K8s Pod

K8s에서 Pod 생성 방법 2가지

- 명령어 kubectl run으로 pod 생성과 실행
- YAML(야믈) 파일을 이용한 pod 생성과 실행

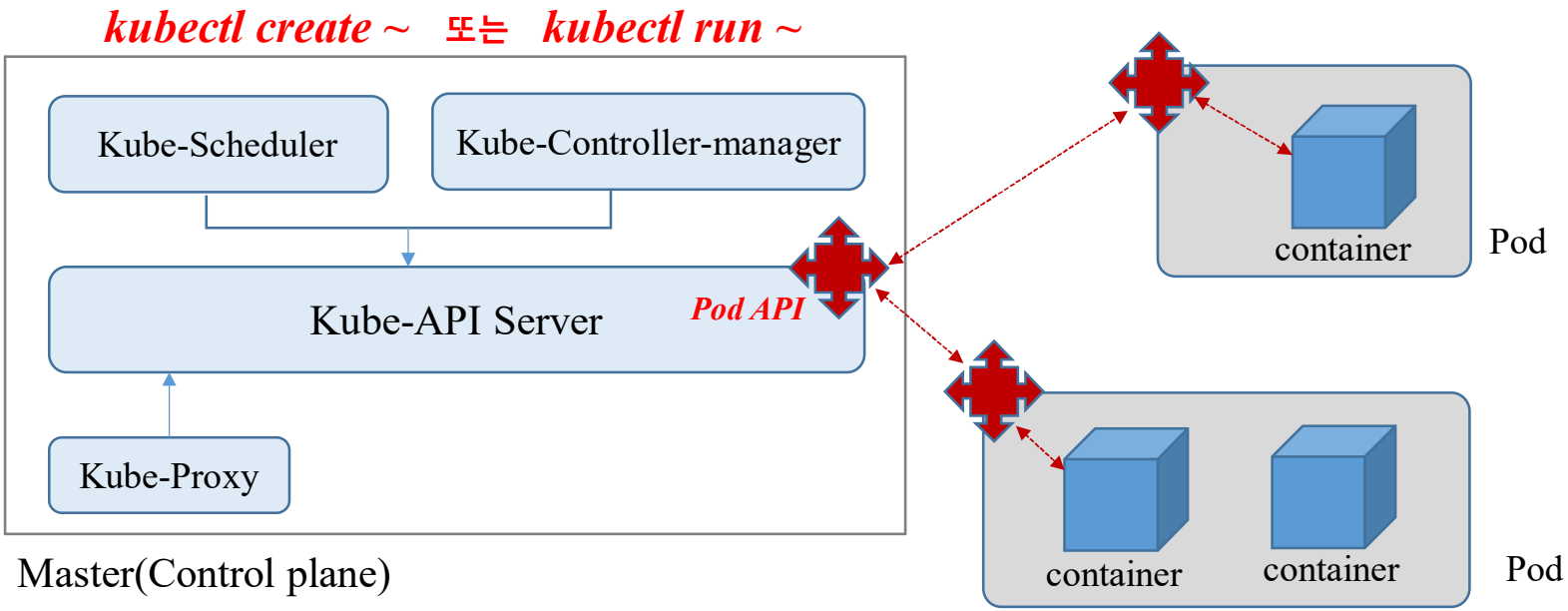
kubectl create ~ 또는 kubectl run ~

Pod(파드)

- API를 통해 container를 동작시킬 수 없음

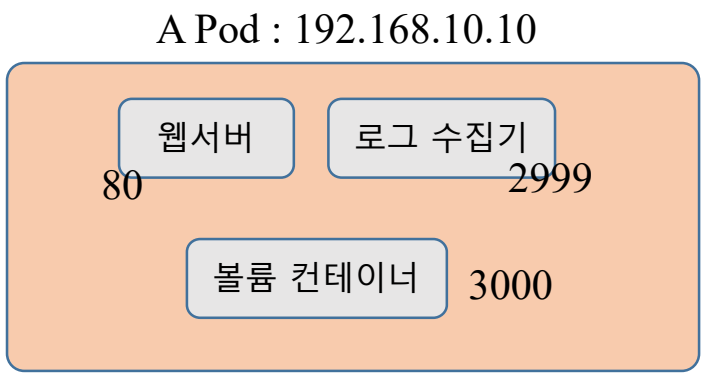
  - K8s에는 container를 생성하는 API가 없음

- Pod API를 통해 pod를 생성/동작시키고 pod를 통해 container를 생성/동작 시킴



Pod(파드)

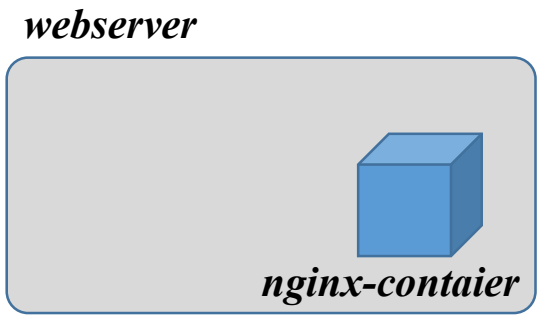
- 컨테이너들을 모은 애플리케이션의 최소 단위
- K8s는 Pod 단위로 컨테이너들을 묶어서 관리
- 파드에는 하나 또는 여러 개의 컨테이너가 포함 될 수 있음



- Pod 안에 3개의 컨테이너로 구성
- Pod 안에 있는 컨터이너들이 IP 하나를 공유
- 외부에서 파드 안 컨테이너와 통신 할 때는 컨테이너마다 다르게 설정된 포트를 사용

YAML 파일을 이용한 pod 생성과 실행

```
apiVersion:  v1
kind: Pod
metadata:
  name: webserver
Spec:
    containers:
       - name: nginx-container
         image: nginx:1.14
         ports:
           - containerPort: 80
           - containerPort : 443
```



YAML(야믈, *YAML ain't markup language*)

- K8S는 클러스터의 Object나 Controller가 어떤 상태여야 하는지 적용 할 때에는 YAML 형식의 template 사용
- YAML 문법

  - 들여쓰기에 따라 구조가 바뀜

  - 들여쓰기는 Tab이 아닌 Space bar 사용 (**공백 문자들로 기본구조를 구성**(단, tab은 사용하지 않음))

  - #로 주석을 표시

  ❶ Scalars(string/numbers) : 'Key: value' 형태로 표시

  ':' 다음에 공백 문자가 와야한다. (붙여쓰면 오류)

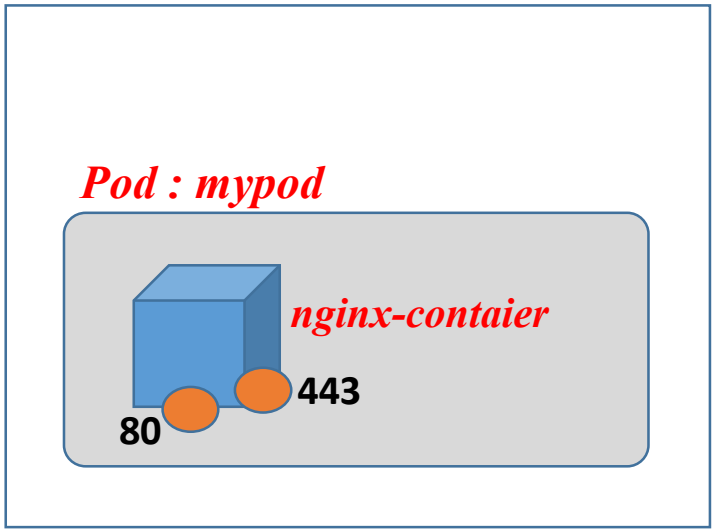
  ❷ Sequence(arrays/lists) : '-'(하이픈) 여러 개 나열

vi test.yaml

```
apiVersion:  v1
kind: Pod
metadata:
  name: mypod
  namespace: cominfo
spec:
  containers:
    - name: nginx-container
      image: nginx:1.14
      ports:
      - containerPort: 80
      - containerPort : 443
```

Scalars

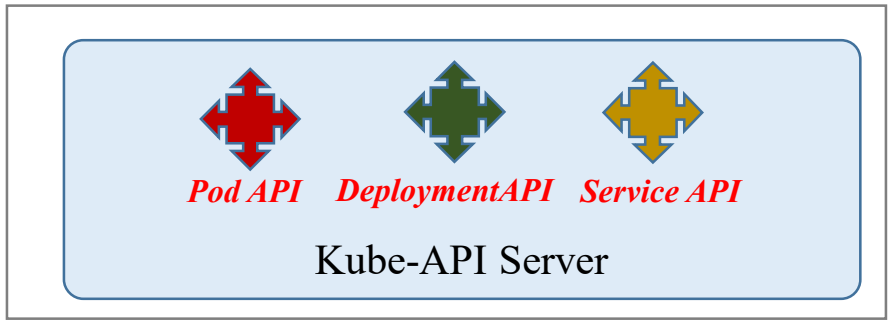
Sequences(Arrary)



API Version

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

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



Master(Control plane)

#kubectl api-resources

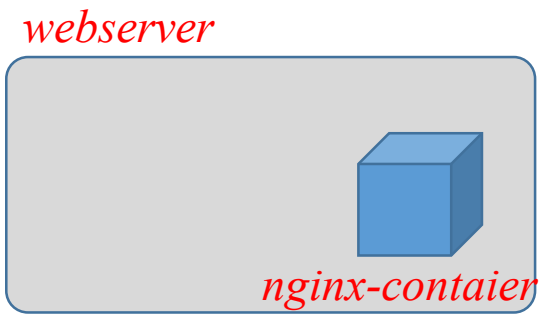
#kubectl explain pod

#kubectl explain deployment

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

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

```
apiVersion:  v1                //사용하려는 K8S API 버전 명시
kind: Pod                      //어떤 종류의 object 또는 controller에 작업인지 명시
metadata:                      //메타데이타 설정(Pod명)
  name: webserver
Spec:                          //파드가 어떤 컨테이너를 갖고 실행하며 실행 시 어떻게 동작해야 할 지 명시
    containers:
      - name: nginx-container
        image: nginx:1.14
        ports:
          - containerPort: 80
          - containerPort : 443
```



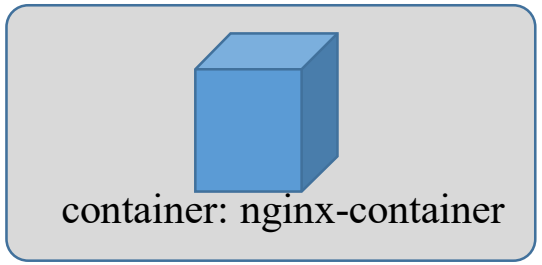
❶ kubectl run 명령어

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

❷ YAML 이용

```
apiVersion:  v1
kind: Pod
metadata:
  name: nginx-pod
spec:
    containers:
      - name: nginx-container
        image: nginx:1.14
        ports:
        - containerPort: 80
          protocol: TCP
```

#kubectl create –f pod-nginx.yaml



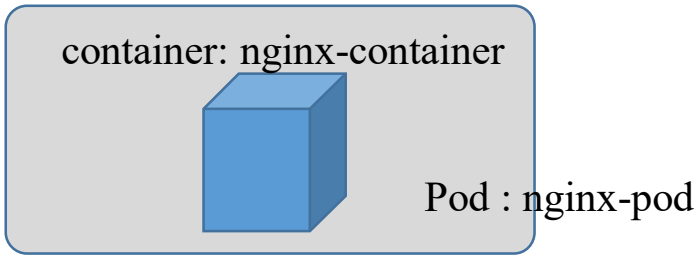
Pod : nginx-pod

<<pod-nginx.yaml>>

실습1. Single Pod

❶

```
root@master:/K8s# cat pod-nginx.yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx-pod
spec:
  containers:
  - name: nginx-container
    image: nginx:1.14
    ports:
    - containerPort: 80
      protocol: TCP
root@master:/K8s#
```

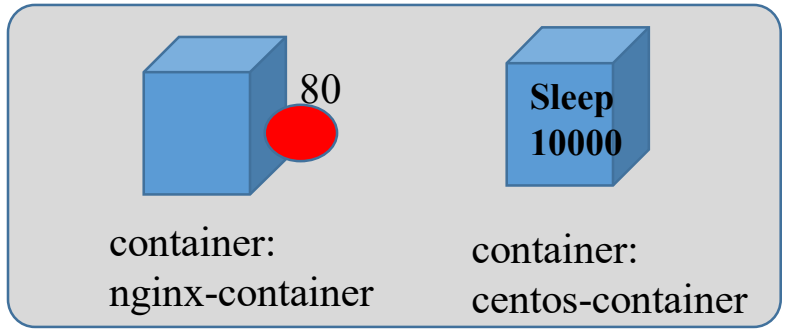


#kubectl create -f pod-nginx.yaml
#kubect get pods –o wide
#watch kubect get pods –o wide

❷

```
root@master:/K8s# kubectl create -f pod-nginx.yaml
pod/nginx-pod created
root@master:/K8s#
root@master:/K8s# kubectl get pods
NAME        READY   STATUS    RESTARTS   AGE
nginx-pod   1/1     Running   0          14s
root@master:/K8s#
root@master:/K8s# kubectl get pods -o wide
NAME        READY   STATUS    RESTARTS   AGE   IP          NODE       NOMINATED NODE   READINESS GATES
nginx-pod   1/1     Running   0          26s   10.42.0.1   worker03   <none>           <none>
root@master:/K8s#
```

실습2. Multi-pod



Pod : multipod

#kubectl create -f pod-multi.yaml
#kubect get pods –o wide
#watch kubect get pods –o wide

❶

```
root@master:/K8s# cat pod-multi.yaml
apiVersion: v1
kind: Pod
metadata:
  name: multipod
spec:
  containers:
  - name: nginx-container
    image: nginx:1.14
    ports:
    - containerPort: 80
  - name: centos-container
    image: centos:7
    command:
    - sleep
    - "10000"
root@master:/K8s#
```

❷

```
root@master:/K8s# kubectl get pods
NAME        READY   STATUS    RESTARTS   AGE
multipod    2/2     Running   0          3m56s
nginx-pod   1/1     Running   0          6m55s
root@master:/K8s#
root@master:/K8s# kubectl get pods -o wide
NAME        READY   STATUS    RESTARTS   AGE    IP          NODE       NOMINATED NODE   READINESS GATES
multipod    2/2     Running   0          4m4s   10.40.0.1   worker01   <none>           <none>
nginx-pod   1/1     Running   0          7m3s   10.42.0.1   worker03   <none>           <none>
root@master:/K8s#
```

Pod안 Container 운영체제 확인

#kubectl exec multipod –c centos-container -- /bin/bash
#cat /etc/os-release

```
root@master:/K8s# kubectl exec multipod -c centos-container -- /bin/bash
root@master:/K8s# cat /etc/os-release
NAME="Ubuntu"
VERSION="20.04 LTS (Focal Fossa)"
ID=ubuntu
ID_LIKE=debian
PRETTY_NAME="Ubuntu 20.04 LTS"
VERSION_ID="20.04"
HOME_URL="https://www.ubuntu.com/"
SUPPORT_URL="https://help.ubuntu.com/"
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-policy"
VERSION_CODENAME=focal
UBUNTU_CODENAME=focal
root@master:/K8s#
```

Pod안 Container 접근 후 내용 수정

#kubectl exec multipod –c nginx-container –it -- /bin/bash
#cd /usr/share/nginx/html
#echo “Hello~ Web” > index.thml
#cat index.html
#exit
#curl 10.40~

```
root@master:/K8s# kubectl exec multipod -c nginx-container -it -- /bin/bash
root@multipod:/# cd /usr/share/nginx/html
root@multipod:/usr/share/nginx/html# ls
50x.html  index.html
root@multipod:/usr/share/nginx/html# echo "Hello~ WEB" > index.html
root@multipod:/usr/share/nginx/html# exit
exit
root@master:/K8s# curl 10.40.0.1:80
Hello~ WEB
root@master:/K8s#
```

kubectl describe pod multipod

```
  default-token-jzsv4:
    Type:        Secret (a volume populated by a Secret)
    SecretName:  default-token-jzsv4
    Optional:    false
QoS Class:       BestEffort
Node-Selectors:  <none>
Tolerations:     node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                 node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type     Reason     Age                   From               Message
  ----     ------     ----                  ----               -------
  Normal   Scheduled  4m50s                 default-scheduler  Successfully assigned default/multiple to node2
  Normal   Pulling    4m49s                 kubelet            Pulling image "nginx:1.14"
  Normal   Pulled     4m42s                 kubelet            Successfully pulled image "nginx:1.14" in 6.844028704s
  Normal   Created    4m41s                 kubelet            Created container nginx4-container
  Normal   Started    4m41s                 kubelet            Started container nginx4-container
  Normal   Pulling    4m41s                 kubelet            Pulling image "nginx:1.13"
  Normal   Pulled     4m33s                 kubelet            Successfully pulled image "nginx:1.13" in 7.647751029s
  Normal   Started    3m40s (x4 over 4m32s) kubelet            Started container nginx3-container
  Warning  BackOff    3m8s (x6 over 4m25s)  kubelet            Back-off restarting failed container
  Normal   Pulled     2m55s (x4 over 4m29s) kubelet            Container image "nginx:1.13" already present on machine
  Normal   Created    2m54s (x5 over 4m32s) kubelet            Created container nginx3-container
```

```
[node1 ~]$ cat mpod.yaml
apiVersion: v1
kind: Pod
metadata:
  name: multiple
spec:
  containers:
  - name: nginx4-container
    image: nginx:1.14
    ports:
    - containerPort: 80
  - name: nginx3-container
    image: nginx:1.13
    ports:
    - containerPort: 8080
[node1 ~]$
```

```
[node1 ~]$ cat mpod.yaml
apiVersion: v1
kind: Pod
metadata:
  name: multiple
spec:
  containers:
  - name: nginx4-container
    image: nginx:1.14
    ports:
    - containerPort: 80
  - name: nginx3-container
    image: nginx:1.13
    command:
    - sleep
    - "10000"

[node1 ~]$
```

kubectl edit pod ***multiple***

kubectl delete pods multiple 또는 kubectl delete pods --all

```
[node1 ~]$
[node1 ~]$ kubectl get pods -o wide
NAME       READY   STATUS    RESTARTS   AGE   IP         NODE    NOMINATED NODE   READINESS GATES
multiple   2/2     Running   0          17m   10.5.1.3   node2   <none>           <none>
single     1/1     Running   0          28m   10.5.2.2   node3   <none>           <none>
[node1 ~]$ kubectl delete pod --all
pod "multiple" deleted
pod "single" deleted
[node1 ~]$ kubectl get pods -o wide
No resources found in default namespace.
[node1 ~]$
```

실습 명령어 정리

❶ Multi pod 생성

vi mpod.yaml

kubectl create –f *mpod.yaml*

❷ Pod 정보 확인

kubectl get pods

kubectl get pods –o wide

❸ Pod 상세 정보 확인(장애처리)와 수정

kubectl describe pod *multiple*

kubectl edit pod *multiple*

❹ Pod 삭제

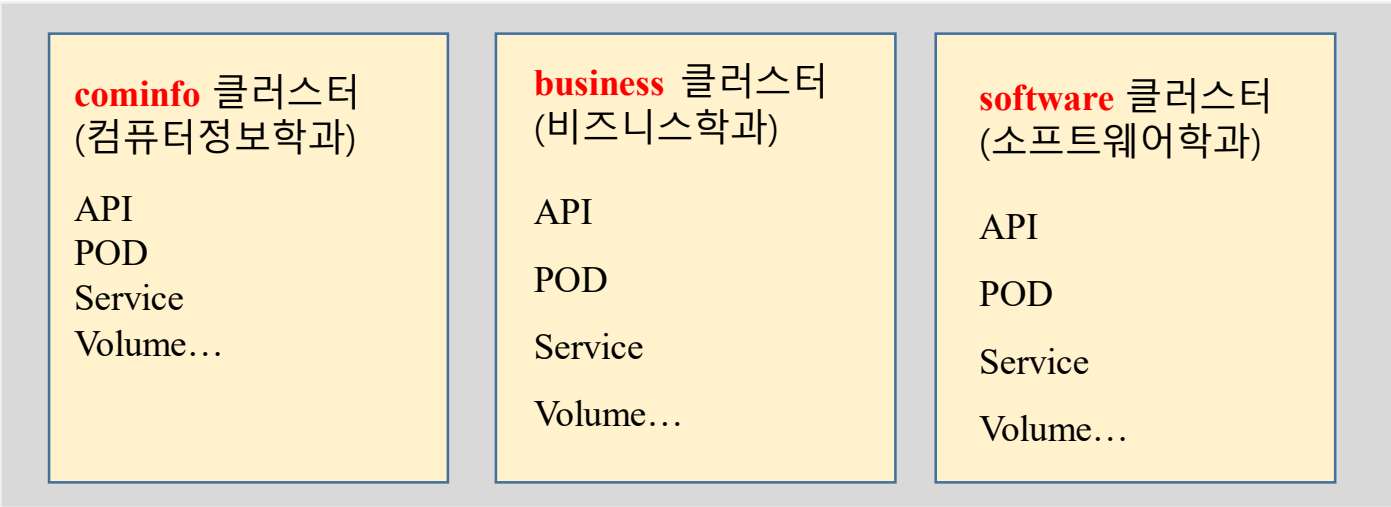
kubectl delete pod *multiple*

kubectl delete pod --all

NameSpace

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

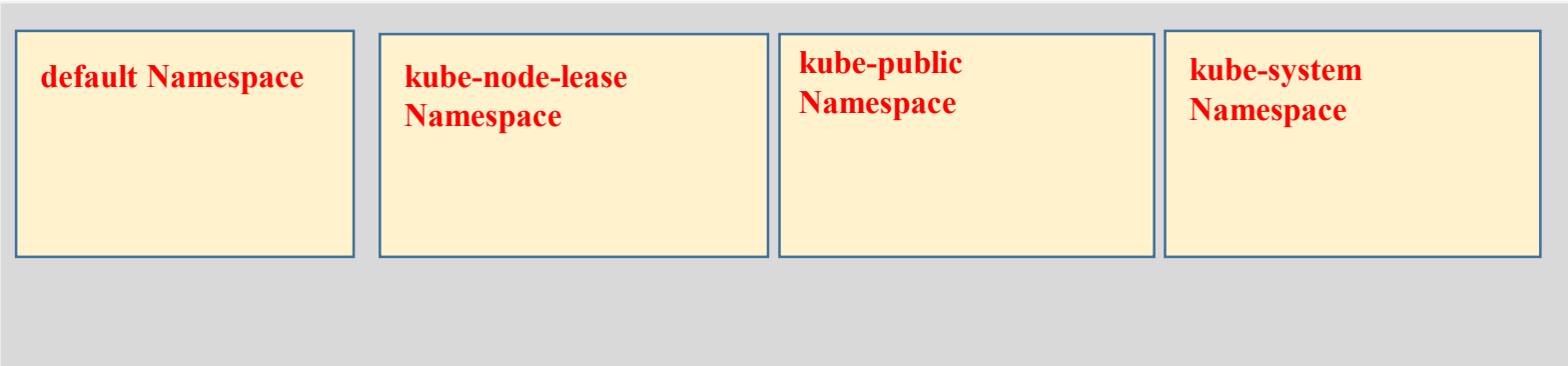


물리적 K8S 클러스터

현재 생성된 NameSpace 확인

kubectl get namespaces

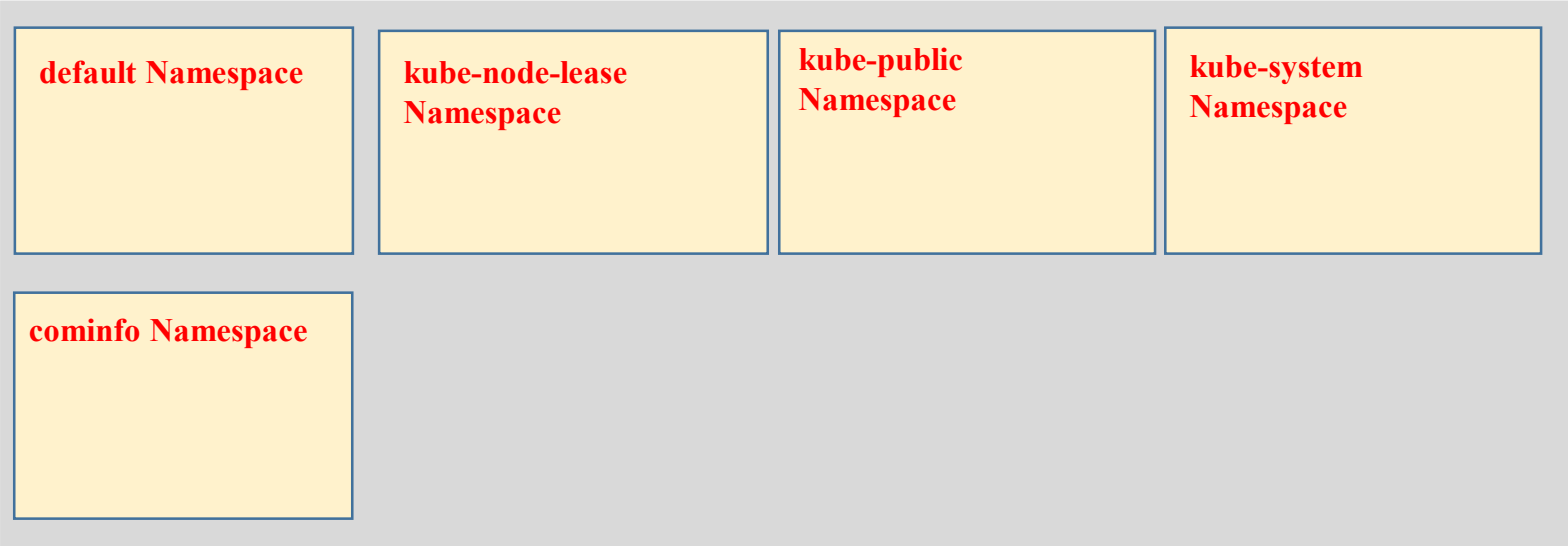
```
Terminal Host 1
controlplane $ kubectl get namespaces
NAME              STATUS   AGE
default           Active   42s
kube-node-lease   Active   43s
kube-public       Active   43s
kube-system       Active   43s
controlplane $
```



NameSpace 생성

kubectl create namespace **cominfo**

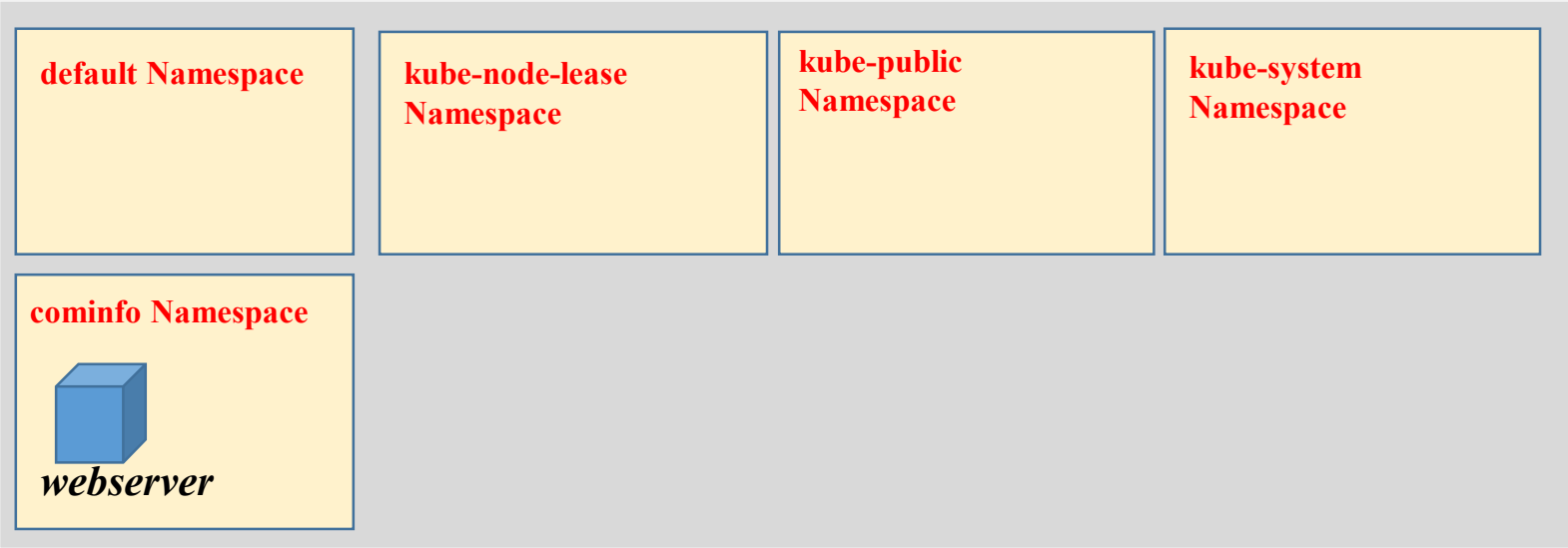
```
controlplane $ kubectl create namespace cominfo
namespace/cominfo created
controlplane $ kubectl get namespaces
NAME              STATUS   AGE
cominfo           Active   11s
default           Active   28m
kube-node-lease   Active   28m
kube-public       Active   28m
kube-system       Active   28m
controlplane $
```



지정된 NameSpace에 pod 생성

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

```
controlplane $ kubectl run webserver --image=nginx:1.14 --port 80 --namespace cominfo
pod/webserver created
controlplane $
controlplane $ kubectl get pods --namespace cominfo
NAME        READY   STATUS    RESTARTS   AGE
webserver   1/1     Running   0          30s
controlplane $
```



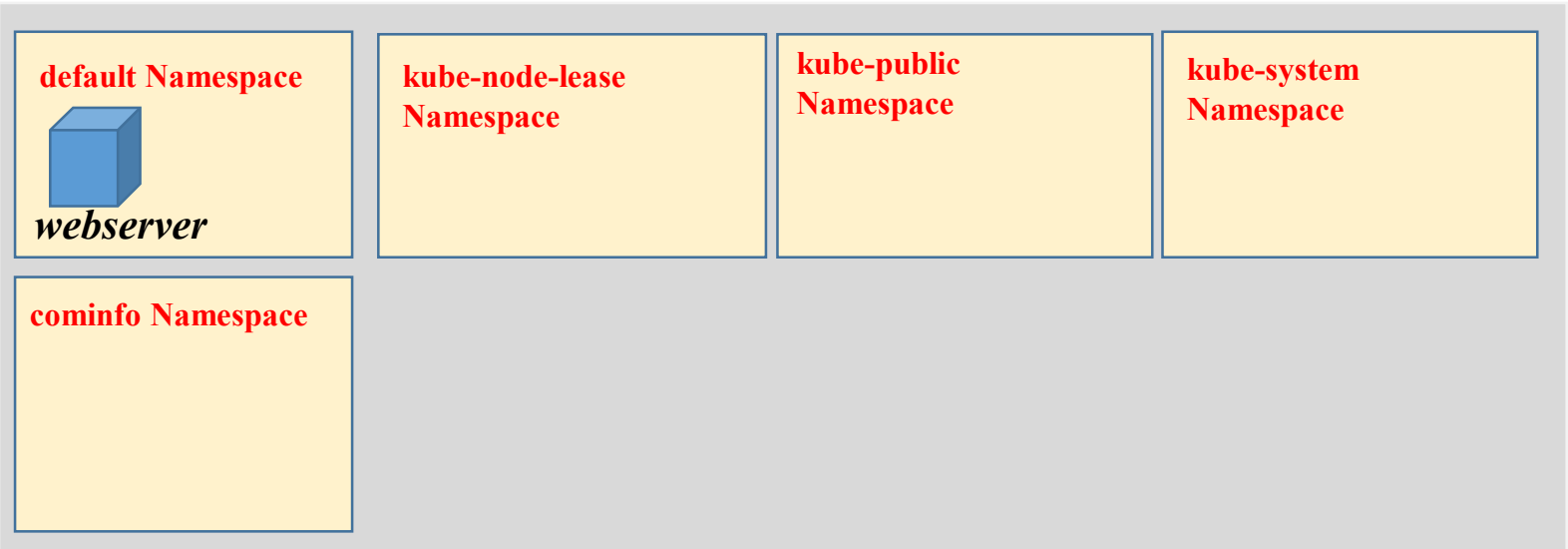
기본 namespace

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
```

kubectl get pod

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

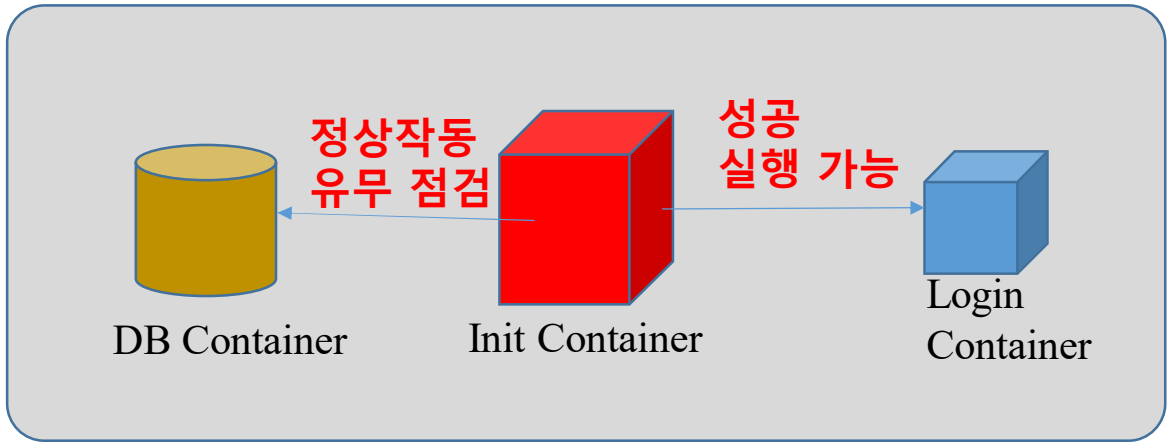


Init Container

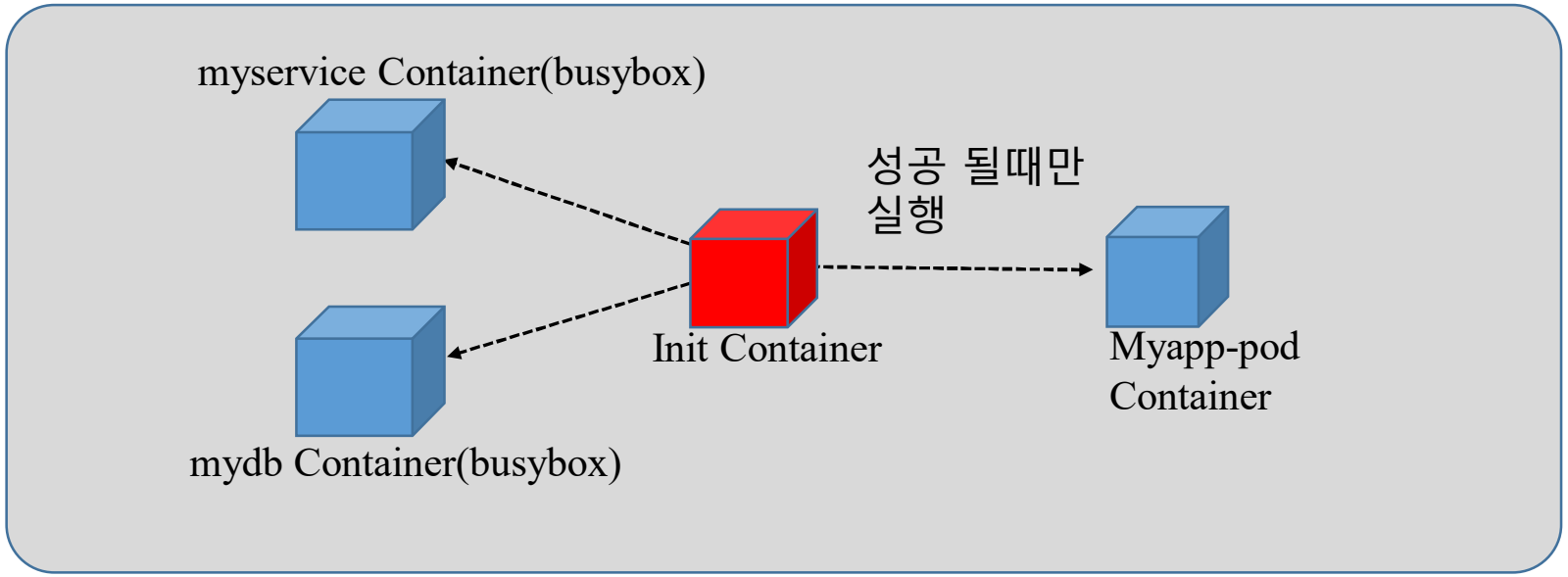
초기화 컨테이너(init container)

- Main container(App container)가 실행되기 전 동작시킨 컨테이너
- Main Container가 실행되기 전에 사전 작업이 필요한 경우 사용
- 초기화 컨터이너가 모두 실행된 후에 App container를 실행
  - Init container는 여러 개 구성할 수 있음
  - Init container 실행이 실패하면 성공할 때까지 재시작함
  - Init container가 모두 실행 된 후 App Container 실행이 시작

초기화 컨테이너(init container)

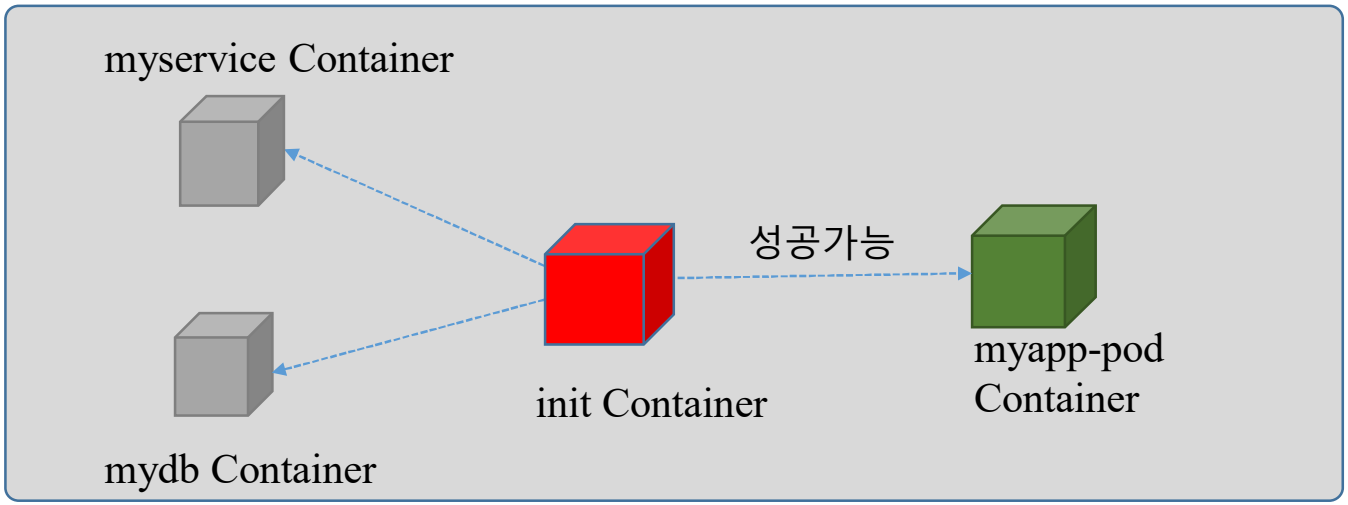


Pod



Alt+Enter

```
[node1 ~]$ kubectl get pods -o wide
NAME        READY   STATUS     RESTARTS   AGE   IP         NODE    NOMINATED NODE   READINESS GATES
myapp-pod   0/1     Init:0/2   0          48s   10.5.1.5   node2   <none>           <none>
[node1 ~]$
```




```
apiVersion: v1
kind: Service
metadata:
  name: myservice
spec:
  ports:
  - protocol: TCP
    port: 80
    targetPort: 9376
```

myservice.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: mydb
spec:
  ports:
  - protocol: TCP
    port: 80
    targetPort: 9377
```

mydb.yaml

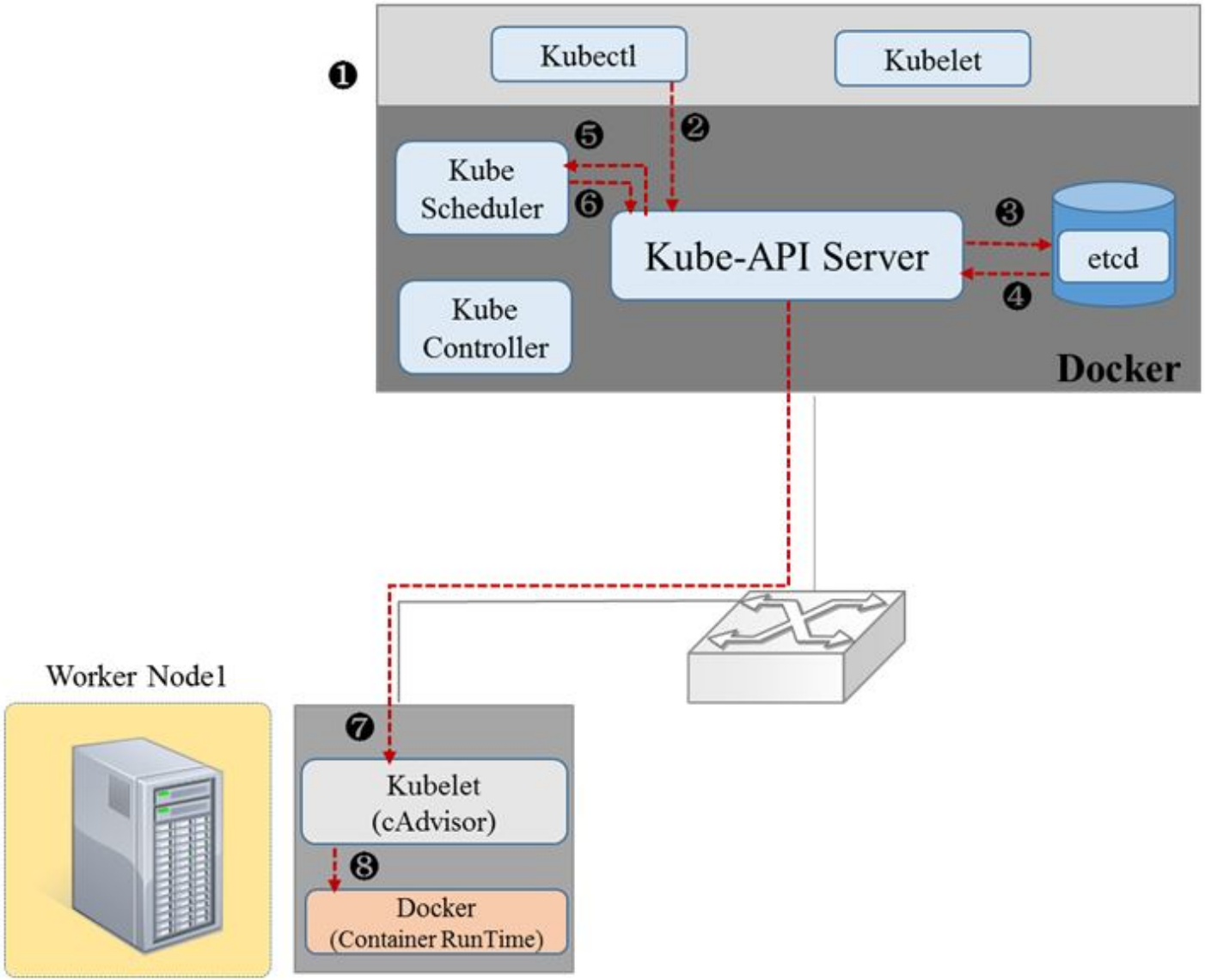
myapp-pod.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
spec:
  containers:
  - name: myapp-container
    중간 생략~~~
  initContainers:
  - name: myservice
    중간 생략~~~
  - name: mydb
    중간 생략~~~
```

init Container

- kubectl delete pod --all
- kubectl get pods –o wide --watch
- kubectl create –f init-container-exam.yaml
- kubectl create –f init-container-exam-svc.yaml
- kubectl create –f init-container-exam-db.yaml
- kubectl get pods - o wide
- kubectl describe pods *pod 명*

Static Pod


❶
Kubectl
Kubelet
❷
❺
Kube
Scheduler
❻
Kube-API Server
❸
etcd
❹
Kube
Controller
Docker
Worker Node1
❼
Kubelet
(cAdvisor)
❽
Docker
(Container RunTime)

Static Pod

- Kube-apiserver를 통하지 않고 kubelet이 직접 실행하는 pod
  - API 서버 없이 특정 노드에 있는 kubelet 데몬에 의해 직접관리
  - kubelet daemon에 의해 관리되는 container
- Kubelet이 직접 관리하면서 이상이 생기면 재시작

실습. Static Pod

```
[node1 ~]$ cd /var/lib/kubelet
[node1 kubelet]$ ls
config.yaml  cpu_manager_state  device-plugins  kubeadm-flags.env
[node1 kubelet]$ cat config.yaml
apiVersion: kubelet.config.k8s.io/v1beta1
authentication:
```

```
kind: KubeletConfiguration
logging: {}
nodeStatusReportFrequency: 0s
nodeStatusUpdateFrequency: 0s
rotateCertificates: true
runtimeRequestTimeout: 0s
shutdownGracePeriod: 0s
shutdownGracePeriodCriticalPods: 0s
staticPodPath: /etc/kubernetes/manifests
streamingConnectionIdleTimeout: 0s
syncFrequency: 0s
volumeStatsAggPeriod: 0s
[node1 kubelet]$
```

Pod lifecycle

- Pod 생성부터 삭제까지의 과정에서 생명주기(lifecycle)가 있음

kubectl get pods –o wide --watch

```
[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 생명주기(lifecycle)

Pending	K8S 시스템에 pod를 생성하는 중임을 뜻함 Container 이미지를 다운로드한 후 전체 container를 실행하는 도중 Pod안에 전체 container가 실행될때까지 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 실행 종료된 상태로 재시작 되지 않았음
Failed	Pod 안 모든 container가 정상적으로 실행 종료되지 않은 container가 있는 상태 Container 종료코드가 0이 아니면 비정상종료 또는시스템이 직접 Container를 종료한것
Unknown	Pod의 상태를 알 수 없는 상태 (Pod가 있는 노드와 통신 할 수 없을 때)

```
[node1 kubelet]$ kubectl get pods
NAME        READY   STATUS     RESTARTS   AGE
myapp-pod   0/1     Init:0/2   0          11s
[node1 kubelet]$ kubectl describe pods myapp-pod
Name:         myapp-pod
Namespace:    default
Priority:     0
Node:         node2/192.168.0.12
Start Time:   Tue, 03 May 2022 04:01:37
Labels:       app=myapp
Annotations:  <none>
Status:       Pending
IP:           10.5.1.2
IPs:
  IP:  10.5.1.2
Init Containers:
```

```
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
```

Condition **kubectl describe pods** *pod명*

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