Kubernetes Daemonset 控制器

前言：

课程名称：Kubernetes Daemonset 控制器

实验环境：

本章节 Kubernetes 集群环境如下：

角色	IP	主机名	组件	硬件
控制节点	192.168.128.11	k8s-master01	apiserver controller-manager scheduler etcd containerd	CPU：4vCPU 硬盘：100G 内存：4GB 开启虚拟化
工作节点	192.168.128.21	k8s-node01	kubelet kube-proxy containerd calico coredns	CPU：6vCPU 硬盘：100G 内存：8GB 开启虚拟化
工作节点	192.168.128.22	k8s-node02	kubelet kube-proxy containerd calico coredns	CPU：6vCPU 硬盘：100G 内存：8GB 开启虚拟化

张岩峰老师微信，加我微信，邀请你加入 VIP 交流答疑群：

微信号：ZhangYanFeng0429

二维码：

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

1、初识 DaemonSet 控制器

1.1、DaemonSet 概述

在 Kubernetes 中，DaemonSet 是一种控制器，用于在每个 Node 上运行一个 Pod 的副本。它确保在 Kubernetes 集群的每个节点上都存在一个 Pod 副本，并且这个 Pod 副本可以执行一些特定的任务，例如监控、日志收集、节点维护等。

与 Deployment 不同，DaemonSet 不指定副本数，而是在每个 Node 上运行一个 Pod 副本。当一个新的节点加入 Kubernetes 集群时，DaemonSet 会自动在这个节点上启动一个 Pod 副本，当一个节点从集群中删除时，该节点上的 Pod 副本也会被删除。

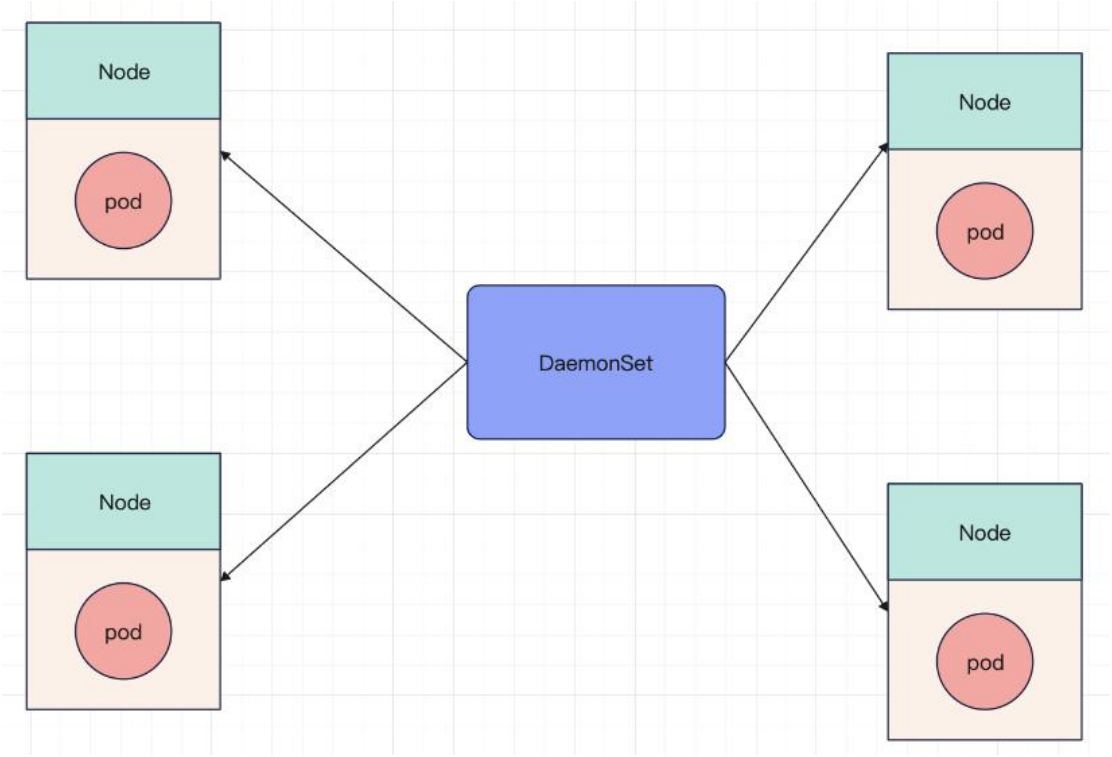
DaemonSet 控制器常用于部署一些系统级别的服务或应用程序，如 fluentd、kube-proxy 和 Node Exporter。

1.2、DaemonSet 工作原理：如何管理 Pod?

DaemonSet 的工作原理是监听节点的变化，通过监听节点的变化，当新的节点加入集群时，DaemonSet 会自动在该节点上创建一个 Pod 副本。而当节点从集群中删除时，DaemonSet 会自动删除该节点上的 Pod 副本。

这样，DaemonSet 保证了集群中每个节点都会运行指定的 Pod。有且只有一个 pod。

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。



1.3、Daemonset 典型的应用场景

DaemonSet 用于在集群中运行一组 Pod，确保每个节点都有一个 Pod 在运行。它通常用于运行一些系统级别的服务或者监控应用程序，例如：

1、日志收集器：DaemonSet 可以在每个节点上运行日志收集器，例如 Fluentd 或者 Filebeat，从而收集所有节点的日志数据，并将其发送到中心日志服务器进行存储和分析。

2、监控代理：DaemonSet 可以在每个节点上运行监控代理，例如 Prometheus Node Exporter，从而收集所有节点的运行状态数据，并将其发送到中心监控服务器进行分析和展示。

3、网络代理：DaemonSet 可以在每个节点上运行网络代理，例如 kube-proxy 或者 Istio Sidecar，从而负责节点之间的网络通信和流量管理。

4、安全代理：DaemonSet 可以在每个节点上运行安全代理，例如 Sysdig Falco 或者 Aqua Security，从而检测所有节点的安全事件，并及时报警或者进行防御。

总之，DaemonSet 适用于需要在每个节点上运行一组 Pod 的场景，可以使集群中的服务更加健壮和可靠。

1.4、DaemonSet 与 Deployment 的区别

DaemonSet 和 Deployment 都是 Kubernetes 中的控制器，但它们的设计目标和用途有所不同，主要区别如下：

1、Pod 管理方式：Deployment 的主要目标是在集群中按需部署、扩展和更新应用程序。它通常会创建多个 Pod 副本，并在整个集群中负载均衡这些 Pod，以便提供高可用性的服务。而 DaemonSet 的主要目标是在每个 Node 上运行一个 Pod 副本，以便在集群中部署基础服务和系统级别的任务，例如网络代理、监控和日志收集器等。它通常只需要一个 Pod 副本，且每个 Pod 只运行在一个 Node 上。

2、更新策略：Deployment 控制器通常会使用“滚动升级”等策略来更新和升级应用程序。即创建一组新的 Pod，逐步替换旧的 Pod，以便在不影响现有服务的情况下，实现平滑的升级。而 DaemonSet 通常会使用“替换升级”策略，即通过删除旧 Pod 和创建新 Pod 的方式来更新 DaemonSet，因为 DaemonSet 通常只有一个 Pod，只需要将旧 Pod 替换为新的 Pod，就可以完成升级。

3、Pod 副本数：Deployment 通常是多个 Pod 的方式来运行的应用程序，而它的数量可以通过控制器来运行和管理，这些 Pod 通常会在多个节点上分布。而 DaemonSet 则只是在每个 Node 上运行一个 Pod 副本，并且通常不需要太多的 Pod 数量。

总之，Deployment 和 DaemonSet 都是控制器，都可以用于部署和管理 Kubernetes 中的 Pod，但它们的目标和用途不同。Deployment 主要用于管理应用程序，DaemonSet 主要用于管理基础设施和系统级别服务。

2、DaemonSet 资源清单文件编写

DaemonSet 资源可以通过如下命令查看相关语法：

```
[root@k8s-master01 ~]# kubectl explain DaemonSet
```

- DaemonSet 资源说明

属性名称	取值类型	取值说明
apiVersion	<string>	Api 版本
kind	<string>	资源类型
metadata	<Object>	元数据
metadata.name	String	控制器的名称
metadata.namespace	String	控制器所属的命名空间，默认值为 default
metadata.labels[]	List	自定义标签列表
metadata.annotation[]	List	自定义注解列表
spec	<Object>	规范 DaemonSet 所需行为的规范
spec.revisionHistoryLimit	<integer>	保留的历史版本，默认是 10
spec.minReadySeconds	<integer>	当新的 pod 启动几秒种后，再 kill 掉旧的 pod
spec.updateStrategy	<Object>	定义更新策略
spec.updateStrategy.type	<string>	可选值：Recreate、RollingUpdate，默认值是 RollingUpdate。 Recreate 是重建式更新，删除一个更新一个。

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

		RollingUpdate 滚动更新，定义滚动更新方式，也就是 pod 能多几个，少几个。
spec.updateStrategy.rollingUpdate	<Object>	滚动更新
spec.updateStrategy.rollingUpdate.maxSurge	<string>	它有两种取值方式，第一种直接给定数量，第二种根据百分比。 假设原本是 5 个，最多可以超出 20%，那就允许多一个，最多可以超过 40%，那就允许多两个
spec.updateStrategy.rollingUpdate.maxUnavailable	<string>	最多允许几个不可用。 假设有 5 个副本，最多一个不可用，就表示最少有 4 个可用。
spec.selector	<Object>	标签选择器
spec.selector.matchLabels	<map[string]string>	匹配 pod 标签，匹配 spec.template.metadata.labels 所定义的标签，必须一模一样。
spec.template	<Object>	Pod 模板

3、DaemonSet 使用案例：部署日志收集组件 fluentd

在日志收集的场景中，使用 DaemonSet 可以确保每个节点都运行一个日志收集组件，而且可以方便地将其面向整个集群进行管理、扩展和操作。

在这里，我们以 fluentd 作为日志收集组件为例，演示如何使用 DaemonSet 在 Kubernetes 中部署 fluentd。

小节目标：

只需要了解 DaemonSet 控制器即可，fluentd 组件后面会详细讲解。

（1）在 master 节点创建 yaml 文件

```
[root@k8s-master01 ~]# vi daemonset.yaml
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: fluentd
  namespace: logging
  labels:
    log: fluentd
spec:
  selector:
    matchLabels:
      name: fluentd
  template:
    metadata:
```

```
        labels:
          name: fluentd
      spec:
        tolerations:
        - key: node-role.kubernetes.io/control-plane
          effect: NoSchedule
        containers:
        - name: fluentd
          image: fluentd:latest
          resources:
            limits:
              memory: 1024Mi
              cpu: 500m
            requests:
              memory: 100Mi
              cpu: 50m
```

（2）更新资源清单文件

```
[root@k8s-master01 ~]# kubectl apply -f daemonset.yaml
daemonset.apps/fluentd created
```

（3）查看创建的 daemonset 控制器

```
[root@k8s-master01 ~]# kubectl get daemonset fluentd -n logging
NAME      DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
fluentd   3         3         3       3            3           <none>          3m44s
[root@k8s-master01 ~]#
```

对上面的输出结果进行说明：

NAME：列出名称空间中 ReplicaSet 资源

DESIRED：显示应用程序的所需副本数，这些副本数是在创建时定义的。这是所需的状态。

CURRENT：显示当前正在运行多少个副本。

READY：显示有多少个副本准备好了。

UP-TO-DATE：显示有多少个最新的副本。

AVAILABLE：显示有多少个可用的副本

AGE：显示应用程序已运行的时间。

（4）查看创建的 pod

```
[root@k8s-master01 ~]#  kubectl get pods -n kube-system -o wide -l name=fluentd
[root@k8s-master01 ~]# kubectl get pods -n logging -o wide -l name=fluentd
NAME            READY   STATUS    RESTARTS   AGE     IP              NODE           NOMINATED NODE   READINESS GATES
fluentd-2lhqg   1/1     Running   0          2m2s    10.244.85.204   k8s-node01     <none>           <none>
fluentd-85774   1/1     Running   0          105s    10.244.58.206   k8s-node02     <none>           <none>
fluentd-nccrx   1/1     Running   0          2m26s   10.244.32.148   k8s-master01   <none>           <none>
[root@k8s-master01 ~]#
```

通过上面可以看到在 k8s 的三个节点均创建了 fluentd 这个 pod。pod 的名字是由控制器的名字-随机数组成的。

4、Daemonset 管理 pod：滚动更新

（1）修改上面文件，增加更新策略

```
[root@k8s-master01 ~]# cat daemonset.yaml
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: fluentd
  namespace: logging
  labels:
    log: fluentd
spec:
  updateStrategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 0
  selector:
    matchLabels:
      name: fluentd
  template:
    metadata:
      labels:
        name: fluentd
    spec:
      tolerations:
      - key: node-role.kubernetes.io/control-plane
        effect: NoSchedule
      containers:
      - name: fluentd
        image: fluentd:latest
        resources:
          limits:
            memory: 1024Mi
            cpu: 500m
          requests:
            memory: 100Mi
            cpu: 50m
```

上面表示 rollingUpdate 更新策略只支持 maxUnavailabe，先删除在更新；

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

因为我们不支持一个节点运行两个 pod，因此需要先创建一个，再删除一个。

（2）更新镜像版本，可以按照如下方法

在更新 Pod 之前，可以在开一个窗口，使用“kubectl get pods -n logging -l name=fluentd -w”命令，动态观察 Pod 状态。

```
[root@k8s-master01 ~]# kubectl set image daemonsets fluentd fluentd=nginx:latest -n logging
daemonset.apps/fluentd image updated

[root@k8s-master01 ~]# kubectl get pods -n logging -l name=fluentd -w
```

```
[root@k8s-master01 ~]# kubectl get pods -n logging -l name=fluentd -w
NAME            READY   STATUS    RESTARTS   AGE
fluentd-2f7jk   1/1     Running   0          54s
fluentd-hl8fv   1/1     Running   0          54s
fluentd-k5hrt   1/1     Running   0          54s
fluentd-gjbsc   0/1     Pending   0          0s
fluentd-gjbsc   0/1     Pending   0          0s
fluentd-gjbsc   0/1     ContainerCreating   0          0s
fluentd-gjbsc   0/1     ContainerCreating   0          0s
fluentd-gjbsc   1/1     Running             0          39s
fluentd-hl8fv   1/1     Terminating         0          2m33s
fluentd-6ptnr   0/1     Pending             0          0s
fluentd-6ptnr   0/1     Pending             0          0s
fluentd-6ptnr   0/1     ContainerCreating   0          0s
fluentd-6ptnr   0/1     ContainerCreating   0          0s
fluentd-hl8fv   1/1     Terminating         0          2m34s
fluentd-hl8fv   0/1     Terminating         0          2m34s
fluentd-hl8fv   0/1     Terminating         0          2m35s
fluentd-hl8fv   0/1     Terminating         0          2m35s
fluentd-hl8fv   0/1     Terminating         0          2m35s
fluentd-6ptnr   1/1     Running             0          45s
fluentd-2f7jk   1/1     Terminating         0          3m18s
fluentd-m26r9   0/1     Pending             0          0s
fluentd-m26r9   0/1     Pending             0          0s
fluentd-m26r9   0/1     ContainerCreating   0          0s
fluentd-m26r9   0/1     ContainerCreating   0          0s
fluentd-2f7jk   1/1     Terminating         0          3m19s
fluentd-2f7jk   0/1     Terminating         0          3m19s
fluentd-2f7jk   0/1     Terminating         0          3m20s
fluentd-2f7jk   0/1     Terminating         0          3m20s
fluentd-2f7jk   0/1     Terminating         0          3m20s
fluentd-m26r9   1/1     Running             0          33s
fluentd-k5hrt   1/1     Terminating         0          3m51s
fluentd-k5hrt   1/1     Terminating         0          3m53s
fluentd-k5hrt   0/1     Terminating         0          3m53s
fluentd-k5hrt   0/1     Terminating         0          3m53s
fluentd-k5hrt   0/1     Terminating         0          3m53s
fluentd-k5hrt   0/1     Terminating         0          3m53s
```

现存的旧Pod

创建第1个新Pod，启动完毕

删除第1个旧pod，后面更新规则类似，直到更新完毕

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。