### DaemonSet

DaemonSet（守护进程集）和守护进程类似，它在符合匹配条件的节点上均部署一个Pod。

什么是DaemonSet

DaemonSet确保全部（或者某些）节点上运行一个Pod副本。当有新节点加入集群时，也会为它们新增一个Pod。当节点从集群中移除时，这些Pod也会被回收，删除DaemonSet将会删除它创建的所有Pod。

使用DaemonSet的一些典型用法：

* 运行集群存储daemon（守护进程），例如在每个节点上运行Glusterd、Ceph等。
* 在每个节点运行日志收集daemon，例如Fluentd、Logstash。
* 在每个节点运行监控daemon，比如Prometheus Node Exporter、Collectd、Datadog代理、New Relic代理或 Ganglia gmond。

编写DaemonSet规范

创建一个DaemonSet的内容大致如下，比如创建一个fluentd的DaemonSet：

apiVersion: apps/v1

kind: DaemonSet

metadata:

name: fluentd-es-v2.0.4

namespace: logging

labels:

k8s-app: fluentd-es

version: v2.0.4

kubernetes.io/cluster-service: "true"

addonmanager.kubernetes.io/mode: Reconcile

spec:

selector:

matchLabels:

k8s-app: fluentd-es

version: v2.0.4

template:

metadata:

labels:

k8s-app: fluentd-es

kubernetes.io/cluster-service: "true"

version: v2.0.4

# This annotation ensures that fluentd does not get evicted if the node

# supports critical pod annotation based priority scheme.

# Note that this does not guarantee admission on the nodes (#40573).

annotations:

scheduler.alpha.kubernetes.io/critical-pod: ''

seccomp.security.alpha.kubernetes.io/pod: 'docker/default'

spec:

serviceAccountName: fluentd-es

containers:

- name: fluentd-es

image: k8s.gcr.io/fluentd-elasticsearch:v2.0.4

env:

- name: FLUENTD\_ARGS

value: --no-supervisor -q

resources:

limits:

memory: 500Mi

requests:

cpu: 100m

memory: 200Mi

volumeMounts:

- name: varlog

mountPath: /var/log

- name: varlibdockercontainers

mountPath: /var/lib/docker/containers

readOnly: true

- name: config-volume

mountPath: /etc/fluent/config.d

nodeSelector:

beta.kubernetes.io/fluentd-ds-ready: "true"

terminationGracePeriodSeconds: 30

volumes:

- name: varlog

hostPath:

path: /var/log

- name: varlibdockercontainers

hostPath:

path: /var/lib/docker/containers

- name: config-volume

configMap:

name: fluentd-es-config-v0.1.4

1. 必需字段

和其他所有Kubernetes配置一样，DaemonSet需要apiVersion、kind和metadata字段，同时也需要一个.spec配置段。

2. Pod模板

.spec唯一需要的字段是.spec.template。.spec.template是一个Pod模板，它与Pod具有相同的配置方式，但它不具有apiVersion和kind字段。

除了Pod必需的字段外，在DaemonSet中的Pod模板必须指定合理的标签。

在DaemonSet中的Pod模板必须具有一个RestartPolicy，默认为Always。

3. Pod Selector

.spec.selector字段表示Pod Selector，它与其他资源的.spec.selector的作用相同。

.spec.selector表示一个对象，它由如下两个字段组成：

* matchLabels，与ReplicationController的.spec.selector的作用相同，用于匹配符合条件的Pod。
* matchExpressions，允许构建更加复杂的Selector，可以通过指定key、value列表以及与key和value列表相关的操作符。

如果上述两个字段都指定时，结果表示的是AND关系（逻辑与的关系）。

.spec.selector必须与.spec.template.metadata.labels相匹配。如果没有指定，默认是等价的，如果它们的配置不匹配，则会被API拒绝。

（4）指定节点部署Pod

如果指定了.spec.template.spec.nodeSelector，DaemonSet Controller将在与Node Selector（节点选择器）匹配的节点上创建Pod，比如部署在磁盘类型为ssd的节点上（需要提前给节点定义标签Label）：

containers:

- name: nginx

image: nginx

imagePullPolicy: IfNotPresent

nodeSelector:

disktype: ssd

提 示

Node Selector同样适用于其他Controller。

创建DaemonSet

在生产环境中，公司业务的应用程序一般无须使用DaemonSet部署，一般情况下只有像Fluentd（日志收集）、Ingress（集群服务入口）、Calico（集群网络组件）、Node-Exporter（监控数据采集）等才需要使用DaemonSet部署到每个节点。本节只演示DaemonSet的使用。

比如创建一个nginx ingress（文件地址：<https://github.com/dotbalo/k8s/blob/master/nginx-ingress/ingress.yaml>）：

# kubectl create -f nginx-ds.yaml

namespace/ingress-nginx created

configmap/nginx-configuration created

configmap/tcp-services created

configmap/udp-services created

serviceaccount/nginx-ingress-serviceaccount created

clusterrole.rbac.authorization.k8s.io/nginx-ingress-clusterrole created

role.rbac.authorization.k8s.io/nginx-ingress-role created

rolebinding.rbac.authorization.k8s.io/nginx-ingress-role-nisa-binding created

clusterrolebinding.rbac.authorization.k8s.io/nginx-ingress-clusterrole-nisa-binding created

daemonset.extensions/nginx-ingress-controller created

此时会在每个节点创建一个Pod：

# kubectl get po -n ingress-nginx

NAME READY STATUS RESTARTS AGE

nginx-ingress-controller-fjkf2 1/1 Running 0 44s

nginx-ingress-controller-gfmcv 1/1 Running 0 44s

nginx-ingress-controller-j89qc 1/1 Running 0 44s

nginx-ingress-controller-sqsk2 1/1 Running 0 44s

nginx-ingress-controller-tgdt6 1/1 Running 0 44s

[root@k8s-master01 2.2.8]# kubectl get po -n ingress-nginx -o wide

NAME READY STATUS RESTARTS AGE

IP NODE NOMINATED NODE

nginx-ingress-controller-fjkf2 1/1 Running 0 50s 192.168.20.30 k8s-node01 <none>

nginx-ingress-controller-gfmcv 1/1 Running 0 50s 192.168.20.21 k8s-master02 <none>

nginx-ingress-controller-j89qc 1/1 Running 0 50s 192.168.20.22 k8s-master03 <none>

nginx-ingress-controller-sqsk2 1/1 Running 0 50s 192.168.20.31 k8s-node02 <none>

nginx-ingress-controller-tgdt6 1/1 Running 0 50s 192.168.20.20 k8s-master01 <none>

更新和回滚DaemonSet

如果修改了节点标签（Label），DaemonSet将立刻向新匹配上的节点添加Pod，同时删除不能匹配的节点上的Pod。

在Kubernetes 1.6以后的版本中，可以在DaemonSet上执行滚动更新，未来的Kubernetes版本将支持节点的可控更新。

DaemonSet滚动更新可参考：<https://kubernetes.io/docs/tasks/manage-daemon/update-daemon-set/>。

DaemonSet更新策略和StatefulSet类似，也有OnDelete和RollingUpdate两种方式。

查看上一节创建的DaemonSet更新方式：

# kubectl get ds/nginx-ds -o go-template='{{.spec.updateStrategy.type}}{{"\n"}}'

RollingUpdate

提 示

如果是其他DaemonSet，请确保更新策略是RollingUpdate（滚动更新）。

1. 命令式更新

kubectl edit ds/<daemonset-name>

kubectl patch ds/<daemonset-name> -p=<strategic-merge-patch>

2. 更新镜像

kubectl set image ds/<daemonset-name><container-name>=<container-new-image>--record=true

3. 查看更新状态

kubectl rollout status ds/<daemonset-name>

4. 列出所有修订版本

kubectl rollout history daemonset <daemonset-name>

5. 回滚到指定revision

kubectl rollout undo daemonset <daemonset-name> --to-revision=<revision>

DaemonSet的更新和回滚与Deployment类似，此处不再演示。