## Kubernetes容器集群中的日志系统集成实践

Kubernetes是原生的容器编排管理系统，对于负载均衡、服务发现、高可用、滚动升级、自动伸缩等容器云平台的功能要求有原生支持。今天我分享一下我们在K8s集群中日志管理的实践方案。在这个方案中，除了Docker和K8s，主要还涉及的技术包括：fluentd，elasticsearch，Kibana和Swift。

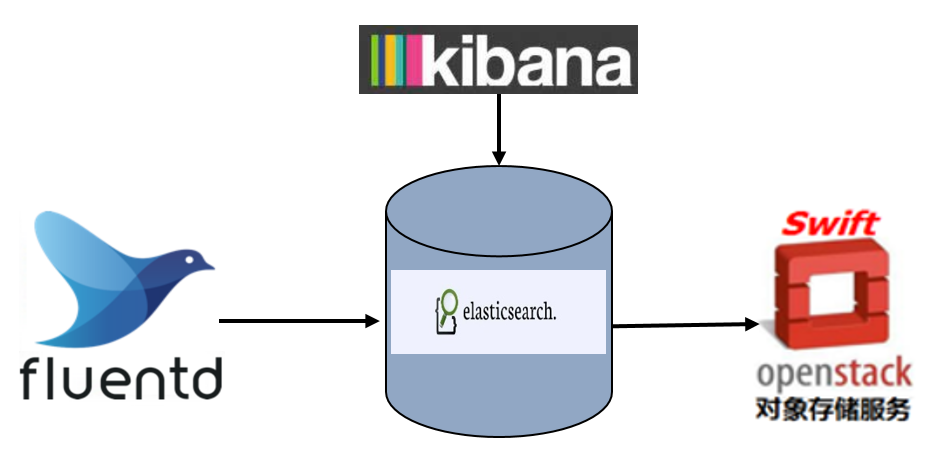


Fig00-K8s日志系统中涉及的技术

**评估容器云平台日志系统的标准：**

1. 易扩展：能够支撑集群规模的增长

2. 开销低：尽量占用较少的系统资源

3. 入侵小：尽量不需要改动应用容器和云平台系统

4. 大集中：将所有分布在各个主机节点上的日志集中在一起分析和查询

5. 易部署：方便自动化部署到分布式集群中

6. 易定制：方便处理不同日志格式，方便对接不同的存储方式

7. 实效性：日志在产生之后需要能在短时间内即可以进行查看分析

8. 社区活跃：方便未来的维护和更新，方便功能扩展

**Fluentd的介绍：**

Fluentd是一个实时日志收集系统，它把JSON作为日志的中间处理格式，通过灵活的插件机制，可以支持丰富多样的日志输入应用、输出应用、以及多种日志解析、缓存、过滤和格式化输出机制。

**Fluentd的架构：**

Fluentd将JSON作为数据处理的中间格式，通过插件式的架构可扩展地支持不同种应用或系统作为日志源和日志输出端。假设有M种输入源Wordpress，MySQL，Tomcat，…；N种输出端MySQL，MongoDB，ElasticSearch，…那么处理日志的代码模块由MxN减少为M+N。我们在Kubernetes集群中收集日志主要用到<https://hub.docker.com/r/fabric8/fluentd-kubernetes/> 这个镜像和<https://github.com/fabric8io/docker-fluentd-kubernetes> 这个plugin。

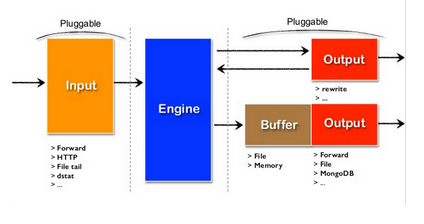


Fig01-Fluend架构

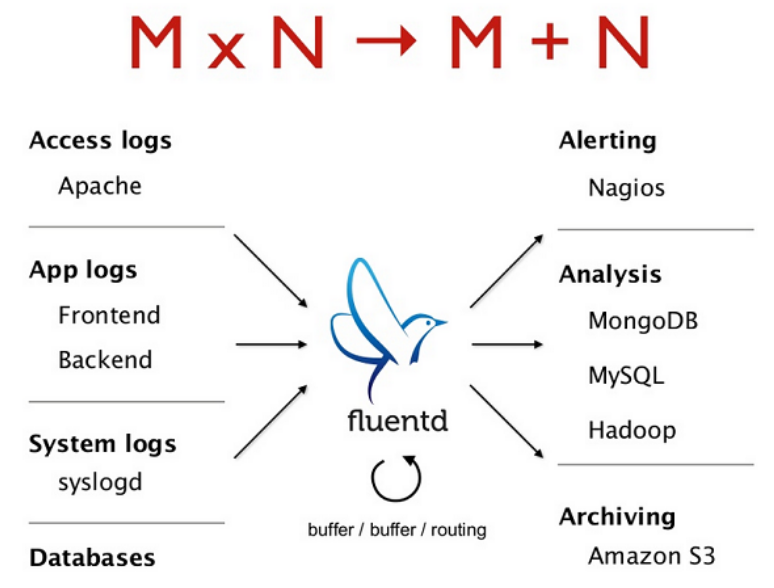


Fig02-Fluentd功能

**Fluentd的特点：**

1. 将JSON作为统一的中间层日志格式做日志处理
2. 基于Ruby的日志收集工具：较基于JRuby的logstash的footprint小
3. 兼容的输入源输出端基本和logstash相当
4. 性能相关的部分为C代码：速度较快
5. 支持插件扩展：Input, Parser, Filter, Output, Formatter and Buffer

**Fluentd在K8s集群中的部署架构：**

每个node节点上都要有fluentd-elasticsearch这个pod，有两种方式支持：1. 放在/etc/kubernetes/manifest下，用脚本自动启动；2. 用daemonset启动。这两种模式都是为了保证在每一个K8s集群节点上都有一个fluentd的驻留Pod运行来收集日志。K8s中DaemonSet这个API对象就是为了驻留Pod而设计的。

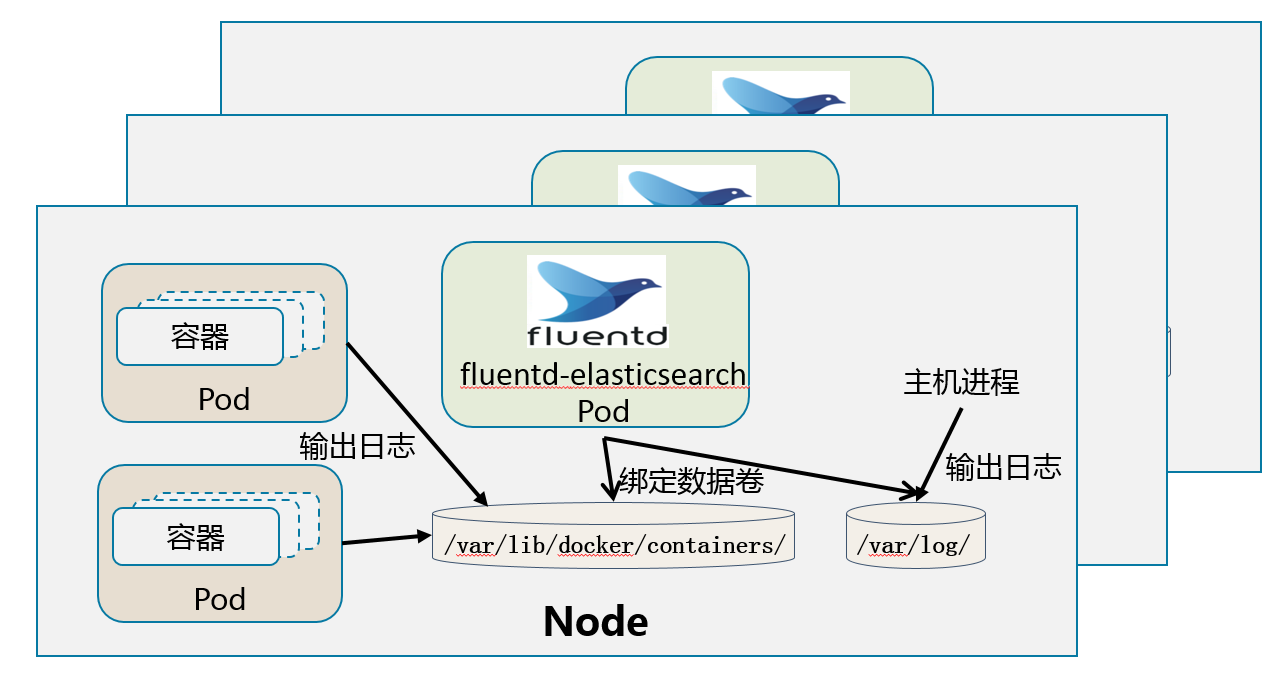


Fig03-Fluentd在K8s集群中的部署架构

**在Kubenetes中的部署案例yaml:**

在K8s集群中部署fluentd时，可以采用类似下面的yaml文件，将使用docker镜像fluentd-elasticsearch的Pod部署到每一个K8s节点上。

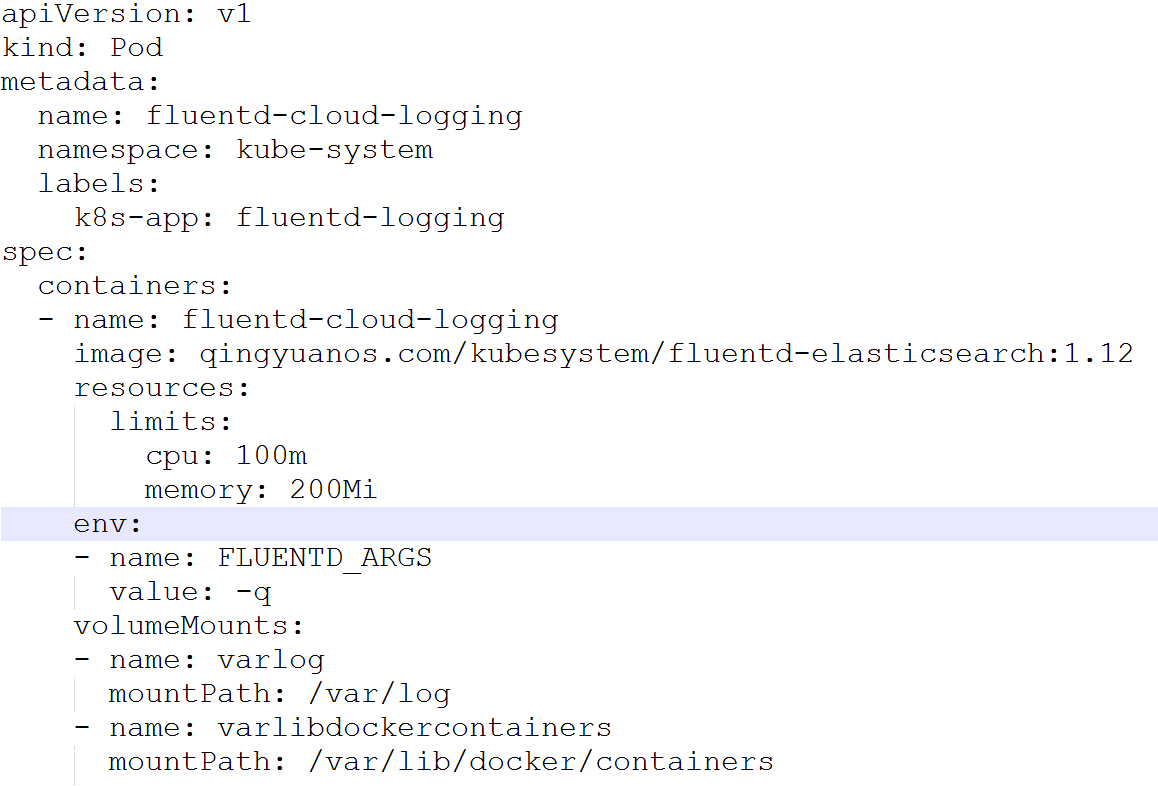


Fig04-Fluentd在K8s集群中的部署yaml

**Fluentd pod的运行时状态：**



Fig05-Fluentd在K8s集群中的运行状态

**选用Fluentd的理由：**

开销低：核心代码为C，插件代码为Ruby，不需要打包JDK

入侵小：用Pod部署，不干扰应用容器和主机服务

易部署：使用容器镜像作为单容器Pod部署

易定制：很方便增加和更改适合自己应用的插件

**ElasticSearch**

Elasticsearch是一个实时的分布式搜索和分析引擎。它可以用于文档存储，全文搜索，结构化搜索以及实时分析，与常见的互联网应用最典型的应用场景是日志分析查询和全文搜索。

**ElasticSearch的架构：**

在ElasticSearch中有三类节点：第一类是Data Node，用来存储数据，ElasticSearch中对于一份数据可以有多个副本，以提供数据高可用能力；第二类是Client Node，或者叫查询节点，提供对查询的负载均衡；第三类是Master Eligible node，或者叫索引节点，这些节点可以被选举为Master Node，而Master Node会控制整个ElasticSearch集群的状态。

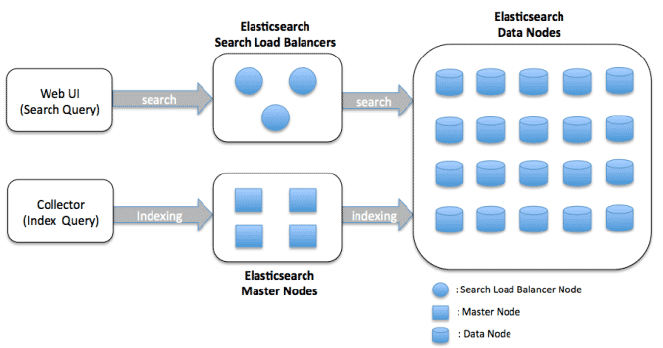


Fig06-Elasticsearch的架构

**ElasticSearch在K8s中的部署架构：**

我们在Kubernetes中，三类节点都是一个包含同一个镜像Pod

**elasticsearch-cloud-kubernetes，**区别只是启动时的环境role不一样。查询和索引节点需要提供对外的Web服务，需要发布为一个Kubernetes Service。数据节点需要绑定一个持久化存储，我们用K8s PV创建出存储卷，数据节点上面通过K8s PVC绑定到相应的数据卷。PV的实际存储可以是NFS，GlusterFS等共享存储。

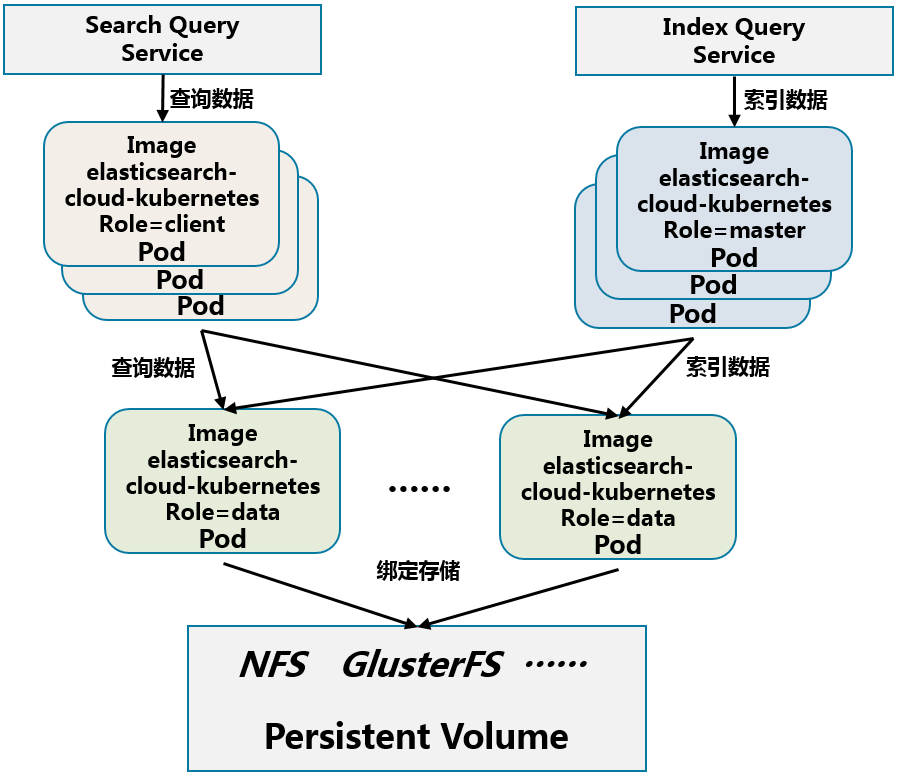


Fig07-ElasticSearch在K8s中的部署架构

**ElasticSearch的特点：**

搜索引擎：基于Apache Lucene的全文搜索引擎，作为开源搜索引擎应用案例开始比solr更流行

文档数据库：可以作为文档数据库使用，存储文档大数据，日志大数据

实时数据分析查询系统：支持大数据量的实时分析查询

完全分布式：可随着节点数水平扩展存储量和查询速度

高可用：可以自动检测破坏的分片，自动重新平衡各个节点上存储的分片数据，可备份冷数据到对象存储

支持插件扩展：可自定义插件支持不同的备份目标

**ElasticSearch与传统关系数据库的类比：**

ElasticSearch与传统数据的概念有许多类似之处，下面是ElasticSearch与传统关系型数据库的对比。

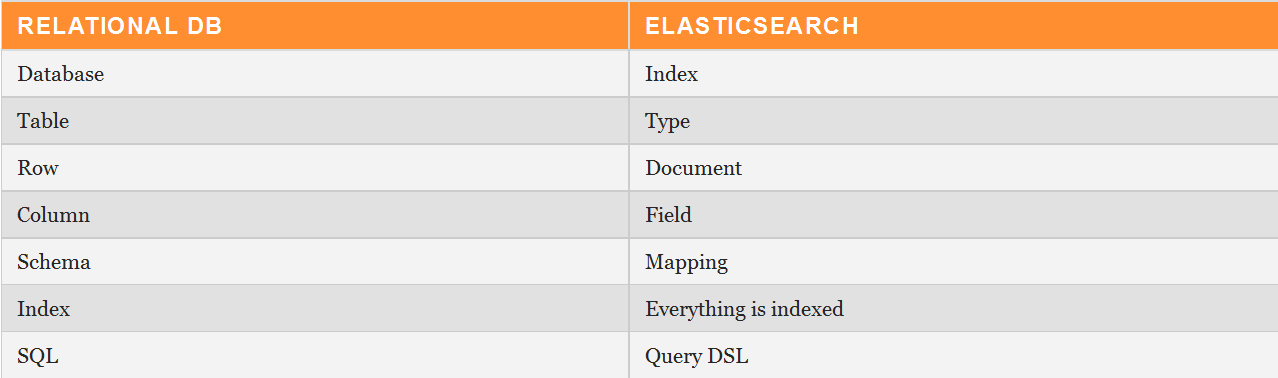


Fig08-ElasticSearch与传统数据库的对比

**ElasticSearch在K8s集群中的部署:**

在K8s集群中部署ElasticSearch，我们会部署类似图中的3种节点：es-data类是用来存放索引数据的；es-master类是用来提供索引写服务的；es-client是用来提供索引查询服务的。

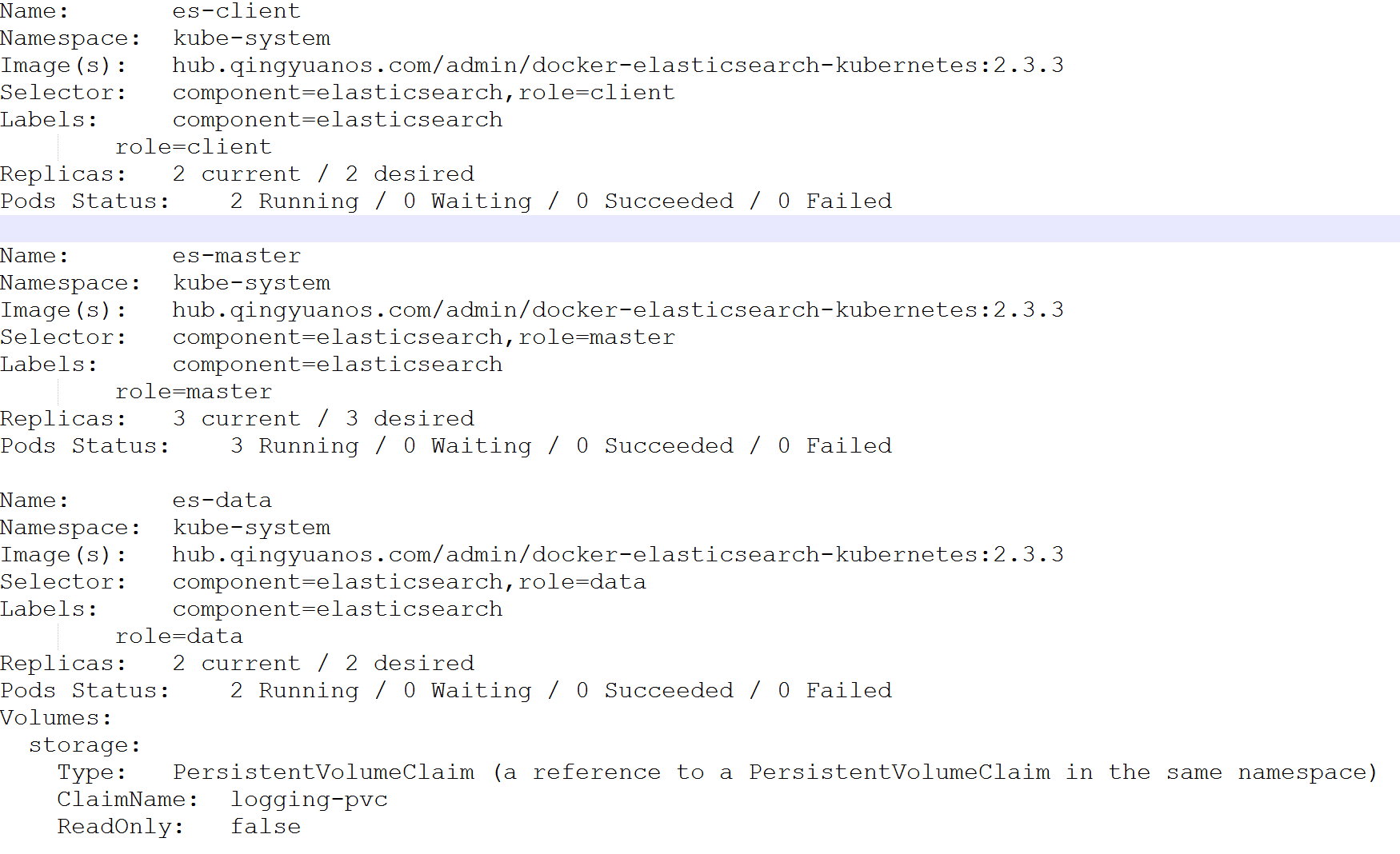


Fig09-ElasticSearch在K8s集群中的部署

**ElasticSearch数据在K8s集群中的持久化存储：**

在部署es-data节点的时候，他们的数据卷我们是以共享存储卷挂载的，这里采用PVC/PV的模式挂载一个NFS的PV提供数据卷，如下图所示。

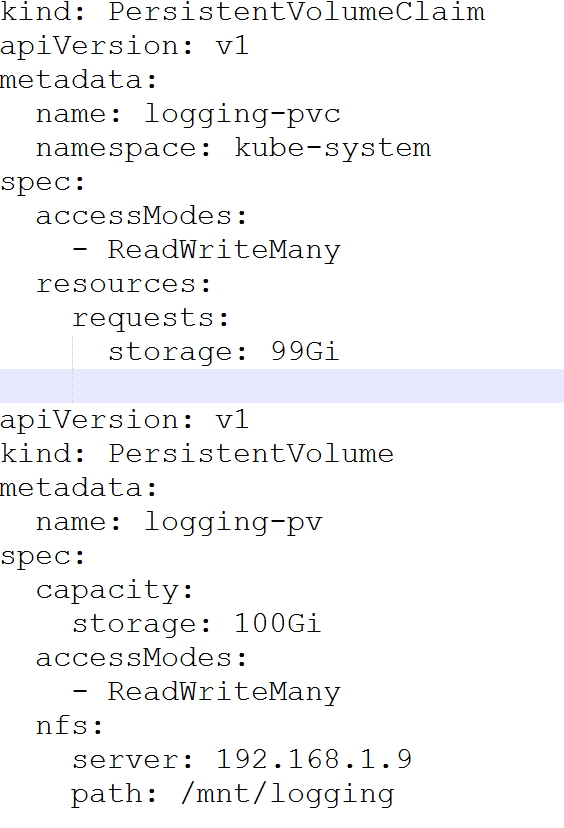


Fig10-ElasticSearch数据在K8s集群中的持久化存储

**日志的备份和恢复**

ElasticSearch允许对于单个索引或整个集群做备份和恢复。备份恢复所支持的目标存储仓库类型包括：

**S3仓库：将AWS S3作为备份仓库**

安装命令：

sudo bin/elasticsearch-plugin install repository-s3

创建仓库示例：

PUT \_snapshot/my-s3-repository-1

{

"type": "s3",

"settings": {

"bucket": "s3\_repository\_1",

"region": "us-west"

}

}

**Azure仓库：将Azure作为备份仓库**

安装命令：

sudo bin/elasticsearch-plugin install repository-azure

创建仓库示例：

PUT \_snapshot/my-azure-repository-1

{

"type": "azure",

"settings": {

"container": "backup-container",

"base\_path": "backups",

"chunk\_size": "32m",

"compress": true

}

}

**HDFS仓库：将HDFS作为备份仓库**

安装命令：

sudo bin/elasticsearch-plugin install repository-hdfs

创建仓库示例：

PUT \_snapshot/my-hdfs-repository-1

{

"type": "hdfs",

"settings": {

"uri": "hdfs://namenode:8020/",

"path": "elasticsearch/respositories/my\_hdfs\_repository",

"conf.dfs.client.read.shortcircuit": "true"

}

}

**GCS仓库：将Google Cloud Storage作为备份仓库**

安装命令：

sudo bin/elasticsearch-plugin install repository-gcs

创建仓库示例：

PUT \_snapshot/my-gcs-repository-1

{

"type": "gcs",

"settings": {

"bucket": "my\_bucket",

"service\_account": "\_default\_"

}

}

**作为私有云部署的环境，多数基于OpenStack的IaaS层，可以采用OpenStack的对象存储Swift作为备份。**

Swift仓库：将OpenStack Swift作为备份仓库

安装命令：

sudo bin/elasticsearch-plugin install org.wikimedia.elasticsearch.swift/swift-repository-plugin/2.1.1

创建仓库示例：

PUT \_snapshot/my-gcs-repository-1

{

"type": "swift",

"settings": {

"swift\_url": "http://localhost:8080/auth/v1.0/",

"swift\_container": "my-container",

"swift\_username": "myuser",

"swift\_password": "mypass!"

}

}

**选用ElasticSearch的理由：**

易扩展：可以随着增加节点水平扩展存储容量和索引能力

大集中：将所有Pod和容器的数据集中在一起方便查询和分析

易部署：使用容器镜像作为单容器Pod部署

易定制：很方便增加和更改适合自己应用的插件

实效性：日志在产生之后需要能在短时间内即可以进行查看分析

社区活跃：ElasticSearch社区越来越活跃，大有赶超Solr之势

**Kibana**

Kibana在Kubernetes中的部署：

Kibana跟ElasticSearch的集成相对来说比较直观，利用https://hub.docker.com/r/fabric8/kibana4/镜像，设置好ELASTICSEARCH\_URL参数就可以，Kibana是一个部署在K8s集群中的Web前端服务，而它引用ELASTICSEARCH\_URL这个环境变量作为资源使用。



Fig11-在K8s集群中部署Kibana

**整体日志管理系统的架构：**

在K8s集群中的每个节点上运行一个fluentd的容器，收集容器的日志发送给到ElasticSearch集群中。ElasticSearch集群会保存一周的日志作为热数据以供实时分析和查询，用户可以通过Kibana查看任意Node，Namespace，Service，Pod和容器的数据。对于超过一周的日志数据，ElasticSearch会自动备份到Swift对象存储的相应Bucket中。

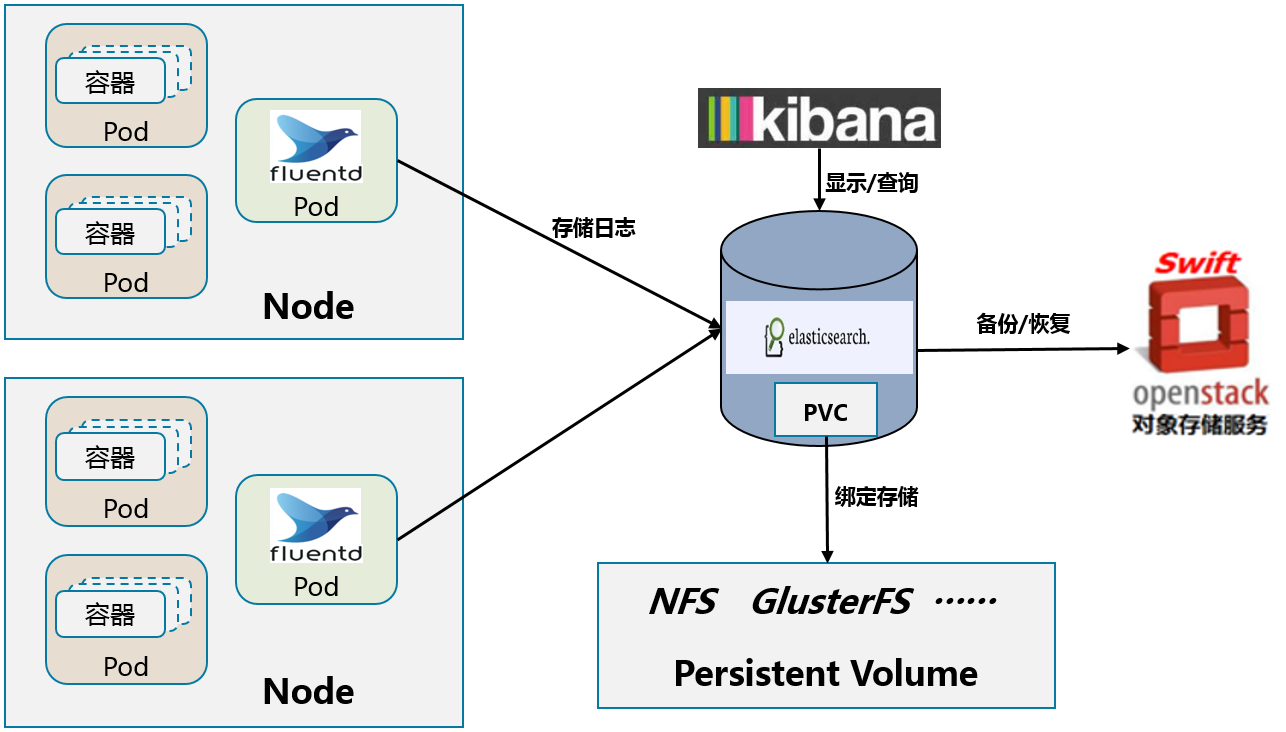


Fig12-整体K8s日志管理系统的架构

**问题和回答**

**Q:**

**请问K8s宿主机的日志是如何收集的？**

**A:**

**跟收集容器的日志是类似的，事实上容器的日志也是从主机的日志目录收集过来的。**

**Q: 如果把移动设备的整机日志输入系统，尤其是移动设备需要注意哪些问题？产生日志目前想到有两种方案：（1）使用app转发给fluentd（2）使用syslog，个人感觉第二个更合适，或者还有其他更好的方案么**

**A:**

**抱歉，我们比较关注的是云平台服务器端的日志，移动设备的日志没有研究过。如果是移动设备日志通过服务器的API上传到服务器了，那么就是一样的处理。一般我们的理解，移动设备的日志是通过应用自己的一些日志程序，定期压缩发送到服务器或者第三方的日志手机平台，然后在服务器端，当作普通的服务器应用日志来处理，只不过要打上移动设备和用户的相关Tag。**

**Q:**

**非常感谢，请问你们是自己内部使用还是作为公有云呢**

**A:**

**我们主要是帮助客户搭建私有云和容器云平台做PaaS使用**

**Q:**

**ElasticSearch 是可以设置备份周期的时间吧？如果我想保留一个月的日志来进行查询？**

**A: 可以设置。但是要通过自己的脚本或者crontab任务来实现。ES目前主要提供的是通过REST API创建、删除备份和通过保留的备份恢复某个集群。**

**Q:**

**对比，logst 快了几倍？怎么没使用syslog−ng?或者做过比较，有缺点？**

**A:**

**fluentd相对于logstash应该是快了2倍以上。当然logtail更快。但是从扩展性和部署容易角度，我们没有选择logtail。syslog-ng没试验过，相对来说，fluentd在K8s集群中的案例很多。。**

**Q:**

**请问如何处理集群自我恢复机制，比如elasticsearch-master,elasticsearch-client 挂了**

**A:**

**这一块儿目前主要是设置discovery的重试次数（es参数：ping\_retries）和超时时间（es参数：ping\_timeout）**

**Q: 请问目前es的规模多大呢？有没有遇到过es规模大以后性能的问题？谢谢。**

**A: 私有云部署的规模大小个不一样，我们现在最大在8台虚拟机，几T的存储，还没碰到瓶颈问题。**

**Q:**

**Fluentd可否设置收集容器应用指定目录日志（非标准输出目录），怎么设置？**

**A:**

**容器应用目录在容器内，Fluentd是收集不到的，除非你的输出目录也是外部挂载的的共享目录。如果是一个单纯docker engine管理的节点，可以通过--volumn-from访问另一个容器存储的数据，当然这也是在那个容器-v声明的volumn而不是任意目录；这种方式对K8s集群没什么帮助。**

**Q:**

**es日志的保留策略， 怎么设置呢，是调api删除还是es自带呢**

**A:**

**我们现在用的方式是用时间做索引，然后脚本定时删除。**

**Q: 数据节点PVC/PV 挂载的文件系统是那种呢？实际使用上遇到什么问题没有**

**A: 我们用过的主要是NFS和GlusterFS。最初实现的PV比较弱，PVC不能通过label与PV匹配，只能通过size和访问类型匹配，无法准确选择PV存储。现在最新K8s支持PVC的selector支持选择带有特定label的PV了。**

**Q:**

**请问k8s宿主机的日志是如何收集的**

**A:**

**用相应的不同的Fluentd插件，类似的mount到相应的主机日志目录即可。现在容器的日志也是通过主机收集的。**

**Q:**

**请问你们es在做discover的时候是否用到k8s api接口？是如何实现的？**

**A:**

**设置cluster名字一样的情况下，es集群会自动发现。**

**Q:**

**日志包括容器的捕获的标准输出日志和应用打到日志文件中的日志。对于这两类场景，如何用fluentd实现新启动容器的日志自动发现和收集？**

**A:**

**对于打到日志文件中的日志，原则上建议日志目录是主机绑定上的或是共享目录。日志的自动发现和收集需要通过fluentd的插件，将指定的目录的文件过滤出来，例如标准输出日志肯定在主机的/var/lib/docker/containers/container-id/下。我们集成的fluentd镜像，已经打包配置好了相应的的插件https://github.com/fabric8io/fluent-plugin-kubernetes\_metadata\_filter，可以参考。**

**Q:**

**我们目前也使用了fluentd收集容器日志，收集容器写到log文件中的日志比收集从标准输出的日志要慢，请问你们有什么调优的方法吗？**

**A:**

**文件日志比标准输出慢是正常的，调优fluentd的性能可能要根据fluentd的说明逐渐积累经验先定位哪里慢，再试验加快方法，跟各种系统的性能调优是同样的思路。下面这个链接有些调优建议。 http://docs.fluentd.org/articles/performance-tuning**

**Q:**

**请问如何处理集群自我恢复机制，比如elasticsearch-master,elasticsearch-client 挂了**

**A:**

**我们在K8s集群中，elasticsearch-master和elasticsearch-client都是以Relication Controller或Replication Set方式启动的，系统会自动保证服务的高可用。其他集群也是类似的机制，和一般Web应用的高可用是一样，要有机制保证重启服务，要有机制做服务发现和负载均衡，在K8s集群是靠Relication Controller和Kube-proxy。**

**Q:**

**请问，在K8s集群中的每个节点上运行一个fluentd的容器，这个节点是容器还是部署docker的节点**

**A:**

**这个是主机节点（可能是物理机或虚拟机），就是K8s的Node，部署docker的节点。推荐的官方方法，是通过K8s的DaemonSet做部署。当然也可以自己在每个节点上维持自动的启动脚本，运行一些每个节点都要启动的Pod服务。**

**Q:**

**请问，k8s的master是单点的，你们是否有优化过，现在1.3了，你们平台升级是否是热部署**

**A:**

**我们是用podmaster做至少3节点master高可用部署，api-server是多活的，controllermanager 和schedule是1活2备。平台升级现在是手动的，不会影响运行中的服务。但是现在平台升级需要工程师小心翼翼的手动操作，还不是自动化的。**

**Q:**

**请问fluentd和flume除了开发语言不一样，有什么本质上的区别？以及ES日索引的分片数量建议是？**

**A:**

**fluentd和flume的设计理念是类似的，一个用CRuby，一个用JRuby，与fluentd和logstash的情况类似，fluentd的镜像会小一些，运行时内存消耗会小一些，而flume的镜像因为要打包JDK差不多要几百兆。在围绕容器的Linux环境中，Java的跨平台性本身带来不了特别大优势，反而fluentd镜像小的优势更加明显。另外一个对我们的系统实践意义比较大，就是fluentd跟K8s集群的方案做的简单明了。ES默认的shards数是5，一般的集群情况要根据使用情况测一下，对于不同的index，这个shards数可以是不一样的。Shards的个数尽量不设1吧，设1的话，将来想要增加分片时，移动的数据量太大了。在没有很充分的生产测试经验之前，设置2到5比较好。**