Logstash的介绍、原理、优缺点、使用、持久化到磁盘、性能测试

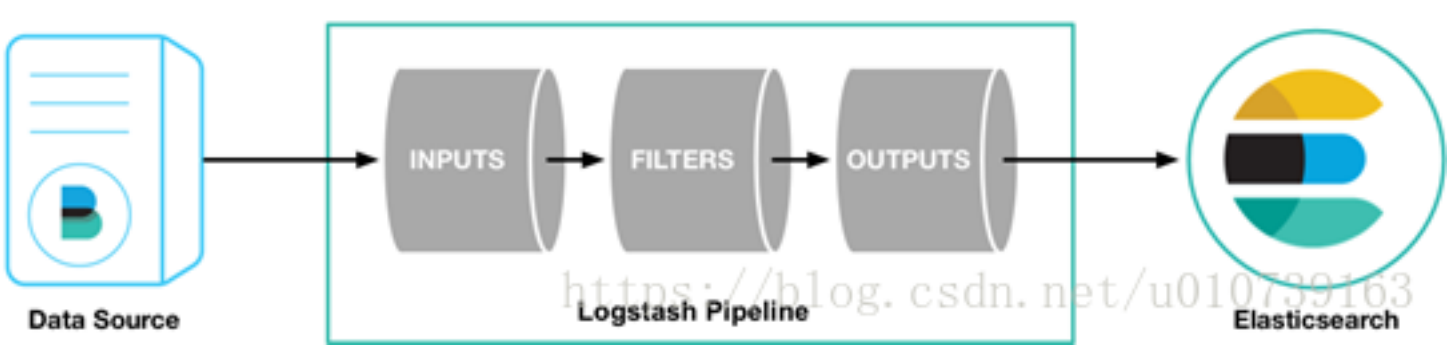
一、ELK介绍

对于日志来说，最常见的需求就是收集、存储、查询、展示，开源社区正好有相对应的开源项目：logstash（收集）、elasticsearch（存储+搜索）、kibana（展示），我们将这三个组合起来的技术称之为ELKStack，所以说ELKStack指的是Elasticsearch、Logstash、Kibana技术栈的结合。

二、Logstash简介

Logstash 是一款强大的数据处理工具，它可以实现数据传输，格式处理，格式化输出，还有强大的插件功能，常用于日志处理。

三、Logstash工作原理



1. **输入，以下是常见得输入内容**
   1) file：从文件系统上的文件读取，与UNIX命令非常相似 tail -0F
   2) syslog：在已知端口上侦听syslog消息进行解析
   3) redis：使用redis通道和redis列表从redis服务器读取。Redis通常用作集中式Logstash安装中的“代理”，该安装将Logstash事件从远程Logstash“托运人”排队。
   4) beats：处理 Beats发送的事件,beats包括filebeat、packetbeat、winlogbeat。
2. **过滤，以下是常见得过滤器**
   1) grok：解析并构造任意文本。Grok是目前Logstash中将非结构化日志数据解析为结构化和可查询内容的最佳方式。Logstash内置了120种模式，您很可能会找到满足您需求的模式！

2) mutate：对事件字段执行常规转换。您可以重命名，删除，替换和修改事件中的字段。
3) drop：完全删除事件，例如调试事件。
4) clone：制作事件的副本，可能添加或删除字段。
5) geoip：添加有关IP地址的地理位置的信息（也在Kibana中显示惊人的图表！）

3. **输出，以下是常见得输出内容**
   1) elasticsearch：将事件数据发送给Elasticsearch。如果您计划以高效，方便且易于查询的格式保存数据..... Elasticsearch是您的最佳选择
   2) file：将事件数据写入磁盘上的文件。
   3) graphite：将事件数据发送到graphite，这是一种用于存储和绘制指标的流行开源工具。http://graphite.readthedocs.io/en/latest/
   4) statsd：将事件数据发送到statsd，这是一种“侦听统计信息，如计数器和定时器，通过UDP发送并将聚合发送到一个或多个可插入后端服务”的服务。如果您已经在使用statsd，这可能对您有用！
4. **编解码器**
   编解码器基本上是流过滤器，可以作为输入或输出的一部分运行。使用编解码器可以轻松地将消息传输与序列化过程分开。流行的编解码器包括json, multiline等。
   json：以JSON格式编码或解码数据。
   multiline：将多行文本事件（例如java异常和堆栈跟踪消息）合并到一个事件中

四、Logstash优点

1. **可伸缩性**
   节拍应该在一组Logstash节点之间进行负载平衡。
   建议至少使用两个Logstash节点以实现高可用性。
   每个Logstash节点只部署一个Beats输入是很常见的，但每个Logstash节点也可以部署多个Beats输入，以便为不同的数据源公开独立的端点。
2. **弹性**
   Logstash持久队列提供跨节点故障的保护。对于Logstash中的磁盘级弹性，确保磁盘冗余非常重要。对于内部部署，建议您配置RAID。

在云或容器化环境中运行时，建议您使用具有反映数据SLA的复制策略的永久磁盘。

3. **可过滤**
   对事件字段执行常规转换。您可以重命名，删除，替换和修改事件中的字段。
4. 可扩展插件生态系统，提供超过200个插件，以及创建和贡献自己的灵活性。

五、Logstash缺点

Logstash耗资源较大，运行占用CPU和内存高。另外没有消息队列缓存，存在数据丢失隐患。

六、Logstash的使用

logstash有两种方式运行，分别是命令行的方式、定义配置文件方式

1. **使用命令行运行一个简单的logstash程序**

```
logstash/bin/logstash -e
'input{stdin{}}output{stdout{codec=>rubydebug}}'
```

在控制台输入 abc ,控制台会输出如下内容:

```
       "message" => "abc",
    "@timestamp" => "2016-08-20T03:33:00.769Z",
          "host" => "iZ23tzjZ"
```

2. **配置语法讲解**
   最基本的配置文件定义，必须包含input 和 output。
   修改logstash.conf文件，如下：

```
# 最基本的配置文件定义，必须包含input 和 output。
# 如果需要对数据进操作，则需要加上filter段
  # 里面可以包含各种数据处理的插件，如文本格式处理 grok、键值定义 kv、字段添加、
      path => ["/var/log/message"]
```

```
        start_position => "beginning"

        path => "/var/datalog/mysystem.log.gz"
```

运行命令：

```
logstash -f logstash.conf
```

七、Logstash持久化到磁盘

当发生异常情况，比如logstash重启，有可能发生数据丢失，可以选择logstash持久化到磁盘，修改之前重启logstash数据丢失，修改之后重启logstash数据不丢失。以下是具体操作：
在config/logstash.yml中进行配置以下内容
queue.type: persisted
path.queue: /usr/share/logstash/data #队列存储路径；如果队列类型为persisted，则生效
queue.page_capacity: 250mb #队列为持久化，单个队列大小
queue.max_events: 0 #当启用持久化队列时，队列中未读事件的最大数量，0为不限制
queue.max_bytes: 1024mb #队列最大容量
queue.checkpoint.acks: 1024 #在启用持久队列时强制执行检查点的最大数量,0为不限制
queue.checkpoint.writes: 1024 #在启用持久队列时强制执行检查点之前的最大数量的写入事件，0为不限制
queue.checkpoint.interval: 1000 #当启用持久队列时，在头页面上强制一个检查点的时间间隔
修改完后，重启logstash

测试步骤:模拟数据写入logstash，重启Logstash

测试结果:数据量没有变少，数据没有丢失

八、Logstash性能测试，模拟数据量1千万条

Logstash有kibana专门的模拟数据插件，以及速率分析工具

操作步骤：

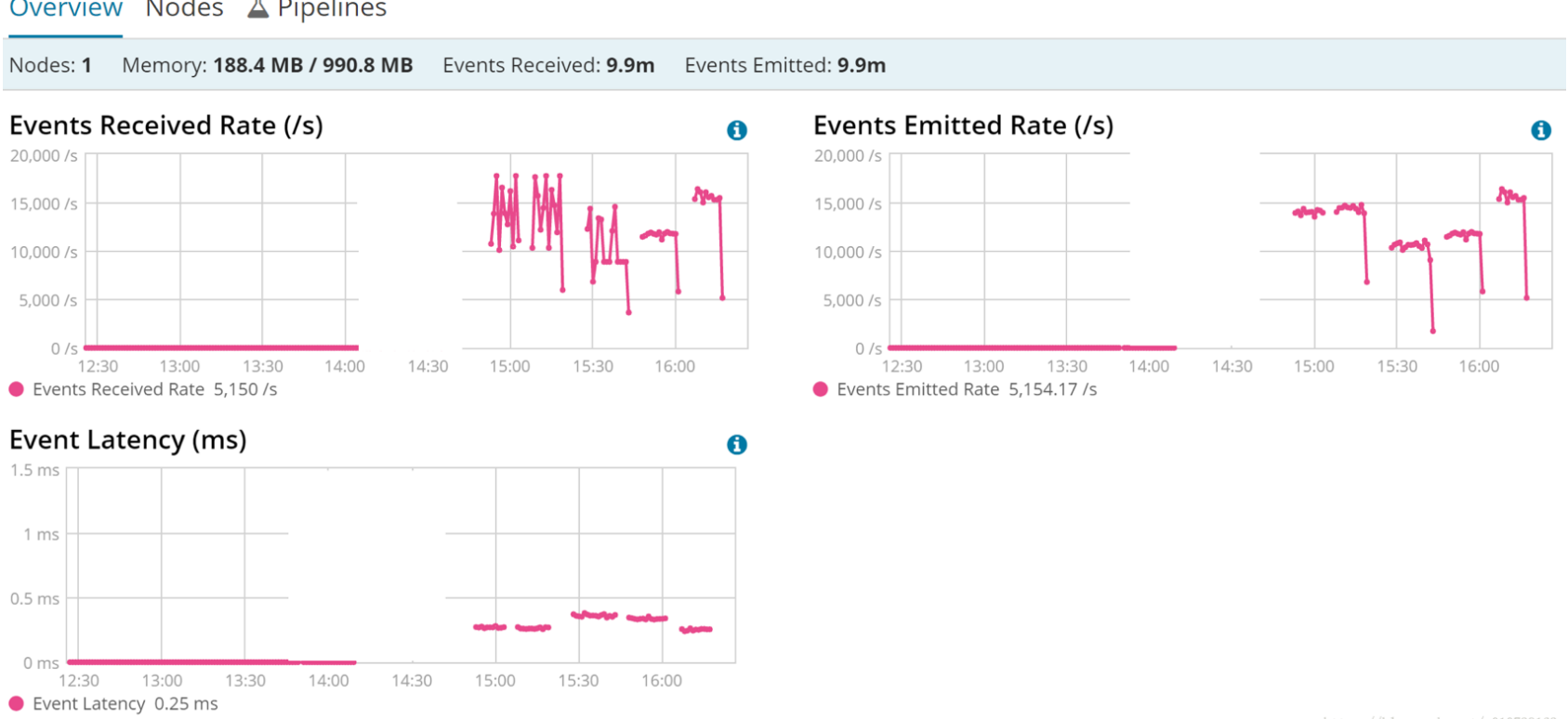
配置Logstash目录下面的pipeline/logstash.conf，配置完重启Logstash

```
count => 10000000 # 模拟数据1千万条

message =>'INFO 2018-07-30 0911
springfox.documentation.spring.web.readers.operation.CachingOperationNameGe
 - Generating unique operation named: getUsingGET1$i' # 模拟数据具体内容

codec=>multiline { #multiline插件，用于合并事件

what=>"previous" #当匹配到空格，合并到上一个事件，如果是next，则合并到下一个事件

match =>{ "message" => "%{LOGLEVEL:level}\s%
{TIMESTAMP_ISO8601:timestamp}\s%{JAVACLASS:class}[%{WORD:method}:%
{NUMBER:line}]\s-\s(?([\s\S]*))" } #通过正则表达式匹配日志

match => [ "timestamp" , "yyyy-MM-dd HHss", "ISO8601" ] #匹配timestamp字段

target => "@timestamp" #覆盖@timestamp

remove_field => "timestamp" #移除timestamp字段

elasticsearch{ #输出到elasticsearch

hosts => ["192.168.2.227:9200"]

index => "logstash-api-%{+YYYY.MM.dd}"
```

九、1千万条数据测试结果

模拟数据量1千万条

通过kibana的monitoring监控



后面四个分别为:使用持久队列过滤、使用持久队列不过滤、使用内存队列过滤、使用内存队列不过滤的情况，估算出下面一张表

	接收速率(event/s)	发送速率(event/s)	管道的发送速率(event/s)	平均延时
使用内存队列,不过滤	15758/s	15745/s	10500/s	0.24ms
使用内存队列,过滤	11982/s	11966/s	7680/s	0.33ms
使用持久队列,不过滤	13760/s	14300/s	9000/s	0.25ms
使用持久队列,过滤	11657/s	10331/s	7000/s	0.39ms

如果有说的不明白或者错误的地方，欢迎指正！