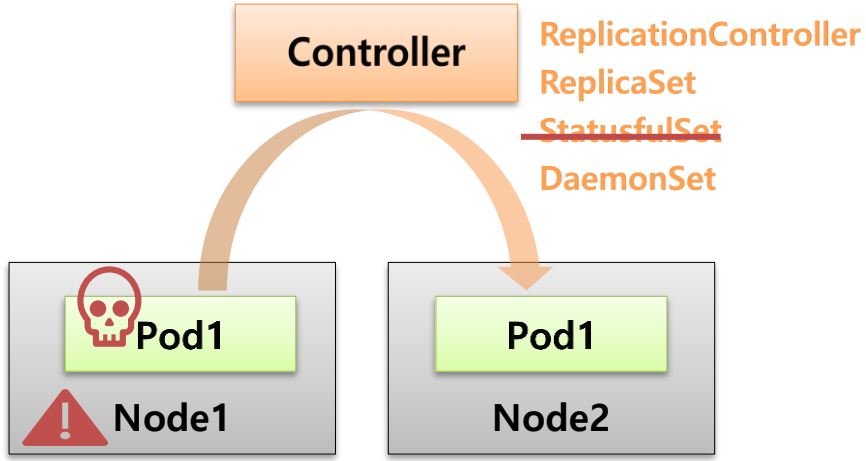
6. Controller

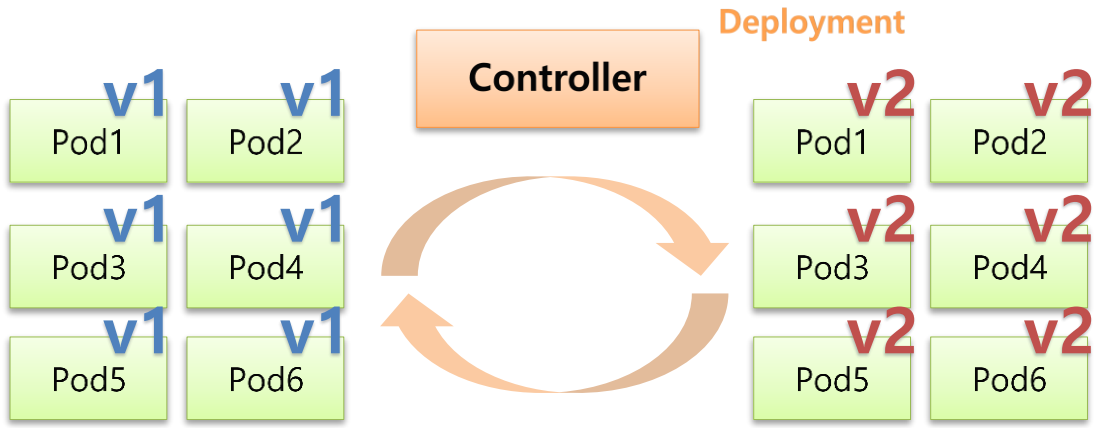
Auto Healing



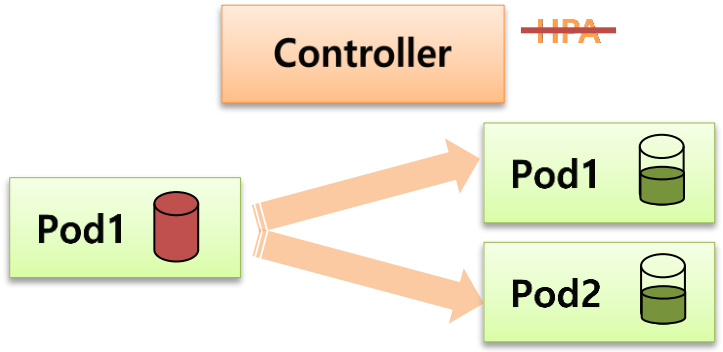
pod나 node가 죽으면 다른 node에 새로 만들어줌

Software Update

버전업, 롤백 제공

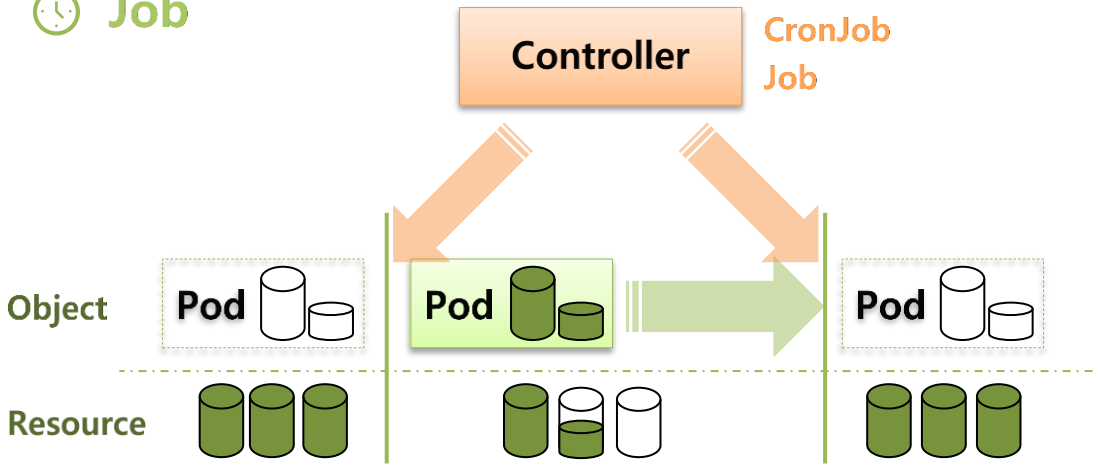


Auto Scaling



부하를 분산

Job



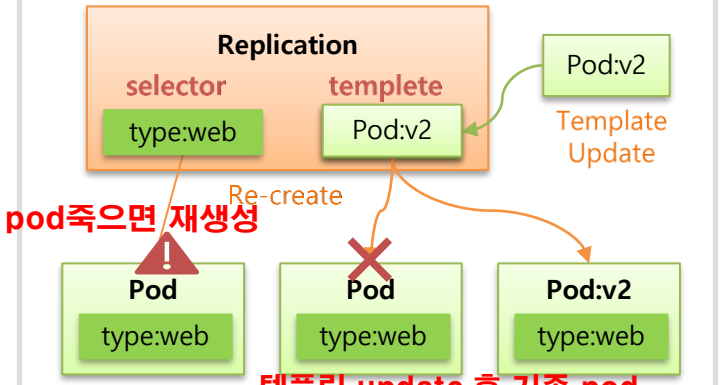
6. Controller - Replication Controller, ReplicaSet

Deprecated → Replaced

replicaset에만 있는 기능

Template

컨트롤러와 pod는 label과 selector로 연결



템플릿 update 후 기존 pod 죽이면 새로 생성되면서 버전 업글

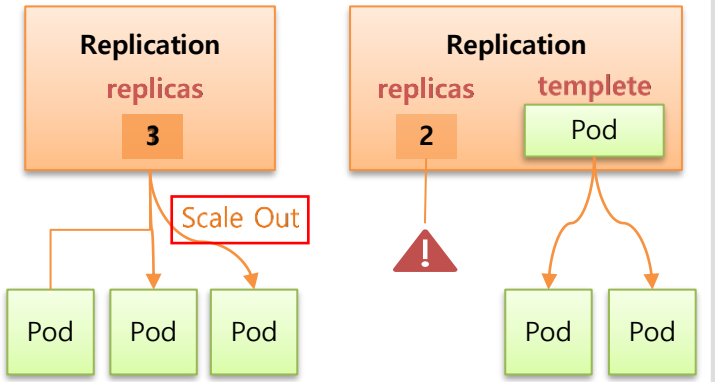
```
apiVersion: v1
kind: Pod
metadata:
  name: pod-1
  labels:
    type: web
spec:
  containers:
  - name: container
    image:
        tmkube/app:v1
```

```
apiVersion: v1
kind: ReplicationController
metadata:
 name: replication-1
spec:
 replicas: 1 -> 3
 selector:
   type: web
 template:
   metadata:
     name: pod-1
     labels:
       type: web
   spec:
     containers:
     - name: container
       image: tmkube/app:v2
```

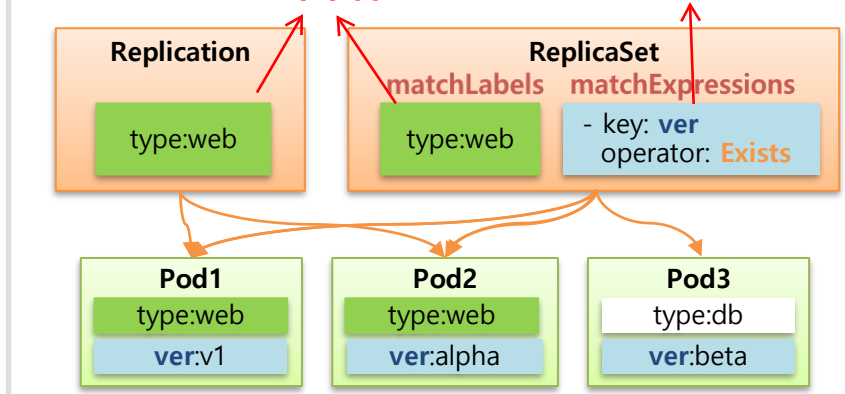
일치해야함

Replicas

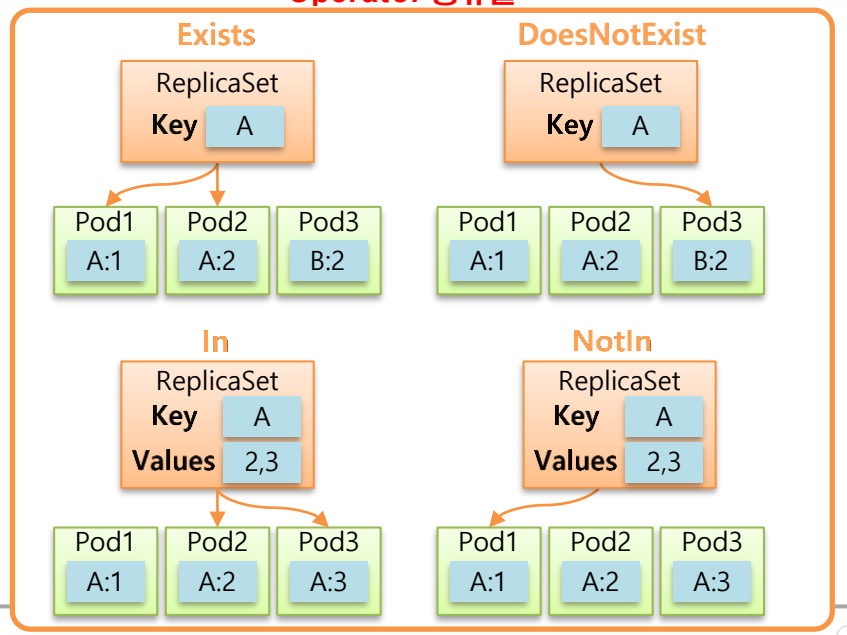
따로 controller와 pod를 만들지 않고 한번에 만들면 replicas의 수 만큼 pod를 만들어주기 때문에 하나만 만들어도 됨



Selector



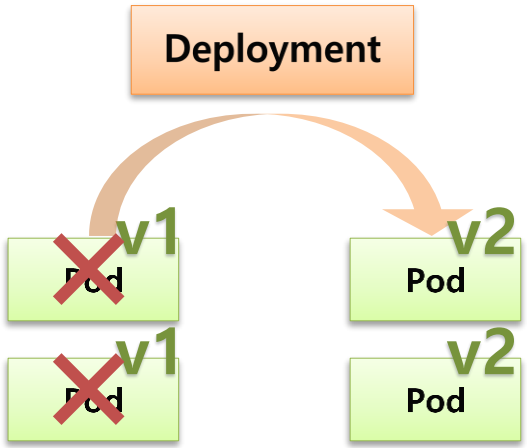
Operator 종류들



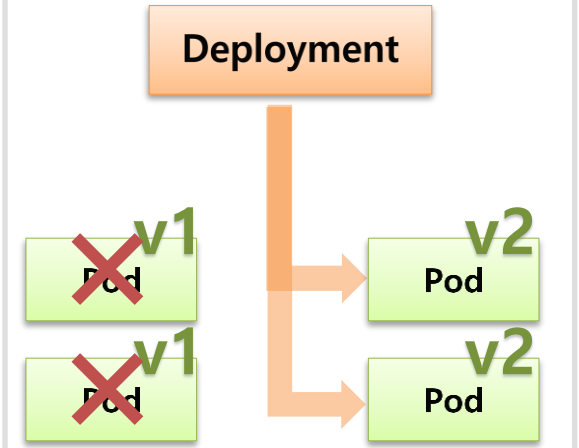

```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
 name: replica-1
spec:
 replicas: 3
 selector:
   matchLabels:
     type: web
   matchExpressions:
   - {key: ver, operator: Exists}
 template:
   metadata:
     name: pod
...
```

6. Controller - Deployment

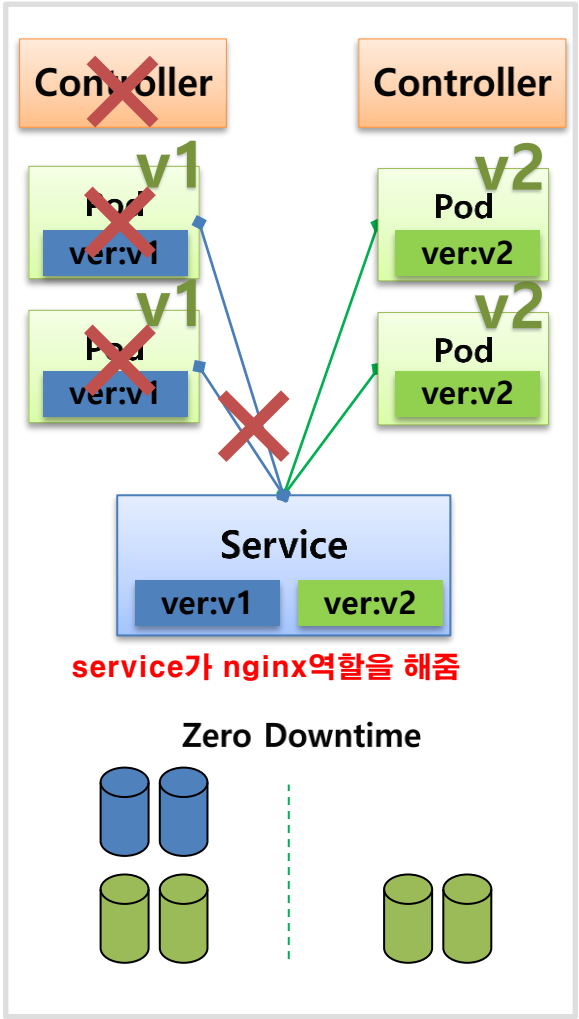
ReCreate



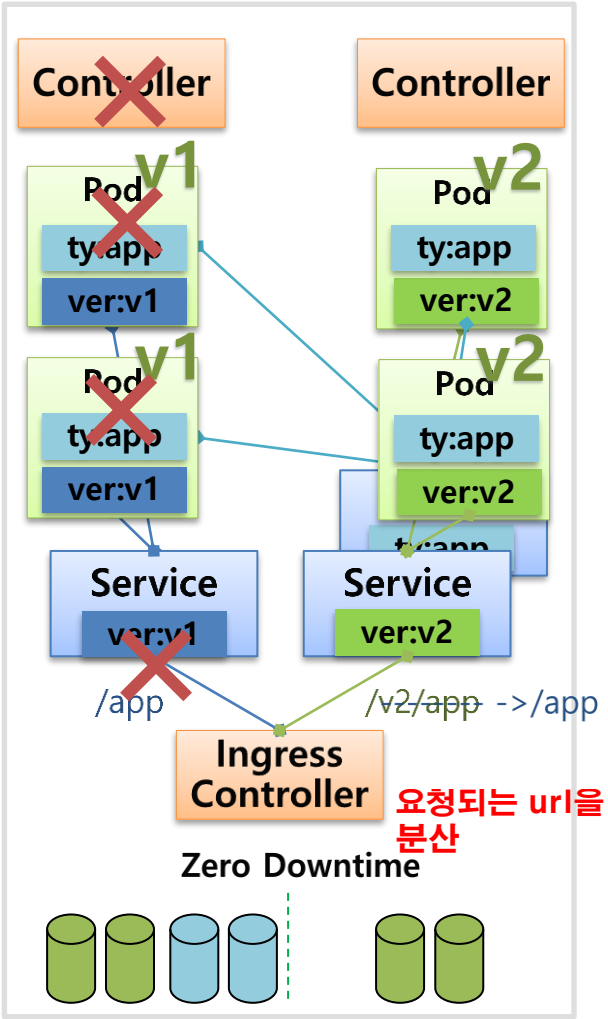
Rolling Update



Blue/Green



Canary



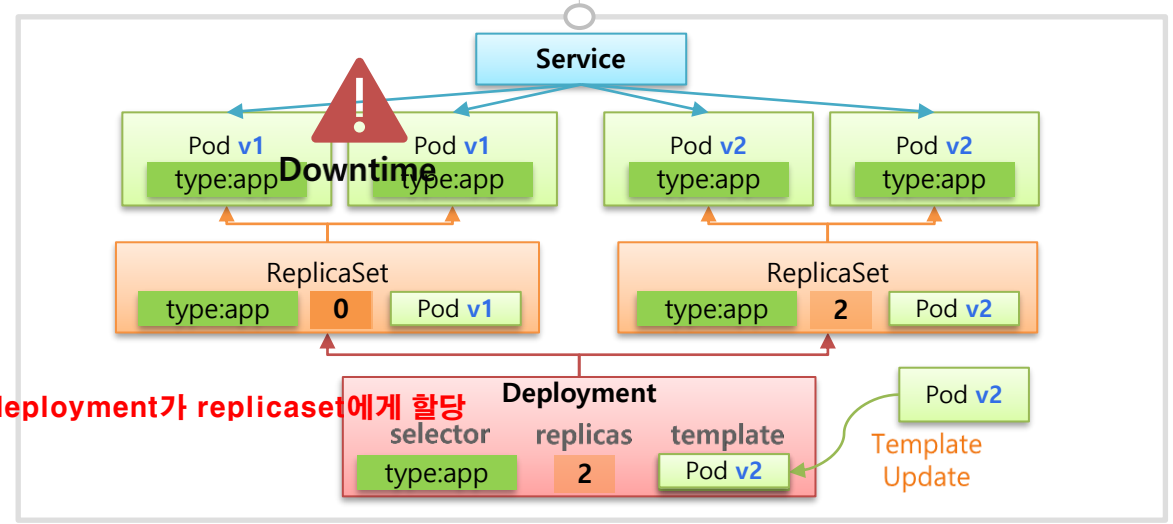
위험한지 검증하고 문제없다면 배포

6. Controller - Deployment

각각의 ReplicaSet은 자신들의 Pod를 구분하기 위해 별도의 Labels를 갖는다.

Pod에도 동일한 Labels가 있음

ReCreate



deployment가 replicaset에게 할당

```yaml
apiVersion: v1
kind: Service
metadata:
  name: svc-1
spec:
  selector:
    type: app
  ports:
    - port: 8080
      protocol: TCP
      targetPort: 8080
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: deployment-1
spec:
  selector:
    matchLabels:
      type: app
  replicas: 2
  strategy:
    type: Recreate
  revisionHistoryLimit: 1
  template:
    metadata:
      labels:
        type: app
    spec:
      containers:
      - name: container
        image: tmkube/app:v1
```

0인 replicaset을 하나만 남기게 함

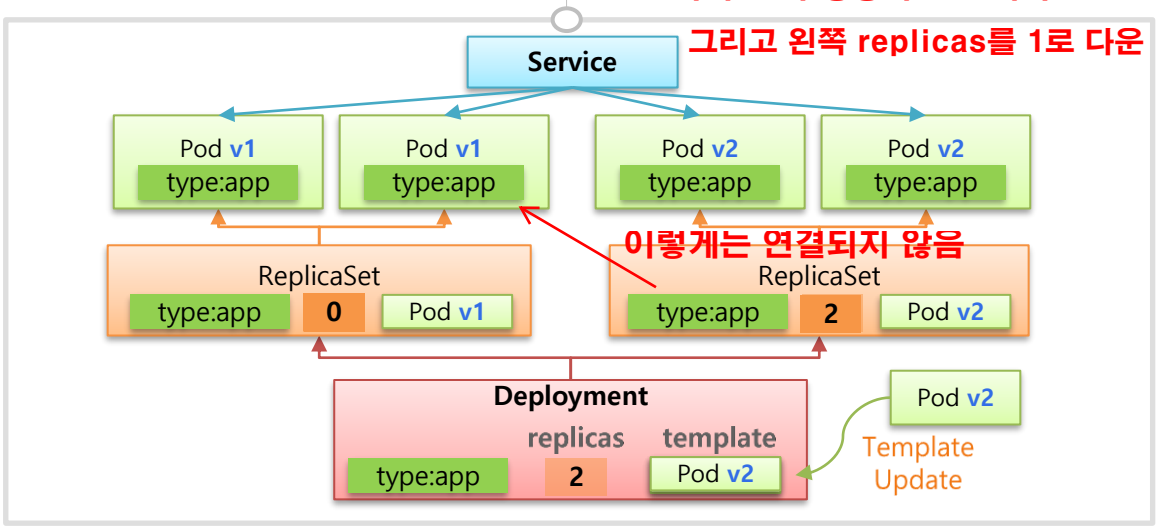
이걸 통해 이전 버전으로 돌아갈 수 있음

replicaset 기록을 볼 수 있는 개수 제한을 거는 옵션

Rolling Update (default)

v2하나 먼저 생성하고 트래픽 분산

그리고 왼쪽 replicas를 1로 다운




```yaml
apiVersion: v1
kind: Service
metadata:
  name: svc-2
spec:
  selector:
    type: app
  ports:
    - port: 8080
      protocol: TCP
      targetPort: 8080
```

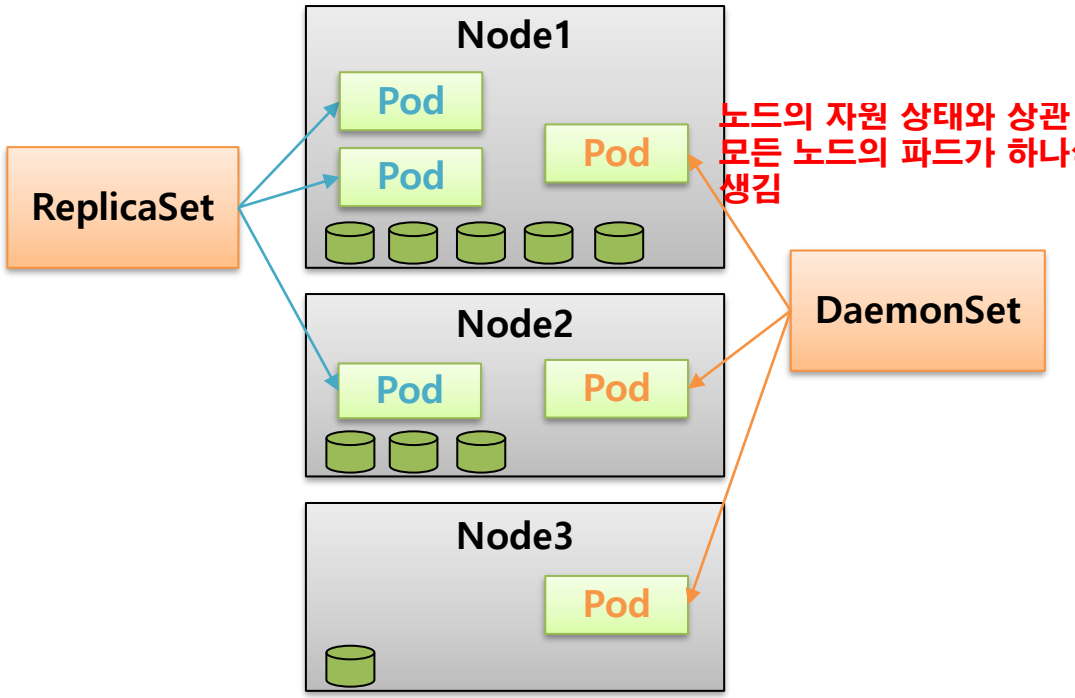
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: deployment-2
spec:
  selector:
    matchLabels:
      type: app
  replicas: 2
  strategy:
    type: RollingUpdate
  minReadySeconds: 10
  template:
    metadata:
      labels:
        type: app
    spec:
      containers:
      - name: container
        image: tmkube/app:v1
```

v2를 생성하고 v1을 지우는 시간

6. Controller - DaemonSet, Job, CronJob

누구에 의해 만들어진 Pod냐에 따라 .. 달라짐

DeamonSet



노드의 자원 상태와 상관 없이 모든 노드의 파드가 하나씩 생김

각각의 노드들이 하나씩 있어야 하는 서비스의 예

Performance — Prometheus

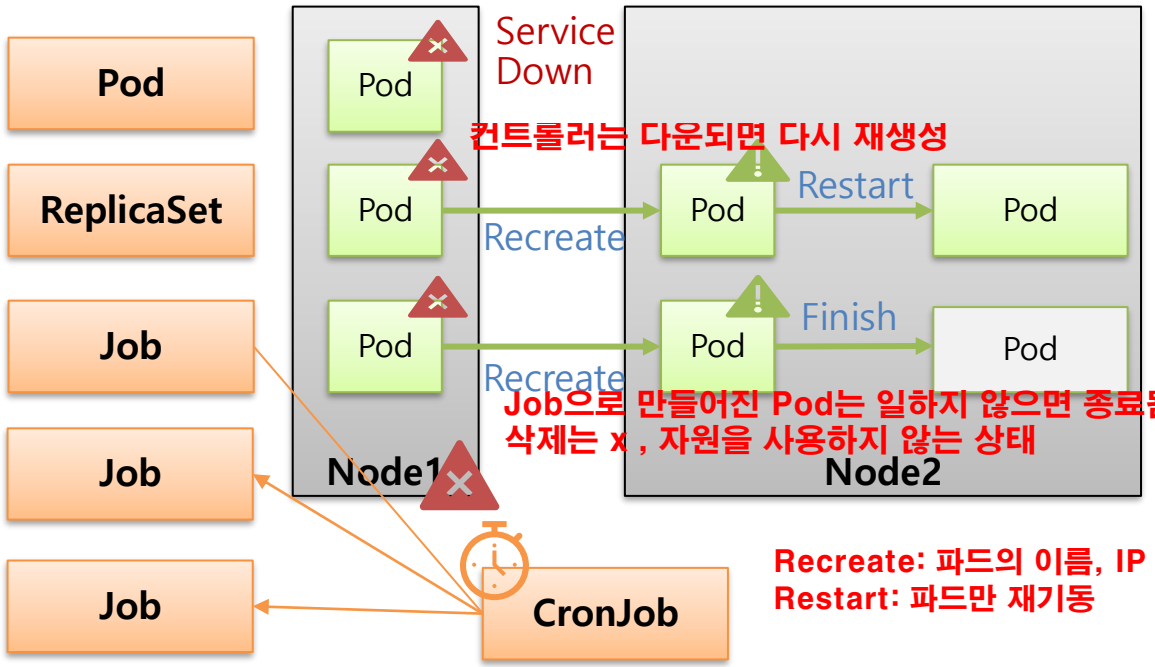
Logging — fluentd

Storage — GlusterFS

Job

CronJob

다운이 됐을 경우



컨트롤러는 다운되면 다시 재생성

Job으로 만들어진 Pod는 일하지 않으면 종료됨
삭제는 x , 자원을 사용하지 않는 상태

Recreate: 파드의 이름, IP 변경
Restart: 파드만 재기동

크론잡 예시

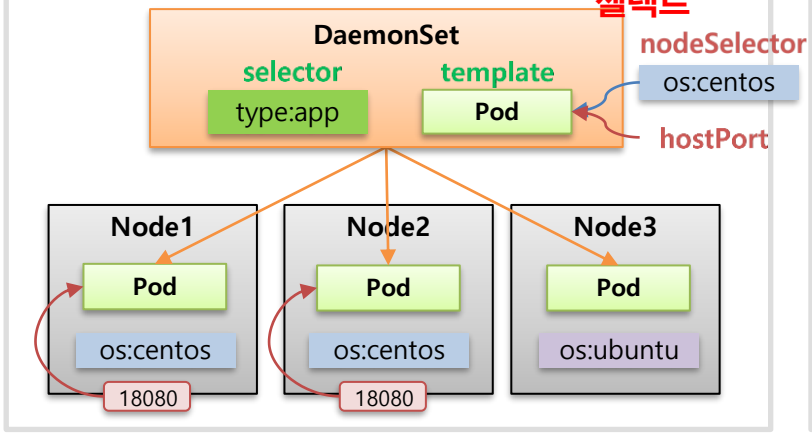
6. Controller - DaemonSet, Job, CronJob

데몬셋을 삭제하면 만들어진 Pod들도 삭제됨

셀렉터와 라벨로 Node, DemonSet연결

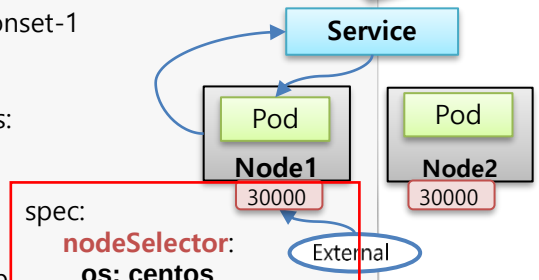
DaemonSet

os:centos가 있는 노드만 셀렉트




```
apiVersion: apps/v1
kind: DaemonSet
metadata:
 name: daemonset-1
spec:
 selector:
   matchLabels:
     type: app
 template:
   metadata:
     labels:
       type: app
```

externalTrafficPolicy: Local

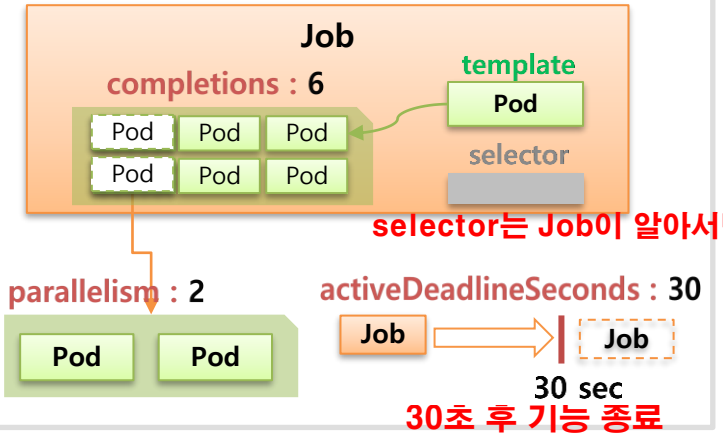



```
spec:
    nodeSelector:
      os: centos
    containers:
    - name: container
      image: tmkube/app
      ports:
      - containerPort: 8080
        hostPort: 18080
```

노드의 18080으로 들어온 port는 해당 8080포트(Pod)로 연결

Job

6개의 Pod를 순차적으로 실행, 종료되면 Job도종료



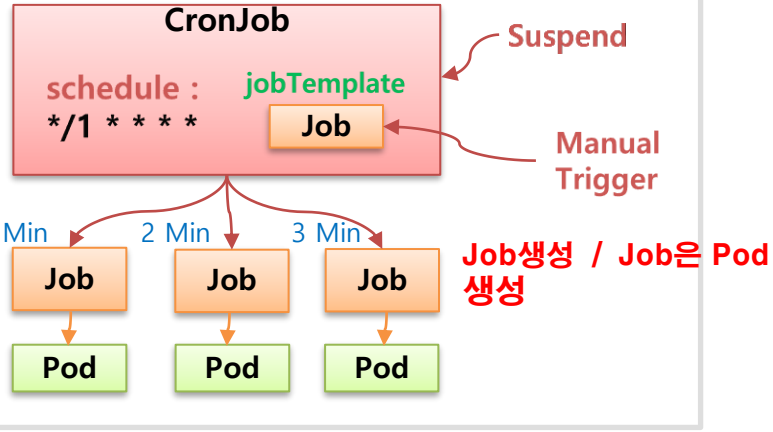
selector는 Job이 알아서만듦

30초 후 기능 종료

```
apiVersion: batch/v1
kind: Job
metadata:
 name: job-1
spec:
  completions: 6
  parallelism: 2
  activeDeadlineSeconds: 30
  template:
   spec:
     restartPolicy: Never
     containers:
     - name: container
       image: tmkube/init
```

Never, Onfailure

CronJob



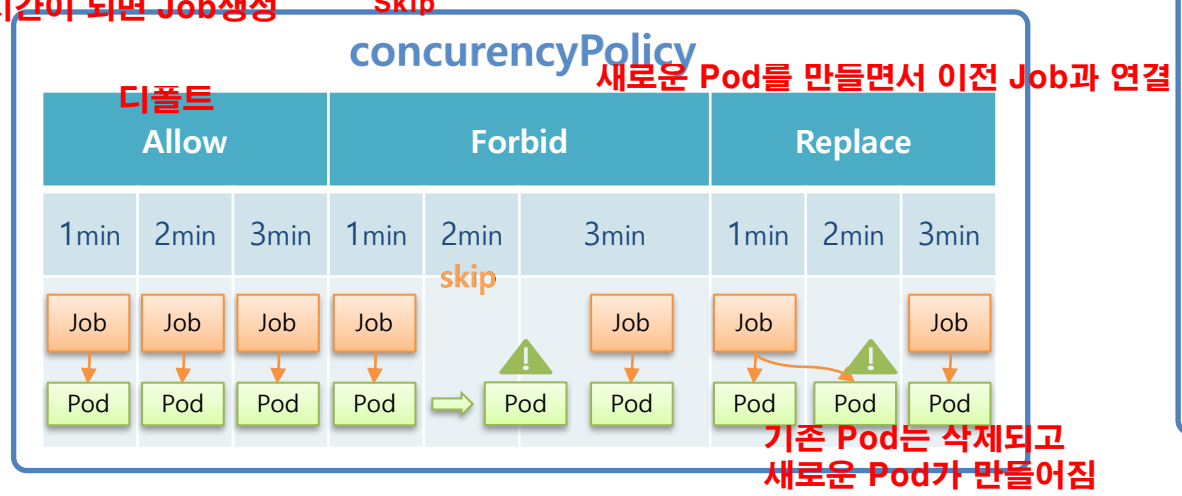
Job생성 / Job은 Pod 생성

```
apiVersion: batch/v1
kind: CronJob
metadata:
  name: cron-job
spec:
  schedule: "*/1 * * * *"
  concurrencyPolicy: Allow
  jobTemplate:
    spec:
      template:
        spec:
          restartPolicy: Never
          containers:
          - name: container
            image: tmkube/app
```

6. Controller - DaemonSet, Job, CronJob

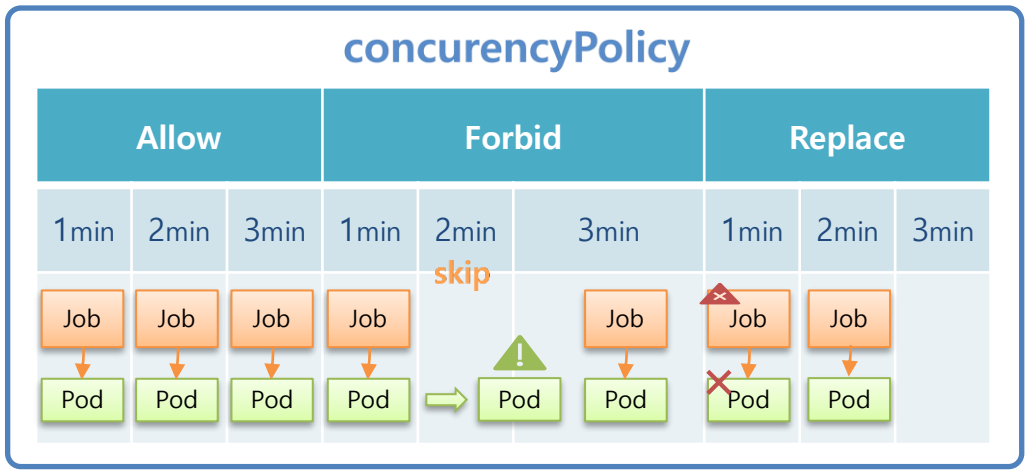
이전의 Job과 상관없이
시간이 되면 Job생성

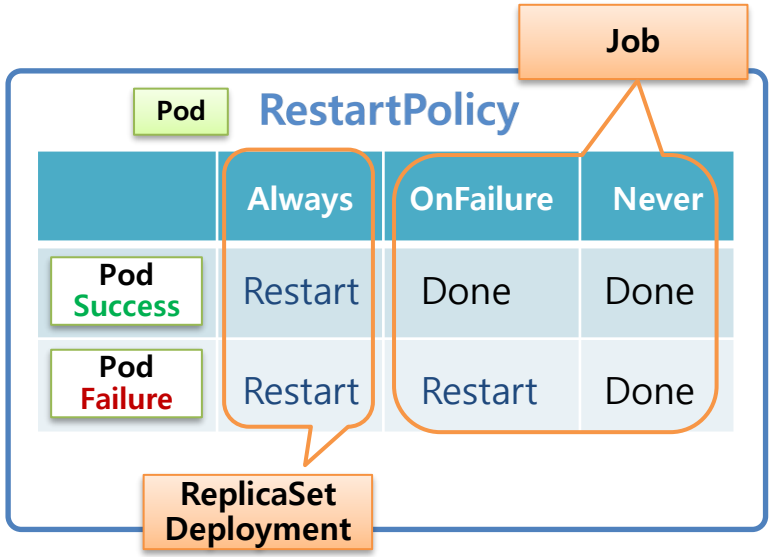
전의 Pod가 실행되고 있다면
Skip



새로운 Pod를 만들면서 이전 Job과 연결

기존 Pod는 삭제되고
새로운 Pod가 만들어짐






```yaml
apiVersion: batch/v1
kind: Job
metadata:
 name: job-1
spec:
  completions: 6
  parallelism: 2
  activeDeadlineSeconds: 30
  template:
   spec:
     restartPolicy: Never
     containers:
     - name: container
       image: tmkube/init
```