K8s 容器日志采集 – fluent bit

[liangdong@smzdm.com](mailto:liangdong@smzdm.com)

[容器日志格式 2](#_Toc533089434)

[工作流程 2](#_Toc533089435)

[Input 2](#_Toc533089436)

[Parser 4](#_Toc533089437)

[Filter 4](#_Toc533089438)

[Buffer 4](#_Toc533089439)

[Routing 4](#_Toc533089440)

[Output 5](#_Toc533089441)

[配置采集/var/log/containers/\* 5](#_Toc533089442)

[分析fluent-bit配置 5](#_Toc533089443)

[service 5](#_Toc533089444)

[Include 6](#_Toc533089445)

[Input 6](#_Toc533089446)

[Parser 7](#_Toc533089447)

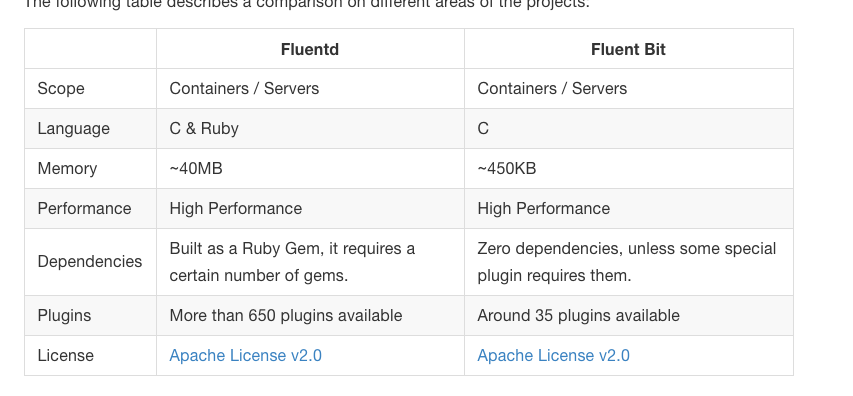
[filter 8](#_Toc533089448)

[Output 10](#_Toc533089449)

[实际操作 11](#_Toc533089450)

Fluentd原先是ruby+c写的，现官方又做了一个新产品叫fluentd bit是纯C写的。

官方的比较：<https://fluentbit.io/documentation/current/about/fluentd_and_fluentbit.html>



观察了一圈，阿里云是用的fluent bit，这个新产品和k8s的结合非常深，配置起来更简单粗暴，性能高，资源占用少，无非是发展时间不长，所以我还是用fluent bit来采集k8s的container输出。

而且这是纯C写的，找问题也有能力，ruby的fluentd就完全不懂了。

官方文档：<https://fluentbit.io/documentation/current/>

# 容器日志格式

观察/var/log/containers/中的容器输出日志，统一采用了json结构，这是docker默认的行为：

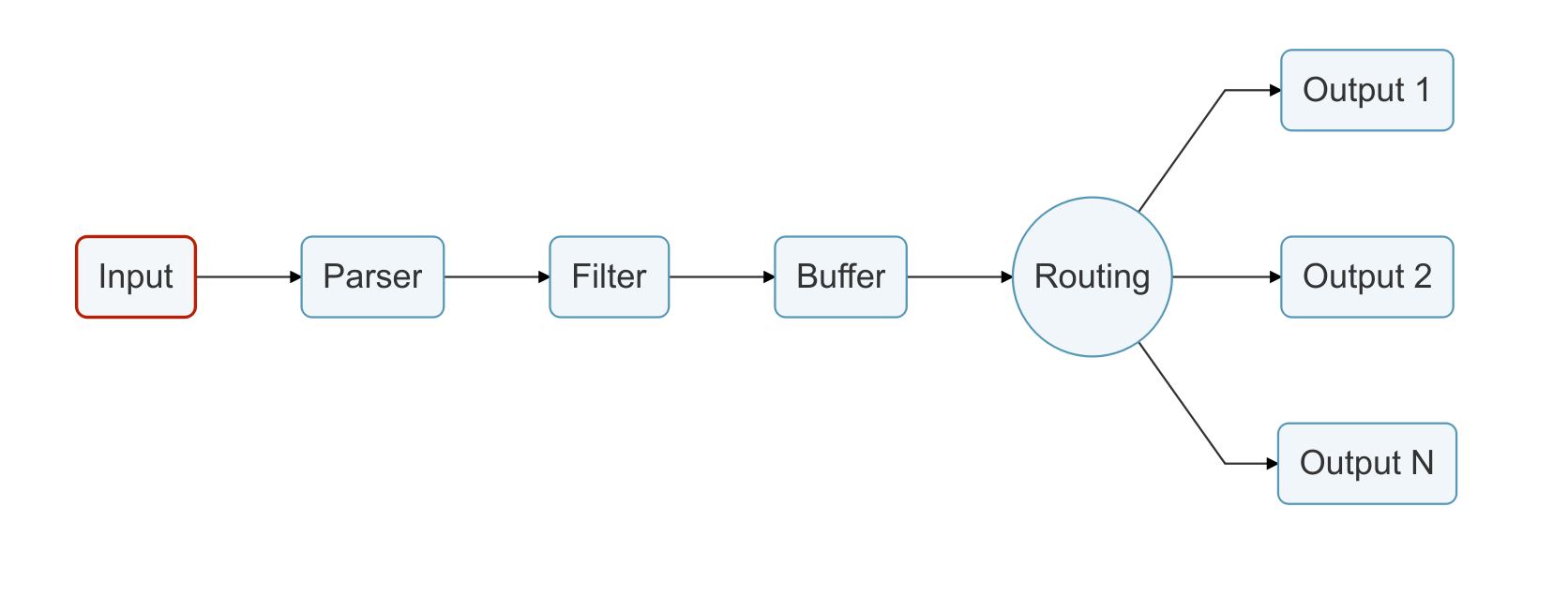
{"log":"I1220 02:52:42.276818 1 controller\_utils.go:1034] Caches are synced for scheduler controller\n","stream":"stderr","time":"2018-12-20T02:52:42.281400963Z"}

三个字段，log，stream，time，对我们有意义的就是log，也就是容器应用输出的原始日志，我们只要把整个json结构丢到ES里面就可以查问题了。

当然，仅仅这3个字段不够，还需要知道容器属于哪个POD，信息越详细越好，这些fluent-bit都通过插件支持好了。

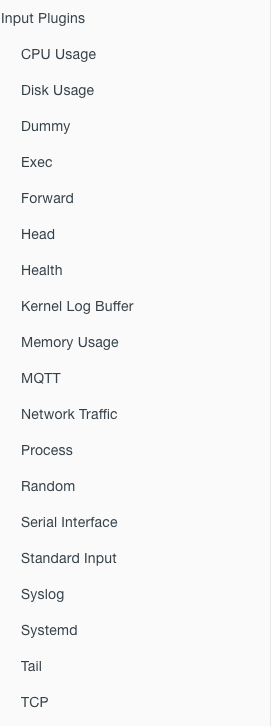
# 工作流程

整体流程：



## Input

有好多input插件可以用，最常见的就是读磁盘上的文件：



## Parser

Input把原始数据取回来，接着就得走parser。

比如input读进来：

192.168.2.20 - - [28/Jul/2006:10:27:10 -0300] "GET /cgi-bin/try/ HTTP/1.0" 200 3395

需要Parser插件把日志行解析成结构化的：

{

"host": "192.168.2.20",

"user": "-",

"method": "GET",

"path": "/cgi-bin/try/",

"code": "200",

"size": "3395",

"referer": "",

"agent": ""

}

常见的parser一种就是json，也就是说原始日志就是个json格式，比如docker的容器日志。

通用的就是regex parser，正则提取所需信息到各种字段里。

## Filter

对结构化数据进行处理，提供了多种插件。

## Buffer

Filter过后的数据，输出到output之前可以缓冲一下，比如打包发送。

## Routing

INPUT的时候可以给日志打tag，然后OUTPUT的时候可以Match匹配对应tag，Match支持通配。

## Output

把日志发到某个地方去，比如ES，Kafaka。

# 配置采集/var/log/containers/\*

我们将参考开源项目：<https://github.com/fluent/fluent-bit-kubernetes-logging>

它配置了daemonset的fluent-bit，直接跑到k8s集群里，采集每个node上的/var/log/containers目录，日志发到ES存储。

打开这个页面：<https://fluentbit.io/documentation/current/configuration/file.html>，我们要用到的基本语法都在里面，我们对照着快速上手。

## 分析fluent-bit配置

接下来，我们分析一下开源的配置文件：<https://github.com/fluent/fluent-bit-kubernetes-logging/blob/master/output/elasticsearch/fluent-bit-configmap.yaml>

### service

|  |
| --- |
| [SERVICE] |
|  | Flush 1 |
|  | Log\_Level info |
|  | Daemon off |
|  | Parsers\_File parsers.conf |
|  | HTTP\_Server On |
|  | HTTP\_Listen 0.0.0.0 |
|  | HTTP\_Port 2020 |

Flush是刷新间隔，buffer里的数据每隔1秒写到output插件里，也就是写到ES里。

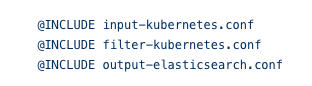
Log\_level是fluent-bit的日志级别

Daemon写Off，因为fluent-bit一会要跑到k8s里，作为container的入口程序，不能后台运行。

没配置log\_file，估计默认fluent-bit输出日志到终端。

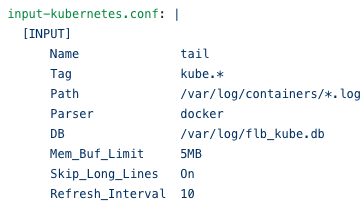
Parsers\_file指向了另外一个配置文件，里面配置所有的parser。

### Include



这是包含其他文件的意思，我们接着看input-kubernetes.conf。

### Input



Name指定了input插件的类型，这里是tail类型，也就是扫描文件增量。

Tail插件的配置说明在这里： <https://fluentbit.io/documentation/current/input/tail.html>

Path用通配符指定要抓取的文件。

Tag就是给采集的日志打个标签，kube.\*中的\*会根据Path被替换为var.log.containers.xxx。

DB是记录哪个文件采集到哪一行了。

一旦buffer里的数据超过Mem\_buf\_limit，tail就会暂停采集，直到buffer数据被flush到output。

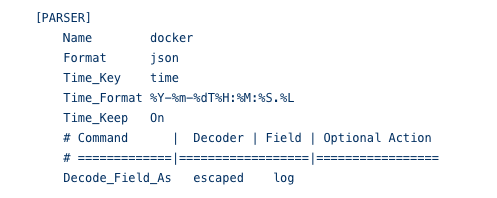
Parser指定了解析器，也就是input -> parser的这个 关系，一会我们会看到docker这个parser。

Skip\_long\_lines跳过太长的行，默认是32K，有其他参数可以配置这个大小。

Refresh\_interval是定时扫描磁盘上的新文件的间隔，这里是10秒。

### Parser

Input把采集的原始数据交给parser，刚才配置的parser叫做docker，在Parser.conf中：



Parser的配置说明在这里： <https://fluentbit.io/documentation/current/parser/>

Name就是给这个parser起个名字，这个parser就是做这样一个事情：



把Input原始日志，按照format解析成一种内部的格式（k-v字典），同时把时间戳加上。

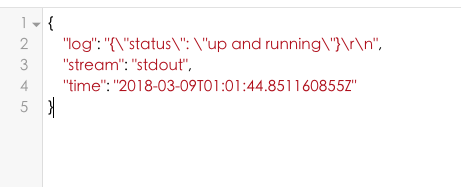
Format是input传来的日志的原始格式，docker日志都是json结构，所以直接json decode即可解析为内部k-v的日志结构。

Time\_key指定了把docker日志中的time字段按照time\_format进行解析得到unix时间戳，time\_keep会time字段不删除，也就是除了内部时间戳以外，在k-v 字典里还有一个time字段。

docker把容器的输出行放在log字段里，但是放之前估计做了一次escape，所以docker日志文件长这样：

{"log":"{\"status\": \"up and running\"}\r\n","stream":"stdout","time":"2018-03-09T01:01:44.851160855Z"}

做json decode后这样：



所以即便使用json parser，仍旧需要对log字段做一次unesacape，所以decode\_fields\_as就是这个作用，它对json decode后的log字段做了一次解转义，完全属于正常操作。

### filter



Parser得到的内部K-v字典，会经过filter。

Filter会根据tag机制去匹配日志，因为source的时候打上了kube.前缀的tag，所以就会匹配这个filter。

文档在这里： <https://fluentbit.io/documentation/current/filter/kubernetes.html>

Name是插件的类型，就是kubernetes插件。

这个插件会根据tag提取出k8s namespace，pod name, container name, container id这些信息，这些在input阶段生成的tag中可以得到，因为tag是根据tail的path动态生成的。

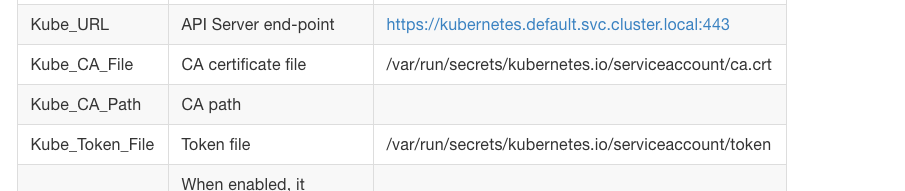
然后这个插件还会监听k8s apiserver，获知这个pod的k8s labels和annotations，都添加到k-v字典中去。

所以，经过这个filter插件，除了原先docker日志中的time,stream,log三个k-v，又会追加一些k8s字段，最终所有这些kv会存到ES中，对于容器日志来说，在ES中的mapping是稳定的。

回到filter配置中：



Kube\_url配置了apiserver的地址，fluent-bit会以daemonset运行在k8s的所有node上，可以直接用in-cluster domain访问apiserver，但需要有足够的权限，我们在配置daemonset的时候会给fluent-bit配置serviceaccount，fluent-bit默认就会加载标准路径下的token。

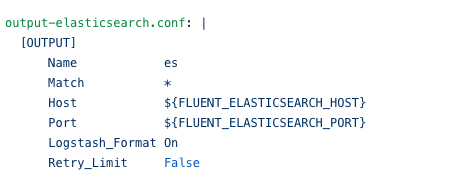


Merge\_log会检查指定的字段，如果是字段是一个json map，就把内部字段提到外部k-v中，这对ES来说就会出现动态mapping问题，也许不同的容器输出的同名字段类型不同，导致ES存储日志失败，我建议off掉。

K8S-Logging.Parser没找到相关文档，保留着吧。

### Output

最后就是输出环节：

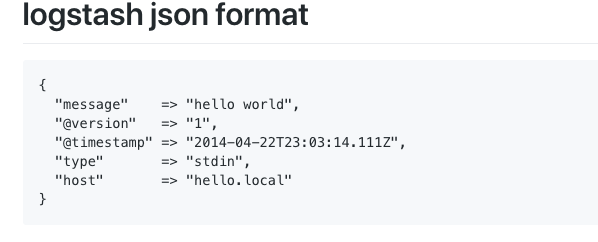


Name就是es插件： <https://fluentbit.io/documentation/current/output/elasticsearch.html>

Match所有的tag，或者kube.\*也可以。

Host,port指定ES地址，这里可以通过${}的形式引用环境变量，稍后在配置daemonset的时候传进去就行。

Logstash\_format就是输出前按logstash的风格输出字段：<https://gist.github.com/jordansissel/2996677>



就是最终ES结构式这样的一个外层结构。

一旦开始logstash\_format，ES的index就会按照logstash- YYYY.MM.DD 的风格滚动，具体都可以根据fluent bit手册配置。

## 实际操作

现在实际操作一下看看。

安装java： apt-get install openjdk-8-jdk

先下载ES： <https://www.elastic.co/downloads/elasticsearch>

修改sysctl -w vm.max\_map\_count=262144

修改config/elasticsearch.yml，让network.host: 0.0.0.0

启动ES：nohup bin/elasticsearch &

根据fluent bit kubernetes项目指引：<https://github.com/fluent/fluent-bit-kubernetes-logging>

先把serviceaccount以及role做好：

$ kubectl create namespace logging

$ kubectl create -f https://raw.githubusercontent.com/fluent/fluent-bit-kubernetes-logging/master/fluent-bit-service-account.yaml

$ kubectl create -f https://raw.githubusercontent.com/fluent/fluent-bit-kubernetes-logging/master/fluent-bit-role.yaml

$ kubectl create -f https://raw.githubusercontent.com/fluent/fluent-bit-kubernetes-logging/master/fluent-bit-role-binding.yaml

然后把fluent bit的配置文件传到configmap里：

kubectl create -f https://raw.githubusercontent.com/fluent/fluent-bit-kubernetes-logging/master/output/elasticsearch/fluent-bit-configmap.yaml

下载fluent bit的daemonset配置：

wget https://raw.githubusercontent.com/fluent/fluent-bit-kubernetes-logging/master/output/elasticsearch/fluent-bit-ds.yaml

修改一下ES的地址就行，其他都是通用的：



然后apply一下，可以看到volume把node的日志目录映射到fluent bit容器里了。

查看pod启动情况：kubectl get pod --all-namespaces，过一会就会拉起来了。

然后看一下fluent bit日志：

root@ubuntu:~/deploy/fluentd# kubectl logs fluent-bit-glpnw -n logging

Fluent-Bit v0.14.9

Copyright (C) Treasure Data

[2018/12/20 09:09:15] [ info] [engine] started (pid=1)

[2018/12/20 09:09:15] [ info] [filter\_kube] https=1 host=kubernetes.default.svc.cluster.local port=443

[2018/12/20 09:09:15] [ info] [filter\_kube] local POD info OK

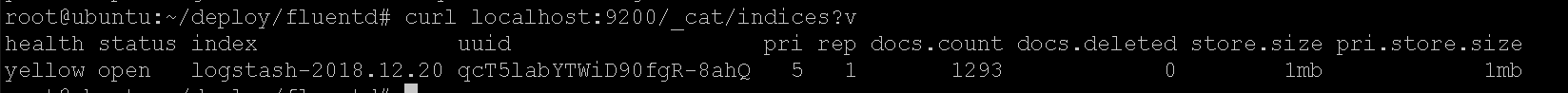
[2018/12/20 09:09:15] [ info] [filter\_kube] testing connectivity with API server...

[2018/12/20 09:09:16] [ info] [filter\_kube] API server connectivity OK

[2018/12/20 09:09:16] [ info] [http\_server] listen iface=0.0.0.0 tcp\_port=2020

一看就是启动成功，连上了apiserver。

然后看一下ES里有日志没：



索引名默认是logstash-日期，打印几条记录看看：

curl localhost:9200/logstash-2018.12.20/\_search?pretty



很完善的日志，遵循logstash+kibana规范。