## NodeName

apiVersion: v1

kind: Pod

metadata:

name: node-name-pod

spec:

#Using Node Name

nodeName: worker-node-1.example.com

containers:

- image: nginx

name: pod

ports:

- containerPort: 80

k get pod node-name-pod -o wide

## NodeSelector

**k label node worker-node-1.example.com disktype=ssd**

k get nodes -l disktype=ssd

kubectl get nodes --show-labels

apiVersion: v1

kind: Pod

metadata:

name: node-selector-pod

spec:

#Using Node labels

nodeSelector:

disktype: ssd

containers:

- image: nginx

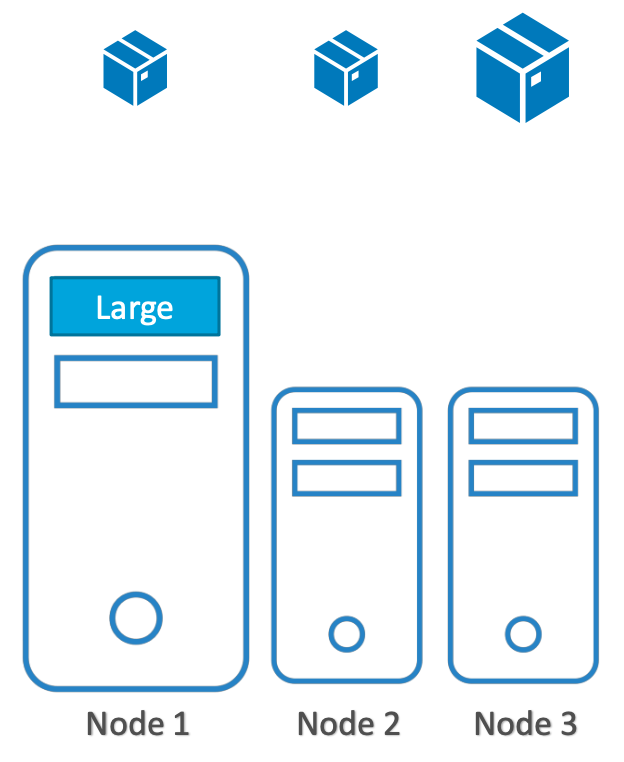
name: pod

ports:

- containerPort: 80

k get pod node-selector-pod -o wide

## Node Affinity



k label node worker-node-2.example.com disktype=ssd

k label node worker-node-1.example.com network=fast

k get nodes -l network=fast

k get nodes -l disktype=ssd

apiVersion: v1

kind: Pod

metadata:

creationTimestamp: null

name: affinity-pod

spec:

containers:

- image: nginx

name: pod

ports:

- containerPort: 80

affinity:

nodeAffinity:

requiredDuringSchedulingIgnoredDuringExecution:

nodeSelectorTerms:

- matchExpressions:

- key: network

operator: In

values:

- fast

preferredDuringSchedulingIgnoredDuringExecution:

- weight: 100

preference:

matchExpressions:

- key: disktype

operator: In

values:

- ssd

labsuser@master:~/scheduler$ k get nodes -l network=fast

NAME STATUS ROLES AGE VERSION

worker-node-1.example.com Ready <none> 13d v1.23.4

labsuser@master:~/scheduler$ k get nodes -l disktype=ssd

NAME STATUS ROLES AGE VERSION

worker-node-1.example.com Ready <none> 13d v1.23.4

worker-node-2.example.com Ready <none> 14d v1.23.4

labsuser@master:~/scheduler$ k apply -f affi

pod/affinity-pod created

labsuser@master:~/scheduler$ k get pods -o wide affinity-pod

NAME READY STATUS RESTARTS AGE IP NODE NOMINATED NODE READINESS GATES

affinity-pod 1/1 Running 0 10s 192.168.47.164 worker-node-1.example.com <none> <none>

#unset Lables

**k label node worker-node-1.example.com disktype-**

**k get nodes -l disktype=ssd**

**k label node worker-node-2.example.com disktype-**

**k get nodes -l disktype=ssd**

**k label node worker-node-1.example.com network-**

## 

## Pod Affinity

**Goal:**

| **node-1** | **node-2** | **node-3** |
| --- | --- | --- |
| *webserver-1* | *webserver-2* | *webserver-3* |
| *redis-cache-1* | *redis-cache-2* | *redis-cache-3* |

**redis-cache**

apiVersion: apps/v1

kind: Deployment

metadata:

name: redis-cache

spec:

selector:

matchLabels:

app: store

replicas: 3

template:

metadata:

labels:

app: store

spec:

affinity:

podAntiAffinity:

requiredDuringSchedulingIgnoredDuringExecution:

- labelSelector:

matchExpressions:

- key: app

operator: In

values:

- store

topologyKey: "kubernetes.io/hostname"

containers:

- name: redis-server

image: redis:3.2-alpine

**web-server**

apiVersion: apps/v1

kind: Deployment

metadata:

name: web-server

spec:

selector:

matchLabels:

app: web-store

replicas: 3

template:

metadata:

labels:

app: web-store

spec:

affinity:

podAntiAffinity:

requiredDuringSchedulingIgnoredDuringExecution:

- labelSelector:

matchExpressions:

- key: app

operator: In

values:

- web-store

topologyKey: "kubernetes.io/hostname"

podAffinity:

requiredDuringSchedulingIgnoredDuringExecution:

- labelSelector:

matchExpressions:

- key: app

operator: In

values:

- store

topologyKey: "kubernetes.io/hostname"

containers:

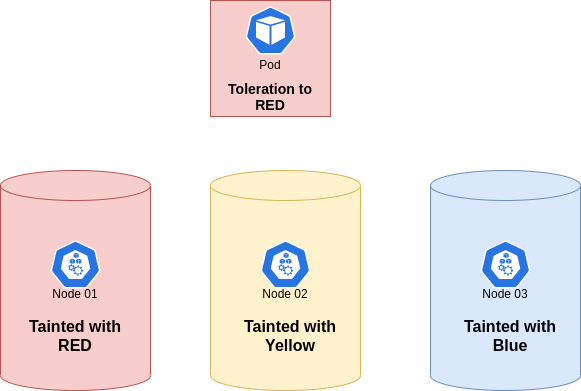
- name: web-app

image: nginx:1.16-alpine

k get pods -l app=store -o wide

k get pods -l app=web-store -o wide

## Taints and Tolerations



Example: Project reserved nodes, Windows containers, Special hardware - GPU

**Taint a node:**

kubectl taint node worker-node-2 type=gpu:NoSchedule

kubectl taint node worker-node-1 type=cpu:NoSchedule

Tolerate the taint in a Pod

apiVersion: v1

kind: Pod

metadata:

name: test-taint-pod

spec:

containers:

- name: nginxcontainer

image: nginx

tolerations:

- key: type

operator: Equal

value: gpu

effect: NoSchedule

**Untaint a node:**

kubectl taint node worker-node-2 type=gpu:NoSchedule-

kubectl taint node worker-node-1 type=cpu:NoSchedule-

? **Node Affinity vs Pod Affinity**

* Node Affinity ensures that pods are hosted on particular nodes.
* Pod Affinity ensures two pods to be co-located in a single node.

? **Node Affinity vs Taint & Tolerations**

*Node affinity* is a property of [Pods](https://kubernetes.io/docs/concepts/workloads/pods/) that *attracts* them to a set of [nodes](https://kubernetes.io/docs/concepts/architecture/nodes/) (either as a preference or a hard requirement).

*Taints* are the opposite -- they allow a node to repel a set of pods.

*Tolerations* are applied to pods, and allow (but do not require) the pods to schedule onto nodes with matching taints.

Taints and tolerations work together to ensure that pods are not scheduled onto inappropriate nodes. One or more taints are applied to a node; this marks that the node should not accept any pods that do not tolerate the taints.

## Security Context

apiVersion: v1

kind: Pod

metadata:

name: security-context-demo

spec:

securityContext:

runAsUser: 1000

runAsGroup: 3000

fsGroup: 2000

volumes:

- name: sec-ctx-vol

emptyDir: {}

containers:

- name: sec-ctx-demo

image: busybox:1.28

command: [ "sh", "-c", "sleep 1h" ]

volumeMounts:

- name: sec-ctx-vol

mountPath: /data/demo

securityContext:

allowPrivilegeEscalation: false

kubectl apply -f security-context.yaml

kubectl exec -it security-context-demo -- sh

> ps

PID USER TIME COMMAND

1 1000 0:00 sleep 1h

6 1000 0:00 sh

...

> cd /data

> ls -l

The output shows /data/demo directory has group ID 2000, which is the value of fsGroup.

drwxrwsrwx 2 root 2000 4096 Jun 6 20:08 demo

> cd demo

> echo hello > testfile

> ls -l

The output shows that testfile has group ID 2000, which is the value of fsGroup.

-rw-r--r-- 1 1000 2000 6 Jun 6 20:08 testfile

> id

The output is similar to this:

uid=1000 gid=3000 groups=2000

## Pod Priority

When Pod priority is enabled, the scheduler orders pending Pods by their priority and a Priority Pod is placed ahead of other pending Pods with lower priority in the scheduling queue. As a result, the higher priority Pod may be scheduled sooner than Pods with lower priority

**PriorityClass.yaml**

apiVersion: scheduling.k8s.io/v1

kind: PriorityClass

metadata:

name: high-priority

value: 1000000

globalDefault: false

description: "This priority class should be used for XYZ service pods only."

**Pod-priority.yaml**

apiVersion: v1

kind: Pod

metadata:

name: nginx-pp

labels:

env: test

spec:

containers:

- name: nginx

image: nginx

imagePullPolicy: IfNotPresent

priorityClassName: high-priority

k apply -f PriorityClass.yaml

k apply -f Pod-priority.yaml

k get pc