Scheduling

NodeName

- the scheduler ignores the pod and the kubelet running on the named node tries to run the pod.
- Limitations:
    - If the named node does not exist, the pod will not be run, and in some cases may be automatically deleted.
    - If the named node does not have the resources to accommodate the pod, the pod will fail and its reason will indicate why, e.g. OutOfmemory or OutOfcpu.
    - Node names in cloud environments are not always predictable or stable.

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - name: nginx
    image: nginx
  nodeName: kube-01
```

Node Selector

- For the pod to be eligible to run on a node, the node must have each of the indicated key-value pairs as labels (it can have additional labels as well). The most common usage is one key-value pair.

```
  apiVersion: v1
kind: Pod
metadata:
  name: nginx
  labels:
    env: test
spec:
  containers:
  - name: nginx
```

```
    image: nginx
    imagePullPolicy: IfNotPresent
  nodeSelector:
    disktype: ssd
```

Node Restriction

- Node labels can be updated by kubelet
- Compromised node and kubelet process can update label values and affect scheduling..
- The **NodeRestriction admission plugin prevents kubelets from setting or modifying labels with a node-restriction.kubernetes.io/ prefix**. To make use of that label prefix for node isolation:
- Example:
  example.com.node-restriction.kubernetes.io/fips=true or example.com.node-restriction.kubernetes.io/pci-dss=true

Affinity

Provides below benefits over nodeSelector

1. the language is **more expressive (not just "AND of exact match")**
2. you can indicate that the rule is **"soft"/"preference" rather than a hard requirement**, so if the scheduler can't satisfy it, the pod will still be scheduled
3. you can **constrain against labels on other pods running on the node (or other topological domain), rather than against labels on the node itself**, which allows rules about which pods can and cannot be co-located

Types:

- **requiredDuringSchedulingIgnoredDuringExecution**
  - **Hard requrement**
  - **rules that *must* be met for a pod to be scheduled onto a node (just like nodeSelector but using a more expressive syntax),**
- **preferredDuringSchedulingIgnoredDuringExecution**
  - **Soft requirement**
  - **scheduler will try to enforce but will not guarantee.**

1. **The “IgnoredDuringExecution” part of the names means that, similar to how nodeSelector works, if labels on a node change at runtime such that the affinity rules on a pod are no longer met, the pod will still continue to run on the node**

- If you specify both nodeSelector and nodeAffinity, *both* must be satisfied for the pod to be scheduled onto a candidate node.
- If you specify multiple nodeSelectorTerms associated with nodeAffinity types, then the pod can be scheduled onto a node **if one of** the nodeSelectorTerms is satisfied.
- If you specify multiple matchExpressions associated with nodeSelectorTerms, then the pod can be scheduled onto a node **only if all** matchExpressions can be satisfied.

Anti-affinity

Taints and tolerations

Resource requests and Limits