

Logstash 介紹與範例



以繁體中文詳細說明:logstash



當然可以！以下是關於 Logstash 的詳細說明（以繁體中文撰寫）：

Logstash 是什麼？

Logstash 是由 Elastic 公司開發的一個 **資料收集與處理工具**，是 Elastic Stack（舊稱 ELK Stack）中的重要組件之一（ElasticSearch、Logstash、Kibana）。

它主要用途是從各種來源收集資料，進行轉換與過濾後，再送到像是 Elasticsearch、Kafka、資料庫等地方做儲存或分析。

Logstash 的架構組成（三大核心概念）

Logstash 的整體流程可以簡化為三個部分：

1. **Input（輸入）**

指定資料的來源。例如：

- 檔案（File）
- TCP/UDP
- Kafka
- Beats（如 Filebeat）

- HTTP 等等

2. Filter (過濾/處理)

在這階段可以轉換、格式化或解析資料。可使用內建的過濾插件：

- ``grok``：解析結構化日誌（如 nginx、apache log）
- ``mutate``：修改欄位（加、減、轉換大小寫等）
- ``date``：處理時間格式
- ``json``：解析 JSON 格式內容
- ``geoip``：解析 IP 地理資訊

3. Output (輸出)

處理完的資料會被送往指定目的地。例如：

- Elasticsearch
- Kafka
- 檔案
- Stdout（標準輸出）
- Email、Websocket 等等

範例：Logstash 設定檔 (Pipeline 配置)

Logstash 的配置檔通常是 `.conf` 檔案，內容如下所示：

```
input {  
  file {  
    path => "/var/log/nginx/access.log"  
    start_position => "beginning"  
  }  
}
```

```

    }
  }

  filter {
    grok {
      match => { "message" => "%{COMBINEDAPACHELOG}" }
    }
    date {
      match => [ "timestamp" , "dd/MMM/yyyy:HH:mm:ss Z" ]
    }
  }

  output {
    elasticsearch {
      hosts => ["http://localhost:9200"]
      index => "nginx-logs-%{+YYYY.MM.dd}"
    }
  }
}

```

這段設定會：

- 從 `/var/log/nginx/access.log` 讀取資料
- 使用 `grok` 解析成結構化格式
- 把解析結果送進本地端的 Elasticsearch 服務

Logstash 插件 (Plugins)

Logstash 使用大量插件，支援擴充與客製化：

- **Input Plugins** : file、beats、tcp、stdin、http、kafka...
- **Filter Plugins** : grok、mutate、json、csv、kv、geoip...
- **Output Plugins** : elasticsearch、stdout、file、kafka、http...

Logstash 與 Beats 差異？

項目	Logstash	Beats (如 Filebeat)
功能	資料收集 + 處理 + 輸出	資料收集（輕量級）
資源消耗	中等偏高	輕量
複雜度	高，可做複雜邏輯處理	低，僅負責輸送資料
常見用途	中央式資料處理	邊緣節點資料收集

通常建議使用 Filebeat 收集資料，再送到 Logstash 處理（若需轉換、解析等進一步邏輯）。

運作方式（簡略流程圖）

```
graph TD; A[資料來源] --> B[Logstash Input]; B --> C[Filter 處理邏輯, 如 grok/json/mutate]; C --> D[Output 至 Elasticsearch、Kafka、檔案等];
```

實用指令與操作

- 啟動 Logstash 並指定設定檔：

```
bin/logstash -f your-config.conf
```

- 檢查設定檔語法是否正確：

```
bin/logstash --config.test_and_exit -f your-config.conf
```

- 使用 `stdout` 作為輸出可用於測試（方便 debug）：

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

小提醒

- 如果你的資料來源為持續新增的日誌檔（如 nginx），建議設定：

```
sourcedb_path => "/dev/null"
```

用來避免 Logstash 跳過已讀過的部分。

- 若需解析複雜格式，可自訂 `grok` pattern 或使用 Grok Debugger 工具輔助。

如果你有使用情境或具體的 Logstash 配置需求，我可以幫你量身打造範例唷！是否想知道如何整合 Filebeat、Logstash 與 Elasticsearch？或者怎麼使用 Logstash 處理 JSON、CSV、或系統日誌？





當然可以！以下是關於 **Logstash** 中的過濾規則 (**Filter**) 的詳細說明 (以繁體中文撰寫)：

過濾規則是什麼？

在 Logstash 中，「過濾規則 (Filter)」是指對輸入的資料進行轉換、清洗、解析或新增欄位的處理邏輯。這些規則會根據你定義的條件，對事件 (event) 進行加工，以便後續儲存或分析。

每一筆資料會經過你設定的 **input → filter → output** 流程，而 filter 就像中間的「資料處理工廠」。

常用的 Filter Plugin (過濾插件)

以下是最常見的幾種過濾器及其用途：

1. `grok` (用來解析文字)

用途：從原始日誌訊息中萃取出結構化的欄位。

常見於解析 Nginx、Apache、Syslog 等日誌格式。

範例：

```
filter {  
  grok {  
    match => { "message" => "%{COMMONAPACHELOG}" }  
  }  
}
```

```
}  
}
```

輸入訊息：

```
127.0.0.1 - frank [10/Oct/2024:13:55:36 -0700] "GET /apache_pb.gif HTTP/1.0"  
200 2326
```

轉換後欄位（範例）：

- ``client``：127.0.0.1
- ``user``：frank
- ``verb``：GET
- ``request``：/apache_pb.gif
- ``response``：200

你也可以自訂 pattern，例如：

```
match => { "message" => "%{IP:client} - %{WORD:user}" }
```

2. ``mutate``（修改欄位內容）

用途：修改欄位的值、刪除欄位、重新命名等。

常用操作：

- ``add_field``：新增欄位
- ``remove_field``：刪除欄位

- ``rename``：更改欄位名稱
- ``convert``：轉型（字串→整數）
- ``lowercase`` / ``uppercase``

範例：

```
filter {  
  mutate {  
    add_field => { "log_source" => "nginx" }  
    remove_field => [ "host", "path" ]  
    convert => { "bytes" => "integer" }  
  }  
}
```

3. ``date``（時間轉換）

用途：將某欄位的字串時間格式轉為 Logstash 的內建時間欄位 ``@timestamp``。

範例：

```
filter {  
  date {  
    match => ["timestamp", "dd/MMM/yyyy:HH:mm:ss Z"]  
  }  
}
```

輸入時間：

```
10/Oct/2024:13:55:36 -0700
```

轉換結果：會變成 ISO 8601 格式，並存到 ``@timestamp``

4. `geoip` (IP 轉地理資訊)

用途：根據 IP 位址查出國家、城市、經緯度等。

範例：

```
filter {
  geoip {
    source => "client_ip"
  }
}
```

結果欄位 (部份)：

- `geoip.country_name`
- `geoip.city_name`
- `geoip.latitude`
- `geoip.longitude`

5. `json` (解析 JSON 格式字串)

用途：如果某欄位是 JSON 字串，可將它解析成 Logstash 的欄位。

範例：

```
filter {
  json {
    source => "message"
  }
}
```

輸入：

```
{"user": "John", "status": "ok", "code": 200}
```

結果：

- ``user``: "John"
- ``status``: "ok"
- ``code``: 200

6. ``if`` 條件式（搭配 `filter` 使用）

可依照欄位的內容執行不同的過濾邏輯。

範例：

```
filter {  
  if [status] == "error" {  
    mutate { add_field => { "level" => "critical" } }  
  } else {  
    mutate { add_field => { "level" => "info" } }  
  }  
}
```

組合範例：Nginx Log 處理範例

```
filter {  
  grok {
```

```
match => { "message" => "%{COMBINEDAPACHELOG}" }
}

date {
  match => [ "timestamp" , "dd/MMM/yyyy:HH:mm:ss Z" ]
}

geoip {
  source => "clientip"
}

mutate {
  add_field => { "log_type" => "nginx_access" }
}

if [response] == "404" {
  mutate { add_field => { "error_tag" => "not_found" } }
}
}
```

小技巧與建議

- 測試 `grok` 時可以使用官方工具 [Grok Debugger](#)
- 建議先用 `stdout { codec => rubydebug }` 輸出觀察解析結果
- 多數 plugin 都可以加上 `tag_on_failure` 幫助偵錯

如果你有具體資料格式或過濾需求，也可以貼上來，我可以幫你撰寫客製化的過濾規則！是否想要了解如何處理多行日誌、或針對某些欄位進行條件分類呢？





當你想要將 Logstash 的輸出資料傳送給一個 **Socket Server**（例如 TCP 或 UDP Server）時，可以使用 Logstash 的 **output plugin**：``tcp`` 或 ``udp``。這些插件可以讓 Logstash 把處理後的資料，透過網路 socket 傳送給其他系統，例如自建的 socket server、資料轉接服務、中介系統等等。

1. Socket Server 是什麼？

Socket Server 就是透過網路監聽特定 port 的服務，常見的協定有：

- **TCP**：連線導向、可靠傳輸（常用）
- **UDP**：無連線、較快但不保證可靠性

你可以用任何語言（如 Python、Go、Java）寫一個 TCP 或 UDP server，接收 Logstash 傳來的資料。

2. Logstash 輸出到 TCP Socket 的範例

Logstash Output 設定（送給 TCP Server）：

```
output {  
  tcp {  
    host => "127.0.0.1"      # Socket Server 的 IP 或主機名  
    port => 5000            # Socket Server 的監聽 port  
    codec => json_lines     # 使用 JSON 格式傳送資料（每行一筆）  
  }  
}
```

```
}  
}
```

對應的 Python TCP Server 範例：

```
import socket  
  
HOST = '0.0.0.0' # 接收所有 IP  
PORT = 5000      # 與 Logstash 一致  
  
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:  
    s.bind((HOST, PORT))  
    s.listen()  
    print(f"Listening on {HOST}:{PORT} ...")  
    conn, addr = s.accept()  
    with conn:  
        print('Connected by', addr)  
        while True:  
            data = conn.recv(4096)  
            if not data:  
                break  
            print(data.decode())
```

這個簡單的 server 會持續列印從 Logstash 收到的每筆資料。

3. 支援 UDP 的 Logstash Output 範例

UDP 不需要建立連線，速度較快但資料可能會遺失。

```
output {  
  udp {  
    host => "127.0.0.1"  
    port => 5001  
    codec => json_lines  
  }  
}
```

```
}  
}
```

對應的 Python UDP Server 範例：

```
import socket  
  
UDP_IP = "0.0.0.0"  
UDP_PORT = 5001  
  
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)  
sock.bind((UDP_IP, UDP_PORT))  
  
print(f"Listening on UDP {UDP_PORT}...")  
  
while True:  
    data, addr = sock.recvfrom(4096)  
    print(f"From {addr}: {data.decode()}")
```

4. codec 的選擇（傳輸格式）

Logstash 支援多種資料格式，常見的包括：

codec 名稱	說明
`json`	輸出為 JSON 格式
`json_lines`	每筆事件為一行 JSON（常用）
`plain`	純文字格式（僅 message 欄位）
`line`	每行一筆，類似 `plain`

建議使用 ``json_lines``，較容易解析並支援多欄位。

5. 小提醒與常見問題

- 若接收端是多線程 server，要注意資料處理時機與同步問題
 - TCP Server 必須**先啟動**，否則 Logstash 會報錯連不上
 - UDP 沒有重傳機制，請用在不要求資料完整性的場景
 - 若資料很大，可以調整 Logstash 的 ``flush_size`` 與 codec 行為
 - 若接收方要依欄位使用，請使用 ``json_lines`` 格式
-

總結

將 Logstash 的輸出送給 Socket Server 是一種靈活的整合方式，可以讓你把資料即時推送到：

- 自建監控系統
 - 自動警報程式
 - 數據轉換或事件處理平台
-

如果你提供：

- 你想要傳的資料範例（log 格式或 JSON）
- 接收端是用什麼語言寫的（Python、Java、Go...） 我可以幫你寫出完整範例配置與程式碼！

是否需要我幫你整合完整的 Logstash → TCP Server 的範例？還是你有現成的 Server 想要連接 Logstash？