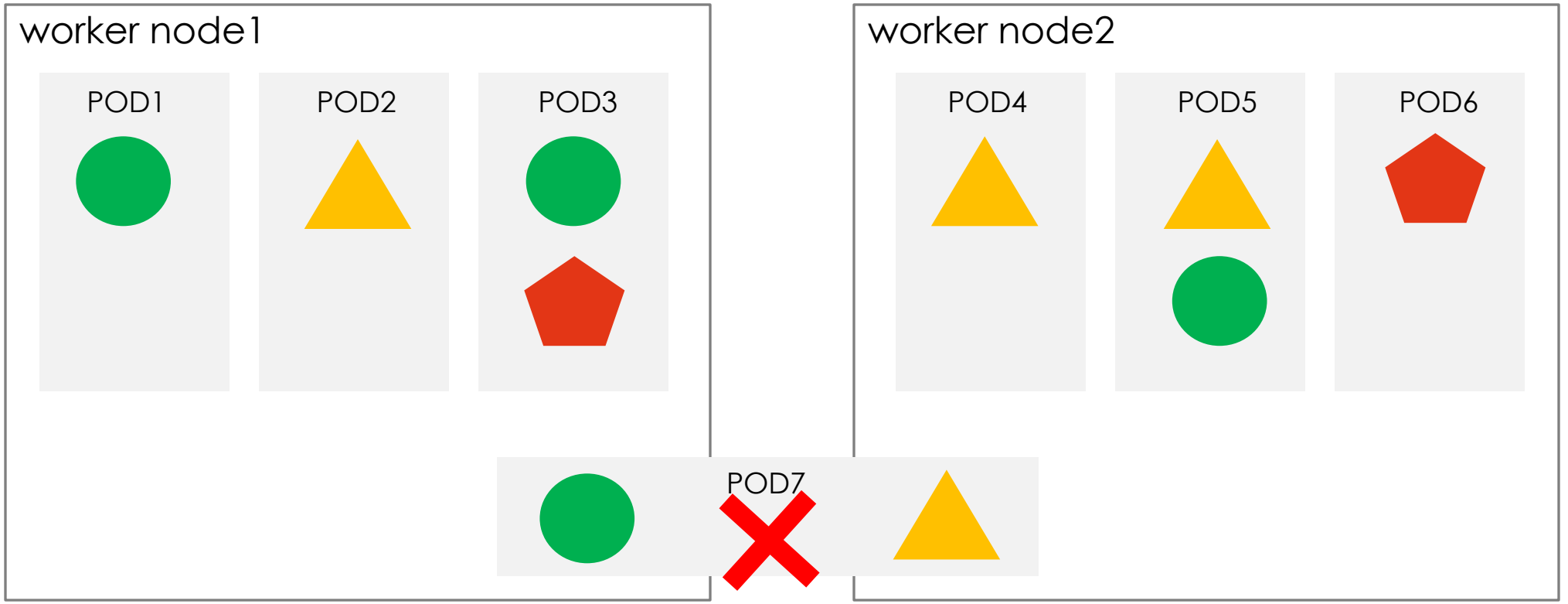
05. POD

POD

- POD(파드) 는 쿠버네티스 애플리케이션의 기본 실행 단위 (만들고 배포할 수 있는 가장 작은 단위)
- 도커는 쿠버네티스 파드에서 사용되는 가장 대표적인 컨테이너 런타임이지만, 파드는 다른 컨테이너 런타임 역시 지원한다. (rtk, containerd)
- 파드당 컨테이너 비율은 대체로 1:1 이지만, 파드당 여러개의 컨테이너가 포함되는 경우도 있음
- 파드내의 컨테이너는 오로지 하나의 노드 내에만 존재 합니다. 노드를 걸쳐서 파드가 존재 하지 않음
- 동일 파드 내의 컨테이너는 네트워킹 및 볼륨을 공유 합니다.



Kubernetes Network Basic

- K8S에 있는 POD들은 단순하고, 공유 가능한 네트워크 Address 주소 값을 가진다 (flat Natwork)
- 각각의 POD은 각각의 IP주소 값을 가지고 있으며, 이 IP를 이용해서 통신 허용
- NAT(Network Address Translation) gateway 같은 장비 없이- 마치 Local Area Network(LAN)처럼 통신이 가능.
- POD 내의 여러개 Container 는 서로 다른 포트를 통해 서비스 해야함

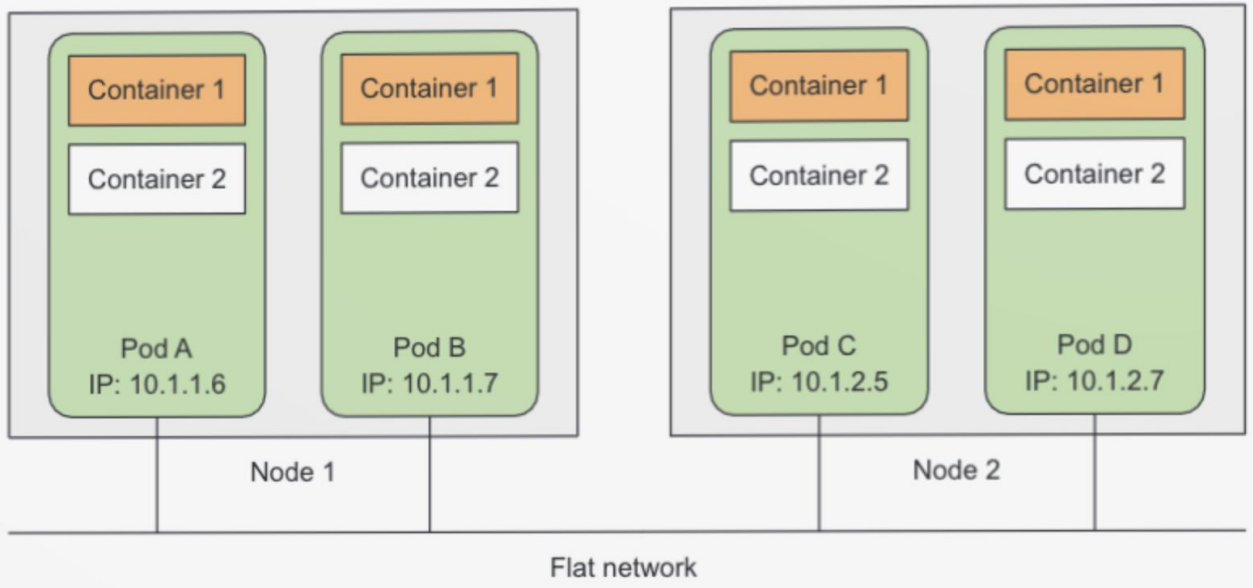


Figure 3.2 Each pod gets a routable IP address and all other pods see the pod under that IP address.

k8s 설치시에. pod network 지정 가능
--pod-network-cidr=192.168.0.0/16

Kubernetes Network Basic

POD 기본 명령

- pod 보기 명령어

```
# 기본 pod 조회 명령어
] kubectl get pods
# 축약어 사용
] kubectl get po
# 상세 정보까지 출력
] kubectl get po -o wide
# 레이블 까지 출력
] kubectl get po --show-labels
```

- K8s CLI를 이용한 POD 생성

```
# POD와 함께 Replication Controller 까지 생성 (Deprecate 될 예정)
] kubectl run <POD-NAME> --image=<IMAGE-NAME> --port=<SERVICE-PORT>
] kubectl run <POD-NAME> --image=<IMAGE-NAME> --port=<SERVICE-PORT>
```

POD 기본 명령

- pod 보기 명령어

```
# 기본 pod 조회 명령어
] kubectl get pod goapp-project-bcv5q -o yaml
# 축약어 사용
] kubectl create -f goapp.yaml
# 상세 정보까지 출력
] kubectl logs goapp-pod
```

- K8s CLI를 이용한 POD 생성

```
# POD와 함께 Replication Controller 까지 생성 (Deprecate 될 예정)
] kubectl run <POD-NAME> --image=<IMAGE-NAME> --port=<SERVICE-PORT> --generator= run/v1
# POD만 생성
] kubectl run <POD-NAME> --image=<IMAGE-NAME> --port=<SERVICE-PORT> --generator= run-pod/v1
```

POD 생성 Template 사용하기

```
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
  labels:
    app: myapp
spec:
  containers:
  - name: myapp-container
    image: busybox
    command: ['sh', '-c', 'echo 안녕하세요 쿠버네티스! &&
sleep 3600']
```

Key	설명
apiVersion	k8s api version
kind	k8s 리소스 타입
metadata	이름,레이블 등의 부가정보
spec	컨테이너 정보

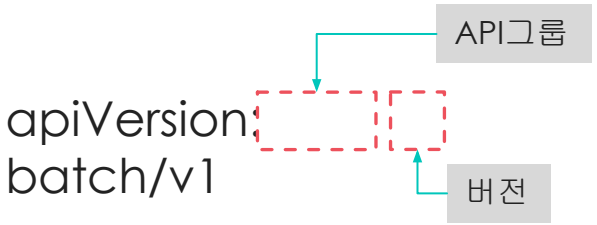
일반적으로 kubectl run 을 사용하는 것보다 실제 운영
환경에서는 yaml 파일을 이용해서 생성 합니다.
무엇보다 pod 에 대한 이력을 관리 하는 것이 중요

```
# yaml 파일을 이용해서 생성 하기
] kubectl create -f goapp.yaml
```

API 그룹

API 그룹은 쿠버네티스 API를 더 쉽게 확장하게 해준다.

Group	Version
admissionregistration.k8s.io	v1, v1beta1
apiextensions.k8s.io	v1, v1beta1
apiregistration.k8s.io	v1, v1beta1
apps	v1
authentication.k8s.io	v1, v1beta1
authorization.k8s.io	v1, v1beta1
autoscaling	v1, v2beta2, v2beta1
batch	v1, v1beta1
certificates.k8s.io	v1, v1beta1
coordination.k8s.io	v1, v1beta1
core	v1
discovery.k8s.io	v1, v1beta1
events.k8s.io	v1, v1beta1
extensions	v1beta1
flowcontrol.apiserver.k8s.io	v1beta1



단, core API 의 경우는
그룹을 명시 하지 않아도 됨

Group	Version
internal.apiserver.k8s.io	v1alpha1
networking.k8s.io	v1, v1beta1
node.k8s.io	v1, v1beta1, v1alpha1
policy	v1, v1beta1
rbac.authorization.k8s.io	v1, v1beta1, v1alpha1
scheduling.k8s.io	v1, v1beta1, v1alpha1
storage.k8s.io	v1, v1beta1, v1alpha1

알파
(Alpha)
- 기본적으로 비활성화
- 활성화 하면 버그에 노출 될 수 있음
- 다음 릴리즈 에서 언제든지 삭제 되거나 변경 될 수 있음

(예: v1alpha1
)

베타
(Beta)
- 기본적으로 활성화 되어 있음. 코드테스트를 많이 했으며, 안전함
- 전반적인 기능에 대한 기술 지원이 중단되지 않음 (지속 가능함)
- 문법이나 사용 방식이 다음 릴리지 에서 바뀔 수 있음

(예: v2beta3
)

안정화
(Stable)
- 버전 이름이 vX 와 같이 표기 한다. 여기서 X 는 정수 이다.

(예: v1
)

생성된 POD에서 yaml 파일 정보 축출 하기

```
#] kubectl get pod <POD-NAME> -o yaml
```

Pod 삭제

```
# 특정 Pod 삭제
$ kubectl delete pod <Pod-Name>

# 현재 네임스페이스의 모든 Pod 삭제
$ kubectl delete pod --all

# 네임스페이스내의 거의 모든 리소스 삭제
$ kubectl delete all --all
```

all –all 옵션을 사용해서 삭제해도 지워지지 않는 리소스가 있음 (예: secret 등)

06. Label

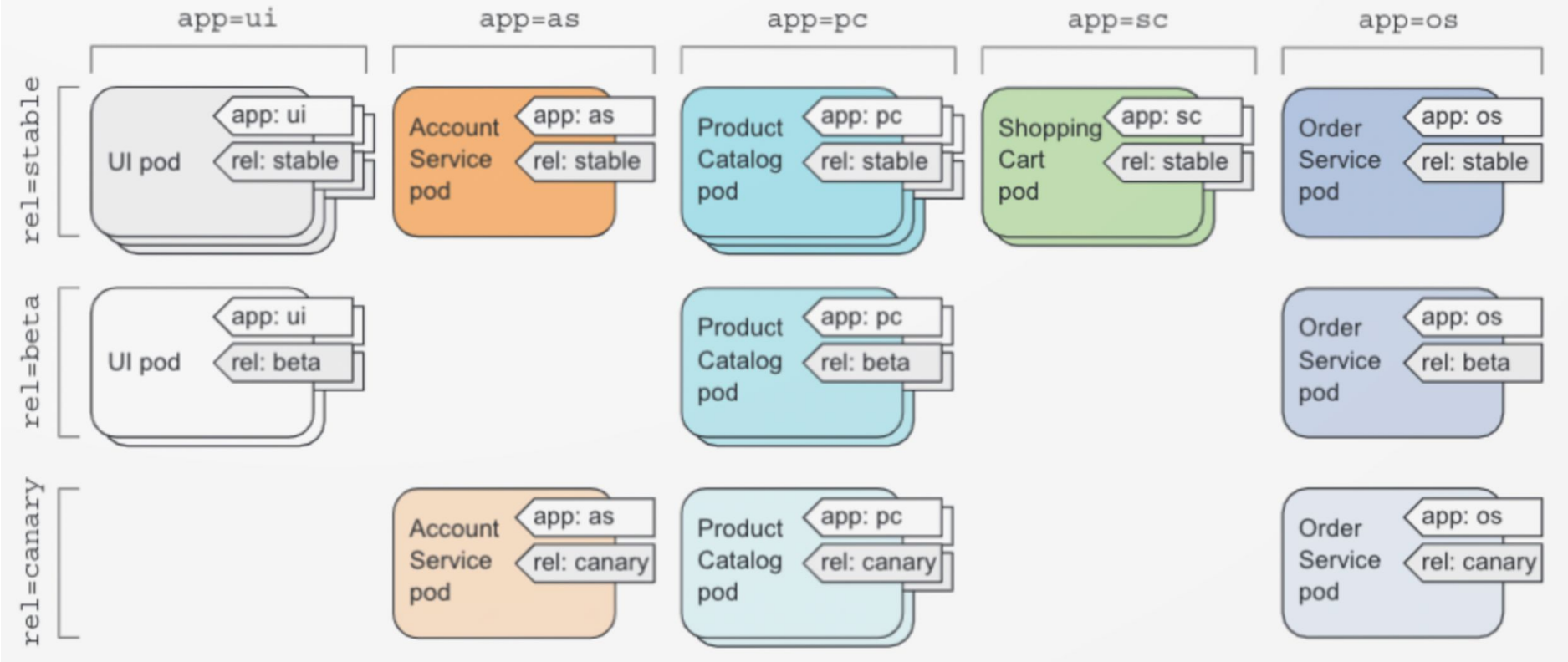
라벨 및 주석 기본

- 라벨 (Label) 이란?
  - 라벨은 Pod뿐 아니라 쿠버네티스의 모든 리소스를 구성하는 간단하고 강력한 기능
  - 리소스에 부여 하는 임의의 Key/Valye 쌍이다
  - 라벨 셀렉터를 사용해 리소스를 선택할 때 사용
  - 일반적으로 리소스를 만들 때 부여 하지만, 만들어진 후에도 라벨을 추가 하거나 변경 할수 있다.

- 주석 (Annotation) 이란?
  - Key/Value 쌍으로 라벨과 유사 하지만 셀렉터로 선택 불가능
  - 라벨보다 훨씬더 많은 정보를 담을 수 있음
  - 주로 주석을 사용해서 쿠버네티스 객체에 상세한 설명을 추가하여 다른 사람이 알아보기 쉽게 하는 용도
  - 예를 들면, 객체를 만든 사람이름을 기록하면 공동 작업 시에 훨씬 쉽게 작업

라벨의 실제 활용

- app는 Pod가 속한 애플리케이션, 구성요소 또는 마이크로 서비스 지정
- rel은 Pod에서 실행 중인 애플리케이션의 안정적 버전, 베타 혹은 카나리 버전인지 여부
- app과 rel 을 추가 하면서 Pod 들을 2가지 차원으로 구성



라벨의 지정 방법

- yaml 파일에 객체 선언 시에 metadata 내에 labels 키워드에 key/value 쌍으로 정의

```
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
  labels:
    app: myapp
    env: prod
    creation_method: manual
spec:
  containers:
  - name: myapp-container
    image: busybox
    command: ['sh', '-c', 'echo 안녕하세요 쿠버네티스! &&
sleep 3600']
```

라벨 추가 및 수정

```
# 기존 Pod 에 라벨 추가
$kubectl label po <Pod-Name> <Label-Key>=<Label-Value>
ex) kubectl label goapp-pod env=prod
```

```
# 기존 Pod 에 라벨 수정
$kubectl label po <Pod-Name> <Label-Key>=<Label-Value> --overwrite
ex) kubectl label nginx-pod rel=stable --overwrite
```

라벨 셀렉터를 사용한 Pod 보기

```
# 라벨 셀렉터를 이용한 Pod 조회
kubectl get pod -l <Label-Key>=<Label-Value>
ex) kubectl get pod -l env=prod
```

```
# 라벨 셀렉터로 Key 값만 이용해서 조회
kubectl get pod -l <Label-Key>
ex) kubectl get pod -l env
```

```
# 라벨 셀렉터로 연산자를 사용해서 조회
kubectl get po -l '!env' # env 라는 라벨Key 가 없는 Pod 조회
#다양한 연산식 예제
creation_method!=manual : manual 이외의 다른 값으로 creation_method 라벨이 있는 Pod 선택
env in (prod, dev) : env 라벨이 prod 또는 dev 로 설정된 모든 Pod를 선택
env notin (prod, dev) : env 라벨이 prod 또는 dev 가 아닌 다른 값으로 설정된 Pod 선택
```

Pod 스케줄링을 위한 라벨 활용

- 쿠버네티스에서는 어떤 노드로 포드가 스케줄 될지 알수 없음
- 라벨을 이용해서 특정 노드로 스케줄링 가능
- GPU 유무, 메모리(High-Memmory), SSD 유무 를 노드의 라벨에 적용

```
# 노드에 라벨 추가
$kubectl label node worker01.acorn.com ssd=true
# kubectl get nodes -l ssd=true
```

```
# SSD 가 있는 노드에 Pod 스케줄링
apiVersion: v1
kind: Pod
metadata:
  name: goapp-pod
spec:
  nodeSelector:
    ssd: "true"
  containers:
  - image: dangtong/goapp
    name: goapp
```

07. Annotation

주석 (Annotation) 사용법

- 250kb 까지 작성 가능하지만, 되도록 짧고 간결하게 사용 할것

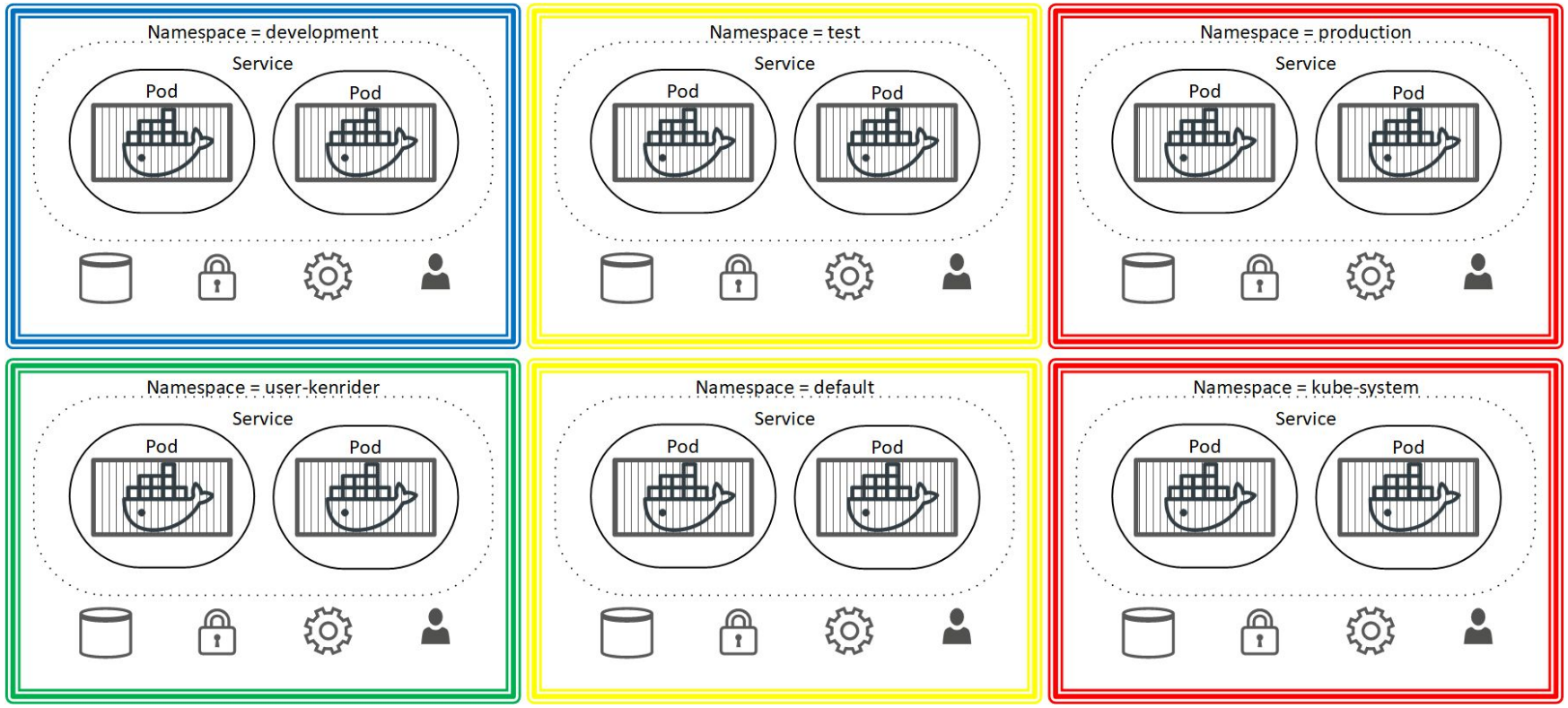
```
# 주석 추가
$kubectl annotate pod goapp-pod mycompany.com/created-by="dangtong byun"
```

```
# 주석 조회 (describe 명령어를 이용)
$kubectl describe pod goapp-pod
```

08. Namespace

라벨 셀렉터에서 다중조건 사용

- 쿠버네티스의 객체 이름의 범위를 제공하는것
- 동일한 리소스 이름을 여러 네임스페이스에서 여러 번 사용 가능(개발 / 검증/ 운영 에서 동일한 리소스 명 사용)
- 멀티-테넌트 환경으로 분리하는 효과



네임스페이스 사용법

```
# 네임스페이스 조회
$kubectl get ns
```

```
# 특정 네임스페이스에 있는 Pod 조회. (--namespace 대신 -n 을 사용 해도 됨)
$kubectl get pod -namespace kube-system
```

```
# 네임스페이스 yaml 작성
apiVersion: v1
kind: Namespace
metadata:
  name: prod-namespace
# 네임스페이스 생성
$kubectl create -f prod-namespace.yaml
```

```
# 명령어를 통한 네임스페이스 생성
$ kubectl create namespace prod-namespace
```

```
# 따른 네임스페이스 관리
$ kubectl create -f goapp-pod.yaml -n. dev-namespace
```

네임스페이스 전환

```
# 다른 네임스페이로 전환 명령어
$ kubectl config set-context $(kubectl config current-context) –namespace <Namespace-Name>
```

```
# 손쉽에 네임스페이스 전환을 위한 Alias 설정
$ alias kcd='kubectl config set-context $(kubectl config current-context) –namespace <Namespace-Name>'
# 쉽게 전환 하기
kcd namespace-prod
```