Kubernetes日志管理方案

前言

- 本文主要讲述kubernetes的日志管理解决方案，对于cluster level logging架构三种比较通用的解决方案做了介绍，包括node agent，sidecar等。

目标

- 学完本课程后，您将能够：
  - 描述kubernetes中常用的几种日志收集解决方案
  - 区分Pod level logging，Node level logging和Cluster level logging
  - 描述EFK日志收集解决方案的架构，各组件的作用

目录

Kubernetes日志管理

- 在正常情况下，容器引擎或运行时提供的本地功能通常不足以支撑完整的日志记录解决方案。例如，如果一个容器崩溃、一个Pod被驱逐、或者一个Node死亡，往往仍然需求可以访问应用程序的日志。
- 因此，日志应该具有独立于Node、Pod或者容器的单独存储和生命周期，这个概念被称为cluster-level-logging 。Cluster级日志记录需要一个独立的后端来存储、分析和查询日志。
- Kubernetes本身并没有为日志数据提供原生的存储解决方案，但可以将许多现有的日志记录解决方案集成到Kubernetes集群中。

Kubernetes日志管理

- 在Kubernetes中，有三个级别的日志：
  - Pod级别日志
  - Node级别日志
  - Cluster级别的日志架构

Pod级别日志

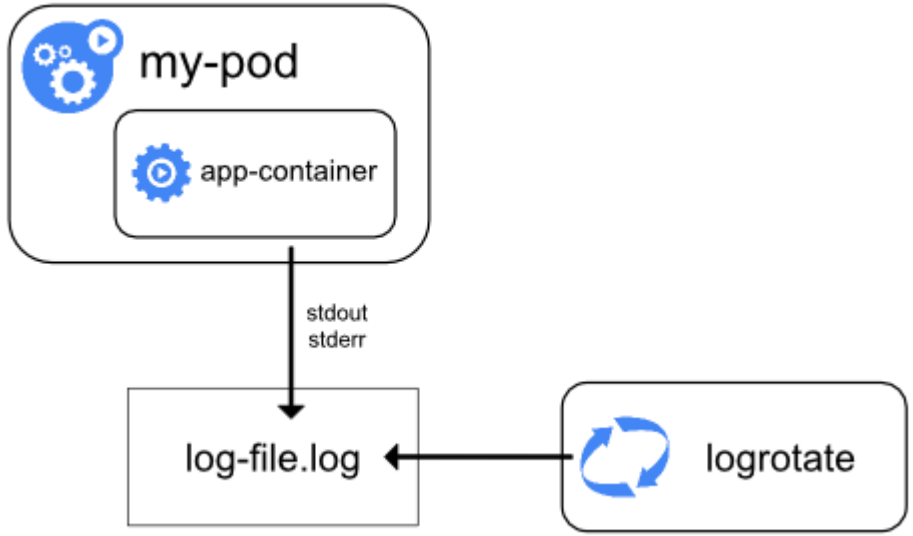
- Kubernetes Pod日志数据输出到标准输出流时，可以使用kubectl logs命令获取容器日志信息。如果Pod中有多个容器，可以通过将容器名称附加到命令来指定要访问哪个容器的日志。
- 例如，在Kubernetes集群中的moniotr命名空间下有一个名称为grafana-core-5f7c6c786b-l4p8l的Pod，就可以通过如下的命令获取日志：

```
[root@master nfs]# kubectl logs grafana-core-5f7c6c786b-l4p8l -n monitor
```

- 但是使用该方式时，当Pod消亡时，这些日志就无法通过这种方式获取了。当你只有少量的Pod的时候，这种方式就足够了。 你可以快速查看Pod的健康状况，而不需要搭建一套大的日志收集集群。

Node级别日志

- 容器化应用写入stdout和stderr的任何数据，都会被容器引擎捕获并被重定向到某个位置。例如，Docker容器引擎将这两个输出流重定向到某个日志驱动，该日志驱动在Kubernetes中配置为以json格式写入文件。
  - 注：Docker json日志驱动将日志的每一行当作一条独立的消息。该日志驱动不直接支持多行消息，所以需要其它工具处理多行消息。



Node级别日志

- Node级别的日志比Pod级别的日志更有持续性。即使一个Pod被重启了，它之前的日志也会保留在节点上。但是如果一个Pod被从一个节点上驱逐了，那么它的日志数据也会被驱逐。
- 节点级日志记录中，需要重点考虑实现日志的切割和压缩，以此来保证日志不会消耗节点上所有的可用空间。
- 每一个节点（node）的日志收集存储在一个JSON文件中。在日志收集过程中这个文件会变得非常大，所以你需要使用logrotate的功能将日志数据切分到多个文件里面，一天执行一次或者当数据增长到某个固定大小（比如100M）后执行。

Cluster级别的日志架构

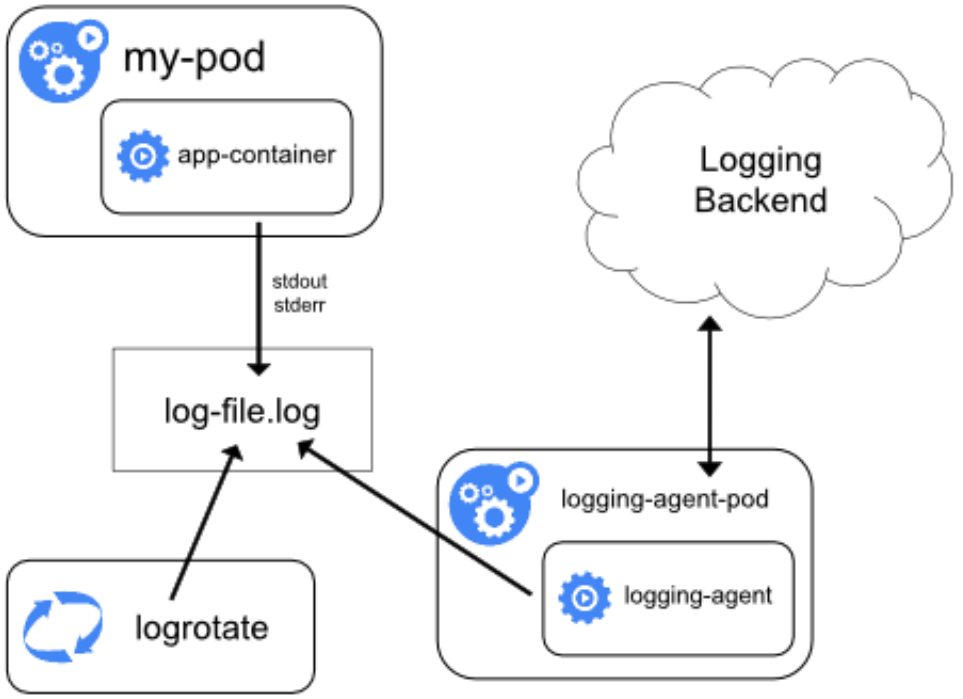
- Node级和Pod级的日志在Kubernetes中是非常重要的概念，但它们本身不是一个实际的解决方案。它们是你构建一个实际的集群级别日志系统解决方案的基石。
- Cluster级别的日志处理系统，与容器、Pod以及Node的生命周期完全无关，这种设计就是为了保证，无论是容器退出、Pod被删除，甚至节点宕机的时候，应用的日志依然可以被正常获取到。

Cluster级别的日志架构

- Kubernetes没有给整个cluster提供一个缺省的日志方案，这一块留给用户和第三方工具来做。对于Cluster级别的日志架构，提供了三种日志方案。
  - 使用运行在各个节点上的节点级日志代理
  - 在应用程序的Pod中，部署专门记录日志的sidecar容器
  - 在应用程序中将日志直接推送到后台

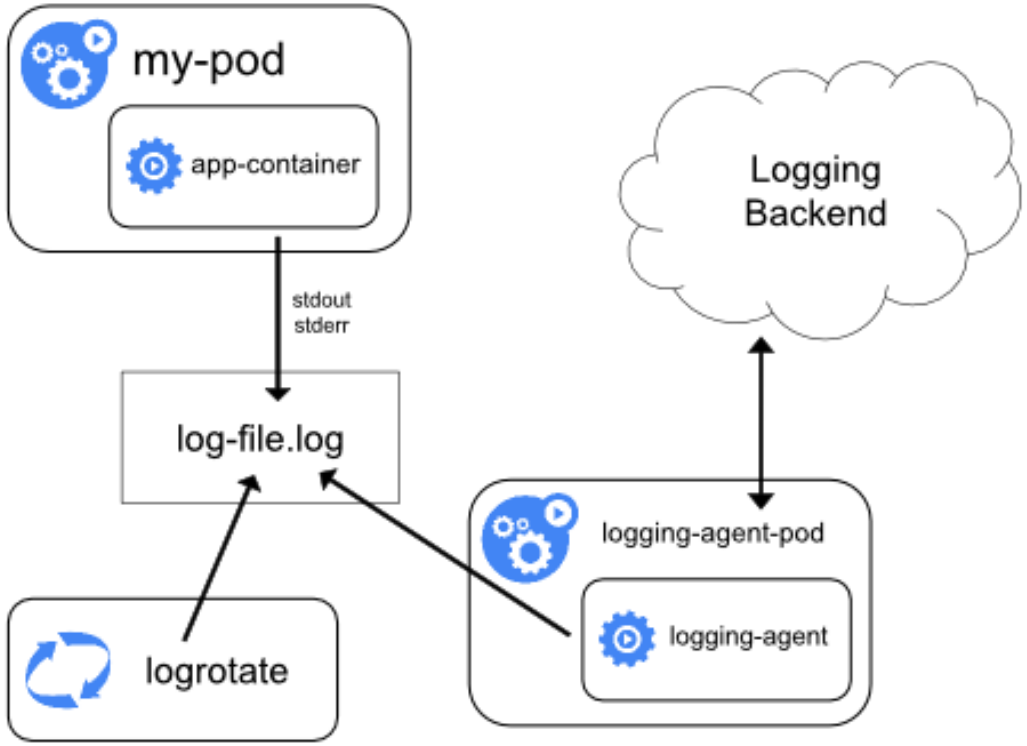
Cluster级别的日志架构 - 节点日志代理

- 节点日志代理：
  - 在Node上部署logging agent，将日志文件转发到后端存储里保存起来。
  - Logging agent是专门的工具，它会暴露出日志或将日志推送到后台。 通常来说，logging agent是一个容器，这个容器可以访问这个节点上所有应用容器的日志目录。因为日志代理必须在每个节点上运行，所以一般采用Daemonset的方式进行部署。



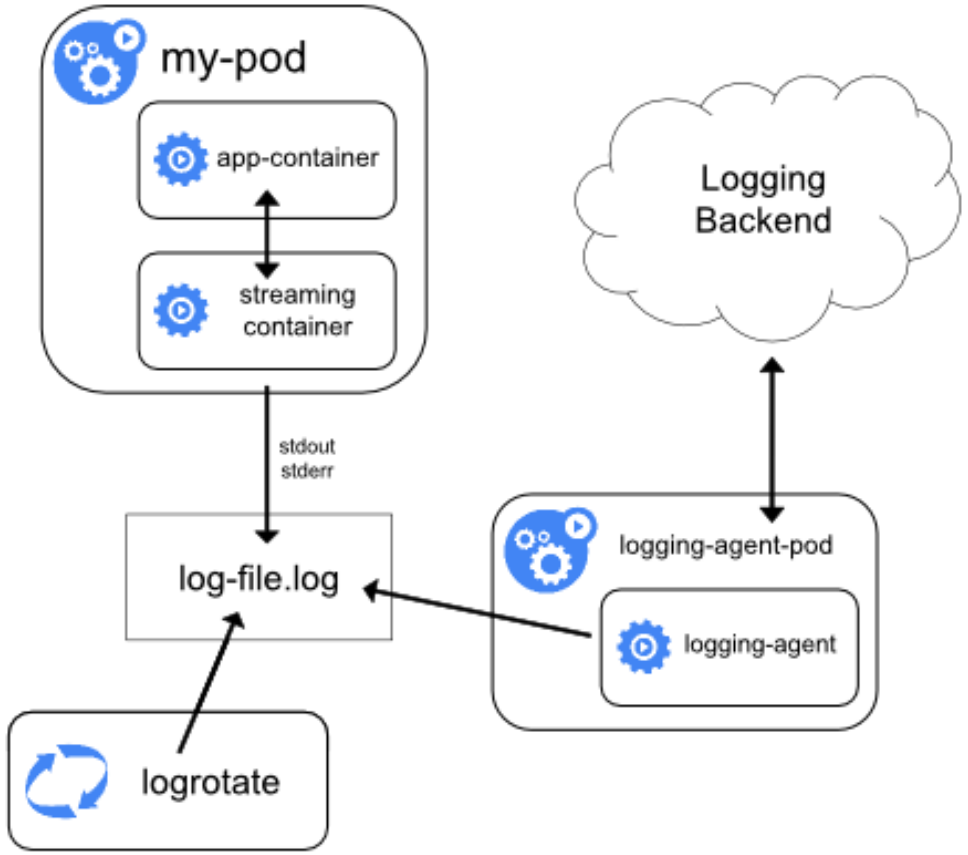
Cluster级别的日志架构 - 节点日志代理

- 节点日志代理：
  - 对于Kubernetes集群来说，使用节点级的日志代理是最常用和被推荐的方式，因为在每个节点上仅创建一个代理，并且不需要对节点上的应用做修改。但是，节点级的日志仅适用于应用程序的标准输出和标准错误输出。
  - 在Node上部署logging agent最大的优点在于一个节点只需要部署一个agent，并且不会对Pod有任何的侵入性，这个方案是社区里最常用的一种。



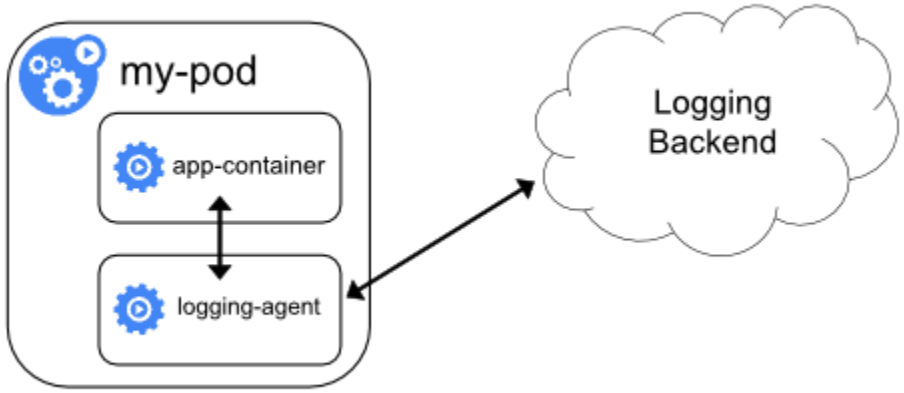
Cluster级别的日志架构 - sidecar方案一

- 使用sidecar容器：
  - sidecar容器将应用程序日志传送到自己的标准输出。这样我们就可以采取前一种方案使用节点上部署的logging agent收集日志后转发出去。
  - 由于sidecar跟主容器之间是共享Volume的，所以这里的sidecar方案的额外性能损耗不高，也就是多占用一点CPU和内存罢了。



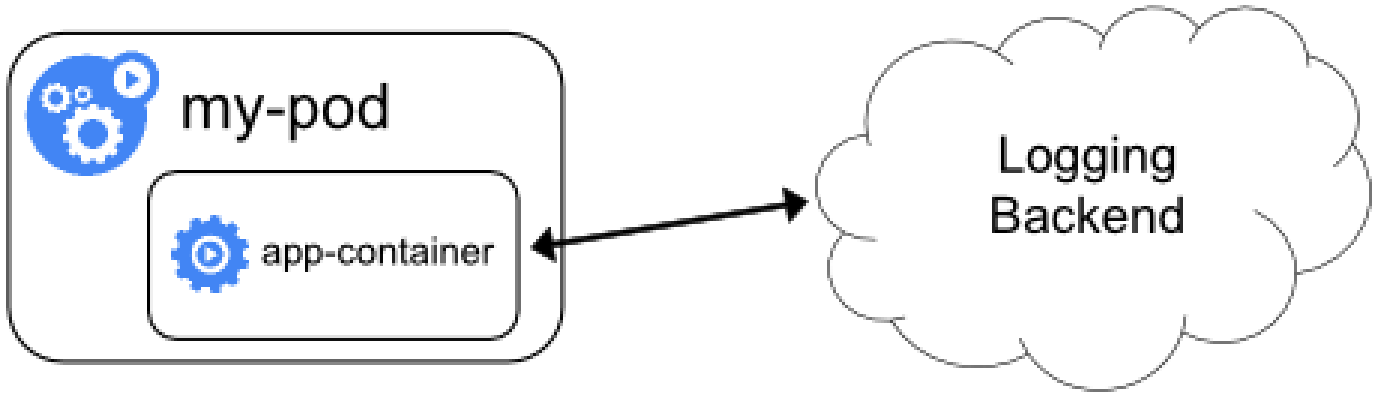
Cluster级别的日志架构 - sidecar方案二

- 使用sidecar容器：
  - sidecar容器运行一个日志代理，配置该日志代理以便从应用容器收集日志。
  - 如果节点级的日志代理对环境来说不够灵活，可以在sidecar容器中创建一个独立的、专门为应用而配置的日志代理。



Cluster级别的日志架构 - 应用推送日志

- 在应用程序中将日志直接推送到后台：
  - 通过暴露或推送每个应用的日志，您可以实现集群级日志记录；然而，这种日志记录机制的实现已然和Kubernetes关系不大。



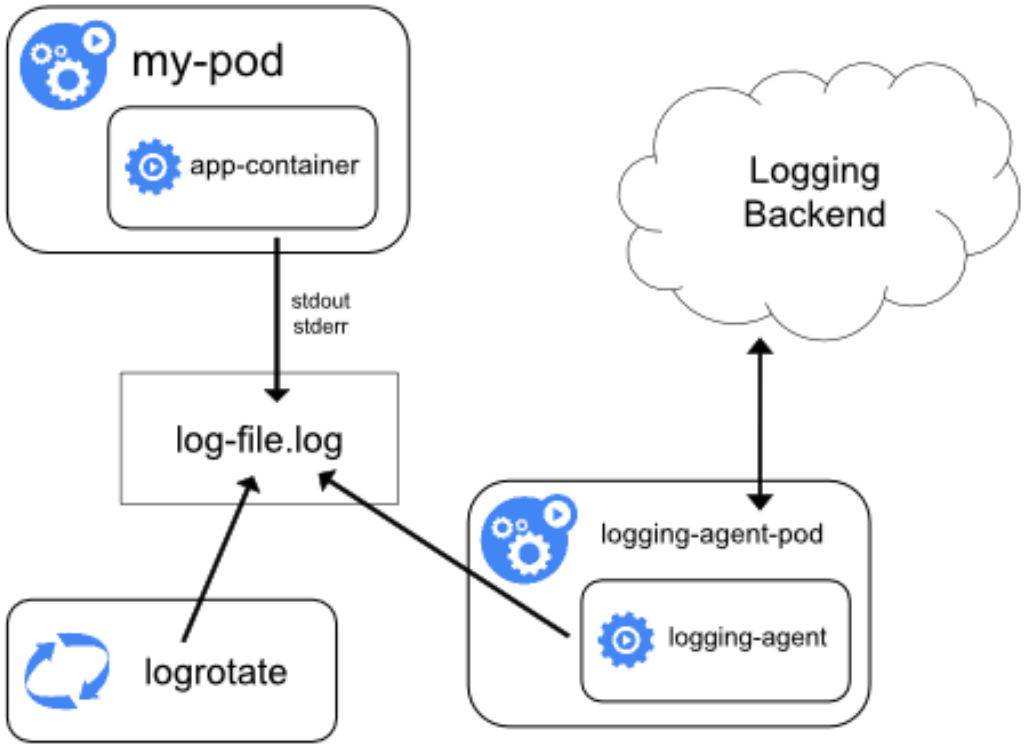
目录

EFK日志管理解决方案

- EFK是一个Elasticsearch、Fluentd和Kibana的组合。
  - Elasticsearch是一个搜索引擎，负责存储日志并提供查询接口；
  - Fluentd负责从Kubernetes搜集日志并发送给Elasticsearch；
  - Kibana提供了一个Web GUI，用户可以浏览和搜索存储在Elasticsearch中的日志通过暴露或推送每个应用的日志。

EFK日志管理方案架构

- EFK采用的是节点日志代理代理的方式，即右图所示架构：
  - 在该架构中，使用Fluentd作为logging-agent，然后把日志转发到远端的Elasticsearch里保存起来供将来进行检索。



EFK日志管理方案基础配置

- 以下为fluentd-es-ds.yaml配置文件中的一段，我们注意volume字段，包括：
  - Pod下Spec中的volumemouts字段：

```
spec:
      volumeMounts:
      - name: varlibdockercontainers
        mountPath: /var/lib/docker/containers
        readOnly: true
```

  - 外部Volumes中的参数：

```
      volumes:
      - name: varlibdockercontainers
       hostPath:
         path: /var/lib/docker/containers
```

- 可以看出外部Volumes，即Host下的/var/lib/docker/containers被挂载到了Fluentd的Pod下/var/lib/docker/containers，Elasticsearch可以从集群中Node上不同的fluentd agent获取日志。

实验任务

- 实验任务
  - 请按照手册2.18章节完成日志相关实验，包括：
    - 使用sidecar输出日志
    - EFK的简单使用

本章总结

- 本章主要讲述了kubernetes的日志解决方案，包括
  - 什么是Pod level logging
  - 什么是Node level logging
  - 什么是Cluster level logging，常用的架构有哪些


谢 谢
www.huawei.com