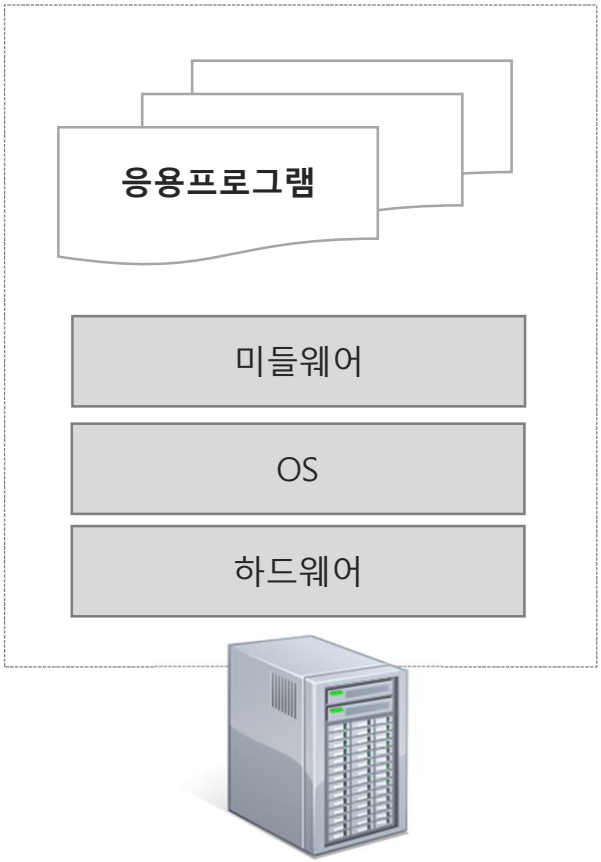
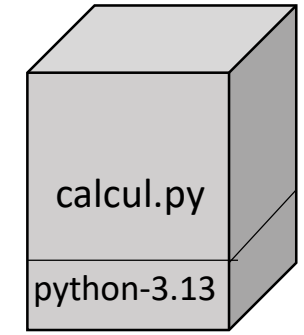
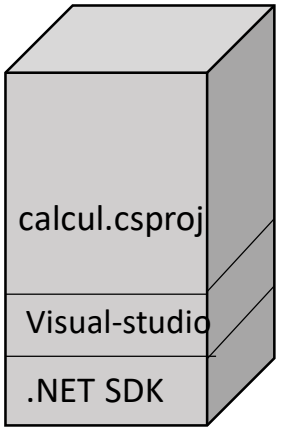
Container & K8S

- 컴퓨터(가상머신)와 컨테이너 차이

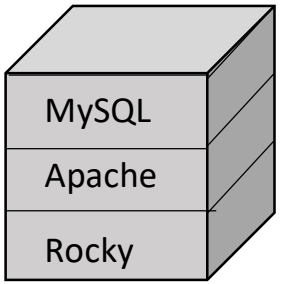




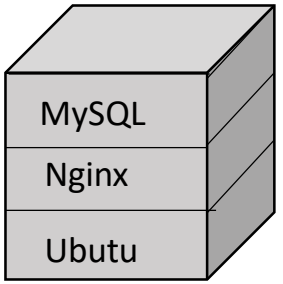
Container



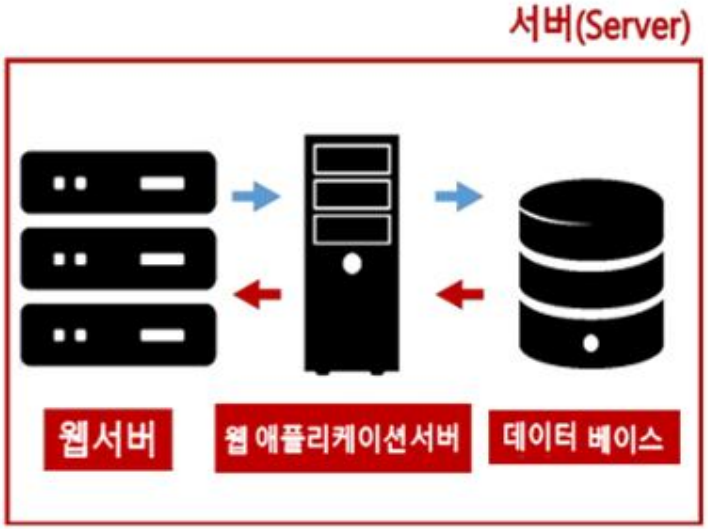
Container

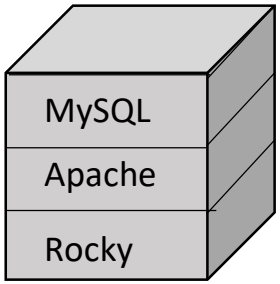


Container

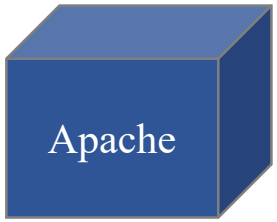


Container

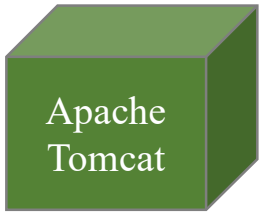




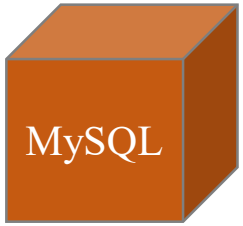
Container



Container

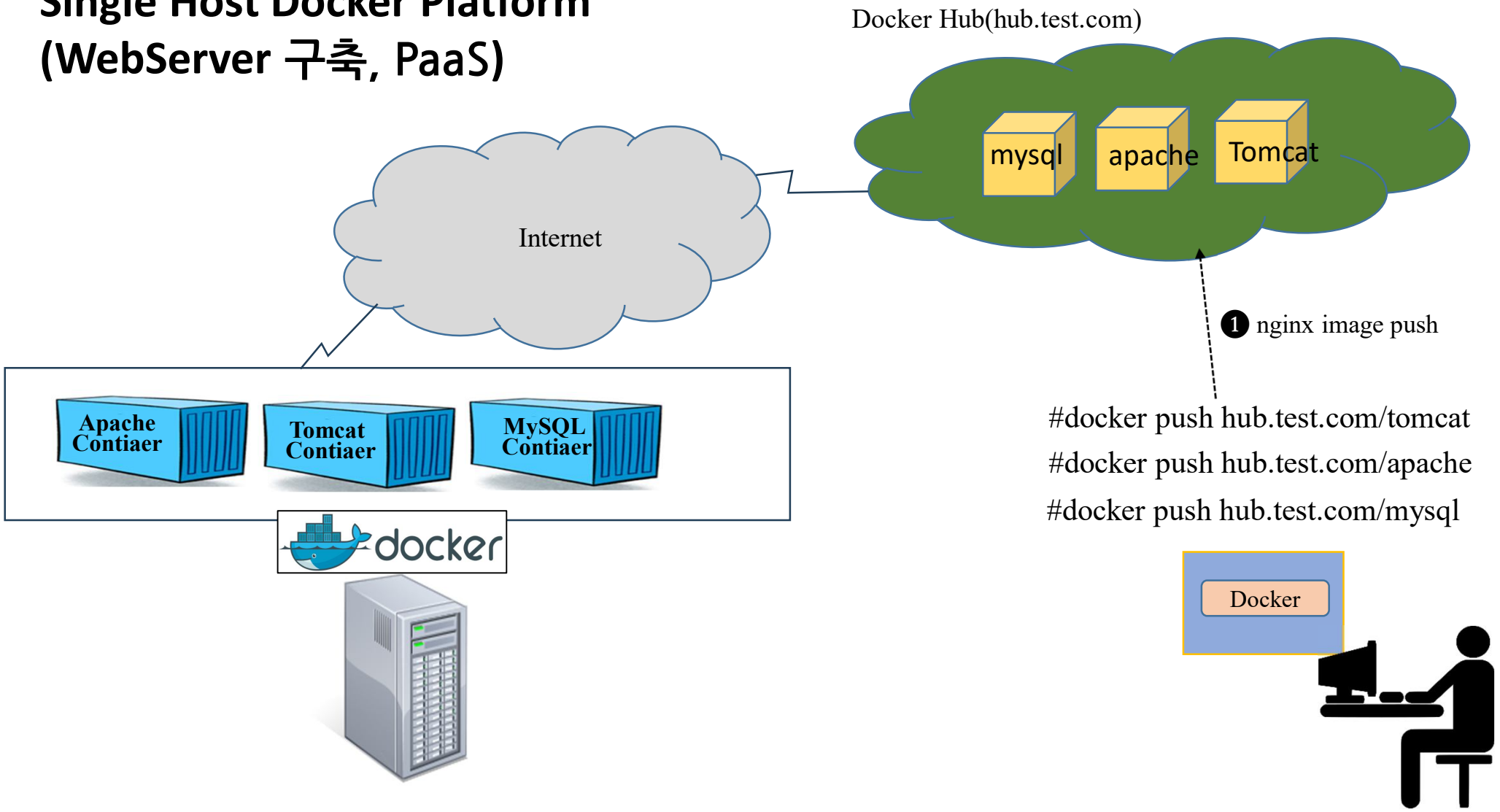


Container

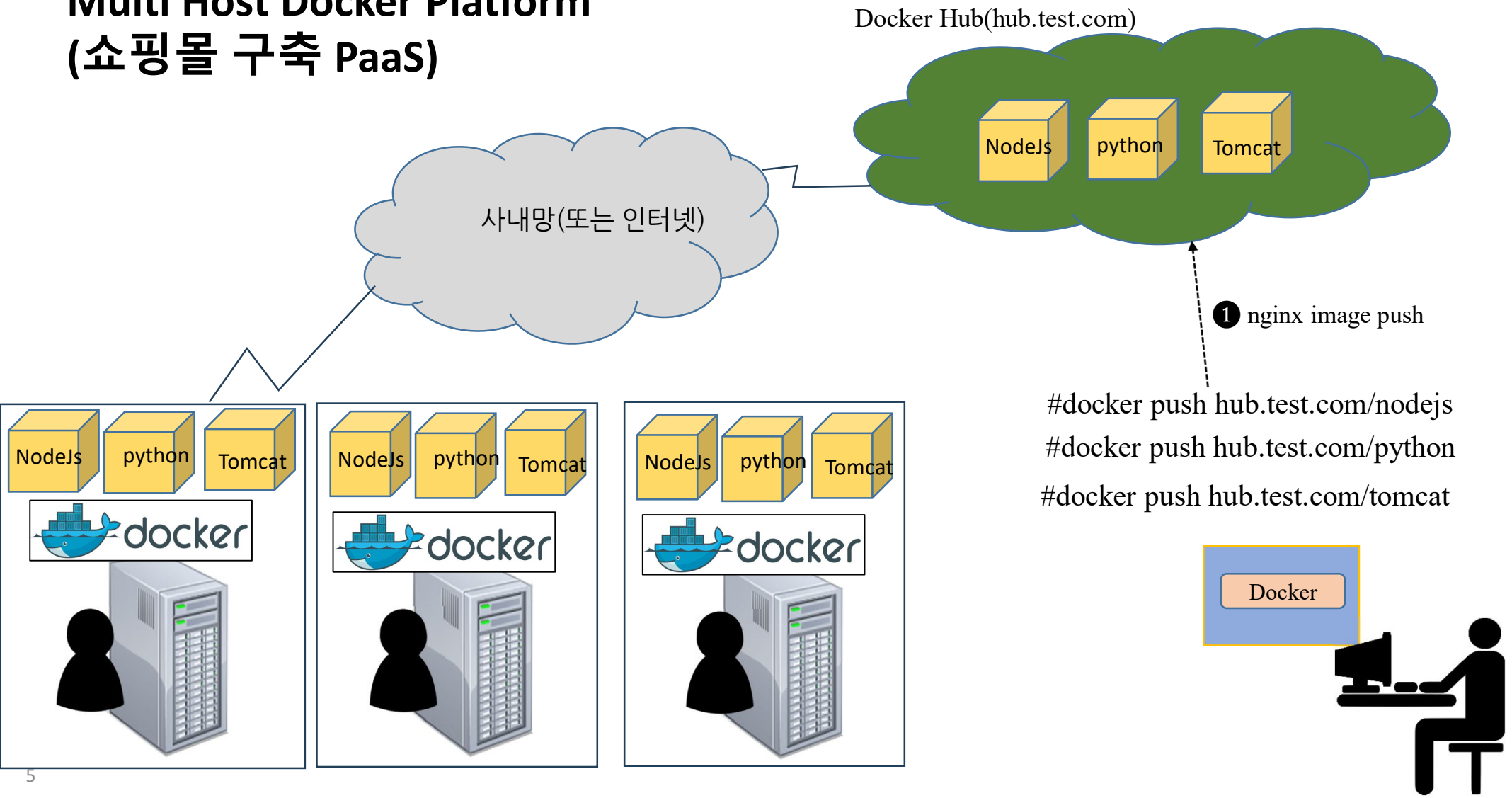


Container

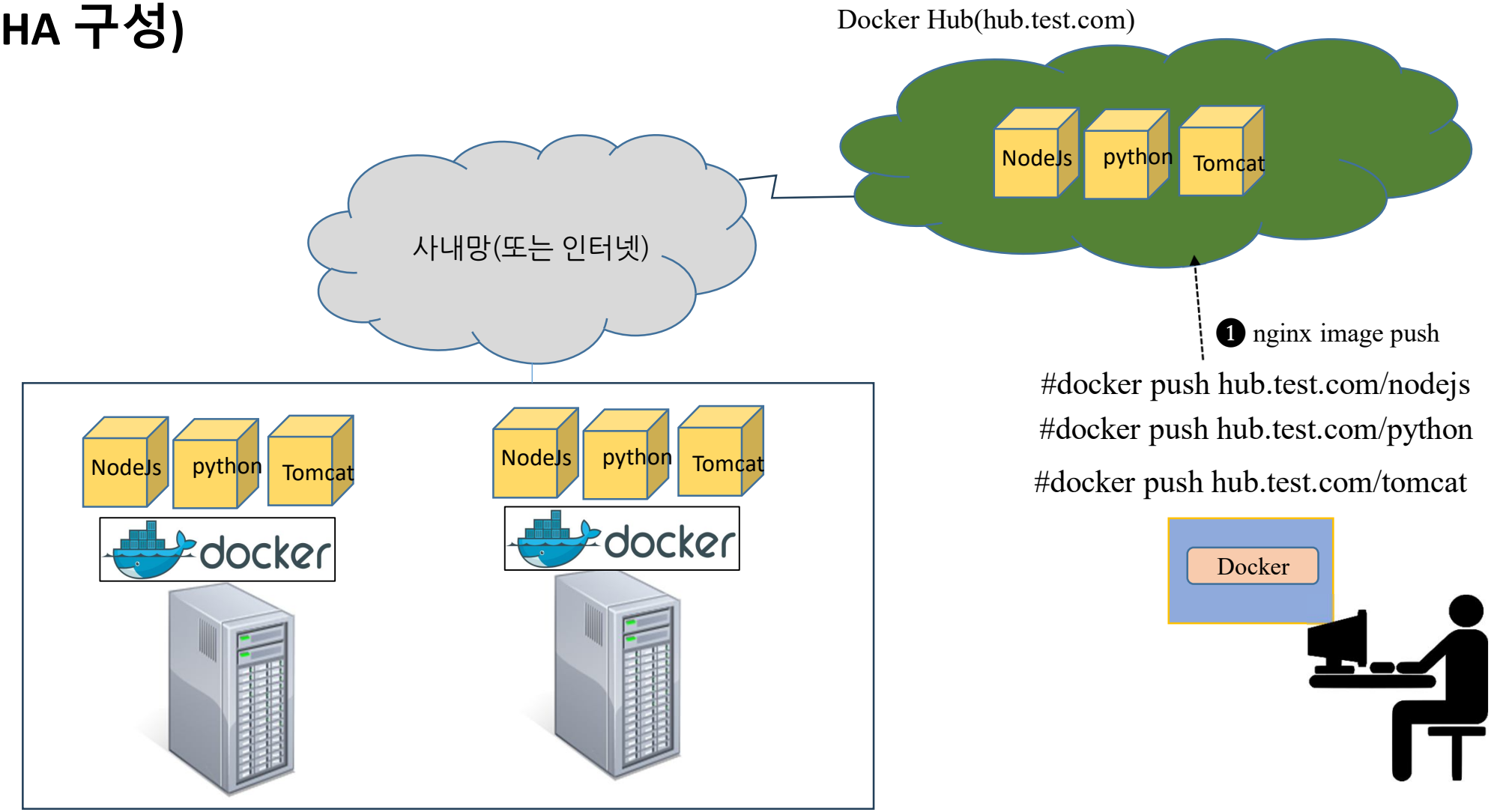
Single Host Docker Platform (WebServer 구축, PaaS)



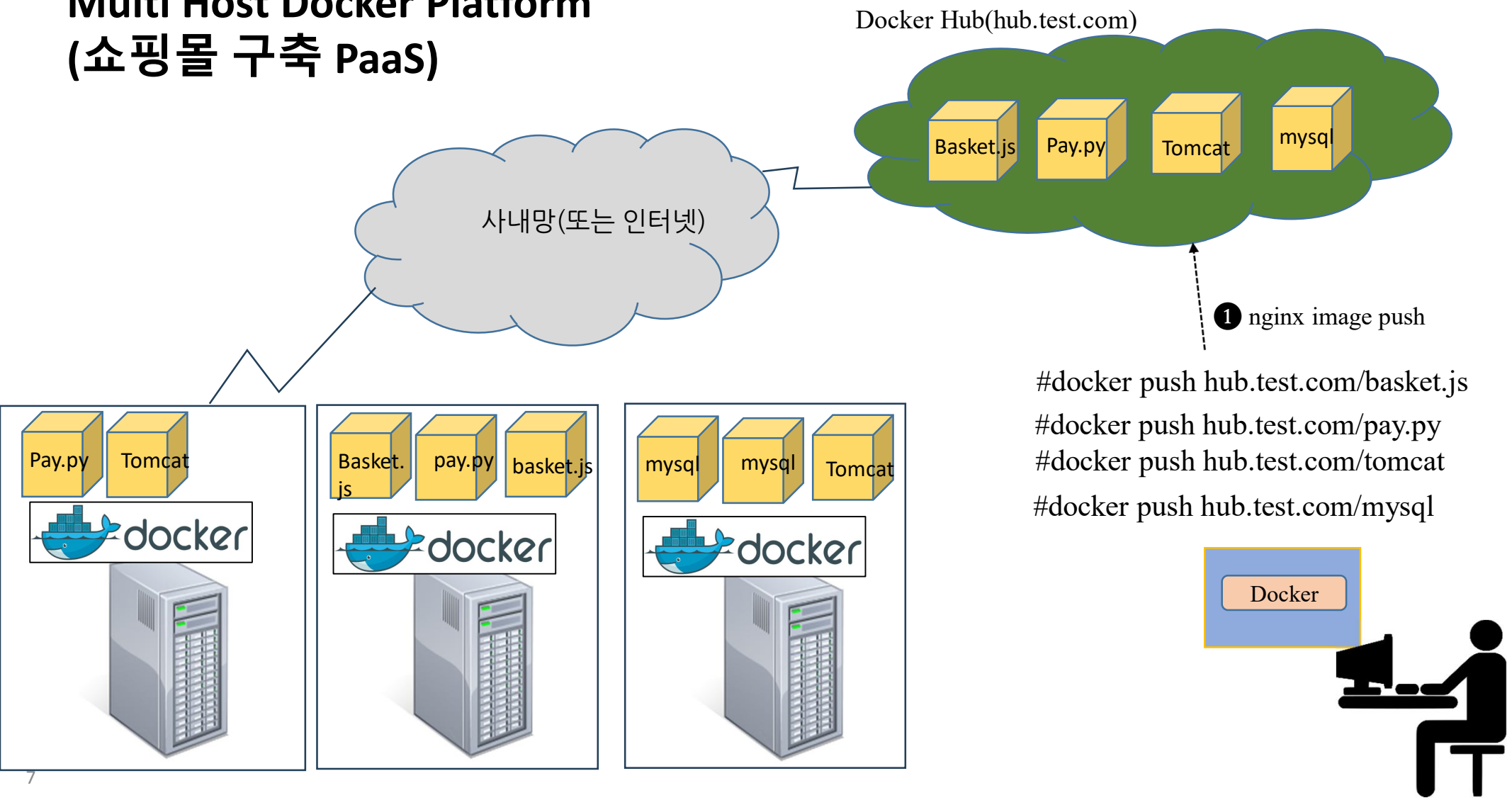
Multi Host Docker Platform
(쇼핑몰 구축 PaaS)



Multi Host Docker Platform
(HA 구성)



Multi Host Docker Platform
(쇼핑몰 구축 PaaS)



Docker & K8S

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

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

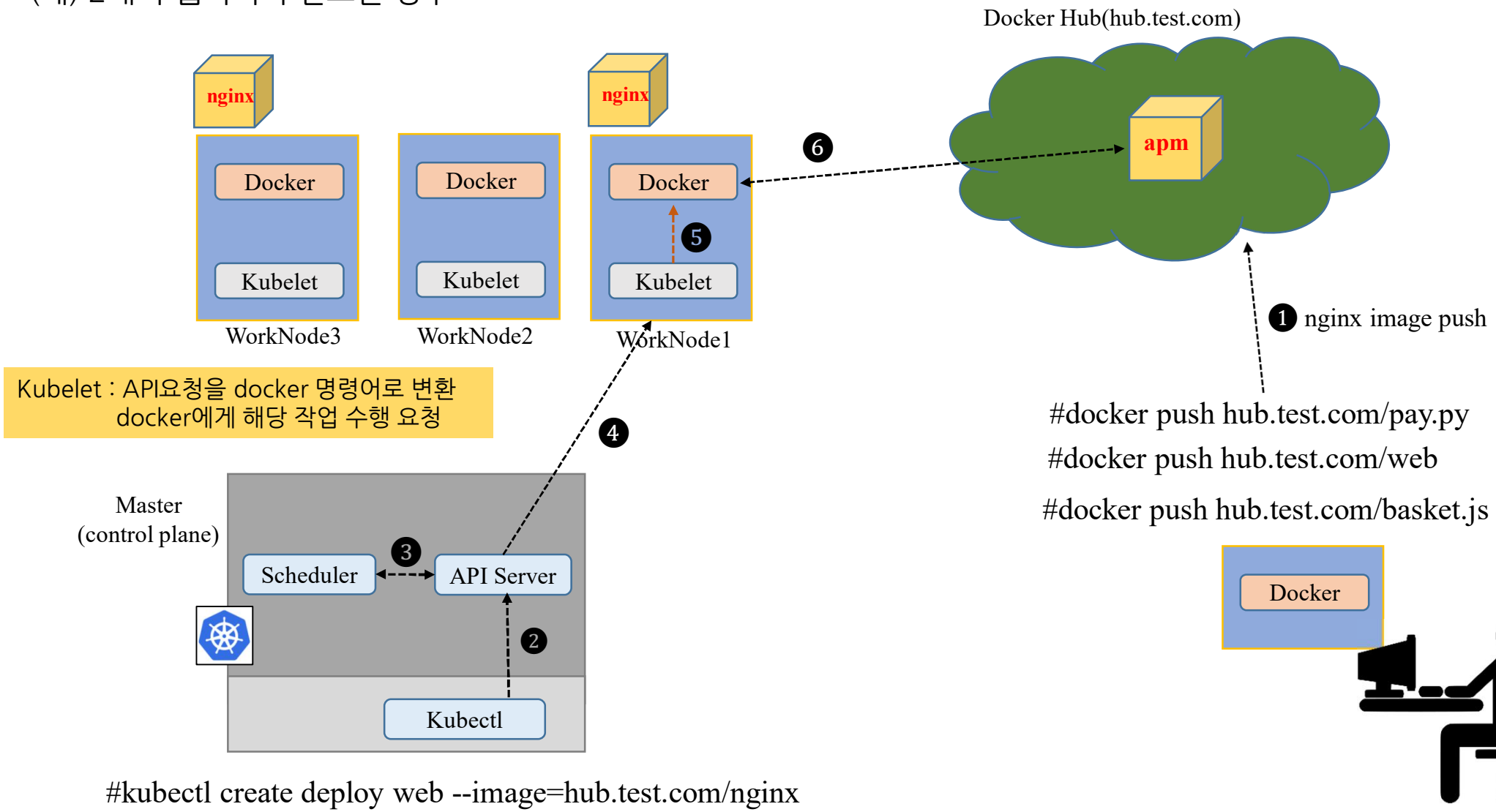
Kubernetes(k8s)

- 배의 조타수란 그리스어에서 유래
- 2014년 구글에서 개발
- Kubernetes== k8s
- **컨테이너 orchestration(오케스트레이션)**
  - 관현악 편곡, 통합, 결집, 편성,조직화, 조정
  - 운영 수준의 컨테이너 오케스트레이션
  - 컨테이너와 호스트의 수가 증가함에 따라 여러 컨테이너의 배포 프로세스를 최적화 시킴
  - 컨테이너화 된 애플리케이션을 자동으로 배포, 스케일링 및 관리해 주는 오픈 소스 시스템

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
- 깃허브, 카카오톡, 다음, 멜로, VISA 등 다수의 기업이 K8S 기반 환경을 구축해서 사용 중

(예) 2대의 웹서버가 필요한 경우



명령어 kubectl

kubectl

- k8s에게 원하는 작업을 요청 시 사용하는 명령어
- k8s cluster를 관리하는 동작은 kubectl이라는 Command line interface로 실행

  - K8s 자원들의 생성, 업데이트, 삭제 (create, update, delete)

  - 디버그, 모니터링, 장애처리(log, exec, cp, top, attach..)

  - 클러스터 관리(cordon, top, drain, traint…)

kubectl --help

```
root@masternode:~# kubectl --help
kubectl controls the Kubernetes cluster manager.

 Find more information at: https://kubernetes.io/docs/reference/kubectl/ove

Basic Commands (Beginner):
  create        Create a resource from a file or from stdin
  expose        Take a replication controller, service, deployment or pod a
  run           Run a particular image on the cluster
  set           Set specific features on objects

Basic Commands (Intermediate):
  explain       Get documentation for a resource
  get           Display one or many resources
  edit          Edit a resource on the server
  delete        Delete resources by file names, stdin, resources and names,
```

kubectl 명령어 형식

kubectl [command] [TYPE] [NAME] [flags]

Command	자원에 실행되는 동작	create, get, delete
TYPE	자원타입	pod, service, ingress
NAME	자원이름	
Flags	부가적으로 설정할 옵션	--help, -o wide

(ex) kubectrl get pod WEBServer –o wide

➔ WEBServer 이름을 가진 Pod 자원정보를 자세히 확인

kubectl --help

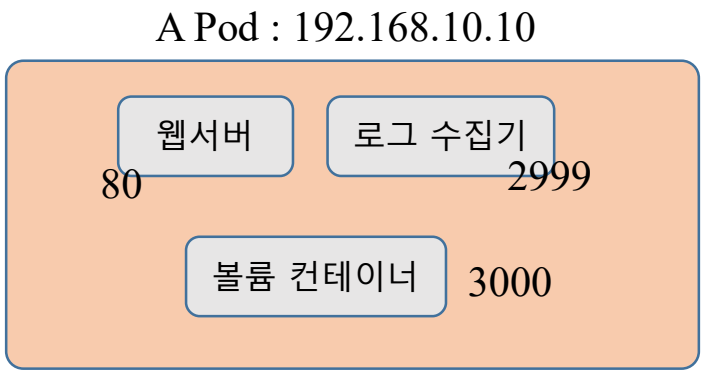
kubuctl run --help

```
root@masternode:~# kubectl run --help
Create and run a particular image in a pod.

Examples:
  # Start a nginx pod
  kubectl run nginx --image=nginx

  # Start a hazelcast pod and let the container expose port 5701
  kubectl run hazelcast --image=hazelcast/hazelcast --port=5701

  # Start a hazelcast pod and set environment variables "DNS_DOMAIN=clus
container
  kubectl run hazelcast --image=hazelcast/hazelcast --env="DNS_DOMAIN=cl

  # Start a hazelcast pod and set labels "app=hazelcast" and "env=prod"
  kubectl run hazelcast --image=hazelcast/hazelcast --labels="app=hazelc

  # Dry run; print the corresponding API objects without creating them
  kubectl run nginx --image=nginx --dry-run=client
```

K8S Pod

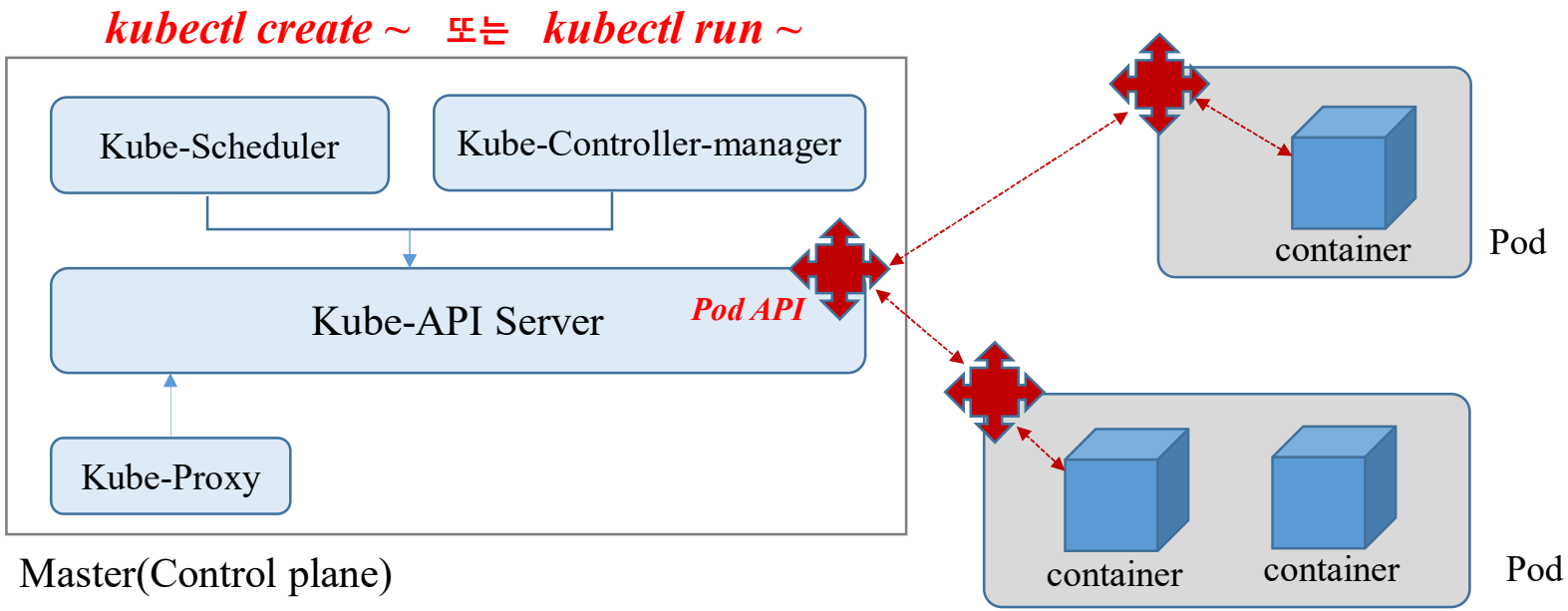
Pod(파드)

- 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

- K8s에는 container를 생성하는 API가 없음
- Pod API를 통해 pod를 생성/동작 시키고 pod를 통해 container를 생성/동작 시킴



K8S에서 Pod 생성 방법 2가지

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

kubectl create ~ 또는 kubectl run ~

1) kubectl run으로 pod 생성

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

```
root@master:/k8s# kubectl run web --image=nginx:1.14 --port 80
pod/web created
root@master:/k8s#
root@master:/k8s# kubectl get pod -o wide
NAME   READY   STATUS    RESTARTS   AGE   IP              NODE       NOMINATED NODE   READINESS GATES
web    1/1     Running   0          45s   192.168.19.70   worker03   <none>           <none>
```

2) YAML 파일을 이용한 pod 생성과 실행

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

  - 마크업 언어가 아니다.

  - 직렬언어임

- YAML 문법

  - 들여쓰기에 따라 구조가 바뀜

  - 들여쓰기는 Tab이 아닌 Space bar 사용 (**공백 문자들로 기본구조를 구성**)

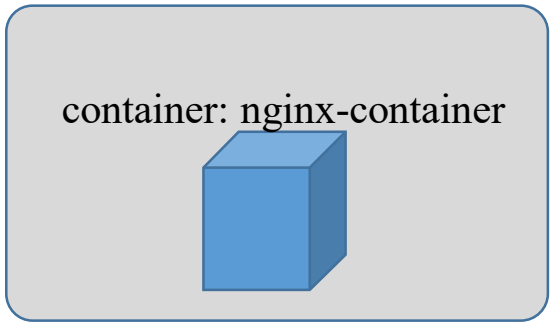
  - #로 주석을 표시

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

    ':' 다음에 공백 문자가 와야한다. (붙여쓰면 오류)

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

실습1. Single Pod



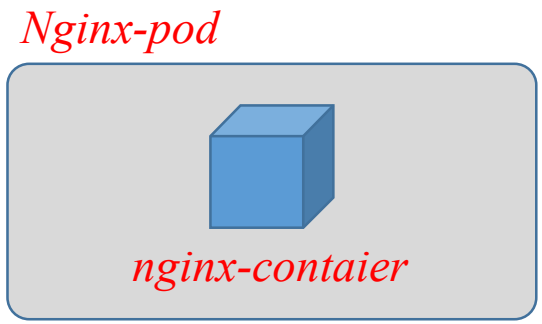
Pod : nginx-pod

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

```
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#
```

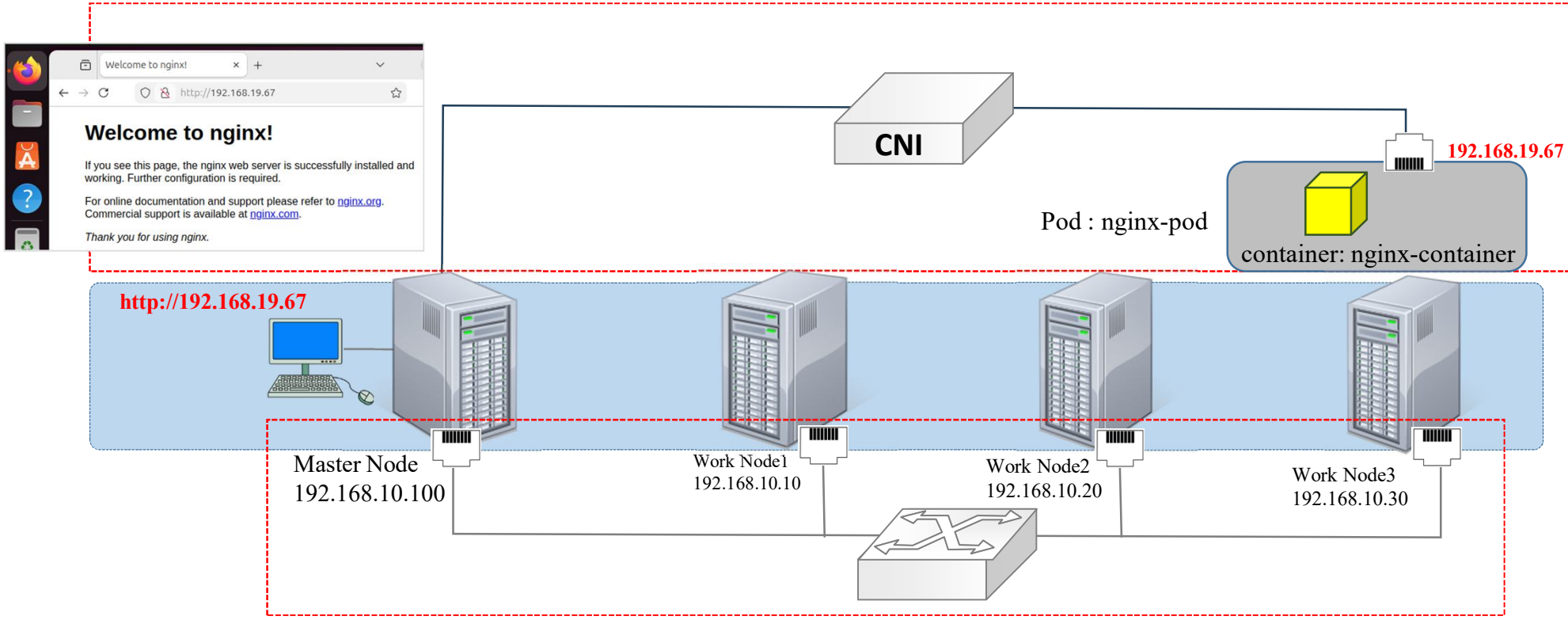
#kubectl describe pod pod-nginx

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

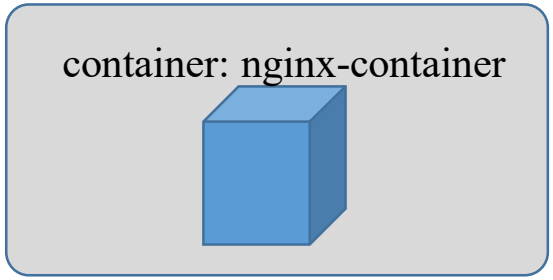


#kubect get pod –o wide

```
root@master:/k8s# kubectl get pod -o wide
NAME        READY   STATUS    RESTARTS   AGE   IP              NODE       NOMINATED NODE   READINESS GATES
nginx-pod   1/1     Running   0          12m   192.168.19.67   worker03   <none>           <none>
```



Pod안 Container 운영체제 확인



Pod : nginx-pod

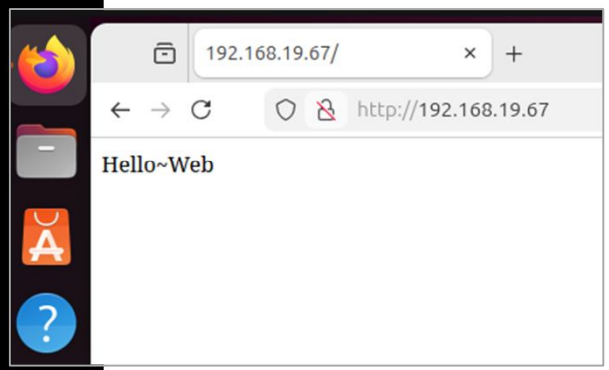
#kubectl exec nginx-pod –c nginx-container -it -- /bin/bash
cat /etc/os-release

```
root@master:/k8s# kubectl exec nginx-pod -c nginx-container -it -- /bin/bash
root@nginx-pod:/# cat /etc/os-release
PRETTY_NAME="Debian GNU/Linux 9 (stretch)"
NAME="Debian GNU/Linux"
VERSION_ID="9"
VERSION="9 (stretch)"
ID=debian
HOME_URL="https://www.debian.org/"
SUPPORT_URL="https://www.debian.org/support"
BUG_REPORT_URL="https://bugs.debian.org/"
root@nginx-pod:/#
```

Pod안 Container 접근 후 내용 수정

#kubectl exec nginx-pod –c nginx-container –it -- /bin/bash
cd /usr/share/nginx/html
echo “Hello~ Web” > index.thml
cat index.html
exit
#curl 192.168.19.67

```
root@master:/k8s# kubectl exec nginx-pod -c nginx-container -it -- /bin/bash
root@nginx-pod:/# cd /usr/share/nginx/html
root@nginx-pod:/usr/share/nginx/html# ls
50x.html  index.html
root@nginx-pod:/usr/share/nginx/html# echo "Hello~Web">index.html
root@nginx-pod:/usr/share/nginx/html# cat index.html
Hello~Web
root@nginx-pod:/usr/share/nginx/html# exit
exit
root@master:/k8s# curl 192.168.19.67
Hello~Web
root@master:/k8s#
```



실습 명령어 정리

❶ Multi pod 생성

nano pod-nginx.yaml

kubectl create –f pod-nginx.yaml

❷ Pod 정보 확인

kubectl get pods

kubectl get pods –o wide

❸ Pod 상세 정보 확인(장애처리)와 수정

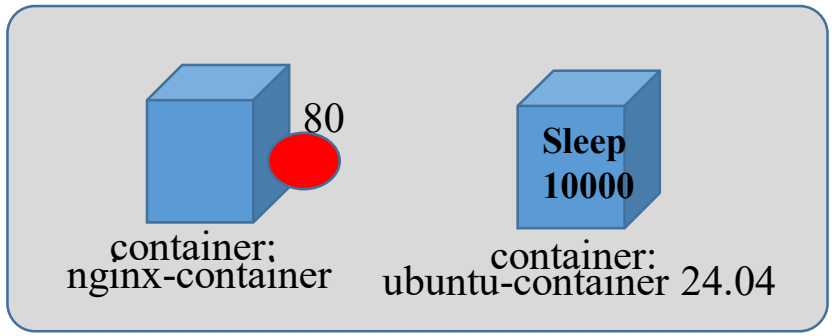
kubectl describe pod pod-nginx

❹ Pod 삭제

kubectl delete pod pod-nginx

kubectl delete pod --all

실습2. Multi-pod



Pod : multipod

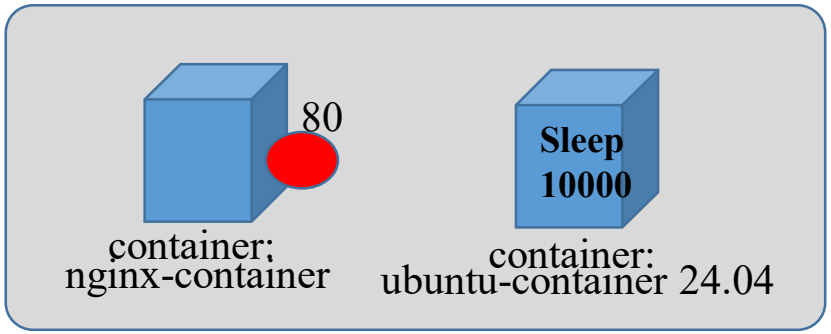
#kubectl create -f pod-multi.yaml
#kubect get pods –o wide

```
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: ubuntu-container
    image: ubuntu:24.04
    command:
    - sleep
    - "10000"
```

#kubectl describe pod multipod

watch kubectl get pod –o wide

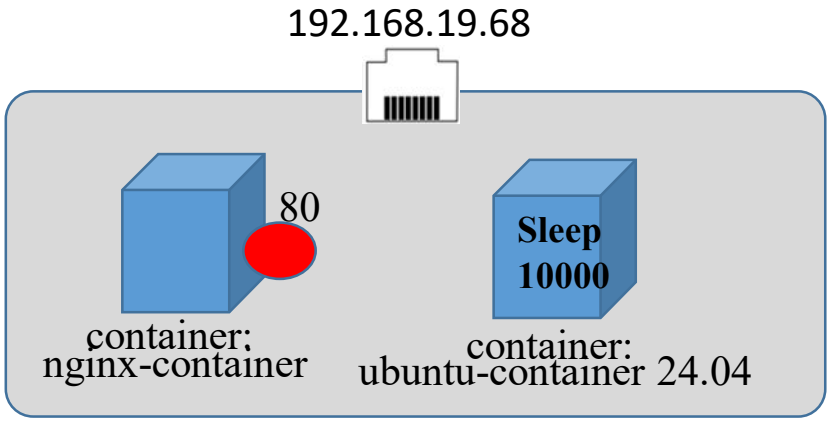
```
NAME       READY   STATUS              RESTARTS   AGE   IP       NODE       NOMINATED NODE   READINESS GATES
multipod   0/2     ContainerCreating   0          8s    <none>   worker03   <none>           <none>
```



kubectl get pod –o wide

```
root@master:/k8s# kubectl get pod -o wide
NAME       READY   STATUS    RESTARTS   AGE   IP              NODE       NOMINATED NODE   READINESS GATES
multipod   2/2     Running   0          12m   192.168.19.68   worker03   <none>           <none>
```

kubectl describe pod multipod



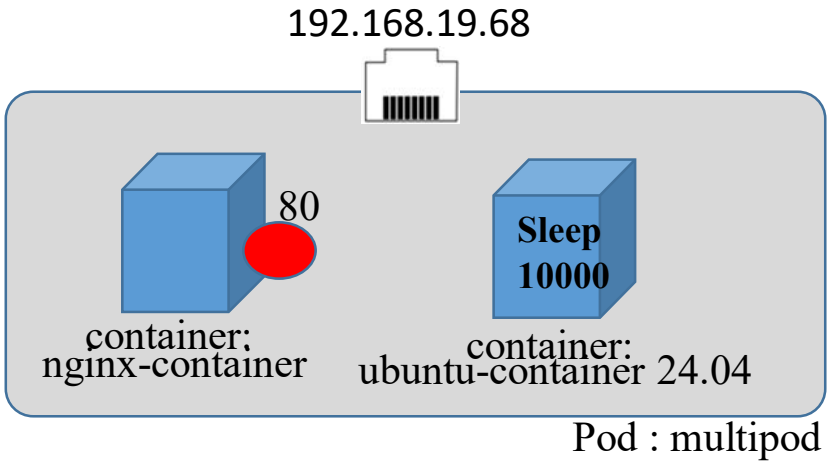
Pod : multipod

```
Node:             worker03/192.168.10.40
Start Time:       Tue, 13 Jan 2026 12:53:49 +0900
Labels:           <none>
Annotations:      cni.projectcalico.org/containerID: 9da97d6cdf1273d7e8
                  cni.projectcalico.org/podIP: 192.168.19.68/32
                  cni.projectcalico.org/podIPs: 192.168.19.68/32
Status:           Running
IP:               192.168.19.68
IPs:
  IP:  192.168.19.68
Containers:
  nginx-container:
    Container ID:   containerd://401fb51084fa56f9b0575d798f5e9b077f7cf2
    Image:          nginx:1.14
    Image ID:       docker.io/library/nginx@sha256:f7988fb6c02e0ce69257
    Port:           80/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Tue, 13 Jan 2026 12:53:50 +0900
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-acces
  ubuntu-container:
    Container ID:  containerd://f7c6c35d8dbb421f7cc801131b1b30797d8e74a
    Image:         ubuntu:24.04
    Image ID:      docker.io/library/ubuntu@sha256:c35e29c9450151419d94
```

kubectl describe pod multipod

```
Events:
  Type    Reason     Age    From               Message
  ----    ------     ----   ----               -------
  Normal  Scheduled  5m13s  default-scheduler  Successfully assigned default/multipod to worker03
  Normal  Pulled     5m12s  kubelet            Container image "nginx:1.14" already present on machine
  Normal  Created    5m12s  kubelet            Created container: nginx-container
  Normal  Started    5m12s  kubelet            Started container nginx-container
  Normal  Pulling    5m12s  kubelet            Pulling image "ubuntu:24.04"
  Normal  Pulled     5m     kubelet            Successfully pulled image "ubuntu:24.04" in 11.657s (11.658s
  Normal  Created    5m     kubelet            Created container: ubuntu-container
  Normal  Started    5m     kubelet            Started container ubuntu-container
```

Pod안 Container 운영체제 확인

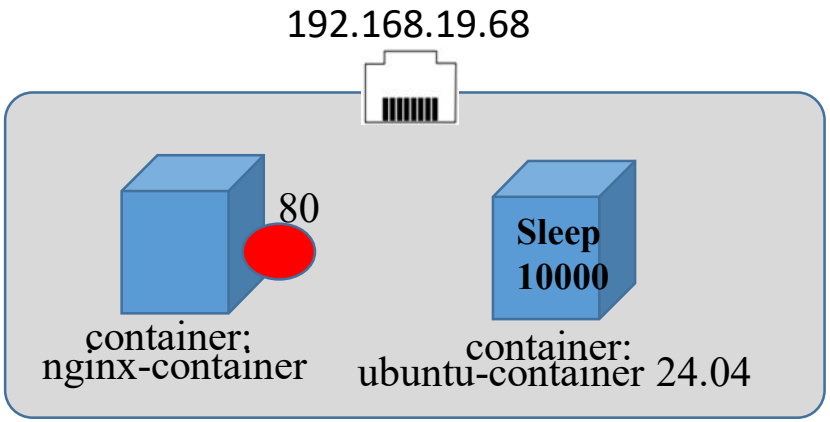


#kubectl exec multipod –c ubuntu-container -it -- /bin/bash
#cat /etc/os-release

```
root@master:/k8s# kubectl exec multipod -c ubuntu-container -it -- /bin/bash
root@multipod:/# cat /etc/os-release
PRETTY_NAME="Ubuntu 24.04.3 LTS"
NAME="Ubuntu"
VERSION_ID="24.04"
VERSION="24.04.3 LTS (Noble Numbat)"
VERSION_CODENAME=noble
ID=ubuntu
ID_LIKE=debian
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"
UBUNTU_CODENAME=noble
LOGO=ubuntu-logo
root@multipod:/#
```

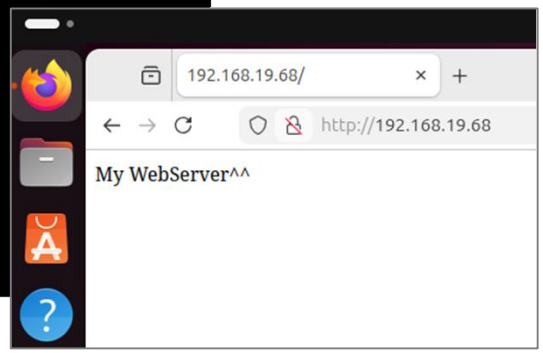
Pod안 Container 접근 후 내용 수정

#kubectl exec multipod –c nginx-container –it -- /bin/bash
#cd /usr/share/nginx/html
#echo “Hello~ Web” > index.thml
#cat index.html
#exit
#curl 192.168.19.68



Pod : multipod

```
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 "My WebServer^^">index.html
root@multipod:/usr/share/nginx/html# cat index.html
My WebServer^^
root@multipod:/usr/share/nginx/html# exit
exit
root@master:/k8s# curl 192.168.19.68
My WebServer^^
root@master:/k8s#
```



kubectl delete pods multiple 또는 kubectl delete pods --all

```
Every 2.0s: kubectl get pod -o wide

NAME       READY   STATUS        RESTARTS   AGE   IP              NODE       NOMINATED NODE   READINESS GATES
multipod   2/2     Terminating   0          27m   192.168.19.68   worker03   <none>           <none>
```

```
1 master × +
root@master:/k8s# kubectl delete pods multipod
pod "multipod" deleted
```

```
Every 2.0s: kubectl get pod -o wide

No resources found in default namespace.
```

```
1 master × +
root@master:/k8s# kubectl delete pods multipod
pod "multipod" deleted
root@master:/k8s#
```

실습 명령어 정리

❶ Multi pod 생성

vi pod-multi.yaml

kubectl create –f pod-multi.yaml

❷ Pod 정보 확인

kubectl get pods

kubectl get pods –o wide

❸ Pod 상세 정보 확인(장애처리)와 수정

kubectl describe pod multipod

kubectl edit pod multipod

❹ Pod 삭제

kubectl delete pod multipod

kubectl delete pod --all