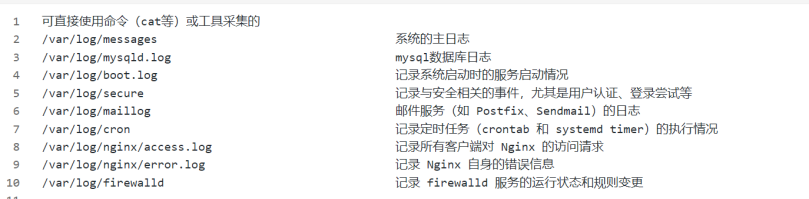
# Logstash操作

一、Logstash用来做什么? 采集：收集多种来源数据 过滤：结构化，清晰，转换数据（正则，Grok，JSON解析等） 输出：写到MySQL,Elasticsearch等

二、Logsash配置文件结构（3部分）input { ... } # 采集数据filter { ... } # 数据结构化/清洗output { ... } # 输出到目标

三、Input插件1、使用file插件直接从文件读取内容（适合于文本日志）常用于系统日志、应用日志（.log文件）以下日志用file插件采集



代码如下：



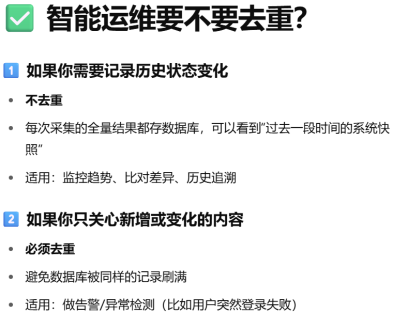
Ps：path填写日志类型对应的日志路径type给你的日志类型加标签，才能让filter插件根据不同类型进行格式化、清洗

问题：是选用一个file采集多个路径的日志还是选择多个file分开采集多个类型日志回答：

第一种方式简单、配置短、维护方便，适用于路径较少时，适用于日志结构相似文件 第二种方法对每个日志类型单独配置标签，过滤处理更加灵活，适用于日志结构化差异大时。 第三种方式是前两种的结合，结构相似的日志写一个file，其他再单独写。"/var/log/messages","/var/log/boot.log","/var/log/cron"这三个可以写在一个file里面2、使用exec插件 作用：执行系统命令，把命令输出当作日志采集输入 作用场景：采集非纯文本日志文件，二进制文件，动态命令结果（1）采集/var/log/wtmp（记录系统中所有用户的成功登录、注销以及系统启动/关机事件。） input { exec { command => "last -F" # 解析 /var/log/wtmp interval => 60 # 每60秒执行一次 type => "wtmp\_login" # 打标签 }}

（2）采集/var/log/btmp（记录所有失败的登录尝试）input { exec { command => "lastb -F" # 解析 /var/log/btmp interval => 60 # 每60秒执行一次 type => "btmp\_login\_fail" # 打标签 }}注意：这两个的解析方式是不同的，一个用命令last-F命令解析，一个用lastb-F命令解析

问题：exec每次都会全量执行命令，它不像file可以只读“新追加的日志”，每次都会重新跑一遍命令，就会有重复的内容，要不要保留重复内容？回答：



问题：如果需要去重，有什么办法？回答：方案1：fingerprint+drop：相同message内容的日志只存一条，后续重复的日志直接丢弃 filter { fingerprint { source => "message" # 对整行命令输出生成唯一哈希 target => "[@metadata][fingerprint]" method => "MD5" }

if [@metadata][fingerprint] in [some\_cache] {  
 drop { } # 如果已存在，丢弃重复记录  
} else {  
 # 将 fingerprint 保存到缓存  
}

}

方案2：aggregate 聚合统计：同fingerprint的数据统计在1分钟内出现了几次，输出1条“统计事件”，而不是重复存储原始数据filter { fingerprint { source => "message" target => "[@metadata][fingerprint]" #生成的哈希值保存到元数据字段 [@metadata][fingerprint] method => "MD5" #使用 MD5 算法生成唯一指纹（哈希值） }

aggregate { task\_id => "%{[@metadata][fingerprint]}" #用 fingerprint 的哈希值作为 task\_id #同一个 fingerprint 的日志会归为同一个“聚合组” code => "map['seen'] ||= 0; map['seen'] += 1;" #每来一条相同 fingerprint 的日志，计数 +1 push\_map\_as\_event\_on\_timeout => true #当超时时间到后（60秒），输出整个聚合组的统计数据 timeout => 60 #设置聚合组的生命周期，如果 60 秒内没有新日志加入，就认为这一组“结束”，触发输出 }}

方案3：MySQL输出时去重：Logstash照常把日志输出到MySQL，让MySQL自己判断数据是否重复，如果数据库里已经有同样的数据，就拒绝插入。



Logstash配置：output { jdbc { driver\_jar\_path => "/path/to/mysql-connector-java.jar" connection\_string => "jdbc:mysql://localhost:3306/logdb" username => "root" password => "123456" statement => [ "INSERT INTO logs (fingerprint, message, seen, timestamp) VALUES (?, ?, 1, ?) ON DUPLICATE KEY UPDATE seen = seen + 1, timestamp = VALUES(timestamp)", "[@metadata][fingerprint]", "%{message}", "%{[@timestamp]}" ] }}

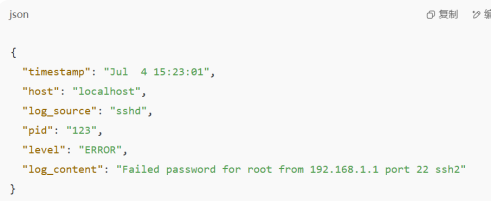
四、filter插件（分割字段）解析字段，格式化数据，做提取、去重、重命名、结构化、丢弃无用信息大致效果如下：



①grok插件（最重要，学结构化解析）grok插件基于正则表达式，从非结构化日志里提取字段，转成键值对

Grok的基本语法：%{模式名:字段名}模式名：预定义好的正则表达式字段名：给匹配到的内容起一个名字

这里写了一个例子：采集的字段有时间戳，主机名，日志来源，PID进程号，日志等级，日志正文内容filter { grok { match => { "message" => "%{SYSLOGTIMESTAMP:timestamp} %{HOSTNAME:host} %{WORD:log\_source}\[%{NUMBER:pid}\]: \[%{LOGLEVEL:level}\] %{GREEDYDATA:log\_content}" } }}Grok解析结果：

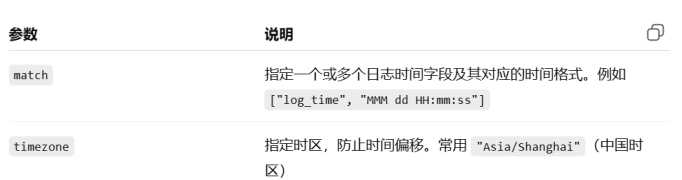


注意：日志等级level：grok里假设日志等级是放在[LEVEL]里，如果日志格式不是这样，level就可能提取失败。这里我们要采集的日志只有/var/log/mysqld.log和/var/log/nginx/error.log是有level的。如果采集的日志类型里面没有被level等级，那么就会直接解析失败，可以用 grok 的 (PATTERN)? （问号表示“可选”）或者 (?:PATTERN)? 非捕获组，让 LOGLEVEL 变成可选字段。或者对其他日志类型重新写grok。

日志来源log\_source:这里是通过程序名提取出来的（比如sshd）,如果你要的是文件来源（如/var/log/secure)，要单独用mutate添加。



②date插件在Logstash中默认给的时间戳是你采集时的系统时间，我们需要用这个插件把时间戳的时间改成日志真实产生的时间，这样出来的时间戳时间才是正确的。在智能运维日志采集中，用date插件把日志原始时间标准化，并覆盖默认的@timestamp，是必须且非常重要的步骤。我们要用到的参数：



代码如下：



"MMM d HH:mm:ss" 和 "MMM dd HH:mm:ss" 是两种时间格式的写法，写两种可以兼容两种情况



③fingerprint和aggregate这两个插件在上面研究去重的时候讲过

五、output插件①在调试的时候可以先用控制台输出output { stdout { codec =>json }}

②output输出，我们在这里将数据存入MySQL，用JDBC驱动。必须下载并放好 mysql-connector-java.jar，不然 Logstash 无法和 MySQL 通信。output { jdbc {driver\_class => "com.mysql.cj.jdbc.Driver"connection\_string => "jdbc:mysql://127.0.0.1:3306/your\_db?useSSL=false&serverTimezone=Asia/Shanghai" username => "your\_username" password => "your\_password" statement => [ "INSERT INTO logs (log\_time, hostname, log\_content, level, log\_source) VALUES (?, ?, ?, ?, ?)", "%{@timestamp}", "%{hostname}", "%{log\_content}", "%{level}", "%{log\_source}" ] }}解析：driver\_class:指定MySQL JDBC驱动的类名。MySQL8及以上版本一般是com.mysql.cj.jdbc.Driverconnection\_string:数据库连接字符串，包括数据库地址、端口、库名以及连接参数（如时区、SSL设置等）username：连接数据库的用户名password：连接数据库的密码statement：SQL语句和参数，支持参数占位符？，后面依次对应Logstash事件中的字段

这里写的JDBC插件中的INSERT INTO logs（...） VALUES（...）语句必须和MySQL数据库中logs表的结构完全匹配。比如你写了log\_time,hostname, log\_content, level, log\_source这些字段，MySQL表里也必须有对应名字且类型合适的字段。CREATE TABLE logs ( id INT AUTO\_INCREMENT PRIMARY KEY, #这个id字段其实是数据库领域的“主键”，主要为了管理和区分日志记录，而不是你日志本身的数据 log\_time DATETIME NOT NULL, hostname VARCHAR(255), log\_content TEXT, level VARCHAR(50), log\_source VARCHAR(100));