Kubernetes 完全教程

Kubernetes 的 Pod 放置放置策略以及应用案例

王渊命 @jolestar

Agenda

1. Kubernetes 的 Pod 放置策略
    - 如何放置 Pod 到 Node
    - Affinity 以及 anti-affinity
    - Taints 和 Tolerations 机制
2. Kubernetes 应用开发
    - Java Spring Cloud 案例
    - PHP Wordpress

如何放置 Pod 到 Node -- 从手动编排过度到 Kubernets

Pod 的 nodeName 和 nodeSelector 属性

```
kubectl label nodes <nodename> disktype=ssd
```

```
apiVersion: v1
kind: Pod
metadata:
name: db
spec:
containers:
- name: db
image: mysql
nodeSelector:
  disktype: ssd
```

内置的 node labels

```
kubectl get nodes --show-labels
```

```
#cloud-provider
kubernetes.io/hostname
failure-domain.beta.kubernetes.io/zone
failure-domain.beta.kubernetes.io/region
beta.kubernetes.io/instance-type
#kubelet
beta.kubernetes.io/os
beta.kubernetes.io/arch
#kubeadm
node-role.kubernetes.io/master
```

Affinity 以及 anti-affinity

为什么需要 Affinity 机制

1. nodeSelector 选择表达式不够强大
2. nodeSelector 不能区分强制要求和优先配置
3. 无法避免同一个应用的多个 pod 调度到同一个 node 上

Affinity

- nodeAffinity
- podAffinity
- podAntiAffinity

> 为什么没有 nodeAntiAffinity?

nodeAffinity

- preferredDuringSchedulingIgnoredDuringExecution: **PreferredSchedulingTerm** array
  - preference: nodeSelectorTerms
  - weight: 0-100
- requiredDuringSchedulingIgnoredDuringExecution: **NodeSelector**

NodeSelector

参看第四章API Spec 以及安全机制 -- Label 和 Selector

- nodeSelectorTerms
  - matchExpressions array
    - key
    - operator: In, NotIn, Exists, DoesNotExist. Gt, Lt
    - values: string array

```
matchExpressions:
  - {key: disktype, operator: In, values: [ssd]}
  - {key: environment, operator: NotIn, values: [dev]}
```

nodeAffinity

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: with-node-affinity
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: failure-domain.beta.kubernetes.io/zone
            operator: In
            values:
            - az1
            - az2
  containers:
  - name: with-node-affinity
    image: example
```

nodeAffinity

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: with-node-affinity
spec:
  affinity:
    nodeAffinity:
      preferredDuringSchedulingIgnoredDuringExecution:
      - weight: 1
        preference:
          matchExpressions:
          - key: disktype
            operator: In
            values:
            - ssd
  containers:
  - name: with-node-affinity
    image: example
```

podAffinity/podAntiAffinity

- preferredDuringSchedulingIgnoredDuringExecution: **WeightedPodAffinityTerm** array
  - podAffinityTerm： PodAffinityTerm
  - weight： 0-100
- requiredDuringSchedulingIgnoredDuringExecution: **PodAffinityTerm** array
  - labelSelector： LabelSelector
  - namespaces： string array
  - topologyKey

podAffinity

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: with-pod-affinity
spec:
  affinity:
    podAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
      - labelSelector:
          matchExpressions:
          - key: security
            operator: In
            values:
            - S1
        topologyKey: failure-domain.beta.kubernetes.io/zone
  containers:
  - name: with-pod-affinity
    image: example
```

podAntiAffinity

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: zk
  lables:
    app: zk
spec:
  affinity:
    podAntiAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        - labelSelector:
            matchExpressions:
            - key: "app"
              operator: In
              values:
              - zk
          topologyKey: "kubernetes.io/hostname"
  containers:
```

为什么要有 Taints 和 Tolerations

1. Taints 隔离 Node
    - 上线新 Node 验证
    - 下线 Node 维护
    - 专用 Node
2. Tolerations Pod 对 Taints Node 的容忍
    - 验证 Taints Node 是否工作正常
    - 表明 Pod 有资格使用专用 Node
    - 每个节点必须运行的基础 Pod

Taints 和 Tolerations

```
kubectl taint nodes node1 key=value:NoSchedule
```

- tolerations
  - operator: Exists,**Equal** (default)
  - effect: NoSchedule,PreferNoSchedule,NoExecute

```
tolerations:
- key: "key"
  operator: "Equal"
  value: "value"
  effect: "NoSchedule"
  tolerationSeconds: 3600
```

Tolerations

```
apiVersion: extensions/v1beta1
kind: DaemonSet
metadata:
  name: kube-proxy
spec:
  template:
    spec:
      containers:
      - name: kube-proxy
        image: gcr.io/google_containers/hyperkube-amd64:1.7.9
      tolerations:
      - key: "CriticalAddonsOnly"
        operator: "Exists"
      - key: "dedicated"
        operator: "Exists"
      - key: "node-role.kubernetes.io/master"
        effect: NoSchedule
      - key: node.cloudprovider.kubernetes.io/uninitialized
        effect: NoSchedule
        value: "true"
```

Taint based Evictions

```
kube-controller-manager --feature-gates=TaintBasedEvictions=true
```

内置的 taints

```
node.kubernetes.io/not-ready
node.alpha.kubernetes.io/unreachable
node.kubernetes.io/out-of-disk
node.kubernetes.io/memory-pressure
node.kubernetes.io/disk-pressure
node.kubernetes.io/network-unavailable
node.cloudprovider.kubernetes.io/uninitialized
```

DaemonSet Tolerations

```
node.alpha.kubernetes.io/unreachable
node.kubernetes.io/not-ready
```

Kubernetes 应用开发案例

1. Java SpringCloud
2. PHP Wordpress

Kubernetes SpringCloud 案例

1. spring-cloud-kubernetes
2. fabric8

spring-cloud-kubernetes

https://github.com/spring-cloud-incubator/spring-cloud-kubernetes

- DiscoveryClient
- ConfigMap/Secrets PropertySource
- Pod Health Indicator
- Ribbon/Zipkin discovery

fabric8

Integrated Development Platform for Kubernetes

https://fabric8.io/

fabric8-maven-plugin

- fabric8:run
- fabric8:build
- fabric8:push
- fabric8:deploy

SpringCloud 案例演示

准备 minikube 环境

```
#第一次启动要翻墙
minikube start
#没开启一个新的 terminal 都需要重新执行一下
eval $(minikube docker-env)
sudo route -n add 10.0.0.0/24 $(minikube ip)
```

spring-cloud-kubernetes-examples

- kubernetes-hello-world-example

```
mvn fabric8:run -Pkubernetes
kubectl get service
curl http://$clusterip
#修改代码 重新执行上面的命令看结果
```

- kubernetes-reload-example

```
kubectl apply -f config-map.yml
mvn fabric8:run -Pkubernetes
kubectl logs -f $spring-cloud-reload-podid
kubectl edit configmap reload-example
# 可看到修改后的 configmap 被重新加载
```

spring-cloud-kubernetes-examples

- kubernetes-circuitbreaker-ribbon-example

```
cd name-service; mvn fabric8:run -Pkubernetes
cd greeting-service; mvn fabric8:run -Pkubernetes
curl $greeting-service-clusterIP/greeting
kubectl scale --replicas=2 deployment name-service
curl $greeting-service-clusterIP/greeting
# 可以看到 ribbon 的效果
```

PHP Wordpress

Kubernetes 上部署 Wordpress

- LoadBalancer
- PersistentVolumeClaim
- Secret
- MySQL

总结

- 对 Kubernetes 的 Pod 放置策略有一个整体的了解
- 了解在 Kubernetes 上开发应用的流程以及应用如何利用 Kubernetes 的特性

作业

1. 思考自己所在公司的服务如果全部部署到 Kubernetes 中，需要如何规划 Node 的 Label，以及那些节点需要通过 taints 机制来专用，那些服务需要使用到 Affinity 机制。
2. 通过 Kubernetes 去部署一个更复杂的微服务应用（根据自己熟悉的语言来选择）
3. 以 Wordpress 为例，思考如果要做高可用，需要做哪些方面的改造（提示: 图片存储，数据库）。

关于我

个人博客: http://jolestar.com
课程 Github： https://github.com/jolestar/kubernetes-complete-course

课程 QQ 群: 451252952

关注我们

QingCloud-IaaS

青云QingCloud

www.qingcloud.com

QINGCLOUD青云