**IngestNode配置文档**

V1.0.0

王智强 @ Enlink 数据中心

**目录**

[1. 概述 3](#_Toc16424)

[1.1. 说明 3](#_Toc24013)

[2. Filebeat日志清洗流程 4](#_Toc18994)

[2.1. 日志清洗流程 4](#_Toc8631)

[2.2. 流程配置介绍 4](#_Toc16121)

[2.2.1. filebeat.yml 4](#_Toc19958)

[2.2.2. casbs.yml 5](#_Toc13860)

[2.2.3. Module,Casbs文件夹 5](#_Toc8665)

[2.2.4. mainfest.yml 5](#_Toc1771)

[3. IngestNode配置修改 6](#_Toc24666)

[3.1. fields.yml 6](#_Toc30351)

[3.2. default.yml 7](#_Toc22293)

[3.3. pipeline.json 7](#_Toc27646)

[3.3.1. 日志数据拆分 7](#_Toc27141)

[3.3.2. 字段格式变换及修改 8](#_Toc22812)

[3.3.3. script脚本 8](#_Toc12801)

[3.3.4. 报错处理 9](#_Toc20421)

[3.3.5. 其它Processor 9](#_Toc27729)

[3.4. Ingest插件 12](#_Toc10127)

[3.4.1. 插件使用 12](#_Toc11230)

[3.4.2. ingest-geoip 13](#_Toc712)

[3.4.3. ingest-user-agent 13](#_Toc184)

[4. Filebeat.yml详细配置 14](#_Toc13389)

[4.1. Filebeat input 14](#_Toc12562)

[4.2. Filebeat modules 14](#_Toc21800)

[4.3. Elasticsearch output 15](#_Toc14055)

# 概述

本文档描述《IngestNode配置文档》的使用，请按照此文档的描述配置。

## 说明

Ingest Node(预处理节点)是ES用于功能上命名的一种节点类型,可以通过在elasticsearch.xml进行配置node.ingest: false/true来标识出集群中的某个节点是否是Ingest Node。若上述将node.ingest设置成false,则表明当前节点不是Ingest Node,不具有预处理能力，当然Elasticsearch默认所有节点都是Ingest Node,即集群中所有的节点都具有预处理能力。

# Filebeat日志清洗流程

## 日志清洗流程

filebeat.yml

casbs.yml

Module

Casbs

Admin,System,User,Haproxy,Res

pipeline.json

default.yml

mainfest.yml

图 2-1 日志清洗流程

如图2-1流程图所示，filebeat.yml中通过配置找到casbs.yml文件，casbs.yml文件里设置了需要用到的module以及该module下的调用的子模块有哪些，子模块里的mainfest.yml配置文件则指向default.yml检索文件和pipeline.json日志清洗文件。

## 流程配置介绍

### filebeat.yml

Filebeat.yml（目录位置/etc/filebeat）中Filebeat modules模块配置了casbs.yml的相关信息，具体配置如表2-1所示：

表 2-1 filebeat.yml配置信息

|  |
| --- |
| filebeat.config.modules:  # 定义要检查的配置路径  path: ${path.config}/modules.d/\*.yml  # 设置为true时，启用动态配置重新加载  reload.enabled: false  # 定义要检查的时间间隔  reload.period: 10s |

### casbs.yml

Casbs.yml文件（目录位置/etc/filebeat/modules.d）描述module模块下子模块的启动配置信息，本项目只需要casbs模块故将其他模块的配置文件删除，具体配置如表2-2所示：

表 2-2 casbs.yml配置信息

|  |
| --- |
| - module: casbs  # 资源日志  res:  # 是否启用  enabled: true  # 管理员操作日志  admin:  enabled: true  # 系统日志  system:  enabled: true  # 系统日志  user:  enabled: true |

module:casbs中casbs为该module模块名，res,dns,admin等为该模块下的子模块名，enabled为该模块是否启用的属性，true为启用,false为关闭。

### Module,Casbs文件夹

module文件夹（目录位置/usr/share/filebeat/module）下为各个module模块，casbs.yml描述哪些模块则回去启动哪些模块，该项目启动的模块只有casbs，casbs下的子模版admin,haproxy,res等如下图2-2所示：



图 2-2 casbs文件夹

### mainfest.yml

Mainfest.yml配置文件在admin,haproxy,res等子模块文件夹里，该文件具体配置信息如表

表 2-3 mainfest.yml配置信息

|  |
| --- |
| module\_version: 1.0  var:  - name: paths  default:  - /var/log/administrator\_operation.log  - /usr/local/enlink/casbs/enwas/admin/administrator\_operation.\*.log  os.darwin:  - /var/log/administrator\_operation.log  - /usr/local/enlink/casbs/enwas/admin/administrator\_operation.\*.log  os.windows:  - "D://tmp/es/data/administrator\_operation.log"  ingest\_pipeline: ingest/pipeline.json  input: config/default.yml  requires.processors:  - name: geoip  plugin: ingest-geoip |

var下为的default，os.darwin，os.windows分别为Lunix，Mac,Windows不同操作系统下对日志扫描目录下的文件，可以为多条路径下的文件。Ingest\_pipeline和input分别定义pipeline.json和default.yml所在位置。Requires.processors下的name，plugin分别定义了自定义插件名(方便后续使用，原名太长)，插件原名。

Pipeline.json，defalut.yml的详细介绍请看下章节。

# IngestNode配置修改

本章节描述配置Ingest Node需要改动的文件（fields.yml，pipeline.json，default.yml）以及所在目录。

## fields.yml

Fields.yml(目录位置/etc/filebeat)定义IngestNode清洗日志对每条日志分割所定义的字段名以及属性。

如表3-1所示，key：casbs中casbs为所定义字段所在module（/usr/share/filebeat/module）的模板名。title和description分别为该module模块的标题和描述。第一个fields下的name描述该模块的名称，type和description分别为属性和介绍。第二个fields为casbs下的子模块，name为casbs子模块的名字。第三个fields为casbs子模块res所生成的字段，name为定义的字段名，type为该字段的属性，description为该字段的描述。若添加新的module模块或者子模块，请按上述介绍进行添加或者修改。

表 3-1 fields.yml配置信息

|  |
| --- |
| - key: casbs  title: "CASBs"  description: >  CASB资源日志模块包含的字段  fields:  - name: casbs  type: group  description: >  casbs下包含的子模块  fields:  - name: res  type: group  description: >  res字段  fields:  - name: keyword\_log\_level  type: keyword  description: >  日志级别 |

## default.yml

config目录下的default.yml为对日志读取的筛选，具体配置信息如表3-2所示：

表 3-2

|  |
| --- |
| type: log  paths:  {{ range $i, $path := .paths }}  - {{$path}}  {{ end }}  exclude\_files: [".gz$"]  exclude\_lines: ["^-", "^\\S+\\|\\|\\|\\|\\|"] |

Type为扫描文件的类型,exclude\_files，exclude\_lines分别为要排除的日志文件和要排除的日志里的莫一行，具体排除条件表3-2的正则表达式。

## pipeline.json

Ingest目录下的pipeline.json则为Ingestnode的核心部分，该文件描述对具体日志如何分割及改动，大致分为四个步骤：日志数据拆分，字段格式变换及修改，script脚本，报错处理。

### 日志数据拆分

对每条日志进行切割得到我们需要的字段，如图3-1所示：

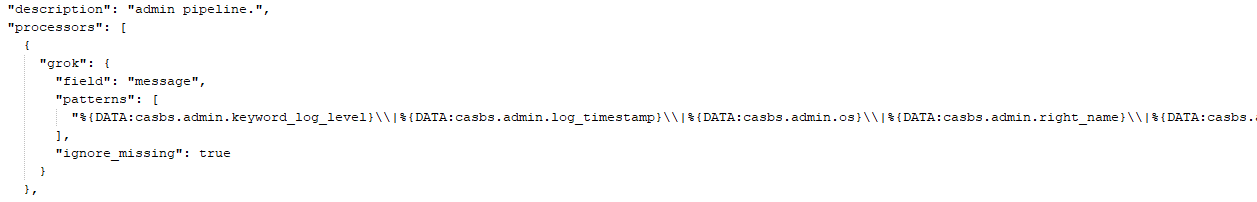


图 3-1 日志数据拆分

Description为当前pipeline的描述,grok下的field指向字段名(每条要分割的日志默认为字段名为message),patterns里则是对相应日志的具体分割,ignore\_missing为是否允许遗漏（一般设为true）。

实例：

|  |
| --- |
| INFO|2018-07-01 18:22:15|Windows|-|修改| |

对应的grok语法：

|  |
| --- |
| %{DATA:casbs.admin.keyword\_log\_level}\\|%{DATA:casbs.admin.log\_timestamp}\\|%{DATA:casbs.admin.os}\\|%{DATA:casbs.admin.right\_name}\\|%{DATA:casbs.admin.operation\_method}\\| |

分割后：

|  |
| --- |
| casbs.admin.keyword\_log\_level：INFO，  casbs.admin.log\_timestamp：2018-07-01 18:22:15，  casbs.admin.os：Windows，casbs.admin.right\_name值为：- ，casbs.admin.operation\_method值为：修改  (DATA为最基本的一种格式，不采取字段格式转变) |

### 字段格式变换及修改

将截取的字段变换成所定义的格式，如下图3-2所示：

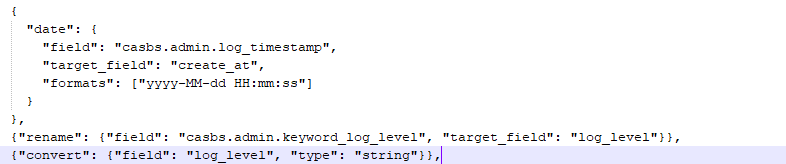


图 3-2 字段格式变换

date下将casbs.admin.log\_timestamp字段里的内容转换成formats所展示的格式并改名为target\_field所指的字段名create\_at，rename下将field的字段名casbs.admin.keyword\_log\_level改成target\_field的字段名log\_level，convert下将field字段名log\_level的属性转换成type的属性string。

### script脚本

可以灵活的对字段进行逻辑判断以及相应操作，lang:painless为固定属性，ctx表示当前总字段，ctx.xxx为当前总字段下的xxx字段具体实例如表3-3,表3-4所示：

表 3-3 示例代码

|  |
| --- |
| {  "script": {  "lang": "painless",  "inline": "ctx.search\_operation\_method = ctx.operation\_method"  }  } |

如上表所示，该script脚本为将ctx.operation\_method字段的值复制给ctx.search\_operation\_method。

表 3-4 示例代码

|  |
| --- |
| {  "script": {  "lang": "painless",  "inline": "ctx.viewCount = (ctx.viewCount) \*10"  }  } |

如上表所示，该script脚本为将ctx.viewCount的字段值乘以10倍。

### 报错处理

将ingestnode日志解析所产是的错误信息统一处理，如表3-5所示：

表 3-5 示例代码

|  |
| --- |
| "on\_failure": [  {  "set": {  "field": "error.message",  "value": "{{ \_ingest.on\_failure\_message }}"  }  }  ] |

如上表所示将所产生的错误信息赋值给error.message。

### 其它Processor

追加处理函数（Append Processor），具体实例如表3-6：

表 3-6 示例代码

|  |
| --- |
| {  "append": {  "field": "field1",  "value": ["item2", "item3", "item4"]  }  } |

如上图所示，字段名为field1的值后添加item2,item3,item4，若没有field1该字段则会新增field1字段并且值为item2,item3,item4。

循环处理函数（Foreach Processor）用来处理一些数组字段,数组内的每个元素都会使用到一个相同的处理函数如表3-7所示：

表 3-7 示例代码

|  |
| --- |
| {  "foreach": {  "field": "values",  "processor": {  "uppercase": {  "field": "\_ingest.\_value"  }  }  }  } |

上面的例子中,文档中的values字段是一个数组类型的字段,其中每个元素都需要使用Foreach Processor中的一个共同的uppercase Processor来保证该字段中的每个元素都能执行相同的操作结果。

替换处理函数（Gsub Processor）能够解决一些字符串中才特有的问题，比如我想把字符串格式的日期格式如"yyyy-MM-dd HH:mm:ss"转换成"yyyy/MM/dd HH:mm:ss"的格式，我们可以借助于Gsub Processor来解决，而Gsub Processor也正是利用正则来完成这一任务的，如表3-8所示：

表 3-8 示例代码

|  |
| --- |
| {  "gsub": {  "field": "message",  "pattern": "-",  "replacement": "/"  }  } |

Gsub Processor处理前数据message为： "2017-11-27 00:00:00"，处理后数据message为： "2017/11/27 00:00:00"。

插入处理函数（Join Processor）能够将原本一个数组类型的字段值，分解成以指定分隔符分割的一个字符串，如表3-9所示：

表 3-9 示例代码

|  |
| --- |
| {  "join": {  "field": "message",  "separator": "、、、、"  }  } |

Join Processor处理前数据message为：["hello","world","felayman"]，处理后数据message为： "hello、、、、world、、、、felayman"。

小写处理函数（Lowercase Processor）也是一个专用于字符串类型的字段处理器，顾名思义，是将字符串都转换成小写格式，如表3-10所示：

表 3-10 示例代码

|  |
| --- |
| {  "lowercase": {  "field": "message"  }  } |

Lowercase Processor处理前数据message为："HELLO,WORLD,FELAYMAN."，处理后数据message为： "hello,world,felayman."。

设置处理函数（Set Processor）作用于两种不同情况：1.指定字段存在时，修改指定字段的值。2.指定字段不存在时，新增该字段并设置该字段的值。如表3-11所示：

表 3-11 示例代码

|  |
| --- |
| {  "set": {  "field": "category",  "value": "家居"  }  } |

Set Processor处理前数据category为："HELLO,WORLD,FELAYMAN."，处理后数据category为： "家居"。

分割处理函数（Split Processor）用于将一个以指定分隔分开的字符串转换成一个数组类型的字段，如表3-12所示：

表 3-12 示例代码

|  |
| --- |
| {  "split": {  "field": "message",  "separator": "-"  }  } |

Split Processor处理前数据category为："hellow-world-felayman."，处理后数据category为： ["hello","world","felayman"]。

若有其他没有描述的processor需求或者说明不够详细，请参考官方api(https://www.elastic.co/guide/en/elasticsearch/reference/current/ingest-processors.html)

## Ingest插件

本项目日志清洗用到了两类插件：ingest-geoip-6.3.0，ingest-user-agent-6.3.0。下面介绍插件的用法以及这两类插件的作用。

### 插件使用

前面介绍manifest.yml文件时提过插件的配置，具体代码如表3-13所示：

表 3-13 示例代码

|  |
| --- |
| requires.processors:  - name: user\_agent //自定义插件别名  plugin: ingest-user-agent //插件名  - name: geoip  plugin: ingest-geoip |

插件的具体使用则在pipeline.json内，具体代码如表3-14：

|  |
| --- |
| {  "geoip": { //manifest.yml定义的插件别名  "field": "remote\_ip\_tmp", //要解析参数  "target\_field": "geoip", //自定义插件别名  "ignore\_failure": true //不允许夹带错误信息  }  } |

则geoip处理器会把参数remote\_ip\_tmp带入插件，插件处理后会自动生成相应的结果。

### ingest-geoip

该插件的作用为根据用户ip得出用户所在洲，国家，省，城市以及所在位置的经纬度。生成的字段如表3-15所示：

表 3-14 生成字段

| **字段名** | **字段属性** | **字段描述** |
| --- | --- | --- |
| **geoip** | group | 包含用户登录系统时的具体位置 |
| **geoip.continent\_name** | text | 用户所在洲 |
| **geoip.city\_name** | text | 用户所在城市 |
| **geoip.country\_iso\_code** | text | 用户所在国家 |
| **geoip.region\_name** | text | 用户所在省 |
| **geoip.location** | group | 用户使用该系统所在的经纬度 |
| **geoip.location.lon** | long | 用户所在经度 |
| **geoip.location.lat** | long | 用户所在纬度 |

geoip为定义的插件别名。

### ingest-user-agent

该插件的作用为根据用户代理信息得到用户所使用的浏览器类型及版本、操作系统及版本、浏览器内核、等信息的标识。生成的字段如表3-16所示：

表 3-15 生成字段

| **字段名** | **字段属性** | **字段描述** |
| --- | --- | --- |
| **user\_agent** | group | 用户代理 |
| **user\_agent.patch** | text | 用户当前浏览器补丁版本号 |
| **user\_agent.major** | long | 用户当前浏览器的主版本号 |
| **user\_agent.minor** | long | 用户当前浏览器的次版本号 |
| **user\_agent.os** | text | 用户当前操作系统 |
| **user\_agent.name** | text | 用户当前所用浏览器 |
| **user\_agent.os\_name** | text | 用户当前操作系统名 |
| **user\_agent.device** | text | 用户所用设备 |

# Filebeat.yml详细配置

## Filebeat input

Filebeat input部分为Filebeat读取日志的配置，表4-1详细介绍该部分的配置：

表 4-1 Filebeat input主要配置

|  |
| --- |
| type: log #定义扫描日志的类型，也可以为：Stdin,Redis,UDP,Docker,TCP,Syslog  enabled: false #控制prospector（探点）的启动和关闭  paths: #日志扫描的路径，可为多条日志  - /var/log/\*.log  fields: #添加额外自定义属性（level为字段名，debug为字段属性）  level: debug  review: 1  fields\_under\_root:true #如果值为true,那么fields存储在输出文档的顶级位置  #如果与Filebeat中字段冲突，自定义字段会覆盖其他字段  exclude\_lines:[‘^DBG’] #Filebeat会丢弃这组正则表达式所匹配的行  #实例配置为Filebeat丢弃以DBG开头的行  include\_lines:[‘^ERR’,’^WARN’] #Filebeat只会导出那些匹配该正则表达式的行  #实例配置为Filebeat只到处以ERR,WARN开头的行  exclude\_files:[‘\.gz$’] #忽略匹配该正则表达式的文件    multiline.pattern: '^\[' #指定用于匹配多行的正则表达式  #实例为Filebeat将所有不以[开始的行与之前的行进行合并  multiline.negate: true #定义模式是否被否定，默认false  multiline.match: after #定义多行内容被添加到模式匹配行之后还是之前，默认无  #可以被设置为after或者before |

其它没有详细描述的配置信息请看官方[API](https://www.elastic.co/guide/en/beats/filebeat/current/configuration-filebeat-options.html)。

## Filebeat modules

Filebeat modules部分则为模块配置的信息，表4-2详细的介绍了该部分的配置信息：

表 4-2 Filebeat modules主要配置

|  |
| --- |
| path: ${path.config}/modules.d/\*.yml #定义要检查的配置路径  reload.enabled: false #设置为true时，启用动态配置重新加载  reload.period: 10s #定义要检查的间隔时间 |

## Elasticsearch output

Elasticsearch output部分为Filebeat向Elasticsearch推送数据的配置，表4-3详细的介绍了该部分的配置信息：

|  |
| --- |
| filebeat.overwrite\_pipelines: false #设置为true时每建立一个新的Elasticsearch链接时Filebeat将覆盖管道，false则不覆盖  output.elasticsearch:  hosts: ["127.0.0.1"] #要连接Elasticsearch节点的地址  #protocol: "https" #如果Elasticsearch节点是用IP:PORT的形式定义的，那么添加protocol:https。  #username: "elastic" #用于认证的用户名  #password: "changeme" #用于认证的密码  compression\_level: 1 # 设置gzip压缩级别。  worker: 2 # 每个Elasticsearch主机的工人数量。  bulk\_max\_size: 1024 # Elasticsearch bulk api批处理数据的数量。  - index: "%{[fileset.name]}-filebeat-%{+yyyy.MM.dd}"  when.equals:  fileset.name: "res" #将索引名为”res”的索引名按照格式修改替换  setup.template.name: "filebeat-%{[beat.version]}" #模板的索引名  setup.template.pattern: "\*-filebeat" #模板的模式  setup.template.overwrite: true #是否覆盖存在的模板。默认false。  setup.template.settings: #设置template属性  index.number\_of\_shards: 1 |