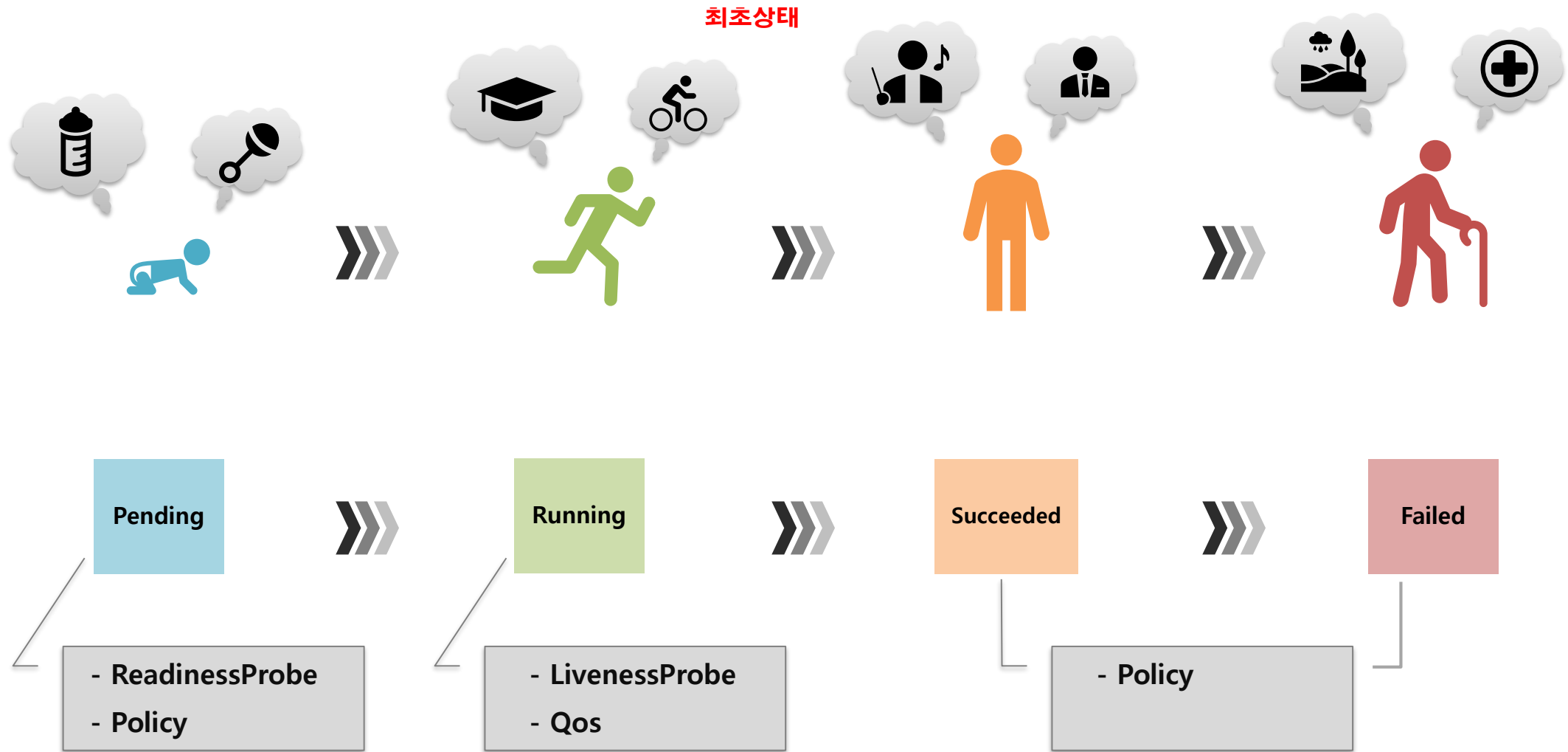


1. Pod - Lifecycle



1. Pod - Lifecycle

status:

phase: Pending

conditions:

- type: **Initialized**
status: 'True'
lastProbeTime: null
lastTransitionTime: '2019-09-26T22:07:56Z'
- type: **ContainersReady**
status: 'False'
lastProbeTime: null
lastTransitionTime: '2019-09-26T22:08:11Z'
reason: ContainersNotReady
- type: **PodScheduled**
status: 'True'
lastProbeTime: null
lastTransitionTime: '2019-09-26T22:07:56Z'
- type: **Ready**
status: 'False'
lastProbeTime: null
lastTransitionTime: '2019-09-26T22:08:11Z'
reason: ContainersNotReady

containerStatuses:

- name: container

state:

waiting:

reason: ContainerCreating

lastState: {}

ready: false

restartCount: 0

image: tmkube/init

imageID: "

started: false

status가 false일 경우 reason이 추가됨

Pod

Status

Pod의 전체 상태를 대표하는 속성

Phase	Pending	Running	Succeeded
	Failed	Unknown	

Pod가 실행하면서의 단계와 속성을 알려줌

Conditions	Initialized	ContainerReady
	PodScheduled	Ready
Reason	ContainersNotReady	
	PodCompleted	

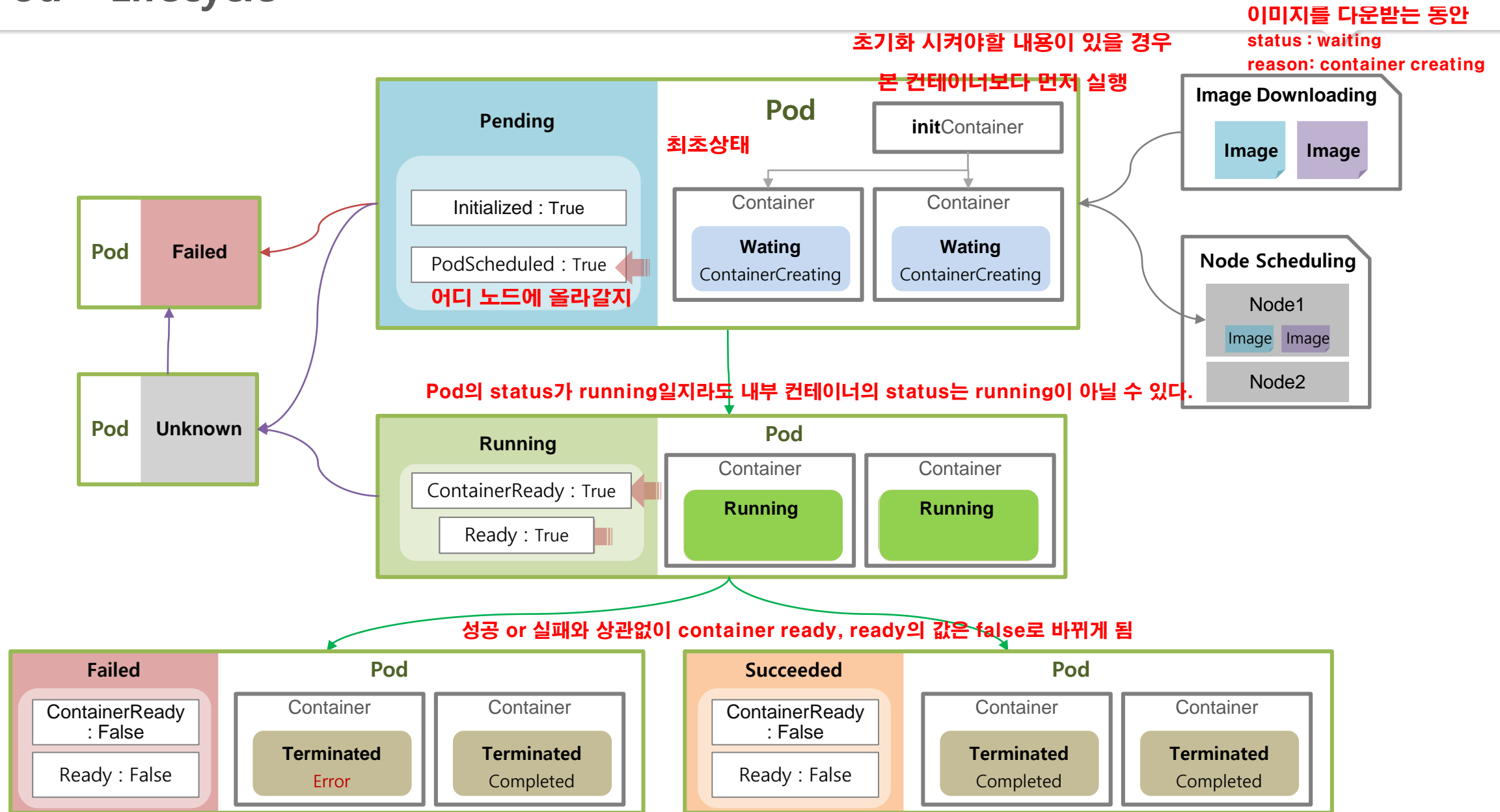
Containers

ContainerStatuses

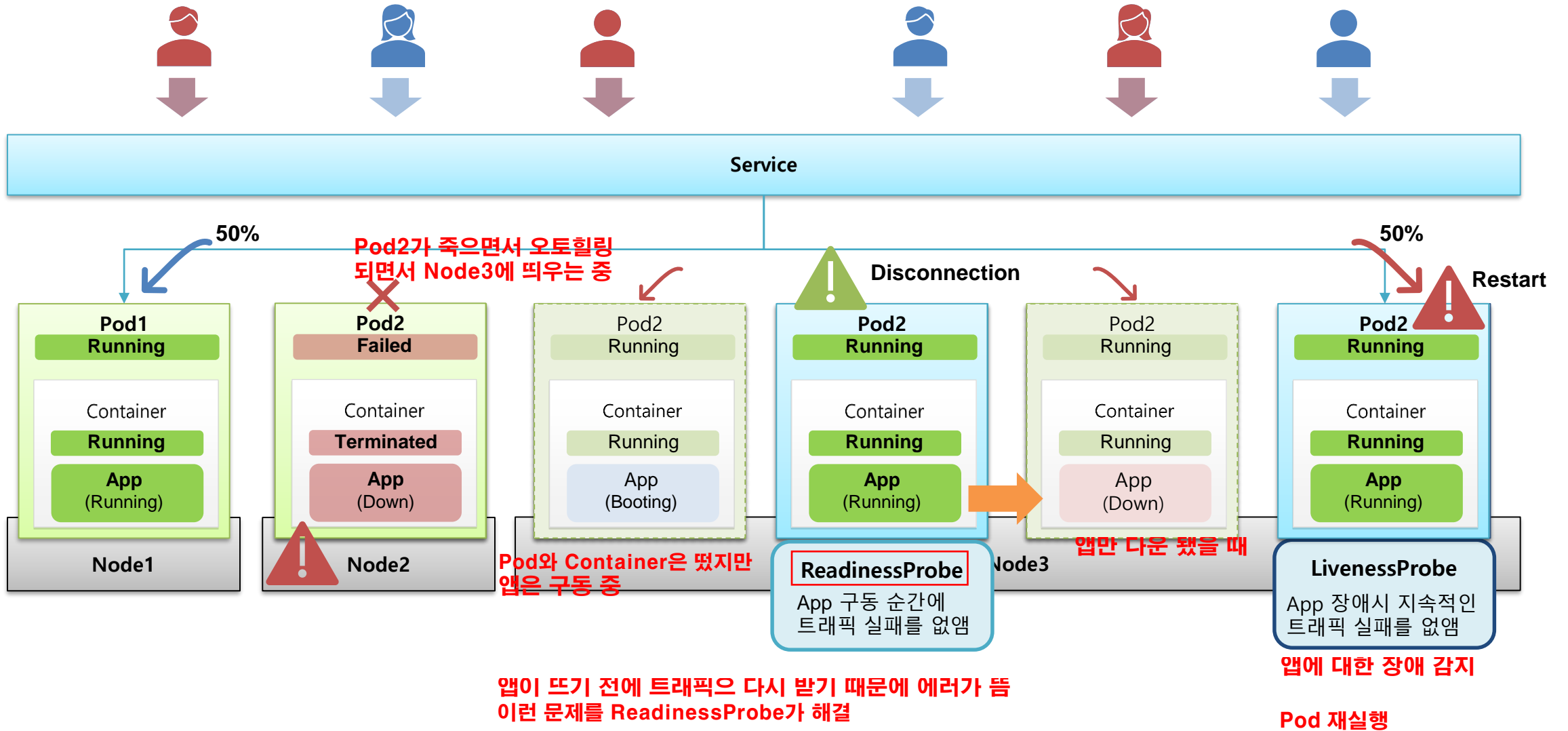
각 컨테이너마다 상태가 있음

State	Waiting	Running	Terminated
Reason	ContainerCreating	CrashLoopBackOff	
	Error	Completed	

1. Pod - Lifecycle



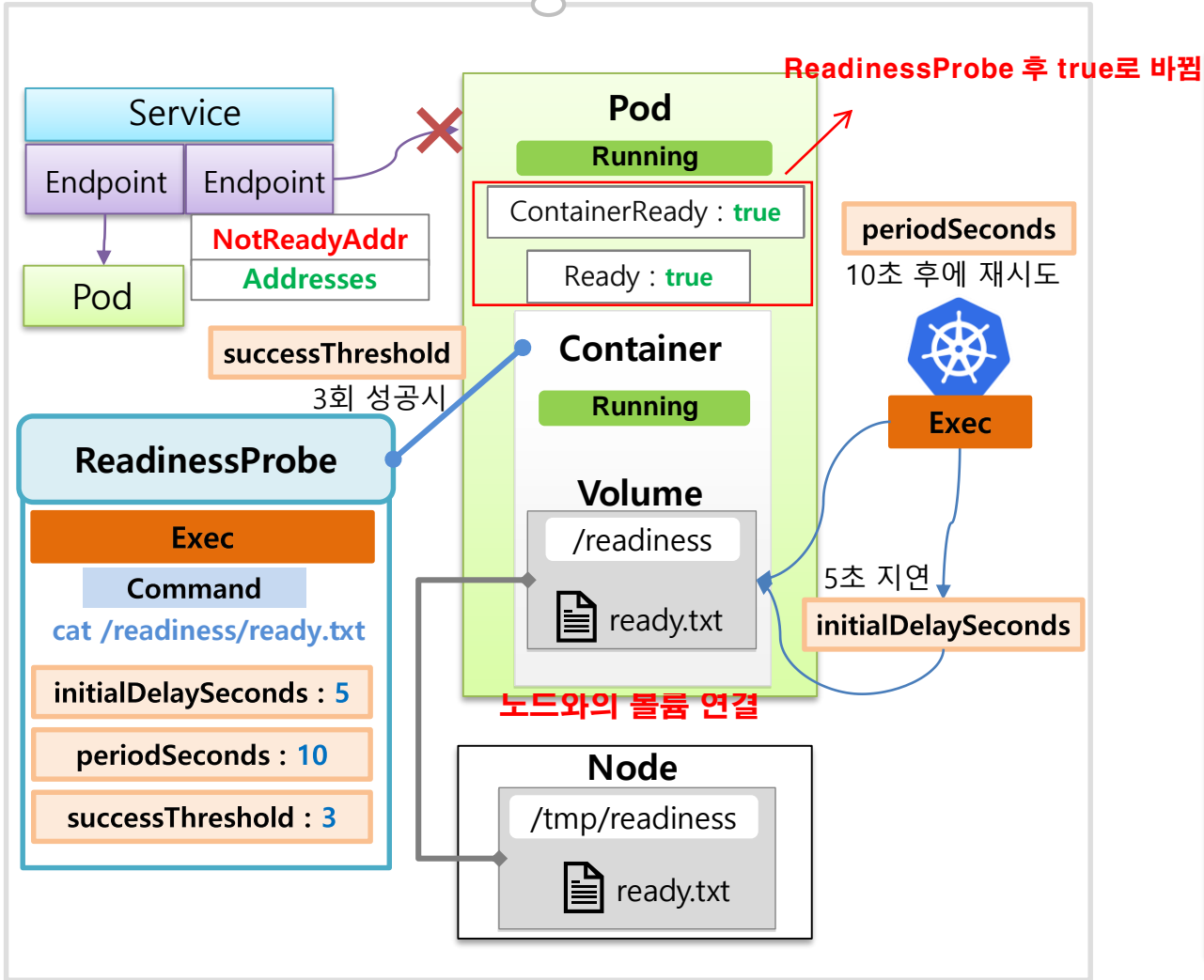
- Pod (ReadinessProbe, LivenessProbe)



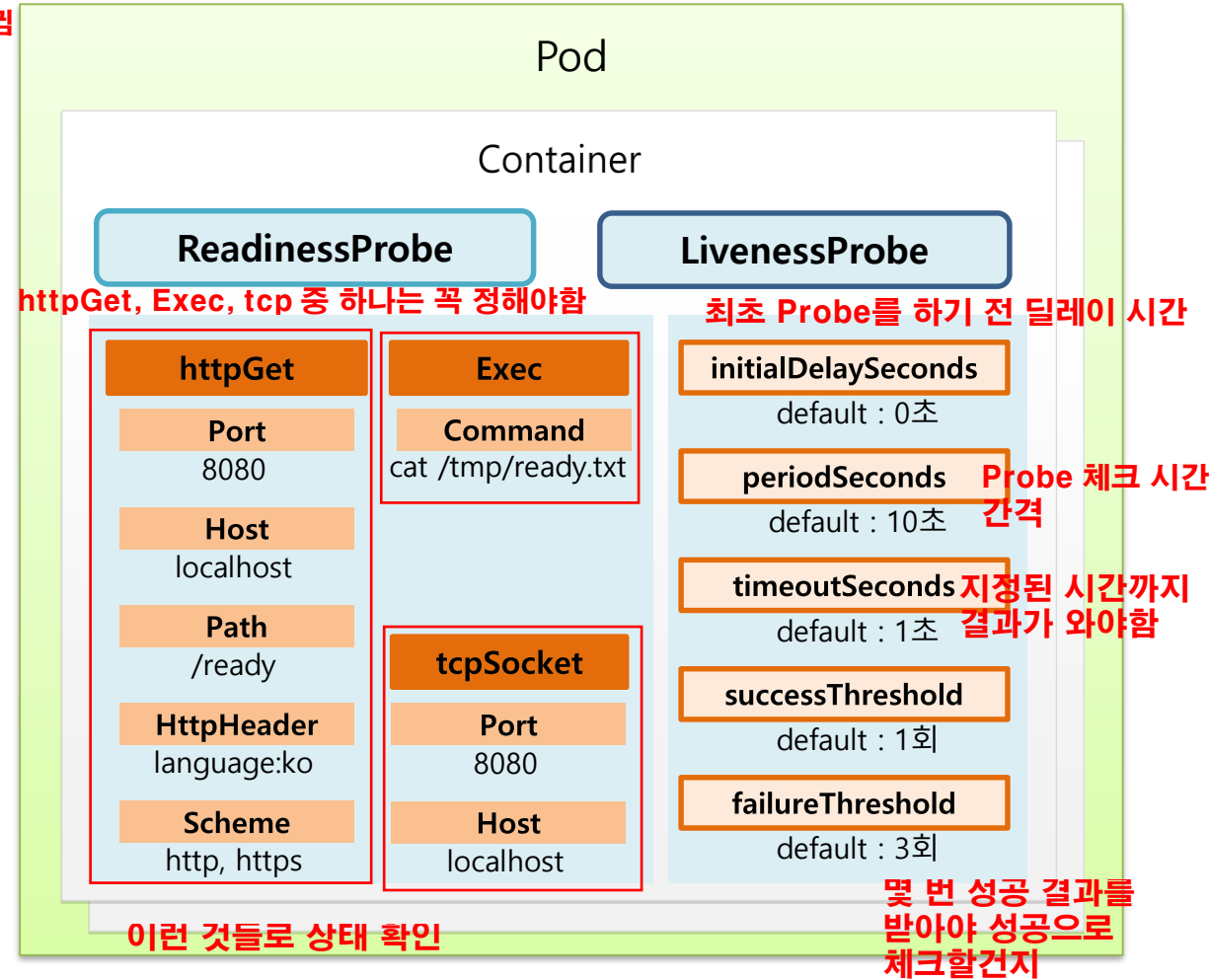
- Pod

ReadinessProbe를 알아보기 위한 그림

ReadinessProbe



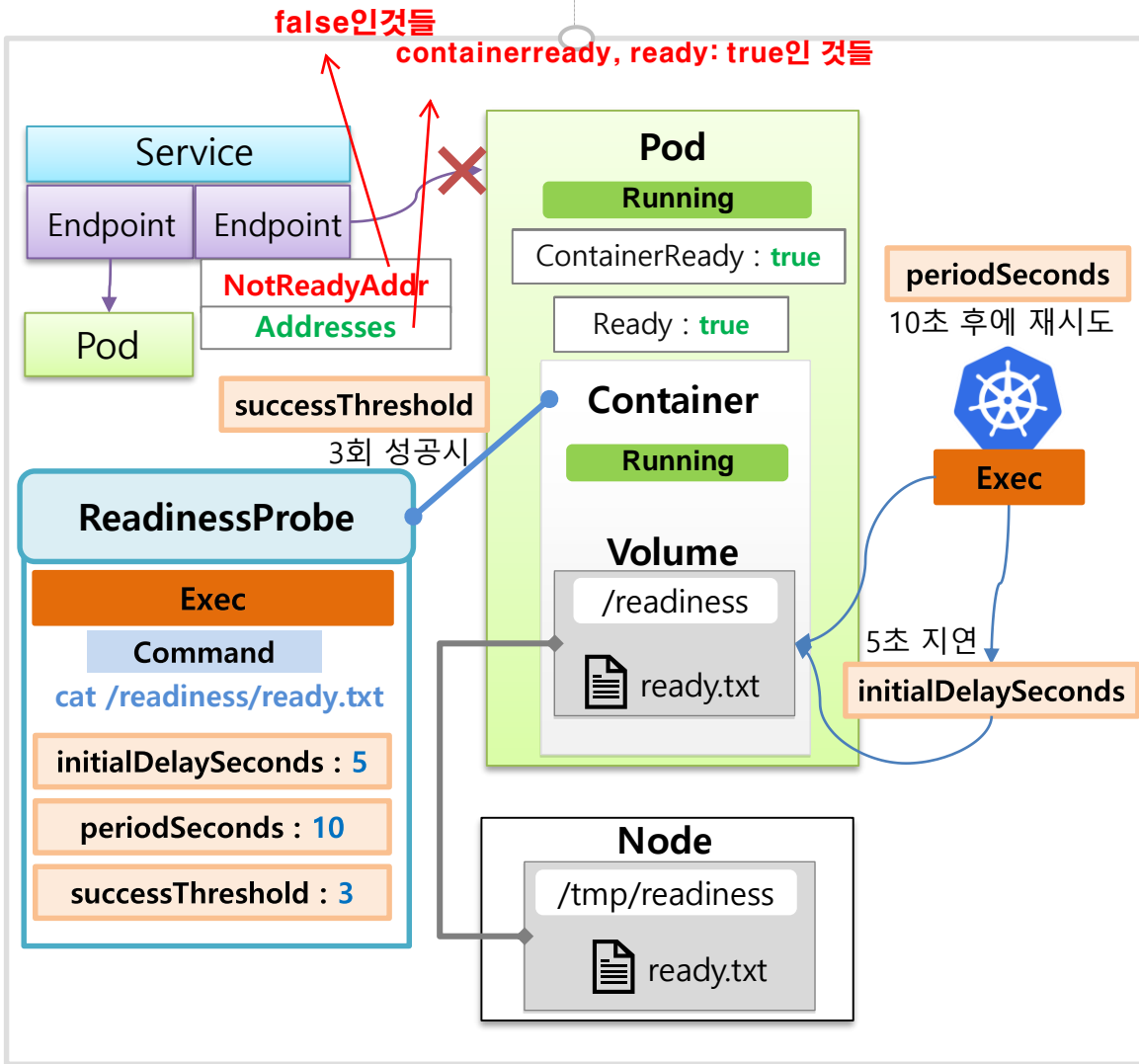
LivenessProbe



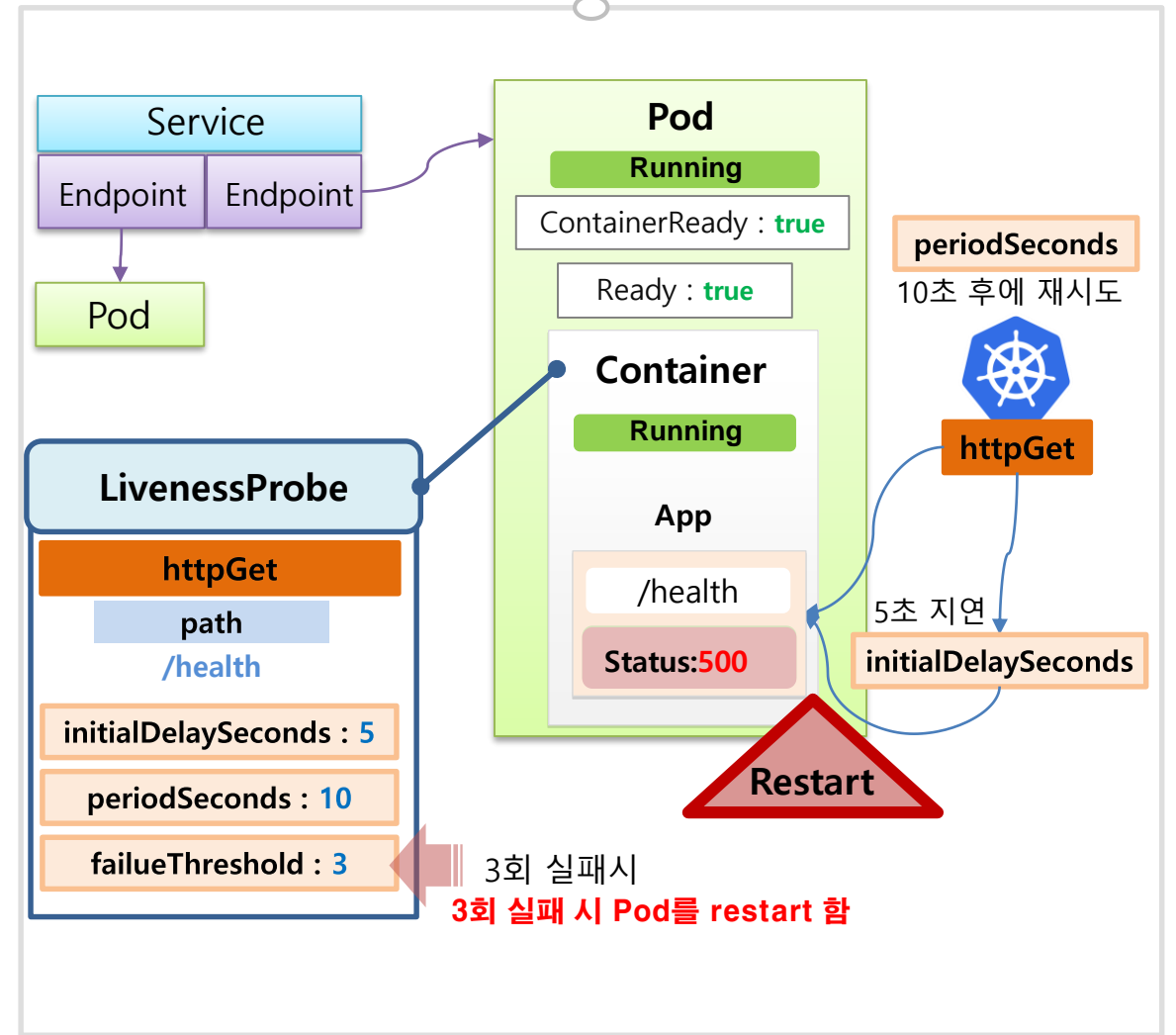
- Pod

LivenessProbe 를 알아보기 위한 그림

ReadinessProbe



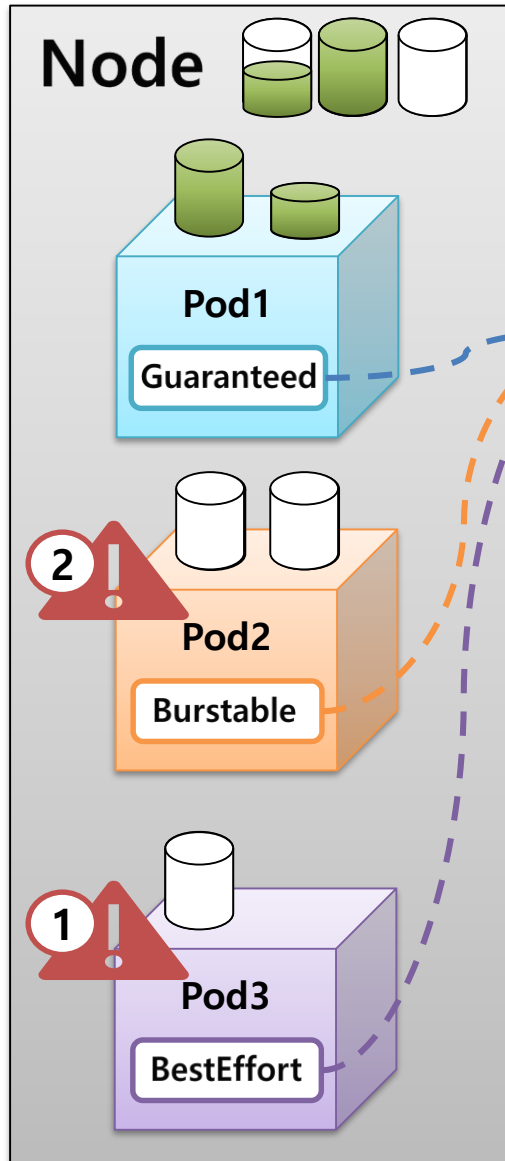
LivenessProbe



- QoS classes (Guaranteed, Burstable, BestEffort)

컨테이너마다 req, lim을 어떻게 설정하느냐에 따라 결정됨

퀄리티 오브 서비스



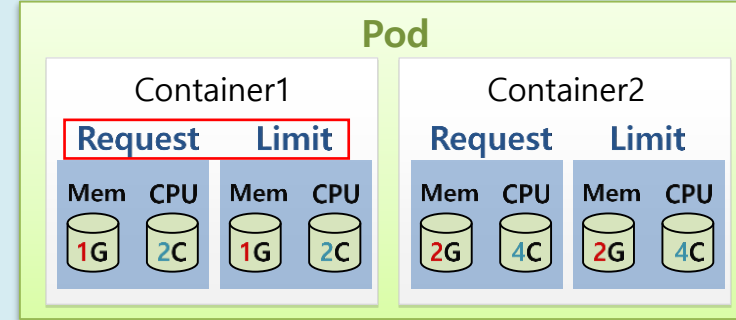
앱에 중요도에 따라 관리

QoS Classes

```
kind: Pod
spec:
  containers:
  - resources:
      requests:
        memory: 1G
        cpu: 2
      limits:
        memory: 2G
        cpu: 4
```

Guaranteed

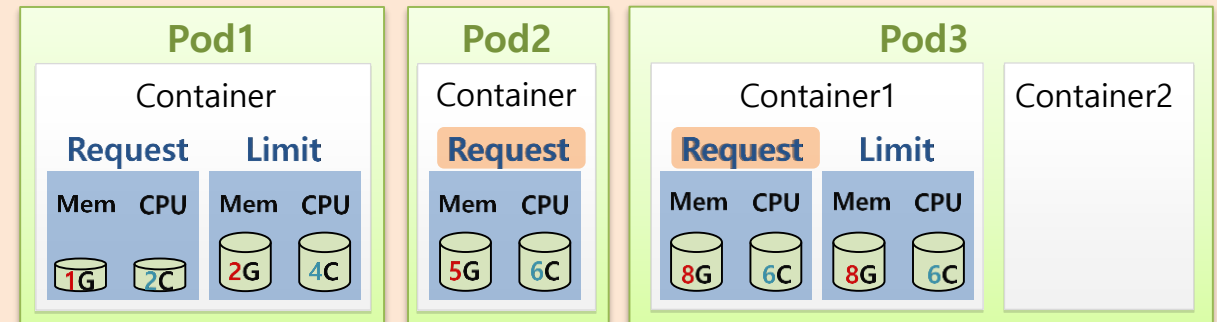
이 규칙이 다 맞으면 쿠버가 Guaranteed로 봄



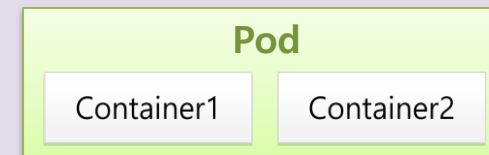
- 모든 Container에 Request와 Limit가 설정
- Request와 Limit에는 Memory와 CPU가 모두 설정
- 각 Container 내에 Memory와 CPU의 Request와 Limit의 값이 같음

Burstable

컨테이너마다 req, lim은 설정되어있지만, req < lim인 경우, 등등

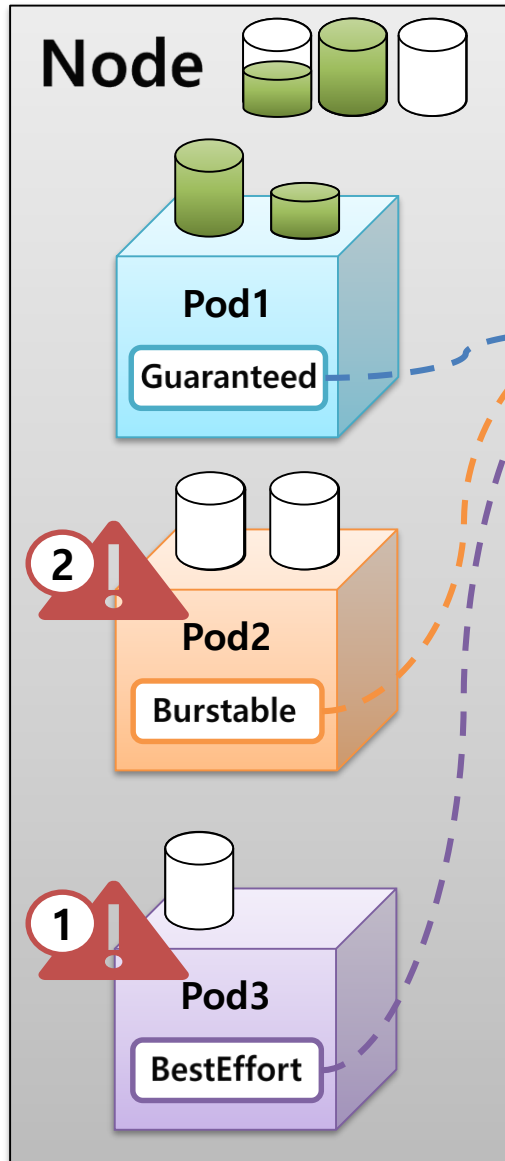


BestEffort



- 어떤 Container 내에도 Request와 Limit 미설정

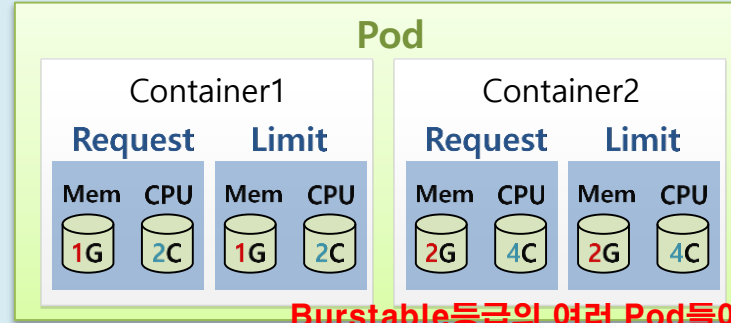
- QoS classes (Guaranteed, Burstable, BestEffort)



QoS Classes

kind: Pod
spec:
containers:
- resources:
 requests:
 memory: 1G
 cpu: 2
 limits:
 memory: 2G
 cpu: 4

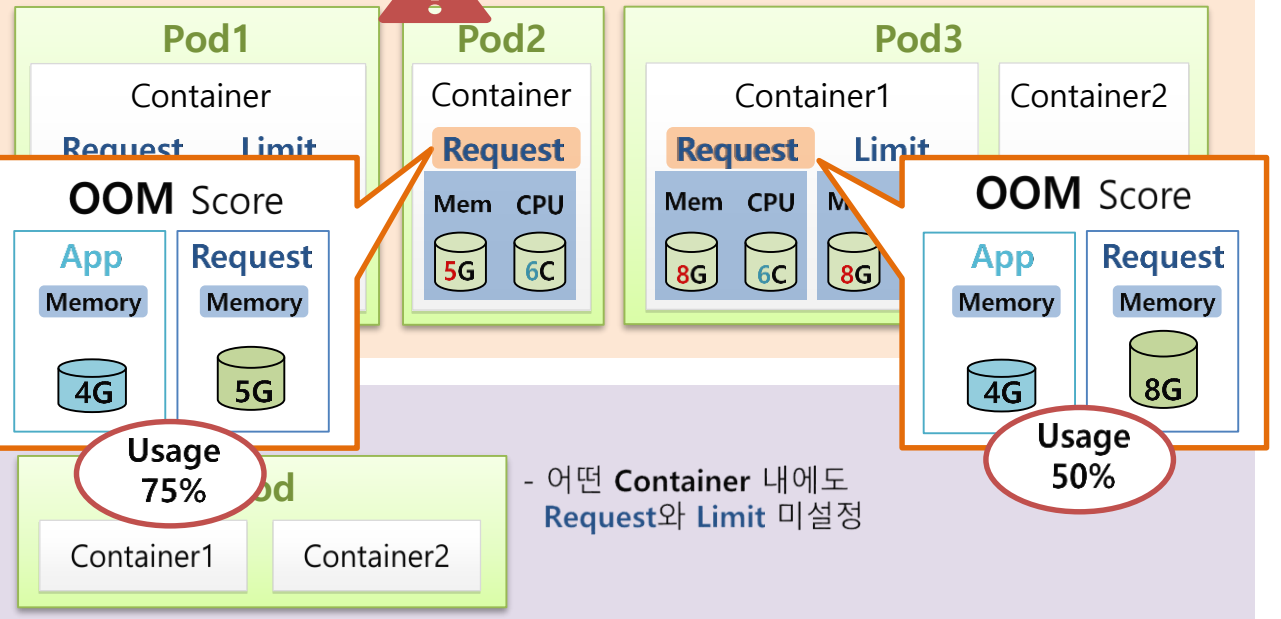
Guaranteed



- 모든 Container에 Request와 Limit가 설정
- Request와 Limit에는 Memory와 CPU가 모두 설정
- 각 Container 내에 Memory와 CPU의 Request와 Limit의 값이 같은

Burstable등급의 여러 Pod들이 있을 경우 누가 먼저 삭제가 되는지는 OOM Score에 따라 결정이 됨
메모리 사용량 (퍼센트)가 높은 Pod2를 먼저 제거함
사용량이 높은 Pod가 먼저 제거됨

Burstable

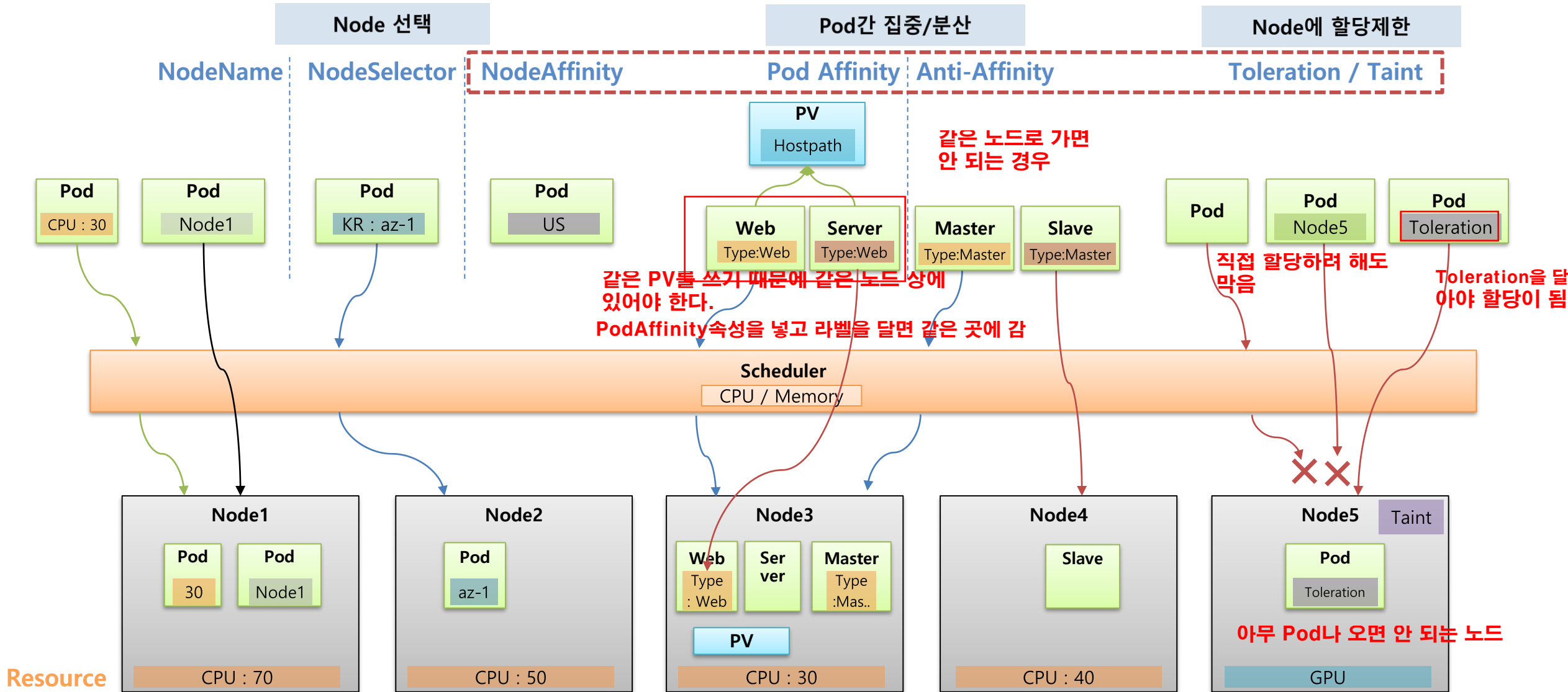


- 어떤 Container 내에도 Request와 Limit 미설정

조건에 맞지 않다면
스케줄러가 판단해서
자원이 많은 노드에
할당되도록 옵션 줄 수 있다.



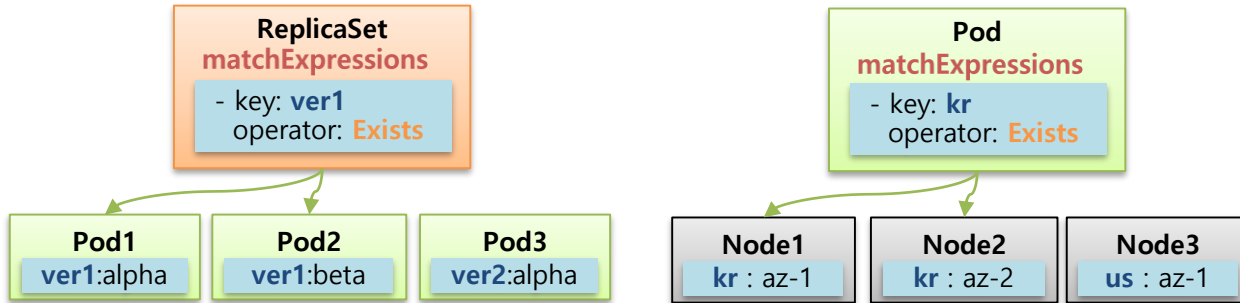
- Pod (Node Scheduling)



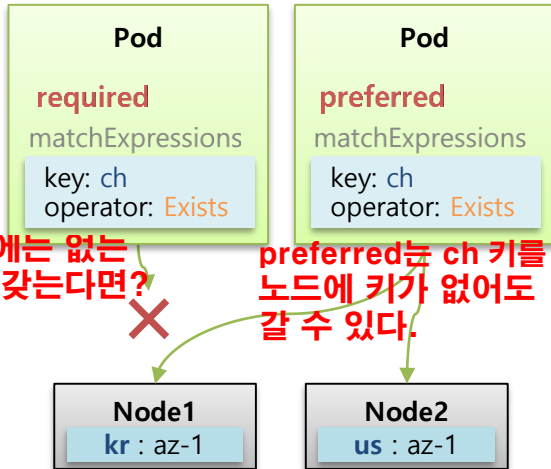
- Pod (Node Scheduling)

Node Affinity

matchExpressions



required vs preferred

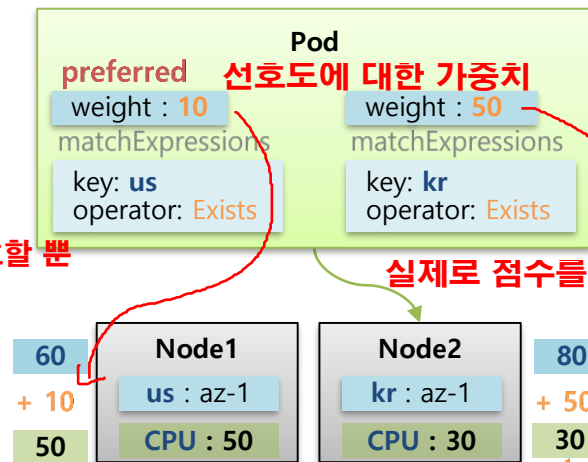


노드에는 없는 키를 갖는다면?



preferred는 ch 키를 선호할 뿐 노드에 키가 없어도 갈 수 있다.

preferred weight



선호도에 대한 가중치

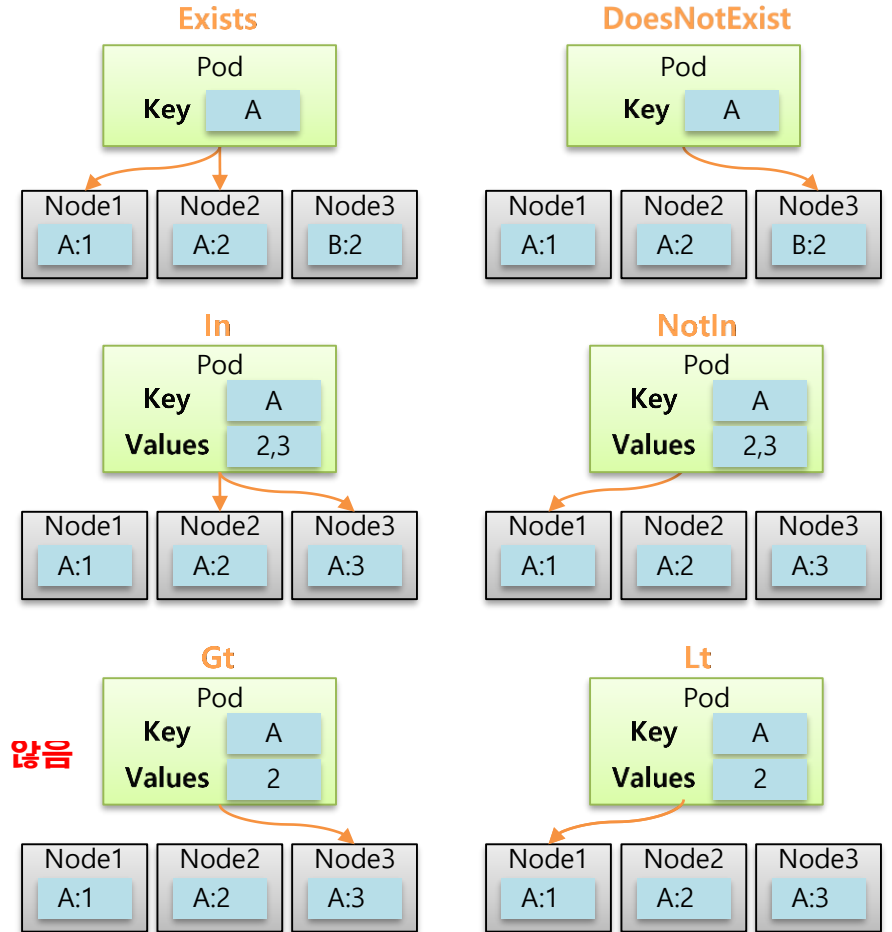
60
+ 10
50

80
+ 50
30

실제로 점수를 이렇게 매기진 않음

Scheduler

operator

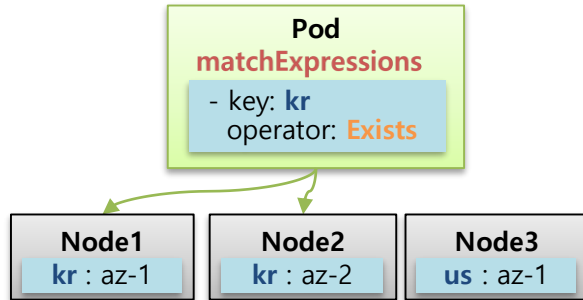


- Pod (Node Scheduling)

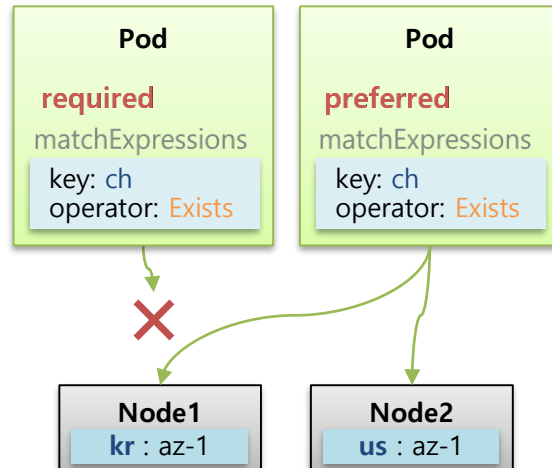
Toleration이 있다고 Node1에 가는데 아니라 Node1에 갔을 때 Toleration이 있어야 할당이 되는거임. 따라서 nodeSelector를 달아야 Node1에 o확정으로 갈 수 있음

Node Affinity

matchExpressions

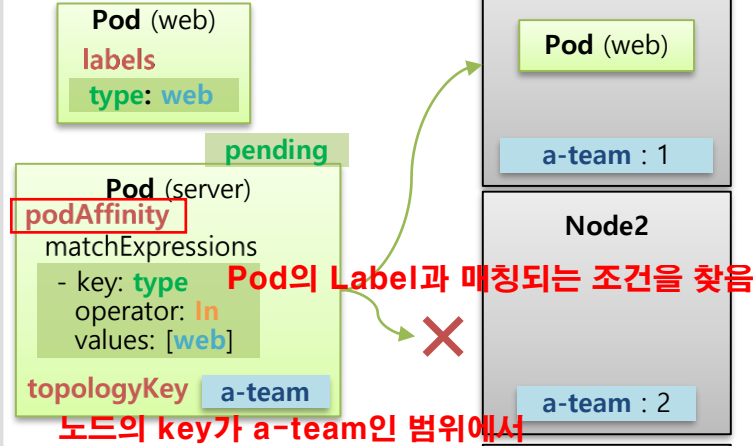


required vs preferred

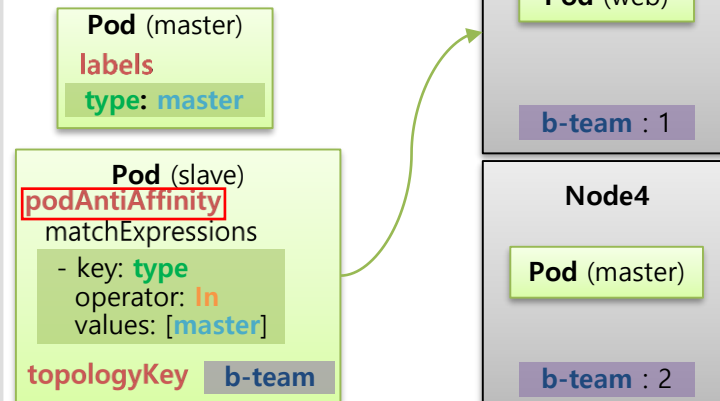


Pod Affinity

Pod Affinity



Pod Anti-Affinity



Taint, Toleration

Pod를 만들 때 Teleration을 주고 노드엔 Taint 설정

