Affinity and anti-affinity

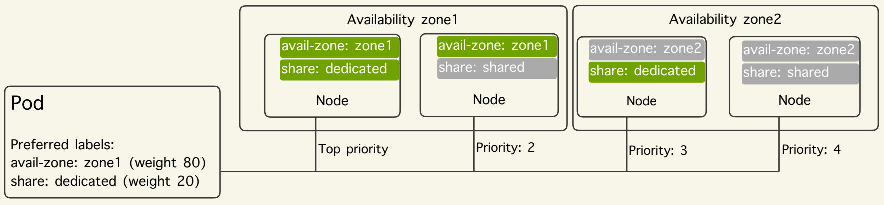
Affinity gives you more control over the scheduling process, allowing you to set rules based on the node's labels or pod's labels. Anti-affinity prevents Pods from being scheduled on the same node or group of nodes.

Affinity Type	Description
Node Affinity	Used to specify rules for which nodes a Pod can be scheduled on based on the labels of the nodes.
Pod Affinity	Used to specify rules for which Pods should be co-located on the same node based on the labels of other Pods running on the node.
Pod Anti-Affinity	Used to specify rules for which Pods should not be co-located on the same node based on the labels of other Pods running on the node.

Each type of Affinity can be further broken down into two categories

Affinity Caregory	Description
Required During Scheduling	Specifies that the rule must be satisfied for the Pod to be scheduled. If the rule is not satisfied, the Pod will not be scheduled.
Preferred During Scheduling	Specifies that the rule should be satisfied for the Pod to be scheduled, but is not required. If the rule is not satisfied, the Pod will still be scheduled.

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: database-pod
spec:
  containers:
  - name: database-pod
    image: postgres: 13.11
  affinity:
    nodeAffinity:
      preferredDuringSchedulingIgnoredDuringExecution:
      - weight: 80
        preference:
          matchExpressions:
            - key: avail-zone
              operator: In
              values:
              - zone1
      - weight: 20
        preference:
          matchExpressions:
            - key: share
              operator: In
              values:
              - dedicated
```



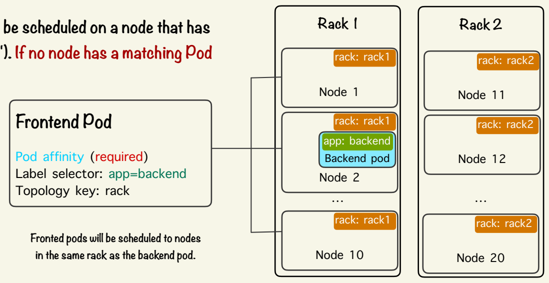
we used preferred Node Affinity to specify that the Pod prefers to be scheduled on nodes with the labels avail-zone: zone1 and share: dedicated. We also assigned a weight to each label to indicate the preference of the Pod. The higher the weight, the higher the priority of the label during scheduling

You can specify a weight between 1 and 100 for each instance of the preferredDuringSchedulingIgnoredDuringExecution affinity type

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: frontend-pod
spec:
  containers:
  - name: frontend-container
    image: frontend-image
  affinity:
    podAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
      - labelSelector:
          matchExpressions:
          - key: app
            operator: In
            values:
            - backend
        topologyKey: "rack"
```

we're using Pod Affinity to specify that the frontend-pod requires that it be scheduled on a node that has a Pod with the label app=backend in the same rack (topologyKey: "rack"). If no node has a matching Pod in the same rack, the frontend-pod will not be scheduled.

You can use the In, NotIn, Exists and DoesNotExist values in the operator field for affinity and anti-affinity.



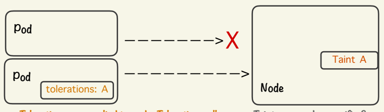
Fronted pods will be scheduled to nodes in the same rack as the backend pod.

Taints & Tolerations

Node affinity is a property of pods that can either prefer or require certain nodes for scheduling. In contrast, taints allow nodes to reject certain pods. Tolerations are applied to pods and enable the scheduler to schedule them on nodes that have the corresponding taints

Taints are defined using the kubectl taint command, and they consist of a key-value pair and an effect. The key-value pair is used to identify the type of taint

```
kubectl  taint nodes node-name key(=value):taint-effect
```



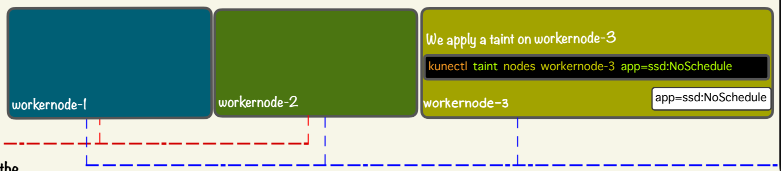
Tolerations are applied to pods. Tolerations allow the scheduler to schedule pods with matching taints

Taints are node-specific & applied to individual nodes

Taint-effect

- NoSchedule: This effect means that no new Pods will be scheduled on the Node unless they have a corresponding toleration. Existing Pods on the Node will continue to run
- NoExecute: This effect means that any Pods that do not have a corresponding toleration will be evicted from the Node. This can be useful for situations where a Node needs to be drained of its Pods for maintenance or other reasons
- PreferNoSchedule: This effect is similar to NoSchedule, but it allows Pods to be scheduled on the Node if there are no other Nodes available that match the Pod's scheduling requirements. However, if there are other Nodes available that do not have the taint, the Pod will be scheduled on one of those Nodes instead.

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx-pod
spec:
  containers:
    - name: nginx-container
      image: nginx:1.18
```



```
kunectl taint nodes workernode-3 app=ssd:NoSchedule
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: db-pod
spec:
  containers:
    - name: mysql-container
      image: mysql:latest
  tolerations:
    - key: "app"
      operator: "Equal"
      value: "ssd"
      effect: "NoSchedule"
```

nginx-pod does not tolerate the taint on the workernode-3 , so it will not be deployed on it

db-pod can still be scheduled on other nodes that do not have that taint

Pods that have this toleration can be scheduled on workernode-3

When you want to deploy a Pod on a specific node, you need to use taint affinity in addition to taints. This is because taints only restrict which nodes a Pod can be scheduled on based on the characteristics of the node, but do not take into account any preferences or constraints specific to the Pod itself

Notice: the node-role.kubernetes.io/master taint is automatically applied by the kubelet on the master node when the cluster is initialized. Its purpose is to reserve the master node for running control plane components and system Pods, ensuring they have dedicated resources and are not scheduled with regular user Pods. To enable scheduling Pods on the master node in Kubernetes, there are two approaches: adding a toleration or removing the applied taint

```
kubectl  describe  node  kubemaster | grep  Taint
Taint:        node-role.kubernetes.io/master:NoSchedule
```

Adding a Toleration: By adding a toleration to the Pod's configuration that matches the taint on the master node, the Pod can be scheduled on the master node despite the taint. This allows specific Pods to run on the master node while preserving its dedicated role for control plane components and system Pods.

```
tolerations:
    - key:  "node-role.kubernetes.io/master"
      operator:  "Exists "
```

Removing the Taint: Another way to allow Pods to be scheduled on the master node is by removing the taint altogether. This approach effectively opens up the master node for scheduling any type of Pod, including regular user Pods. However, removing the taint means that the master node may no longer be exclusively reserved for control plane components and system Pods, potentially affecting the stability and performance of the cluster.

```
kubectl  taint  nodes  <node-name>  node-role.kubernetes.io/master-
kubectl  taint  nodes   kubemaster   node-role.kubernetes.io/master-
```

Warning: the default master taint exists to protect the stability and reliability of the control plane. Removing it is not recommended as it can lead to overloading the master, reduced HA, and potentially cluster failures

Notice: when a node becomes not ready, indicating that it is no longer available to run new workloads, two taints are automatically added to the node: "node.kubernetes.io/not-ready:NoSchedule" and "node.kubernetes.io/not-ready:NoExecute". These taints serve different purposes and affect the scheduling and behavior of pods on the node.

```
kubectl  taint  nodes  <node-name>  node.kubernetes.io/not-ready:NoSchedule-
kubectl  taint  nodes  <node-name>  node.kubernetes.io/not-ready:NoExecute-
```

You can remove the taints using the "kubectl taint" command with the "--remove" option

Taint/Tolerations & Node Affinity

To achieve fine-grained control over pod scheduling and ensure pods are scheduled on specific nodes while those nodes only accept certain pods, you can use a combination of Node Affinity and Taints and Tolerations.

First, we use Node Affinity to specify the rules for selecting nodes based on their labels

==> Node Affinity: Node Affinity is used to specify rules that determine which nodes a pod can be scheduled on. You can define node affinity rules based on node labels, node fields, or node selectors. By applying node affinity to a pod, you can restrict its scheduling to specific nodes that meet the defined criteria.

```
kunectl  label  node  kubeworker-1  drive=ssd
kunectl  label  node  kubeworker-2  cpu=xeon
kunectl  label  node  kubeworker-3   gpu=yes
```

Second, we use Taints and Tolerations to indicate which pods can tolerate which taints on nodes. We can apply a taint to nodes that should only accept certain pods, and then specify the corresponding tolerations in the pod specification

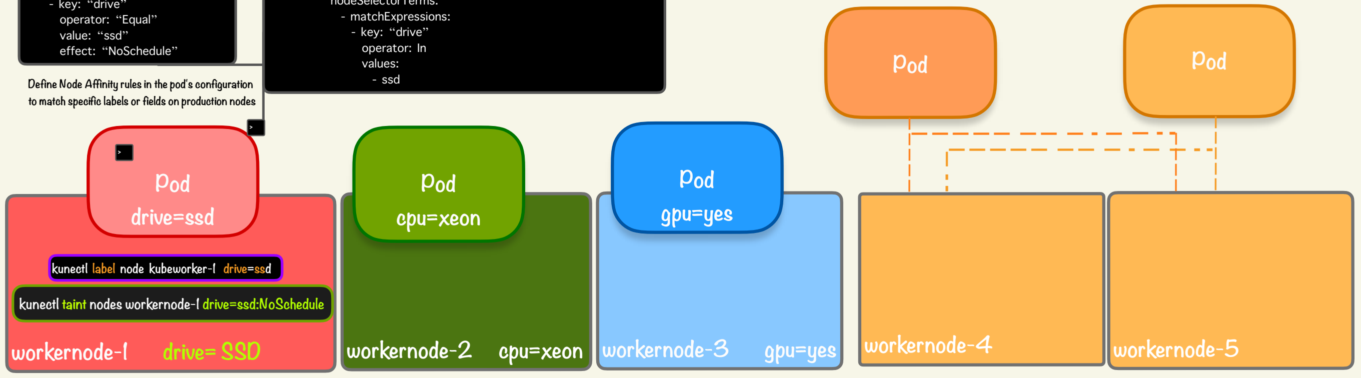
==> Taints and Tolerations: Taints are applied to nodes to repel or prevent pod scheduling by default. However, you can configure tolerations in the pod's configuration to allow specific pods to tolerate specific taints on nodes. Tolerations enable pods to be scheduled on tainted nodes by matching the taint's key and value.

```
kunectl  taint  nodes  workernode-1  drive=ssd:NoSchedule
kunectl  taint  nodes  workernode-2  cpu=xeon:NoSchedule
kunectl  taint  nodes  workernode-3  gpu=yes:NoSchedule
```

```
tolerations:
      - key:  "drive"
        operator:  "Equal"
        value:  "ssd"
        effect:  "NoSchedule"
```

```
affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
              - matchExpressions:
                - key:  "drive"
                  operator:  In
                  values:
                    - ssd
```

Define Node Affinity rules in the pod's configuration to match specific labels or fields on production nodes



```
kunectl label node kubeworker-1  drive=ssd
kunectl taint nodes workernode-1 drive=ssd:NoSchedule
```

With this approach, only pods that have the appropriate tolerations and satisfy the Node Affinity rules will be scheduled on the these nodes. Other nodes without the specific taint or lacking the required labels/fields won't receive these pods

Two Pods do not have any tolerations specified in their PodSpec, while the other two nodes do not have any taints applied. Therefore, the scheduler can schedule these two Pods on either of the taintless nodes.

```
apiVersion:  apps/v1
kind:  Deployment
metadata:
  name:  nfs-app1
  namespace:  dev
spec:
  replicas:  3
  selector:
    matchLabels:
      app:  nfs
  template:
    metadata:
      labels:
        app:  nfs
    spec:
      containers:
        - name:  nfs
          image:  nfs:latest
==>   affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
              - matchExpressions:
                - key:  "drive"
                  operator:  In
                  values:
                    - ssd
==>   tolerations:
      - key:  "drive"
        operator:  "Equal"
        value:  "ssd"
        effect:  "NoSchedule"
```

```
apiVersion:  apps/v1
kind:  Deployment
metadata:
  name:  nginx
  namespace:  dev
spec:
  replicas:  3
  selector:
    matchLabels:
      app:  nginx
  template:
    metadata:
      labels:
        app:  nginx
    spec:
      containers:
        - name:  nginx
          image:  nginx:1.17
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
              - matchExpressions:
                - key:  "drive"
                  operator:  In
                  values:
                    - ssd
      tolerations:
      - key:  "cpu"
        operator:  "Equal"
        value:  "xeon"
        effect:  "NoSchedule"
```

```
apiVersion:  apps/v1
kind:  Deployment
metadata:
  name:  image-processor-app1
  namespace:  dev
spec:
  replicas:  3
  selector:
    matchLabels:
      app:  image-processor
  template:
    metadata:
      labels:
        app:  image-processor
    spec:
      containers:
        - name:  image-processor
          image:  image-processor
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
              - matchExpressions:
                - key:  "drive"
                  operator:  In
                  values:
                    - ssd
      tolerations:
      - key:  "gpu"
        operator:  "Equal"
        value:  "yes"
        effect:  "NoSchedule"
```