监控K8S集群日志

1. 架构选择

目前，在K8S的日志监控解决方案中，EFK也是较常用的架构。所谓的EFK，即Elasticsearch + Fluentd + Kibana。

其中 Fluentd 负责日志收集、过滤、传输给 Elasticsearch，Elasticsearch 是一个分布式、可扩展、实时的搜索与数据分析引擎，Kibana 是一款开源的数据分析和可视化平台，它是用于和 Elasticsearch 协作。

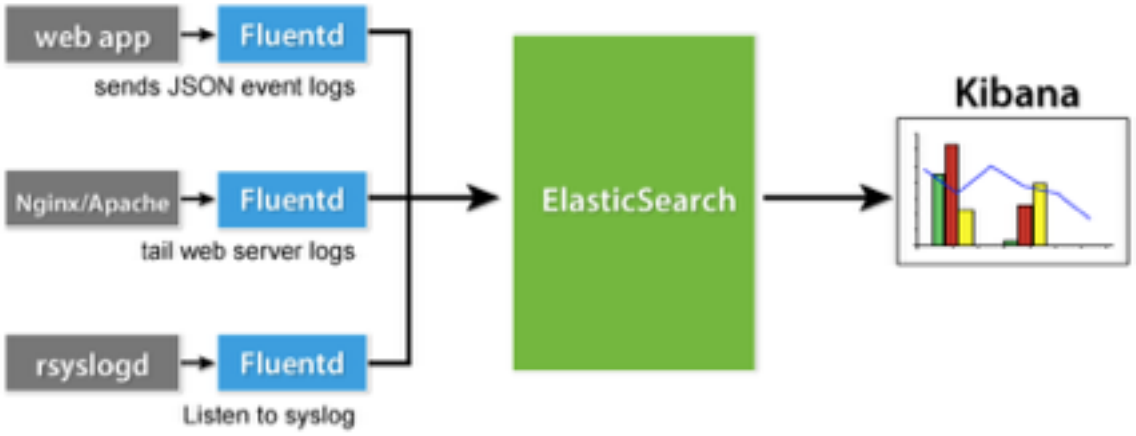

图1 EFK架构图

2. 原理介绍

2.1 Fluentd

Fluentd 是一个日志收集系统，Fluentd 最重要的部分就是插件来管理数据的输入、过滤和输出，其各部分均是可定制化的。

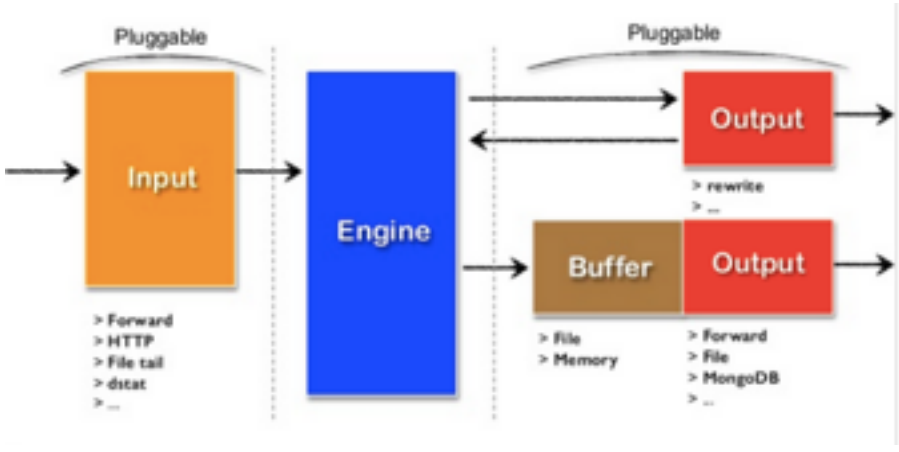

图2 Fluentd 数据流图

2.1.1 Fluentd 输入插件

在 Fluentd 的配置文件 containers.input.conf 中，可以通过配置 source 模块来选择输入插件，通过配置 source 中 @type 来确定输入插件的类型，如下所示是一个 in_tail 类型的输入插件的配置范例

```
<source>
    @id fluentd-containers.log
    @type tail
    path /var/log/containers/*.log
    pos_file /var/log/es-containers.log.pos
    tag kubernetes.*
</source>
```

in_tail 插件的行为类似于 tail -f，通过读取配置中 path 所在路径下的所有满足条件的 log 文件，默认读取的方式是从文件末尾读取日志内容，当所读取的日志文件发生轮转，in_tail 插件将读取新的文件，读取文件的位置信息是放在配置文件的 pos_file 中，如果 Fluentd 重启，Fluentd 就会从 pos 文件中标注的位置开始读取日志文件，保证不丢失重启过程中产生的日志。

输入插件配置中有个非常关键的配置就是tag，当输入插件读取到日志，就会产生一个事件，tag 就是用来决定这个输入插件产生的事件在 Fluentd 中的路由，即由哪些过滤插件或者输出插件进行后续的处理。

2.1.2 Fluentd 解析插件

那么 Fluentd 是如何解析 container 的 log 呢，需要在 source 模块配置中配置 parse 模块， 解析模块也是通过 @type 来决定使用什么类型的解析插件，范例中使用的解析插件的类型是 json，它负责解析日志中的字段。

```
<source>
  ..
  ..
  <parse>
     @type json
     time_key time
     time_format %Y-%m-%dT%H:%M:%S.%NZ
   </parse>
</source>
```

例如下面是 container 产生的一条 json 格式的 log，通过解析模块可得到一个包含 log、stream 和 time 三个字段的值的结构体对象，其中 time 的格式是 time_format。

```
  {"log":"25-Dec-2018 10:36:22.849 client 10.0.0.31 33064: view
default: facebook.com IN A NOERROR - NS NE NT ND NC A 34006
Response: facebook.com 238 IN A
69.171.244.11;\n”,“stream”:"stdout",“time":"2018-12-25T10:36:22.85
443662Z"}
```

2.1.3 Fluentd 过滤插件

在 Fluentd 的配置文件 output.conf 中，通过配置 filter 模块来选择过滤插件，@type 来选择过滤插件的类型，<filter tag> 的 tag 来选择所要处理的 log 事件，过滤插件主要用来修改 log 事件：

（1）选择或丢弃包含某个字段或者某几个字段的 log 事件，例如使用 out_grep 类型的插件，选出 log 内容中包含 client 的 log 事件

```
<filter kubernetes.**>
   @type grep
   <regexp>
     key log
     pattern /client/
   </regexp>
</filter>
```

（2）通过新增字段来丰富 log 事件内容，例如想获取 pod、namespace 和 container 原数据信息，可以配置 kubernetes_metadata 插件

```
 <filter kubernetes.**>
   @type kubernetes_metadata
 </filter>
```

如果想自定义一个字段，可以使用 record_transformer 插件，如果要用 ruby 语句赋值必须设置 enable_ruby 为 true。例如自定义一个名字为 domain_name 的字段，并将log以空格为分隔符分割成的数组，将数据的第8个值赋给 domain_name 字段（2.1.2 log 中的 facebook.com）。

```
<filter kubernetes.**>
  @type record_transformer
  enable_ruby true
  <record>
    domain_name ${record["log"].split[7]}
  </record>
</filter>
```

2.1.4 Fluentd 输出插件

在 Fluentd 的配置文件 output.conf 中，通过配置 match 模块来选择输出插件，@type 来确定所用的输出插件的类型，<match tag> 的 tag 来匹配所要处理的 log 事件，如果想匹配所有的 log 事件，tag 需要设置成 **。

例如想把 log 输出到 Elasticsearch，需要使用 out_elasticsearch 插件，并设置 Elasticsearch 的 host 和 port，设置 logstash_format 为 true，参数 logstash_format 的作用是为了 Kibana 管理 Elasticsearch 的数据索引。

```
<match **>
   @id elasticsearch
   @type elasticsearch
   host elasticsearch-logging
   port 9200
   logstash_format true
</match>
```

2.1.5 Fluentd 缓存模块

需要在 match 模块中配置 buffer 模块，@type 来选择缓存模块的类型。

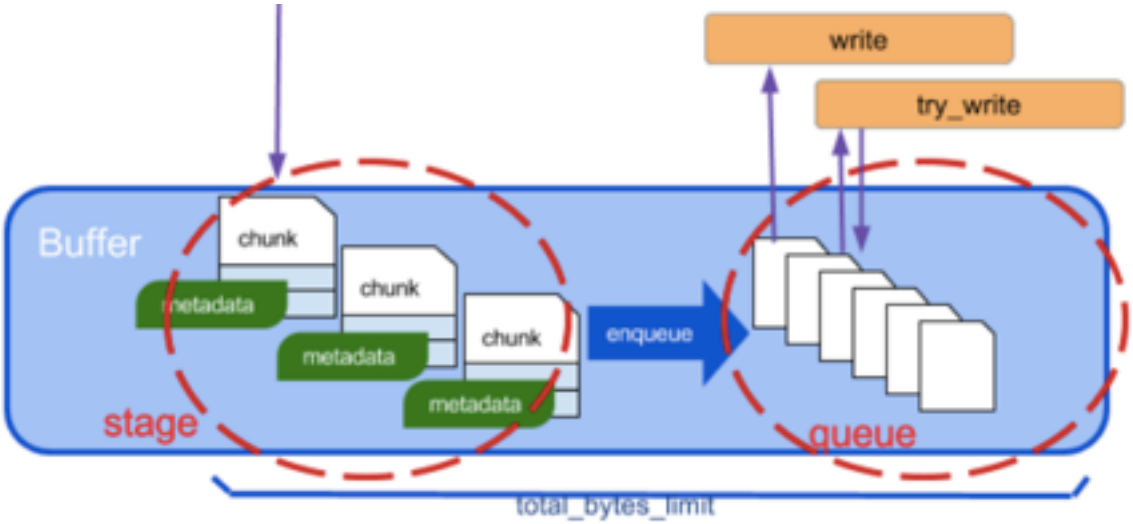

图3 Fluentd buffer原理

缓存插件本质上是一组块（chunk）。块是存储log事件的集合。缓存插件是通过文件（buf_file）或连续内存块（buf_memory）来管理每个块。缓存插件有两个分开的地方来存储块：stage 用来存储 log 事件，而 queue 是块在传输之前等待队列。每个新创建的块都从 stage 开始，然后进入 queue，最后转移到目的地。

下面是配置 buffer 模块的范例，配置中 flush_mode 和 flush_interval 一起使用，表示刷新缓存是5s一次，flush_thread_count 表示并行写数据到 Elasticsearch 的线程。

retry_forever 表示写数据失败会一直重试，retry_type 表示写失败后等待间隔时间模式，其中 exponential_backoff 表示等待时间会几何级数增长，retry_max_interval 表示最长等待时间。

chunk_limit_size 表示每个块的最大值，queue_limit_length 表示等待队列的长度，overflow_action 表示队列满后的行为。

```
<match **>
  ..
  ..
  <buffer>
    @type file
    path /var/log/fluentd_buffers/kubernetes.system.buffer
    flush_mode interval
    flush_thread_count 2
    flush_interval 5
    retry_forever
    retry_type exponential_backoff
    retry_max_interval 30
    chunk_limit_size 2M
    queue_limit_length 8
    overflow_action block
  </buffer>
</match>
```

2.2 Elasticsearch

Elasticsearch 是一个开源的搜索引擎，建立在一个全文搜索引擎库 Apache Lucene 基础之上。 Lucene 可以说是当下最先进、高性能、全功能的搜索引擎库。并且也不仅仅只是一个全文搜索引擎。 它可以被下面这样准确的形容：

- 一个分布式的实时文档存储，每个字段都可以被索引与搜索。
- 一个分布式实时分析搜索引擎。
- 能胜任上百个服务节点的扩展，并支持 PB 级别的数据。

Elasticsearch 是面向文档的，它不仅存储文档，而且索引每个文档的内容，使之可以被检索。在 Elasticsearch 中，你对文档进行索引、检索、排序和过滤，而不是对行列数据。这是一种完全不同的思考数据的方式，也是 Elasticsearch 能支持复杂全文检索的原因。

2.2.1 基础概念

- 文档

文档是一个能被索引的基本单元，它是一个数据对象（如一条 log ）被序列化成 JSON 并存储到 Elasticsearch 中，文档中不仅包含数据，也包含一些文档信息相关的元数据，其中三个必须的元数据如下：

索引（_index）文档在哪存放。

类型（_type）表示文档中对象的类别。

ID（_id）文档唯一标识，是一个字符串， 当它和 _index 以及 _type 组合就可以唯一确定 Elasticsearch 中的一个文档，

- 索引

索引是一个有共同特性的文档集合，在 Elasticsearch 中， 每个字段的所有数据都是默认被索引的。即每个字段都有为了快速检索设置的专用倒排索引，倒排索引用来记录有哪些文档包含了某个单词。一般在文档集合里会有很多文档包含某个单词，每个文档会记录文档编号，单词在这个文档中出现的次数及单词在文档中哪些位置出现过等信息。

- 分片

如果一个索引包含文件过大，对于节点有存储压力，所以需要将一个索引分割成多个分片，分片是一个底层的工作单元 ，Elasticsearch 是利用分片将数据分发到集群内各处的。分片是数据的容器，文档保存在分片内，分片又被分配到集群内的各个节点里。 当集群规模扩大或者缩小时， Elasticsearch 会自动的在各节点中迁移分片，使得数据仍然均匀分布在集群里。

- 节点和集群

节点是一个提供存储数据、参与集群索引和搜索能力的服务器，集群是多个节点的集合。

2.2.2 Elasticsearch 负载均衡

Elasticsearch 尽可能地屏蔽了分布式系统的复杂性。这里列举了一些在后台自动执行的操作：

- 分配文档到不同的容器或分片中，文档可以储存在一个或多个节点中
- 按集群节点来均衡分配这些分片，从而对索引和搜索过程进行负载均衡
- 复制每个分片以支持数据冗余，从而防止硬件故障导致的数据丢失
- 将集群中任一节点的请求路由到存有相关数据的节点
- 集群扩容时无缝整合新节点，重新分配分片以便从离群节点恢复

2.2.3 Elasticsearch 聚合

通过聚合，我们会得到一个数据的概览。通常需要的是分析和总结全套的数据而不是寻找单个文档。

聚合是由桶和指标组成的。聚合可能只有一个桶，可能只有一个指标，或者可能两个都有。也有可能有一些桶嵌套在其他桶里面。

- 桶（Buckets）：满足特定条件的文档的集合。当聚合开始被执行，每个文档里面的值通过计算来决定符合哪个桶的条件。如果匹配到，文档将放入相应的桶并接着进行聚合操作。桶也可以被嵌套在其他桶里面，提供层次化的或者有条件的划分方案。
- 指标（Metrics）：对桶内的文档进行统计计算。桶能让我们划分文档到有意义的集合， 但是最终我们需要的是对这些桶内的文档进行一些指标的计算。分桶是一种达到目的的手段：它提供了一种给文档分组的方法来让我们可以计算感兴趣的指标。大多数指标是简单的数学运算（例如最小值、平均值、最大值，还有汇总），这些是通过文档的值来计算。

下面是一个简单聚合的例子，用来查询数量最多的100个 domain_name，聚合操作被置于顶层参数 aggs 之下，该聚合名字是 top100（也可以是其他随便什么名字），定义单个桶的类型 terms，这个 terms 桶会为每个碰到的唯一词项动态创建新的桶。 例子中配置的词项是 domain_name.keyword 字段，所以 terms 桶会为每个 domain_name 动态创建新桶。order 用来配置桶的排序方式，_count 是按文档数排序。

```
{
  "aggs": {
    “top100”: {
        "terms": {
           "field": "domain_name.keyword",
           "size": 100,
           "order": {
             "_count": “desc"
           }
         }
       }
    }
 }
```

运行聚合后会产生如下结果，名字为 top100 的聚合响应包含多个桶，每个桶都对应唯一词项 domain_name（taobao.com 或 baidu.com）。每个桶也包

括聚合进该桶的所有文档的数量 doc_count。如包含 taobao.com 文档数目是10395个。

```
    {
       "aggregations": {
         "top100": {
           "buckets": [
              {
                "key": "taobao.com",
                "doc_count": 10395
              },
              {
                "key": "baidu.com",
                "doc_count": 6930
              },
        ..
        ..
  }
```

2.3 Kibana

Kibana 是一款开源的数据分析和可视化平台，设计用于和 Elasticsearch 协作。可以使用 Kibana 对 Elasticsearch 索引中的数据进行搜索、查看、交互操作。也可以很方便的利用图表、表格及地图对 Elasticsearch 存储的数据进行多元化的分析和呈现。

Kibana 使大数据通俗易懂。它很简单，基于浏览器的界面便于快速创建和分享动态数据仪表板来追踪 Elasticsearch 的实时数据变化。

2.3.1 Kibana 与 Elasticsearch 连接

需要在 Kibana 配置文件配置 ELASTICSEARCH_URL 配置项，如果 Elasticsearch 暴露的服务名字为 elasticsearch-logging，默认的服务端口为 9200，配置范例如下：

```
containers:
    env:
     - name: ELASTICSEARCH_URL
       value: http://elasticsearch-logging:9200
```

通过 Kibana 暴露的 IP 和 Port 可以登录到 web 页面，在开始使用 Kibana 前，需要告诉 Kibana 想要探索的 Elasticsearch 索引。在

Management 页面增加 index pattern (索引模式)，定义一个索引模式匹配一个或多个 Elasticsearch 索引。默认情况下使用默认的 logstash-* 作为索引模式。

如果您想做一些基于时间序列的数据比较，可以选择索引中包含时间戳的字段。Kibana 会读取索引映射，列出包含时间戳的所有字段。如果索引中没有基于时间序列的数据，则禁用 Index contains time-based events 选项

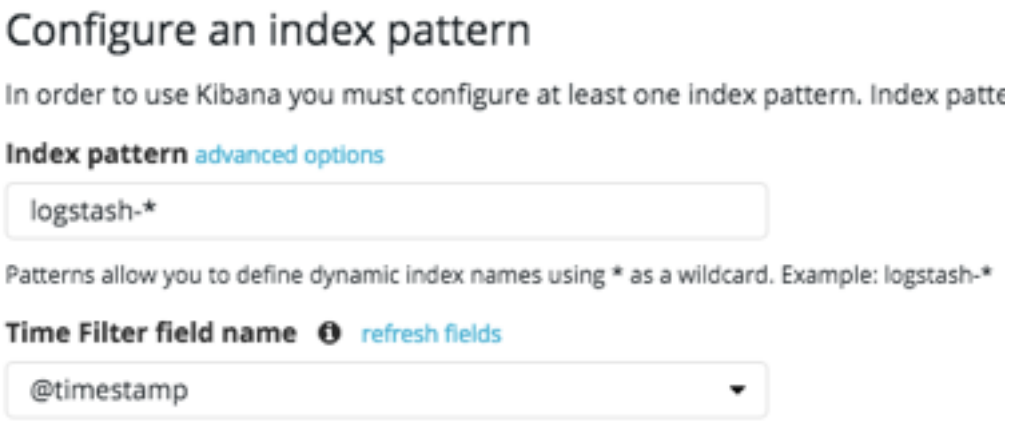

图4 Kibana 创建索引

2.3.2 使用 Kibana 查看数据

在创建索引模式后，可以在 Discover 页面查看 Elasticsearch 数据，可以在 Available Fields 中添加关注的字段到 Selected Fields，例如图5中选择关注的字段为 domain_name 和 log，那么显示数据就只显示这两个字段内容，还可以在右上角修改显示数据的时间段，默认显示15分钟内的数据。

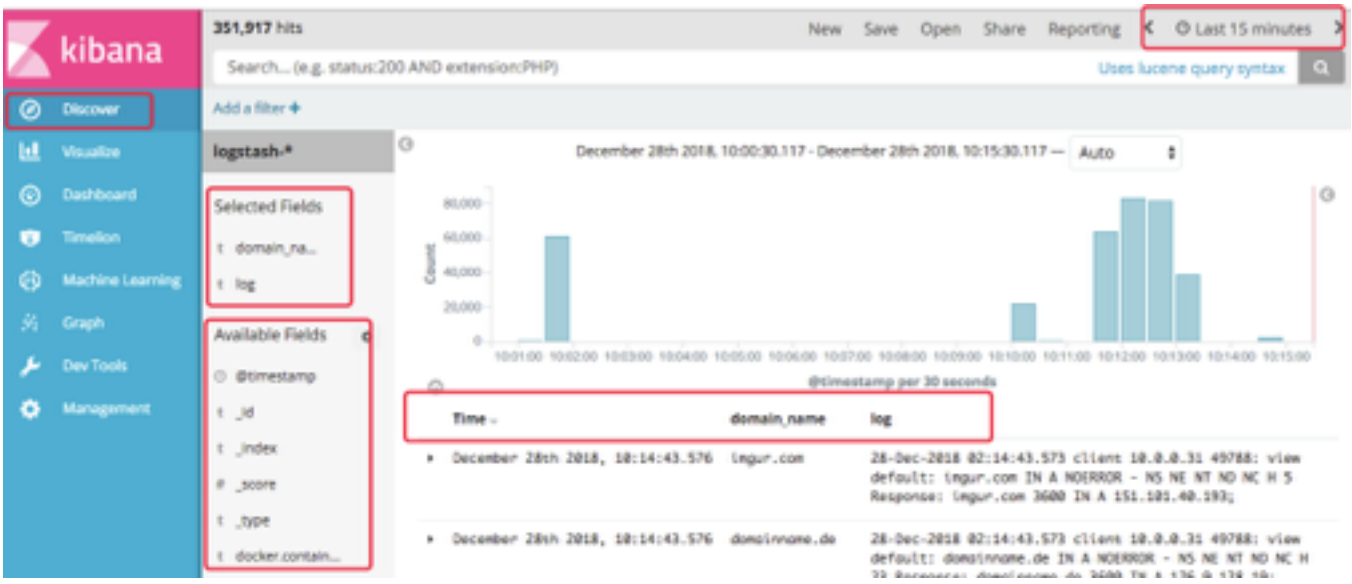

图5 Kibana Discover

2.3.4 Kibana 数据可视化

Kibana 可视化控件 Visualize 基于 Elasticsearch 的查询（Discover）。利用一系列的 Elasticsearch 查询聚合功能来提取和处理数据，可以通过创建图表来呈现关心的数据分布和趋势。

点击左侧导航栏的 Visualize，然后点击 Create new visualization 按钮或者 + 按钮。然后选择想创建的可视化类型，指定一个查询，为可视化获取数据。想要从一个已有的搜索来构建一个可视化视图，只需点击想使用的已有查询名称即可。这将打开一个视图编辑器并加载所选的查询。最后选择指标聚合数据。

例如通过创建一个条形图，查看一个时间段内，一个索引包含的所有数据中字段 domain_name 的top 10，点击 Vertical Bar，选择一个索引模式（logstash-*），默认Y轴是 Count 字段，显示时间段内的所有数据总量，可以通过添加X轴来分隔数据和聚合，X轴通过配置 Aggregation 配置来确定要聚合的类型，由于想看字段，所以选择 Terms 类型，在 Field 配置中选择 domain_name.keyword，在 Order 配置中选择查看顺序，本例中选择递减顺序（Descending），Size 配置中填写 10 代表想看的 top 10。点击运行符号，此时看条形图中可以看到 top 10的域名（X轴）和域名对应的数量（Y轴）。

图6 Kibana 条形图

3 部署

3.1 软件版本

- Elasticsearch： bikecn81/elasticsearch:v5.6.4
- Kibana：bikecn81/kibana:5.6.4
- Fluentd：bikecn81/fluentd-elasticsearch:v2.2.0

3.2 部署步骤

- 创建 namespace: kubectl -f namespace.yaml
- 安装 elasticsearch: kubectl -f elasticsearch/
- 安装 kibana: kubectl -f kibana/
- 安装 fluentd: kubectl -f fluentd/

4 参考文档

Fluentd：https://docs.fluentd.org/v1.0/articles/quickstart
Elasticsearch：https://www.elastic.co/guide/en/elasticsearch/reference/5.6/index.html
Kibana：https://www.elastic.co/guide/en/kibana/5.6/index.html