

# K8s 亲和性

AUTHOR: 彭玲 TIME: 2022/5/13

## K8s 亲和性

亲和性

Node 亲和性

给节点添加标签

1. 依据 强制的节点亲和性 (required node affinity) 调度 Pod

yaml 清单

应用 yaml

验证

2. 使用 首选的节点亲和性 (preferred node affinity) 调度 Pod

Pod 亲和性

yaml 清单

Pod 亲和性测试

节点打标签

准备 3 个 Pods

应用 (apply)

测试 Pod 亲和性

项目上应用

## 亲和性

Node 和 Pod 均具有 亲和性 这一特性，注意，只有 Pod 具有 反亲和性。

## Node 亲和性

### 给节点添加标签

```
1 | anxin@node38:~$ kubectl label nodes node35 disktype=ssd
2 | node/node35 labeled
```

### 1. 依据 强制的节点亲和性 (required node affinity) 调度 Pod

#### yaml 清单

下面清单描述了一个 Pod，它有一个 节点亲和性 (nodeAffinity) 配置

`requiredDuringSchedulingIgnoredDuringExecution`，`disktype=ssd`。这意味着 pod 只会被调度到具有 `disktype=ssd` 标签的节点上。

```
1 | anxin@node38:~/pengling/k8s/affinity-demo$ vi pod-nginx-required-
  | affinity.yaml
2 |
```

```

3  apiVersion: v1
4  kind: Pod
5  metadata:
6    name: nginx
7  spec:
8    affinity:
9      nodeAffinity:
10     requiredDuringSchedulingIgnoredDuringExecution: # 强制的节点亲和性
11     nodeSelectorTerms:
12     - matchExpressions:
13     - key: disktype
14       operator: In
15       values:
16     - ssd
17   containers:
18   - name: nginx
19     image: nginx
20     imagePullPolicy: IfNotPresent

```

## 应用 yaml

```

1  anxin@node38:~/pengling/k8s/affinity-demo$ kubectl apply -f pod-nginx-
   required-affinity.yaml
2  pod/nginx created

```

## 验证

验证 pod 已经在所选节点（node35）上运行：

```

1  anxin@node38:~$ kubectl get pods -o=wide | { head -1; grep nginx; }
2  NAME      READY   STATUS    RESTARTS   AGE   IP           NODE      NOMINATED NODE
   READINESS GATES
3  nginx    1/1     Running   0           2m27s  10.244.2.158  node35   <none>
   <none>

```

## 2. 使用 首选的节点亲和性 (preferred node affinity) 调度 Pod

本清单描述了一个Pod，它有一个节点亲和性设置

`preferredDuringSchedulingIgnoredDuringExecution`，`disktype: ssd`。这意味着 pod 将首选具有 `disktype=ssd` 标签的节点。

```

1  # pods/pod-nginx-preferred-affinity.yaml
2  apiVersion: v1
3  kind: Pod
4  metadata:
5    name: nginx
6  spec:
7    affinity:
8      nodeAffinity:
9        preferredDuringSchedulingIgnoredDuringExecution: # 首选的节点亲和性
10     - weight: 1
11       preference:
12       matchExpressions:
13     - key: disktype
14       operator: In

```

```
15         values:
16         - ssd
17     containers:
18     - name: nginx
19       image: nginx
20       imagePullPolicy: IfNotPresent
```

## Pod 亲和性

要使用 Pod 间亲和性，可以使用 Pod 规约中的 `.affinity.podAffinity` 字段。对于 Pod 间反亲和性，可以使用 Pod 规约中的 `.affinity.podAntiAffinity` 字段。

## yaml 清单

下面的示例定义了一条 Pod 亲和性规则和一条 Pod 反亲和性规则：

```
1  anxin@node38:~/pengling/k8s/affinity-demo$ vi pod-with-pod-affinity.yaml
2
3  apiVersion: v1
4  kind: Pod
5  metadata:
6    name: with-pod-affinity
7  spec:
8    affinity:
9      podAffinity: # Pod 亲和性 (亲和 S1)
10       requiredDuringSchedulingIgnoredDuringExecution:
11         - labelSelector:
12             matchExpressions:
13             - key: security
14               operator: In
15               values:
16               - S1
17           topologyKey: disktype # topologyKey: topology.kubernetes.io/zone
18       podAntiAffinity: # Pod 反亲和性 (排斥 S2)
19         preferredDuringSchedulingIgnoredDuringExecution:
20         - weight: 100
21           podAffinityTerm:
22             labelSelector:
23               matchExpressions:
24               - key: security
25                 operator: In
26                 values:
27                 - S2
28             topologyKey: disktype # topologyKey: topology.kubernetes.io/zone
29     containers:
30     - name: with-pod-affinity
31       image: rancher/pause:3.6
```

# Pod 亲和性测试

## 节点打标签

前提条件 1: 给 node35 和 node37 打标签

```
1 # 给 node35 打标签: disktype=ssd
2 anxin@node38:~$ kubectl label nodes node35 disktype=ssd
3 node/node35 labeled
4 # 给 node37 打标签: disktype=hdd
5 anxin@node38:~$ kubectl label nodes node37 disktype=hdd
6 node/node37 labeled
```

## 准备 3 个 Pods

前提条件 2: 准备 3 个 Pods, 2 个 s1 标签的 Pods 分别运行在 node35 和 node37, 1 个 s2 标签的 Pod 运行在 node35

```
1 anxin@node38:~/pengling/k8s/affinity-demo$ vi pod-with-pod-affinity-pod-
  label.yaml
2
3 apiVersion: v1
4 kind: Pod
5 metadata:
6   name: with-pod-affinity-pod-label
7   labels:
8     security: S1 # S1 标签的 Pod
9 spec:
10  containers:
11  - name: with-pod-affinity-pod-label
12    image: rancher/pause:3.6
13  nodeSelector:
14    disktype: hdd # Pod 运行在 node37 节点
15  ---
16 apiVersion: v1
17 kind: Pod
18 metadata:
19   name: with-pod-affinity-pod-label-s1
20   labels:
21     security: S1 # S1 标签的 Pod
22 spec:
23  containers:
24  - name: with-pod-affinity-pod-label-s1
25    image: rancher/pause:3.6
26  nodeSelector:
27    disktype: ssd # Pod 运行在 node35 节点
28
29  ---
30 apiVersion: v1
31 kind: Pod
32 metadata:
33   name: with-pod-affinity-pod-label-s2
34   labels:
35     security: S2 # S2 标签的 Pod
36 spec:
37  containers:
```

```
38 - name: with-pod-affinity-pod-label-s2
39   image: rancher/pause:3.6
40   nodeSelector:
41     disktype: ssd # Pod 运行在 node35 节点
```

## 应用 (apply)

```
1 # 准备 3 个 Pods
2 anxin@node38:~/pengling/k8s/affinity-demo$ kubectl apply -f pod-with-pod-
  affinity-pod-label.yaml
3 pod/with-pod-affinity-pod-label created
4 pod/with-pod-affinity-pod-label-s1 created
5 pod/with-pod-affinity-pod-label-s2 created
6 # 创建目标 Pod
7 anxin@node38:~/pengling/k8s/affinity-demo$ kubectl apply -f pod-with-pod-
  affinity.yaml
8 pod/with-pod-affinity created
```

## 测试 Pod 亲和性

目标 Pod 亲 (亲和) s1, 排 (排斥) s2

- s1 标签的 Pods 有 2 个, 分别跑在 node35 和 node37 上
- s2 标签的 Pos 有 1 个, 跑在 node35 上

因此, 目标 Pod 应该跑在 node37 上。

```
1 # node37 上运行了 with-pod-affinity Pod, Pod 反亲和性起效
2 anxin@node38:~$ kubectl get po -o wide --show-labels | { head -1; grep with-
  pod-affinity; }
3 NAME                                READY  STATUS   RESTARTS  AGE    IP             NODE   ...
  LABELS
4 with-pod-affinity                   1/1   Running  0         13m   10.244.1.88   node37 ... <none>
5 with-pod-affinity-pod-label         1/1   Running  0         14m   10.244.1.87   node37 ... security=s1
6 with-pod-affinity-pod-label-s1      1/1   Running  0         14m   10.244.2.178  node35 ... security=s1
7 with-pod-affinity-pod-label-s2      1/1   Running  0         14m   10.244.2.177  node35 ... security=s2
```

## 项目上应用

下面使用了 Pod 的亲和性和反亲和性实现 kafka 集群的部署。

```
1 apiVersion: apps/v1beta1
2 kind: StatefulSet
3 metadata:
4   name: kafka
5 spec:
6   serviceName: kafka-hs
7   replicas: 3
8   podManagementPolicy: Parallel
9   updateStrategy:
```

```
10   type: RollingUpdate
11   template:
12     metadata:
13       labels:
14         app: kafka
15     spec:
16       affinity:
17         podAntiAffinity: # Pod 反亲和性
18           requiredDuringSchedulingIgnoredDuringExecution:
19             - labelSelector:
20                 matchExpressions:
21                   - key: "app"
22                     operator: In
23                     values:
24                       - kafka
25                 topologyKey: "kubernetes.io/hostname"
26         podAffinity: # Pod 亲和性
27           preferredDuringSchedulingIgnoredDuringExecution:
28             - weight: 1
29               podAffinityTerm:
30                 labelSelector:
31                   matchExpressions:
32                     - key: "app"
33                       operator: In
34                       values:
35                         - zk
36                 topologyKey: "kubernetes.io/hostname"
37       terminationGracePeriodSeconds: 300
38     containers:
39       - name: k8skafka
40         imagePullPolicy: Always
41         image: gcr.io/google_containers/kubernetes-kafka:1.0-10.2.1
```