阿里云开发者社区

日志服务数据加工系列培训

主题：扫平日志分析路上障碍，实时海量日志加工实践培训

讲师：丁来强（成喆）- 阿里高级技术专家 | 唐恺(风毅) - 阿里技术专家

分享介绍

8月7日 19:30-20:30	8月8日 19:30-20:30	8月13日 19:30-20:30	8月14日 19:30-20:30	8月20日 19:30-20:30	8月21日 19:30-20:30	8月28日 19:30-20:30	8月29日 19:30-20:30
数据加工 介绍与实战	数据加工DSL 核心语法介绍	数据加工DSL 语法实践	数据加工动态 数据分发汇集实践	非结构化数据 解析实践	结构化数据 解析实践	数据映射 富化实践	数据加工 可靠性与排错实践

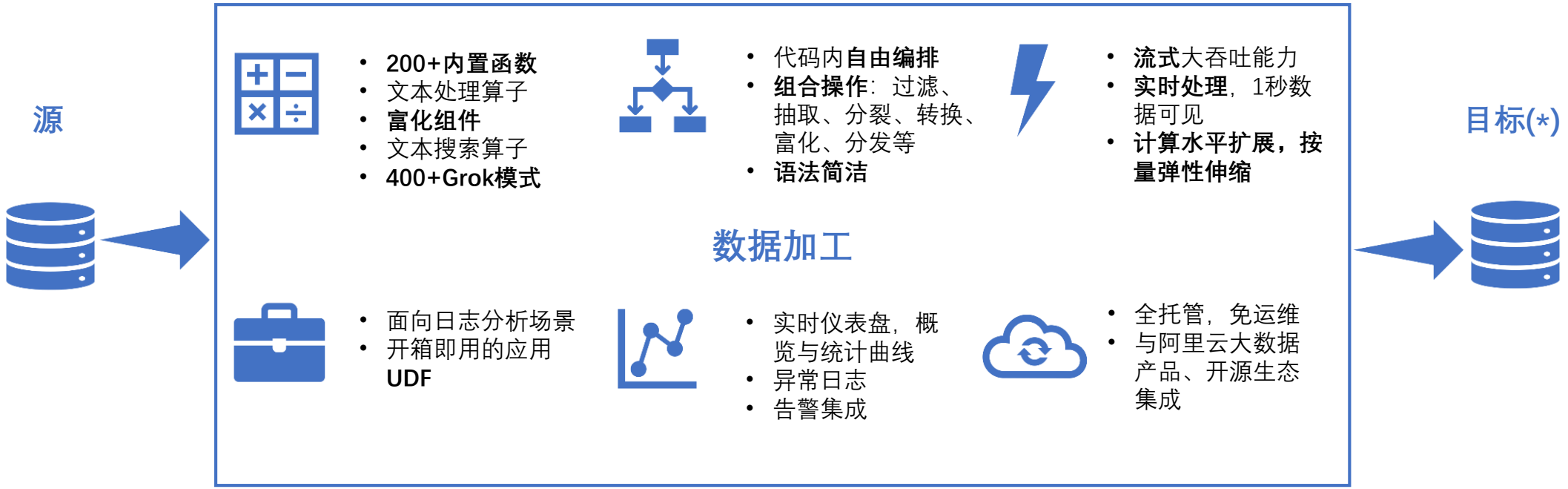
数据加工:结构化化数据解析实践

系列培训六

唐恺

日志服务-数据加工简介

- 功能概述
  - 将各类日志处理为结构化数据，具备全托管、实时、高吞吐的特点
  - 面向日志分析领域，提供丰富算子、开箱即用的场景化UDF（Syslog、非标准json、AccessLog UA/URI/IP解析等）
  - 丰富的阿里云大数据产品（OSS、MC、EMR、ADB等）、开源生态（Flink、Spark等）集成能力，降低数据分析门槛
- 典型场景
  - 数据规整：对混乱格式的日志进行字段提取、格式转换，获取结构化数据以支持后续的流处理、数仓计算
  - 数据富化：日志（例如业务订单）与维表（例如用户信息MySQL表）进行字段join，为日志添加更多维度信息供分析
  - 数据分发：将全量日志按转发规则分别提取到多个下游存储供不同业务使用



分隔符日志

分隔符格式

```
"Date","Pupil","Grade"
"25 May","Bloggs, Fred","C"
"25 May","Doe, Jane","B"
"15 July","Bloggs, Fred","A"
"15 April","Muniz, Alvin ""Hank""","A"
```

e_csv/e_psv/e_tsv

功能：使用分隔符与预定义字段名, 提取事件特定字段为多个字段. 分隔符可以自定义.

语法:

e_csv(源字段名, 目标字段列表, sep=",", quote="", restrict=True, mode="fill-auto")

e_psv(源字段名, 目标字段列表, sep="|", quote="", restrict=True, mode="fill-auto")

e_tsv(源字段名, 目标字段列表, sep="\t", quote="", restrict=True, mode="fill-auto")

参数名称	字段属性	是否必填	说明
源字段名	任意	是	任意字符（特殊字段名的设置, 可以参考事件类型）
目标字段列表	任意	是	源字段值经过分隔符切分后的每个值对应的字段名； 可以字符串的列表, 例如: ["a", "b", "c"]； 也可以直接用逗号分隔的字符串, 例如: "a, b, c"
sep	String	否	分隔符, 只能是单个字符
quote	String	否	将值的括起来的字符, 当值也包含分隔符时需要使用.
restrict	Bool	否	严格模式, 分隔的值的个数是否与目标字段列表数一致, 默认False（当不匹配, 严格模式下, 则不进行任何操作； 当不匹配, 非严格模式下, 对前几个可以配对的字段进行赋值）
mode	String	否	默认fill-auto， 请参考覆盖模式（无任何匹配时, 不进行任何操作）

e_csv示例

```
2 e_csv("content", "f_time, f_ip, f_method, f_uri, f_protocol, f_status, f_ua, f_else", quote='\1', restrict=False)
3 e_csv("content", "s_time, s_ip, s_method, s_uri, s_protocol, s_status, s_ua, s_else", quote='"', restrict=False)
```

预览任务开始时间： **2019-08-20 15:41:56**

原始日志 | 数据加工 new

>	输出目标	时间	内容
1	target0	08-21 13:02:51	__source__ : 1.2.3.4 __tag__:__receive_time__ : 1566364383 __topic__ : csv_log content : 2019-08-21 11:13:43,112.120.150.12,GET,/mock/ref-uri/x7.icon,HTTP/1.0,400,"Mozilla/5.0 (compatible; DotBot/1.1; http://www.opensiteexplorer.org/dotbot, help@moz.com)" f_else : help@moz.com)" f_ip : 112.120.150.12 f_method : GET f_protocol : HTTP/1.0 f_status : 400 f_time : 2019-08-21 11:13:43 f_ua : "Mozilla/5.0 (compatible; DotBot/1.1; http://www.opensiteexplorer.org/dotbot f_uri : /mock/ref-uri/x7.icon s_ip : 112.120.150.12 s_method : GET s_protocol : HTTP/1.0 s_status : 400 s_time : 2019-08-21 11:13:43 s_ua : Mozilla/5.0 (compatible; DotBot/1.1; http://www.opensiteexplorer.org/dotbot, help@moz.com) s_uri : /mock/ref-uri/x7.icon

KV日志

KV格式


```
2012-06-17 17:02:08,880 -0700 INFO  [hostId=prod-frontend-5] [module=SERVICE]
[logger=service.endpoint.search.v1.impl.SearchServiceImpl] [thread=Thread-797 (group:HornetQ-client-global-threads-2122452307)]
[auth=User:test@demo.com:000000000000BE79:0000000000000005:false] [remote_ip=50.18.185.122] [web_session=q9hkzjfa...]
[session=B046486365A098F9] [customer=0000000000000005] [call=OutboundStreamProtocol.records]
Count update for stream session: 'B046486365A098F9', stream query: 'B046486365A098F9', alias: 'query.count', count: '11000'
```

e_kv

功能：通过quote等自动提取多个源字段中的键值对信息.

语法:

e_kv(源字段或源字段列表, sep="=", quote="", escape=False, prefix="", suffix="", mode="fill-auto")

参数名称	字段属性	是否必填	说明
源字段或源字段列表	任意	是	字段名或多个字段名的列表（特殊字段名的设置, 可以参考事件类型；对每个匹配的字段进行操作, 如果没有匹配的, 则不进行任何操作.
sep	String	否	分隔关键字与值的正则字符串, 不限于单个字符, 注意：不能使用捕获分组（但可以使用不捕获分组）
quote	String	否	将值的括起来的字符
escape	Bool	否	是否自动提取反转字符的值, 例如key="abc\"xyz" 默认提取字段key值为abc\"xyz, 设置escape时, 值为abc"xyz
prefix	String	否	给提取的字段名添加前缀
suffix	String	否	给提取的字段名添加后缀
mode	String	否	默认fill-auto，请参考覆盖模式

e_kv示例

```
1 e_keep_fields("request_uri")
2 e_kv("request_uri")
```

预览任务开始时间： **2019-08-20 15:34:38**

原始日志 | 数据加工 new

	输出目标	时间	内容
1	target0	08-21 15:35:25	__source__ : __topic__ : qc33_ft : 0 qc33_rx93 : 100 rbtq_xg : 4201201 rbxqo3fl_rlclnr : WAIT_SHIP request_uri : /rxf3l3/rbxqo3flr/33l?qlcl9txo_txg3x_rfr=&lxcxzxf3_fno73x=&ot7xl3=&qxtxxfx3_xg=&rbxqo3fl_rlclnr=WAIT_SHIP&rlclnr=CONFIRM&xr_qxxfl_lxcxzxf3=&qc33_rx93=100&qc33_ft=0&x3cl_qc33_ft=0&3ttgr_xg=&rzn_lxrl=&xr_ftl3=cll&ftl3_r3cxxb=&xr_tflg_rbtw_xcf_o3x33=0&rlcxl_lxo3=&3fg_lxo3=&3ttgr_cotnfl=&lb3xocl_lgq3=&rbtq_xg=4201201 rlclnr : CONFIRM x3cl_qc33_ft : 0 xr_ftl3 : cll xr_tflg_rbtw_xcf_o3x33 : 0

e_kv_delimit

功能：通过分隔符，提取源字段中的键值对信息.

语法:

e_kv_delimit(源字段或源字段列表, pair_sep=r"\s", kv_sep="=", prefix="", suffix="", mode="fill-auto")

参数名称	字段属性	是否必填	说明
源字段或源字段列表	任意	是	字段名或多个字段名的列表（特殊字段名的设置, 可以参考事件类型；对每个匹配的字段进行操作, 如果没有匹配的, 则不进行任何操作.
pair_sep	String	否	用于分隔键值对的正则字符集，如\s\w, abc\s等, 但不是子串, 任意满足的字符会分隔键值对,
kv_sep	String	否	用于分隔KV的正则字符串, 不限于单个字符, 注意：不能使用捕获分组（但可以使用不捕获分组, 如(?:=|&
prefix	String	否	给提取的字段名添加前缀
suffix	String	否	给提取的字段名添加后缀
mode	String	否	默认fill-auto，请参考覆盖模式

e_kv_delimit

```
1 e_kv_delimit("content", pair_sep=r"\t", kv_sep=":", prefix="res_", mode="fill-auto")
```

预览任务开始时间： **2019-08-21 11:27:09**

原始日志 | 数据加工 new

>	输出目标	时间	内容
1	target0	08-21 12:55:19	__source__ : 1.2.3.4 __tag__:__receive_time__ : 1566364449 __topic__ : kv_log content : time:2019-08-21 11:48:45 client_ip:101.120.152.18 request_method:GET request_uri:/mock/ref-uri/z4.js server_protocol:HTTP/1.0 status:200 http_user_agent:Microsoft Internet Explorer res_client_ip : 101.120.152.18 res_http_user_agent : Microsoft Internet Explorer res_request_method : GET res_request_uri : /mock/ref-uri/z4.js res_server_protocol : HTTP/1.0 res_status : 200

e_kv vs e_kv_delimit

- e_kv
  - 支持escape
  - 支持quote
  - 非标准格式可能存在歧义
- e_kv_delimit
  - 不支持escape
  - 不支持quote
  - 严格按照设定的两种分隔符进行

```
2 e_kv("content", sep="=", quote='"', escape=True, prefix='f_')
3 e_kv_delimit("content", pair_sep="|", kv_sep="=", prefix="s_")
```

预览任务开始时间： 2019-08-20 15:52:32

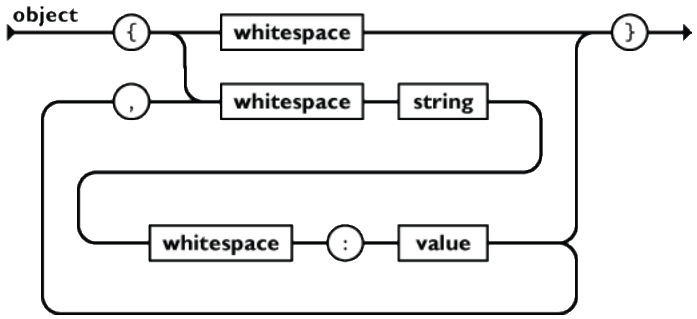
原始日志 | 数据加工 new

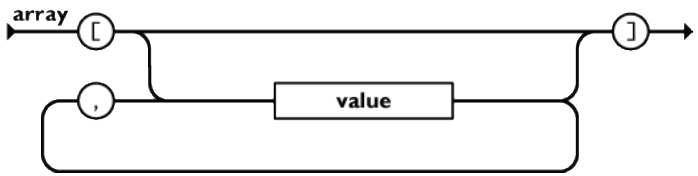
	输出目标	时间	内容
1	target0	08-21 12:55:19	__source__ : 1.2.3.4 __tag__:__receive_time__ : 1566364449 __topic__ : kv_log content : time=2019-08-21 12:55:51|client_ip=115.120.156.8|request_method=GET|request_uri="/mock/ref-uri/z1.icon"|server_protocol=HTTP/2.0|status=403|http_user_agent=Mozilla/5.0 (compatible; DotBot/1.1; http://www.opensiteexplorer.org/dotbot, help@moz.com) f_client_ip : 115.120.156.8 f_http_user_agent : Mozilla f_request_method : GET f_request_uri : /mock/ref-uri/z1.icon f_server_protocol : HTTP f_status : 403 f_time : 2019-08-21 s_client_ip : 115.120.156.8 s_http_user_agent : Mozilla/5.0 (compatible; DotBot/1.1; http://www.opensiteexplorer.org/dotbot, help@moz.com) s_request_method : GET s_request_uri : "/mock/ref-uri/z1.icon" s_server_protocol : HTTP/2.0 s_status : 403 s_time : 2019-08-21 12:55:51

JSON日志

JSON格式

- JSON is built on two structures（https://www.json.org）
  - A collection of name/value pairs. In various languages, this is realized as an *object*, record, struct, dictionary, hash table, keyed list, or associative array.
  - An ordered list of values. In most languages, this is realized as an *array*, vector, list, or sequence.





- JMES
  - http://jmespath.org/tutorial.html

数据加工中的JSON对象

一般指JSON表达式函数json_select或者json_parse解析提取后的对象

原始字符串	解析出的JSON对象	实际类型
1	1	整数
1.2	1.2	浮点
true	True	布尔
false	False	布尔
"abc"	"abc"	字符串
null	None	None
["v1", "v2", "v3"]	["v1", "v2", "v3"]	列表
["v1", 3, 4.0]	["v1", 3, 4.0]	列表
{"v1": 100, "v2": "good"}	{"v1": 100, "v2": "good"}	字典
{"v1": {"v11": 100, "v2": 200}, "v3": "good"}	{"v1": {"v11": 100, "v2": 200}, "v3": "good"}	字典

json_select

功能：以JMES语法对(字段或表达式表示JSON的)值提取或计算特定值

语法：

json_select(字段, "jmes表达式", default=None, restrict=False)

参数名称	字段属性	是否必填	说明
字段	String	是	填入需要被转换的字段
jmes表达式	String	是	填入需要被提取的值
default	任意	否	默认None
restrict	Bool	否	默认False

json_select示例

```
1 e_keep(e_search("__topic__: user_login and content:user6043"))
2 e_set("user_7199_login_records", json_select(v("content"), "users[?name=='user6043'].login_histories[*]"))
3 e_set("first_login_date", json_select(v("content"), "users[0].login_histories[0].date"))
```

预览任务开始时间： **2019-08-20 16:43:49**

原始日志 | 数据加工 new

>	输出目标	时间	内容
1	target0	08-21 13:07:19	__source__ : 1.2.3.4 __tag__:__receive_time__ : 1566364054 __topic__ : user_login content : {"users": [{"name": "user7199", "login_histories": [{"date": "2019/08/21 11:37:18", "login_ip": "107.120.155.4"}]}, {"name": "user6043", "login_histories": [{"date": "2019/08/21 11:44:37", "login_ip": "103.120.150.9"}, {"date": "2019/08/21 12:05:05", "login_ip": "101.120.155.14"}, {"date": "2019/08/21 13:01:43", "login_ip": "111.120.159.8"}]}, {"name": "user4403", "login_histories": [{"date": "2019/08/21 12:13:20", "login_ip": "109.120.153.12"}]}]} first_login_date : 2019/08/21 11:37:18 user_7199_login_records : [[{"date": "2019/08/21 11:44:37", "login_ip": "103.120.150.9"}, {"date": "2019/08/21 12:05:05", "login_ip": "101.120.155.14"}, {"date": "2019/08/21 13:01:43", "login_ip": "111.120.159.8"}]]

e_json

功能：对事件特定字段中的JSON值进行JSON操作, 包括: 展开, 或者JMES提取, 或者JMES提取后展开, 可以进行深度的展开配置.

语法:

e_json(源字段名, expand=None, depth=100, prefix="__", suffix="__", fmt="simple", sep=".", expand_array=True, fmt_array="{parent}_{index}", include_node=r"[\u4e00-\u9fa5\u0800-\u4e00a-zA-Z][\w\-\.]*", exclude_node="", include_path="", exclude_path="", jmes="", output="", jmes_ignore_none=False, mode='fill-auto')

参数名称	字段属性	是否必填	说明
源字段名	String	是	任意字符（特殊字段名的设置, 可以参考事件类型; 注意: 如果字段在事件中不存在, 则不进行任何操作.）
expand	Bool	否	是否展开, 在没有配置jmes时默认展开（True）， 设置了jmes时， 默认不展开（False）
depth	Number	否	展开深度, 默认100层
prefix	String	否	展开时节点名作为字段名加的前缀
suffix	String	否	展开时节点名作为字段名加的后缀

e_json

参数名称	字段属性	是否必填	说明
fmt	String	否	simple, full, parent, root, 默认simple,参考样例3
sep	String	否	full, parent, root格式化时与父子节点的分隔符, 默认"."
expand_array	Bool	否	数组展开的格式化方式, 默认: {parent}_{index}
fmt_array	String	否	数组展开的格式化方式 , 默认: {parent_rlist[0]}_{index}，也可以使用最多五个占位符自定义格式化字符串：parent_list，current，sep，prefix，suffix
include_node	String/ Number	否	根据节点名称过滤, 默认只有中文数字字母和_.-的节点才会被自动展开
exclude_node	String	否	根据节点名称排除
include_path	String	否	根据节点路径过滤
exclude_path	String	否	根据节点路径排除
jmes	String	否	对字段值转化为json按照jmes提取特定值. 参考JSON使用JMES过滤
output	String	否	jmes提取的输出字段名
jmes_ignore_none	Bool	否	当jmes提取不到值时, 是否忽略, 默认True, 否则输出output为一个空值.
mode	String	否	默认fill-auto，请参考覆盖模式（无任何匹配时, 不进行任何操作）

e_json示例

```
1 e_keep(e_search("__topic__:  server_status"))
2 e_json("content", prefix="s_", fmt='simple')
3 e_json("content", fmt='full', prefix="f_", sep='->', exclude_path=r'.*status|service', depth=2)
4 e_json("content",fmt='full',  prefix="fa_", expand_array=True, fmt_array='{parent_rlist[0]}[{index}]')
```

预览任务开始时间： **2019-08-20 17:14:23**

15 target0 08-21 12:42:09

__source__ : 1.2.3.4
__tag__:__receive_time__ : 1566364246
__topic__ : server_status
content : {"service": "DSL_service", "overall_status": "running", "servers": [{"host": "114.120.154.9", "status": "green"}, {"host": "101.120.151.11", "status": "yellow"}], "clients": [{"host": "102.120.159.19", "status": "red"}, {"host": "113.120.153.13", "status": "red"}]}
content->clients->clients_0->f_clients_0 : {"host": "102.120.159.19", "status": "red"}
content->clients->clients_1->f_clients_1 : {"host": "113.120.153.13", "status": "red"}
content->f_service : DSL_service
content->servers->servers_0->f_servers_0 : {"host": "114.120.154.9", "status": "green"}
content->servers->servers_1->f_servers_1 : {"host": "101.120.151.11", "status": "yellow"}
content.clients.clients[0].fa_host : 102.120.159.19
content.clients.clients[0].fa_status : red
content.clients.clients[1].fa_host : 113.120.153.13
content.clients.clients[1].fa_status : red
content.fa_overall_status : running
content.fa_service : DSL_service
content.servers.servers[0].fa_host : 114.120.154.9
content.servers.servers[0].fa_status : green
content.servers.servers[1].fa_host : 101.120.151.11
content.servers.servers[1].fa_status : yellow
s_host : 113.120.153.13
s_overall_status : running
s_service : DSL_service
s_status : red

e_split

- **功能**：基于字段的值进行分裂出多个事件. 事件所有值都一样, 除了基于的字段的值是具体某一项. 也支持基于JMES提取字段后再进行分裂
- **语法**:

e_split(字段名, sep=',', quote='"', lstrip=True, jmes=None, output=None)

参数名称	字段属性	是否必填	说明
字段名	String	是	针对的字段名（特殊字段名的设置, 可以参考事件类型、分裂规则.）
sep	String	否	按照此分隔符分隔字段的值
quote	String	否	这个字符配对类的分隔符视为值的一部分
lstrip	String	否	是否将提取的到的值左边的空格去掉. 默认True
jmes	String	否	对字段值转化为json按照jmes提取特定值后再进行分裂操作.
output	String	否	输出的字段默认覆盖原字段, 可以设置一个新的字段名.

e_split示例

```
1 e_keep(e_search("__topic__:  server_status"))
2 e_split("content", jmes='clients[*]', output='c')
3 e_json("c", fmt="root")
```

预览任务开始时间： **2019-08-20 17:25:10**

原始日志 | 数据加工 new

>	输出目标	时间	内容
1	target0	08-21 13:08:09	__source__ : 1.2.3.4 __tag__:__receive_time__ : 1566364252 __topic__ : server_status c : {"host": "120.120.152.18", "status": "red"} c.host : 120.120.152.18 c.status : red content : {"service": "DSL_service", "overall_status": "running", "servers": [{"host": "110.120.156.5", "status": "green"}, {"host": "110.120.151.9", "status": "green"}, {"host": "116.120.153.20", "status": "red"}, {"host": "105.120.150.9", "status": "green"}, {"host": "114.120.158.9", "status": "red"}], "clients": [{"host": "110.120.154.17", "status": "yellow"}, {"host": "120.120.152.18", "status": "red"}]}
2	target0	08-21 13:08:09	__source__ : 1.2.3.4 __tag__:__receive_time__ : 1566364252 __topic__ : server_status c : {"host": "110.120.154.17", "status": "yellow"} c.host : 110.120.154.17 c.status : yellow content : {"service": "DSL_service", "overall_status": "running", "servers": [{"host": "110.120.156.5", "status": "green"}, {"host": "110.120.151.9", "status": "green"}, {"host": "116.120.153.20", "status": "red"}, {"host": "105.120.150.9", "status": "green"}, {"host": "114.120.158.9", "status": "red"}], "clients": [{"host": "110.120.154.17", "status": "yellow"}, {"host": "120.120.152.18", "status": "red"}]}

其它结构化日志

其它结构化日志

- XML
- protobuf
- 更多格式（评估需求后支持）

综合实践：复杂JSON解析

场景

- 非标json
  - 单引号
  - logstash风格
- 复杂json
  - array展开
  - 多层嵌套
- 多种方式组合（消除歧义，预处理）
  - e_regex
  - 字符串函数

logstash config

```
1 e_keep(e_search('__topic__:  logstash'))
2 e_set('fixed_content', str_logtash_config_normalize(v('content')))
3 e_json('fixed_content')
```

预览任务开始时间： **2019-08-20 17:44:07**

原始日志 | 数据加工 new

>	输出目标	时间	内容
1	target0	08-21 13:09:39	__source__ : 1.2.3.4 __tag__:__receive_time__ : 1566380493 __topic__ : logstash content : {"service" => "search_service", "overall_status" => "running", "servers" => [{"host" => "119.120.160.5", "status" => "red"}, {"host" => "107.120.160.7", "status" => "green"}, {"host" => "111.120.155.12", "status" => "green"}, {"host" => "115.120.155.20", "status" => "yellow"}, {"host" => "116.120.153.8", "status" => "red"}], "clients" => [{"host" => "102.120.152.2", "status" => "yellow"}]} fixed_content : {"service" : "search_service", "overall_status" : "running", "servers" : [{"host" : "119.120.160.5", "status" : "red"}, {"host" : "107.120.160.7", "status" : "green"}, {"host" : "111.120.155.12", "status" : "green"}, {"host" : "115.120.155.20", "status" : "yellow"}, {"host" : "116.120.153.8", "status" : "red"}], "clients" : [{"host" : "102.120.152.2", "status" : "yellow"}]} host : 102.120.152.2 overall_status : running service : search_service status : yellow

非标json

```
1 e_keep(e_search('__topic__:  single_quotation'))
2 e_set('fixed_content', str_replace(v('content'), "'", '"'))
3 e_json('fixed_content')
```

预览任务开始时间： **2019-08-20 17:45:34**

原始日志 | 数据加工 new

>	输出目标	时间 ▲▼	内容
1	target0	08-21 12:55:19	__source__ : 1.2.3.4 __tag__:__receive_time__ : 1566380418 __topic__ : single_quotation content : {'service': 'DSL_service', 'overall_status': 'stopped', 'servers': [{'host': '102.120.150.8', 'status': 'yellow'}, {'host': '118.120.153.18', 'status': 'red'}, {'host': '116.120.158.19', 'status': 'green'}, {'host': '117.120.154.5', 'status': 'green'}, {'host': '113.120.150.2', 'status': 'green'}], 'clients': [{'host': '120.120.153.2', 'status': 'green'}, {'host': '120.120.158.2', 'status': 'red'}]} fixed_content : {"service": "DSL_service", "overall_status": "stopped", "servers": [{"host": "102.120.150.8", "status": "yellow"}, {"host": "118.120.153.18", "status": "red"}, {"host": "116.120.158.19", "status": "green"}, {"host": "117.120.154.5", "status": "green"}, {"host": "113.120.150.2", "status": "green"}], "clients": [{"host": "120.120.153.2", "status": "green"}, {"host": "120.120.158.2", "status": "red"}]} host : 120.120.158.2 overall_status : stopped service : DSL_service status : red

复杂json

```
e_keep(e_search("__topic__:user_login"))
e_split("content", jmes='users[*]', output='tmp_user')
e_json("tmp_user", depth=1, prefix='tmp_')
e_split("tmp_login_histories", output='tmp_login')
e_json('tmp_login')
e_rename('tmp_name', 'name')
e_drop_fields('content', r'tmp_.*')
```

预览任务开始时间： **2019-08-20 18:00:38**

原始日志 | 数据加工 new

	输出目标	时间	内容
1	target0	08-21 12:53:49	__source__ : 1.2.3.4 __tag__:__receive_time__ : 1566364050 __topic__ : user_login date : 2019/08/21 13:07:13 login_ip : 113.120.158.14 name : user4411

总结

结构化数据解析实践总结

- 分隔符日志
  - 优先考虑e_csv/e_psv/e_tsv函数
  - 如果日志可能存在分隔符歧义，请使用quote
  - 如果是自定义格式，可以考虑用正则（e_regex）做预先提取
- KV日志
  - 标准格式（带quote），直接用e_kv
  - 自定义格式，可以用e_kv_delimit
- JSON日志
  - 学习JMES
  - 掌握：e_json、json_select、e_split
  - 非标准格式：结合字符串或正则函数做预处理
- 其它日志
  - 数据加工支持：XML、Protobuf等解析函数
  - 更多请联系我们

阿里云开发者社区

日志服务数据加工系列培训

主题：扫平日志分析路上障碍，实时海量日志加工实践培训

讲师：丁来强（成喆）- 阿里高级技术专家 | 唐恺(风毅) - 阿里技术专家

分享介绍

8月7日 19:30-20:30	8月8日 19:30-20:30	8月13日 19:30-20:30	8月14日 19:30-20:30	8月20日 19:30-20:30	8月21日 19:30-20:30	8月28日 19:30-20:30	8月29日 19:30-20:30
数据加工 介绍与实战	数据加工DSL 核心语法介绍	数据加工DSL 语法实践	数据加工动态 数据分发汇集实践	非结构化数据 解析实践	结构化数据 解析实践	数据映射 富化实践	数据加工 可靠性与排错实践