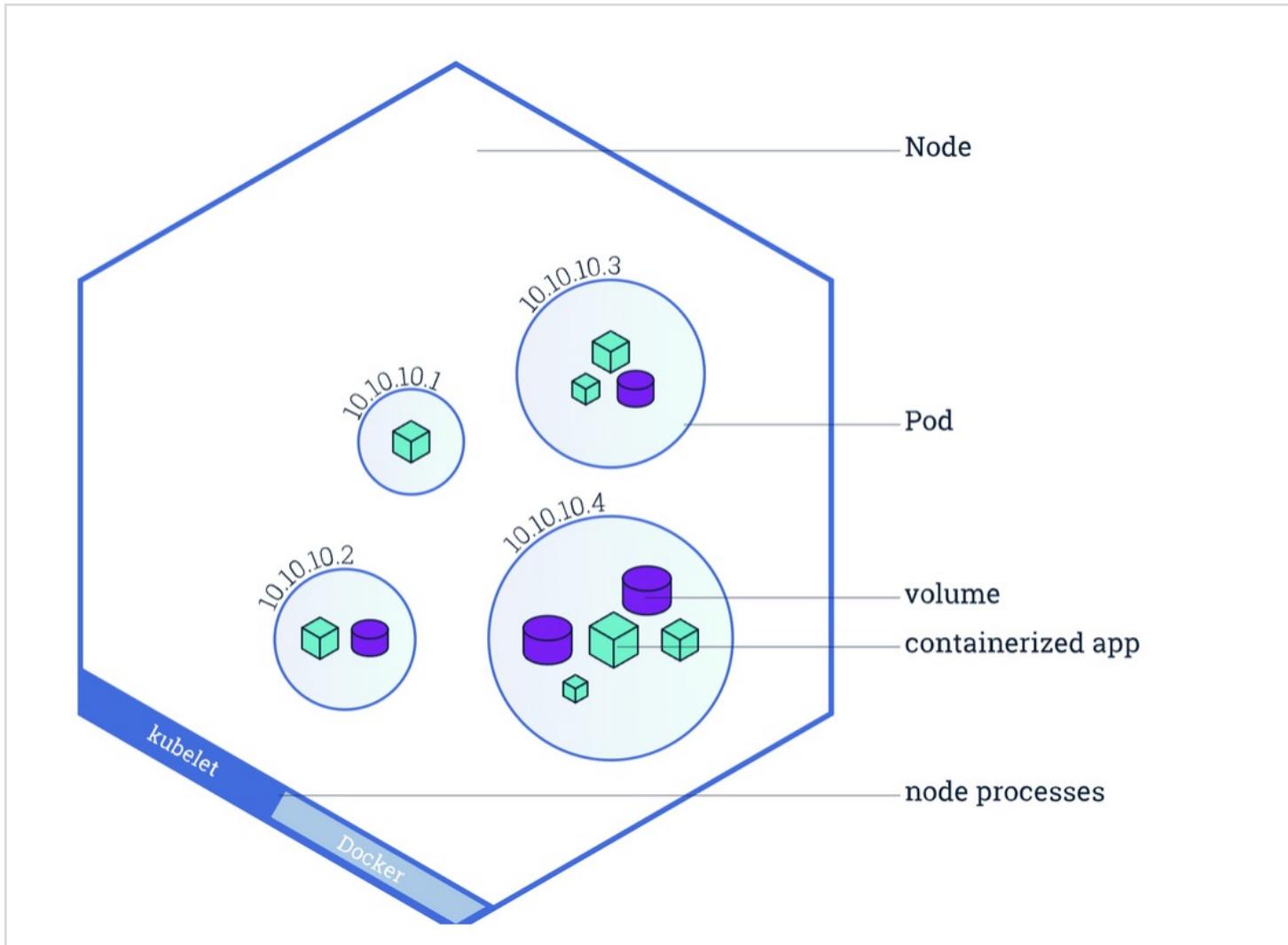
쿠버네티스 구성 요소



클러스터 내에서 워커 머신으로
동작하는 하나의 가상머신 또는
물리적 컴퓨터

컨테이너화된 애플리케이션을
실행하는 노드라고 하는 워커 머
신의 집합
컨트롤 플레인과 한개 이상의 노
드로 이루어진 구조

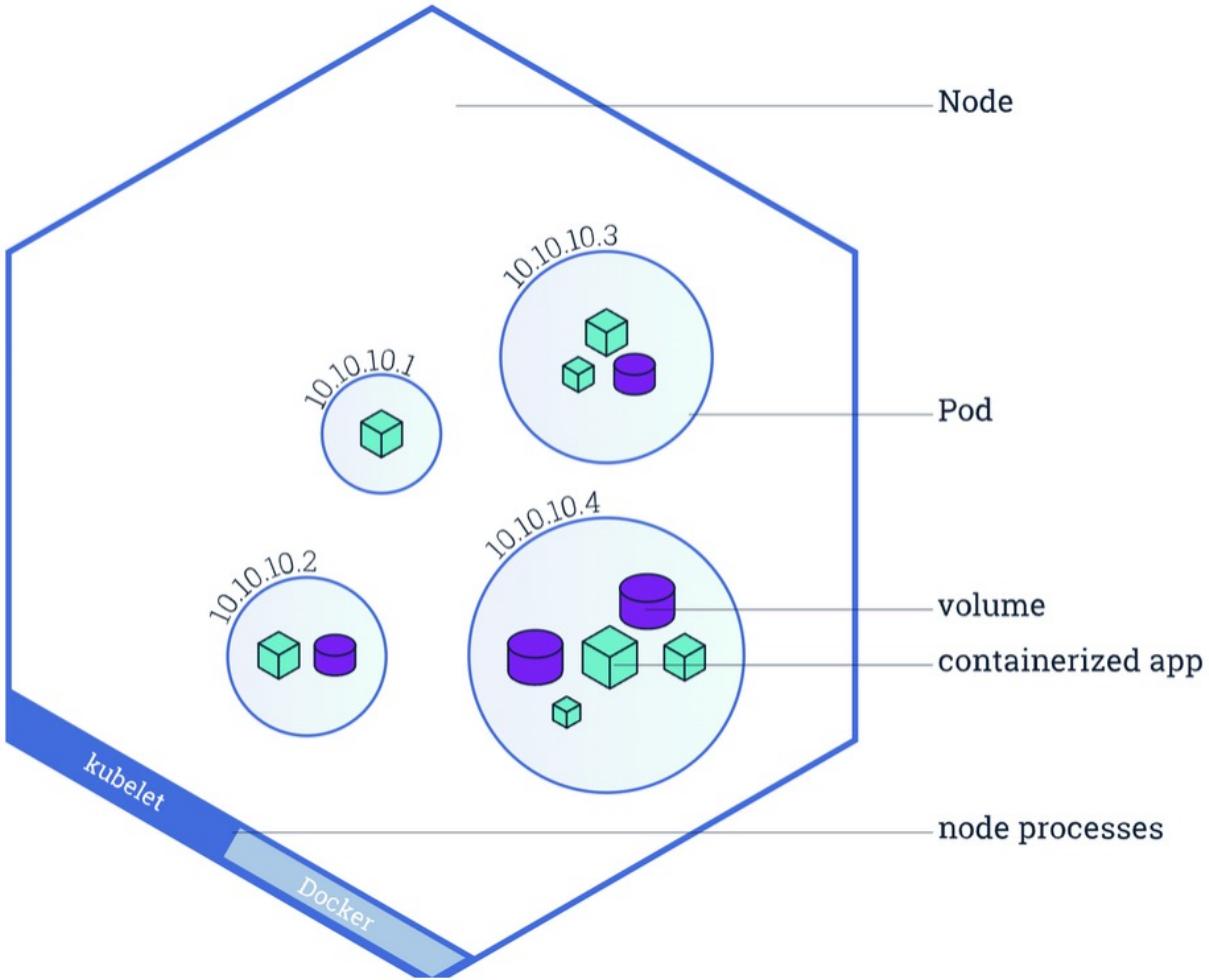
클러스터에 관한 전반적인 일을
수행하는 요소로 애플리케이션
을 스케줄링하고 클러스터를 조
율하는 역할을 수행. 즉, 클러스
터를 관리하는 구성 요소



Kubelet

k8s control plane과 Node 간 통신을 책임지는 프로세스이며 하나의 머신 상에서 동작하는 Pod와 Container를 관리

이외에도 container runtime과 kube-proxy라는 서비스가 존재..



Pod

K8s에서 최소 기능 단위이며 Container, storage, ip 네트워크 등으로 묶여 있음.

Pod은 desired state에 따라 죽고 살아나기를 반복 -> IP가 매번 달라짐 -> label을 사용

Labels은 key value 형태!

```
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: my-deployment
5    labels:
6      app: my-deployment-example
7  spec:
8    replicas: 3
9    selector:
10      matchLabels:
11        app: my-deployment-example
12  template:
13    metadata:
14      labels:
15        app: my-deployment-example
16  spec:
17    containers:
18      - name: nginx
19        image: nginx:latest
20        ports:
21          - containerPort: 80
22      - name: redis
23        image: redis
24        volumeMounts:
25          - name: redis-storage
26            mountPath: /data/redis
27        volumes:
28          - name: redis-storage
29            emptyDir: {}
```

apiVersion: 쿠버네티스 api 버전

kind: 만들 오브젝트의 종류

metadata: 오브젝트를 식별하기 위한 데이터

Spec : Pod에 대한 명세

Selector: 어떤 pod들을 대상으로 서비스할지

template : pod을 만들기 위한 template

Kubernetes objects란?!

```
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4      name: my-deployment
5      labels:
6          app: my-deployment-example
7  spec:
8      replicas: 3
9      selector:
10         matchLabels:
11             app: my-deployment-example
12 template:
13     metadata:
14         labels:
15             app: my-deployment-example
16 spec:
17     containers:
18         - name: nginx
19             image: nginx:latest
20             ports:
21                 - containerPort: 80
22         - name: redis
23             image: redis
24             volumeMounts:
25                 - name: redis-storage
26                     mountPath: /data/redis
27             volumes:
28                 - name: redis-storage
29                     emptyDir: {}
```

Kubernetes objects란 k8s 시스템(cluster내에서)에서 영 속성을 가지는 오브젝트.
보통 하나의 의도를 담은 레코드라고 표현!

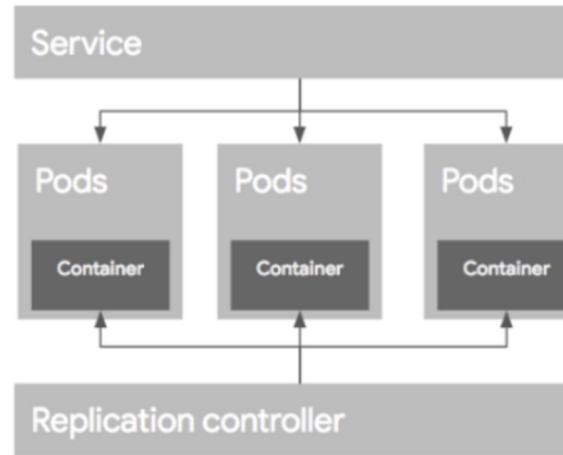
쉽게 생각하면 yaml파일에서 k8s가 제공하는 리소스를 가지고 만들어낸 인스턴스라고 생각하면 됨.
yaml -> k8s에게 리소스에 대한 spec을 제공.

spec이 이제 object에 대한 desired state가 되는거고 k8s 시스템에 전달하면 이 status를 유지시켜줌.

```
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: my-deployment
5    labels:
6      app: my-deployment-example
7  spec:
8    replicas: 3
9    selector:
10   matchLabels:
11     app: my-deployment-example
12   template:
13     metadata:
14       labels:
15         app: my-deployment-example
16   spec:
17     containers:
18       - name: nginx
19         image: nginx:latest
20       ports:
21         - containerPort: 80
22       - name: redis
23         image: redis
24         volumeMounts:
25           - name: redis-storage
26             mountPath: /data/redis
27         volumes:
28           - name: redis-storage
29             emptyDir: {}
```

```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: my-replicaset
  labels:
    app: my-replicaset
spec:
  replicas: 3
  selector:
    matchLabels:
      app: my-little-pod
  template:
    metadata:
      labels:
        app: my-little-pod
    spec:
      containers:
        - name: nginx
          image: nginx:latest
          ports:
            - containerPort: 80
        - name: redis
          image: redis
          volumeMounts:
            - name: redis-storage
              mountPath: /data/redis
          volumes:
            - name: redis-storage
              emptyDir: {}
```

ReplicaSet은 레플리카 파드 집합의 실행을 항상 안정적으로 유지하기 위한 목적으로 사용되고 지정된 수의 replica가 항상 실행되도록 보장.
Deployment는 더 상위 개념으로서 ReplicaSet을 관리하고 다른 유용한 기능과 함께 Pod에 대한 선언적 업데이트 등을 제공.



```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx-container
          image: nginx
          ports:
            - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: ingress-nginx
spec:
  type: NodePort
  ports:
    - name: http
      port: 80
      targetPort: 80
      nodePort: 30001
      protocol: TCP
  selector:
    app: nginx
```

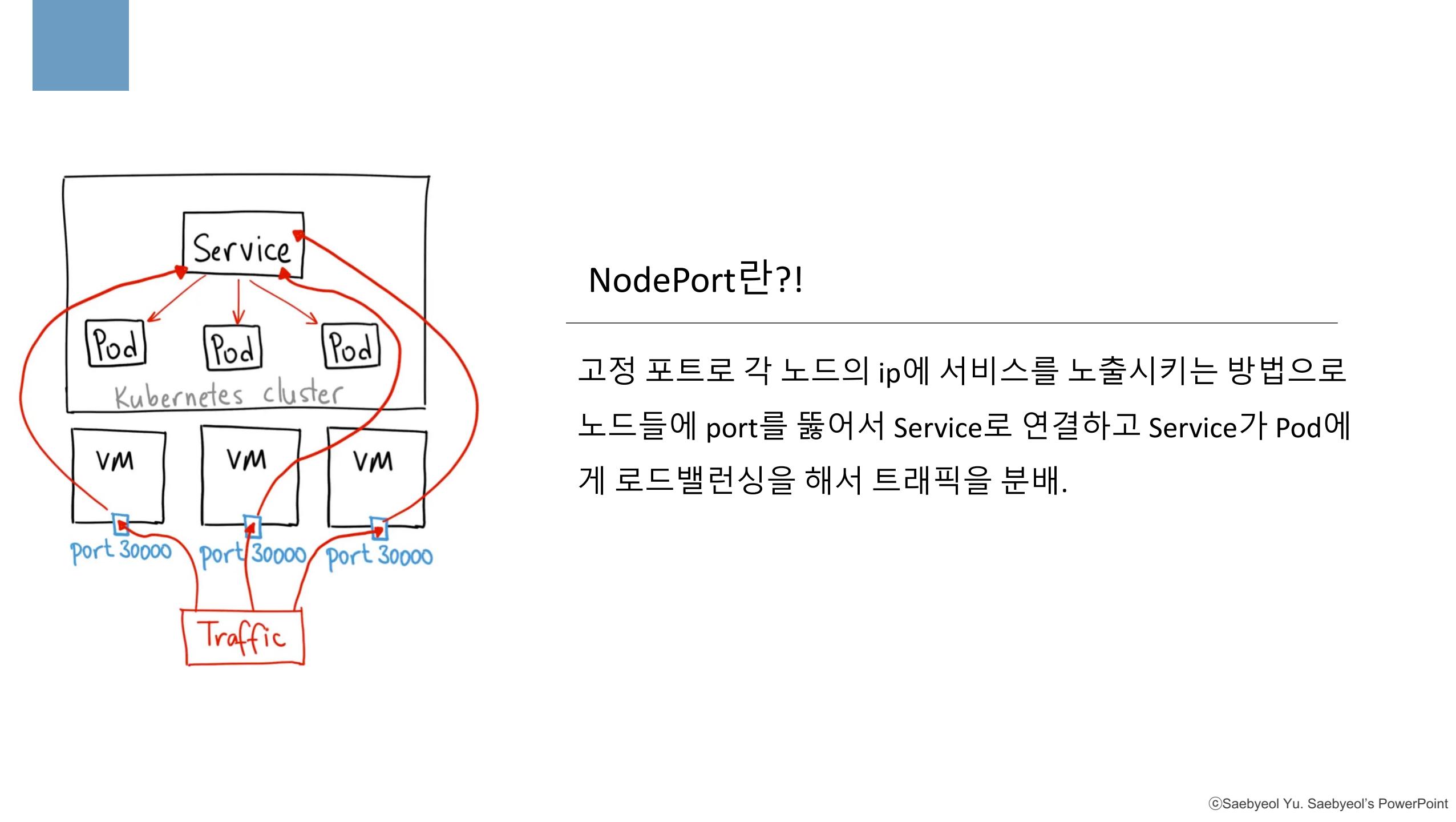
Service란?!

Pod 집합에서 실행중인 애플리케이션을 네트워크 서비스로 노출하는 추상화 방법!

selector로 서비스할 대상 pod집합을 결정.

ServiceSpec을 type에 명시해주고 방법은 여러가지가 존재

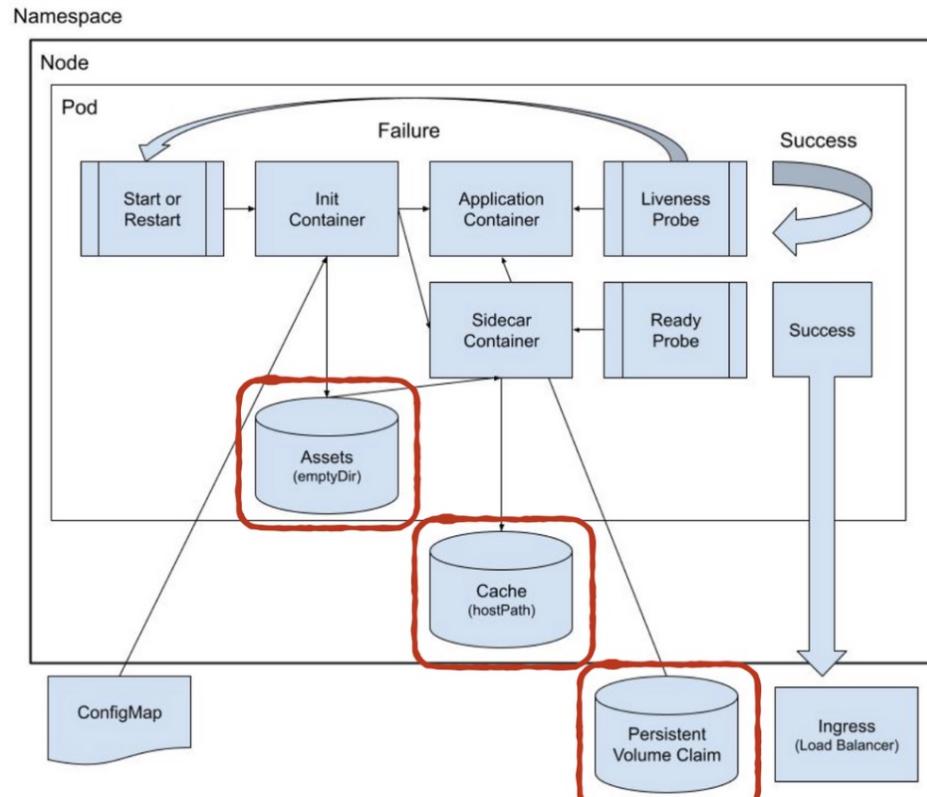
- ClusterIP
- NodePort
- LoadBalancer
- Ingress(얘는 서비스 type은 아님)
- ExternalName



NodePort란?!

고정 포트로 각 노드의 ip에 서비스를 노출시키는 방법으로 노드들에 port를 뚫어서 Service로 연결하고 Service가 Pod에게 로드밸런싱을 해서 트래픽을 분배.

Volume란?!

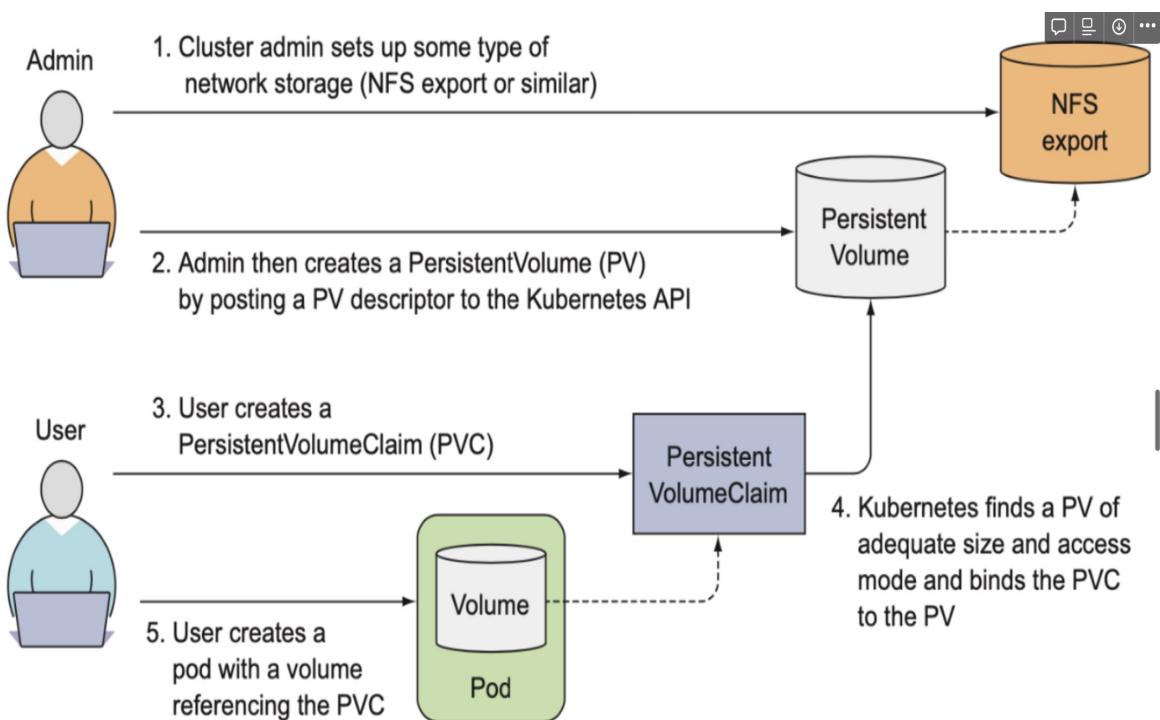


기본적으로 볼륨은 디렉토리이며 일부 데이터가 있을 수 있음.

보통 데이터를 저장하거나 공유하기 위해서 볼륨을 사용.

종류 또한 다양....

- emptyDir
- hostPath
- nfs
- PersistentVolumeClaim
- 회사를 이용하는 방법(gcePersistentDist, awsElasticeBlockStore, azureDist)



1,2 과정으로 admin에 의해 디스크가 설치가 되고 물리적인 장치를 가상화 -> 가상의 디스크를 만들어서 공유하도록 공개(PV에 대한 descriptor를 posting)

3번 과정을 통해 User는 PV에 접근할 수 있도록 어느정도 사양을 쓸건지 PVC를 작성

4번 과정을 통해 승인이 되고 내가 만든 PVC가 PV에 bind되면

5번 과정을 통해 내가 만든 컨테이너와 Pod들이 PV를 마치 자신의 볼륨인 것처럼 사용

Part 3

간단한 실습

```
~/Desktop/dcp-dockerHW/k8s/services git:(main) (19.297s)
minikube start
😊 Darwin 13.1 (arm64) 의 minikube v1.28.0
🌟 기존 프로필에 기반하여 docker 드라이버를 사용하는 중
👍 minikube 클러스터의 minikube 컨트롤러를 플레이어 노드를 시작하는 중
🚜 베이스 이미지를 다운받는 중 ...
🔄 Restarting existing docker container for "minikube" ...
📦 쿠버네티스 v1.25.3 을 Docker 20.10.20 런타임으로 설치하는 중
🔎 Kubernetes 구성 요소를 확인 ...
💡 ■ Using image docker.io/kubernetesui/dashboard:v2.7.0
💡 ■ Using image gcr.io/k8s-minikube/storage-provisioner:v5
💡 ■ Using image docker.io/kubernetesui/metrics-scraper:v1.0.8
💡 Some dashboard features require the metrics-server addon. To enable all features please run:
  minikube addons enable metrics-server

🌟 애드온 활성화 : default-storageclass, storage-provisioner, dashboard
🌟 끝났습니다! kubectl의 "minikube" 클러스터와 "default" 네임스페이스를 기본적으로 사용하도록 구성되었습니다.

~/Desktop/dcp-dockerHW/k8s/services git:(main) (0.255s)
kubectl get all
NAME           TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
service/kubernetes   ClusterIP   10.96.0.1    <none>           443/TCP   46d

~/Desktop/dcp-dockerHW/k8s/services git:(main) (0.291s)
kubectl apply -f nodeport_example.yaml
deployment.apps/nginx-deployment created
service/ingress-nginx created

~/Desktop/dcp-dockerHW/k8s/services git:(main) (0.122s)
kubectl get all
NAME                           READY   STATUS      RESTARTS   AGE
pod/nginx-deployment-6d7c859598-79jqz  0/1    ContainerCreating   0          5s
pod/nginx-deployment-6d7c859598-qsbtj  0/1    ContainerCreating   0          5s

NAME           TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
service/ingress-nginx   NodePort   10.96.100.148  <none>           80:30001/TCP  5s
service/kubernetes   ClusterIP   10.96.0.1    <none>           443/TCP   46d

NAME           READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/nginx-deployment  0/2      2           0          5s

NAME           DESIRED   CURRENT   READY   AGE
replicaset.apps/nginx-deployment-6d7c859598  2        2           0          5s
```

생성이든 수정이든 삭제든 쿠버네티스 오브젝트를 동작시키려면 k8s api를 이용해야하는 kubectl CLI가 우리가 필요한 k8s api를 대신 호출해주고 알아서 정보를 json으로 바꿔서 api request를 만들어줌.

```
~/Desktop/dcp-dockerHW/k8s/services git:(main)
minikube service ingress-nginx --url
http://127.0.0.1:60970
! Because you are using a Docker driver on darwin, the terminal needs to be open to run it.
```

```
~/Desktop/dcp-dockerHW/k8s/services git:(main)
kubectl run -i --rm --tty debug --image=alpine:latest --restart=Never -- ash -il
If you don't see a command prompt, try pressing enter.
debug:/# apk add curl
fetch https://dl-cdn.alpinelinux.org/alpine/v3.17/main/aarch64/APKINDEX.tar.gz
fetch https://dl-cdn.alpinelinux.org/alpine/v3.17/community/aarch64/APKINDEX.tar.gz
(1/5) Installing ca-certificates (20220614-r4)
(2/5) Installing brotli-libs (1.0.9-r9)
(3/5) Installing nghttp2-libs (1.51.0-r0)
(4/5) Installing libcurl (7.87.0-r1)
(5/5) Installing curl (7.87.0-r1)
Executing busybox-1.35.0-r29.trigger
Executing ca-certificates-20220614-r4.trigger
OK: 10 MiB in 20 packages
debug:/# curl http://nginx-deployment
curl: (6) Could not resolve host: nginx-deployment
debug:/# curl http://ingress-nginx
<!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>
```

Minikube를 이용하면 service에 접근할 수 있음(예전에는 url과 NodePort로 정의한 port로 접근할 수 있었지만 보안 때문에 minikube를 이용해야함)

다른 방법으로 pod을 만들고 접근하는 방법도 존재!

The screenshot shows the Kubernetes dashboard interface. The top navigation bar includes the Kubernetes logo, a dropdown for 'default' namespace, a search bar with placeholder '검색' (Search), and a '+' button. The main content area is titled '워크로드' (Workload) and displays three main sections: '워크로드 상태' (Workload Status), '디플로이먼트' (Deployment), and '파드' (Pod).

워크로드 상태

- 크론 잡
- 데몬 세트
- 디플로이먼트
- 잡
- 파드
- 레플리카 세트
- 레플리케이션 컨트롤러
- 스테이트풀 세트

디플로이먼트

이름	이미지	레이블	파드	생성 시간
nginx-deployment	nginx	app: nginx	2 / 2	12.minutes.ago

파드

이름	이미지	레이블	노드	상태	제시작	CPU 사용량 (cores)	메모리 사용량 (bytes)	생성 시간
nginx-deployment-6d7c859598-79jqz	nginx	app: nginx pod-template-hash: 6d7c859598	minikube	Running	0	-	-	12.minutes.ago
nginx-deployment-6d7c859598-q8tj	nginx	app: nginx pod-template-hash: 6d7c859598	minikube	Running	0	-	-	12.minutes.ago

Dashboard도 제공됨!

```
~/Desktop/dcp-dockerHW/k8s/services git:(main)
minikube dashboard
🟡 Verifying dashboard health ...
🔴 프록시를 시작하는 중 ...
🟡 Verifying proxy health ...
🟡 Opening http://127.0.0.1:61017/api/v1/namespaces/kubernetes-dashboard/services/http:kubernetes-
-dashbaord:/proxy/ in your default browser...
```

```

~/Desktop/dcp-dockerHW/k8s/services git:(main) (0.131s)
kubectl get pods
NAME READY STATUS RESTARTS AGE
nginx-deployment-6d7c859598-79jqz 1/1 Running 0 14m
nginx-deployment-6d7c859598-qsbtj 1/1 Running 0 14m

~/Desktop/dcp-dockerHW/k8s/services git:(main) (0.101s)
kubectl get pods -o wide
NAME INATED NODE READINESS GATES READY STATUS RESTARTS AGE IP NODE NOM
nginx-deployment-6d7c859598-79jqz <none> 1/1 Running 0 14m 172.17.0.5 minikube <no
nginx-deployment-6d7c859598-qsbtj <none> 1/1 Running 0 14m 172.17.0.6 minikube <no

~/Desktop/dcp-dockerHW/k8s/services git:(main) (0.69s)
kubectl delete pod nginx-deployment-6d7c859598-79jqz
pod "nginx-deployment-6d7c859598-79jqz" deleted

~/Desktop/dcp-dockerHW/k8s/services git:(main) (0.096s)
kubectl get pods -o wide
NAME INATED NODE READINESS GATES READY STATUS RESTARTS AGE IP NODE NOM
nginx-deployment-6d7c859598-4kqrw <none> 1/1 Running 0 3s 172.17.0.5 minikube <no
nginx-deployment-6d7c859598-qsbtj <none> 1/1 Running 0 15m 172.17.0.6 minikube <no

~/Desktop/dcp-dockerHW/k8s/services git:(main)±3
|
```

```

~/Desktop/dcp-dockerHW/k8s/services git:(main) (0.234s)
kubectl delete -f nodeport_example.yaml
deployment.apps "nginx-deployment" deleted
service "ingress-nginx" deleted

~/Desktop/dcp-dockerHW/k8s/services git:(main) (0.121s)
kubectl get all
NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE
service/kubernetes ClusterIP 10.96.0.1 <none> 443/TCP 46d

```

Pod을 삭제하는건 가능하지만 영구적으로
삭제가 되진 않음.

(k8s가 deployment에 정의된 desired state
를 만족시키기 위해 pod을 재생성)

아래처럼 pod을 관리하고 있는
deployment를 삭제해줘야함!!

참고 자료

<https://kubernetes.io/>
쿠버네티스 공식 홈

<https://kubernetes.io/ko/docs/reference/glossary/?all=true#term-control-plane>
쿠버네티스 용어집