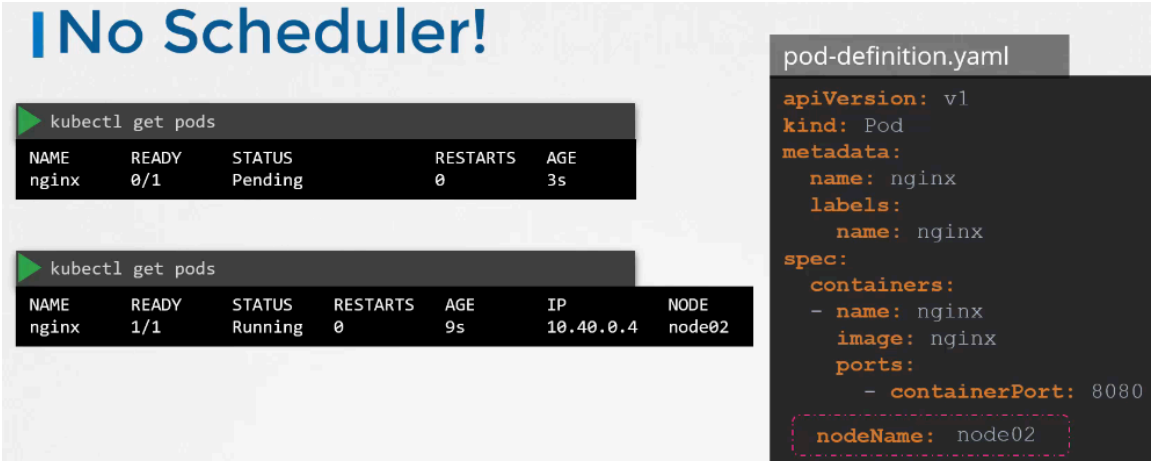
Section 3: Scheduling

50-54 Manual Scheduling

- 클러스터에 Scheduler가 없을 때는 bulit-in Scheduler에 의존하는 대신, 직접 노드에 pod를 할당하여 스케쥴링 할 수 있다.

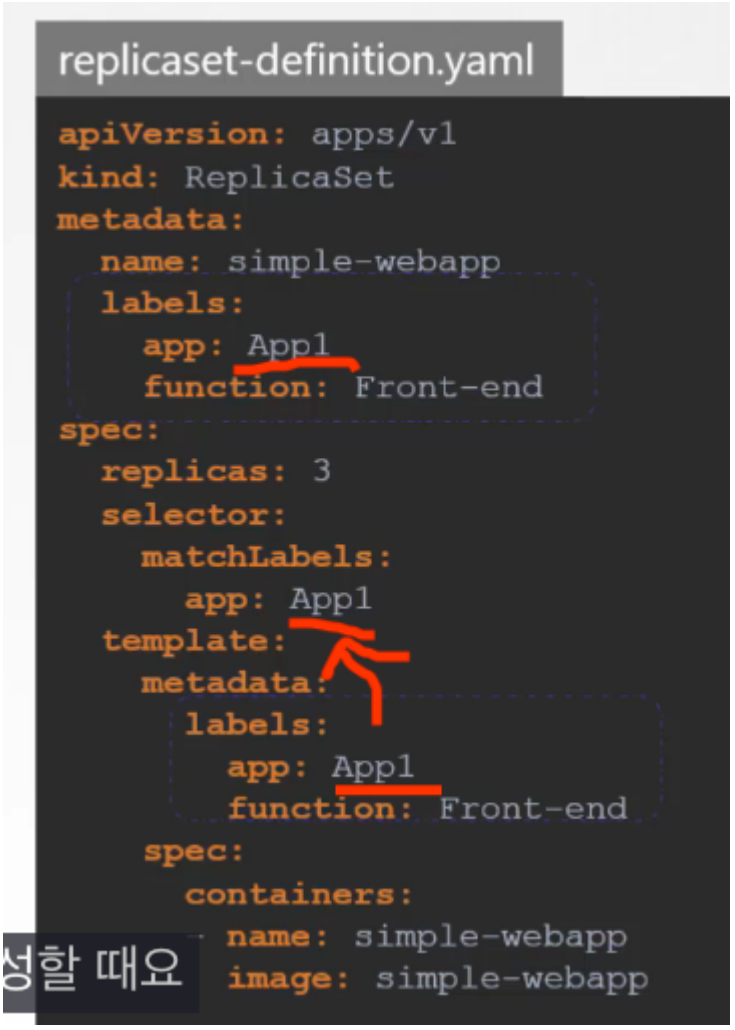


- pod의 status가 `Pending` 이라면, Scheduler가 없어 노드에 있는 pod의 스케줄링을 제대로 못했을 가능성 있음
  - `kubectl get pods -n kube-system` 으로 scheduler pod가 있는지 확인할 수 있다.
- 이미 생성된 포드의 nodeName을 변경하고 싶다면?
  - 직접 yaml 파일의 nodeName을 수정하는 방법
    - pod를 delete 다시 create 하거나, replace 를 사용하여 기존 파드를 대체
  - Binding 리소스를 yaml파일을 통해 만들고, 직접 POST api 요청을 하는 방법

55-57. Labels and Selectors

- selector를 이용해 pod 목록 조회하는 명령어: `kubectl get pods --selector env=dev`

- ReplicaSet 정의 파일에서 두 곳에서 정의된 라벨이 있어 초보자들은 주의할 필요가 있다.
  - ReplicaSet 자체의 label과 pod의 라벨이 있음. 파드를 레플리카셋에 연결하기 위해, 파드의 label과 ReplicaSet 자체의 label을 일치시켜야 한다.



- Labels and Selectors vs Annotations
  - Labels와 Selectors : 그룹과 객체를 선택하는 데 사용
  - Annotations: 정보 수집 목적으로 다른 세부 사항을 기록하는 데 사용
    - ex) 이름, 버전, 빌드 정보같은 툴 세부 정보
    - ex) 전화번호, e-mail 등은 통합 목적으로 사용 가능

58-60. Taints and Tolerations

- taints와 tolerations는 함께 작동하여 파드가 부적절한 노드에 스케줄되지 않게 한다.
- taint를 설정한 노드에는 파드들을 스케쥴링하지 않는다. 단, tolerations를 설정한 파드에는 스케쥴링할 수 있다.

- taint는 노드에, tolerations는 파드에 설정
- taint 설정 명령어: `kubectl taint nodes node1 dedicated=special-user:NoSchedule`
  - `node1` 이라는 노드에 key 가 `dedicated` , value가 `special-user` , effect가 `NoSchedule` 인 taint를 설정한다.
- taint 제거 명령어: 설정 명령어에 `-` 를 붙인다.
- 파드에 대한 tolerations 지정은 pod yaml 파일의 spec 하위에서 작성해준다.

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  labels:
    env: test
spec:
  containers:
  - name: nginx
    image: nginx
  tolerations:
    - key: "dedicated"
      operator: "Equal"
      value: "special-user"
      effect: "NoSchedule"
```

61-64. Node Selectors, Node Affinity

Node Selectors

- pod를 특정 label의 노드로 제한하는 가장 간단한 방법
- Node에 label 붙이는 명령어 : `kubectl label nodes <node-name> <label-key>=<label-value>`
  - ex) `kubectl label nodes node-1 size=Large`
- pod가 size=Large 라벨이 붙은 노드에만 할당되도록 제한 : 아래 yaml 참고

```
apiVersion:
kind: Pod
```

```
metadata:
    name: myapp-pod
spec:
    containers:
    - name: data-processor
        image: data-processor
    nodeSelector:
        size: Large
```

Node Affinity

- nodeSelector는 advanced expressions(고급 표현식)을 표현할 수 없다. Node Affinity를 이용해 고급 표현식을 표현할 수 있다.
    - ex) nodeSelector로 a 라벨인 것으로 제한은 가능하지만, a 라벨이 아닌 것, a 혹은 b 라벨인 것 등 복잡한 제한은 불가능하다.
- pod가 size=Large 이거나 size=Medium 인 노드에만 할당되도록 제한 : 아래 yaml 참고

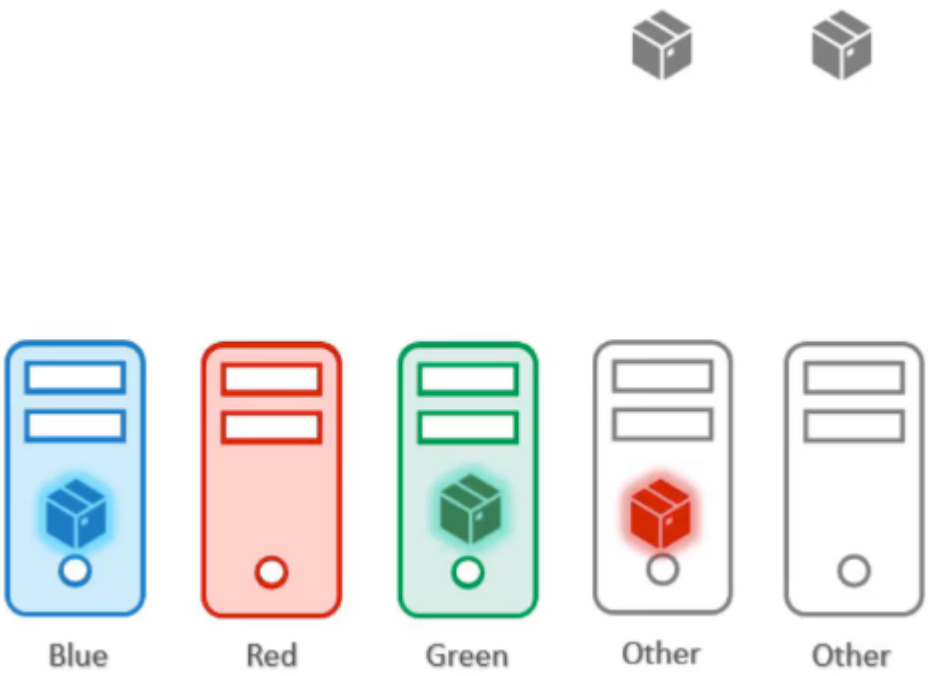
```
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: size
            operator: In   # NotIn, Exists, DoesNotExist, Gt, Lt ...
            values:
            - Large
            - Medium
```

    - Node Affinity Type : Node Affinity에 대한 스케줄러의 동작과 pod의 생명주기 단계를 정의한다. :

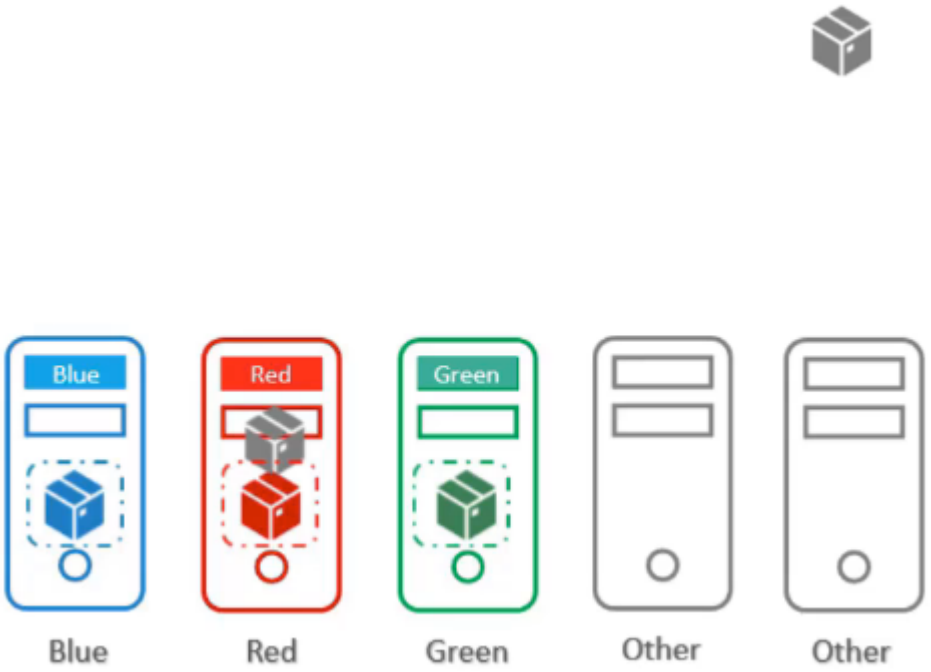
- 아래 두 type은 파드가 특정 노드에서 실행 중이라면, 노드의 조건이 바뀌어도 이미 실행 중인 파드는 그대로 실행되도록 한다.
  - `requiredDuringSchedulingIgnoredDuringExecution` : 스케쥴링 시 꼭 만족해야 하는 조건 (해당 조건에 만족하는 노드가 존재하지 않으면 파드를 스케쥴링하지 않음)
  - `preferredDuringSchedulingIgnoredDuringExecution` : 스케쥴링 시 만족하면 좋은 조건 (해당 조건에 만족하는 노드가 존재하지 않아도 가능한 노드에 스케쥴링함)
- `requiredDuringSchedulingRequiredDuringExecution` : 해당 조건에 만족하지 않는 노드에서 실행 중인 파드를 종료시킴

65. Taints and Tolerations vs Node Affinity

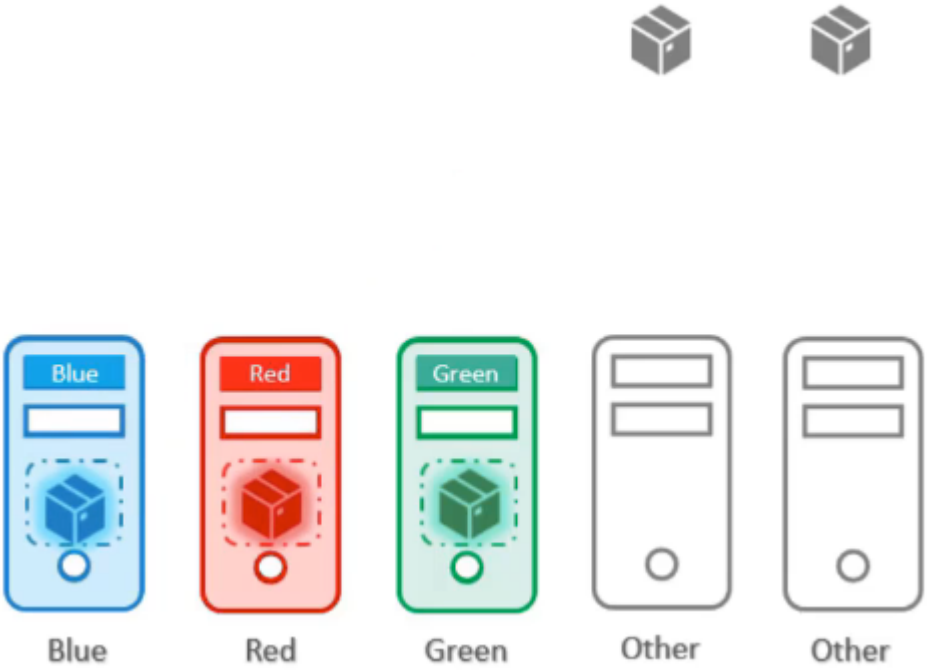
- taints 와 tolerations 만으로는 파드를 특정 노드로"만" 배치하도록 보장할 수 없다. (아래 상황이 가능)



- Node Affinity 만으로는 label이 붙은 노드에 해당 label의 pod만 배치되도록 보장할 수 없다. (아래처럼 다른 파드가 해당 label이 붙은 노드로 배치될 수도 있다)



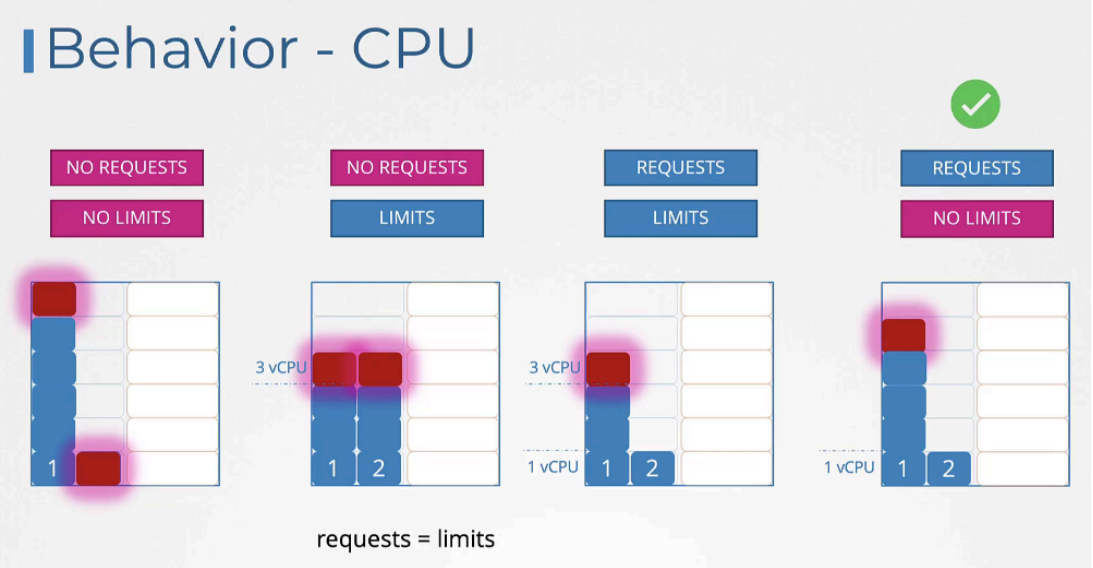
- 특정 노드에 특정 노드만이 배치되도록 보장하려면, Taints/Tolerations와 Node Affinity 둘 다 사용하여야 한다. Taints/Tolerations로 특정 노드에 특정 파드를 제외한 파드들이 배치되는 것을 막고, Node Affinity를 통해 특정 노드에 특정 노드만 배치되도록 한다.



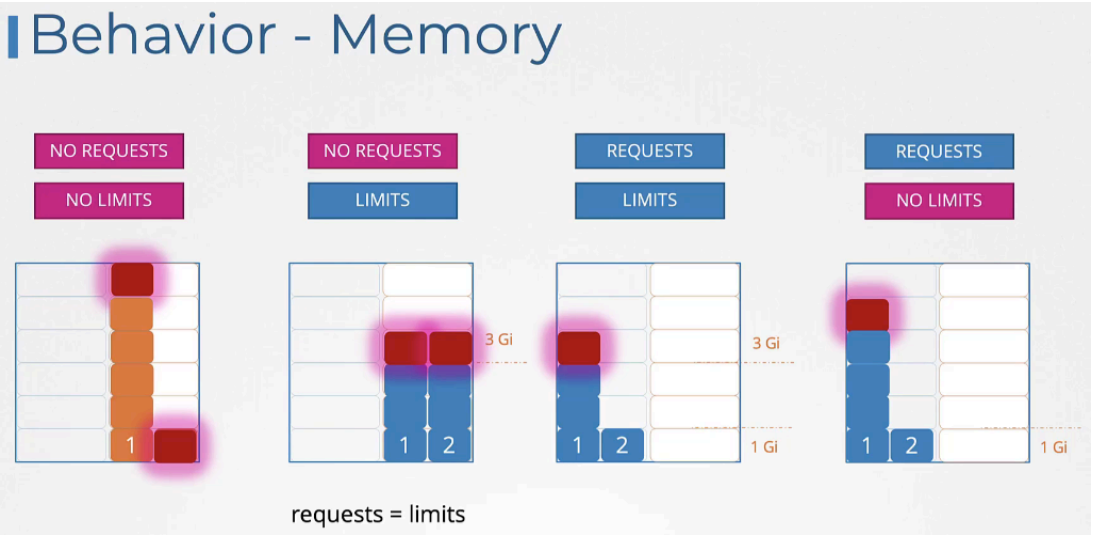
66-69. Resource Requirements and limits

```
kind: Pod
...
spec:
   contianers:
   - name: simple-webapp-color
      image: simple-webapp-color
      resources:
         requests:
            memory: "4Gi
            cpu: 2
         limits:  # 필요 시 limit 지정 가능
            memory: "2Gi"
            cpu: 3
```

- 1 CPU = 100m = 1 AWS vCPU = 1 GCP Core ...
- MEM = Memory
- 중요: 쿠버네티스는 기본적으로 CPU나 Memory request 혹은 limit set이 없다. 즉, 따라서 모든 포드는 노드에서 필요한 만큼의 자원을 얼마든지 소비할 수 있으며, 노드의 resource에서 실행되는 다른 포드나 프로세스를 죽일 수 있다.
- CPU
    - 가장 이상적인 것은 모든 노드에 대해 request 를 설정하고, limit을 두지 않는 것. resource를 불필요하게 제한하면 충분한 resource가 있을 때에도 제한적인 리소스를 사용하게 되기 때문에 이상적이지 않음. 하지만 모든 노드에 request를 설정해야 한다는 것을 기억해라



- Memory
  - CPU와는 달리 메모리는 조절할 수 없다. 따라서 pod 2가 pod1을 확보하기 위해 더 많은 메모리를 요청하면, 유일한 방법은 파드1을 죽이는 것



- LimitRange
  - LimitRange는 포드가 생성될때 기준으로 적용된다. 기존 파드에는 영향을 주지 않음.
- Quotas : 쿠버네티스 클러스터에 배포된 앱이 사용할 수 있는 전체 리소스를 제한
  - ex) 모든 포드가 특정 수치의 CPU나 메모리를 소비해선 안된다고 해야 한다
  - namespace level에서 Quota를 생성 가능

67. A quick note on editing Pods and Deployments

- 다음 외에는 기존 POD의 사양을 편집할 수 없다는 점을 기억할 것
  - spec.containers[*].image
  - spec.initContainers[*].image
  - spec.activeDeadlineSeconds
  - spec.tolerations

예를 들어, 실행 중인 포드의 환경 변수, service account, resource limit을 편집할 수 없다. 하지만 꼭 편집해야한다면 두 가지 옵션이 있다:

1. `kubectl edit pod <pod name>` 명령을 실행한다. 이렇게 하면 편집기(vi editor)에서 pod 사양이 열린다. 그런 다음 필요한 속성을 편집한다. 저장하려고 하면 아래와 같이 오류가 난다. 이는 편집할 수 없는 pod 필드를 편집하려고 하기 때문이다.

```
master $ kubectl edit pod webapp
error: pods "webapp" is invalid
A copy of your changes has been stored to "/tmp/kubectl-edit-ccvrq.yaml"
error: Edit cancelled, no valid changes were saved.
master $
```

변경 사항이 포함된 파일 사본은 위와 같이 임시 위치에 저장된다.

그 다음 명령을 실행하여 기존 포드를 삭제할 수 있다: `kubectl delete pod webapp`

그 다음, 임시 파일을 사용하여 변경 사항이 포함된 새 포드를 만들면 된다 : `kubectl create -f /tmp/kubectl-edit-ccvrq.yaml`

2. 아래 명령어를 사용하여 파일에 YAML 형식으로 포드 정의를 추출한다.

   `kubectl get pod webapp -o yaml > my-new-pod.yaml`

   그런 다음 편집기(vi 편집기)를 사용하여 내보낸 파일을 변경하고 저장한다.: `vi my-new-Pod.yaml`

   그 다음 기존 포드를 삭제합니다: `kubectl delete pod webapp`

   그런 다음 편집된 파일로 새 포드 만들기 : `kubectl create -f my-new-pod.yaml`

deployment를 사용하면 Pod template의 모든 필드/프로퍼티를 쉽게 편집할 수 있습니다. pod template은 deployment spec의 하위 항목이므로 변경할 때마다 deployment는 자동으로 삭제되고 새로운 변경 사항이 포함된 새로운 pod를 생성합니다. 따라서 deployment의 pod 부분 속성을 편집하라는 요청을 받으면 명령을 실행하여 간단히 편집할 수 있습니다: `kubectl edit deployment my-deployment`

70-75. DaemonSets, Static Pods

DaemonSets

클러스터 전체 노드에 특정 파드를 실행할 때 사용한다.

- 데몬셋의 좋은 예: kube-proxy, networking solution 등
- yaml 파일은 ReplicaSet과 구조가 비슷. kind 빼고 똑같다

daemon-set-definition.yaml

```yaml
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: monitoring-daemon
spec:
  selector:
    matchLabels:
      app: monitoring-agent
  template:
    metadata:
      labels:
        app: monitoring-agent
    spec:
      containers:
      - name: monitoring-agent
        image: monitoring-agent
```

replicaset-definition.yaml

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: monitoring-daemon
spec:
  selector:
    matchLabels:
      app: monitoring-agent
  template:
    metadata:
      labels:
        app: monitoring-agent
    spec:
      containers:
      - name: monitoring-agent
        image: monitoring-agent
```

Static Pods

API server나 다른 쿠버네티스 클러스터 구성 요소 없이 특정 노드에 있는 kubelet 데몬에 의해 직접 관리되는 파드

ex) etcd, api-server

- `/etc/kubernetes/manifests/` (=staticPodPath) 에 pod.yaml 파일 생성하면, kubelet이 주기적으로 이 디렉토리를 확인하고 pod를 만든다.

Static PODs	DaemonSets
Created by the Kubelet	Created by Kube-API server (DaemonSet Controller)
Deploy Control Plane components as Static Pods	Deploy Monitoring Agents, Logging Agents on nodes
Ignored by the Kube-Scheduler	

76-80. Multiple Schedulers, Configuring Scheduler Profiles

Multiple Schedulers

- default-scheduler: kube-scheduler에서 자동으로 만들어진 스케줄러

```
apiVersion: kubescheduler.config.k8s.io/v1
kind: KubeSchedulerConfiguration
profiles:
- schedulerName: default-scheduler #이름을 설정하지 않으면 기본 스케줄러
```

- custom scheduler (사용자 지정 스케줄러)

```
apiVersion: kubescheduler.config.k8s.io/v1
kind: KubeSchedulerConfiguration
profiles:
- schedulerName: my-scheduler
```

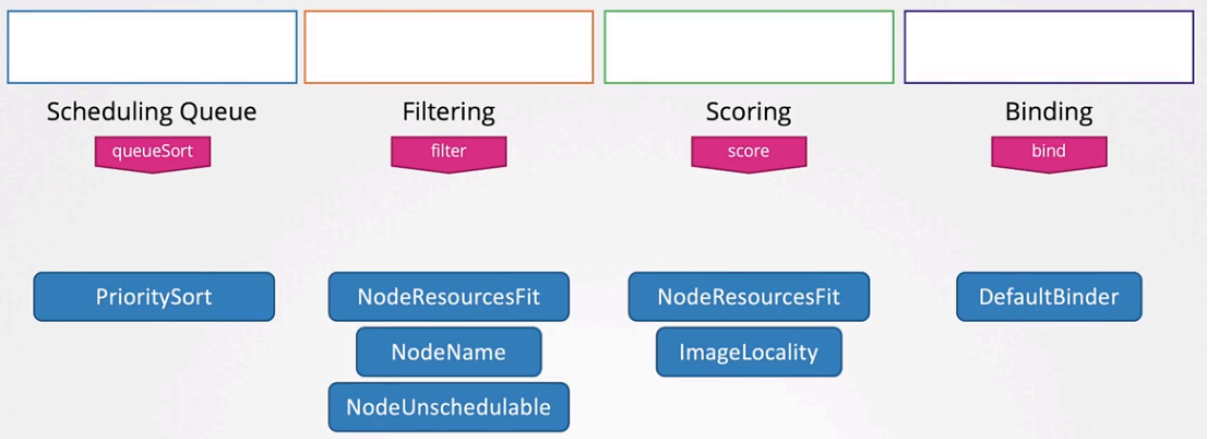
- custom scheduler를 이용하여 파드 배포 : `schedulerName: my-scheduler`

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    app: custom-scheduler
  name: custom-scheduler-pod
spec:
  containers:
  - image: nginx
```

```
  name: label-pod-1
schedulerName: my-scheduler
```

- **모든 이벤트 & 스케줄러 적용 확인:** `kubectl get events -o wide`

Scheduler profile



- Scheduling Queue: PriorityClass(plugin)를 사용하여 pod의 scheduling 우선순위 지정
- Filtering: 조건에 따라 pod를 scheduling할 후보 node들을 선정
- Scoring: sheduilng 가능한 node가 2개 이상일 때 최종 node를 선택하기 위해서 점수 매김
- Binding: 최종 Node를 선택하여 pod를 스케쥴링

81-86. Admission Controllers & Validating and Mutating Admission Controllers

Admission Controllers

- RBAC = Role Based Access Control
  - 특정 역할이 있는 사용자에게 pod, deployment, service와 같은 다양한 종류의 object를 생성 / 삭제할 수 있도록 허용 / 거부하는 등 다양한 종류의 제한을 설정 가능

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
```

```
metadata:
  name: pod-reader
rules:
- apiGroups: [""] # "" indicates the core API group
  resources: ["pods"]
  verbs: ["get", "watch", "list"]
```

- Admission Controllers
  - 구성의 유효성 검사 → 보안 강화
  - 파드가 생성되기 전에 request 자체를 변경하거나, 추가 작업을 수행할 것을 제안하는 등 많은 작업 가능
  - 기본적으로 enabled된 default plugin 확인: `kubectl exec -it kube-apiserver-controlplane -n kube-system -- kube-apiserver -h | grep 'enable-admission-plugins'`
  - enabled된 Admission controller plugin 확인:
    - `grep enable-admission-plugins /etc/kubernetes/manifests/kube-apiserver.yaml`
    - `ps -ef | grep kube-apiserver | grep admission-plugins`
  - **enable/disable Admission controller plugin**

```
vi /etc/kubernetes/manifests/kube-apiserver.yaml

---

...
spec:
  containers:
  - command:
    - kube-apiserver
    - --advertise-address=10.6.9.3
    - --allow-privileged=true
    - --authorization-mode=Node,RBAC
    - --client-ca-file=/etc/kubernetes/pki/ca.crt
    - --disable-admission-plugins=DefaultStorageClass
    - --enable-admission-plugins=NodeRestriction,NamespaceAutoPro
    - --enable-bootstrap-token-auth=true
...
```

Validating & Mutating Admission Controllers

- Mutating Admission Controller: 요청을 변경할 수 있음
- Validating Admission Controller: 요청의 유효성을 검사하여 허용 또는 거부할 수 있음
- 위 두 가지 기능을 모두 수행할 수 있는 Admission Controller도 있음