ELK Stack企业级日志收集平台

个人介绍

讲师：李振良

资深运维工程师，曾混在IDC，大数据，金融，安全行业。下到搬服务器，上到Linux平台架构设计。经重重磨练，具备各方综合能力。

技术博客：http://blog.51cto.com/lizhenliang

关注微信公众号：DevOps大咖

专注于分享DevOps工具链及经验总结。
例如：Linux|Shell|Python|Docker|K8S|Jenkins|Git|Ansible等主流技术。
每日一篇高质量文章，助你打造一套完整且靠谱的自动化流程体系。

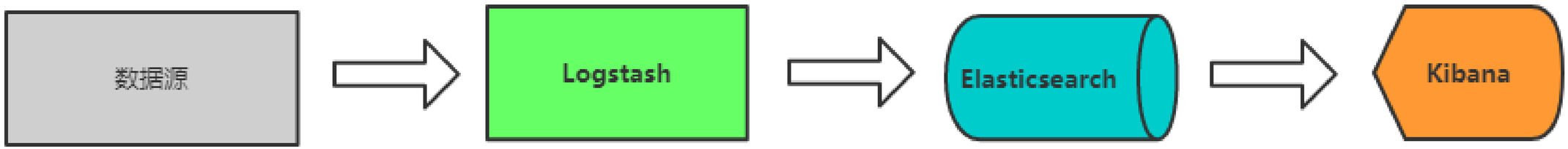
Linux运维学员群：545214087

ELK Stack企业级日志收集平台

目 录

ELK Stack企业级日志收集平台

ELK Stack介绍



Logstash ： 开源的服务器端数据处理管道，能够同时从多个来源采集数据、转换数据，然后将数据存储到数据库中。

Elasticsearch：搜索、分析和存储数据。

Kibana：数据可视化。

Beats ： 轻量型采集器的平台，从边缘机器向 Logstash 和 Elasticsearch 发送数据。

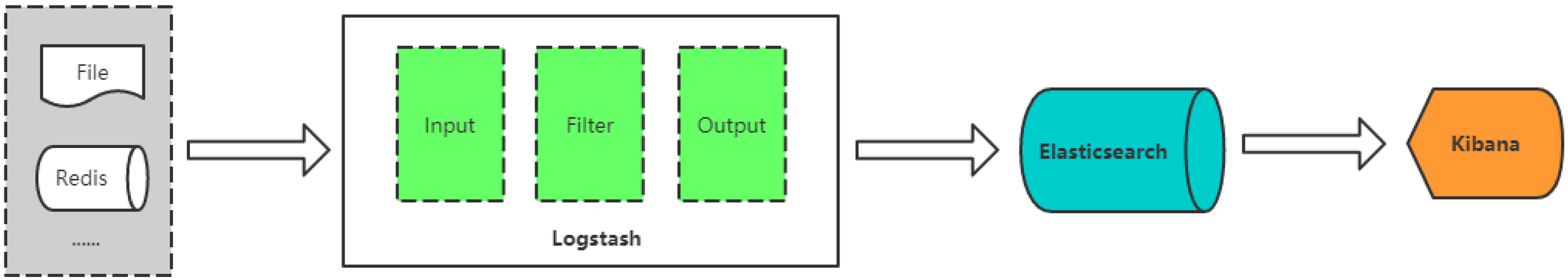
Filebeat：轻量型日志采集器。

https://www.elastic.co/cn/

https://www.elastic.co/subscriptions

ELK Stack企业级日志收集平台

ELK Stack架构



Input：输入，输出数据可以是Stdin、File、TCP、Redis、Syslog等。
Filter：过滤，将日志格式化。有丰富的过滤插件：Grok正则捕获、Date时间处理、Json编解码、Mutate数据修改等。
Output：输出，输出目标可以是Stdout、File、TCP、Redis、ES等。

https://www.elastic.co/guide/en/logstash/current/input-plugins.html

Elasticsearch

- 基本概念
- 集群部署
- 数据操作
- 常用查询
- Head插件

Elasticsearch – 基本概念

Node：运行单个ES实例的服务器

Cluster：一个或多个节点构成集群

Index：索引是多个文档的集合

Document：Index里每条记录称为Document，若干文档构建一个Index

Type：一个Index可以定义一种或多种类型，将Document逻辑分组

Field：ES存储的最小单元

Shards：ES将Index分为若干份，每一份就是一个分片

Replicas：Index的一份或多份副本

Elasticsearch	关系型数据库（比如Mysql）
Index	Database
Type	Table
Document	Row
Field	Column

Elasticsearch – 集群部署

```
# yum install java-1.8.0-openjdk -y
# rpm --import https://artifacts.elastic.co/GPG-KEY-elasticsearch
# vi /etc/yum.repos.d/elastic.repo
[elastic-6.x]
name=Elastic repository for 6.x packages
baseurl=https://artifacts.elastic.co/packages/6.x/yum
gpgcheck=1
gpgkey=https://artifacts.elastic.co/GPG-KEY-elasticsearch
enabled=1
autorefresh=1
type=rpm-md
```

```
# vi /etc/elasticsearch/elasticsearch.yml
cluster.name: elk-cluster
node.name: node-1
path.data: /var/lib/elasticsearch
path.logs: /var/log/elasticsearch
network.host: 192.168.1.196
http.port: 9200
discovery.zen.ping.unicast.hosts: ["192.168.1.195", "192.168.1.196", "192.168.1.197"]
discovery.zen.minimum_master_nodes: 2
```

Elasticsearch – 数据操作

RestFul API格式

```
curl -X<verb> '<protocol>://<host>:<port>/<path>?<query_string>' -d '<body>'
```

参数	描述
verb	HTTP方法，比如GET、POST、PUT、HEAD、DELETE
host	ES集群中的任意节点主机名
port	ES HTTP服务端口，默认9200
path	索引路径
query_string	可选的查询请求参数。例如?pretty参数将格式化输出JSON数据
-d	里面放一个GET的JSON格式请求主体
body	自己写的 JSON格式的请求主体

Elasticsearch – 数据操作

- 增
- 删
- 改
- 查

Elasticsearch – 常用查询

- match_all
- from,size
- match
- bool
- range

Elasticsearch – head插件

```
wget https://npm.taobao.org/mirrors/node/latest-v4.x/node-v4.4.7-linux-x64.tar.gz
tar -zxvf node-v4.4.7-linux-x64.tar.gz
# vi /etc/profile
NODE_HOME=/usr/local/node-v4.4
PATH=$NODE_HOME/bin:$PATH
export NODE_HOME PATH
# source /etc/profile
# git clone git://github.com/mobz/elasticsearch-head.git
cd elasticsearch-head
npm install
npm run start
```

Logstash

- 安装
- 条件判断
- 输入插件
- 编码插件
- 过滤器插件
- 输出插件

Logstash – 条件判断

比较操作符：

相等：==，!=，<，>，<=，>=

正则：=~(匹配正则)，!~(不匹配正则)

包含：in(包含)，not in(不包含)

布尔操作符：

and(与)，or(或)，nand(非与)，xor(非或)

一元运算符：

!(取反)

()(复合表达式)，!()(对复合表达式结果取反)

Logstash – 输入（Input）插件

Stdin示例：

```
input {
  stdin {
  }
}
filter {
}
output {
  stdout {
     codec => rubydebug
  }
}
```

File示例：

```
input {
  file {
     path =>"/var/log/messages"
     tags =>"123"
     type =>"syslog"
  }
}
filter {
}
output {
  stdout {
     codec => rubydebug
  }
}
```

TCP示例：

```
input {
  tcp {
     port =>12345
     type =>"nc"
  }
}
filter {
}
output {
  stdout {
     codec => rubydebug
  }
}
```

Beats示例：

```
input {
  beats {
    port => 5044
  }
}

filter {
}
output {
   stdout {
     codec => rubydebug
   }
}
```

Logstash – 编码（Codec）插件

Json/Json_lines示例：

```
input {
  stdin {
     codec =>json {
        charset => ["UTF-8"]
     }
  }
}
filter {
}
output {
   stdout {
     codec => rubydebug
   }
}
```

Multline示例：

```
input {
  stdin {
    codec => multiline {
      pattern => "^\s"
      what => "previous"
    }
  }
}
```

rubydebug

Logstash – 过滤器（Filter）插件

Json示例：

```
input {
  stdin {
  }
}

filter {
  json {
    source => "message"
    target => "content"
  }
}
output {
   stdout {
     codec => rubydebug
  }
}
```

Kv示例：

```
filter {
  kv {
     field_split => "&?"
  }
}
```

Logstash – 过滤器（Filter）插件

Grok示例：

https://github.com/logstash-plugins/logstash-patterns-core/tree/master/patterns

http://grokdebug.herokuapp.com

日志示例：223.72.85.86 GET /index.html 15824 0.043

```
filter {
  grok {
    match => {
       "message" => "%{IP:client} %{WORD:method} %{URIPATHPARAM:request} %{NUMBER:bytes} %{NUMBER:duration}"
    }
  }
}
```

自定义模式：

```
# vi /opt/patterns
ID [0-9A-Z]{10,11}

filter {
  grok {
    patterns_dir =>"/opt/patterns"
    match => {
      "message" => "%{IP:client} %{WORD:method} %{URIPATHPARAM:request} %{NUMBER:bytes} %{NUMBER:duration} %{ID:id}"
    }
  }
}
```

Logstash – 过滤器（Filter）插件

多模式匹配：

```
filter {
  grok {
    patterns_dir =>"/opt/patterns"
    match => [
      "message", "%{IP:client} %{WORD:method} %{URIPATHPARAM:request} %{NUMBER:bytes} %{NUMBER:duration} %{ID:id}",
      "message", "%{IP:client} %{WORD:method} %{URIPATHPARAM:request} %{NUMBER:bytes} %{NUMBER:duration} %{TAG:tag}"
    ]
  }
}
```

Logstash – 过滤器（Filter）插件

Geoip示例：223.72.85.86 GET /index.html 15824 0.043

```
filter {
  grok {
    match => {
      "message" => "%{IP:client} %{WORD:method} %{URIPATHPARAM:request} %{NUMBER:bytes} %{NUMBER:duration}"
    }
  }
  geoip {
      source => "client"
      database => "/opt/GeoLite2-City.mmdb"
  }
}
```

Date示例：

```
filter {
  date {
      match => [ "logdate", "MMM dd HH:mm:ss"]
  }
}
```

Logstash – 输出（Output）插件

ES示例：

```
output {
  elasticsearch {
     hosts => "localhost:9200"
     index => "ytjh-admin-%{+YYYY.MM.dd}"
  }
}
```

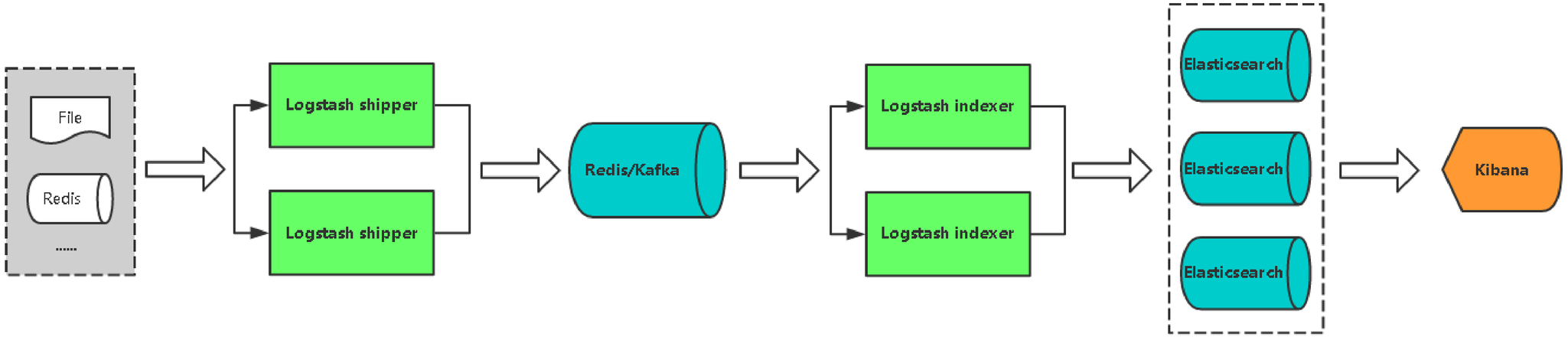
ELK Stack企业级日志收集平台

Kibana

安装、配置、图形介绍

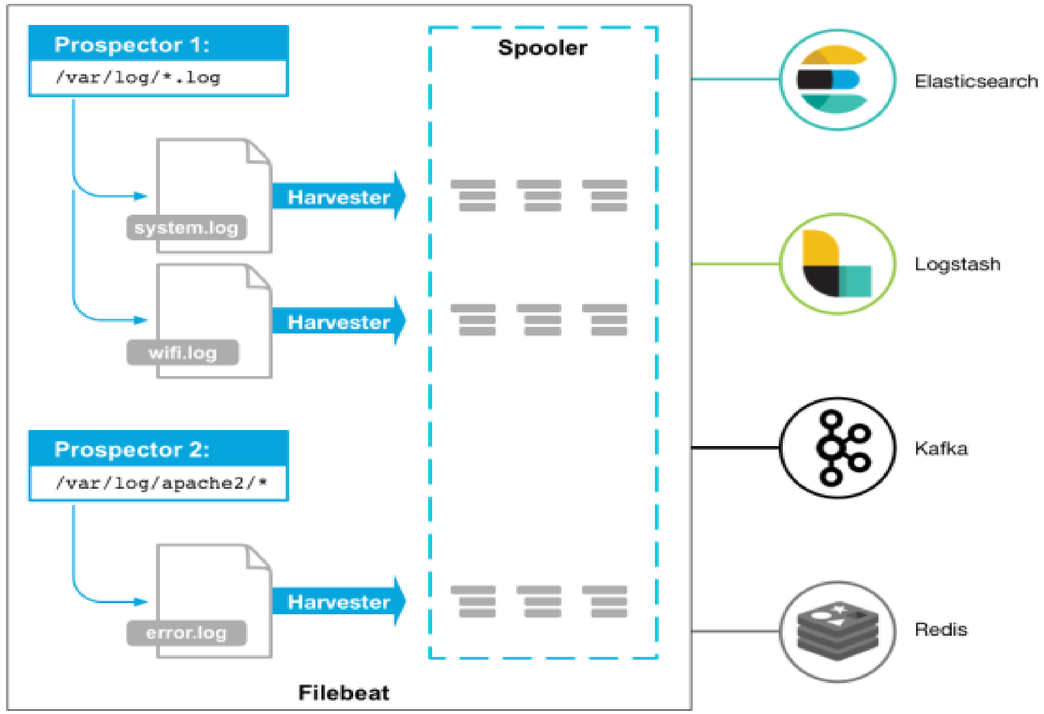
ELK Stack企业级日志收集平台

引入Redis



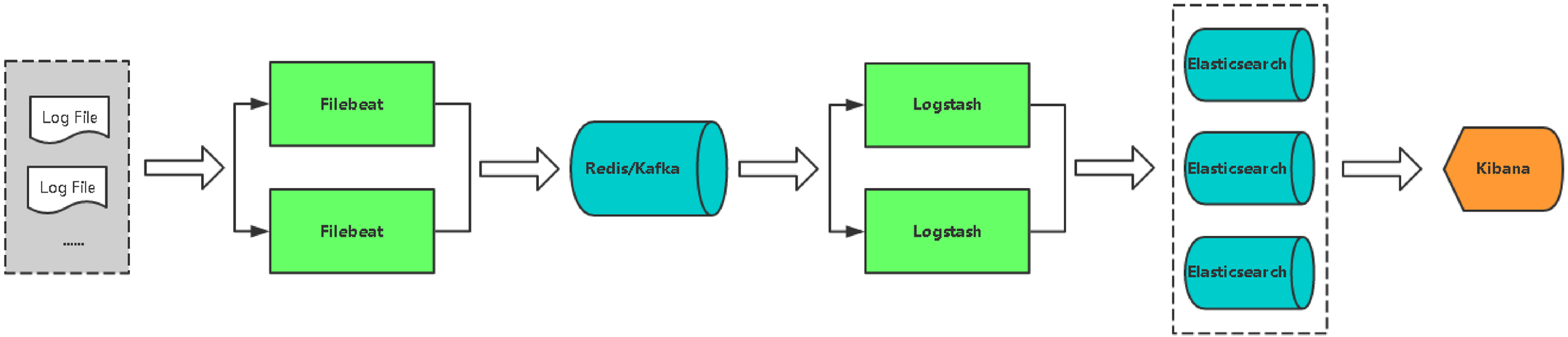
ELK Stack企业级日志收集平台

引入Filebeat



ELK Stack企业级日志收集平台

引入Filebeat



应用案例

- 收集Nginx访问日志
- 收集Java堆栈日志
- 定制日志格式收集
- Kibana仪表盘与可视化
  - PV/UV
  - 用户地理位置分布
  - URL、HTTP Status、IP TOP10

谢谢