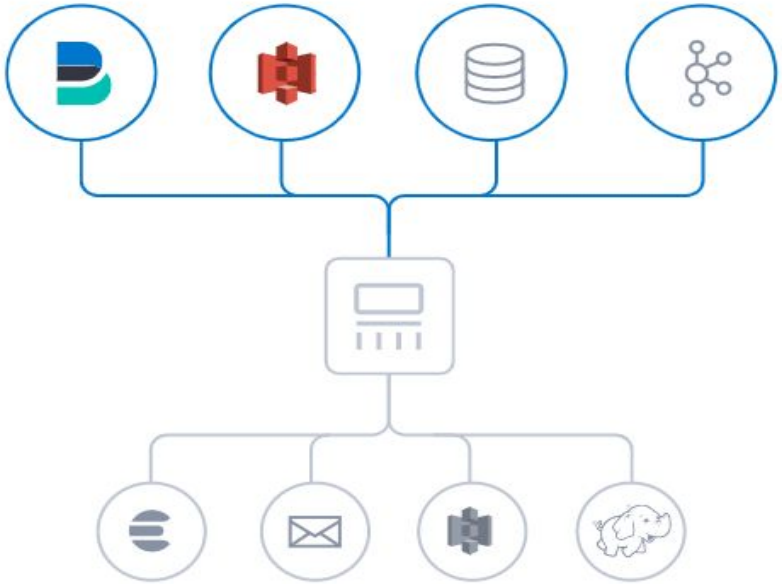
Logstash

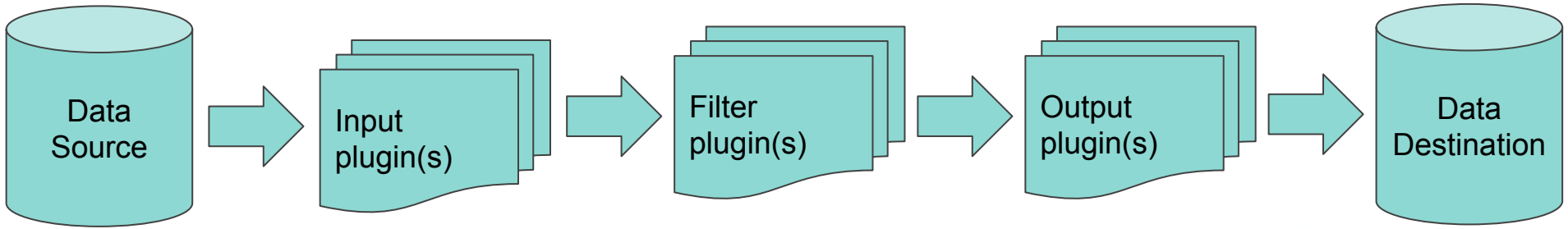
¿Que es logstash?

- Herramienta para procesamiento de datos
- Arquitectura pipeline
- Múltiples fuentes de entrada
- Múltiples fuentes de salida



Arquitectura

Logstash se encuentra compuesto por distintos tipos de plugins, que van a conformar un pipeline para encargarse de la ingesta, procesamiento y salida de los datos.



Ingesta de datos

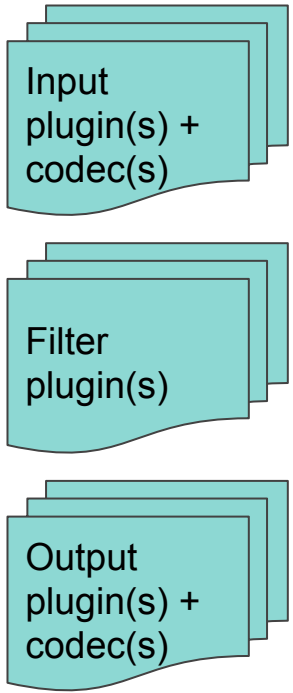
Logstash nos ofrece plugins para ingesta de datos de múltiples fuentes, un aplicación común es ingestar de archivos de logs, para procesarlo y darles un formato que permita su explotación.

Codecs Plugins

- Json
- Json_lines
- Line
- Java_line
- Multiline
- Cloudfront
- Cloudtrail
- Fluent
- Plain
- Rubydebug
- Otros...

Codecs

Un complemento de códec cambia la representación de datos de un evento. Los códecs son esencialmente filtros de flujo que pueden funcionar como parte de una entrada o salida.



Input Plugins

- Stdin
- File
- Beats
- Kafka
- Kinesis
- Redis
- Sqlite
- S3
- Twitter
- Otros...

Configuración de input

El primer paso en logstash es configurar la entrada de los datos, en el ejemplo se selecciona beats (otro ejemplos podrían ser: stdin, kafka, nagios, google cloud,etc).

Link a documentación:

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

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

Ejemplos inputs: file

- Ingestar los datos de un archivo directamente de logstash
- Permite especificar posición de comienzo
- Utilización de codec multiline para eventos multilínea

```
input {
  file {
    path =>
"D:/Users/Usuario/Desktop/Elastic/logs/log-generator*.log"
    exclude => "*.gz"
    start_position => "beginning"
    sincedb_path =>
"D:/Users/Usuario/Desktop/Elastic/logs/log-generator.sincedb"
    codec => multiline {
      pattern => "^(DEBUG|INFO|ERROR|FATAL|TRACE|WARN).*"
      negate => "true"
      what => "previous"
    }
  }
}
```

Ejemplos inputs: twitter

- Recibe datos desde un cuenta de twitter
- Permite la selección de tweets según palabras claves
- Permite seleccionar un lenguaje en particular

```
input {
  twitter {
    consumer_key => "consumer_key"
    consumer_secret => "consumer_secret"
    oauth_token => "access_token"
    oauth_token_secret => "access_token_secret"
    keywords => [ "dotscale" ]
    full_tweet => true
  }
}
```

Output Plugins

- Stdout
- Nagios
- S3
- MongoDB
- ElasticSearch
- Redis
- Google_cloud_storage
- Cloudwatch
- Csv
- Kafka
- Otros...

Configuración de output

Como último paso se define el output del resultado del procesamiento en logstash, en el ejemplo será elasticsearch, pero existen muchas más posibilidades como pueden ser: MongoDB, Kafka, etc.

Link a documentación:
https://www.elastic.co/guide/en/logstash/current/output-plugins.html

```
output {
  stdout{
  }
}
```

Ejemplos Output: MongoDB

- Salida de los datos a mongodb desde logstash
- Recibe especificación de la coleccion, database, uri
- Se agrega el codec “json” para adaptar nuestros docs a la salida de mongo

```
output {
  mongodb {
  id => "my_mongodb_plugin_id"
  collection => "bitcoin"
  database => "uela"
  uri =>
"mongodb://<USER>:<PASS>@ddddddd.mlab.com:63156
/uela"
  codec => "json"
 }
}
```

Ejemplo Output: Csv

- Salida de datos a un archivo ".csv"
- Especificación dé campos que conforman el archivo
- Path absoluto donde se va a generar el archivo

```
output {
  csv {
    fields => ["field1", "field2", "field3", "field4", "field5"]
    path => "/D:/csv-export.csv"
  }
}
```

Configuración de output + Codec

Ejemplo: si se aplica el codec de json_lines el formato de los documentos pasaría del formato por default de logstash:

```
{
   "host" => "PROXYTECH",
   "@version" => "1",
   "@timestamp" => 2019-07-23T13:24:40.107Z,
    "message" => "hola\r"
}
```

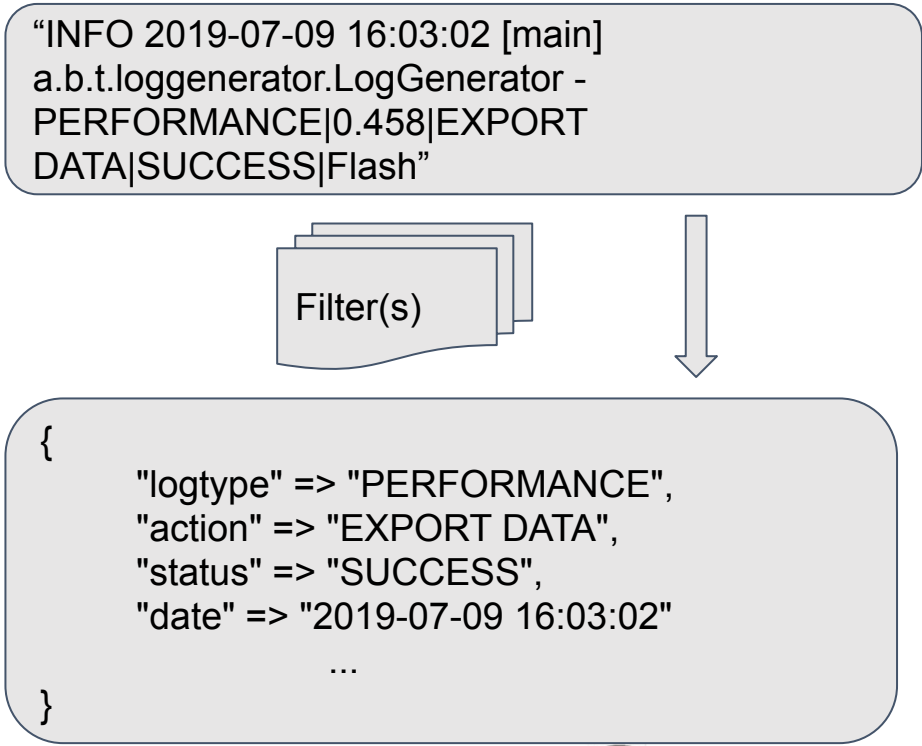
a:

```
{"message":"hola\r","@timestamp":"2019-07-23T13:16:45.
220Z","@version":"1","host":"PROXYTECH"}
```

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

Procesamiento de datos

- Procesamiento por filtros dispuestos en forma de pipeline
- Convierte los datos a un formato explotable
- Caso particular de los logs pasan de ser líneas de texto a un formato clasificado por campo.

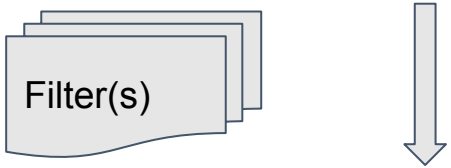


UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Filtros

- Va procesar los datos a través de filtros.
- Los filtros requieren un orden determinado, uno recibe el resultado del anterior
- El resultado del último filtro es enviado al output designado
- Resulta en un formato más simple para explotar en el output designado

```
“INFO 2019-07-09 16:03:02 [main]
a.b.t.loggenerator.LogGenerator -
PERFORMANCE|0.458|EXPORT
DATA|SUCCESS|Flash”
```




```
{
      "logtype" => "PERFORMANCE",
      "action" => "EXPORT DATA",
      "status" => "SUCCESS",
      "date" => "2019-07-09 16:03:02"
      ...
}
```


UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

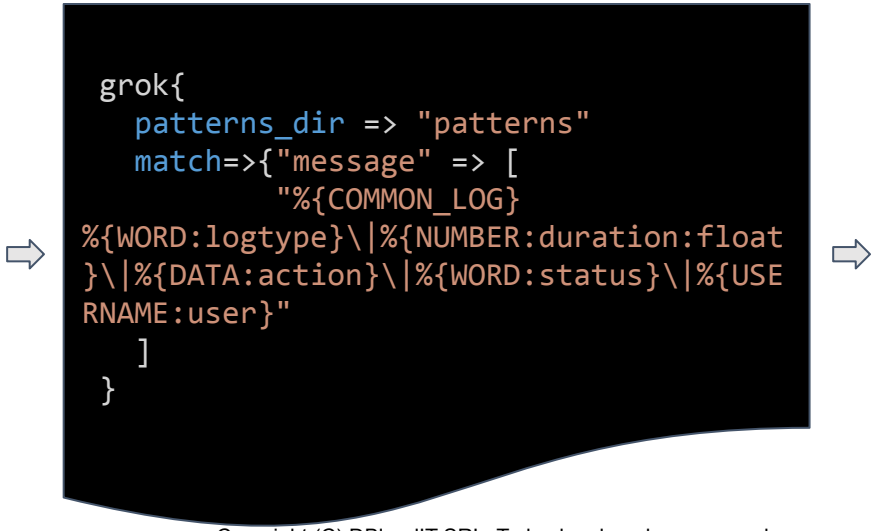
Filter Plugins

- Grok + Dissect
- Mutate + Alter
- Date
- Translate
- Geoip
- Ruby
- Csv
- Json
- Kv (key-value)
- Http
- Aggregate
- Metrics
- Otros...

Grok

- Clasifica los datos en base a un patrón y al nombre que se le asigne a los campos
- Alto costo de procesamiento
- Sirve para registros no estructurados que varían de una línea a otra.

INFO 2019-07-09 15:50:51 [main] a.b.t.loggenerator.LogGenerator - LOGIN|100|Sacarino|68.50.122.189

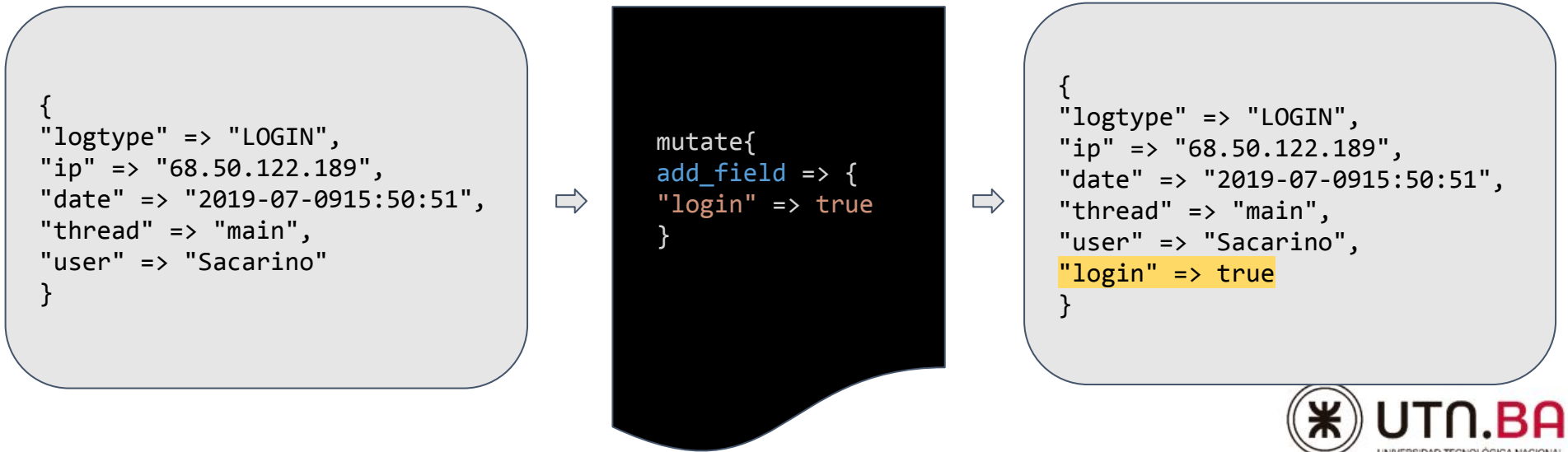


```
grok{
  patterns_dir => "patterns"
  match=>{"message" => [
          "%{COMMON_LOG}
%{WORD:logtype}\|%{NUMBER:duration:float
}\|%{DATA:action}\|%{WORD:status}\|%{USE
RNAME:user}"
  ]
}
```

```
{
"logtype" => "LOGIN",
"ip" => "68.50.122.189",
"date" =>
"2019-07-0915:50:51",
"thread" => "main",
"user" => "Sacarino"
}
```

Mutate

- Sirve para realizar conversiones generales en los campos.
- Permite reemplazar, renombrar, remover y modificar campos.



Condicionales en logstash

- Evaluar según un valor si corresponde aplicar un filtro o no.
- Acepta evaluar contra expresiones regulares con el operador “=~”
- Fundamental para parseo de registros variados.

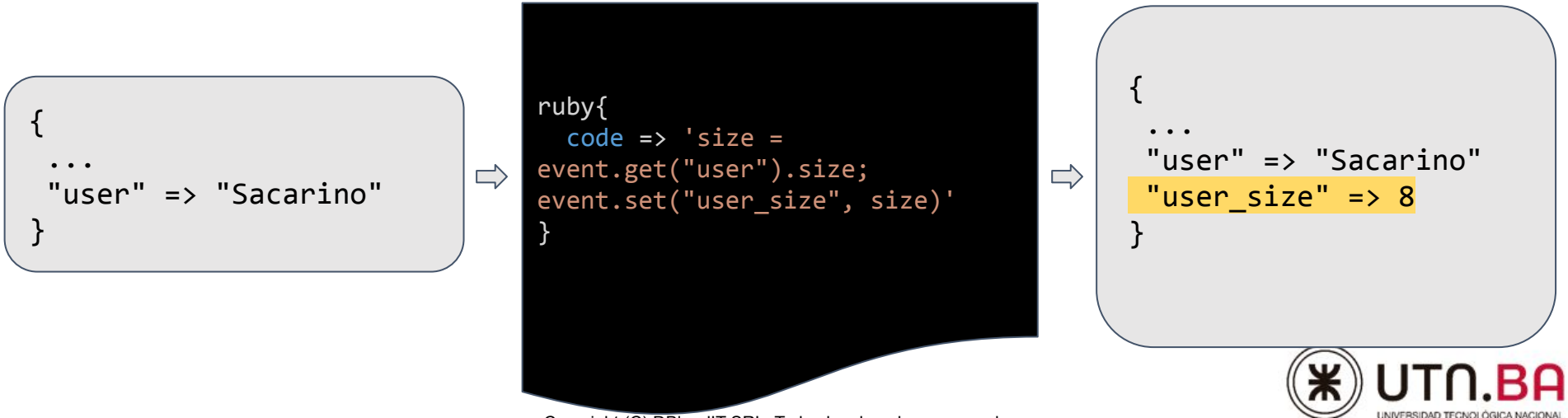
```
{
"ip" => "68.50.122.189",
"date" =>
"2019-07-0915:50:51",
"thread" => "main",
"user" => "Sacarino"
}
```

```
if ![logtype] {
  mutate{
    add_field => {
      "logtype" => "JAVA ERROR"
    }
  }
}
```

```
{
"ip" => "68.50.122.189",
"date" =>
"2019-07-0915:50:51",
"thread" => "main",
"user" => "Sacarino",
"logtype" => "JAVA ERROR"
}
```

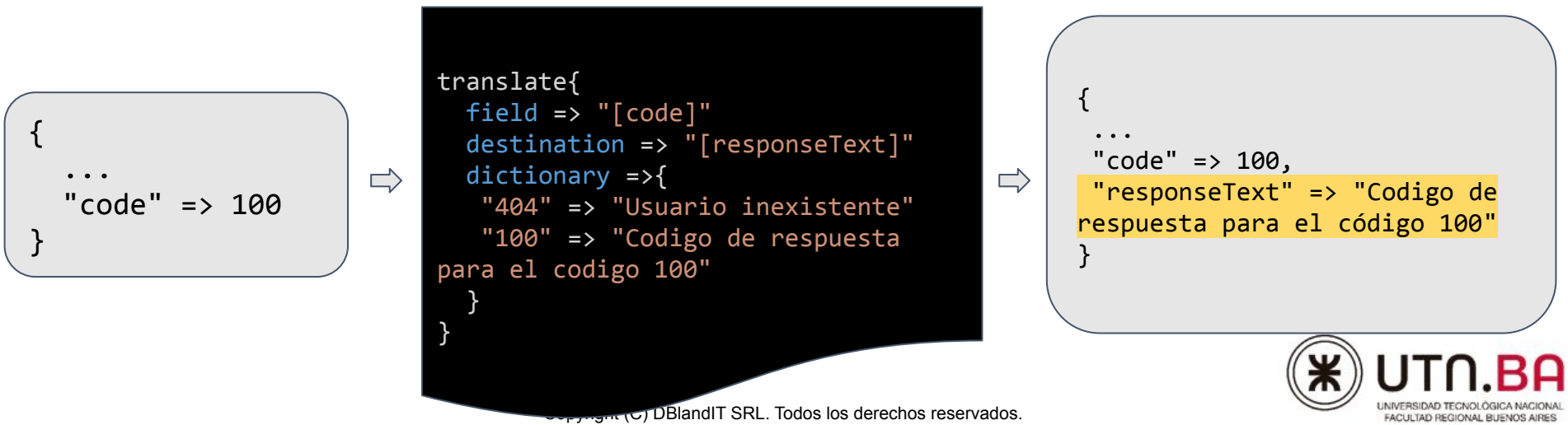
Ruby

- Ejecutar código ruby sobre nuestros eventos.
- Personalización de filtros, agregaciones personalizadas.
- Se pueden ejecutar “inline” o dentros dé un archivo “*.rb”



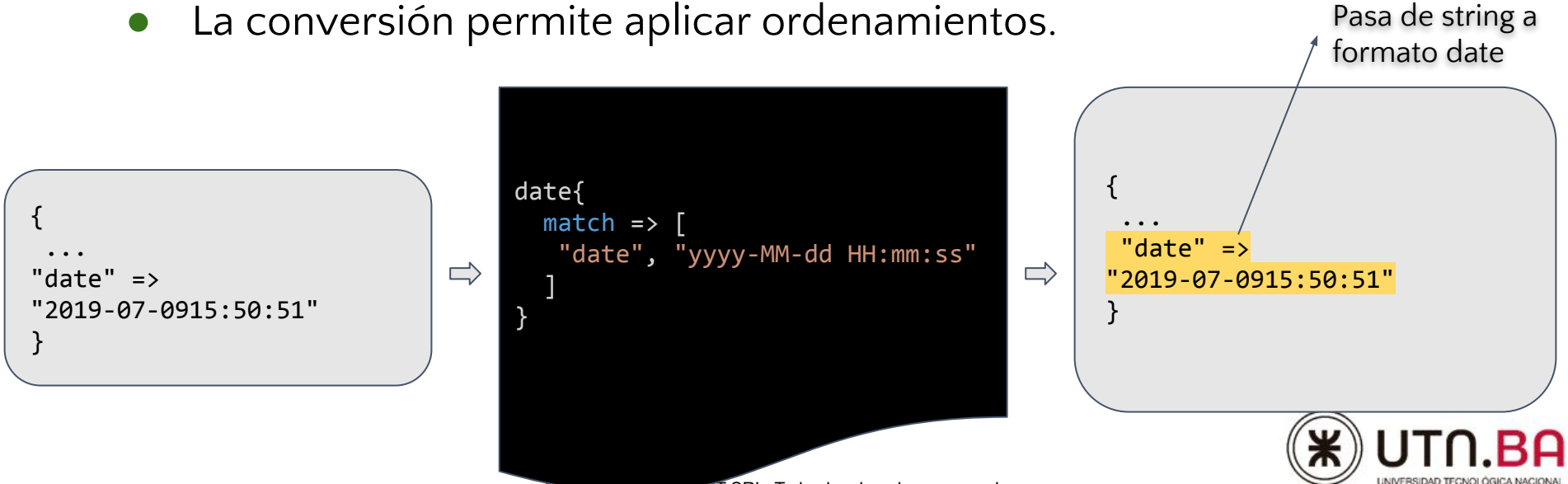
Translate

- Configurar un diccionario para reemplazar el valor de un campo o traducirlo en uno nuevo
- Podemos armar el hash “inline” o en un archivo aparte.



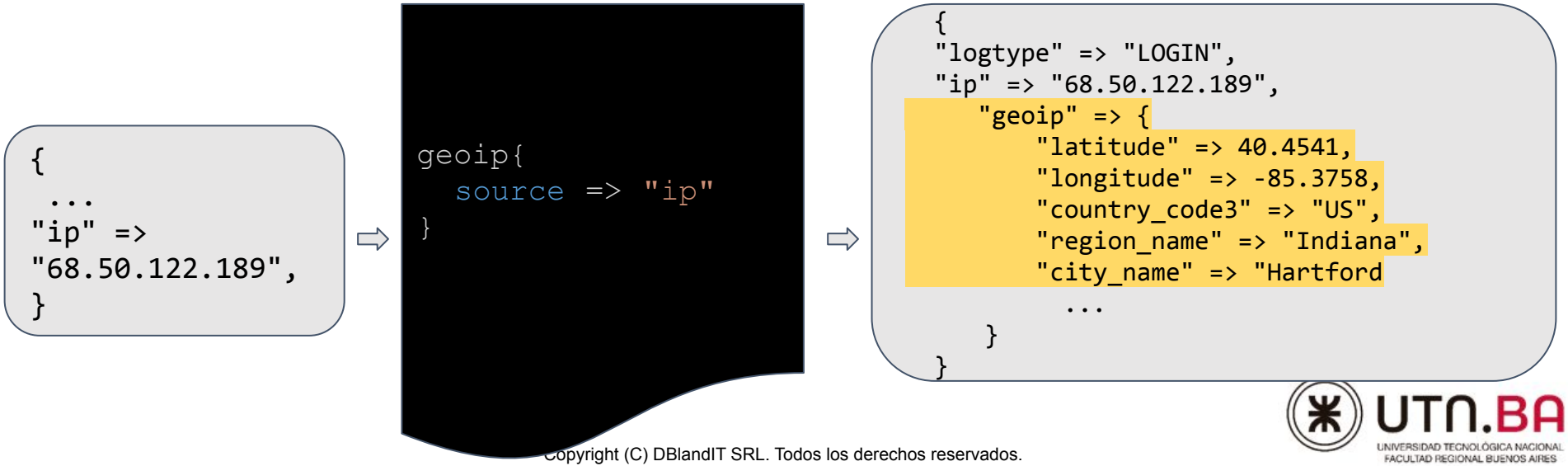
Date

- Filtro particular para el parseo de fechas.
- Usar los campos date como un logstash timestamp
- La conversión permite aplicar ordenamientos.



Geoip

- Agrega información sobre la localización geográfica dé una ip.
- Los campos son de tipo geoPoint, compatibles con geoJSON.
- Obtiene información basado en geoLite2



Errores:Tags

- Contienen información del procesamiento de cada documento
- Informa errores de cada documento en particular
- Utilización de condicionales para evitar indexar documentos erróneos

```
        "tags" => [
        [0] "_grokparsefailure"
    ]
}
```

```
output {
    if "_grokparsefailure" not in [tags] {
        elasticsearch{
            hosts => ["localhost:9200"]
        }
    }else{
        stdout{
        }
    }
}
```

Configuraciones para producción

- Pipeline.Workers: Cantidad de threads a utilizar
- Pipeline.batch.size: límite de tamaño que va a tener el buffer antes de ser procesado y enviado al output.
- Pipeline.batch.delay: límite de tiempo que espera a que se llene el buffer antes de ser enviado

```
This defaults to the number of the host's CPU cores.

pipeline.workers: 2

How many events to retrieve from inputs before sending to filters+workers

pipeline.batch.size: 125

How long to wait in milliseconds while polling for the next event
before dispatching an undersized batch to filters+outputs

pipeline.batch.delay: 50
```

Múltiples pipelines

- Procesar registros en múltiples pipelines
- Configuración y asignación de recursos independiente para cada pipeline
- Configuración “inline” o en un archivo particular

```
- pipeline.id: test
  pipeline.workers: 1
  pipeline.batch.size: 1
  config.string: "input { generator {} } filter { sleep { time => 1 } } output { stdout { codec => dots } }"
- pipeline.id: another_test
  queue.type: persisted
  path.config: "/tmp/logstash/*.config"
```

Configuración de colas

- queue.type:
  - Memory: Mayor Velocidad y Mayor Volatilidad
  - Persisted: Menor Velocidad Y Mayor durabilidad, conveniente para carga de lotes grandes
- queue.page_capacity: tamaño de página de la cola
- queue.max_events : cantidad de eventos máximos a almacenar en la cola

```
queue.type: memory
queue.page_capacity: 64mb
queue.max_events: 0
```

Configuración automática

Carga la configuración sin necesidad de pasarle el parámetro

- “path.config” carga la configuración desde un archivo
- “config.string” carga la configuración desde una cadena

```
Where to fetch the pipeline configuration
for the main pipeline

path.config:/tmp/logstash/*.config

Pipeline configuration string for the
main pipeline

config.string:"input { generator {} }
filter { sleep { time => 1 } } output {
stdout { codec => dots } }"
```

Directorio de configuraciones

- Montaje de todas las configuraciones del directorio en un solo archivo
- Problema: todos los logs son procesados por todas las configuraciones

Directorio de configuraciones (Solucionar Problema)

- Agregar un campo por registro en el input que especifique a qué configuración corresponde
- Aplicar condicionales con el campo agregado anteriormente, para identificar los filtros que corresponden a cada aplicación

```
input {
    beats{
        port => 5044
        add_field =>
{application=>"log-generator"}
    }
}
```

Monitoreo de Logstash

- Tiempo de procesamiento para cada filtro
- Recursos utilizados para el procesamiento
- Api para consultar métricas de procesamiento:

  http://localhost:9600/_node/stats?pretty


```
{
 "id" :
"4a51b13e4b67376d926f16ef8e366047aac099ce239f8d6a
f8bb71bf26779a4b",
 "events" : {
   "out" : 1,
   "duration_in_millis" : 175,
   "in" : 1
 },
 "name" : "geoip"
}
```

Ids Monitorización

- Facilita identificación de filtros a la hora de monitorizar
- Se agregan en nuestro archivo de configuración de logstash
- Resuelve la identificación de filtros repetidos

```
geoip{
        source => "ip"
        id => "geolocalizacion-ips"
    }
```

```
{
  "id" : "geolocalizacion-ips",
  "name" : "geoip",
  "events" : {
    "duration_in_millis" : 0,
    "out" : 0,
    "in" : 0
  }
}
```

Logstash + Elastic Output

- El campo index asigna un patrón para el almacenamiento y rotación de índice
- Fundamental para evitar índices de gran tamaño
- Evita que las búsquedas sean ineficientes

```
output {
  elasticsearch{
    hosts => ["localhost:9200"]
    index =>
"log-workshop-%{+YYYY.MM.dd}"
  }
}
```

Preguntas?