

Docker 的日志收集

张军

2016 年 3 月 24日



HanSight 瀚思

Who am I

- 张军 @ HanSight瀚思，资深研发工程师
- Hortonworks Hadoop 1.0及2.0认证国内第一人，现担任 Hortonworks Hadoop 认证讲师
- 参与了国内众多的大数据项目，是大数据项目落地的最早实践者之一
- 目前主要从事基于大数据技术在安全领域的应用
- 拥有多年的项目开发及运维经验

企业背景



- 成立于 2014 年的创业公司
- **数据驱动安全** – 以大数据分析的方式解决安全问题
- 创始人来自趋势科技、天云、微软、Oracle
- 两大类产品线：**企业版**和**SaaS**版
- 企业版销售对象是各种大型企业总部、事业单位等
- 基于 Elasticsearch 1.5
- 重点在性能、安全、算法
- SaaS 版 – **安全易** – 面向中小企业
- 云端的多租户 SIEM
- 基于 Elasticsearch 1.7 + Kibana 4.1 改编
- 重点在可视化



瀚思专注于大数据安全分析

- 已拥有招商银行、公安部、天津公安、北京联通、河北金融学院、北京燃气集团等十余家客户，涵盖公安、金融、电信、制造、政府等领域。
- 积极拓展大数据安全行业与标准联盟，包括加入《CNCert中国互联网网络安全威胁治理联盟》，与公安部三所共同制定大数据安全体系标准，参与制定数据治理国家标准，与华为达成大数据生态战略合作等。
- 瀚思于2015年获得美国硅谷Red Herring全球创新公司百强、亚洲创新公司百强，为上榜唯一中国安全企业。2015中国信息产业年度人物获得者。



Docker 的日志处理

1. 为什么要采集日志
 - 概述日志收集的目的
 2. 传统linux日志收集、处理
 - 收集方式
 - 处理方式
 3. docker日志采集方法的演变
 - docker v1.6之前
 - docker v1.6之后
 - 演示
 4. 日志简单分析
- Q&A



Log, Why?

- 为了跟踪、定位、排除故障，需要服务程序故障位置的上下文。
- 监控服务，定时更新服务状态，确保服务正常工作。
- 性能优化，甚至实时获取机器环境，如CPU使用率，内存消耗量和网络性能。
- 机器学习与预测

为什么SaaS选择Docker

构建SaaS服务的方法之一 <http://12factor.net/>

- 基准代码 一份代码，多份部署
- 依赖 显示声明依赖关系
- 配置 在环境中存储配置
- 后端服务 把后端服务当做附加资源
- 构建，发布，运行 严格分离构建和运行
- 进程 以一个或者多个无状态运行应用

日志：应用本身从不考虑存储自己的输出流。不应该试图去写或者管理日志文件。相反，每一个运行的进程都会直接的标准输出(stdout)事件流。开发环境中，开发人员可以通过这些数据流，实时在终端看到应用的活动。

- 开发环境与线上环境等价 尽可能的保持开发，预发布，线上环境相同
- 日志 把日志当做事件流
- 管理进程 后台管理任务当做一次性进程运行

传统linux怎么收集、处理日志

- 收集方式：

- 建立一个中央日志服务器，修改客户端日志配置文件，将日志备份到服务器上。（使用syslog 或者rsyslog）

[root@wwwserver /]# vi /etc/syslog.conf

添加下面的代码到syslog.conf中:*. * @logserver

- 或者安装数据库，进行日志数据库管理。

- 处理：

1. 以手动方式搜索日志文件，find、grep、awk、sed、tail、cut
2. logsurfer、swatch

Docker 怎样？

Docker上日志收集方法的演变

Docker v1.6之前:

- 存储方式：
 1. Docker仅仅是从容器中采集stdout和stderr
 2. 用JSON进行简单的封装并存储到磁盘
- 收集原理和演进：
 1. Docker的早期使用者会收集 `/var/lib/docker/containers/**`
缺点：必须用root用户才能得到
 1. 之后较好的用户体验方式：`docker logs`，直接使用获取日志的 daemon API
 2. 开源项目的出现，对接API

Docekr日志收集的开源项目

Logspout:

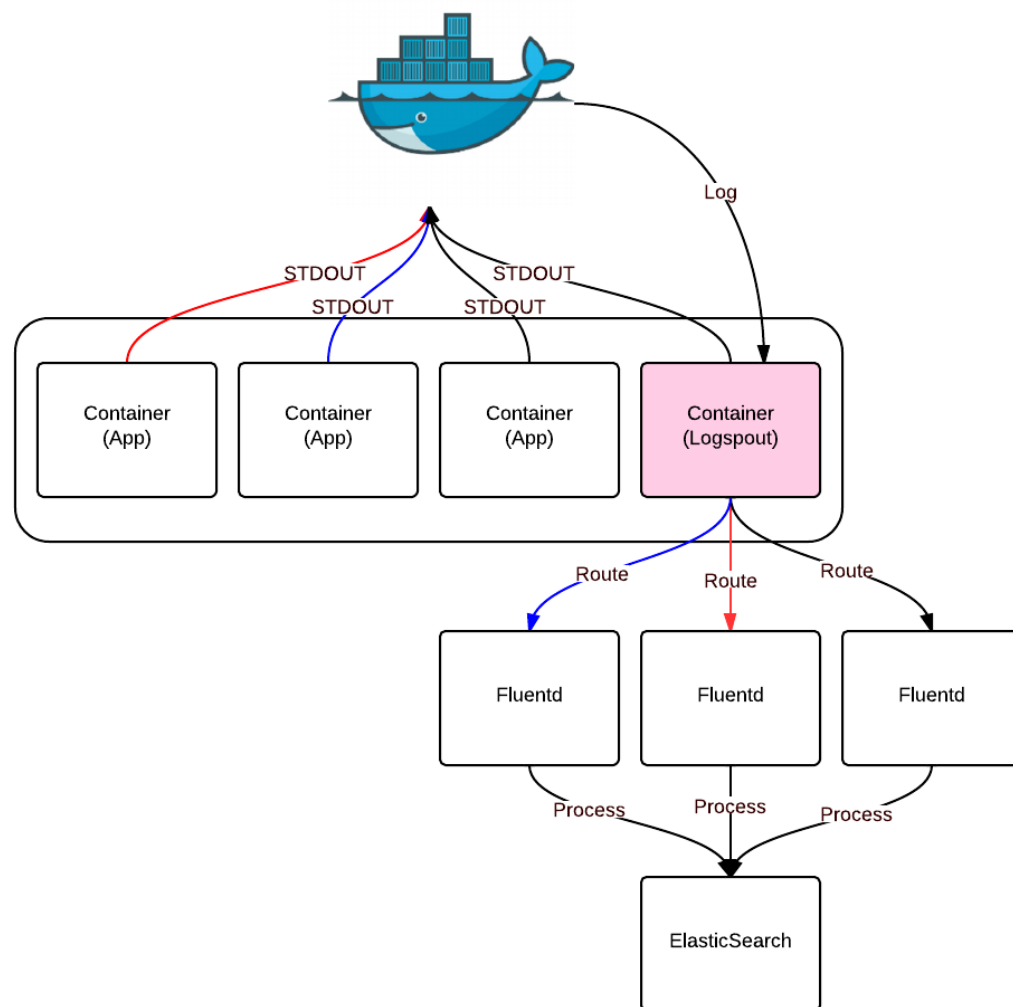
- 支持的认证和各种协议 (tls 、 tcp、 udp...)
- 拥有很多第三方的插件 (logspout-kafka、 logspout-logstash...)

example :

转发所有docker容器的日志到远程的syslog:

```
docker run --name="logspout" \  
  --volume=/var/run/docker.sock:/var/run/docker.sock \  
  gliderlabs/logspout \  
  syslog://logs.papertrailapp.com:55555
```

Docker 日志收集架构



方式总结：

- 应用程序直接将log上传入日志服务器
 - es 插件 'com.internetitem:logback-elasticsearch-appender:1.2'
- log写入一个挂载在docker上的文件
- 安装系统日志收集器
 - docker exec 注入收集器
 - Curl -XGET /containers/(id)/logs
- 直接在 /var/lib/docker/containers/**
- 缺点：
 - 所有日志都在 json-file 中
 - 文件可能不断暴涨，不能分成几个文件
 - 多个容器一起收集时，日志不能自然区分

Docker v1.6之后

- 引入日志驱动(Log Drivers), 除了默认json-file外, 还支持: 将日志写入syslog、journald、gelf、fluentd、awslogs、splunk、null, 指定方式收集日志。

`docker daemon --log-driver=journald`

`docker run --log-driver=journald ...`

`docker run --log-driver null` 就是屏蔽掉日志,不进行输出

可以用 --log-opt向 Log Driver传入参数。

- Json-file :

--log-opt max-size=[0-9+][k|m|g] 设置文件大小

--log-opt max-file=[0-9+] 文件日志保留数量

- Syslog :

--log-opt syslog-address=[tcp|udp|tcp+tls]://host:port

--log-opt syslog-address=unix://path

--log-opt syslog-tls-ca-cert=/etc/ca-certificates/custom/ca.pem

--log-opt syslog-tls-cert=/etc/ca-certificates/custom/cert.pem

--log-opt syslog-tls-key=/etc/ca-certificates/custom/key.pem

--log-opt syslog-tls-skip-verify=true

- gelf:

--log-opt gelf-address=udp://host:port

--log-opt tag="database"

Log tags的使用解决多个容器之间共享一个syslog进程，传送到一个log driver

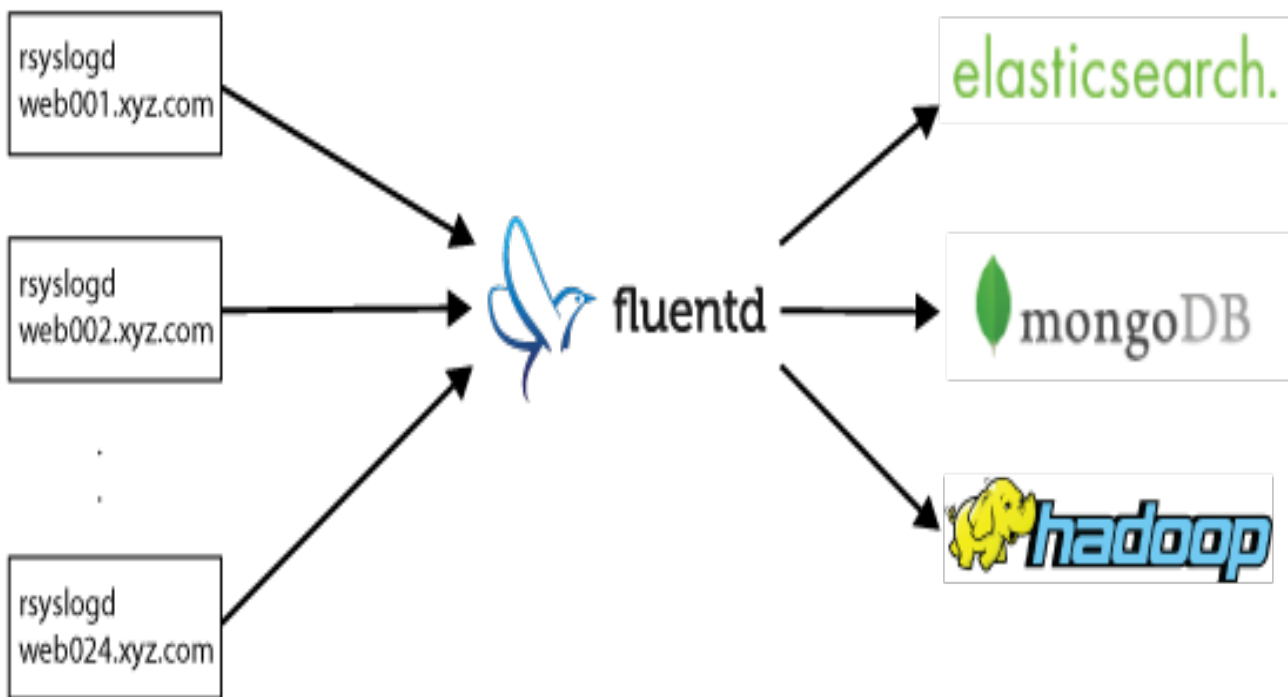
标记	描述
{{.ID}}	容器Id的前12个字符
{{.FullID}}	容器Id
{{.Name}}	容器名字
{{.ImageID}}	容器的image Id的前12个字符
{{.ImageFullID}}	容器的image ID
{{.ImageName}}	容器的image所使用的名字

--log-opt tag=" {{.ImageName}}/{{.Name}}/{{.ID}}"

Aug 7 18:33:19 HOSTNAME docker/hello-world/foobar/5790672ab6a0[9103]: Hello from Docker.

架构和集群演示

Instantly push data to any backend system



日志的处理：交易异常

威胁名称：资金转入转出异常

分析过程：

- 自定义分析的网银交易日志周期；
- 将海量用户手机号、账户号和身份证号码进行关联并可视化展示；
- 发现盗用大量身份证频繁开户的用户；进一步钻取分析，查看到可疑的用户手机号及所有相关的账户和身份证号；
- 同时发现相关账户有金融业务风险中的“火山”和“黑洞”情况出现；

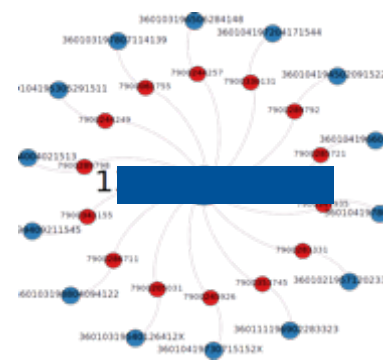
第一步：建立数据模型



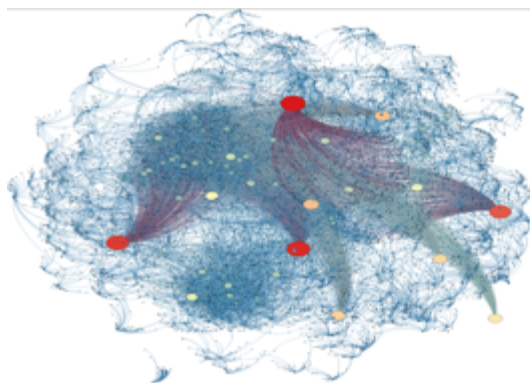
第二步：发现可疑号码



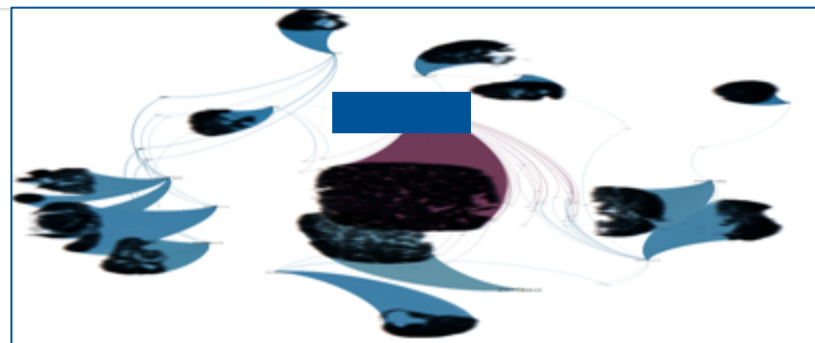
第三步：钻取关联分析



黑洞模型：账户大量转入交易



火山模型：账户大量转出交易



谢谢 |  HanSight 瀚思

www.HanSight.com

微信公众号：瀚思安信

北京市海淀区中关村软件园9号楼2区306A



日志的处理 RPCA算法

