第六讲-ELK 快速构建大型互联网网站流量监控实现

1 分布式带来的变革

1.1 多节点

项目特点：
拆分：SOA,微服务架构
部署：大规模集群网络

试想一下：

目前集群网络只有 100 台服务器。采用人肉运维的方式，处理服务器各种异常。

运维人员：每一台打一个几号，Excel 记录下每一台机器运行特点，如果某一台服务器出现

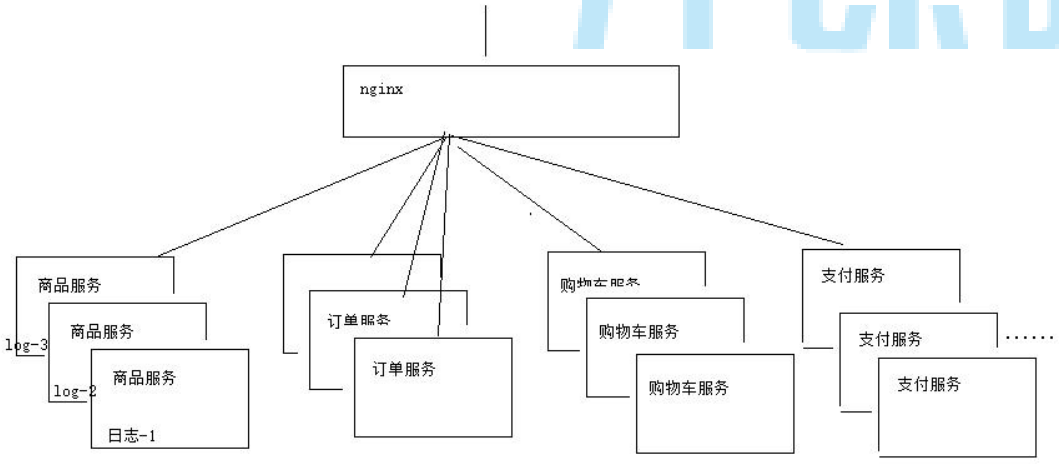
问题，运维人员直接定位机器，查询日志，修复问题。

此时：10000 台 服务器，Excel 表格

有一个问题：如何收集海量服务器节点上的日志，进行分析，处理？

1、监控服务器状态

2、收集一些有用的数据，进行分析。



数据分析：

1、ELK 一周时间搞定的东西

2、hadoop 一个月都搞不定

E: elasticsearch　存储日志数据
L: logstash　　收集日志
K:Kibana　　UI 视图，把收集的日志数据，进行图形化界面方式直接展示给用户

1.2 日志分散

微服务，SOA(面向服务架构)---→ 分布式的架构 ： 日志分散在每一个服务器

1000 台服务器：
A　B　…….. 1000 台(请求恰好打到第 1000 台)
0　0　　1

1.3 运维成本高

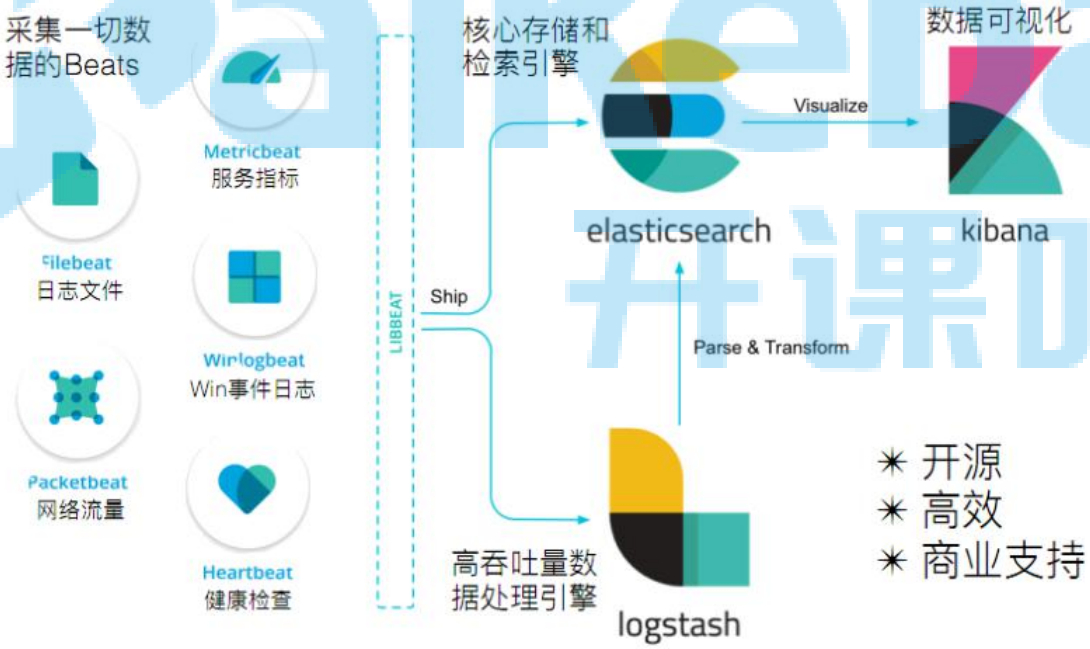
通常，日志被分散在储存不同的设备上。如果你管理数十上百台服务器，你还在使用依次登录每台机器的传统方法查阅日志。这样是不是感觉很繁琐和效率低下。当务之急我们使用集中化的日志管理，例如：开源的 syslog，将所有服务器上的日志收集汇总。
集中化管理日志后，日志的统计和检索又成为一件比较麻烦的事情，一般我们使用 grep、awk 和 wc 等 Linux 命令能实现检索和统计，但是对于要求更高的查询、排序和统计等要求和庞大的机器数量依然使用这样的方法难免有点力不从心。

命令方式:
```
tail -n 300 myes.log | grep 'node-1'　　##搜索某个日志在哪里
tail -100f myes.log
```

2 Elastic Stack 认识

如果你没有听说过 Elastic Stack，那你一定听说过 ELK，实际上 ELK 是三款软件的简称，分别是 Elasticsearch、Logstash、Kibana 组成，在发展的过程中，又有新成员 Beats 的加入，所以就形成了 Elastic Stack。所以说，ELK 是旧的称呼，Elastic Stack 是新的名字。

全系的 Elastic Stack 技术栈包括：



1）Elasticsearch

Elasticsearch 基于 java，是个开源分布式搜索引擎，它的特点有：分布式，零配置，自动发现，索引自动分片，索引副本机制，restful 风格接口，多数据源，自动搜索负载等。

2）Logstash

Logstash 基于 java，是一个开源的用于收集,分析和存储日志的工具。

3）Kibana

Kibana 基于 nodejs，也是一个开源和免费的工具，Kibana 可以为 Logstash 和 ElasticSearch 提供的日志分析友好的 Web 界面，可以汇总、分析和搜索重要数据日志。

4）Beats

Beats 是 elastic 公司开源的一款采集系统监控数据的代理 agent，是在被监控服务器上以客户端形式运行的数据收集器的统称，可以直接把数据发送给 Elasticsearch 或者通过 Logstash 发送给 Elasticsearch，然后进行后续的数据分析活动。

Beats 由如下组成:

Packetbeat: 是一个网络数据包分析器，用于监控、收集网络流量信息，Packetbeat 嗅探服务器之间的流量，解析应用层协议，并关联到消息的处理，其支 持 ICMP (v4 and v6)、DNS、HTTP、Mysql、PostgreSQL、Redis、MongoDB、Memcache 等协议；

Filebeat: 用于监控、收集服务器日志文件，其已取代 logstash forwarder；

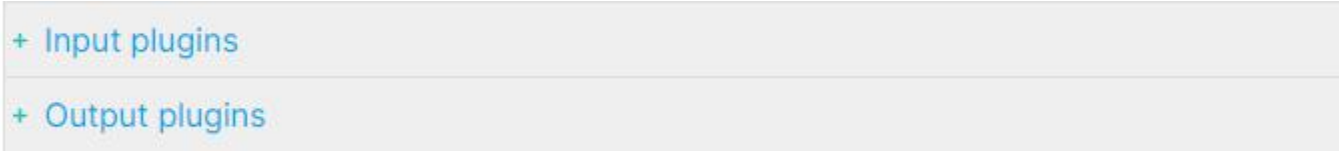
Metricbeat: 可定期获取外部系统的监控指标信息，其可以监控、收集 Apache、HAProxy、MongoDB、MySQL、Nginx、PostgreSQL、Redis、System、Zookeeper 等服务；

Winlogbeat: 用于监控、收集 Windows 系统的日志信息；

3 Logstash

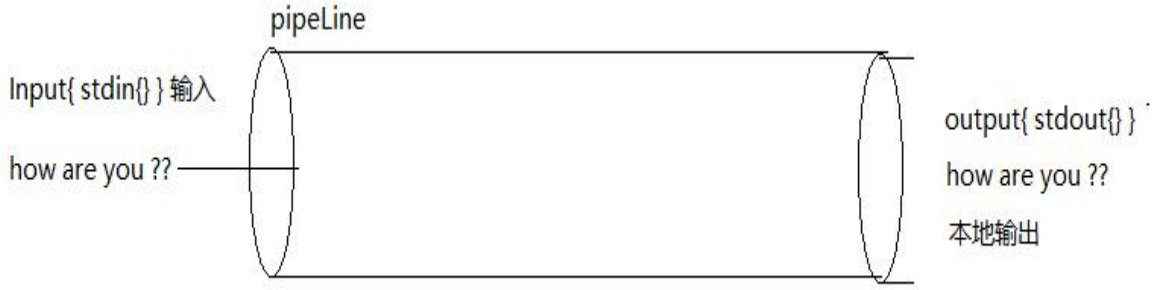
3.1 Logstash 指令

（1）、./logstash -e 'input{ stdin{} } output{ stdout{} }'

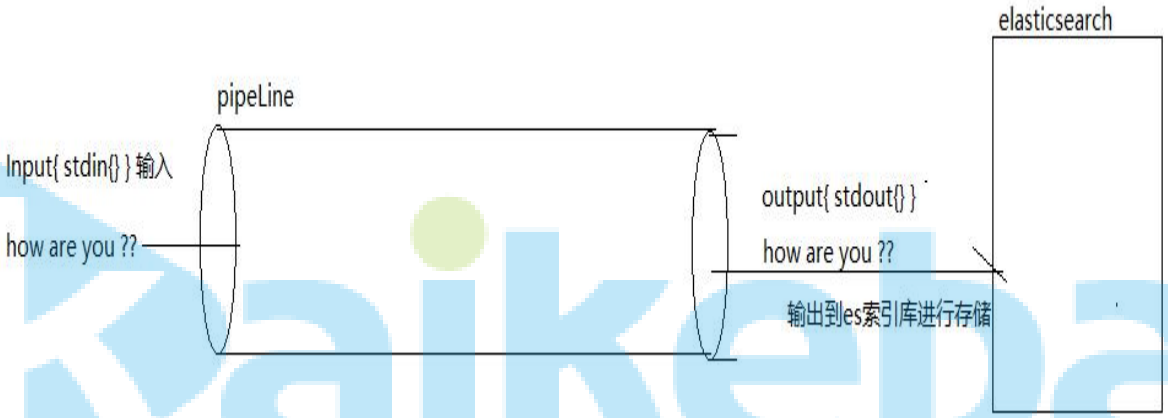


（2）、./logstash -e 'input{ stdin{} } output{ stdout{ codec => json }}

1）-e 设置启动参数
2）input ,output --- logstash 插件
3）stdin{} 等待输入插件
4）stdout{} 输出日志插件
5）codec 编码插件



思考：加入输入的是日志，把日志存储在 elasticsearch 索引库，是否应该指定输出地址为 es 地址？



（3）./logstash -e 'input{ stdin{} } output{ elasticsearch{ hosts => ["192.168.66.66:9200"] } stdout{}}'

1）output 中插件：elasticsearch 插件，表示把数据输出到 es 中进行存储。

输出地址：

hosts

edit

- Value type is uri
- Default value is `[//127.0.0.1]`

Sets the host(s) of the remote instance. If given an array it will load balance requests across the hosts specified in the `hosts` parameter. Remember the `http` protocol uses the http address (eg. 9200, not 9300). `"127.0.0.1"` `["127.0.0.1:9200","127.0.0.2:9200"]` `["http://127.0.0.1"]` `["https://127.0.0.1:9200"]` `["https://127.0.0.1:9200/mypath"]` (If using a proxy on a subpath) It is important to exclude dedicated master nodes from the `hosts` list to prevent LS from sending bulk requests to the master nodes. So this parameter should only reference either data or client nodes in Elasticsearch.

Any special characters present in the URLs here MUST be URL escaped! This means `#` should be put in as `%23` for instance.

索引库名称：

index

- Value type is string
- Default value is `"logstash-%{+YYYY.MM.dd}"`

The index to write events to. This can be dynamic using the `%{foo}` syntax. The default value will partition your indices by day so you can more easily delete old data or only search specific date ranges. Indexes may not contain uppercase characters. For weekly indexes ISO 8601 format is recommended, eg. logstash-%{+xxxx.ww}. LS uses Joda to format the index pattern from event timestamp. Joda formats are defined here.

3.2 Logstash 配置

思考：刚才启动 logstash 时候，通过-e 把相关插件直接追加到启动命名后面，是否有一种方法，把插件语法放入
配置文件？？

```
[root@jackhu logconf]# vi log-simple.conf
   stdout{}
input { stdin{} }
output {
   elasticsearch { hosts => ["192.168.66.66:9200"] }
   stdout {codec => rubydebug }
}
```

启动 logstash:

bin/logstash -f logconf/log-simple.conf

思考问题：

在线上环境中，日志是使用文件方式进行保存，logstash 是否能够直接从日志文件中读取数据呢？？

3.3 File 日志收集

```
[root@jackhu logconf]# vi log-file.conf
input {
   file {
       path => "/root/supergo.log"
       type => "java"
       start_position => "beginning"
        }
 }
output {
   elasticsearch {
        hosts => ["192.168.66.66:9200"]
        index => "supergo-%{+YYYY.MM.dd}"
        }
   stdout {codec => rubydebug }
}
```

File: 插件，从文件中读取日志数据

1）path 指定日志文件的位置路径

2）type 类型，相当于定义名称

3）start_position 读取日志文件从头开始，还是从尾开始，如果日志文件未读完退出，下次会从上次读取的位置
继续读取。

思考：如何读取多个文件日志，输入多个索引库呢？？

```
input {
   file {
       path => "/root/supergo.log"
       type => "java"
       start_position => "beginning"
        }

   file {
       path => "/var/log/boot.log"
       type => "boot"
       start_position => "beginning"
        }
 }
output {
   if [type] == "java" {
   elasticsearch {
        hosts => ["192.168.66.66:9200"]
        index => "supergo-%{+YYYY.MM.dd}"
        }
     }
   if [type] == "boot" {
   elasticsearch {
        hosts => ["192.168.66.66:9200"]
        index => "boot-%{+YYYY.MM.dd}"
        }
     }


   stdout {codec => rubydebug }
```

4 Kibana

5 日志分析

5.1 Nginx 日志收集

动态统计网站流量

1） 安装 nginx

```
./configure \
--prefix=/usr/local/src/nginx \
--pid-path=/var/run/nginx/nginx.pid \
--lock-path=/var/lock/nginx.lock \
--error-log-path=/var/log/nginx/error.log \
--http-log-path=/var/log/nginx/access.log \
--http-client-body-temp-path=/var/temp/nginx/client \
--http-proxy-temp-path=/var/temp/nginx/proxy \
--http-fastcgi-temp-path=/var/temp/nginx/fastcgi \
--http-uwsgi-temp-path=/var/temp/nginx/uwsgi \
--http-scgi-temp-path=/var/temp/nginx/scgi \
```

--with-http_gzip_static_module
make && make install

2） 日志格式
为了方便处理日志： 把 nginx 输入的日志格式约定为为 json 格式，且约定好字段
log_format log_json
'{ "@timestamp": "$time_local", '
'"remote_addr": "$remote_addr", '
'"referer": "$http_referer", '
'"request": "$request", '
'"status": $status, '
'"bytes": $body_bytes_sent, '
'"agent": "$http_user_agent", '
'"x_forwarded": "$http_x_forwarded_for", '
'"up_addr": "$upstream_addr",'
'"up_host": "$upstream_http_host",'
'"up_resp_time": "$upstream_response_time",'
'"request_time": "$request_time"'
' }';

使用 json 格式日志形式：
server {
listen 80;
server_name localhost;

#charset koi8-r;

access_log logs/access.log log_json;

location / {
root html;
index index.html index.htm;
#proxy_pass http://ticket;
}

5.2 项目日志收集

1）坐标
```
<dependency>
    <groupId>net.logstash.logback</groupId>
    <artifactId>logstash-logback-encoder</artifactId>
    <version>5.2</version>
</dependency>
```

2）配置文件

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
    <include
resource="org/springframework/boot/logging/logback/base.xml" />

    <appender name="LOGSTASH"
class="net.logstash.logback.appender.LogstashTcpSocketAppender
">
        <destination>192.168.66.66:9601</destination>
        <encoder charset="UTF-8"
class="net.logstash.logback.encoder.LogstashEncoder" />
    </appender>

    <root level="INFO">
        <appender-ref ref="LOGSTASH" />
      <!--  <appender-ref ref="CONSOLE" />-->
    </root>
</configuration>
```

3）logstash 安装插件

```
./logstash-plugin install logstash-codec-json_lines
```

3）加载配置文件

```
# 加载日志文件
logging:
  config: classpath:log/logback-spring.xml
```

4）logstash 配置文件

```
input {
    tcp {
            port => 9601
            codec => json_lines
            }
 }
output {
    elasticsearch {
            hosts => ["192.168.66.66:9200"]
            index => "kkb-log-%{+YYYY.MM.dd}"
            }
    stdout {codec => rubydebug }
```

```
}
```


kaikeba
开课吧