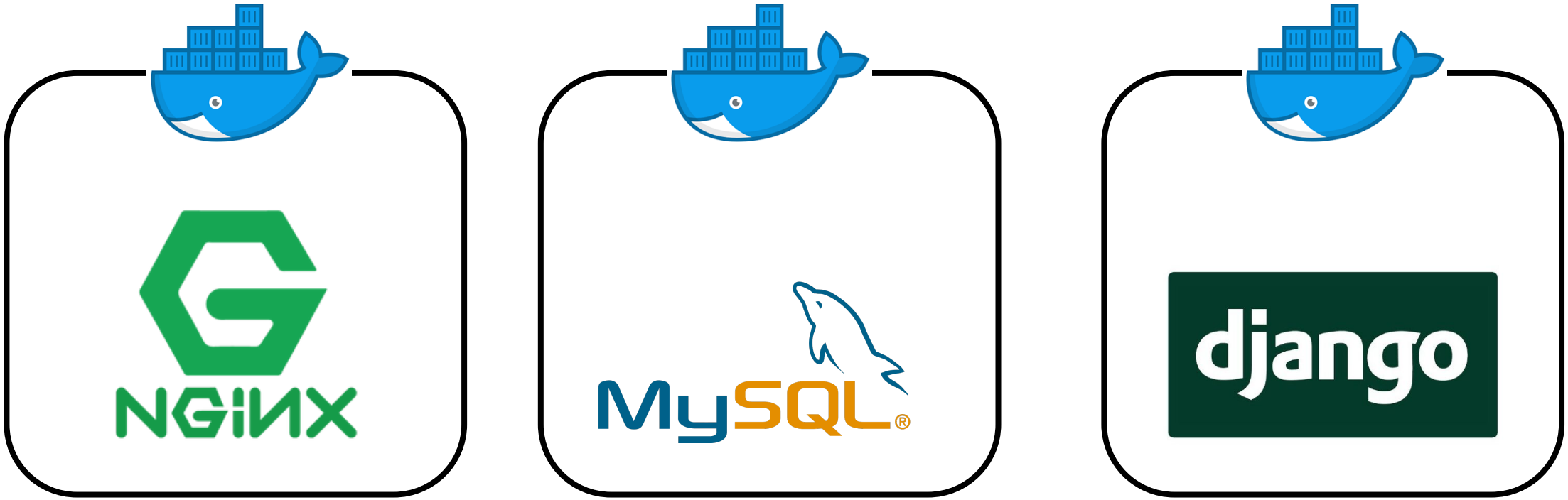
MLops를 위한 Kubernetes 기초

kubernetes

하나의 서비스를 만드는데 필요한 컨테이너는?

간단한 웹 서비스를 만든다면?



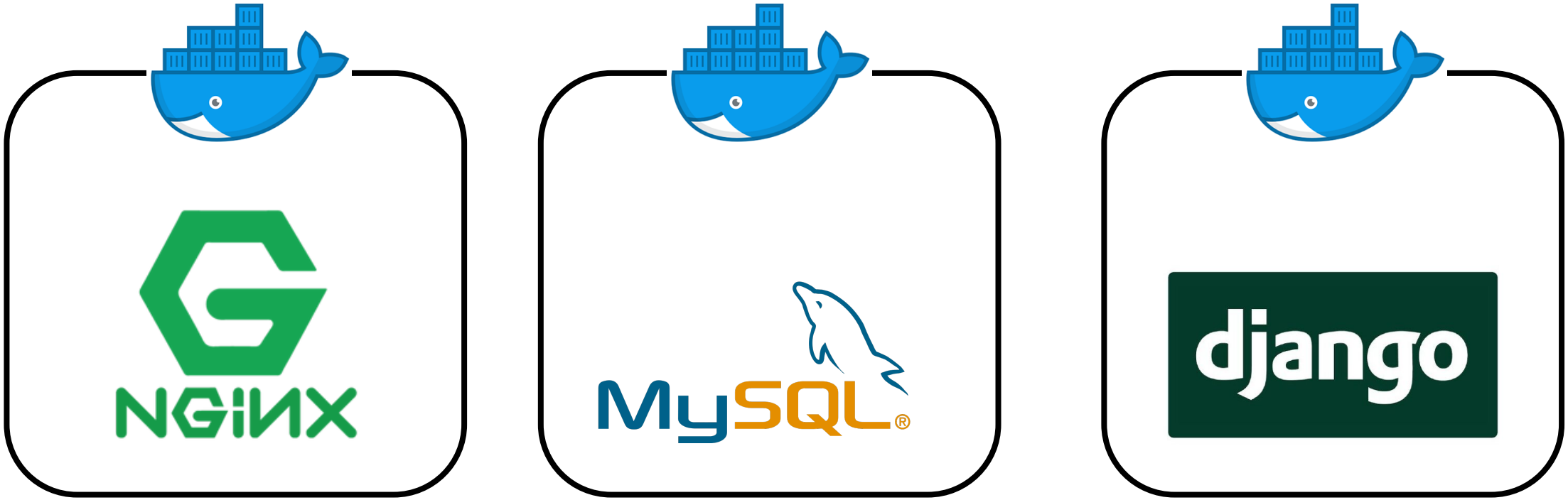
docker run --name mysql –p 3306:3306 mysql:latest

docker run --name myapp –p 8080:80 myapp:django

docker run --name nginx –p 80:80 nginx:latest

하나의 서비스를 만드는데 필요한 컨테이너는?

간단한 웹 서비스를 만든다면?



docker run --name mysql –p 3306:3306 mysql:latest

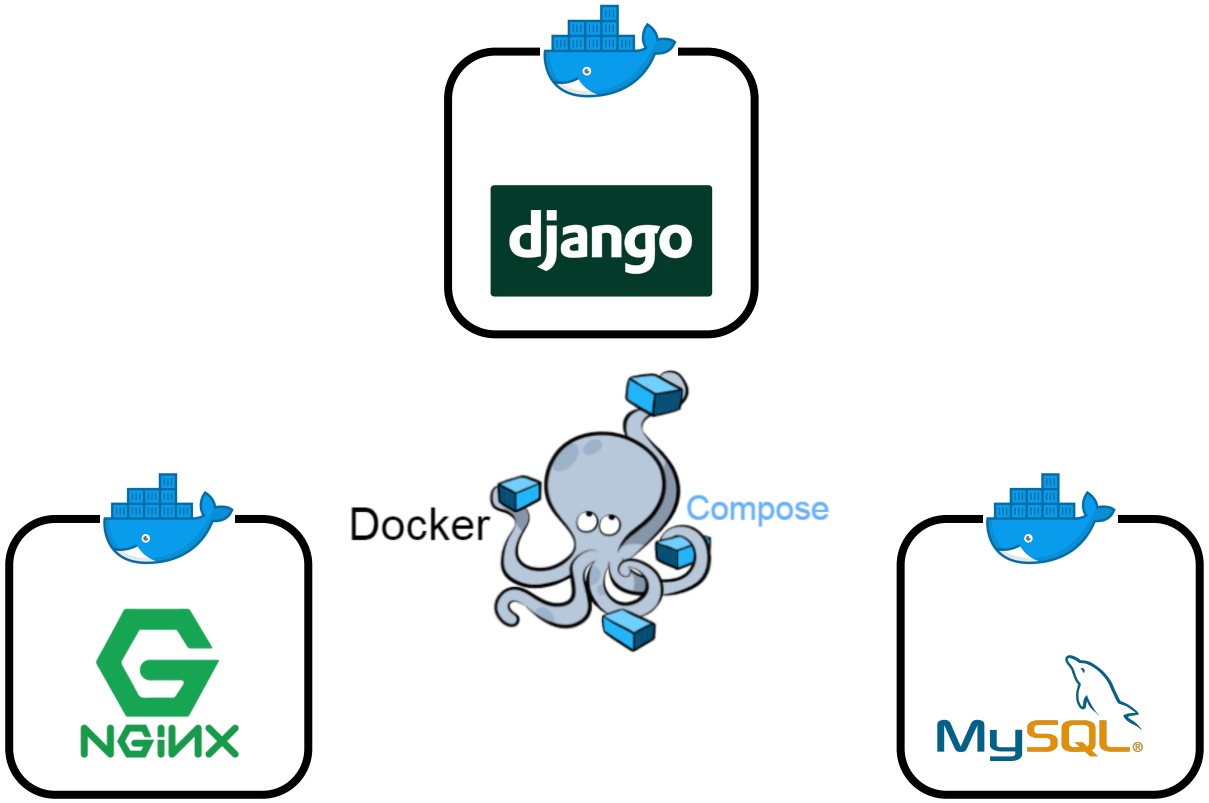
docker run --name myapp –p 8080:80 myapp:django

docker run --name nginx –p 80:80 nginx:latest

뭔가 귀찮다
순서도 지켜야 한다.

하나의 서비스를 만드는데 필요한 컨테이너는?

간단한 웹 서비스를 만든다면?



docker-compose up

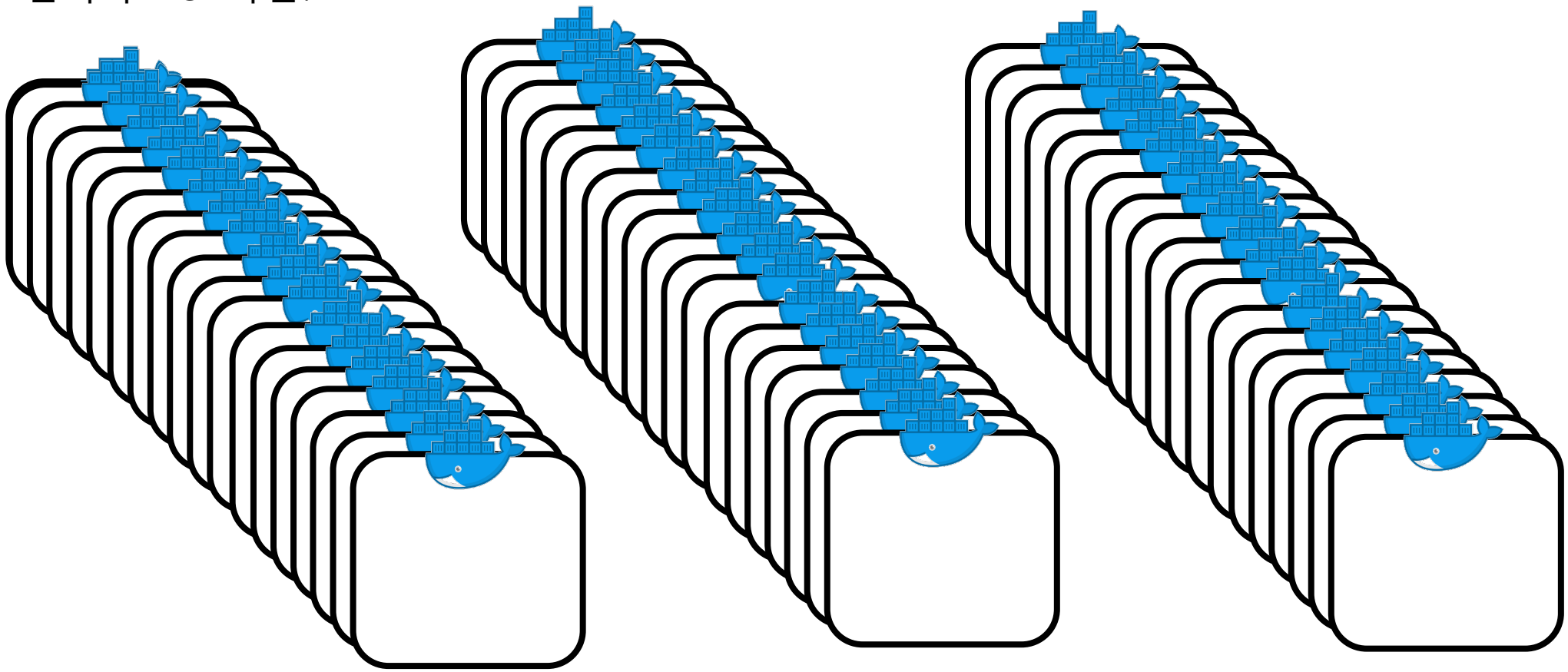
```
python:
    build: ./python
    command: uwsgi --socket :8001 --module app.wsgi --py-autoreload 1 --logto /tmp/mylog.log
    volumes:
      - ./src:/code
      - ./static:/static
    expose:
      - "8001"
    depends_on:
      - db

db:
    image: mysql:5.7
    command: mysqld --character-set-server=utf8mb4 --collation-server=utf8mb4_unicode_ci
    ports:
      - "3306:3306"
    environment:
      MYSQL_ROOT_PASSWORD: root
      MYSQL_DATABASE: todoList
      MYSQL_USER: user
      MYSQL_PASSWORD: password
      TZ: 'Asia/Tokyo'
    volumes:
      - ./mysql:/var/lib/mysql
      - ./sql:/docker-entrypoint-initdb.d

nginx:
    image: nginx:1.13
    ports:
      - "8000:8000"
    volumes:
      - ./nginx/conf:/etc/nginx/conf.d
      - ./nginx/uwsgi_params:/etc/nginx/uwsgi_params
      - ./static:/static
    depends_on:
      - python
```

하나의 서비스를 만드는데 필요한 컨테이너는?

좀 더 복잡한 시스템이라면?

하나의 서비스를 만드는데 필요한 컨테이너는?

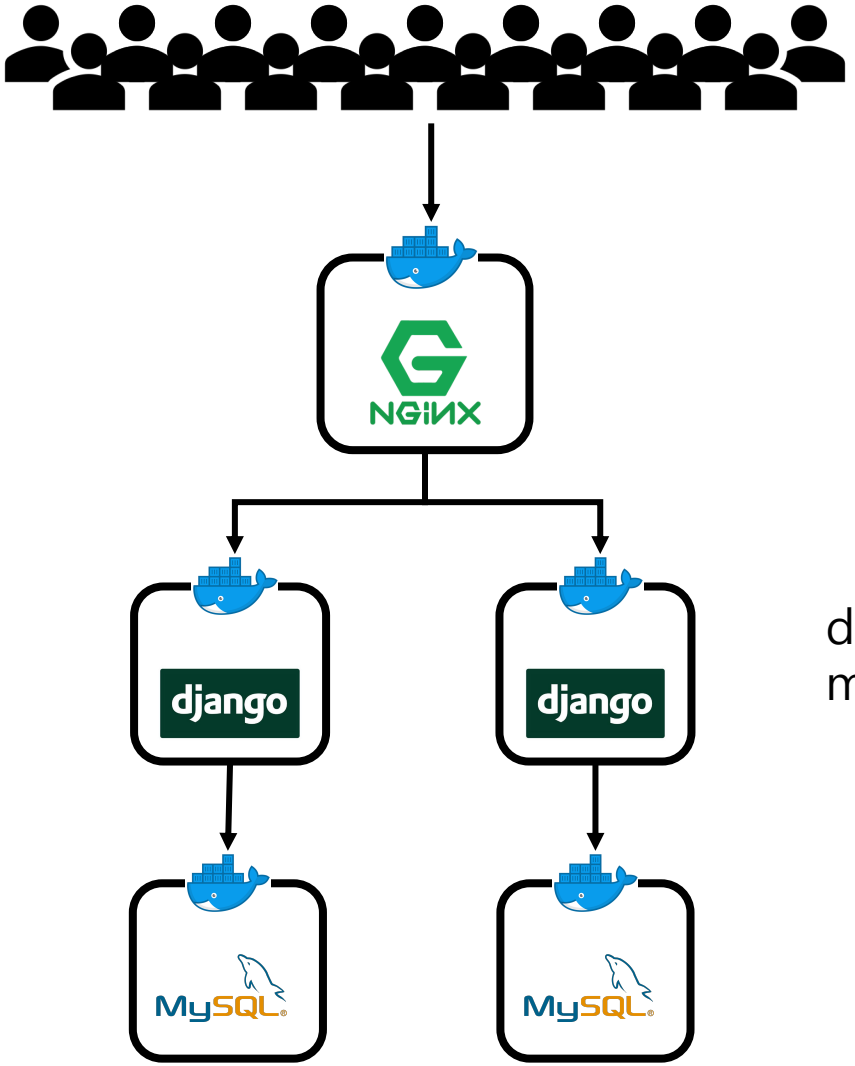
심지어 MSA라면?

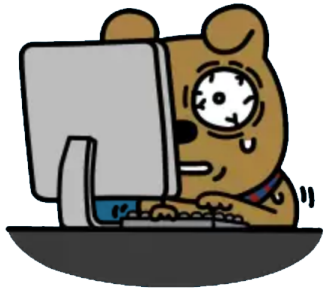


Google은 1주에 약 1억 개 정도의 컨테이너를 띄운다고 합니다.

컨테이너의 관리

부하가 많이 생기면?



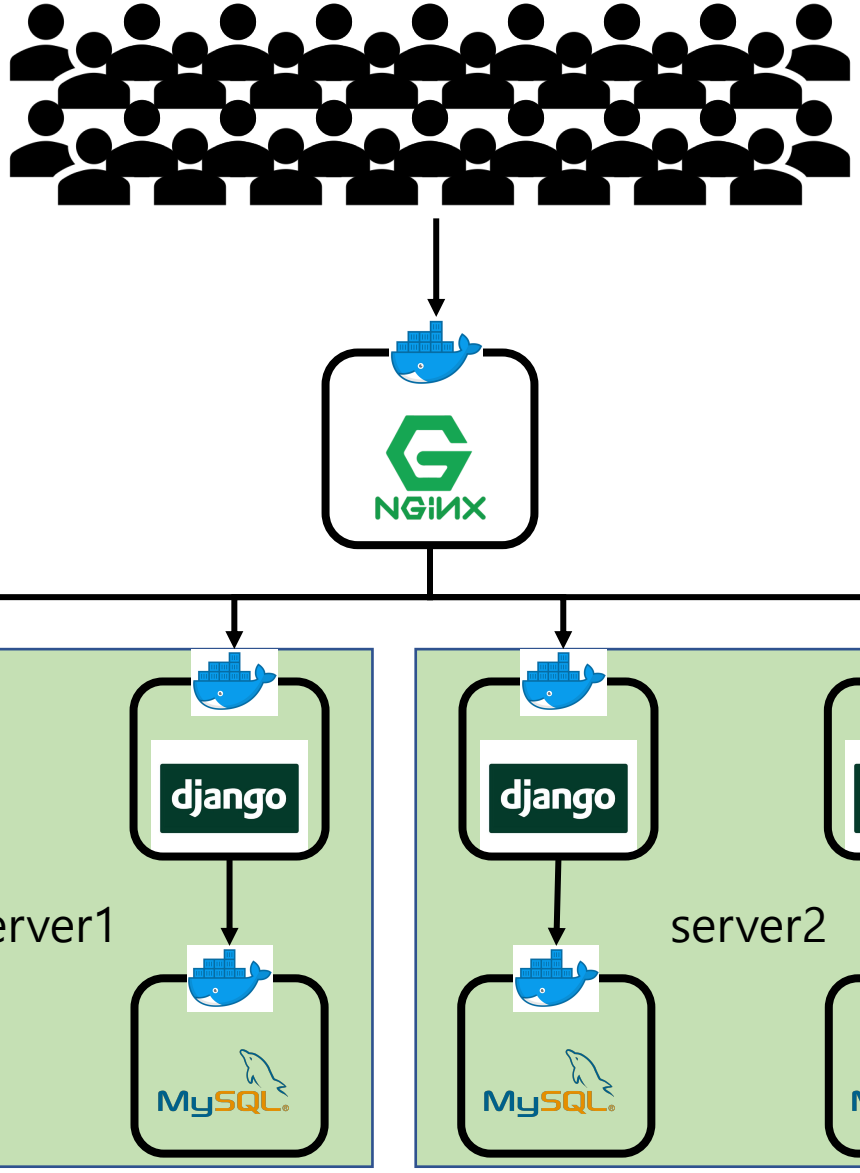


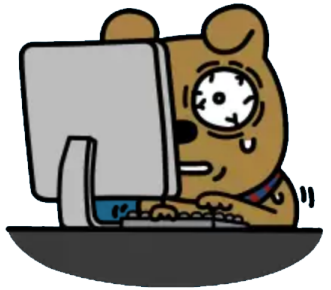
developer 또는 operation team

docker-compose up --scale myserver=2,db=2

컨테이너의 관리

부하가 많이 생기면?





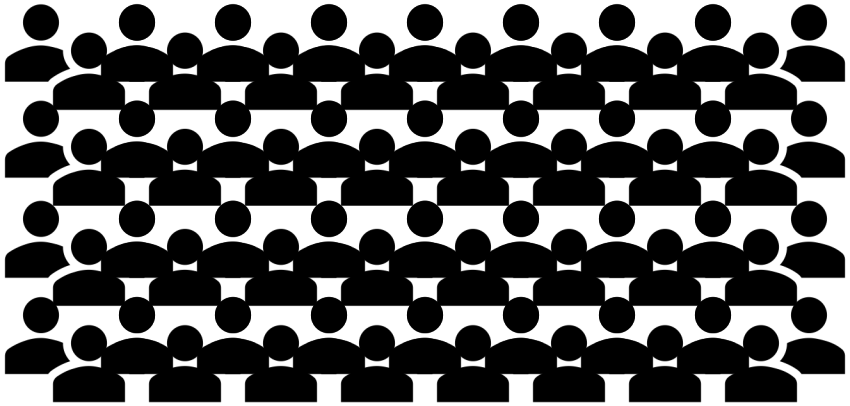
developer 또는 operation team

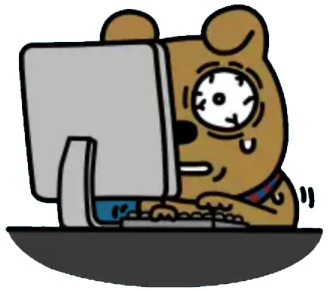
docker-compose up --scale
myserver=2,db=2

docker-compose up --scale
myserver=2,db=2

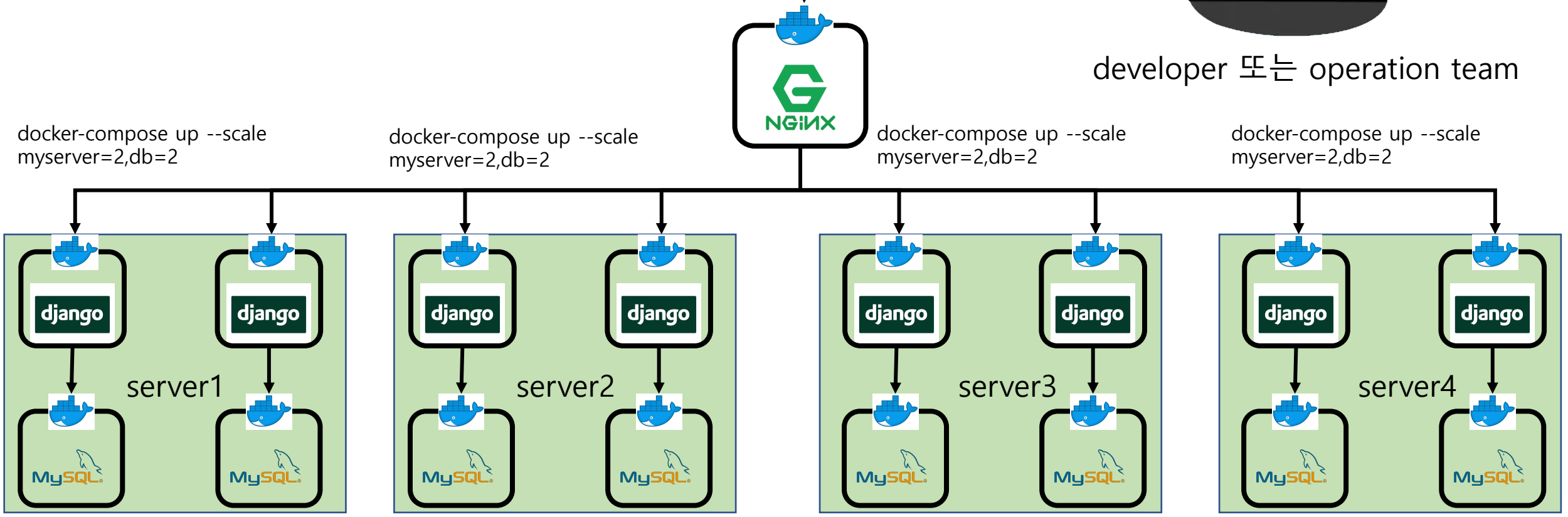
컨테이너의 관리

부하가 많이 생기면?



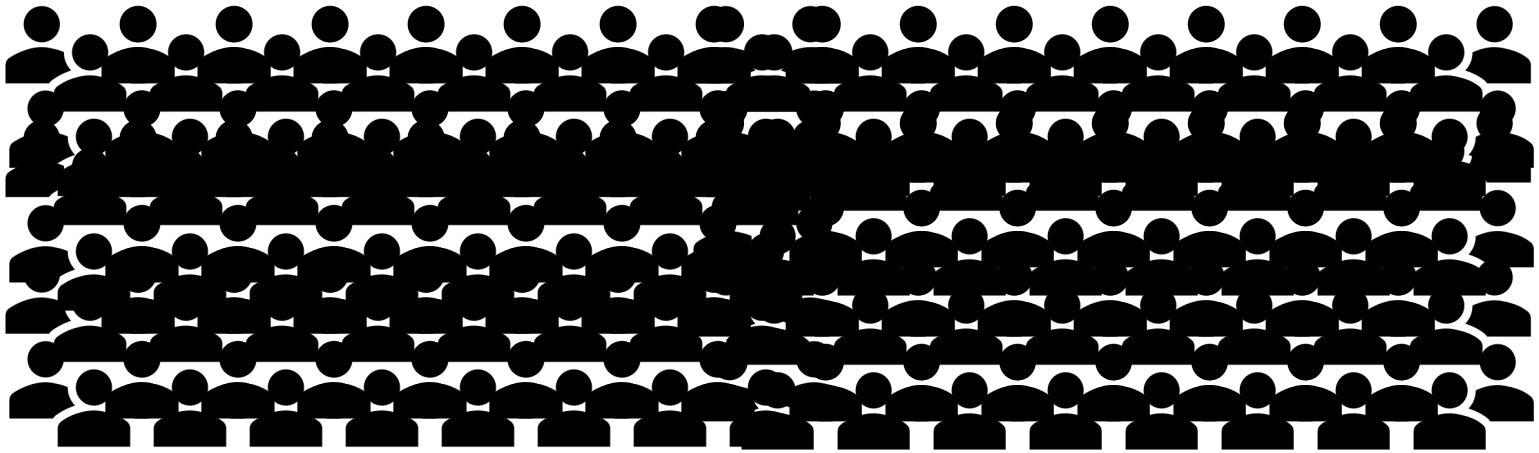


developer 또는 operation team



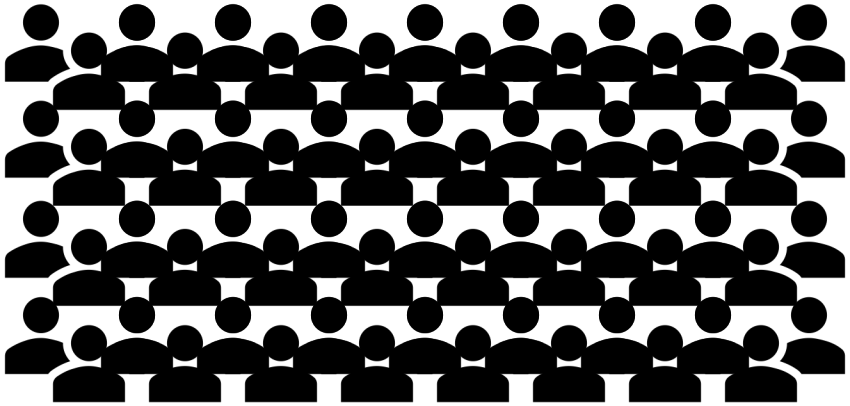
컨테이너의 관리

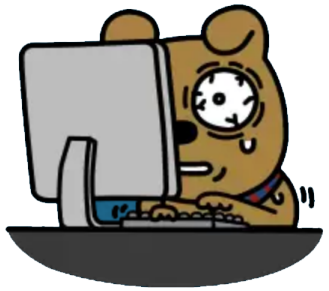
더 많아지면?



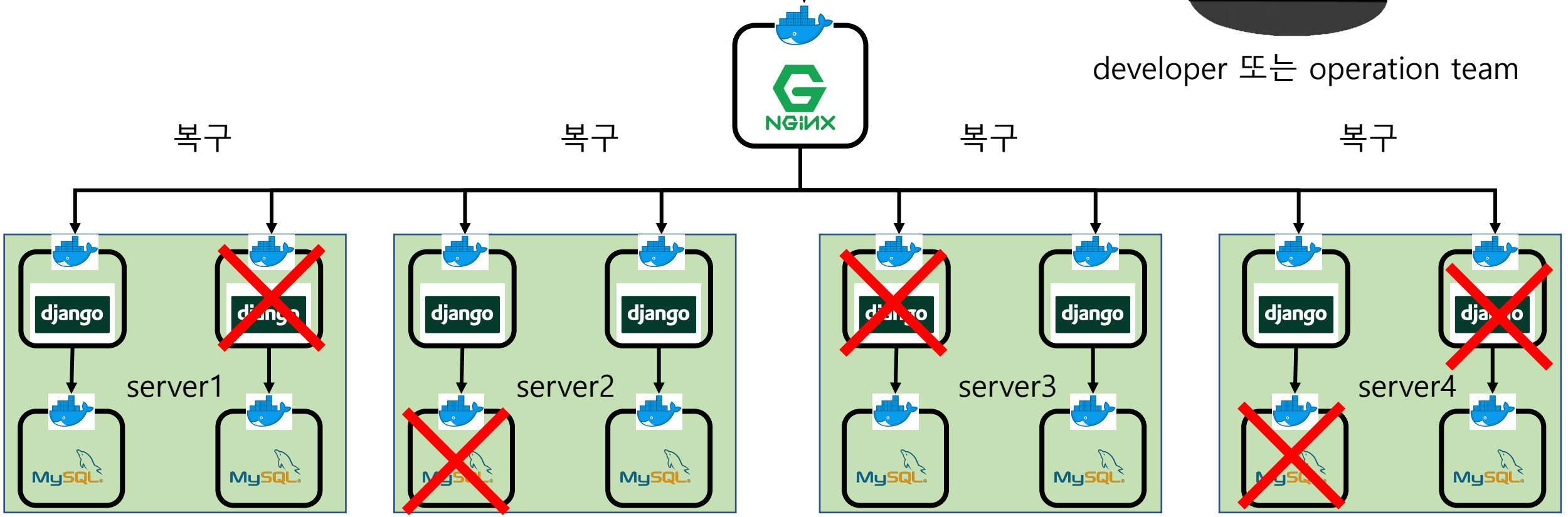
컨테이너의 관리

컨테이너가 망가지면?





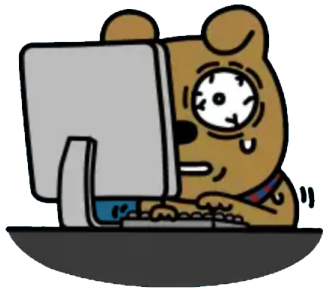
developer 또는 operation team



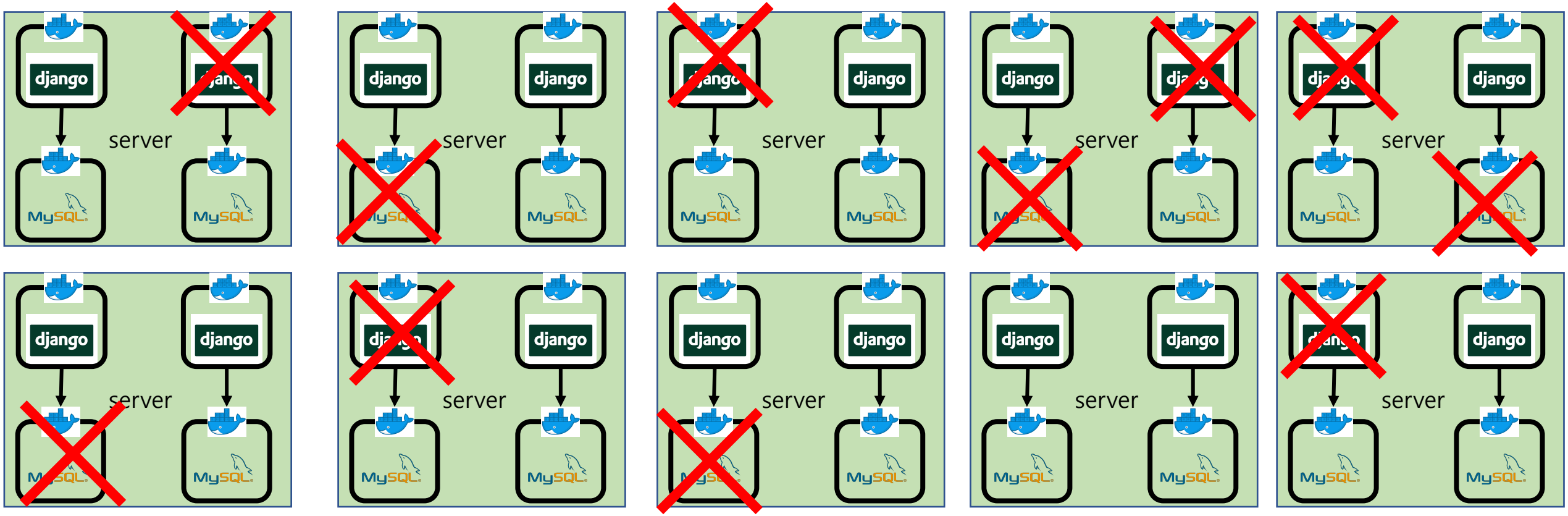
컨테이너의 관리

컨테이너가 더 많으면?

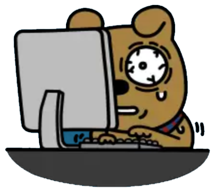
이거 일일히 다 복구하라구요?



developer 또는 operation team

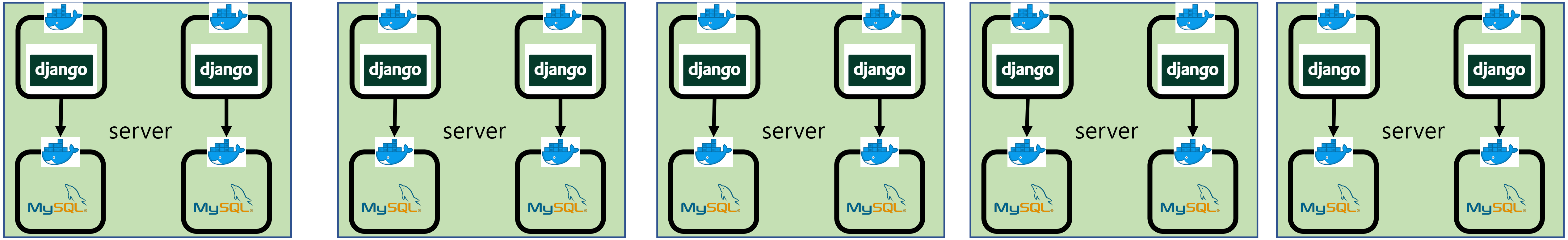


컨테이너의 관리

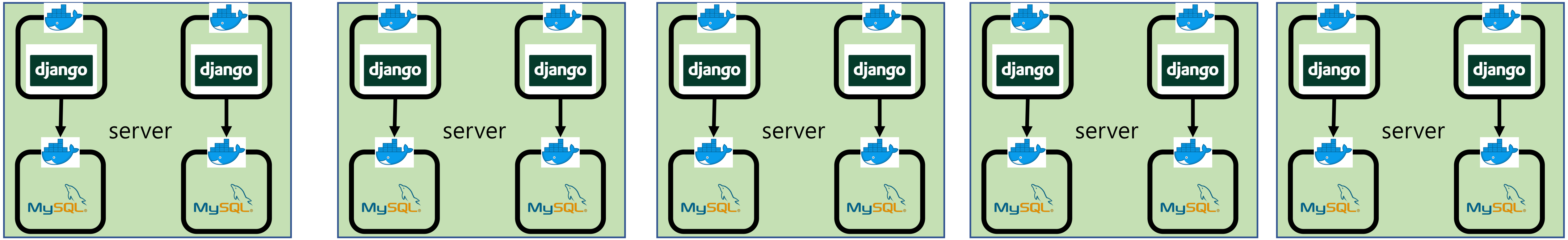


서버가 업데이트되면?

developer 또는 operation team



내리고 -> 이미지 pull -> 올리고 내리고 -> 이미지 pull -> 올리고 내리고 -> 이미지 pull -> 올리고 내리고 -> 이미지 pull -> 올리고 내리고 -> 이미지 pull -> 올리고



나는 누구 여긴 어디 퇴사만이 답인가 아아흑

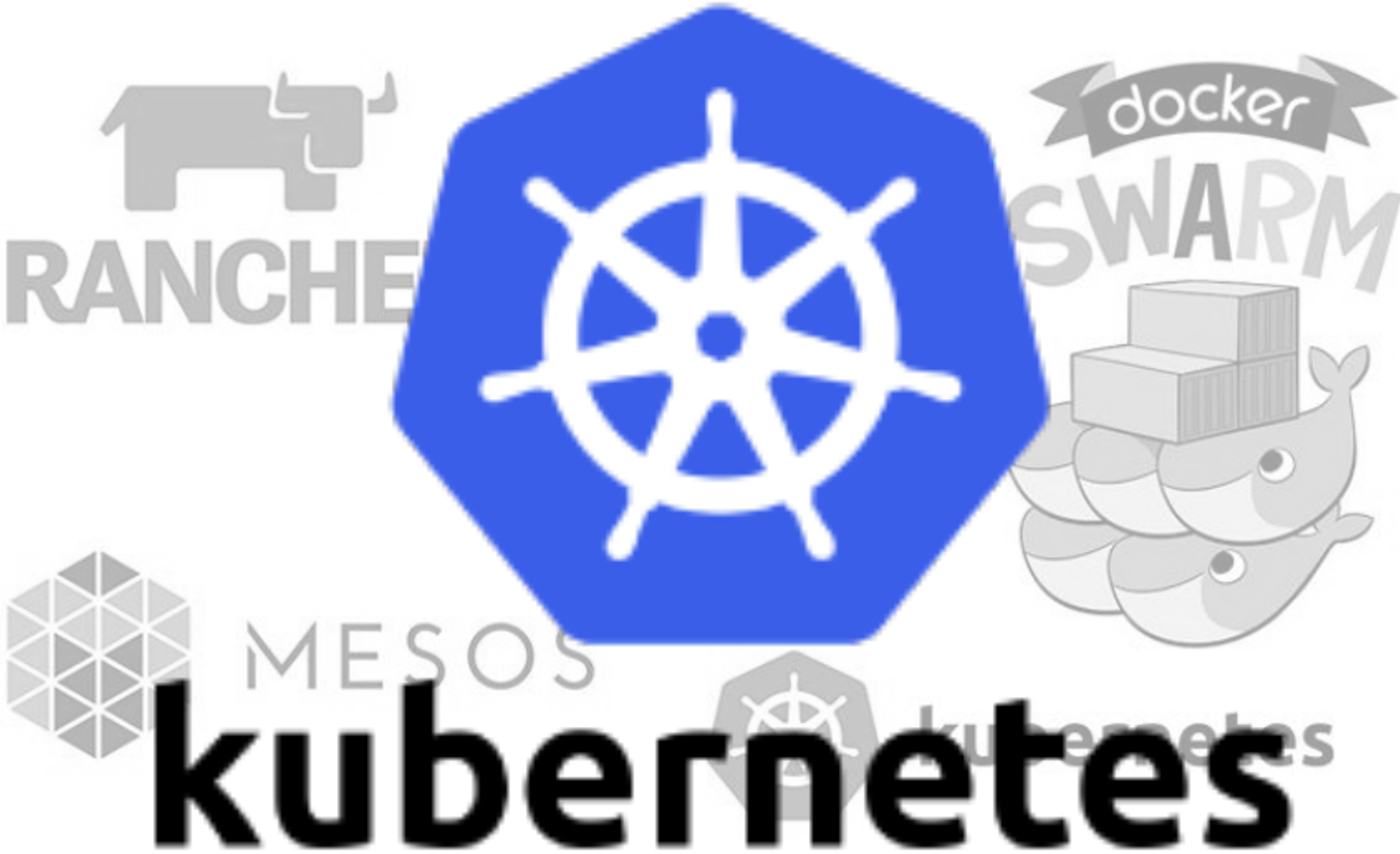

RANCHER


docker
SWARM


Nomad


MESOS


kubernetes


RANCHER
docker
SWARM
MESOS
kubernetes
kubernetes

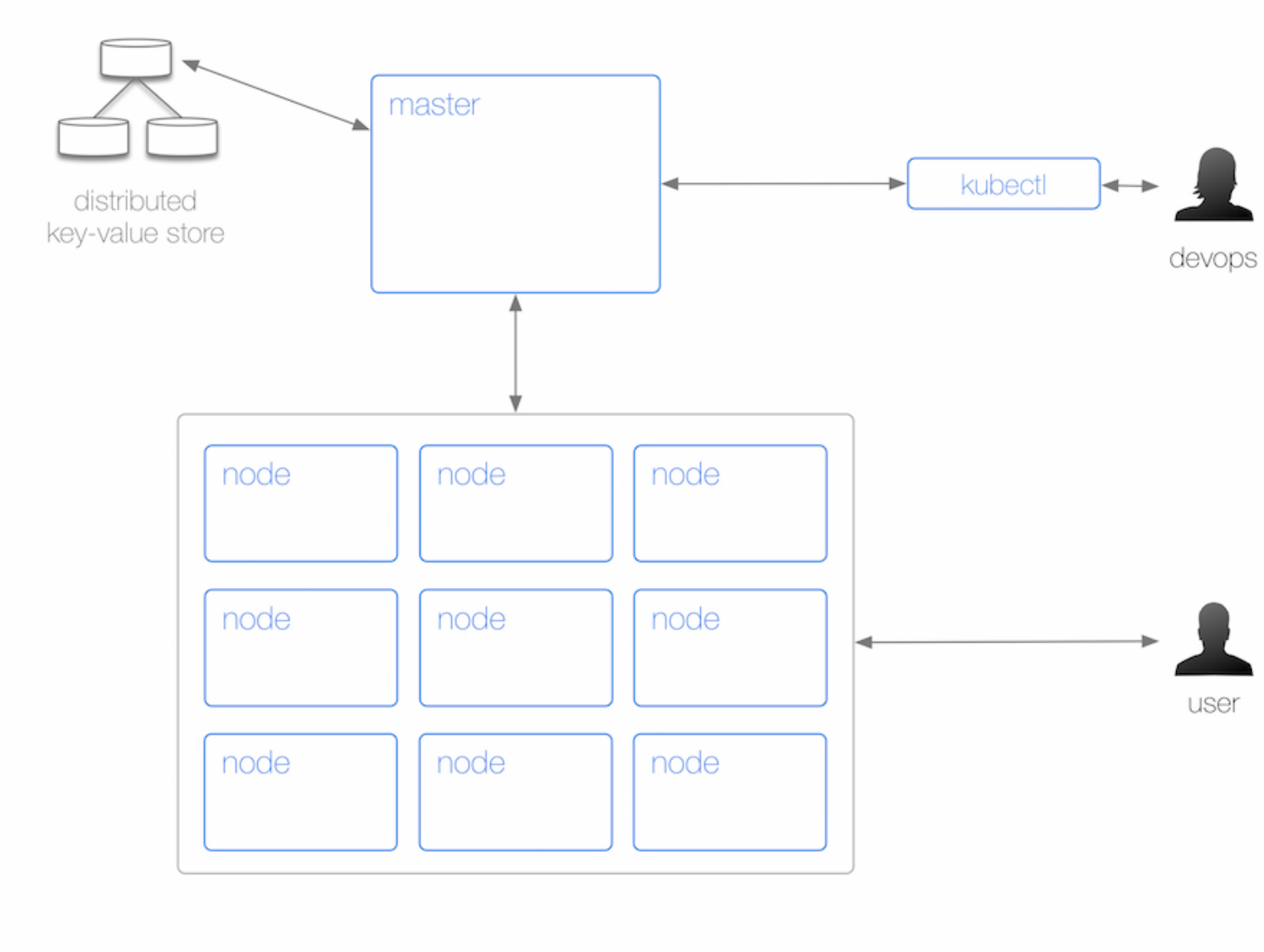
What does Orchestration do

- 여러 Host (Node)를 묶어 클러스터를 구성
- Container 를 적절한 위치에 배포 (Auto-placement)
- Container 가 죽으면 자동으로 복구 (Auto-restart)
- 필요에 따라 container를 Scaling, Replication, rolling update, rollback
- 기타 기능들(What is k8s?)

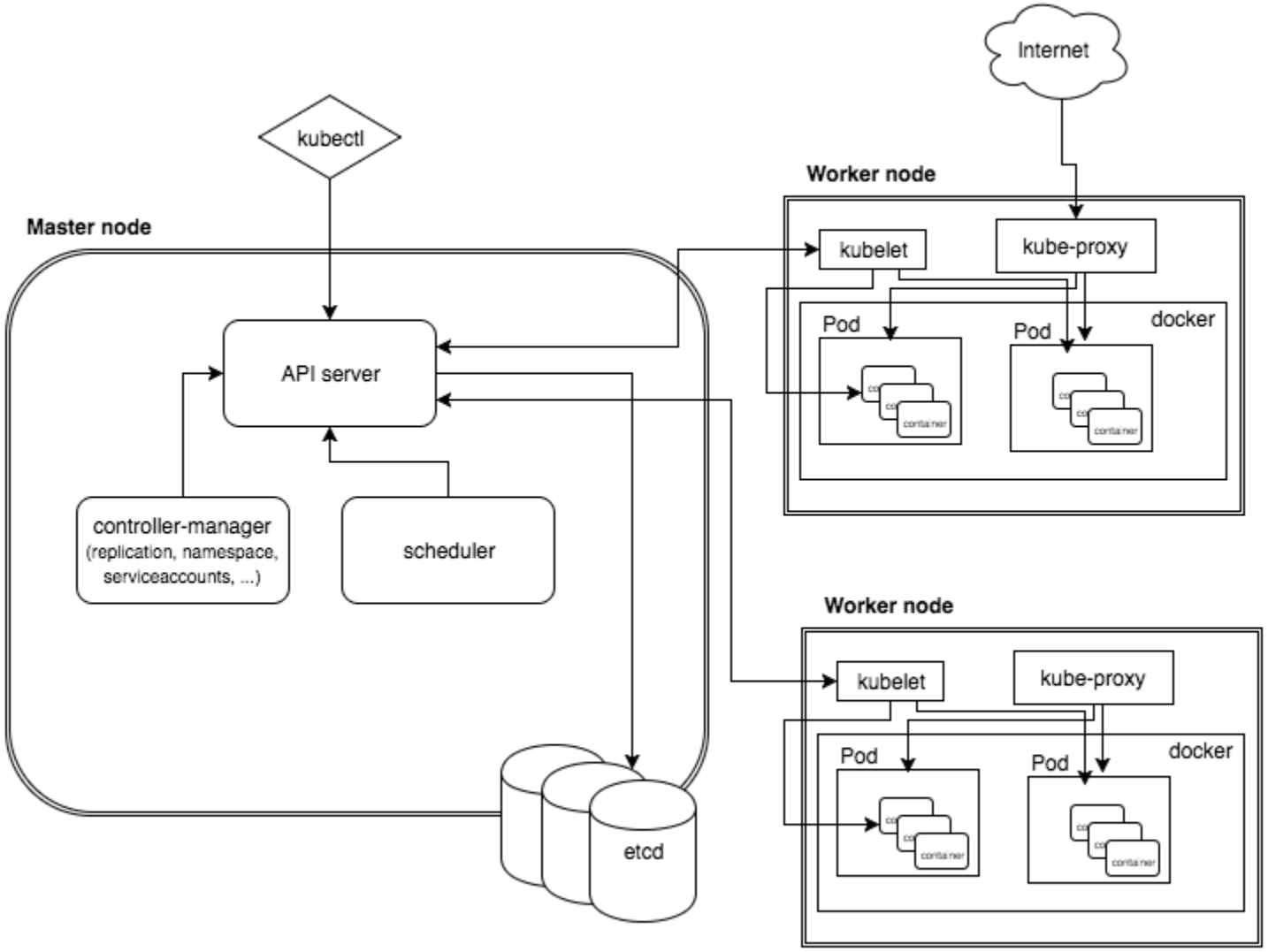
Should be understood to use

- kubernetes object
  (e. g. pod, volume, service, controller)
- kubernetes architecture
- multi-host 위에서 container 실행 시 고려해야 할 것들
  - service discovery
  - networking
  - volume management
  - log aggregation

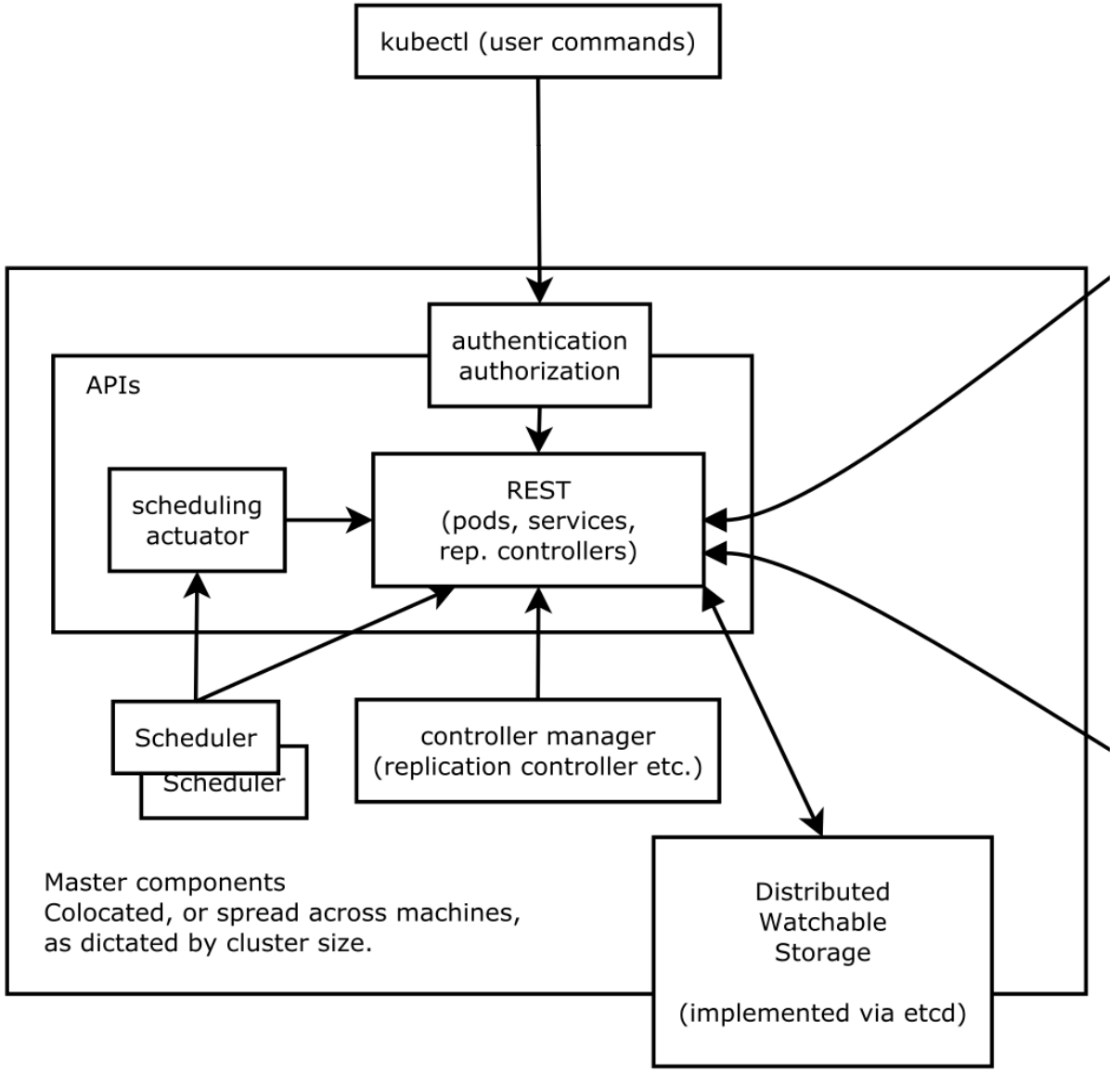
Kubernetes Architecture (abbreviated)



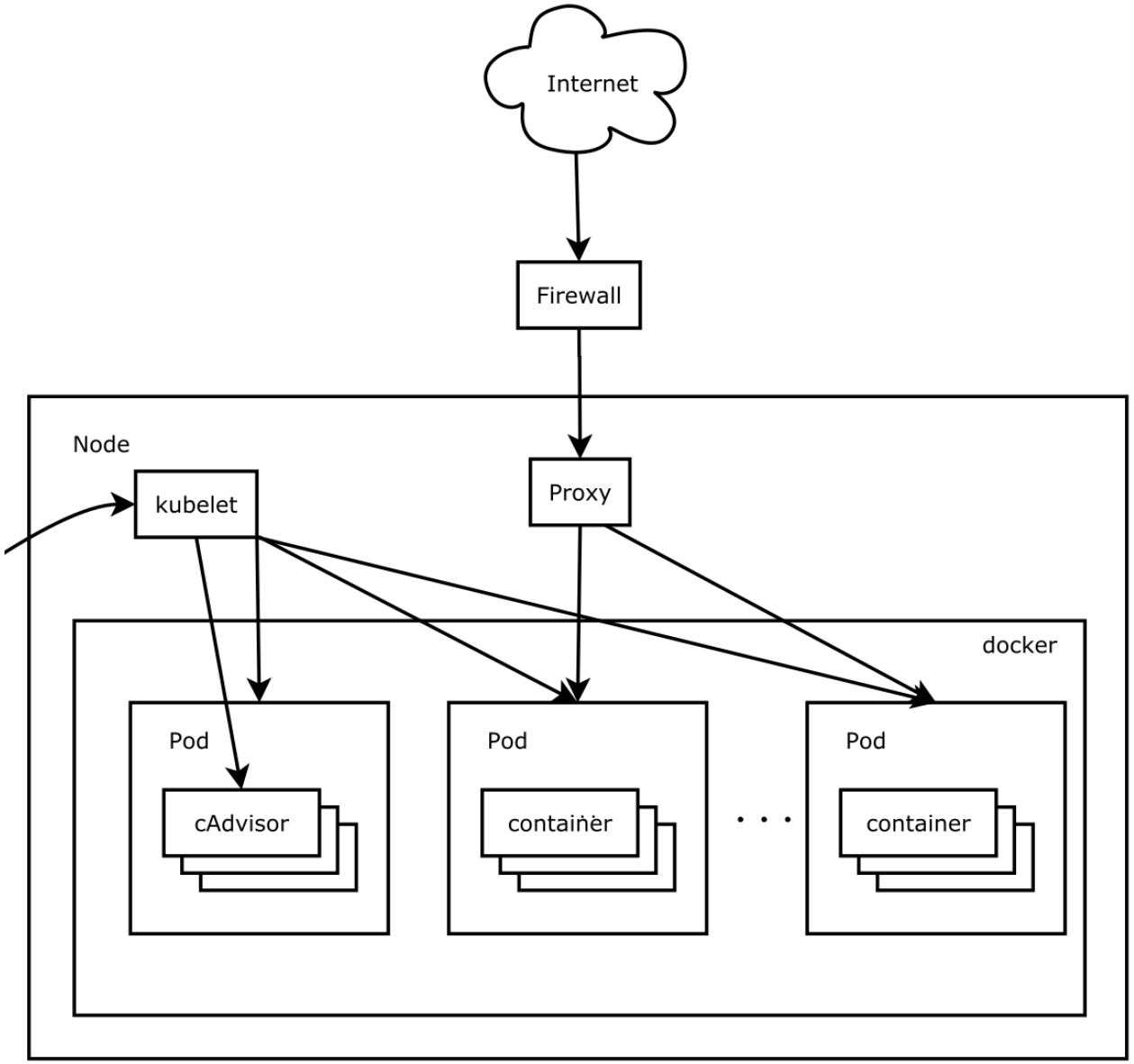
Kubernetes Architecture (detail)



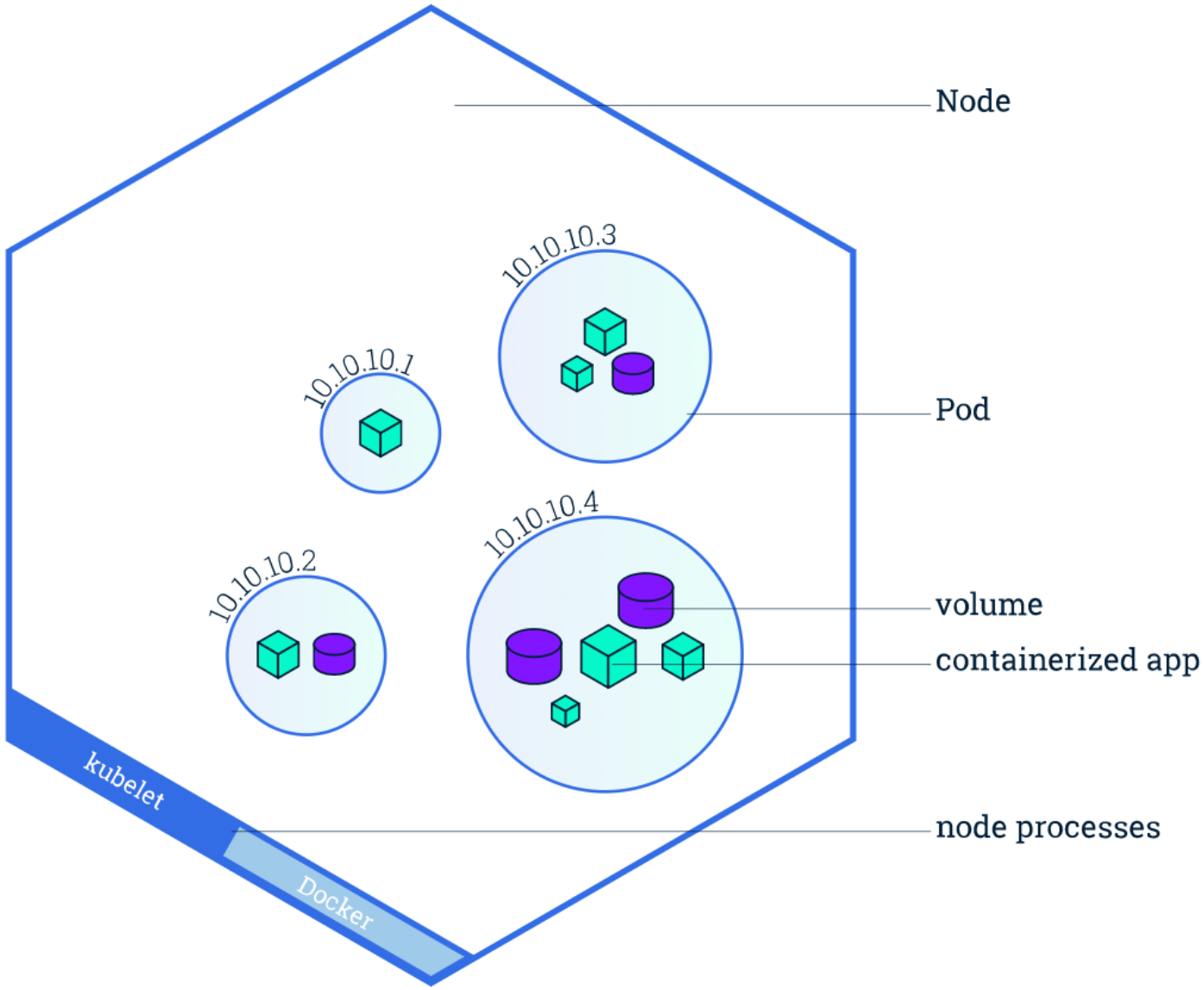
Kubernetes Architecture (master)



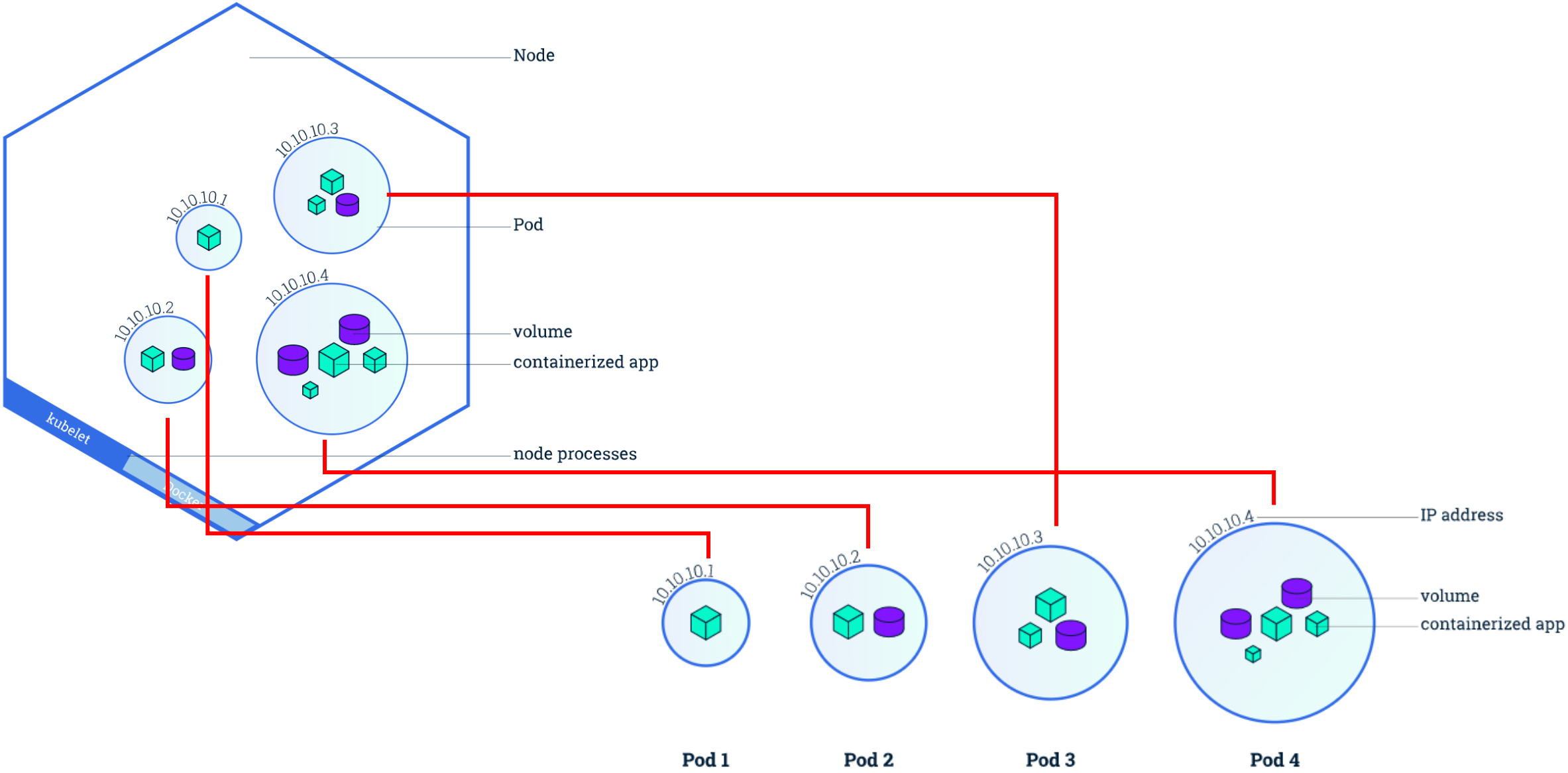
Kubernetes Architecture (Node)



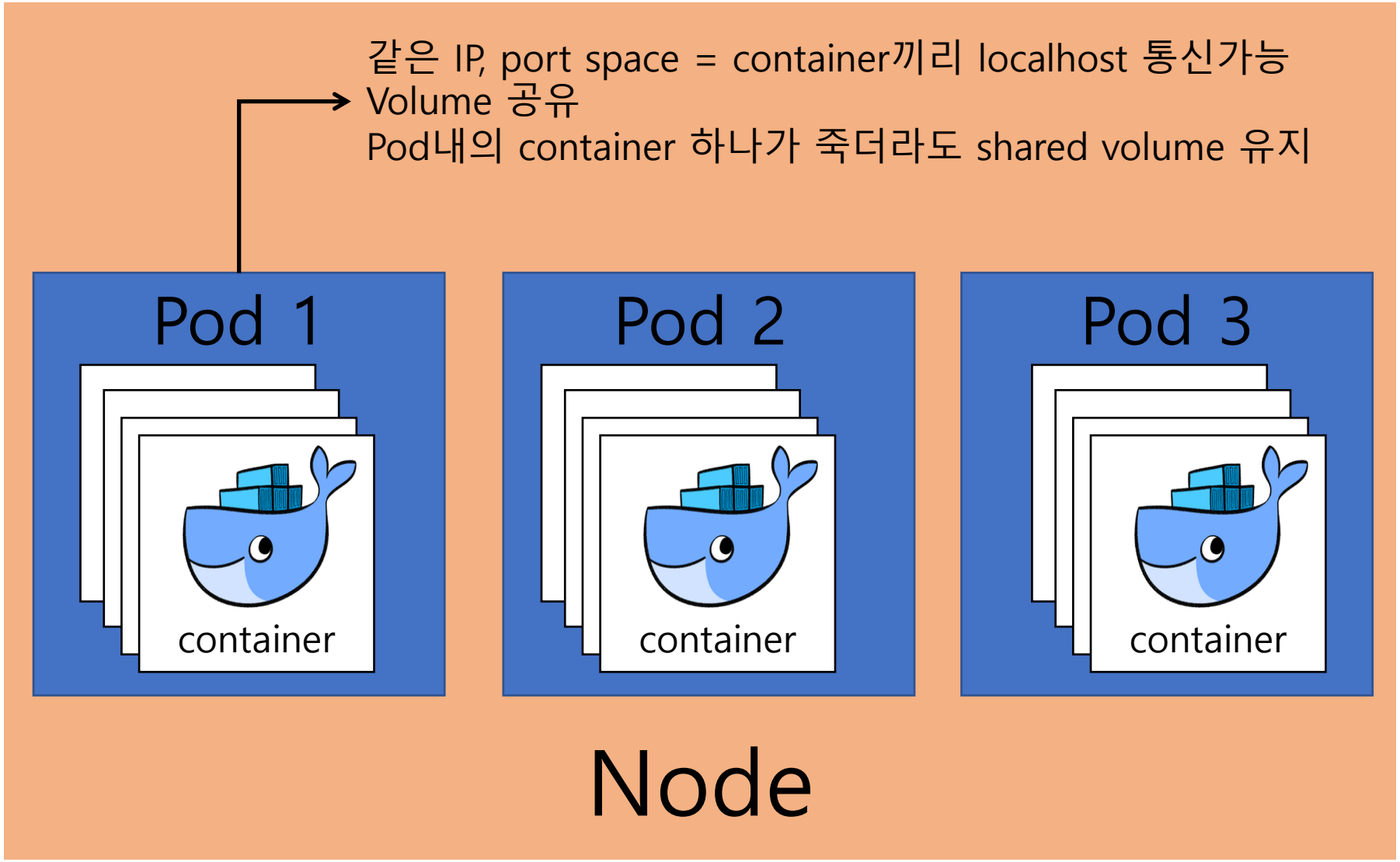
Kubernetes Object: Pod



Kubernetes Object: Pod

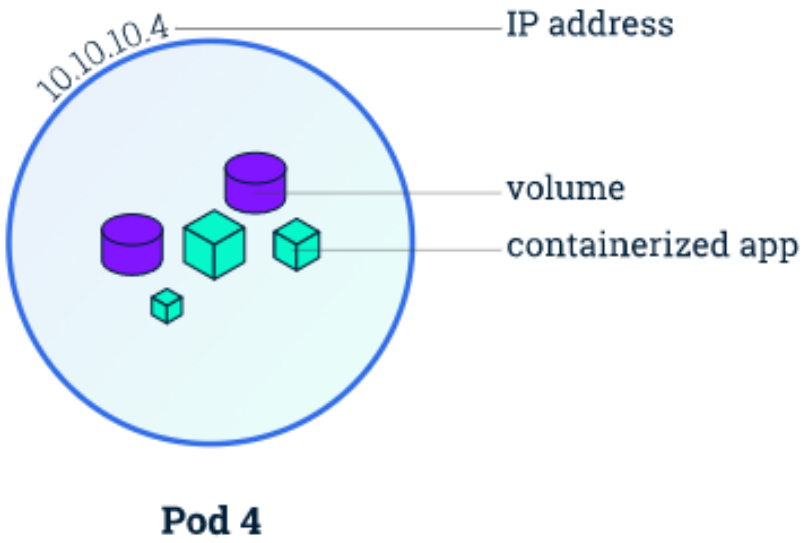


Kubernetes Object: Pod



Pod는 kubernetes의 작업 단위

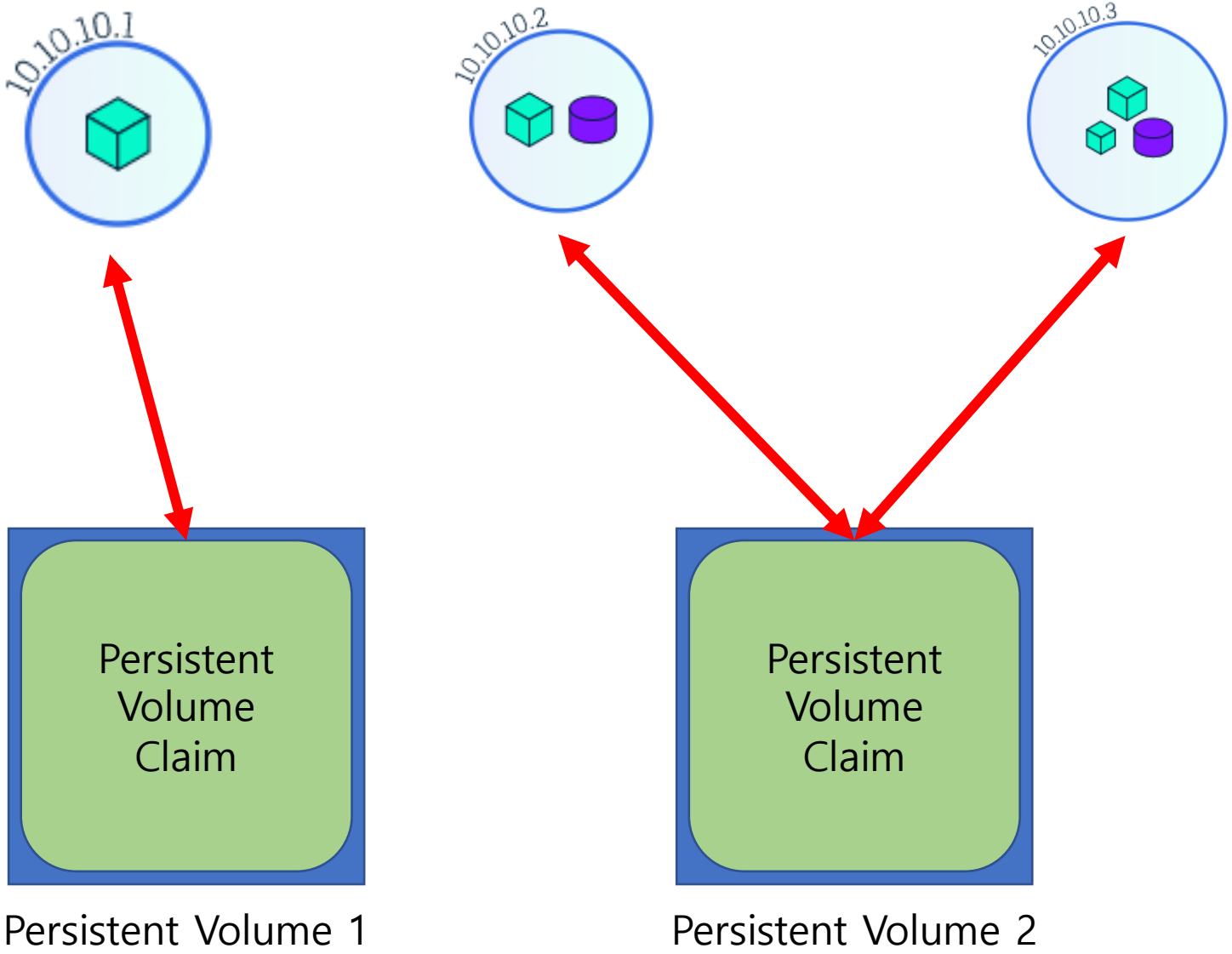
Kubernetes Object: Volume



stateless

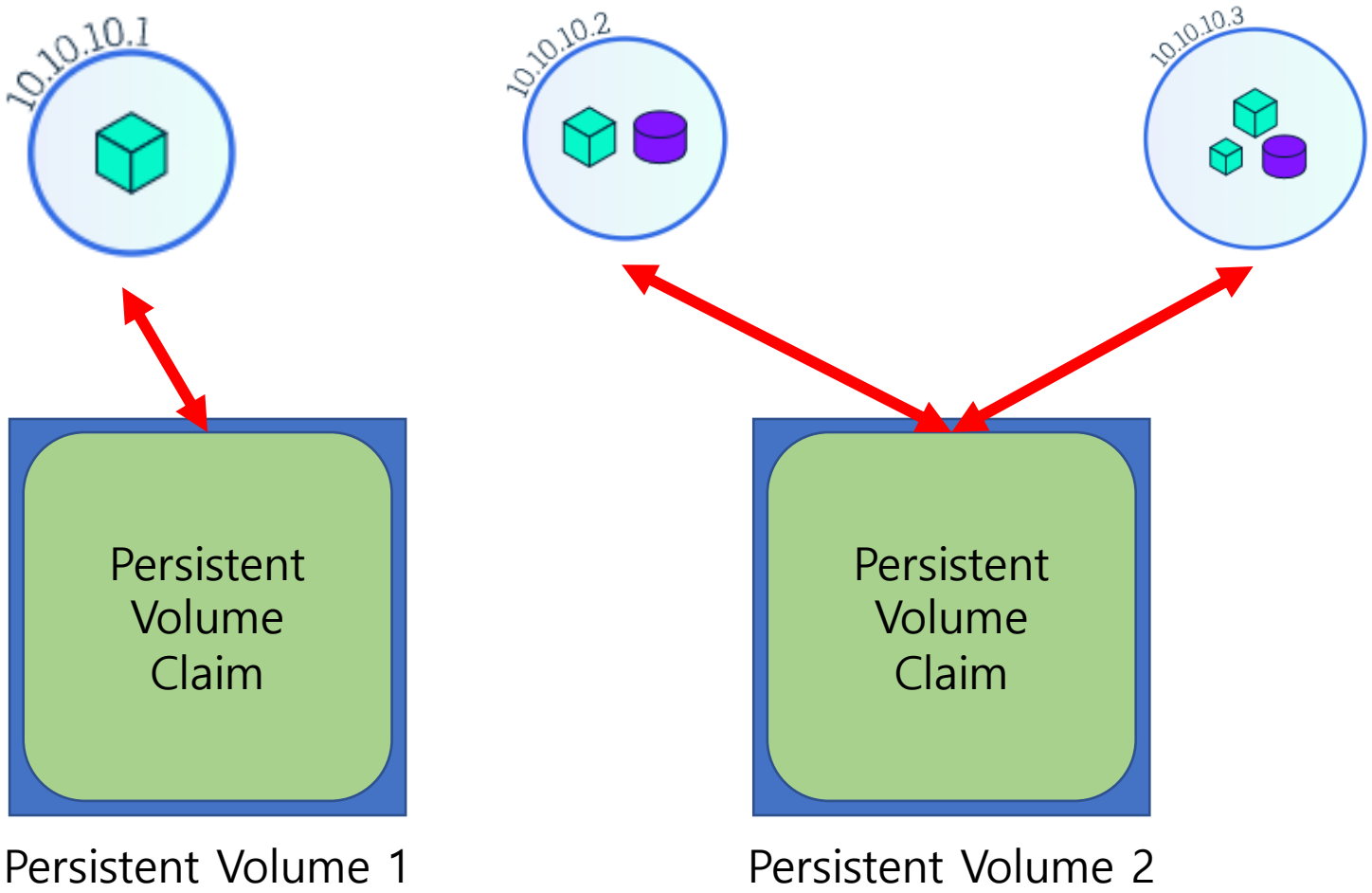
Pod는 상태저장이 되지 않음
즉 Pod가 사라지면 Pod의 volume도 사라짐

Kubernetes Object: Volume



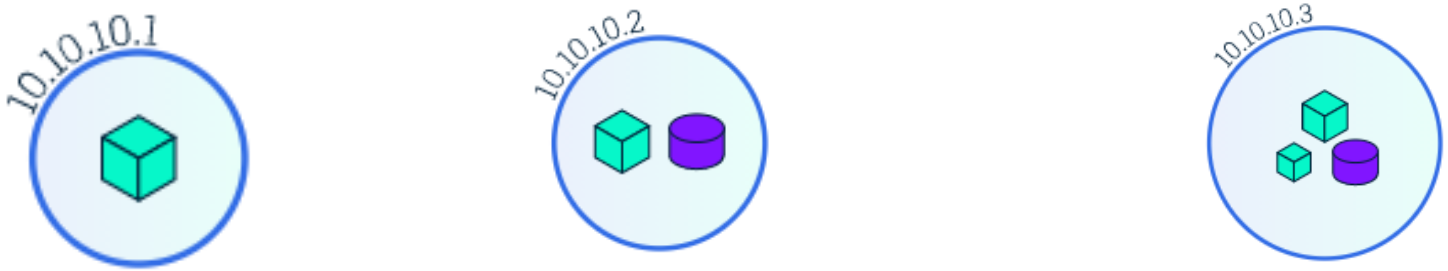
Kubernetes Object: Volume

정적 프로비저닝



Kubernetes Object: Volume

정적 프로비저닝

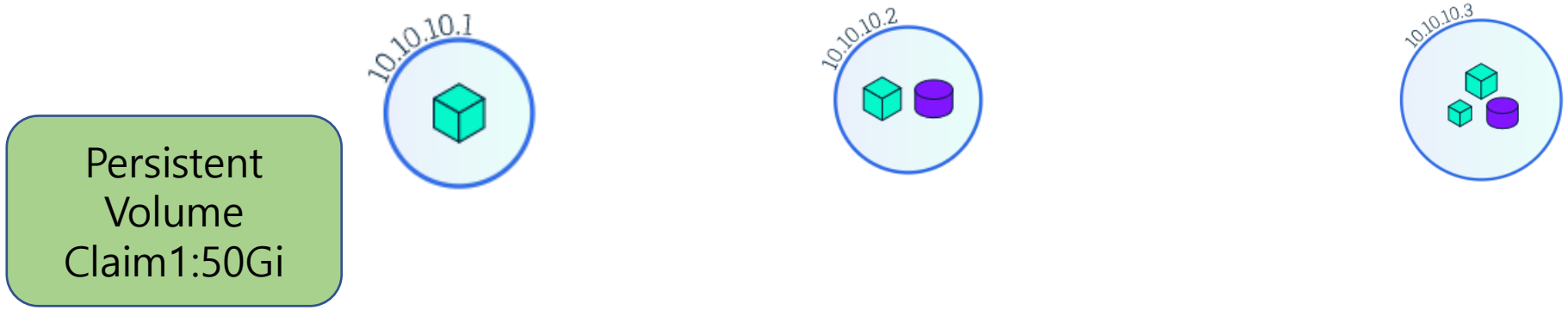


미리 PV를 만들어 놓음

Persistent Volume 1 Persistent Volume 2 Persistent Volume 3

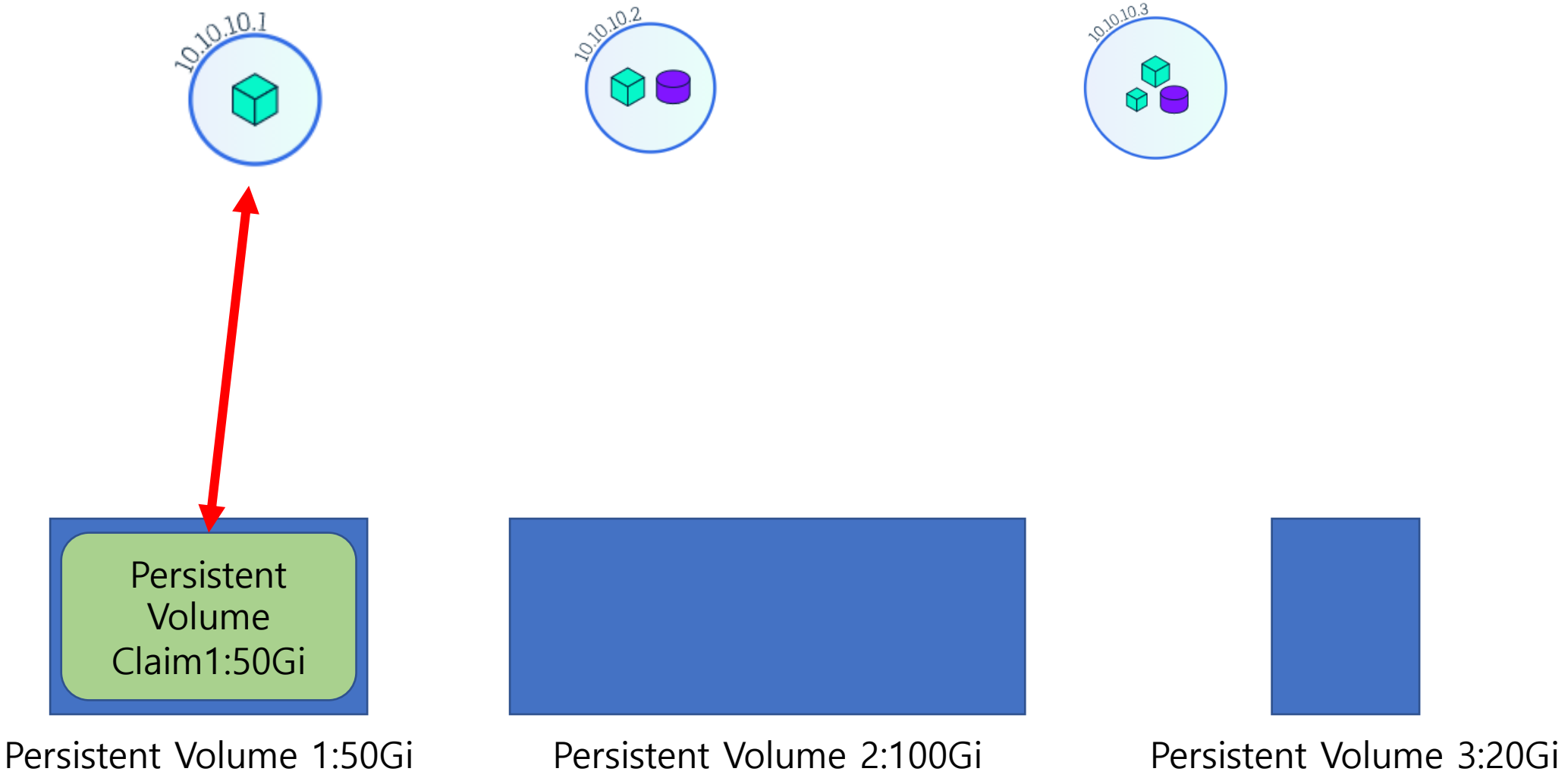
Kubernetes Object: Volume

정적 프로비저닝



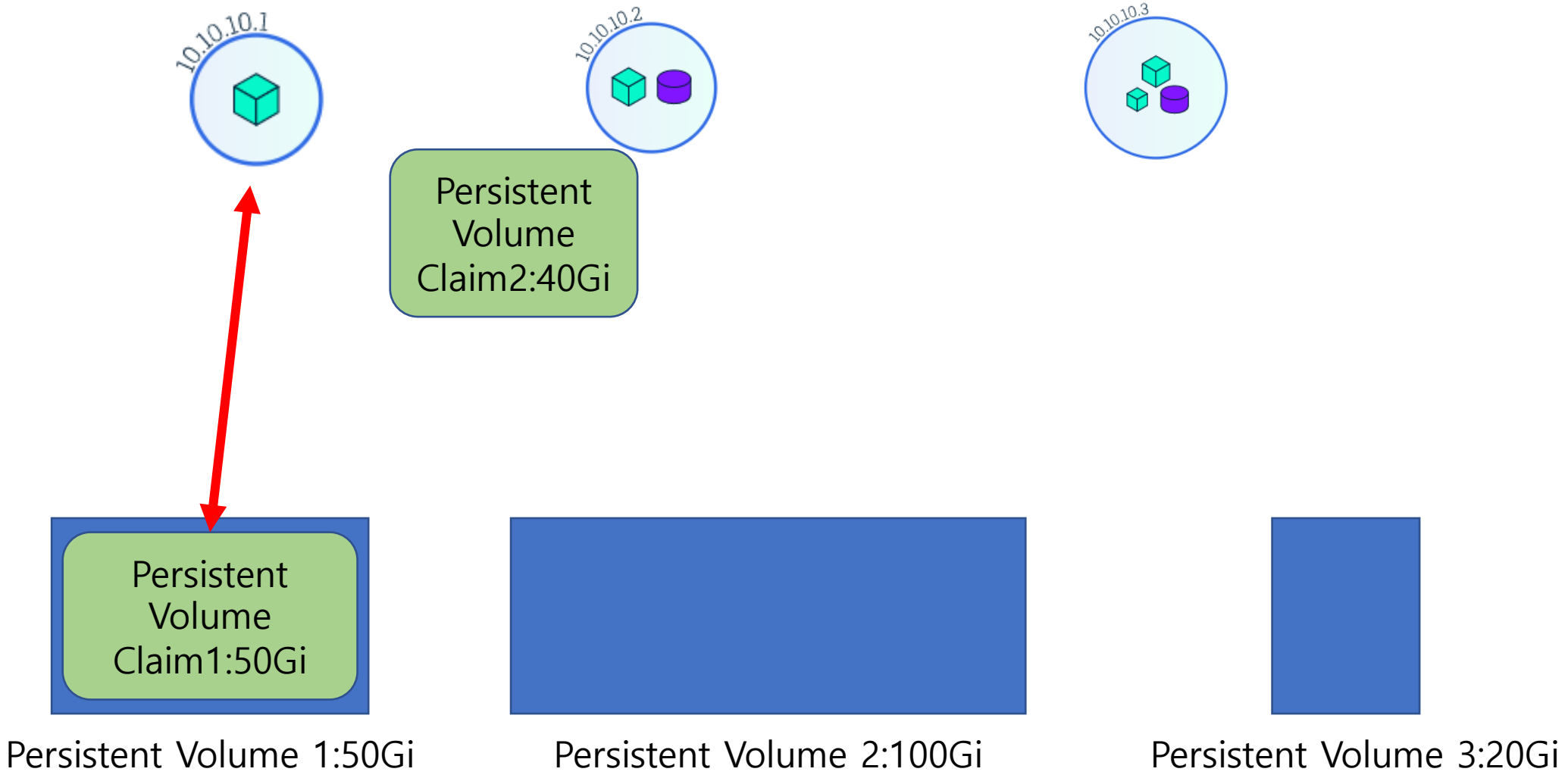
Kubernetes Object: Volume

정적 프로비저닝



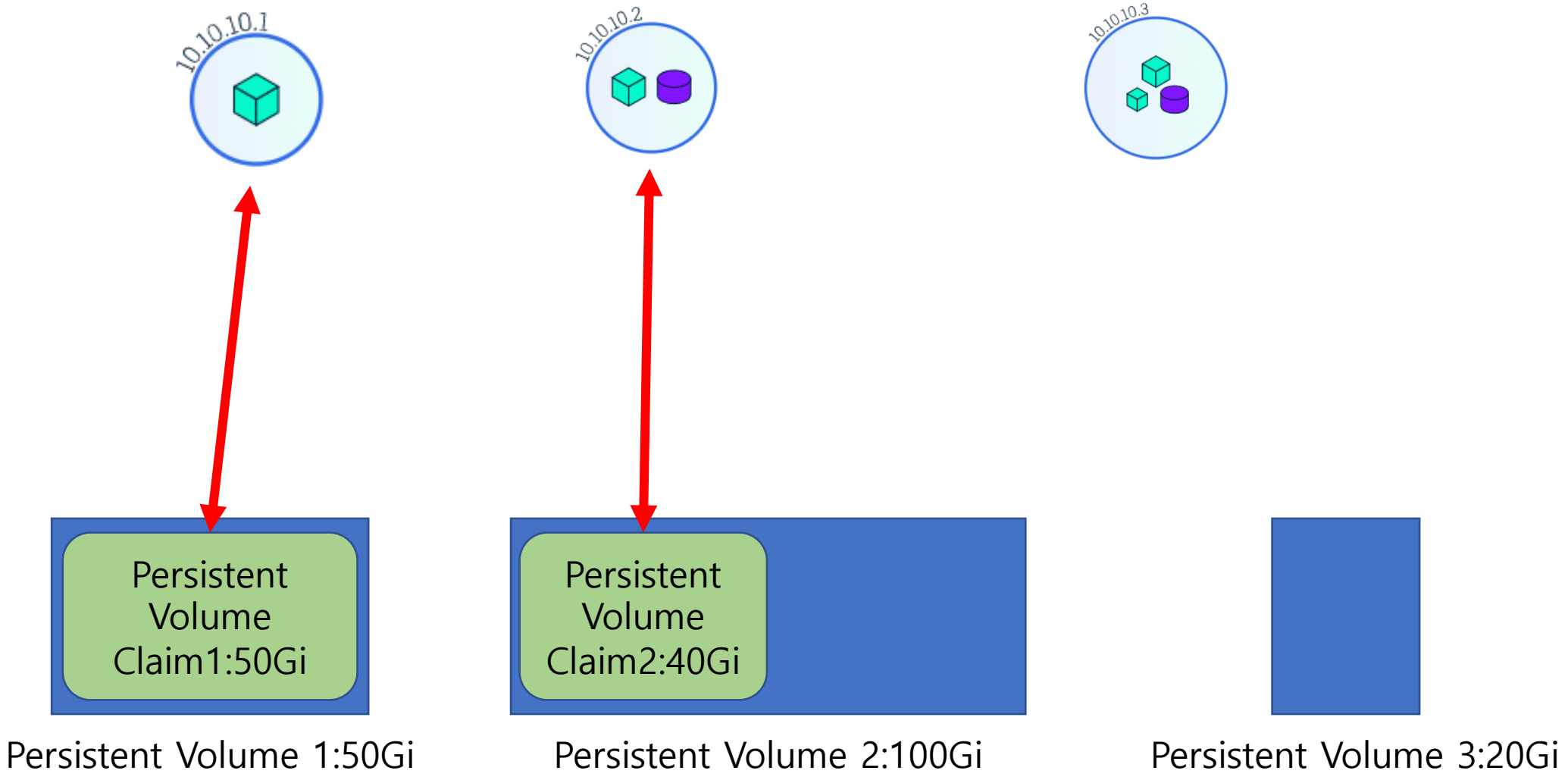
Kubernetes Object: Volume

정적 프로비저닝



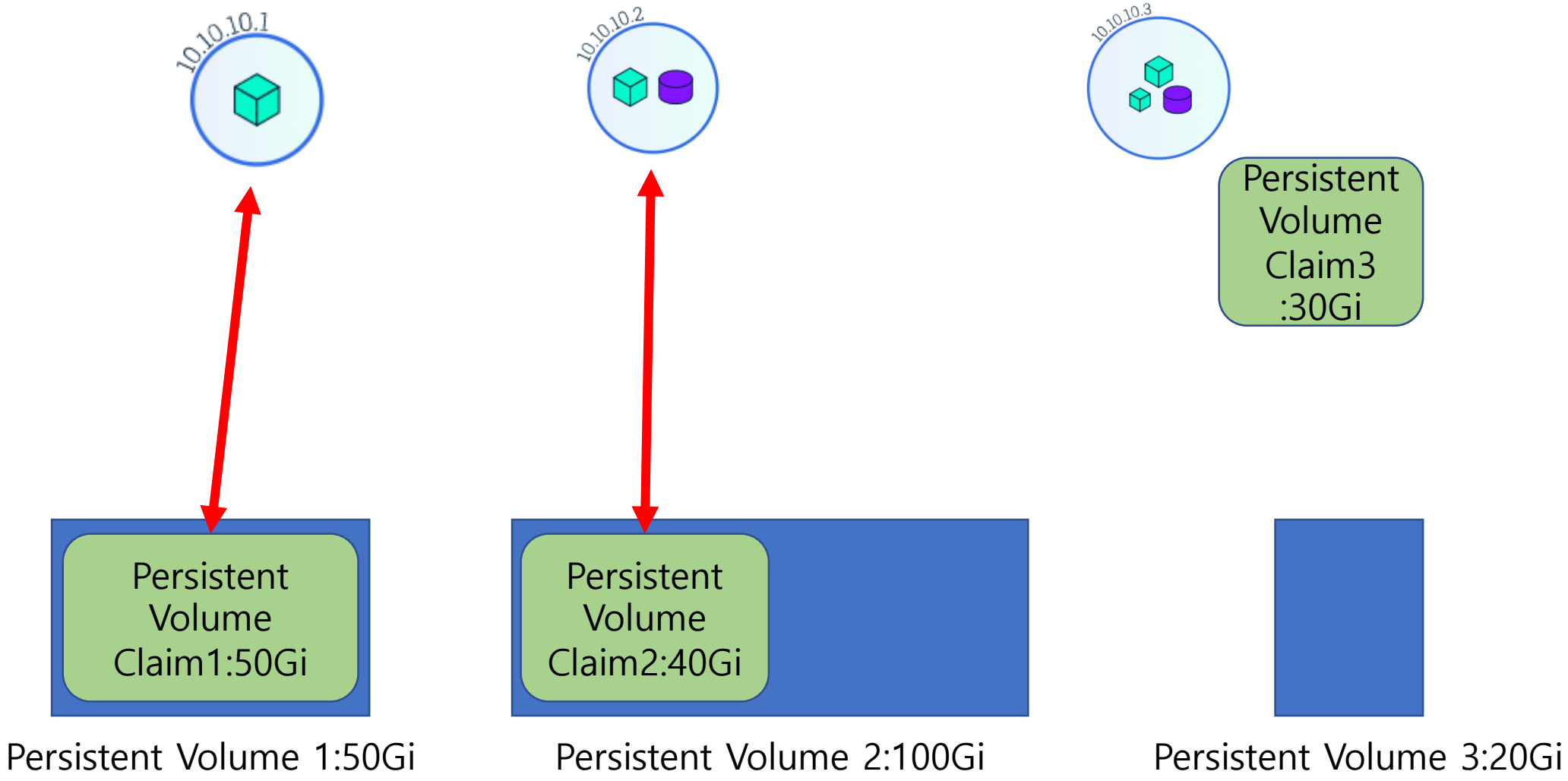
Kubernetes Object: Volume

정적 프로비저닝



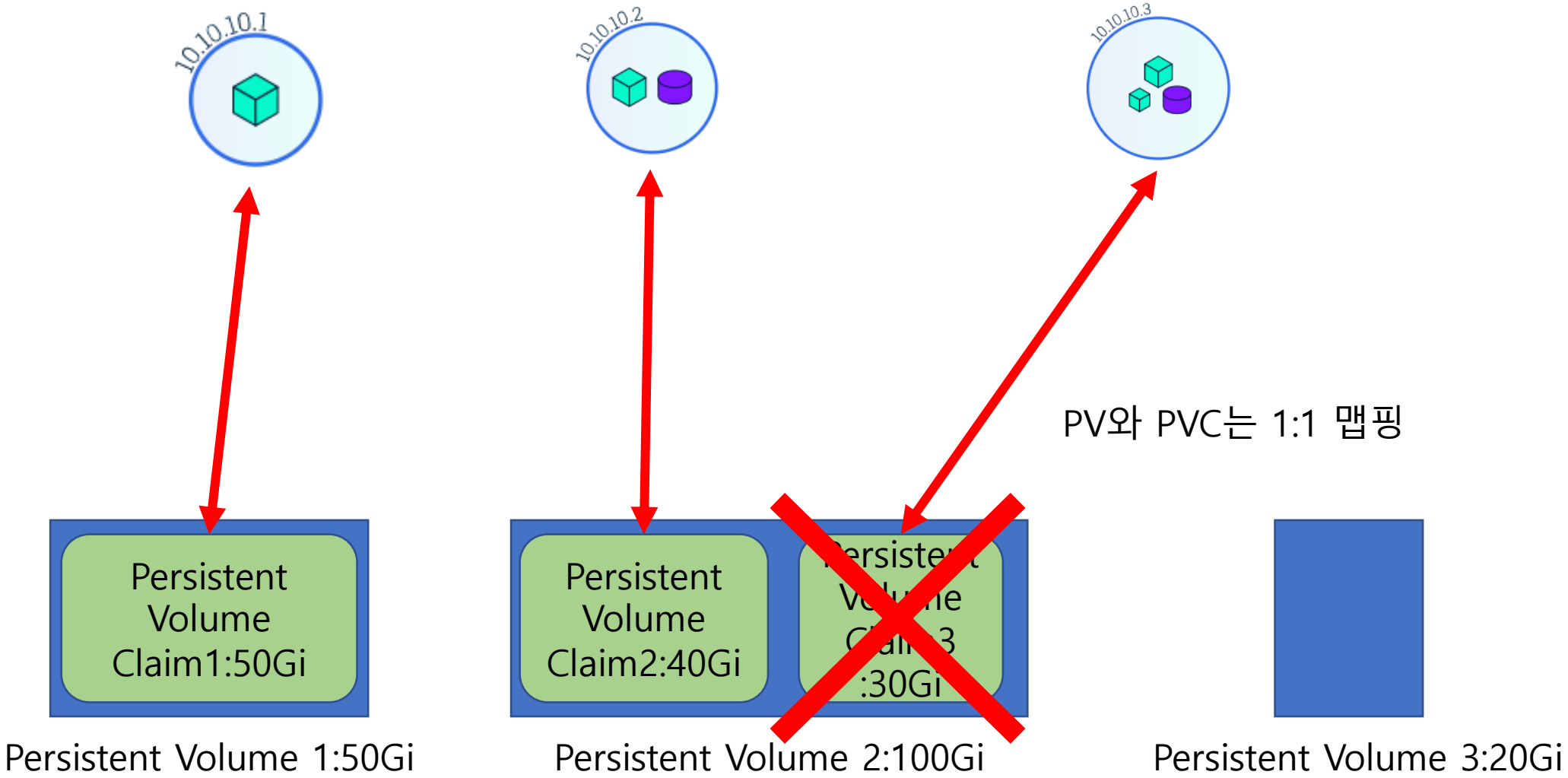
Kubernetes Object: Volume

정적 프로비저닝



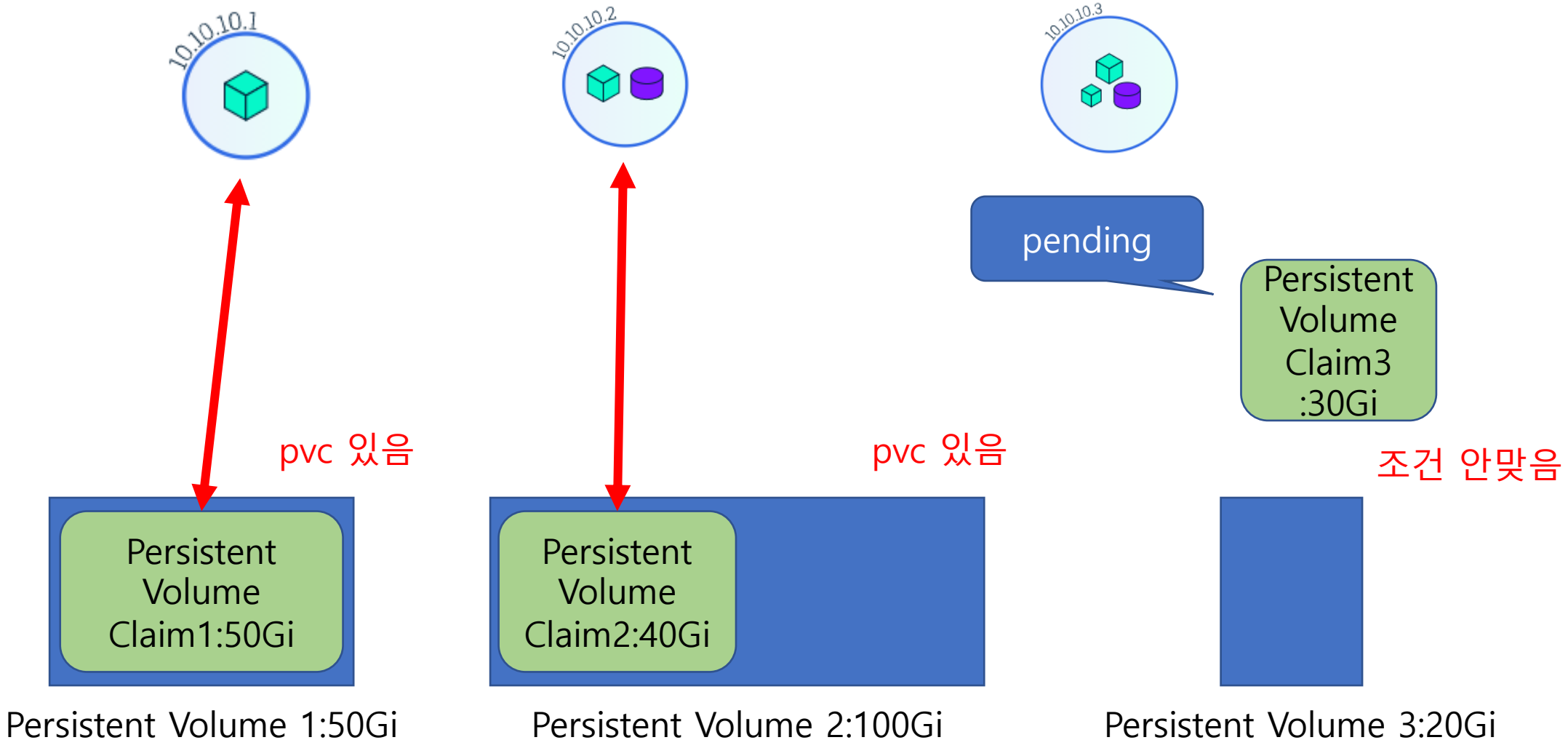
Kubernetes Object: Volume

정적 프로비저닝



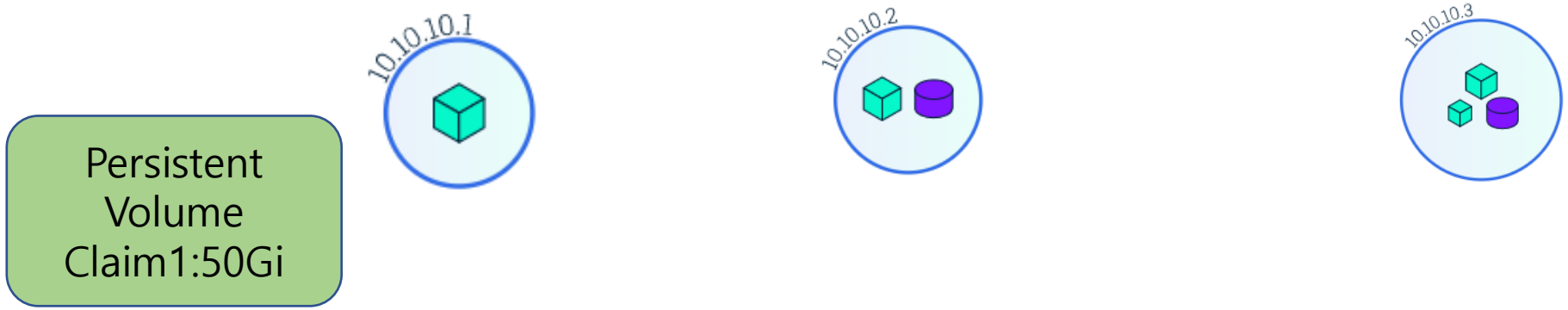
Kubernetes Object: Volume

정적 프로비저닝



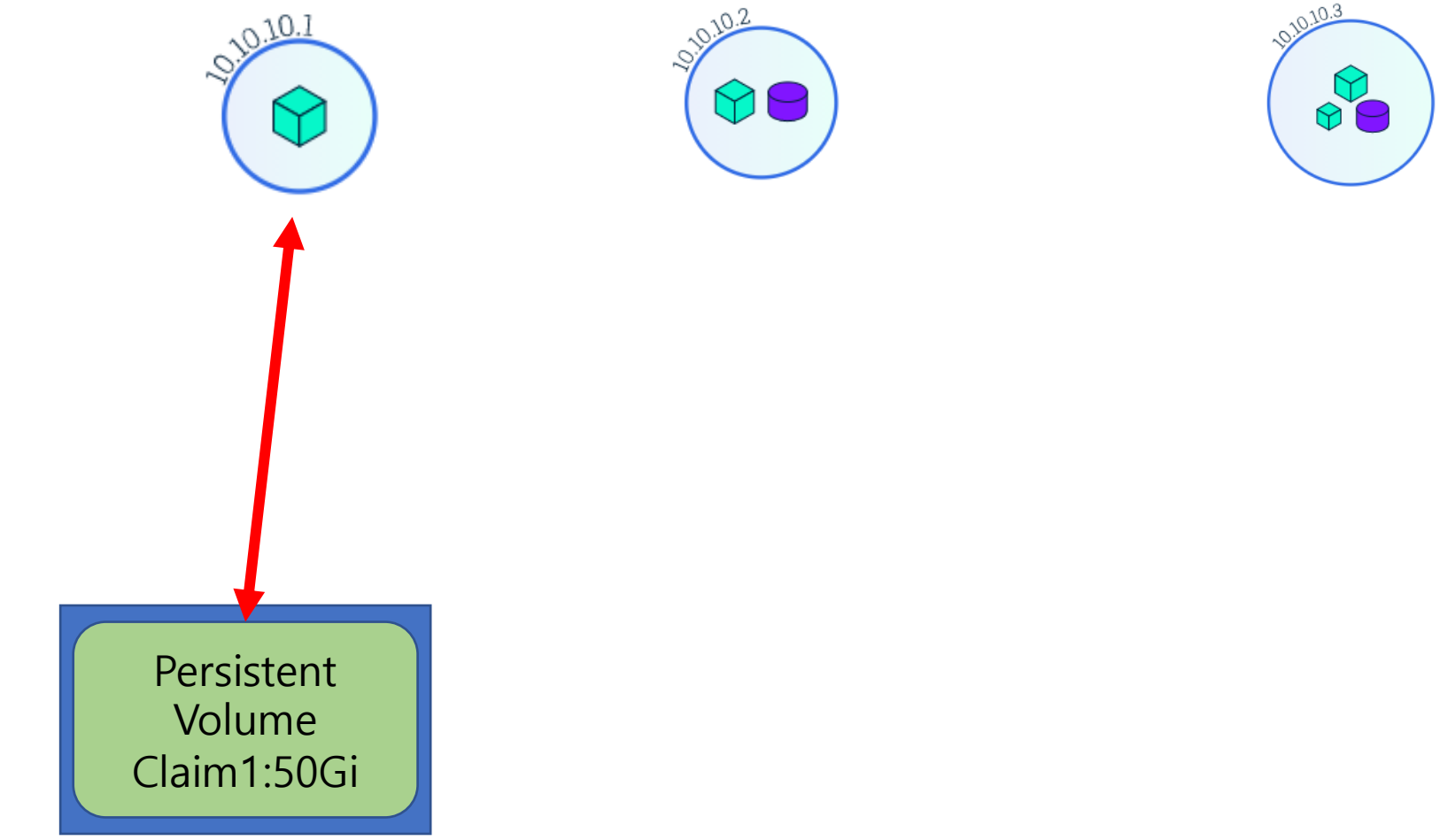
Kubernetes Object: Volume

동적 프로비저닝



Kubernetes Object: Volume

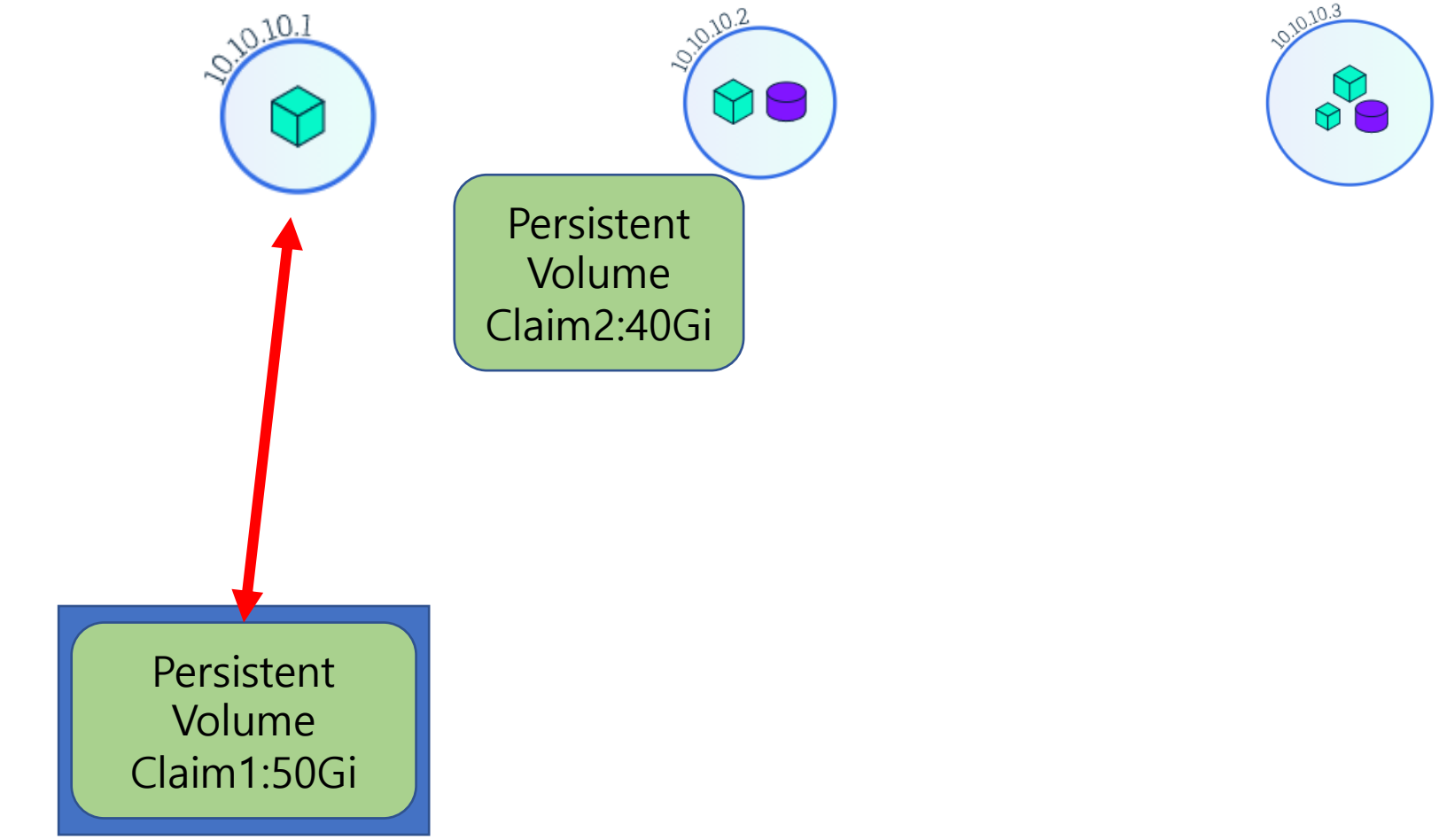
동적 프로비저닝



Persistent Volume 1:50Gi

Kubernetes Object: Volume

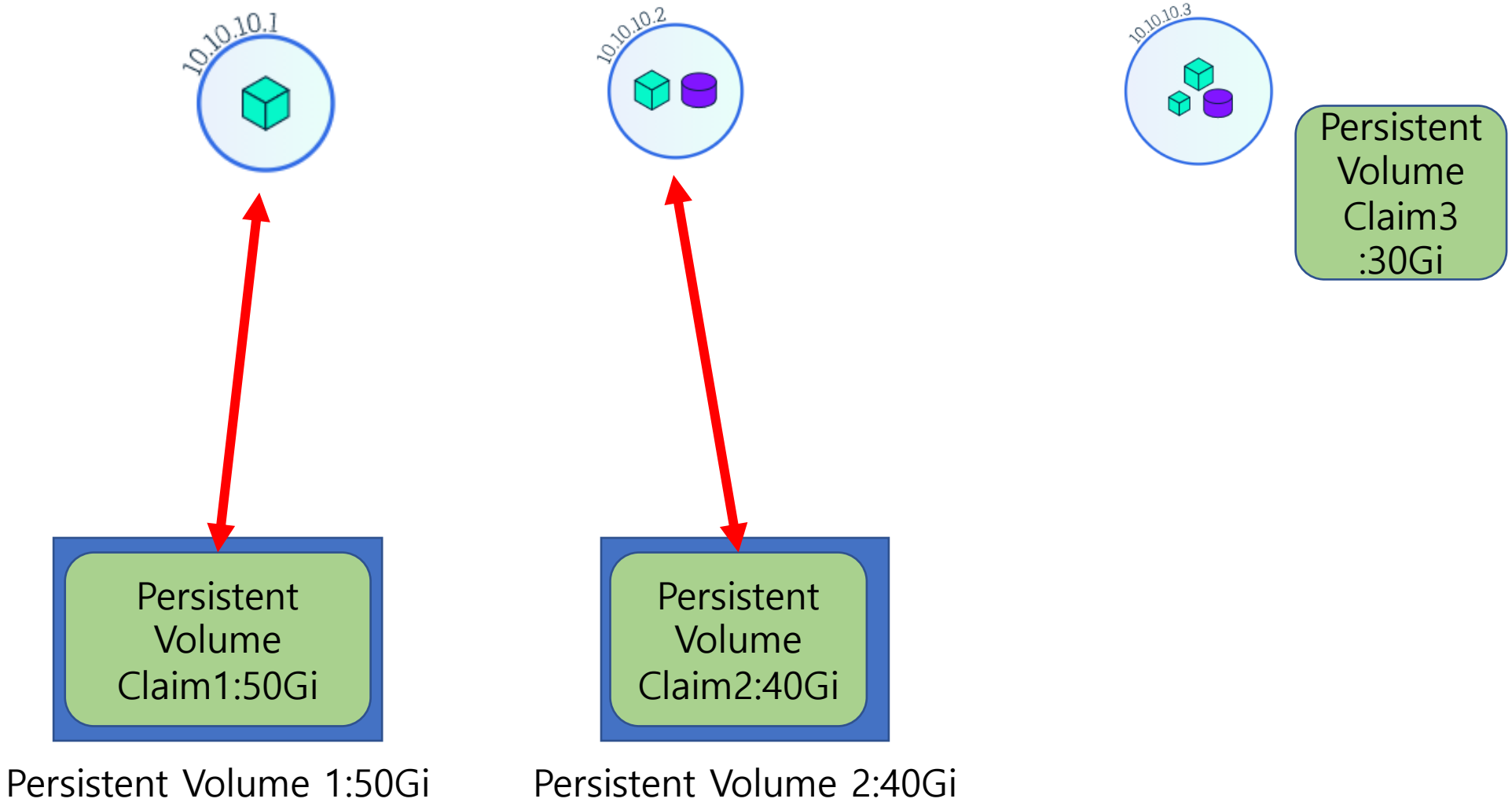
동적 프로비저닝



Persistent Volume 1:50Gi

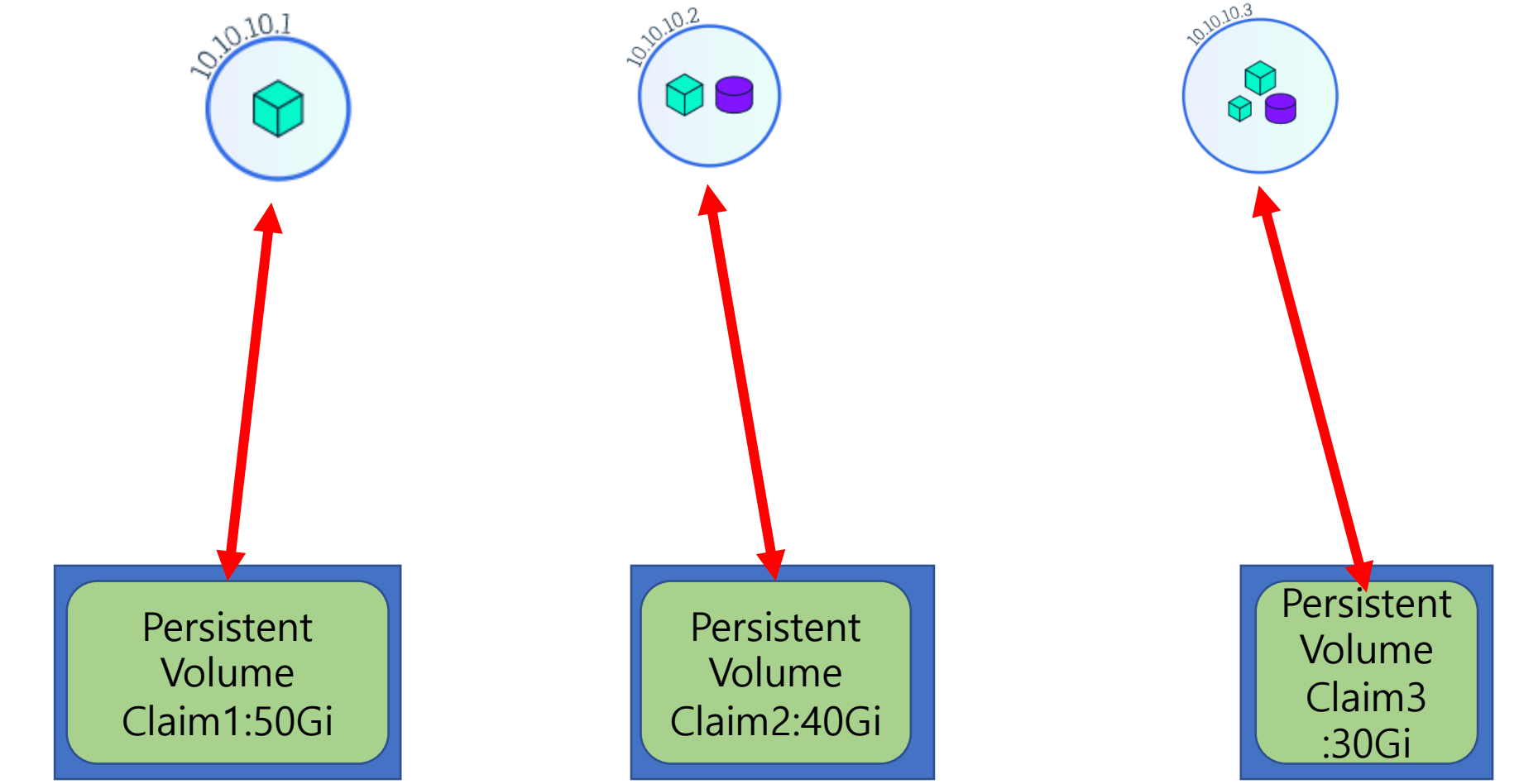
Kubernetes Object: Volume

동적 프로비저닝



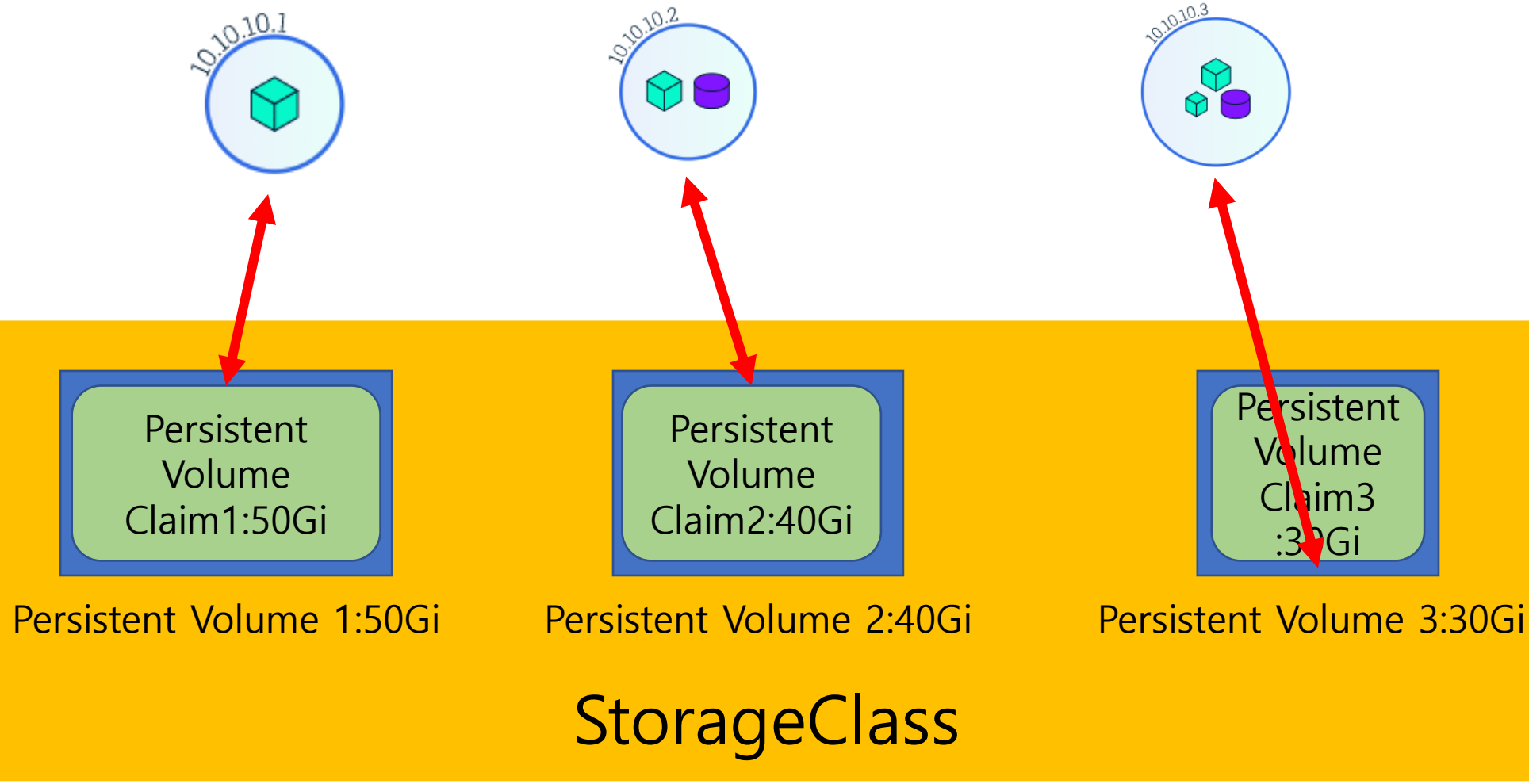
Kubernetes Object: Volume

동적 프로비저닝



Kubernetes Object: Volume

동적 프로비저닝



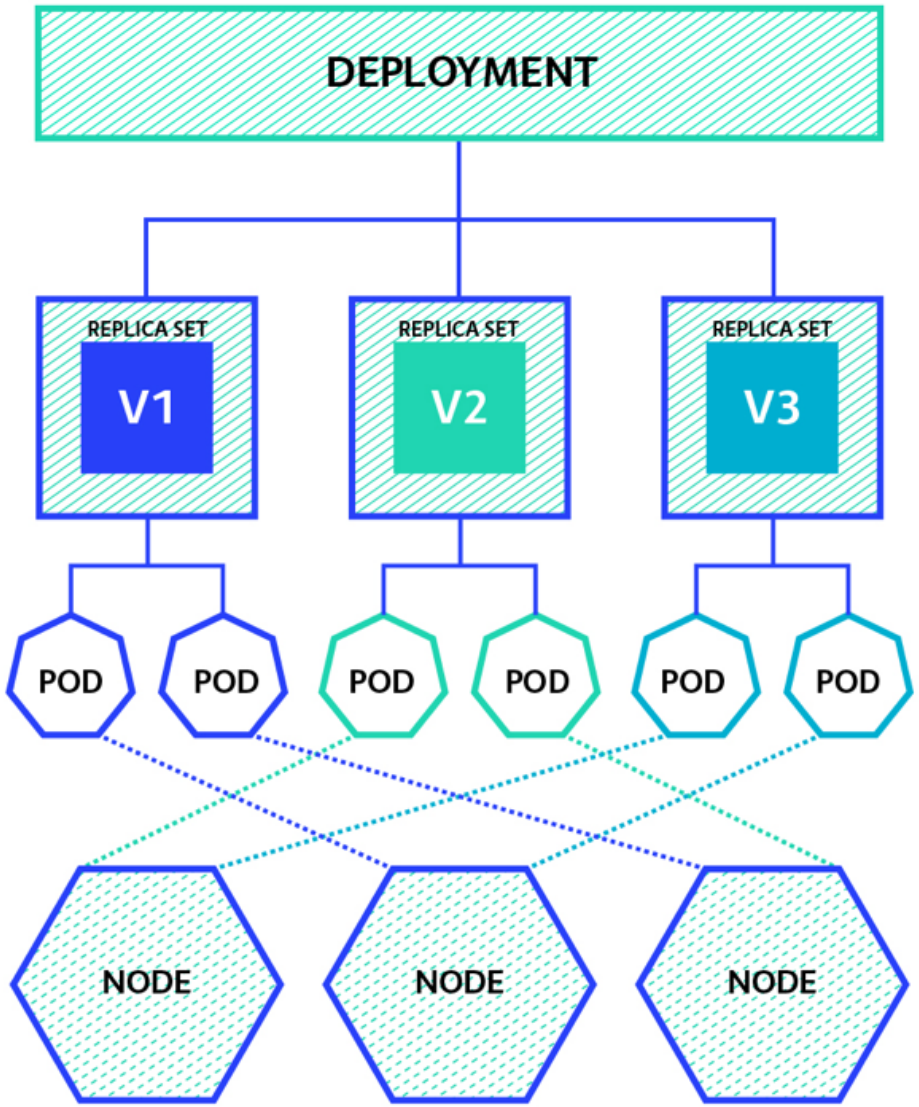
StorageClass

접근설정	정책
ReadWriteOnce	하나의 싱글노드에서만 읽고 쓰기
ReadOnlyMany	하나의 싱글노드에서 쓰고 여러 노드에게 읽기만 가능
ReadWriteMany	여러 노드에게 읽고 쓰기 가능

Kubernetes controller: Deployment

- managing pod, replica set
- Support Rolling update
- Support Rollback

Kubernetes controller: Deployment



Kubernetes controller: Deployment

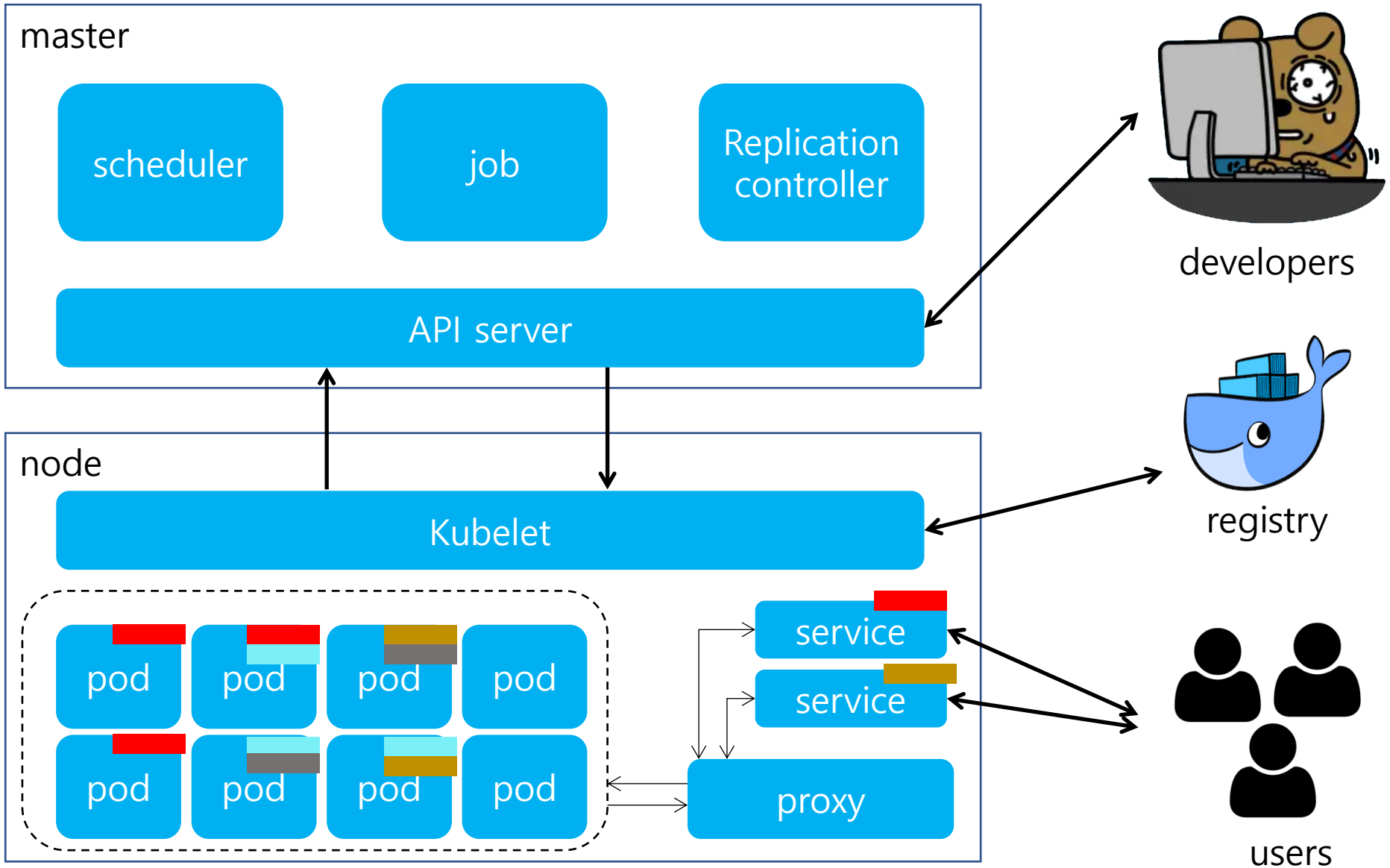
```yaml
kind: Deployment
apiVersion: extensions/v1beta1
metadata:
  name: hostname-101-deployment
spec:
  replicas: 3
  selector:
    # Like saying "Make sure there are three pods running
    # with the label app = hostname and version = v101"
    matchLabels:
      app: hostname
      version: v101
  template:
    metadata:
      labels:
        # The `app` label is used by both the service
        # and the deployment to select the pods they operate on.
        app: hostname
        # The `version` label is used only by the deployment
        # to control replication.
        version: v101
    spec:
      containers:
        - name: nginx-hostname
          image: kubegoldenguide/nginx-hostname:1.0.1
          ports:
            - containerPort: 80
```

https://matthewpalmer.net/kubernetes-app-developer/articles/kubernetes-deployment-tutorial-example-yaml.html

Kubernetes Object: service

- Pod의 IP는 클러스터 내부 IP -> 그럼 외부에서는 어떻게 접속?
- service를 이용해서 클러스터 외부로 노출
- OSI7계층 중 3계층(네트워크계층 : ip부여 / 라우팅)
- 서비스에 속한 Pod는 label selector로 판단
- 로드밸런싱 및 서비스 디스커버리 수행

Kubernetes Object: service

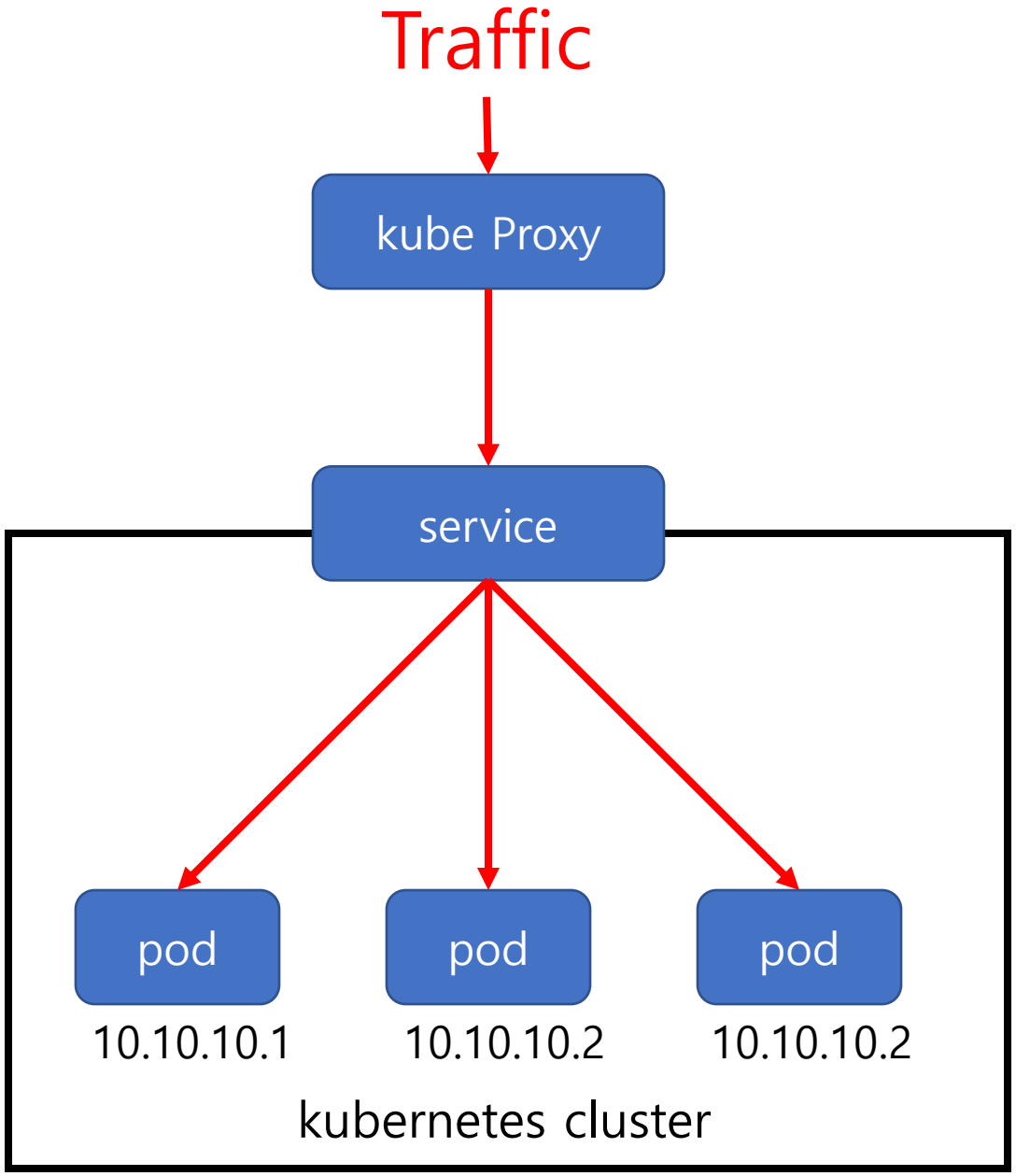


Cluster IP

- Cluster 내부 통신을 위한 IP
- 외부에서 Access 불가능
- 외부에서 접근을 위해 kube proxy이용

```
apiVersion: v1
kind: Service
metadata:
  name: my-internal-service
spec:
  selector:
    app: my-app
  type: ClusterIP
  ports:
  - name: http
    port: 80
    targetPort: 80
    protocol: TCP
```

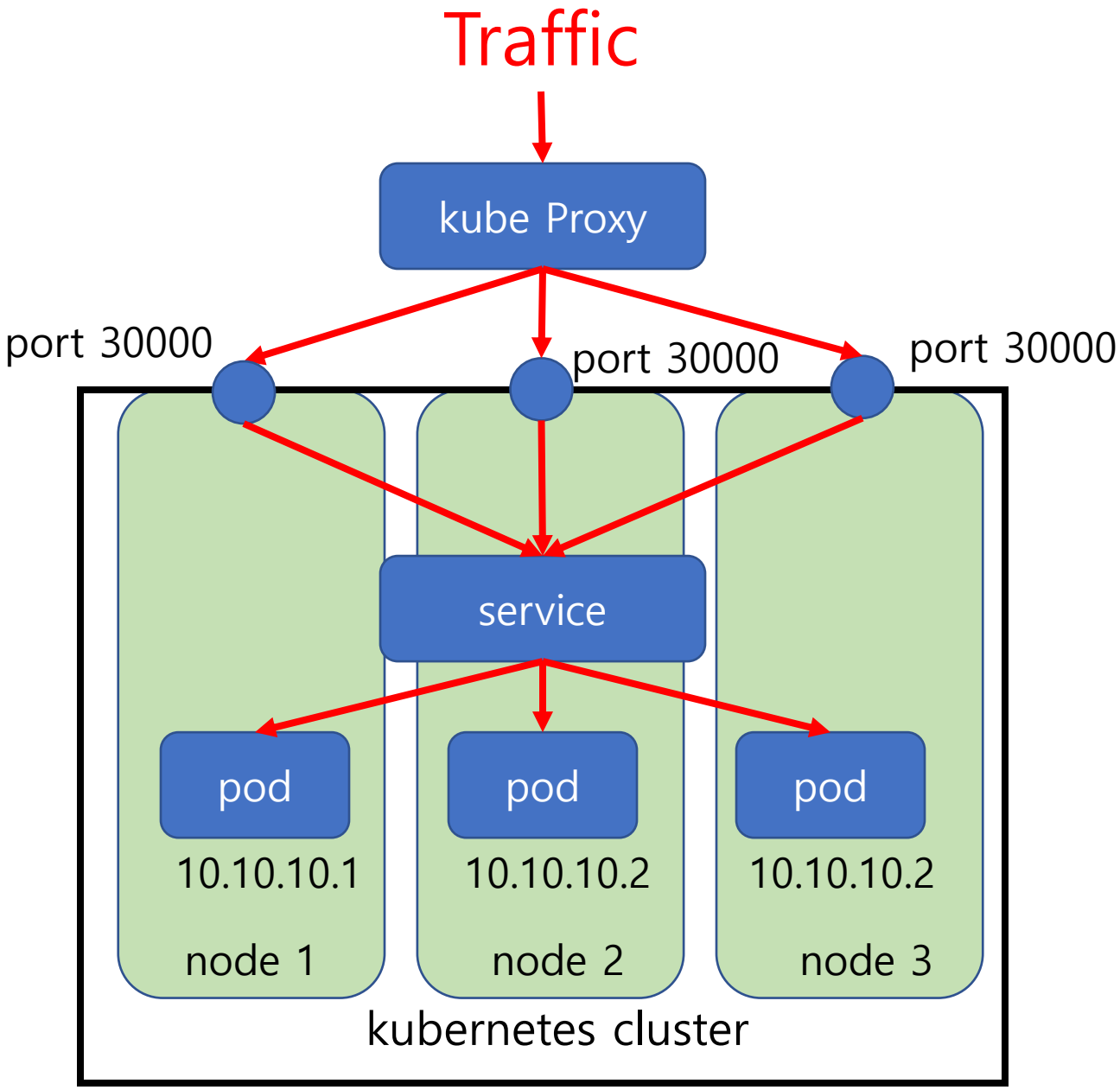
```
$ kubectl proxy --port=8080
```



Node Port

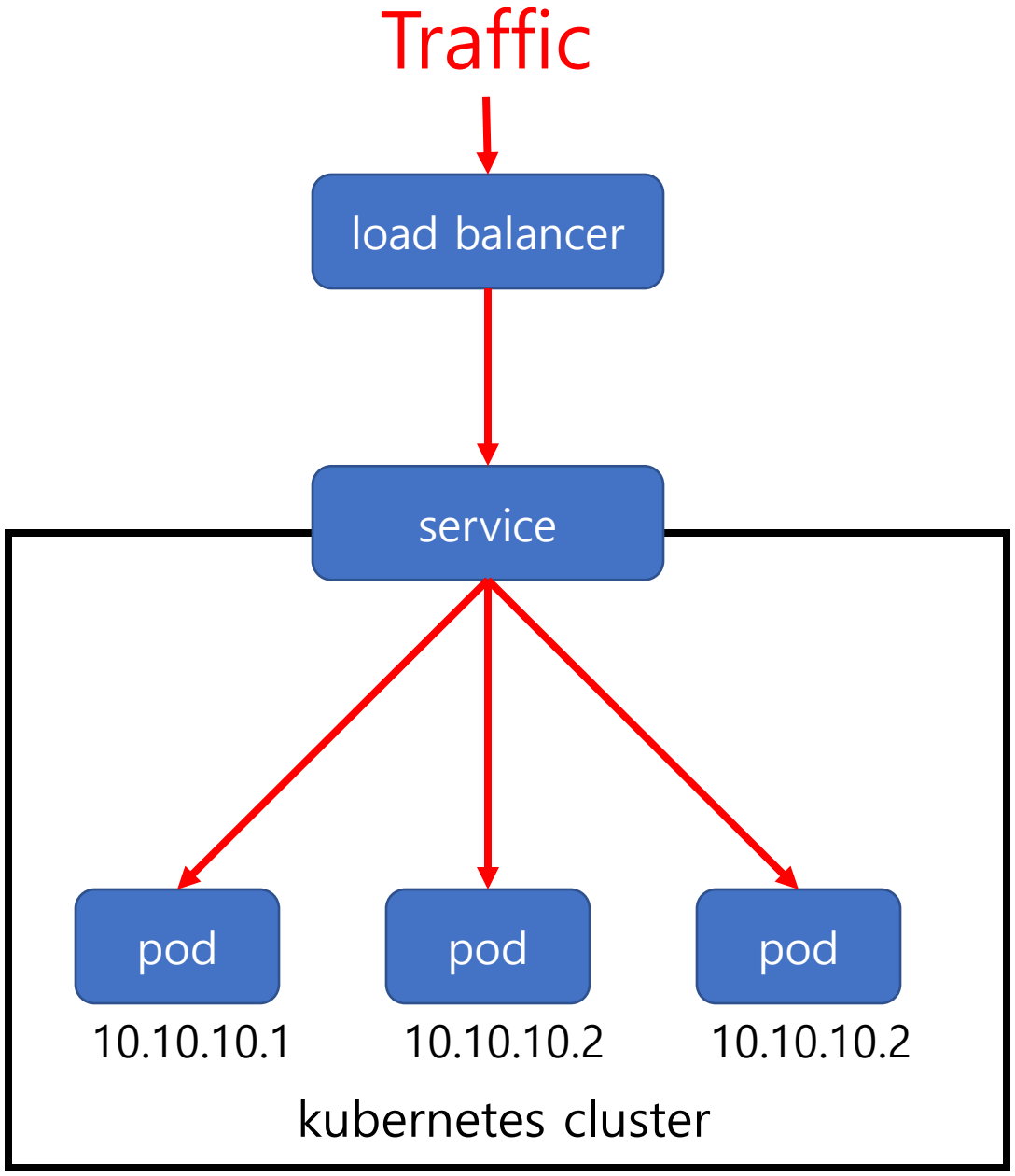
- NAT을 사용하여 각 노드의 Port 오픈
- 30000 ~ 32767 범위의 포트 할당
- 명시적 지정이 아니면 랜덤 할당

```
apiVersion: v1
kind: Service
metadata:
  name: my-nodeport-service
spec:
  selector:
    app: my-app
  type: NodePort
  ports:
  - name: http
    port: 80
    targetPort: 80
    nodePort: 30036
    protocol: TCP
```

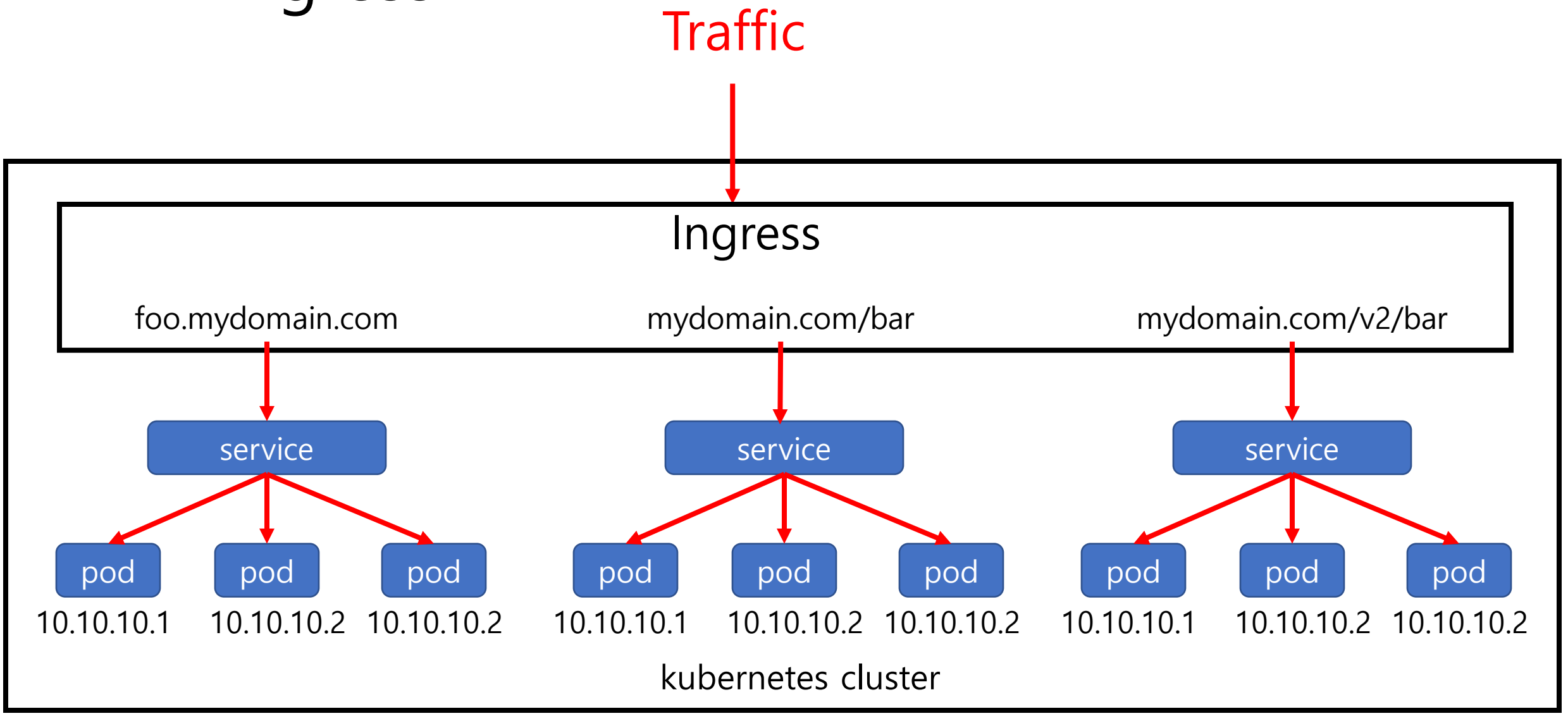


Load Balancer

- External 로드밸런서를 통해 고정된 외부 IP 배정



Ingress


Traffic
Ingress
foo.mydomain.com
mydomain.com/bar
mydomain.com/v2/bar
service
service
service
pod
pod
pod
pod
pod
pod
pod
pod
pod
10.10.10.1
10.10.10.2
10.10.10.2
10.10.10.1
10.10.10.2
10.10.10.2
10.10.10.1
10.10.10.2
10.10.10.2
kubernetes cluster

라벨 (label)

- key-value 쌍의 객체 집합이며 객체의 식별에 사용
- Pod를 그룹화할 때 사용
- 복제 컨트롤러(Replication Controller), 복제 집합(Replica set) 등에서 사용

라벨 셀렉터

- 라벨을 기반으로 객체를 선택
- 동일한 라벨이 부여된 객체들을 연산자를 통해 일괄 선택 가능
  - ex) 라벨 키가 role이고 값이 webserver인 객체 선택
    - role = webserver
      - = 또는 ==, != 을 통해 라벨에 따른 객체 선택
    - 복수의 조건은 쉼표로 구분
      - role = webserver, application != foo
    - 집합은 다양한 값을 기반으로 객체 선택
      - role in (webserver, backend)