Table of Contents

dtle 中文技术参考手册

目录

参考 gitbook 左侧目录区 或 SUMMARY.md

PDF下载

PDF下载

官方技术支持

- 代码库 github: github.com/actiontech/dtle
- 文档库 github: github.com/actiontech/dtle-docs-cn
- 文档库 github pages: actiontech.github.io/dtle-docs-cn
- QQ group: 852990221
- 网站：爱可生开源社区
- 开源社区微信公众号

联系我们

如果想获得 dtle 的商业支持, 您可以联系我们:

- 全国支持: 400-820-6580
- 华北地区: 86-13910506562, 汪先生
- 华南地区: 86-18503063188, 曹先生
- 华东地区: 86-18930110869, 梁先生
- 西南地区: 86-13540040119, 洪先生

概述

dtle (Data-Transformation-le) 是上海爱可生信息技术股份有限公司 开发并开源的 CDC 工具. 其功能特点是:

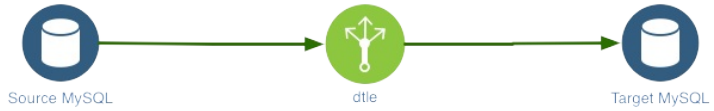
- 多种数据传输模式
  - 支持链路压缩
  - 支持同构传输和异构传输
  - 支持跨网络边际的传输
- 多种数据处理模式
  - 支持库/表/行级别 数据过滤
- 多种数据通道模式
  - 支持多对多的数据传输
  - 支持回环传输
- 多种源/目标端
  - 支持MySQL - ActionDB的数据传输
    - 支持MySQL - MySQL的数据传输
  - 支持MySQL - Kafka的数据传输
  - 支持Oracle - MySQL的数据传输
- 集群模式
  - 提供可靠的元数据存储
  - 可进行自动任务分配
  - 支持自动故障转移

1.0 兼容MySQL的单向复制/聚合/分散

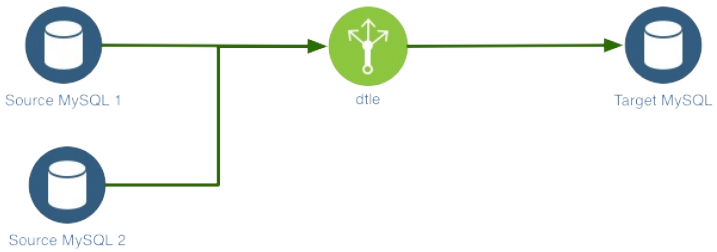
如下图, dtle 支持 兼容MySQL的 单向数据复制的常见场景如下:

- 按数据源/数据目标的映射关系划分
  - 支持1:1的复制
  - 支持n:1的数据汇聚, 将多个数据源的数据 聚合到 同一个数据目标
  - 支持1:n的数据拆分, 将一个数据源的数据 拆分到 多个数据目标
- 按网络类型划分
  - 支持网络内的数据传输
  - 支持跨网络边际的数据传输 (可使用 链路压缩/链路限流 等功能)
- 按集群规模划分
  - 可配置 单一dtle实例 处理 单一数据通道
  - 可配置 dtle集群 处理 多个数据通道

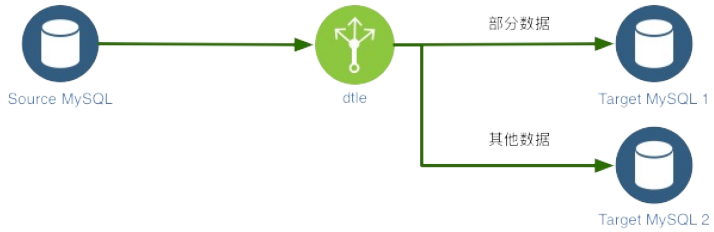
1-1复制



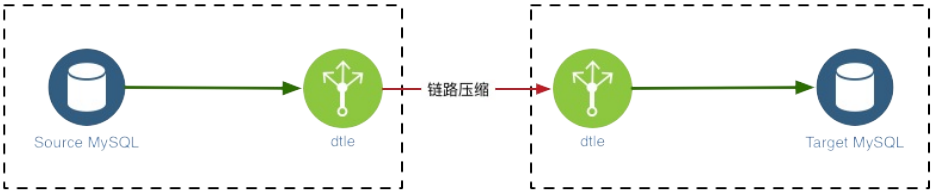
n-1汇聚



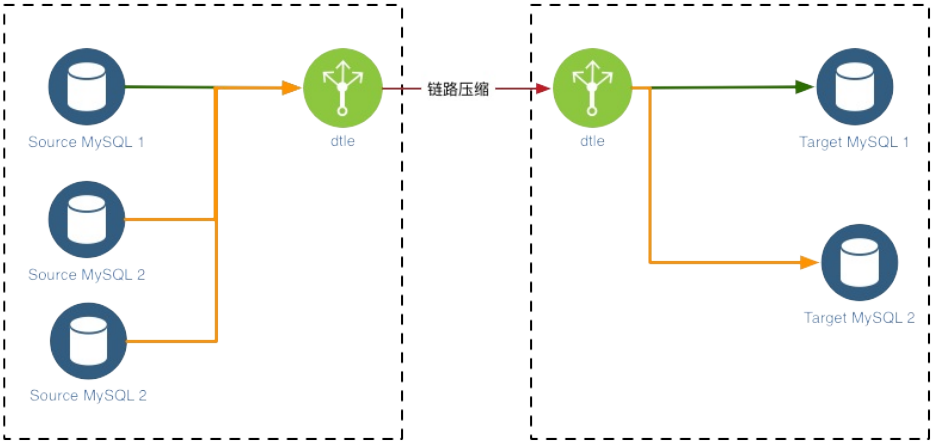
1-n拆分



跨网络边际的1-1复制

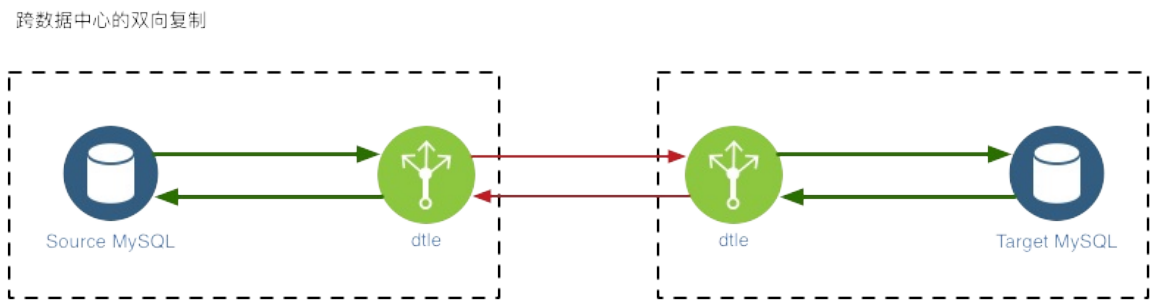


跨网络边际的多个复制通道



1.1 跨数据中心的双向复制

如下图, dtle支持MySQL间的双向复制, 其普遍场景是用于跨数据中心的数据双向同步.



其中:

- dtle 会对数据的回环状况进行判断, 不会重复传输同一事务.
- dtle 在传输过程中维持数据的事务性, 对于数据源的事务产生的数据, 在数据目标端是以相同的事务方式进行回放. 对于双写的场景, 目标端不会受到不完整的事务的影响.
- dtle 在数据链路上, 可使用压缩/限速等功能, 更适合于跨数据中心的场景.

1.2 公有云间的数据同步

dtle 可用于公有云间的数据同步, 可支持的部署方式同 1.0 和 1.1 两节介绍的方式. 其中的不同之处在于:

- dtle 可部署于公有云的云主机服务上
- dtle 对公有云上RDS服务给予的权限进行了适配, 不需高权限即可实现数据复制
- dtle 对公有云的 MySQL 非官方版 进行了适配 (如阿里云RDS会增加隐式主键列, 导致 binlog中的数据与表结构不符)

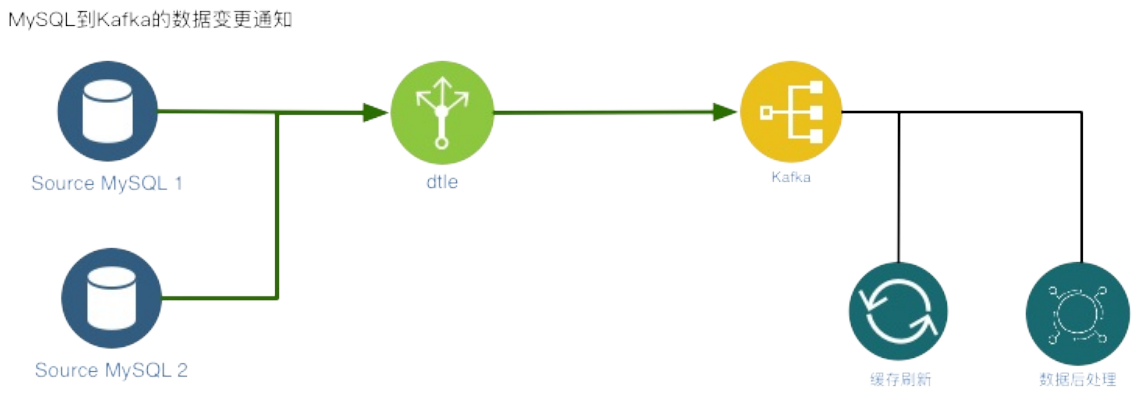
目前支持的公有云同步通道:

- 阿里云 -> 京东云

1.3 MySQL到Kafka的数据变更通知

如下图, dtle支持MySQL到Kafka的数据变更通知, 其普遍场景是:

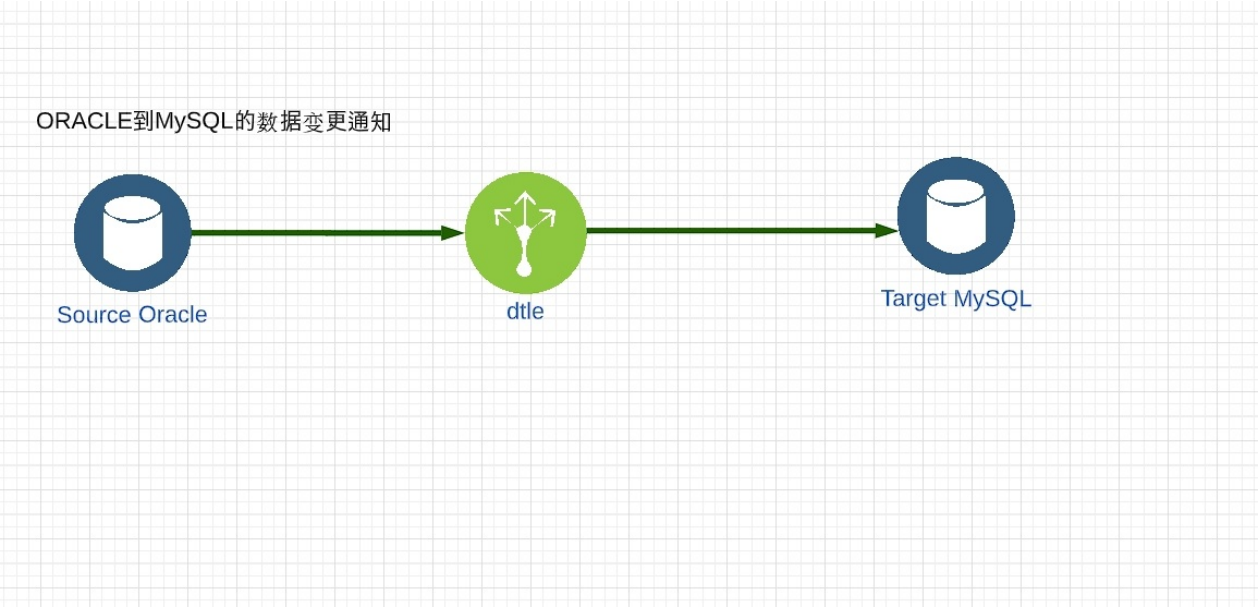
- 当数据变更时, 通知 缓存件 进行缓存刷新
- 当数据变更时, 通知 数据后处理件 进行数据扫描



1.4 Oracle到MySQL的数据同步

如下图, dtle支持Oracle到MySQL的数据同步通知, 其场景是:

- 当Oracle执行DDL时, 同步库/表结构到MySQL
- 当Oracle执行DML时，同步字段变更到MySQL



MySQL 的单向复制

以下步骤以docker容器的方式快速演示如何搭建MySQL的单向复制环境.

创建网络

```
docker network create dtle-net
```

创建源端/目标端 MySQL

```
docker run --name mysql-src -e MYSQL_ROOT_PASSWORD=pass -p 33061:3306 --network=dtle-net -d mysql:5.7 --gtid-mode=ON --enforce
-gtid-consistency=1 --log-bin=bin --server-id=1

docker run --name mysql-dst -e MYSQL_ROOT_PASSWORD=pass -p 33062:3306 --network=dtle-net -d mysql:5.7 --gtid-mode=ON --enforce
-gtid-consistency=1 --log-bin=bin --server-id=2
```

检查是否联通:

```
> mysql -h 127.0.0.1 -P 33061 -uroot -ppass -e "select @@version\G"
< *************************** 1. row ***************************
@@version: 5.7.23-log

> mysql -h 127.0.0.1 -P 33062 -uroot -ppass -e "select @@version\G"
< *************************** 1. row ***************************
@@version: 5.7.23-log
```

创建 dtle

```
docker run --name dtle-consul -p 8500:8500 --network=dtle-net -d consul:latest
docker run --name dtle -p 4646:4646 --network=dtle-net -d actiontech/dtle
# 如需要使用dtle 2.x HTTP API兼容层，则需要额外映射8190端口：-p 8190:8190
```

检查是否正常:

```
> curl -XGET "127.0.0.1:4646/v1/nodes" -s | jq
< [
    {
      "Address": "127.0.0.1",
      "Datacenter": "dc1",
      "Drivers": {
        "dtle": {
          "Attributes": {
            "driver.dtle": "true",
            "driver.dtle.version": "..."
          },
          "Detected": true,
          "Healthy": true,
        }
      },
      "ID": "65ff2f9a-a9fa-997c-cce0-9bc0b4f3396c",
      "Name": "nomad0",
      "Status": "ready",
    }
  ]
# （部分项目省略）
```

准备作业定义文件

准备文件job.json, 内容如下:

```
{
  "Job": {
    "ID": "dtle-demo",
    "Datacenters": ["dc1"],
    "TaskGroups": [{
        "Name": "src",
        "Tasks": [{
          "Name": "src",
          "Driver": "dtle",
          "Config": {
            "Gtid": "",
            "ReplicateDoDb": [{
              "TableSchema": "demo",
              "Tables": [{
                "TableName": "demo_tbl"
              }]
            }],
            "SrcConnectionConfig": {
              "Host": "mysql-src",
              "Port": 3306,
              "User": "root",
              "Password": "pass"
            },
            "DestConnectionConfig": {
              "Host": "mysql-dst",
              "Port": 3306,
              "User": "root",
              "Password": "pass"
            }
          }
        }]
      }, {
        "Name": "dest",
        "Tasks": [{
          "Name": "dest",
          "Driver": "dtle",
          "Config": {
            "DestType": "mysql"
          }
        }]
    }]
  }
}
```

其中定义了:

- 源端/目标端的连接字符串
- 要复制的表为 `demo.demo_tbl`
- GTID点位为空, 表示此复制是 全量+增量 的复制. 如只测试增量复制, 可指定合法的GTID

准备测试数据

可在源端准备提前建表 `demo.demo_tbl` , 并插入数据, 以体验全量复制过程. 也可不提前建表.

创建复制任务

```
> curl -XPOST "http://127.0.0.1:4646/v1/jobs" -d @job.json -s | jq
< {
  "EvalCreateIndex": 50,
  "EvalID": "a5e9c353-5eb9-243e-983d-bc096a93ddca",
  "Index": 50,
  "JobModifyIndex": 49,
  "KnownLeader": false,
  "LastContact": 0,
  "Warnings": ""
}
```

查看作业状态

```
> curl -XGET "http://127.0.0.1:4646/v1/job/dtle-demo" -s | jq '.Status'
< "running"
```

测试

此时可在源端对表 `demo.demo_tbl` 进行DDL/DML等各种操作, 查看目标端数据是否一致

HTTP API、nomad 命令行工具 和 Web界面

HTTP API

curl命令实际上是调用nomad agent端的HTTP接口，将本地的job.json提交到nomad agent端。

```
curl -XPOST "http://127.0.0.1:4646/v1/jobs" -d @job.json -s | jq
```

dtle rpm安装包提供了json和hcl格式的job样例。

jq

jq是一款格式化、提取json内容的工具。一般需使用Linux包管理器安装。

典型用法

```
# 格式化json内容:
some_command_print_json | jq

# 提取字段（Status）:
some_command_print_json | jq '.Status'
```

具体参考 https://stedolan.github.io/jq/tutorial/

nomad 命令行工具

此外还可以使用nomad命令行工具。nomad将命令行工具和agent端放在了同一个可执行文件中。

使用 nomad 命令行工具运行job, 使用hcl格式:

```
nomad job run -address="http://192.168.1.1:4646" job1.hcl
# 或
export NOMAD_ADDR="http://192.168.1.1:4646"
nomad job run job1.hcl
```

该用法本质上是对HTTP API的封装。

nomad Web 界面

浏览器访问 http://127.0.0.1:4646, 为 nomad Web 界面。可查看Jobs、Servers和Clients。

在Jobs界面，点击Run Job，可运行HCL或JSON格式的job。

consul

- nomad 本体使用consul进行多节点注册和发现
- dtle nomad 插件使用consul进行任务元数据储存

浏览器访问 http://127.0.0.1:4646，为 consul Web 界面。可查看KV中的Job进度（Gtid）。

或

```
curl -XGET "127.0.0.1:8500/v1/kv/dtle/aa/Gtid?raw
```

MySQL 的汇聚复制

以下步骤以docker容器的方式快速演示如何搭建MySQL的汇聚复制环境.

创建网络

```
docker network create dtle-net
```

创建源端**(2**个**)**和目标端**(1**个**) MySQL**

```
docker run --name mysql-src1 -e MYSQL_ROOT_PASSWORD=pass -p 33061:3306 --network=dtle-net -d mysql:5.7 --gtid-mode=ON --enforc
e-gtid-consistency=1 --log-bin=bin --server-id=1

docker run --name mysql-src2 -e MYSQL_ROOT_PASSWORD=pass -p 33062:3306 --network=dtle-net -d mysql:5.7 --gtid-mode=ON --enforc
e-gtid-consistency=1 --log-bin=bin --server-id=2

docker run --name mysql-dst -e MYSQL_ROOT_PASSWORD=pass -p 33063:3306 --network=dtle-net -d mysql:5.7 --gtid-mode=ON --enforce
-gtid-consistency=1 --log-bin=bin --server-id=3
```

检查是否联通:

```
> mysql -h 127.0.0.1 -P 33061 -uroot -ppass -e "select @@version\G"
< *************************** 1. row ***************************
@@version: 5.7.23-log

> mysql -h 127.0.0.1 -P 33062 -uroot -ppass -e "select @@version\G"
< *************************** 1. row ***************************
@@version: 5.7.23-log

> mysql -h 127.0.0.1 -P 33063 -uroot -ppass -e "select @@version\G"
< *************************** 1. row ***************************
@@version: 5.7.23-log
```

在源端**MySQL**中创建表结构**,** 获取**GTID**点位**,** 并插入数据

```
> mysql -h 127.0.0.1 -P 33061 -uroot -ppass -e "CREATE DATABASE demo; CREATE TABLE demo.demo_tbl(a int primary key)"
< ...

> mysql -h 127.0.0.1 -P 33061 -uroot -ppass -e "show master status\G" | grep "Executed_Gtid_Set"
< Executed_Gtid_Set: f6def853-cbaa-11e8-8aeb-0242ac120003:1-7

> mysql -h 127.0.0.1 -P 33061 -uroot -ppass -e "insert into demo.demo_tbl values(1),(2),(3)"
< ...

---

> mysql -h 127.0.0.1 -P 33062 -uroot -ppass -e "CREATE DATABASE demo; CREATE TABLE demo.demo_tbl(a int primary key)"
< ...

> mysql -h 127.0.0.1 -P 33062 -uroot -ppass -e "show master status\G" | grep "Executed_Gtid_Set"
< Executed_Gtid_Set: f74aacb5-cbaa-11e8-bdd1-0242ac120004:1-7


> mysql -h 127.0.0.1 -P 33062 -uroot -ppass -e "insert into demo.demo_tbl values(4),(5),(6)"
< ...
```

在目标端**MySQL**中创建表结构

```
> mysql -h 127.0.0.1 -P 33063 -uroot -ppass -e "CREATE DATABASE demo; CREATE TABLE demo.demo_tbl(a int primary key)"
< ...
```

创建 **dtle**

```
docker run --name dtle-consul -p 8500:8500 --network=dtle-net -d consul:latest
docker run --name dtle -p 4646:4646 --network=dtle-net -d actiontech/dtle
```

检查是否正常:

```
> curl -XGET "127.0.0.1:4646/v1/nodes" -s | jq
< [{...}]
```

准备作业定义文件

src1到**dst**的复制定义文件

准备src1_dst.json, 内容如下:

```
{
  "Job": {
    "ID": "dtle-demo-src1-dst",
    "Datacenters": ["dc1"],
    "TaskGroups": [{
        "Name": "src",
        "Tasks": [{
          "Name": "src",
          "Driver": "dtle",
          "Config": {
            "Gtid": "f6def853-cbaa-11e8-8aeb-0242ac120003:1-7",
            "ReplicateDoDb": [{
              "TableSchema": "demo",
              "Tables": [{
                "TableName": "demo_tbl"
              }]
            }],
            "SrcConnectionConfig": {
              "Host": "mysql-src1",
              "Port": 3306,
              "User": "root",
              "Password": "pass"
            },
            "DestConnectionConfig": {
              "Host": "mysql-dst",
              "Port": 3306,
              "User": "root",
              "Password": "pass"
            }
          }
        }]
      }, {
        "Name": "dest",
        "Tasks": [{
          "Name": "dest",
          "Driver": "dtle",
          "Config": {
            "DestType": "mysql"
          }
        }]
    }]
  }
}
```

其中定义了:

- 源端/目标端的连接字符串
- 要复制的表为 `demo.demo_tbl`
- GTID点位为 准备数据阶段 插入数据之前的src1上的GTID点位

src2到**dst**的复制定义文件

准备src2_dst.json, 内容如下:

```
{
  "Job": {
    "ID": "dtle-demo-src2-dst",
    "Datacenters": ["dc1"],
    "TaskGroups": [{
        "Name": "src",
        "Tasks": [{
          "Name": "src",
          "Driver": "dtle",
          "Config": {
            "Gtid": "f74aacb5-cbaa-11e8-bdd1-0242ac120004:1-7",
            "ReplicateDoDb": [{
              "TableSchema": "demo",
              "Tables": [{
                "TableName": "demo_tbl"
              }]
            }],
            "SrcConnectionConfig": {
              "Host": "mysql-src2",
              "Port": 3306,
              "User": "root",
              "Password": "pass"
            },
            "DestConnectionConfig": {
              "Host": "mysql-dst",
              "Port": 3306,
              "User": "root",
              "Password": "pass"
            }
          }
        }]
      }, {
        "Name": "dest",
        "Tasks": [{
          "Name": "dest",
          "Driver": "dtle",
          "Config": {
            "DestType": "mysql"
          }
        }]
    }]
  }
}
```

其中与 `src1_dst.json` 不同的是:

- 源端的连接字符串
- GTID点位为 准备数据阶段 插入数据之前的src2上的GTID点位

创建复制任务

```
> curl -XPOST "http://127.0.0.1:4646/v1/jobs" -d @src1_dst.json -s | jq
< {...}

> curl -XPOST "http://127.0.0.1:4646/v1/jobs" -d @src2_dst.json -s | jq
< {...}
```

查看作业ID和状态:

```
> curl -XGET "127.0.0.1:4646/v1/jobs" -s | jq '.[] | .ID, .Status'
< "dtle-demo-src1-dst"
"running"
"dtle-demo-src2-dst"
"running"
```

测试

在src1和src2中分别插入数据, 查看dst中的数据, 验证全量和增量的数据均存在

```
> mysql -h 127.0.0.1 -P 33061 -uroot -ppass -e "insert into demo.demo_tbl values(11)"
< ...

> mysql -h 127.0.0.1 -P 33062 -uroot -ppass -e "insert into demo.demo_tbl values(12)"
< ...

> mysql -h 127.0.0.1 -P 33063 -uroot -ppass -e "select * from demo.demo_tbl"
<
+----+
| a  |
+----+
|  1 |
|  2 |
|  3 |
|  4 |
|  5 |
|  6 |
| 11 |
| 12 |
+----+
```

MySQL 的数据分散

以下步骤以docker容器的方式快速演示如何搭建MySQL的数据分散环境. 数据分散复制, 将源表中的数据中, `主键<5` 的行复制到目标库1, `主键>=5` 的行复制到目标库2.

创建网络

```
docker network create dtle-net
```

创建源端(1个)和目标端(2个) MySQL

```
docker run --name mysql-src -e MYSQL_ROOT_PASSWORD=pass -p 33061:3306 --network=dtle-net -d mysql:5.7 --gtid-mode=ON --enforce
-gtid-consistency=1 --log-bin=bin --server-id=1

docker run --name mysql-dst1 -e MYSQL_ROOT_PASSWORD=pass -p 33062:3306 --network=dtle-net -d mysql:5.7 --gtid-mode=ON --enforc
e-gtid-consistency=1 --log-bin=bin --server-id=2

docker run --name mysql-dst2 -e MYSQL_ROOT_PASSWORD=pass -p 33063:3306 --network=dtle-net -d mysql:5.7 --gtid-mode=ON --enforc
e-gtid-consistency=1 --log-bin=bin --server-id=3
```

检查是否联通:

```
> mysql -h 127.0.0.1 -P 33061 -uroot -ppass -e "select @@version\G"
< *************************** 1. row ***************************
@@version: 5.7.23-log

> mysql -h 127.0.0.1 -P 33062 -uroot -ppass -e "select @@version\G"
< *************************** 1. row ***************************
@@version: 5.7.23-log

> mysql -h 127.0.0.1 -P 33063 -uroot -ppass -e "select @@version\G"
< *************************** 1. row ***************************
@@version: 5.7.23-log
```

在源端MySQL中创建表结构, 获取GTID点位, 并插入数据

```
> mysql -h 127.0.0.1 -P 33061 -uroot -ppass -e "CREATE DATABASE demo; CREATE TABLE demo.demo_tbl(a int primary key)"
< ...

> mysql -h 127.0.0.1 -P 33061 -uroot -ppass -e "show master status\G" | grep "Executed_Gtid_Set"
< Executed_Gtid_Set: 167dd42f-d076-11e8-8104-0242ac120003:1-7

> mysql -h 127.0.0.1 -P 33061 -uroot -ppass -e "insert into demo.demo_tbl values(1),(2),(3)"
< ...
```

在目标端MySQL中创建表结构

```
> mysql -h 127.0.0.1 -P 33062 -uroot -ppass -e "CREATE DATABASE demo; CREATE TABLE demo.demo_tbl(a int primary key)"
< ...

> mysql -h 127.0.0.1 -P 33063 -uroot -ppass -e "CREATE DATABASE demo; CREATE TABLE demo.demo_tbl(a int primary key)"
< ...
```

创建 **dtle**

```
docker run --name dtle-consul -p 8500:8500 --network=dtle-net -d consul:latest
docker run --name dtle -p 4646:4646 --network=dtle-net -d actiontech/dtle
```

检查是否正常:

```
> curl -XGET "127.0.0.1:4646/v1/nodes" -s | jq
< [{...}]
```

准备作业定义文件

src到**dst1**的复制定义文件

准备src_dst1.json, 内容如下:

```
{
  "Job": {
    "ID": "dtle-demo-src-dst1",
    "Datacenters": ["dc1"],
    "TaskGroups": [{
        "Name": "src",
        "Tasks": [{
          "Name": "src",
          "Driver": "dtle",
          "Config": {
            "Gtid": "",
            "ReplicateDoDb": [{
              "TableSchema": "demo",
              "Tables": [{
                "TableName": "demo_tbl",
                "Where":"a<5"
              }]
            }],
            "SrcConnectionConfig": {
              "Host": "mysql-src",
              "Port": 3306,
              "User": "root",
              "Password": "pass"
            },
            "DestConnectionConfig": {
              "Host": "mysql-dst1",
              "Port": 3306,
              "User": "root",
              "Password": "pass"
            }
          }
        }]
      }, {
        "Name": "dest",
        "Tasks": [{
          "Name": "dest",
          "Driver": "dtle",
          "Config": {
            "DestType": "mysql"
          }
        }]
    }]
  }
}
```

其中定义了:

- 源端/目标端的连接字符串

- 要复制的表为 `demo.demo_tbl`
- `demo_tbl` 的复制数据条件为 `a<5`

src到**dst2**的复制定义文件

准备src_dst2.json, 内容如下:

```
{
  "Job": {
    "ID": "dtle-demo-src-dst2",
    "Datacenters": ["dc1"],
    "TaskGroups": [{
        "Name": "src",
        "Tasks": [{
          "Name": "src",
          "Driver": "dtle",
          "Config": {
            "Gtid": "",
            "ReplicateDoDb": [{
              "TableSchema": "demo",
              "Tables": [{
                "TableName": "demo_tbl",
                "Where":"a>=5"
              }]
            }],
            "SrcConnectionConfig": {
              "Host": "mysql-src",
              "Port": 3306,
              "User": "root",
              "Password": "pass"
            },
            "DestConnectionConfig": {
              "Host": "mysql-dst2",
              "Port": 3306,
              "User": "root",
              "Password": "pass"
            }
          }
        }]
      }, {
        "Name": "dest",
        "Tasks": [{
          "Name": "dest",
          "Driver": "dtle",
          "Config": {
            "DestType": "mysql"
          }
        }]
    }]
  }
}
```

其中定义了:

- 源端/目标端的连接字符串
- 要复制的表为 `demo.demo_tbl`
- `demo_tbl` 的复制数据条件为 `a>=5`

其中与 `src1_dst.json` 不同的是:

- 源端的连接字符串
- `demo_tbl` 的复制数据条件

创建复制任务

```
> curl -XPOST "http://127.0.0.1:4646/v1/jobs" -d @src_dst1.json -s | jq
< {...}

> curl -XPOST "http://127.0.0.1:4646/v1/jobs" -d @src_dst2.json -s | jq
< {...}
```

查看作业ID和状态:

```
> curl -XGET "127.0.0.1:4646/v1/jobs" -s | jq '.[] | .ID, .Status'
< "dtle-demo-src-dst1"
"running"
"dtle-demo-src-dst2"
"running"
```

测试

在src中插入数据, 查看dst1/dst2中的数据, 验证全量和增量的数据均存在

```
> mysql -h 127.0.0.1 -P 33061 -uroot -ppass -e "insert into demo.demo_tbl values(0),(10)"
< ...


> mysql -h 127.0.0.1 -P 33062 -uroot -ppass -e "select * from demo.demo_tbl"
<
+----+
| a  |
+----+
|  0 |
|  1 |
|  2 |
|  3 |
|  4 |
+----+

> mysql -h 127.0.0.1 -P 33063 -uroot -ppass -e "select * from demo.demo_tbl"
<
+----+
| a  |
+----+
|  5 |
|  6 |
|  7 |
|  8 |
|  9 |
| 10 |
+----+
```

MySQL的跨数据中心的双向复制

以下步骤以docker容器的方式快速演示如何搭建MySQL的跨数据中心的双向复制.

创建两个网络

```
docker network create dtle-net-dc1
docker network create dtle-net-dc2
```

在两个网络中分别创建**MySQL**

```
docker run --name mysql-dc1 -e MYSQL_ROOT_PASSWORD=pass -p 33061:3306 --network=dtle-net-dc1 -d mysql:5.7 --gtid-mode=ON --enf
orce-gtid-consistency=1 --log-bin=bin --server-id=1

docker run --name mysql-dc2 -e MYSQL_ROOT_PASSWORD=pass -p 33062:3306 --network=dtle-net-dc2 -d mysql:5.7 --gtid-mode=ON --enf
orce-gtid-consistency=1 --log-bin=bin --server-id=2
```

检查MySQL是否启动成功:

```
> mysql -h 127.0.0.1 -P 33061 -uroot -ppass -e "select @@version\G"
< *************************** 1. row ***************************
@@version: 5.7.23-log

> mysql -h 127.0.0.1 -P 33062 -uroot -ppass -e "select @@version\G"
< *************************** 1. row ***************************
@@version: 5.7.23-log
```

在两个网络中分别创建**dtle**

```
docker run --name dtle-consul -p 8500:8500 --network=dtle-net-dc1 -d consul:latest
docker run --name dtle-dc1 -p 4646:4646 --network=dtle-net-dc1 -d actiontech/dtle

# dtle-dc2 will work as a client only. No need to start consul-dc2.
docker run --name dtle-dc2 -p 5646:4646 --network=dtle-net-dc2 -d actiontech/dtle
```

将两个**dtle**通过公网连通

```
docker network create dtle-net-public
docker network connect dtle-net-public dtle-dc1
docker network connect dtle-net-public dtle-consul
docker network connect dtle-net-public dtle-dc2
```

修改**dtle**的配置

修改容器**dtle-dc1**内的配置并重启

修改容器dtle-dc1内的配置并重启:

```
docker exec -u root -it dtle-dc1 vi /dtle/etc/dtle/nomad.hcl
...
docker exec -u root -it dtle-dc1 rm -rf /dtle/var/lib/nomad
docker restart dtle-dc1
```

配置 `/dtle/etc/dtle/nomad.hcl` 修改的内容如下:

```
name = "nomad1" # rename for each node

# ... (省略未更改项目)

bind_addr = "172.22.0.2"
advertise {
  http = "172.22.0.2"
  rpc  = "172.22.0.2"
  serf = "172.22.0.2"
}

plugin "dtle" {
  config {
    nats_bind = "172.22.0.2:8193"
    nats_advertise = "172.22.0.2:8193"
    nomad_addr = "172.22.0.2:4646"
    # ...
  }
}
```

其中:

- 由于dtle-dc1容器存在两个网络 (与MySQL通信的内网 `dtle-net-dc1` , 和与dtle-dc2通信的公网 `dtle-net-public`), 需要指定 `bind_addr` 和 `advertise.rpc` 为本机的 `dtle-net-public` 的网络地址, 此处为 `172.22.0.2`

修改容器**dtle-dc2**内的配置并重启

修改容器dtle-dc2内的配置并重启:

```
docker exec -u root -it dtle-dc2 vi /dtle/etc/dtle/nomad.hcl
...
docker exec -u root -it dtle-dc2 rm -rf /dtle/var/lib/nomad
docker restart dtle-dc2
```

配置 `/dtle/etc/dtle/nomad.hcl` 修改的内容如下:

```
name = "nomad2" # rename for each node

# ... (省略未更改项目)

bind_addr = "172.22.0.3"
advertise {
  http = "172.22.0.3"
  rpc  = "172.22.0.3"
  serf = "172.22.0.3"
}

server {
  # 重要!
  # 只有 dtle-dc1 作为server. dtle-dc2 仅作为 client.
  enabled          = false
}

plugin "dtle" {
  config {
    nats_bind = "172.22.0.3:8193"
    nats_advertise = "172.22.0.3:8193"
    nomad_addr = "172.22.0.3:4646"
    # ...
  }
}
```

其中:

- 由于dtle-dc2容器存在两个网络 (与MySQL通信的内网 `dtle-net-dc2` , 和与dtle-dc1通信的公网 `dtle-net-public`), 需要指定 `bind_addr` 和 `advertise.rpc` 为本机的 `dtle-net-public` 的网络地址, 此处为 `172.22.0.3`

检查是否正常

```
> curl -XGET "127.0.0.1:4646/v1/nodes" -s | jq
```

或查看Web UI，确定我们构建了一个 1 server 2 client 的nomad部署。

配置**dc1**到**dc2**的复制

获取mysql-dc1的GTID:

```
> mysql -h 127.0.0.1 -P 33061 -uroot -ppass -e "show master status\G" | grep "Executed_Gtid_Set"
< Executed_Gtid_Set: 41f102d4-d29f-11e8-8de7-0242ac130002:1-5
```

准备文件job-dc1-dc2.json, 内容如下:

```
{
  "Job": {
    "ID":"dtle-demo-dc1-2-dc2",
    "Datacenters": ["dc1"],
    "TaskGroups": [{
        "Name": "src",
        "Tasks": [{
          "Name": "src",
          "Driver": "dtle",
          "Constraints": [{
            "LTarget": "${node.unique.name}",
            "RTarget": "nomad1",
            "Operand": "="
          }],
          "Config": {
            "Gtid":"41f102d4-d29f-11e8-8de7-0242ac130002:1-5",
            "ReplicateDoDb": [{
              "TableSchema": "demo",
              "Tables": [{
                "TableName": "demo_tbl"
              }]
            }],
            "SrcConnectionConfig": {
              "Host":"mysql-dc1",
              "Port": 3306,
              "User": "root",
              "Password": "pass"
            },
            "DestConnectionConfig": {
              "Host":"mysql-dc2",
              "Port": 3306,
              "User": "root",
              "Password": "pass"
            }
          }
        }]
      }, {
        "Name": "dest",
        "Tasks": [{
          "Name": "dest",
          "Driver": "dtle",
          "Constraints": [{
            "LTarget": "${node.unique.name}",
            "RTarget": "nomad2",
            "Operand": "="
          }],
          "Config": {
            "DestType": "mysql"
          }
        }]
    }]
  }
}
```

其中定义了:

- 源端/目标端的连接字符串
- 要复制的表为 `demo.demo_tbl`
- GTID点位, 表示此复制是 增量复制 (双向复制 只支持增量复制)
- 源任务(src)配置在dc1的dtle节点上执行 (通过 Constraints 指定)
- 目标任务(dest)配置在dc2的dtle节点上执行 (通过 Constraints 指定)

创建**dc1**到**dc2**的复制任务

```
> curl -XPOST "http://127.0.0.1:4646/v1/jobs" -d @job-dc1-dc2.json -s | jq
```

查看作业状态

```
> curl -XGET "127.0.0.1:4646/v1/job/dtle-demo-dc1-2-dc2" -s | jq '.Status'
< "running"
```

配置**dc2**到**dc1**的复制

获取mysql-dc2的GTID:

```
> mysql -h 127.0.0.1 -P 33062 -uroot -ppass -e "show master status\G"
< *************************** 1. row ***************************
             File: bin.000003
         Position: 537
     Binlog_Do_DB:
 Binlog_Ignore_DB:
Executed_Gtid_Set: 41f102d4-d29f-11e8-8de7-0242ac130002:6-7,
42158e2f-d29f-11e8-b322-0242ac150002:1-5
```

准备文件job-dc2-dc1.json, 内容如下:

```
{
  "Job": {
    "ID":"dtle-demo-dc2-2-dc1",
    "Datacenters": ["dc1"],
    "TaskGroups": [{
        "Name": "src",
        "Tasks": [{
          "Name": "src",
          "Driver": "dtle",
          "Constraints": [{
            "LTarget": "${node.unique.name}",
            "RTarget": "nomad2",
            "Operand": "="
          }],
          "Config": {
            "Gtid":"41f102d4-d29f-11e8-8de7-0242ac130002:6-7,42158e2f-d29f-11e8-b322-0242ac150002:1-5",
            "ReplicateDoDb": [{
              "TableSchema": "demo",
              "Tables": [{
                "TableName": "demo_tbl"
              }]
            }],
            "SrcConnectionConfig": {
              "Host":"mysql-dc2",
              "Port": 3306,
              "User": "root",
              "Password": "pass"
            },
            "DestConnectionConfig": {
              "Host":"mysql-dc1",
              "Port": 3306,
              "User": "root",
              "Password": "pass"
            }
          }
        }]
      }, {
        "Name": "dest",
        "Tasks": [{
          "Name": "dest",
          "Driver": "dtle",
          "Constraints": [{
            "LTarget": "${node.unique.name}",
            "RTarget": "nomad1",
            "Operand": "="
          }],
          "Config": {
            "DestType": "mysql"
          }
        }]
    }]
  }
}
```

其中与 dc1到dc2的复制任务 不同的是:

- 源端/目标端的连接字符串
- GTID点位
- 源任务(src)配置在dc2的dtle节点上执行
- 目标任务(dest)配置在dc1的dtle节点上执行

创建**dc2**到**dc1**的复制任务

```
> curl -XPOST "http://127.0.0.1:4646/v1/jobs" -d @job-dc2-dc1.json -s | jq
```

查看作业状态

```
> curl -XGET "127.0.0.1:4646/v1/job/dtle-demo-dc2-2-dc1" -s | jq '.Status'
< "running"
```

测试

此时可在任一端对表 `demo.demo_tbl` 进行DDL/DML等各种操作, 查看目标端数据是否一致

数据冲突

dtle不检测数据冲突。如果回放报错（如应数据冲突导致update了不存在的列），则job报错。

其中，DML insert使用replace回放，故insert冲突时，效果是last-win。

建议由业务端确保数据不会冲突。

阿里云到京东云的**MySQL**复制

以下步骤演示如何搭建从阿里云RDS到京东云RDS的MySQL复制.

检查阿里云**RDS**的环境

MySQL版本为5.7.18

检查权限:

```
mysql> select user();
+---------------------+
| user()              |
+---------------------+
| root@180.169.60.146 |
+---------------------+
1 row in set (0.02 sec)


mysql> show grants for 'root'@'%' \G
*************************** 1. row ***************************
Grants for root@%: GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, RELOAD, PROCESS, REFERENCES, INDEX, ALTER, CREATE TEMPO
RARY TABLES, LOCK TABLES, EXECUTE, REPLICATION SLAVE, REPLICATION CLIENT, CREATE VIEW, SHOW VIEW, CREATE ROUTINE, ALTER ROUTIN
E, CREATE USER, EVENT, TRIGGER ON *.* TO 'root'@'%' WITH GRANT OPTION
1 row in set (0.02 sec)
```

检查京东云**RDS**的环境

MySQL版本为5.7.21

注意**:** 京东云**RDS**实例的用户权限是以**schema**为基础的**.** 需要在创建迁移**job**前，通过京东云**RDS**为该**MySQL**实例创建两个**schema: dtle(**存储**dtle**元数据**)** 和 迁移的目标库

```
mysql> select user();
+---------------------------+
| user()                    |
+---------------------------+
| actiontech@180.169.60.146 |
+---------------------------+
1 row in set (0.00 sec)


mysql> show grants for 'actiontech'@'%';
+-----------------------------------------------------------+
| Grants for actiontech@%                                   |
+-----------------------------------------------------------+
| GRANT USAGE ON *.* TO 'actiontech'@'%'                    |
| GRANT ALL PRIVILEGES ON `actiontech`.* TO 'actiontech'@'%' |
+-----------------------------------------------------------+
2 rows in set (0.00 sec)
```

申请京东云**ECS**

需要申请京东云ECS, 用于

示例主机IP为 `192.168.0.17` , 规格是1c4g40g

安装并配置**dtle**

安装dtle:

```
rpm -ivh dtle-xxx.rpm
```

配置 `/etc/dtle/nomad.hcl` :

```
# 省略未修改配置
bind_addr = "192.168.0.17"

advertise {
  http = "192.168.0.17"
  rpc = "192.168.0.17"
  serf = "192.168.0.17"
}
```

启动dtle:

```
systemctl start dtle-consul dtle-nomad
```

增加复制任务

复制配置文件 job.json 内容如下:

```
{
  "Job": {
    "ID":"ali-jd-demo",
    "Datacenters": ["dc1"],
    "TaskGroups": [{
        "Name": "src",
        "Tasks": [{
          "Name": "src",
          "Driver": "dtle",
          "Config": {
            "Gtid":"",
            "ReplicateDoDb": [{
              "TableSchema": "actiontech",
              "Tables": []
            }],
            "SrcConnectionConfig": {
              "Host":"rm-xxxx.mysql.rds.aliyuncs.com",
              "Port":"3306",
              "User":"root",
              "Password":"Acti0ntech"
            },
            "DestConnectionConfig": {
              "Host":"mysql-cn-east-2-yyyy.public.jcloud.com",
              "Port":"3306",
              "User":"actiontech",
              "Password":"Acti0ntech"
            }
          }
        }]
      }, {
        "Name": "dest",
        "Tasks": [{
          "Name": "dest",
          "Driver": "dtle",
          "Config": {
            "DestType": "mysql"
          }
        }]
    }]
  }
}
```

向dtle发布任务:

```
curl -XPOST "192.168.0.17:4646/v1/jobs" -d @job.json
```

检查任务运行状态:

```
curl -XGET "192.168.0.17:4646/v1/job/ali-jd-demo" -s | jq '.Status'
```

其他

如要使用链路压缩等功能, 可参照MySQL的跨数据中心的双向复制

consul 默认只能从从本机查询。若要从外部访问KV，请更改/etc/dtle/consul.hcl中的 `client_addr` 。 并相应配置nomad.hcl。

MySQL到**Kafka**的数据变更通知

以下步骤以docker容器的方式快速演示如何搭建MySQL的单向复制环境.

创建网络

```
docker network create dtle-net
```

创建源端 **MySQL**

```
docker run --name mysql-src -e MYSQL_ROOT_PASSWORD=pass -p 33061:3306 --network=dtle-net -d mysql:5.7 --gtid-mode=ON --enforce
-gtid-consistency=1 --log-bin=bin --server-id=1
```

检查是否联通:

```
> mysql -h 127.0.0.1 -P 33061 -uroot -ppass -e "select @@version\G"
< *************************** 1. row ***************************
@@version: 5.7.23-log
```

创建源端表结构

```
> mysql -h 127.0.0.1 -P 33061 -uroot -ppass -e "CREATE DATABASE demo; CREATE TABLE demo.demo_tbl(a int primary key)"
```

创建目标端 **Kafka**

```
docker run --name kafka-zookeeper -p 2181:2181 -e ALLOW_ANONYMOUS_LOGIN=yes --network=dtle-net -d bitnami/zookeeper
docker run --name kafka-dst -p 9092:9092 -e KAFKA_ZOOKEEPER_CONNECT=kafka-zookeeper:2181 -e ALLOW_PLAINTEXT_LISTENER=yes --net
work=dtle-net -d bitnami/kafka
```

检查是否联通:

```
> docker run -it --rm \
    --network dtle-net \
    -e KAFKA_ZOOKEEPER_CONNECT=kafka-zookeeper:2181 \
    bitnami/kafka:latest kafka-topics.sh --list  --zookeeper kafka-zookeeper:2181
< Welcome to the Bitnami kafka container
Subscribe to project updates by watching https://github.com/bitnami/bitnami-docker-kafka
Submit issues and feature requests at https://github.com/bitnami/bitnami-docker-kafka/issues
```

创建 **dtle**

```
docker run --name dtle-consul -p 8500:8500 --network=dtle-net -d consul:latest
docker run --name dtle -p 4646:4646 --network=dtle-net -d actiontech/dtle
```

检查是否正常:

```
> curl -XGET "127.0.0.1:4646/v1/nodes" -s | jq
< [{...}]
```

准备作业定义文件

准备文件job.json, 内容如下:

```
{
  "Job": {
    "ID": "dtle-demo",
    "Datacenters": ["dc1"],
    "TaskGroups": [{
        "Name": "src",
        "Tasks": [{
          "Name": "src",
          "Driver": "dtle",
          "Config": {
            "Gtid": "",
            "ReplicateDoDb": [{
              "TableSchema": "demo",
              "Tables": [{
                "TableName": "demo_tbl"
              }]
            }],
            "SrcConnectionConfig": {
              "Host": "mysql-src",
              "Port": 3306,
              "User": "root",
              "Password": "pass"
            },
            "KafkaConfig": {
              "Topic": "demo-topic",
              "Brokers": ["kafka-dst:9092"],
              "Converter": "json"
            }
          }
        }]
      }, {
        "Name": "dest",
        "Tasks": [{
          "Name": "dest",
          "Driver": "dtle",
          "Config": {
            "DestType": "kafka"
          }
        }]
    }]
  }
}
```

其中定义了:

- 源端 MySQL 的连接字符串
- 目标端 Kafka 的 broker 访问地址
- 要复制的表为 `demo.demo_tbl`
- GTID点位为空, 表示此复制是 全量+增量 的复制. 如只测试增量复制, 可指定合法的GTID

创建复制任务

```
> curl -XPOST "http://127.0.0.1:4646/v1/jobs" -d @job.json -s | jq
< {...}
```

查看作业状态:

```
> curl -XGET "127.0.0.1:4646/v1/job/dtle-demo" -s | jq '.Status'
< "running"
```

测试

在源端写入数据:

```
> mysql -h 127.0.0.1 -P 33061 -uroot -ppass -e "INSERT INTO demo.demo_tbl values(1)"
...
```

验证相关的topic存在:

```
> docker run -it --rm \
    --network dtle-net \
    -e KAFKA_ZOOKEEPER_CONNECT=kafka-zookeeper:2181 \
    bitnami/kafka:latest kafka-topics.sh --list  --zookeeper kafka-zookeeper:2181
< Welcome to the Bitnami kafka container
Subscribe to project updates by watching https://github.com/bitnami/bitnami-docker-kafka
Submit issues and feature requests at https://github.com/bitnami/bitnami-docker-kafka/issues

demo-topic.demo.demo_tbl
```

验证数据:

```
> docker run -it --rm \
    --network dtle-net \
    -e KAFKA_ZOOKEEPER_CONNECT=kafka-zookeeper:2181 \
    bitnami/kafka:latest kafka-console-consumer.sh --bootstrap-server kafka-dst:9092 --topic demo-topic.demo.demo_tbl --from-b
eginning
< ...
{"schema":{"type":"struct","optional":false,"fields":[{"type":"struct","optional":true,"field":"before","fields":[{"type":"int
32","optional":false,"field":"a"}],"name":"demo-topic.demo.demo_tbl.Value"},{"type":"struct","optional":true,"field":"after","
fields":[{"type":"int32","optional":false,"field":"a"}],"name":"demo-topic.demo.demo_tbl.Value"},{"type":"struct","optional":f
alse,"field":"source","fields":[{"type":"string","optional":true,"field":"version"},{"type":"string","optional":false,"field":
"name"},{"type":"int64","optional":false,"field":"server_id"},{"type":"int64","optional":false,"field":"ts_sec"},{"type":"stri
ng","optional":true,"field":"gtid"},{"type":"string","optional":false,"field":"file"},{"type":"int64","optional":false,"field"
:"pos"},{"type":"int32","optional":false,"field":"row"},{"type":"boolean","optional":true,"field":"snapshot"},{"type":"int64",
"optional":true,"field":"thread"},{"type":"string","optional":true,"field":"db"},{"type":"string","optional":true,"field":"tab
le"}],"name":"io.debezium.connector.mysql.Source"},{"type":"string","optional":false,"field":"op"},{"type":"int64","optional":
true,"field":"ts_ms"}],"name":"demo-topic.demo.demo_tbl.Envelope","version":1},"payload":{"before":null,"after":{"a":11},"sour
ce":{"version":"0.0.1","name":"demo-topic","server_id":0,"ts_sec":0,"gtid":null,"file":"","pos":0,"row":1,"snapshot":true,"thr
ead":null,"db":"demo","table":"demo_tbl"},"op":"c","ts_ms":1539760682507}}
```

此时可在源端对表 `demo.demo_tbl` 进行DDL/DML等各种操作, 查看目标端数据是否一致

关于Kafka的消息格式, 参看5.3 Kafka 消息格式

Oracle到**MySQL**的数据同步

以下步骤以docker容器的方式快速演示如何搭建Oracle到MySQL的单向复制环境.

创建网络

```
docker network create dtle-net
```

创建源端 **Oracle**

```
  # 启动oracle镜像
    docker run -it -d -p 1521:1521 --name oracle-src --network=dtle-net  -e ORACLE_ALLOW_REMOTE=true  wnameless/oracle-xe-11g-
r2

  # 环境配置并启动oracle
    docker exec -it oracle-src bash
    mkdir /u01/app/oracle/oradata/archive_log
    chown oracle /u01/app/oracle/oradata/archive_log

    export ORACLE_HOME=/u01/app/oracle/product/11.2.0/xe
    export PATH=$ORACLE_HOME/bin:$PATH
    export ORACLE_SID=XE

    service oracle-xe start

  # 设置同步配置
    sqlplus SYS/oracle AS SYSDBA
    alter system set log_archive_dest_1='location=/u01/app/oracle/oradata/archive_log' scope=spfile;
    alter system set db_recovery_file_dest_size = 10G;

    shutdown immediate;
    startup mount;
    alter database add logfile group 3 '/u01/app/oracle/fast_recovery_area/XE/onlinelog/redo01.log' size 500m;
    alter database add logfile group 4 '/u01/app/oracle/fast_recovery_area/XE/onlinelog/redo02.log' size 500m;
    alter database add logfile group 5 '/u01/app/oracle/fast_recovery_area/XE/onlinelog/redo03.log' size 500m;
    alter database archivelog;
    alter database add supplemental log data (all) columns;
    alter database open;

  # 创建同步账号
    create role roma_logminer_privs;
    grant create session,execute_catalog_role,select any transaction,select_catalog_role,select any dictionary to roma_logmine
r_privs;
    grant select on SYSTEM.LOGMNR_COL$ to roma_logminer_privs;
    grant select on SYSTEM.LOGMNR_OBJ$ to roma_logminer_privs;
    grant select on SYSTEM.LOGMNR_USER$ to roma_logminer_privs;
    grant select on SYSTEM.LOGMNR_UID$ to roma_logminer_privs;
    create user roma_logminer identified by oracle default tablespace users;
    grant roma_logminer_privs to roma_logminer;
    alter user roma_logminer quota unlimited on users;
```

创建目标端 **MySQL**

```
docker run --name mysql-dst -e MYSQL_ROOT_PASSWORD=pass -p 33061:3306 --network=dtle-net -d mysql:5.7 --gtid-mode=ON --enforce
-gtid-consistency=1 --log-bin=bin --server-id=1
```

检查是否联通:

```
> mysql -h 127.0.0.1 -P 33061 -uroot -ppass -e "select @@version\G"
< *************************** 1. row ***************************
@@version: 5.7.23-log
```

创建 **dtle**

```
docker run --name dtle-consul -p 8500:8500 --network=dtle-net -d consul:latest
docker run --name dtle -p 4646:4646 --network=dtle-net -d actiontech/dtle
# 如需要使用dtle 2.x HTTP API兼容层，则需要额外映射8190端口：-p 8190:8190
```

检查是否正常:

```
> curl -XGET "127.0.0.1:4646/v1/nodes" -s | jq
< [
    {
      "Address": "127.0.0.1",
      "Datacenter": "dc1",
      "Drivers": {
        "dtle": {
          "Attributes": {
            "driver.dtle": "true",
            "driver.dtle.version": "..."
          },
          "Detected": true,
          "Healthy": true,
        }
      },
      "ID": "65ff2f9a-a9fa-997c-cce0-9bc0b4f3396c",
      "Name": "nomad0",
      "Status": "ready",
    }
  ]
# （部分项目省略）
```

准备作业定义文件

准备文件job.json, 内容如下:

```
{
  "Job": {
    "ID": "dtle-demo",
    "Datacenters": ["dc1"],
    "TaskGroups": [{
      "Name": "src",
      "Tasks": [{
        "Name": "src",
        "Driver": "dtle",
        "Config": {
          "ReplicateDoDb": [{
            "TableSchema": "TEST",
            "Tables": [{
              "TableName": "t1"
            }]
          }],
          "SrcOracleConfig": {
            "User":        "roma_logminer",
            "Password":    "oracle",
            "Host":        "oracle-src",
            "Port":        1521,
            "ServiceName": "XE",
            "Scn": 0
          },
          "DestConnectionConfig": {
            "Host": "mysql-dst",
            "Port": 3306,
            "User": "root",
            "Password": "pass"
          }
        }
      }]
    }, {
      "Name": "dest",
      "Tasks": [{
        "Name": "dest",
        "Driver": "dtle",
        "Config": {
          "DestType": "mysql"
        }
    }]
   }]
  }
}
```

其中定义了:

- 源端 Oracle 的连接配置
- 目标端 MySQL 的连接配置
- 要复制的表为 `TEST.t1`
- SCN点位为0, 表示此复制是从任务启动时间点开始复制. 如需测试指定位置增量复制, 可指定合法的SCN

创建复制任务

```
> curl -XPOST "http://127.0.0.1:4646/v1/jobs" -d @job.json -s | jq
< {...}
```

查看作业状态:

```
> curl -XGET "127.0.0.1:4646/v1/job/dtle-demo" -s | jq '.Status'
< "running"
```

测试

在源端写入数据:

```
sqlplus SYS/oracle AS SYSDBA
create user TEST identified by oracle;
grant connect,resource to TEST;
create table TEST."t1" (id int,name varchar(255));
insert into TEST."t1" values(1,'ryan');
commit;
```

验证目标端数据

```
查看目标端数据是否一致
```

字段映射关系参看 5.4 Oracle MySQL 字段映射

多**server**部署配置

nomad可以配置成

- 单server，单client
- 单server，多client
- 多server，多client

其中

- server管理job数据
- server数量为奇数，一般使用1或3个，不超过5个。
- client（运行dtle插件）执行job
- client数量任意
- server和client可运行于同一进程，也可单独启动server或client

需另外运行consul，用于

- nomad 服务发现（多节点自动注册）
- dtle 保存运行信息

一般每个nomad server搭配一个consul server，两者运行于同一台主机。

下面描述 多**server**多**client**配置 。

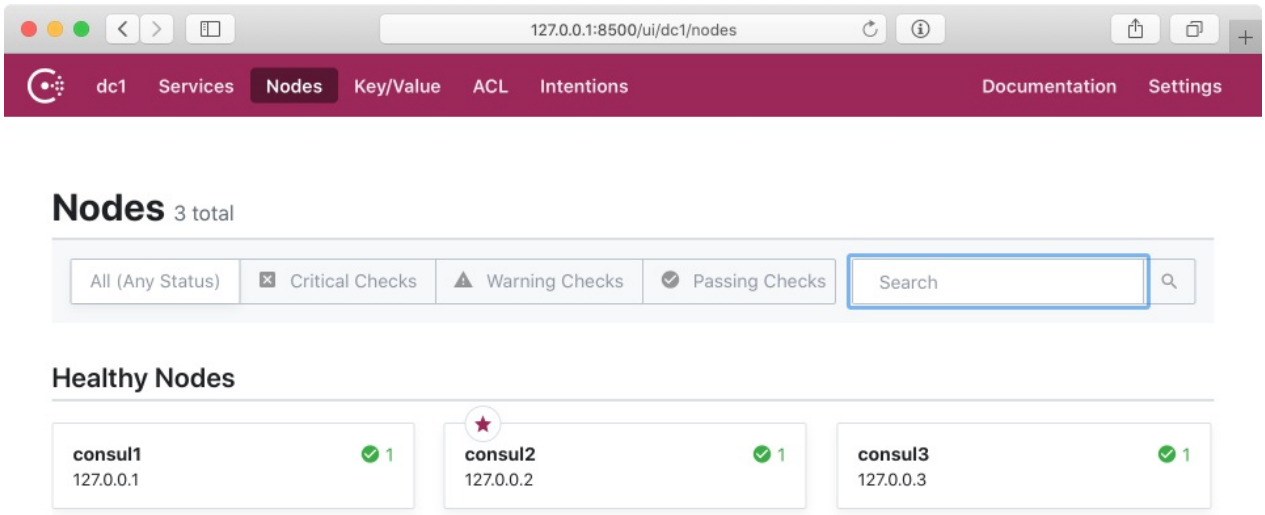
consul配置

修改 /etc/dtle/consul.hcl

```
# Rename for each node
node_name = "consul1"

# 配置IP

# Address that should be bound to for internal cluster communications
bind_addr = "0.0.0.0"
# Address to which Consul will bind client interfaces, including the HTTP and DNS servers
client_addr = "127.0.0.1"
advertise_addr = "127.0.0.1"

# ... 省略未更改项

bootstrap_expect = 3
retry_join = ["127.0.0.1", "127.0.0.2", "127.0.0.3"] # will use default serf port
```

为另外两个节点也做出修改。

全部启动后，从Web UI中可以看出，组成了3节点consul，其中一个为Leader。

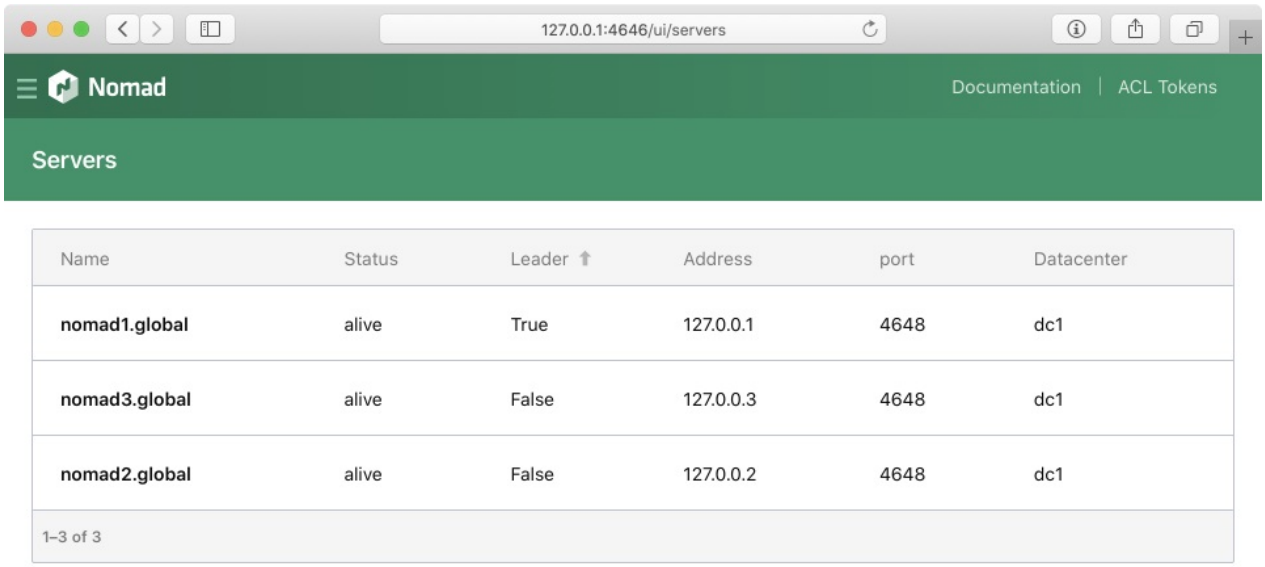


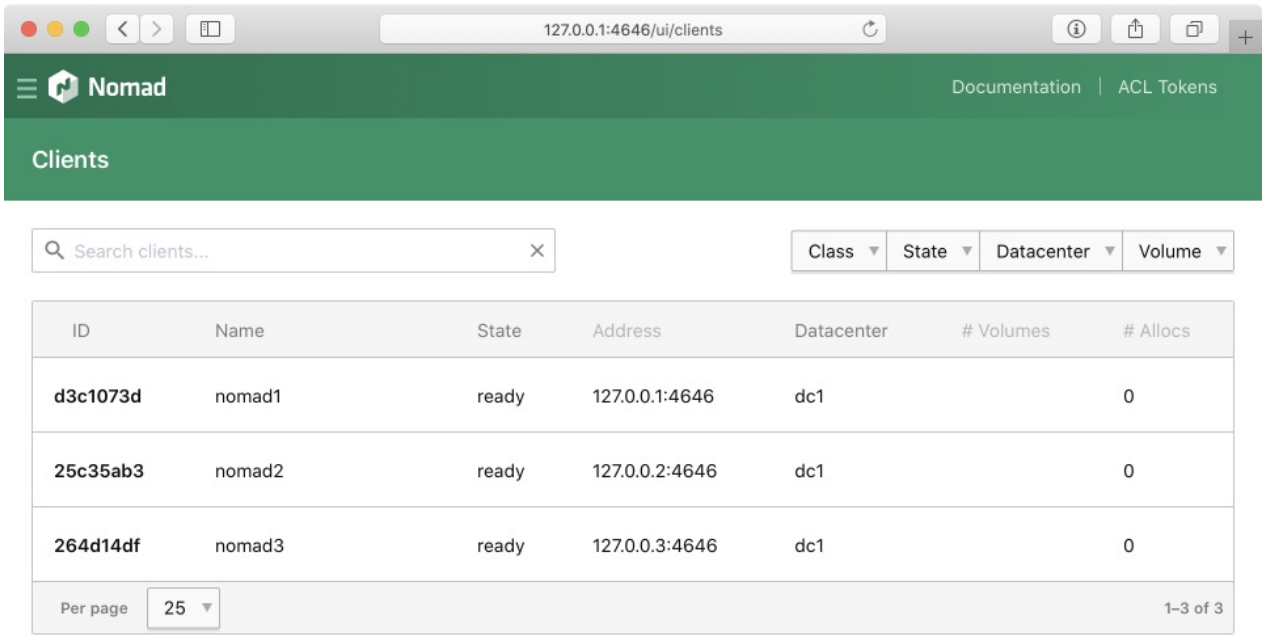
nomad配置

修改 /etc/dtle/nomad.hcl:

```
name = "nomad1" # rename for each node
# ... 省略未更改项
advertise {
  http = "127.0.0.1:4646"
  rpc  = "127.0.0.1:4647"
  serf = "127.0.0.1:4648"
}
server {
  enabled          = true
  bootstrap_expect = 3
}
consul {
  address = "127.0.0.1:8500"
}
plugin "dtle" {
  config {
    # ... 省略未更改项
    nats_bind = "127.0.0.1:8193"
    nats_advertise = "127.0.0.1:8193"
    consul = "127.0.0.1:8500"
    nomad_addr = "127.0.0.1:4646" # compatibility API need to access a nomad server
  }
}
```

全部启动后, nomad将自动向consul注册，组成集群:


127.0.0.1:4646/ui/servers
Nomad
Documentation | ACL Tokens
Servers
Name Status Leader Address port Datacenter
nomad1.global alive True 127.0.0.1 4648 dc1
nomad3.global alive False 127.0.0.3 4648 dc1
nomad2.global alive False 127.0.0.2 4648 dc1
1–3 of 3


127.0.0.1:4646/ui/clients
Nomad
Documentation | ACL Tokens
Clients
Search clients...
Class State Datacenter Volume
ID Name State Address Datacenter # Volumes # Allocs
d3c1073d nomad1 ready 127.0.0.1:4646 dc1 0
25c35ab3 nomad2 ready 127.0.0.2:4646 dc1 0
264d14df nomad3 ready 127.0.0.3:4646 dc1 0
Per page 25
1–3 of 3

功能/场景的映射列表

场景	复制手段(binlog-binlog)	复制手段(binlog-sql)	复制模式(全量+增量)	复制模式(增量)
单个MySQL 单向复制到 单个MySQL	支持	支持	支持	支持
单个MySQL 双向复制到 单个MySQL	支持	支持	-	支持
多个MySQL的表 合并到 单个MySQL	支持	支持	支持	支持
单个MySQL的不同表 分发到 多个MySQL	支持	支持	支持	支持
单个MySQL的同一表的不同记录 分发 到 多个MySQL	支持按主键分发; 不支持按函数分发	支持	支持	支持按主键分发; 不支持按函数分发
公有云间的数据同步	不支持	支持	支持	支持
单个MySQL复制到Kafka	-	-	支持	支持
多个MySQL复制到Kafka	-	-	支持	支持

场景	复制对象(整库复制)	复制对象(整表复制)	复制对象(按条件复制部分记录)
单个MySQL 单向复制到 单个MySQL	支持	支持	支持
单个MySQL 双向复制到 单个MySQL	支持; 不支持建表语句包含 `if not exists` #361	支持	支持
多个MySQL的表 合并到 单个MySQL	-	支持	支持
单个MySQL的不同表 分发到 多个MySQL	-	支持	支持
单个MySQL的同一表的不同记录 分发 到 多个MySQL	-	支持	支持
公有云间的数据同步	支持	支持	支持
单个MySQL复制到Kafka	支持	支持	支持
多个MySQL复制到Kafka	支持	支持	支持

场景	复制链路(链路压缩)	复制链路(跨网络边际)	回访模式(并行回放)
单个MySQL 单向复制到 单个MySQL	支持	支持	支持
单个MySQL 双向复制到 单个MySQL	支持	支持	支持
多个MySQL的表 合并到 单个MySQL	支持	支持	支持
单个MySQL的不同表 分发到 多个MySQL	支持	支持	支持
单个MySQL的同一表的不同记录 分发 到 多个MySQL	支持	支持	支持
公有云间的数据同步	支持	支持	支持
单个MySQL复制到Kafka	支持	支持	-
多个MySQL复制到Kafka	支持	支持	-

场景	自动创建表结构	支持**DDL**	**Agent** 水平扩展
单个MySQL 单向复制到 单个MySQL	支持	支持	支持
单个MySQL 双向复制到 单个MySQL	-	支持	支持
多个MySQL的表 合并到 单个MySQL	支持	支持	支持
单个MySQL的不同表 分发到 多个MySQL	支持	支持	支持
单个MySQL的同一表的不同记录 分发到 多个MySQL	支持	支持按主键分发; 不支持按函数分发	支持按主键分发; 不支持按函数分发
公有云间的数据同步	支持	支持	支持
单个MySQL复制到Kafka	支持	不支持	支持
多个MySQL复制到Kafka	支持	不支持	支持

场景	高可用**(**故障转移**)**	高可用**(**断点续做**)**	任务暂停/恢复	监控
单个MySQL 单向复制到 单个MySQL	支持	支持	支持	支持
单个MySQL 双向复制到 单个MySQL	支持	支持	支持	支持
多个MySQL的表 合并到 单个MySQL	支持	支持	支持	支持
单个MySQL的不同表 分发到 多个MySQL	支持	支持	支持	支持
单个MySQL的同一表的不同记录 分发 到 多个MySQL	支持按主键分发; 不支持按函数分发	支持按主键分发; 不支持按函数分发	支持按主键分发; 不支持按函数分发	支持
公有云间的数据同步	支持	支持	支持	支持
单个MySQL复制到Kafka	支持	不支持	支持	支持
多个MySQL复制到Kafka	支持	不支持	支持	支持

使用限制

限制

- 仅支持 MySQL 5.6/5.7 版本
- 仅支持 InnoDB 引擎
- 仅支持以下字符集:
  - latin1
  - latin2
  - gb2312, gbk, gb18030
  - utf8, utf8mb4
  - utf32
  - binary
- 在latin1/2表中，不支持非latin字符（如中文）（#388）
- 对于非UTF8编码执行的DDL，不支持DDL中含有混合编码字符串，如 `(col varchar default _utf32"...")`
- binlog 仅支持 `row` 模式
- binlog image 仅支持 `FULL` 模式
- 源端和目标端大小写敏感配置 (`lower_case_table_names`) 需保持一致
- 需要开启 GTID
- 不支持 Trigger
- 暂不支持 View
- 支持procedure，function，event的增量部分迁移（须创建库级别的迁移job），但存在源端与目标端字符集不完全一致的问题#357
- 支持user增量部分的迁移（须创建实例级别的迁移job），且支持grant，revoke（要求回放用户有 `grant option` ）
- 支持MySQL认证方式 `mysql_native_password` (MySQL 5.7)和 `caching_sha2_password` (MySQL 8.0)，其他认证方式不详
- 在dtle的增量复制过程中，如果源端执行 `replace into` 语句或者执行产生Duplicate entry冲突insert语句，可能导致目标端的 `AUTO_INCREMENT` 值和源端不一致（MySQL Bug#83030）

源端 **MySQL** 需配置如下参数

参数	值	检查方式
`log_bin`	ON (my.cnf中填写合法文件名)	`show global variables like 'log_bin'`
`binlog_format`	ROW	`show global variables like 'binlog_format';`
`binlog_row_image`	FULL	`show global variables like 'binlog_row_image';`
`log_slave_updates`	ON	`show global variables like 'log_slave_updates';`
`gtid_mode`	ON	`show global variables like 'gtid_mode';`

- 对于 `lower_case_table_names` 参数, dtle支持的值为 `0` 或 `1` 。
  - 原则上要求源端和目标端设置相同。
  - 且job存续期间，MySQL上该参数的值不可改变。
  - 允许设置参数值 `2` , 但不支持大小写混用。

关于外键 **(foreign key)**

在3.21.10.0以前，dtle回放时会设置 `set @@foreign_key_checks=OFF` 。外键的级连操作（如on update cascade）将无法生效。

从3.21.10.0开始，dtle增量回放时，默认 `set @@foreign_key_checks=ON` 。可以触发外键级连操作。

对于存在外键关系的一系列表，需要这些表在同一个job的复制范围内，才能正常执行。

该行为可用job配置中dest部分 `ForeignKeyChecks` 控制，默认为true。如有必要，可将其设为false。

涉及外键引用父表的事务，回放时会单独回放，不能并行。

遗留问题：在外键子表上 `alter table drop foreign key` 后，原外键父表依然会被认为是外键父表。

端口使用说明

默认情况下, nomad 和 consul的传输/通信会使用如下端口:

端口号	说明
8190	dtle 2.x HTTP API兼容层的端口
8500	consul HTTP 端口
4646	nomad HTTP 端口
4647	nomad RPC 端口
4648	nomad serf端口
8193	数据传输的端口

如何修改

端口配置可在/etc/dtle/nomad.hcl中修改

对目标端数据库的影响(gtid_executed表)

表 `dtle.gtid_executed_v4`

当目标端是MySQL数据库时, dtle会在目标端自动创建表 `dtle.gtid_executed_v4` , 目标端的用于回放数据的数据库用户需要对这张表有相应权限.

表 `dtle.gtid_executed_v4` 的作用是存储已经回放的事务的GTID, 用作断点续传/数据检查等.

使用表 `dtle.gtid_executed_v4` 模仿GTID机制, 而不使用MySQL原生GTID机制的原因是: 在回放时, `set GTID_NEXT=...` 语句需要 SUPER 权限, 而云环境下, 数据库用户可能无法拥有 SUPER 权限.

`dtle.gtid_executed_v4` 的建表语句如下:

```
CREATE TABLE IF NOT EXISTS dtle.gtid_executed_v4 (
    job_name varchar(64) NOT NULL,
    source_uuid binary(16) NOT NULL,
    gtid int NOT NULL,
    gtid_set longtext,
  primary key (job_name, source_uuid, gtid)
);
```

表结构说明:

- job_name: 执行同步的任务名
- source_uuid: 源端数据库UUID号
- gtid: 执行过的GTID gno编号。若某行该列为0，则表明这是一个汇总行
  - 行数过多时，会触发汇总机制
- gtid_set: 对于gtid=0的汇总行，该列批量储存gno编号，如1-100:200:300-400

典型的查询方法

```
SELECT job_name, HEX(source_uuid), gtid, gtid_set FROM dtle.gtid_executed_v4;
-- 注意source_uuid以binary储存，直接查询会乱码，需要HEX()转换
```

监控项说明

nomad原生metrics可访问： `http://127.0.0.1:4646/v1/metrics?format=prometheus`

由于nomad plugin并不能访问nomad监控接口，dtle有关的监控需要通过API兼容层访问。

注意：通过兼容层只能看到本节点运行的任务的监控项。

配置

首先配置nomad.hcl中打开api兼容层，并配置 `publish_metrics = true` 。

```
plugin "dtle" {
  config {
    api_addr = "127.0.0.1:8190"
    nomad_addr = "127.0.0.1:4646"
    publish_metrics = true
    stats_collection_interval = 15
    ...
```

访问 `127.0.0.1:8190/metrics` 可查看监控项，或在prometheus中配置从此地址获取监控项。

监控项

类别	监控项	说明
网络流量状态	-	-
-	network.in_msgs	-
-	network.out_msgs	-
-	network.in_bytes	-
-	network.out_bytes	-
缓存/队列状态	-	-
-	buffer.src_queue_size	-
-	buffer.dest_queue_size	-
-	buffer.send_by_timeout	-
-	buffer.send_by_size_full	-
内存使用估计	-	-
--全量计数值	memory.full_kb_count	-
--增量计数值	memory.incr_kb_count	-
--全量估计值	memory.full_kb_est	-
--增量估计值	memory.incr_kb_est	-

延迟统计	-	-
-	delay.time	-
表统计（未实现）	-	-
-	table.insert	-
-	table.update	-
-	table.delete	-
吞吐统计（未实现）	-	-
-	throughput.num	-
-	throughput.time	-
事务统计	-	-
-	src_extracted_incr_tx_count	增量阶段中源端完成抽取并解析的事务总量。从源端任务启动开始计数，重启任务时计数清零。可配合prometheus的irate()计算tps，如：irate(demo_src_extracted_tx_count[1m])
-	dest_applied_incr_tx_count	增量阶段中目标端完成回放的事务总量。从目标端任务启动开始计数，重启任务时计数清零。可配合prometheus的irate()计算tps，如：irate(demo_dest_applied_tx_count[1m])
sql执行量统计	-	-
-	src_extracted_incr_query_count	增量阶段中源端完成抽取并解析的dml/ddl数量。从源端任务启动开始计数，重启任务时计数清零。可配合prometheus的irate()计算qps，如：irate(demo_src_extracted_query_count[1m])
-	dest_applied_incr_query_count	增量阶段中目标端执行的ddl/dml总量(未commit前也算)。从目标端任务启动开始计数，重启任务时计数清零。可配合prometheus的irate()计算qps，如：irate(demo_dest_applied_query_count[1m])

内存使用

- dtle根据数据量（内存计数值）来估计内存占用。因程序处理，实际使用的内存有放大效应
- 内存估计值 = 内存计数值 x 放大系数
- 根据Go内存分配器原理，job处理完后，内存可能不会立刻被释放给操作系统

任务延迟

延迟统计仅对增量（含Kafka输出）有效，其原理为：

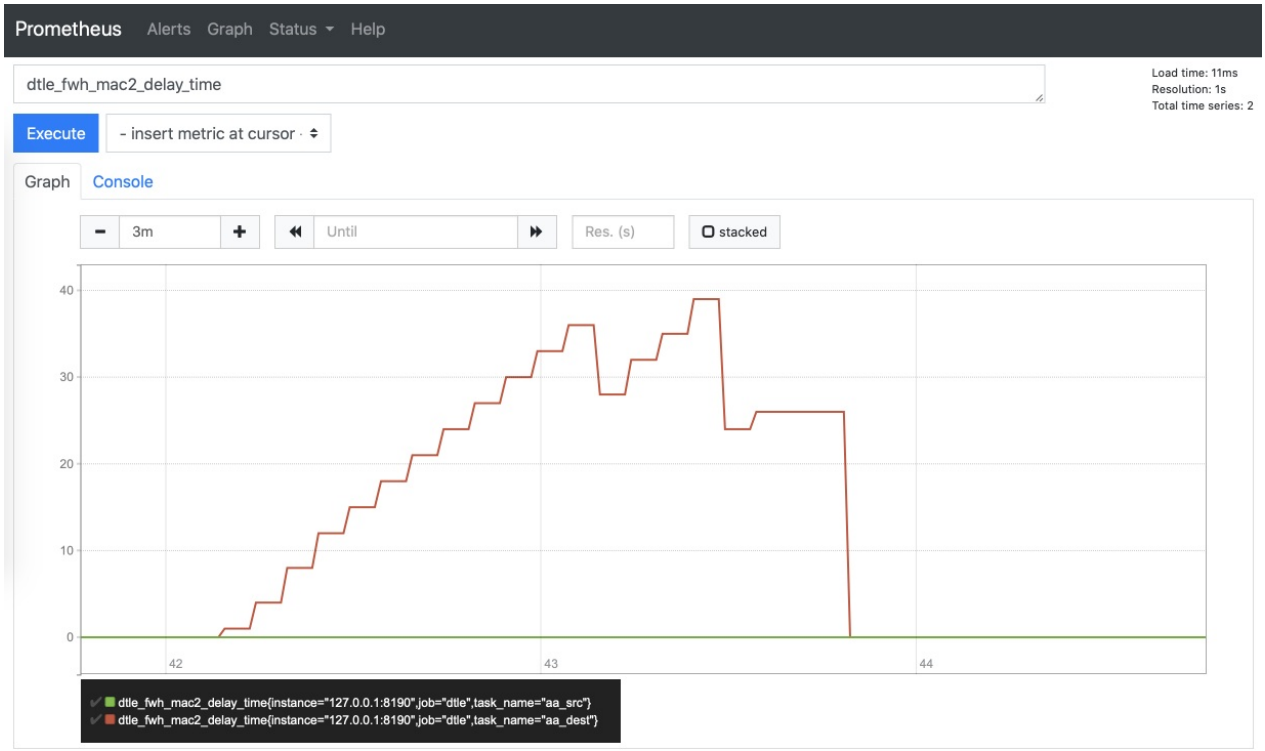
- 源端MySQL在执行事务时，binlog中记录了时间戳
- dtle在传输/回放事务时，取时间戳和当前时间的差值为延迟值
- 如果一段时间（15s）没有事务，则重置延迟值为0

注意事项

- 需要MySQL和dtle主机的时间基本正确
- 源端和目标端都有延迟统计，取两者中大值为延迟

为了便于查看延迟曲线以及跟踪高延迟情况，可用Prometheus抓取dtle的监控项并使用Alertmanager发送告警， 步骤可参考 延迟告警示例。

效果图：


Prometheus
Alerts
Graph
Status
Help
dtle_fwh_mac2_delay_time
Load time: 11ms
Resolution: 1s
Total time series: 2
Execute
- insert metric at cursor -
Graph
Console
3m
Until
Res. (s)
stacked
40
30
20
10
0
42
43
44
dtle_fwh_mac2_delay_time{instance="127.0.0.1:8190",job="dtle",task_name="aa_src"}
dtle_fwh_mac2_delay_time{instance="127.0.0.1:8190",job="dtle",task_name="aa_dest"}

延迟监控告警示例

Prometheus可直观查看监控项并记录历史值, 可通过搭建Prometheus查看延迟情况, Alertmanager 可针对异常监控项及时发出告警信息, 通过配置Alertmanager对延迟异常任务发出告警

Prometheus配置

查看监控项并记录历史值

- 准备配置文件 prometheus.yml :

```
# 设定alertmanager和prometheus交互的接口，即alertmanager监听的ip地址和端口
alerting:
 alertmanagers:
 - static_configs:
   - targets: ["127.0.0.1:9093"]

# 告警规则文件
rule_files:
  - 'prometheus_rule.yml'

scrape_configs:
  - job_name: 'dtle'

    # Override the global default and scrape targets from this job every 5 seconds.
    scrape_interval: 5s

    static_configs:
      - targets: ['127.0.0.1:8190','127.0.0.2:8190'] # 填写dtle兼容层的地址。可填多个。
```

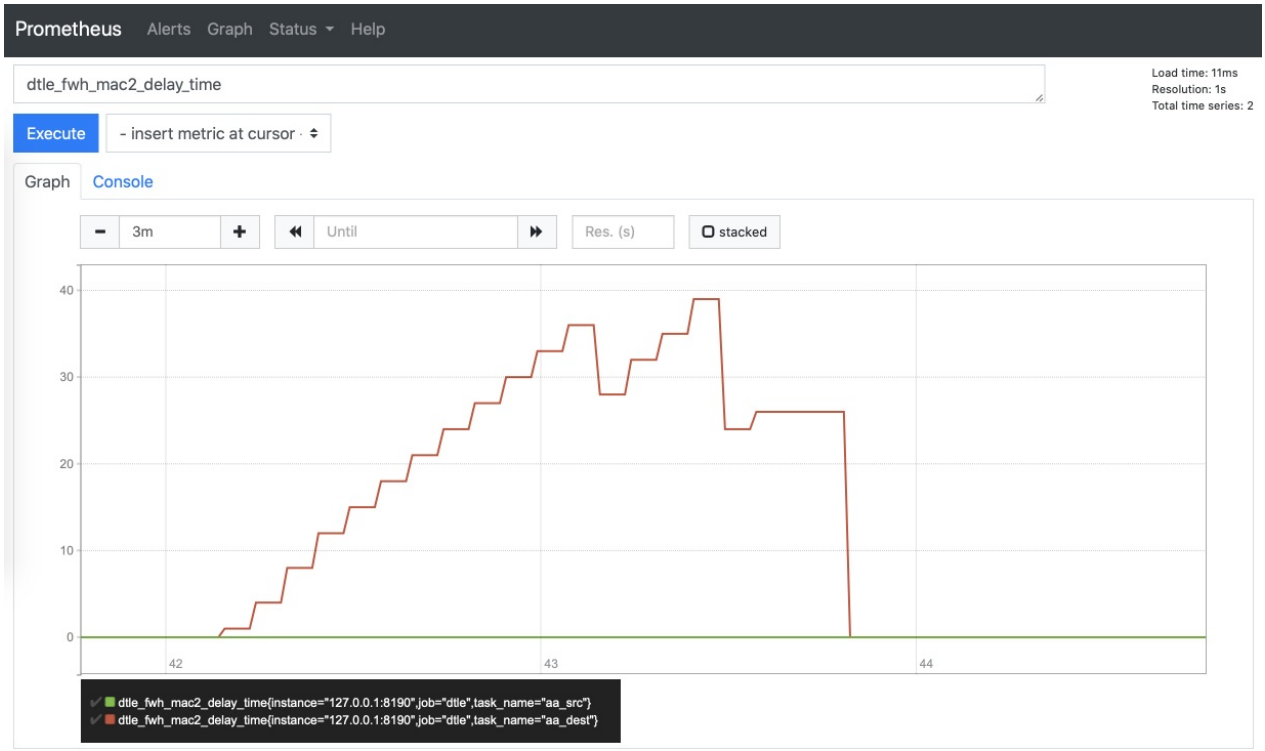
- 准备告警规则文件 prometheus_rule.yml

```
groups:
- name: simple_example
  rules:

  # Alert for task that is delay more than 5s for >1 minutes.
  - alert: TaskDelay
    expr:  dtle_delay_time > 5
    for: 1m
    labels:
      severity: warning
    annotations:
      summary: "task {{ $labels.task_name }} has delay"
      description: "Task {{ $labels.task_name }} of instance {{ $labels.instance }} has delay 5s  more than 1 minutes."
```

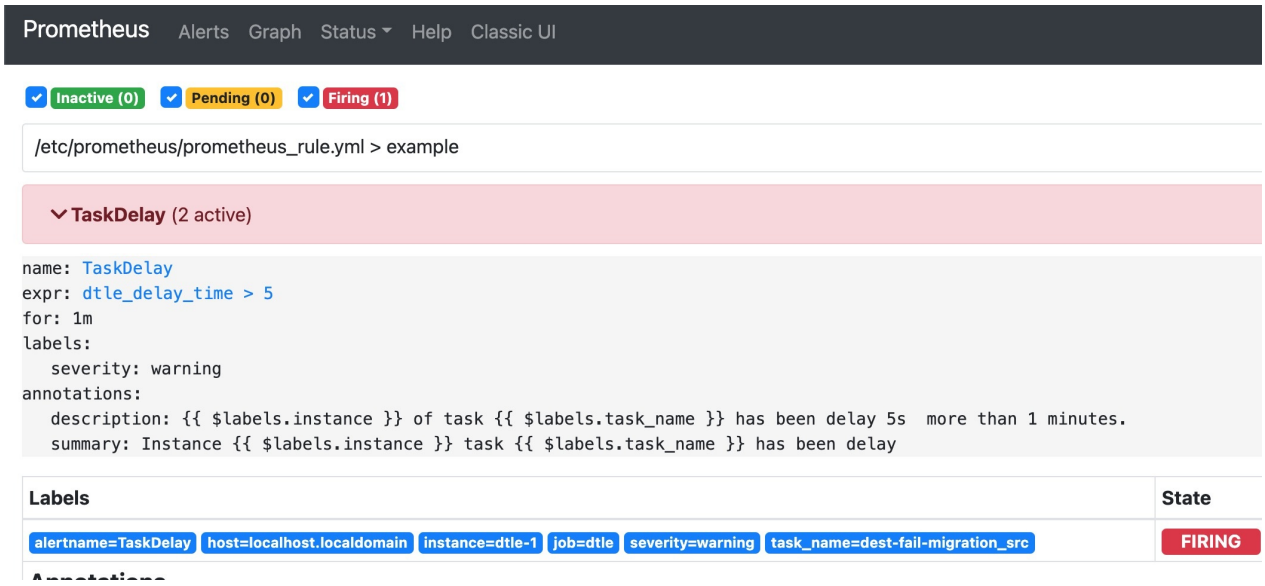
- 使用docker运行Prometheus：

```
docker run \
    -p 9090:9090 \
    -v ${PWD}/prometheus.yml:/etc/prometheus/prometheus.yml \
    -v ${PWD}/prometheus_rule.yml:/etc/prometheus/prometheus_rule.yml \
    prom/prometheus
```

- 然后浏览器访问 http://127.0.0.1:9090, 并查询(Prometheus提供补全)需要的监控项。



- 访问http://127.0.0.1:9090/alerts, 获取当前告警规则/内容



Alertmanager配置

针对任务延迟异常发送告警

- 创建配置文件 alertmanager.yml 配置示例如下

```
global:
  smtp_smarthost: 'smtp.gmail.com:587'
  smtp_from: 'SENDER_ACCOUNT'
  smtp_auth_username: 'SENDER_ACCOUNT'
  smtp_auth_password: 'email smtp verify password'
  smtp_require_tls: false
route:
  # If an alert has successfully been sent, wait 'repeat_interval' to resend them.
  repeat_interval: 10s
  #  A default receiver
  receiver: team-dtle-mails

receivers:
  - name: 'team-dtle-mails'
    email_configs:
    - to: 'receiver@actionsky.com'
```

- 启动alertmanager

```
docker run -p 9093:9093 -v  ${PWD}/alertmanager.yml:/etc/alertmanager/alertmanager.yml prom/alertmanager
```

- 根据配置延迟5s以上并持续1min时,receiver@actionsky.com 邮箱收到告警如下:

```
[1] Firing
Labels
alertname = TaskDelay
host = localhost.localdomain
instance = dtle-1
job = dtle
severity = warning
task_name = dest-fail-migration_src
Annotations
description = Task dest-fail-migration_src of instance dtle-1  has delay 5s more than 1 minutes.
summary = task dest-fail-migration_src has delay
Source
```

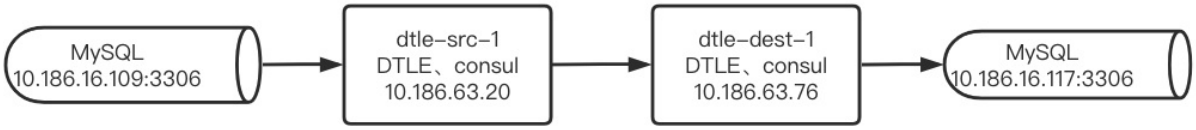
如何搭建**DTLE**的监控系统

背景:

虽然在DTLE的文档里提供各种监控项的介绍，但是对于不熟悉prometheus和grafana配置的同学来说上手还是有些难度的。今天我就来DTLE 3.21.07.0来搭建一个DTLE的监控系统。

一、搭建**DTLE**运行环境

- 配置两个节点的DTLE集群来演示,其拓扑如下:



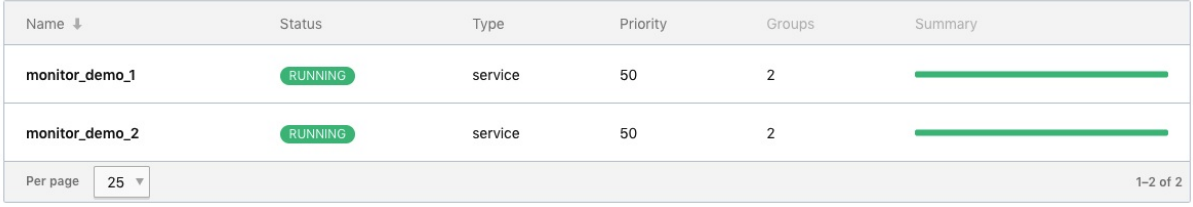
在修改DTLE配置文件的时候需要注意以下两点:

1. 开启DTLE的监控，确保publish_metrics的值为ture
2. 开启nomad的监控，确保正确配置telemetry

这里以dtle-src-1的配置为例，具体配置参考节点配置:

```
# DTLE 3.21.07.0中nomad升级为1.1.2，需要添加如下配置使nomad提供监控数据
# 之前版本的DTLE无需添加此配置
telemetry {
  prometheus_metrics         = true
  collection_interval        = "15s"
}

plugin "dtle" {
  config {
    data_dir = "/opt/dtle/var/lib/nomad"
    nats_bind = "10.186.63.20:8193"
    nats_advertise = "10.186.63.20:8193"
    # Repeat the consul address above.
    consul = "10.186.63.76:8500"

    # By default, API compatibility layer is disabled.
    api_addr = "10.186.63.20:8190"   # for compatibility API
    nomad_addr = "10.186.63.20:4646" # compatibility API need to access a nomad server

    publish_metrics = true
    stats_collection_interval = 15
  }
}
```

- 添加两个job模拟两个MySQL实例之间传输数据



二、部署**prometheus**

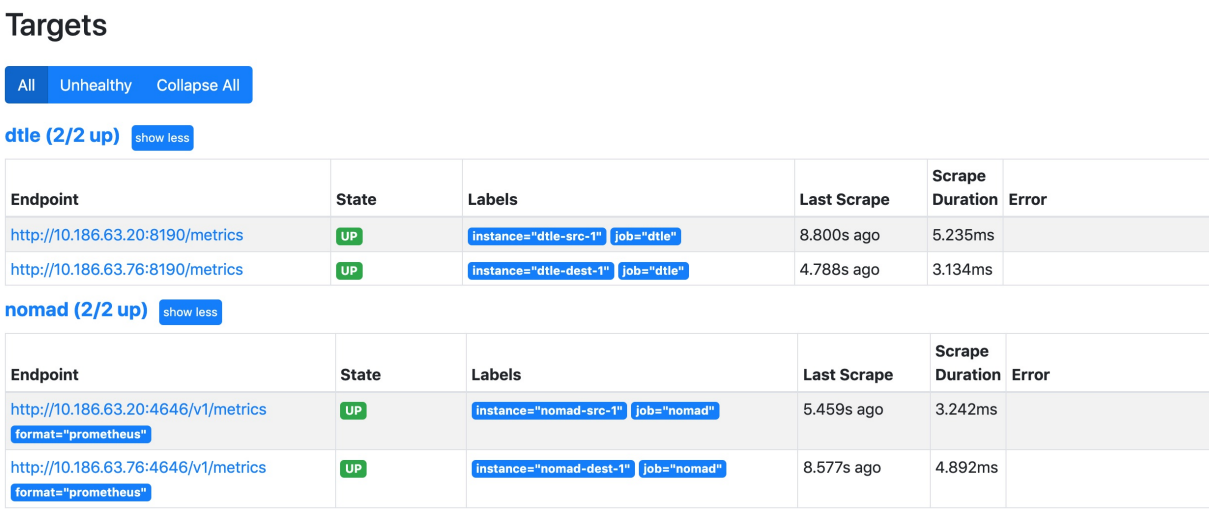
- 准备prometheus配置文件同时接收nomad和DTLE的metrics
- DTLE监控labels:instance的值建议设置为DTLE服务器的hostname

```
shell> cat /path/to/prometheus.yml
global:
  scrape_interval:     15s # Set the scrape interval to every 15 seconds. Default is every 1 minute.
  evaluation_interval: 15s # Evaluate rules every 15 seconds. The default is every 1 minute.

scrape_configs:
  - job_name: 'nomad'
    scrape_interval: 15s
    metrics_path: '/v1/metrics'
    params:
      format: ['prometheus']
    static_configs:
      - targets: ['10.186.63.20:4646']
        labels:
          instance: nomad-src-1
      - targets: ['10.186.63.76:4646']
        labels:
          instance: nomad-dest-1

  - job_name: 'dtle'
    scrape_interval: 15s
    metrics_path: '/metrics'
    static_configs:
      - targets: ['10.186.63.20:8190']
        labels:
          instance: dtle-src-1
      - targets: ['10.186.63.76:8190']
        labels:
          instance: dtle-dest-1
```

- 利用docker部署prometheus服务

```
shell> docker run -itd -p 9090:9090 --name=prometheus --hostname=prometheus --restart=always -v /path/to/prometheus.yml:/
etc/prometheus/prometheus.yml prom/prometheus
```

- 在浏览器上访问prometheus的页面 http://${prometheus_server_ip}:9090/targets 验证配置生效

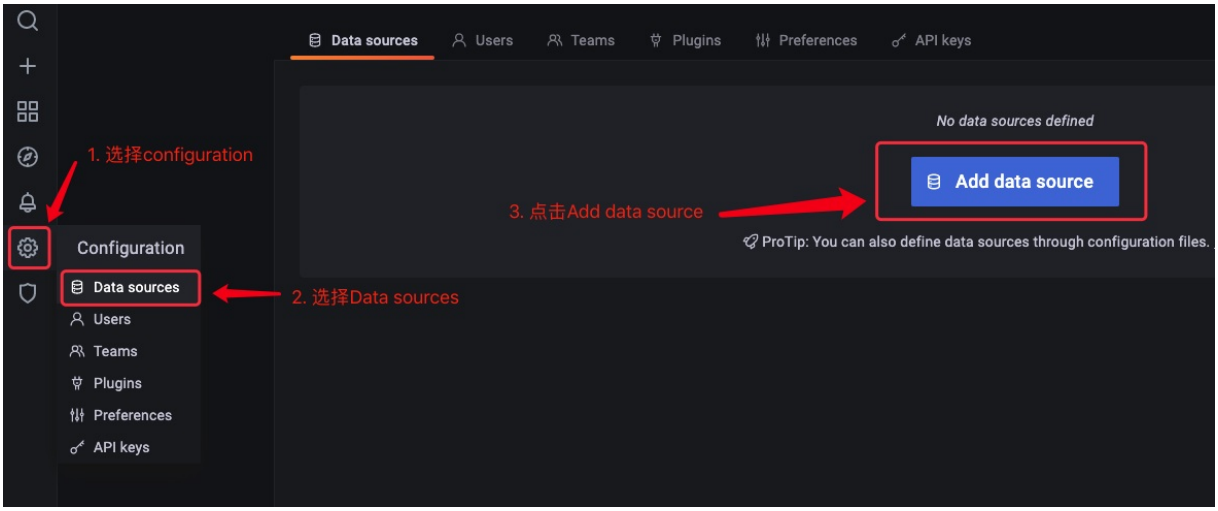


三、部署**grafana**

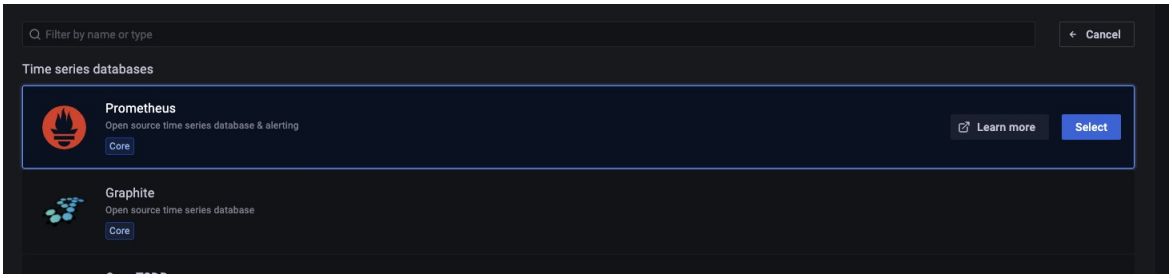
- 利用docker部署grafana服务

```
shell> docker run -d --name=grafana -p 3000:3000 grafana/grafana
```

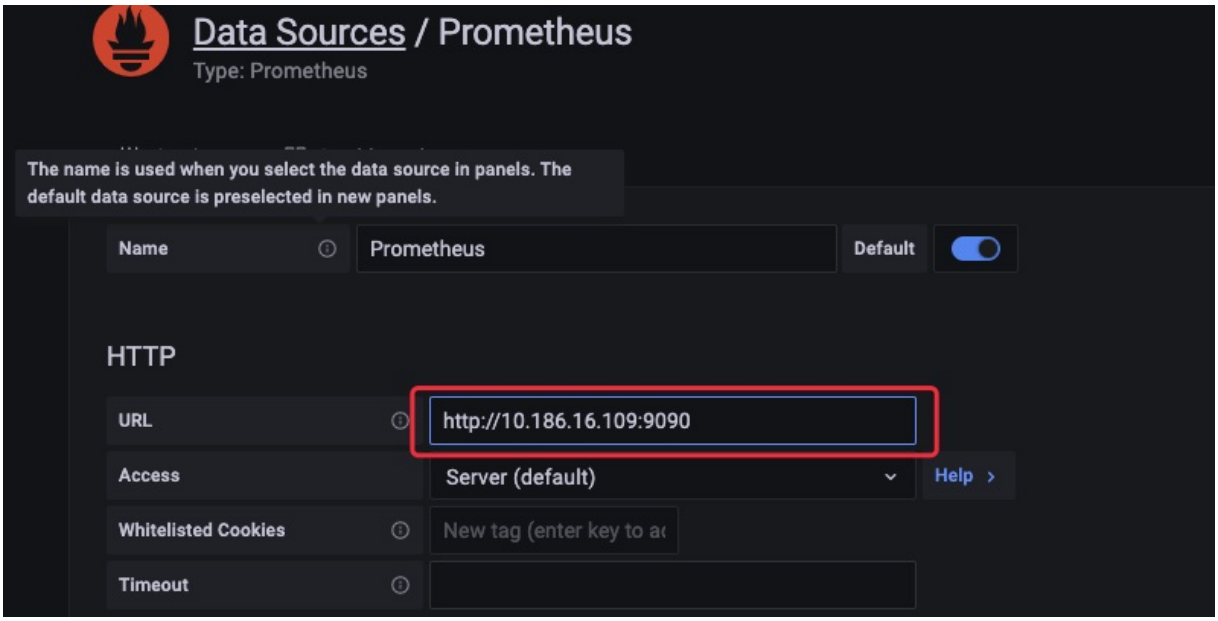
- 在浏览器上访问grafana的页面 http://${grafana_server_ip}:3000 ， 使用默认用户 admin/admin登录
- 配置添加数据源



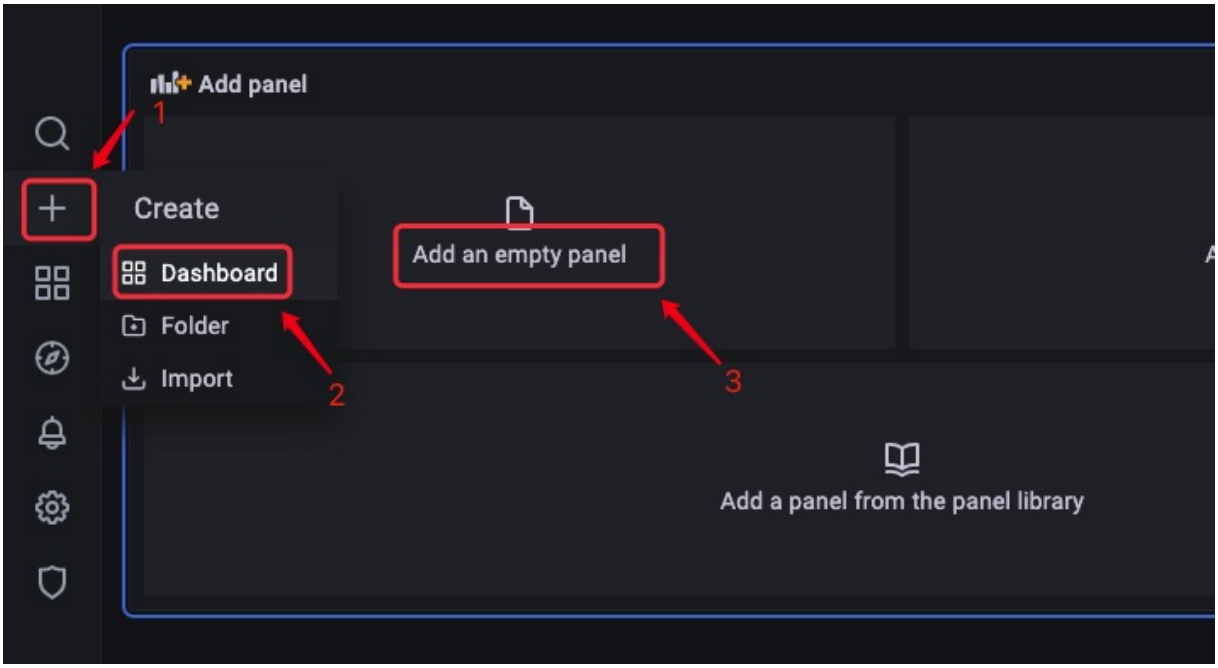
  - 选择添加promethues



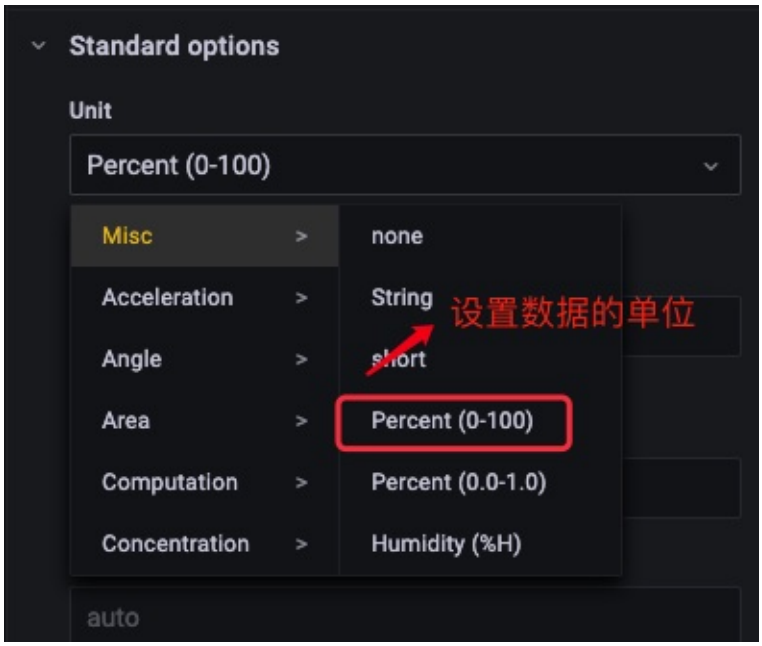
  - 只需将promethues的访问地址添加到URL中，点击“sava & test”按钮



- 添加panel



- 以添加一个CPU使用率监控为例配置一个panel



四、常用的监控项

nomad所有监控项： https://www.nomadproject.io/docs/operations/metrics

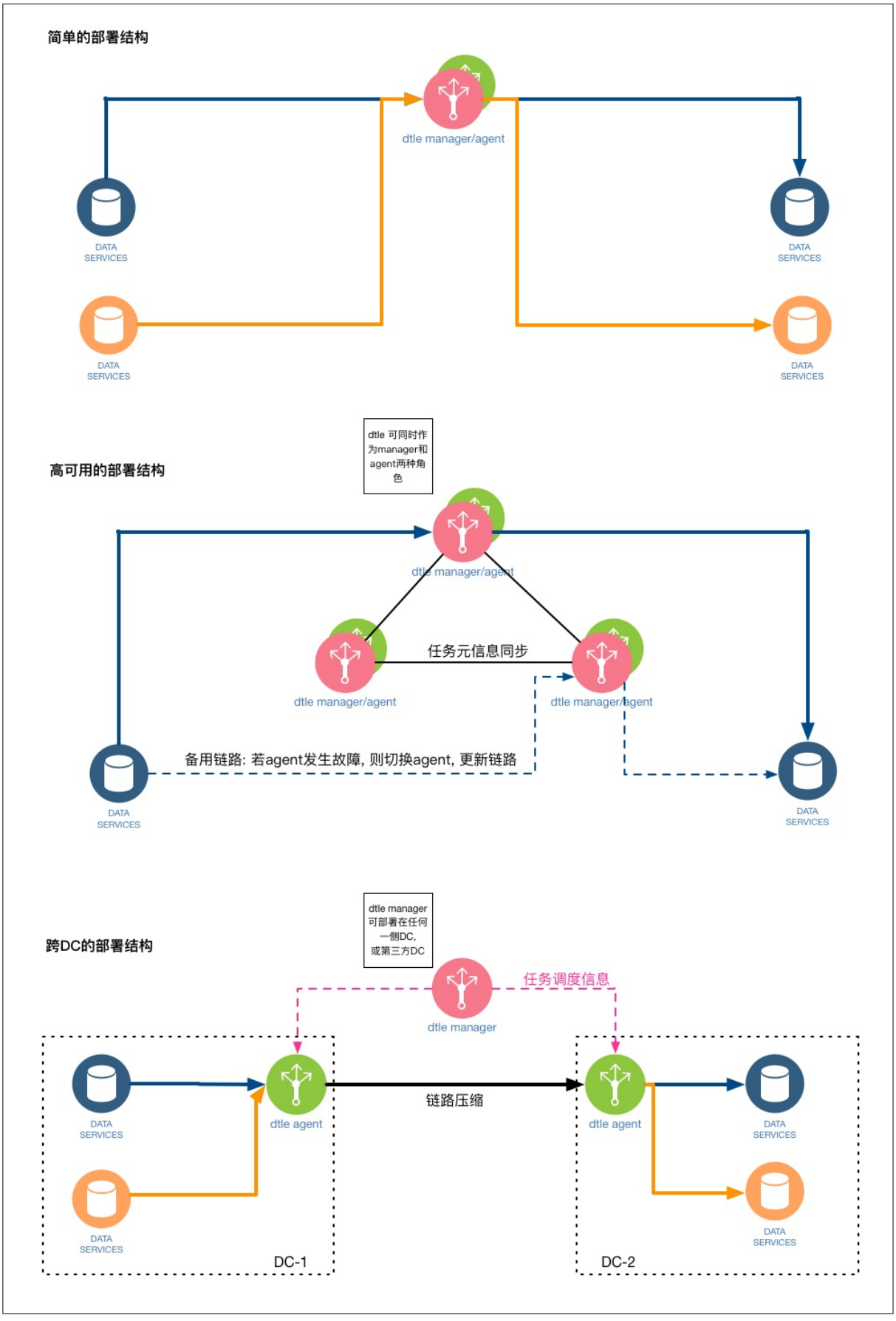
DTLE所有监控项：https://actiontech.github.io/dtle-docs-cn/3/3.4_metrics.html

说明	公式示例	单位
CPU使用率(总计)	`sum(rate(process_cpu_seconds_total{instance=~"nomad-src-1 | dtle-src-1"}[60s])) * 100`	Misc / Percent(0-100)
CPU使用率(DTLE)	`rate(process_cpu_seconds_total{instance="dtle-src-1"}[60s]) * 100`	Misc / Percent(0-100)
CPU使用率(nomad)	`rate(process_cpu_seconds_total{instance="dtle-src-1"}[60s]) * 100`	Misc /Percent(0-100)
内存使用(总计)	`sum(process_resident_memory_bytes{instance=~"nomad-src-1 | dtle-src-1"}) /1024 /1024`	Data / mebibyte
内存使用(DTLE)	`process_resident_memory_bytes{instance="dtle-src-1"} /1024 /1024`	Data / mebibyte
内存使用(nomad)	`process_resident_memory_bytes{instance="nomad-src-1"} /1024 /1024`	Data / mebibyte
带宽(总计 - 源端发送)	`sum(increase(dtle_network_out_bytes{host="dtle-src-1"}[30s]) /30 /1024) * 8`	Data rate / kibibits/sec
带宽(按task分组 - 源端发送)	`increase(dtle_network_out_bytes{host="dtle-src-1"}[30s]) /30 /1024 * 8`	Data rate / kibibits/sec
带宽(总计 - 目标端接收)	`sum(increase(dtle_network_in_bytes{host="dtle-dest-1"}[30s]) /30 /1024) * 8`	Data rate / kibibits/sec
带宽(按task分组 - 目标端接收)	`increase(dtle_network_in_bytes{host="dtle-dest-1"}[30s]) /30 /1024 * 8`	Data rate / kibibits/sec
数据延迟(源端)	`dtle_delay_time{host="dtle-src-1"}`	Time / seconds(s)
数据延迟(目标端)	`dtle_delay_time{host="dtle-dest-1"}`	Time / seconds(s)
TPS(源端)	`irate(dtle_src_extracted_incr_tx_count[30s])`	Misc / none
TPS(目标端)	`irate(dtle_dest_applied_incr_tx_count[30s])`	Misc / none
QPS(源端)	`irate(dtle_src_extracted_incr_query_count[30s])`	Misc / none
QPS(目标端)	`irate(dtle_dest_applied_incr_query_count[30s])`	Misc / none
Buffer(源端)	`dtle_buffer_src_queue_size`	Misc / none
Buffer(目标端)	`dtle_buffer_dest_queue_size`	Misc / none

五、最后创建多个**panel**同时展示


Delay dtle-src-1
1 s
500 ms
0 s
-500 ms
-1 s
17:33
17:34
17:35
17:36
17:37
dtle_delay_time{host="dtle-src-1", instance="dtle-src-1", job="dtle", task_name="monitor_demo_1_src"}
dtle_delay_time{host="dtle-src-1", instance="dtle-src-1", job="dtle", task_name="monitor_demo_2_src"}
Buffer dtle-src-1
1
0.500
0
-0.50
-1
17:33
17:34
17:35
17:36
17:37
dtle_buffer_src_queue_size{host="dtle-src-1", instance="dtle-src-1", job="dtle", task_name="monitor_de
dtle_buffer_src_queue_size{host="dtle-src-1", instance="dtle-src-1", job="dtle", task_name="monitor_de
TPS dtle-src-1
100
75
50
25
0
17:33
17:34
17:35
17:36
17:37
{host="dtle-src-1", instance="dtle-src-1", job="dtle", task_name="monitor_demo_1_src"}
{host="dtle-src-1", instance="dtle-src-1", job="dtle", task_name="monitor_demo_2_src"}
CPU dtle-src-1
20 %
15 %
10 %
5 %
0 %
17:33
17:34
17:35
17:36
17:37
Total
{instance="dtle-src-1", job="dtle"}
{instance="nomad-src-1", job="nomad"}
Memory dtle-src-1
150 MiB
125 MiB
100 MiB
75 MiB
50 MiB
25 MiB
17:33
17:34
2021-08-23 17:34:00
Total: 125 MiB
{instance="dtle-src-1", job="dtle"}: 75.5 MiB
{instance="nomad-src-1", job="nomad"}: 49.9 MiB
Total
{instance="dtle-src-1", job="dtle"}
{instance="nomad-src-1", job="nomad"}
Bandwidth dtle-src-1
500 Kib/s
400 Kib/s
300 Kib/s
200 Kib/s
100 Kib/s
0 Kib/s
17:33
17:34
17:35
17:36
17:37
Total
{host="dtle-src-1", instance="dtle-src-1", job="dtle", task_name="monitor_demo_1_src"}
{host="dtle-src-1", instance="dtle-src-1", job="dtle", task_name="monitor_demo_2_src"}
Delay dtle-dest-1
1 s
500 ms
0 s
-500 ms
-1 s
17:33
17:34
17:35
17:36
17:37
dtle_delay_time{host="dtle-dest-1", instance="dtle-dest-1", job="dtle", task_name="monitor_demo_1_de
dtle_delay_time{host="dtle-dest-1", instance="dtle-dest-1", job="dtle", task_name="monitor_demo_2_de
Buffer dtle-dest-1
1
0.500
0
-0.50
-1
17:33
17:34
17:35
17:36
17:37
dtle_buffer_dest_queue_size{host="dtle-dest-1", instance="dtle-dest-1", job="dtle", task_name="monito
dtle_buffer_dest_queue_size{host="dtle-dest-1", instance="dtle-dest-1", job="dtle", task_name="monito
TPS dtle-dest-1
100
75
50
25
0
17:33
17:34
17:35
17:36
17:37
{host="dtle-dest-1", instance="dtle-dest-1", job="dtle", task_name="monitor_demo_1_dest"}
{host="dtle-dest-1", instance="dtle-dest-1", job="dtle", task_name="monitor_demo_2_dest"}
CPU dtle-dest-1
15 %
10 %
5 %
0 %
17:33
17:34
17:35
17:36
17:37
Total
{instance="dtle-dest-1", job="dtle"}
{instance="nomad-dest-1", job="nomad"}
Memory dtle-dest-1
200 MiB
150 MiB
100 MiB
50 MiB
17:33
17:34
17:35
17:36
17:37
Total
{instance="dtle-dest-1", job="dtle"}
{instance="nomad-dest-1", job="nomad"}
Bandwidth dtle-dest-1
500 Kib/s
400 Kib/s
300 Kib/s
200 Kib/s
100 Kib/s
0 Kib/s
17:33
17:34
17:35
17:36
17:37
Total
{host="dtle-dest-1", instance="dtle-dest-1", job="dtle", task_name="monitor_demo_1_dest"}
{host="dtle-dest-1", instance="dtle-dest-1", job="dtle", task_name="monitor_demo_2_dest"}

3.5 部署结构



如上图, nomad (运行dtle插件)支持多种不同的部署结构, 其中:

- 简单的部署结构:
  - 适用于简单的场景, 用一个nomad节点同时作为server (管理节点)和client (执行节点，运行dtle插件)
  - 一个节点可同时处理多个传输链路
- 高可用的部署结构:
  - 适用于对可用性较高的场景, 将 nomad 和 consul 进行三节点集群部署, 任务元数据信息在集群中同步
  - 一个 nomad 节点可同时作为 server 和 client, 也可将 server 和 client 分开部署
  - 当server发生故障时, 传输任务会转移到其他server执行 (需要server集群存活一半以上)
  - 当client发生故障时, 传输任务会转移到其他client执行
- 跨DC的部署结构
  - 适用于多个数据中心间的数据同步
  - server集群可部署在任一数据中心, 或第三方数据中心
  - 源数据库和目标数据库 不必要保障 直接网络连通
  - client需部署在网络边界上

DDL支持度

以**DATABASE**为对象的**DDL**

DDL类型		DDL语句示例	全量	增量	备注
CREATE DATABASE	basic	CREATE DATABASE db_name; CREATE DATABASE IF NOT EXISTS db_name; CREATE SCHEMA db_name; CREATE SCHEMA IF NOT EXISTS db_name;	支持	支持	1. 支持server级默认字符集不一致情形
	character set and collate	CREATE SCHEMA db_name CHARACTER SET utf8 COLLATE utf8_general_ci; CREATE SCHEMA db_name DEFAULT CHARACTER SET utf8mb4 DEFAULT COLLATE utf8mb4_general_ci; CREATE SCHEMA db_name CHARACTER SET=latin1 COLLATE=latin1_swedish_ci; CREATE DATABASE db_name COLLATE latin2_general_ci; CREATE DATABASE db_name CHARACTER SET binary; CREATE DATABASE db_name DEFAULT CHARACTER SET=gbk DEFAULT COLLATE=gbk_chinese_ci;	支持	支持	1. 支持字符集 latin1、latin2、gbk、utf8、utf8mb4、binary
ALTER DATABASE		ALTER DATABASE db_name CHARACTER SET utf8 COLLATE utf8_general_ci	支持	支持	支持字符集 latin1、latin2、gbk、utf8、utf8mb4、binary
DROP DATABASE		DROP DATABASE db_name; DROP DATABASE IF EXISTS db_name; DROP SCHEMA db_name; DROP SCHEMA IF EXISTS db_name;	支持	支持	

以**TABLE**为对象的**DDL**

DDL类型			DDL语句示例	全量	增量	备注
	column data types and attributes	basic	CREATE TABLE [IF NOT EXISTS] tbl_name (col_name column_definition);	支持	支持	
		character set	CREATE TABLE [IF NOT EXISTS] tbl_name (col_name column_definition CHARACTER SET utf8 COLLATE utf8_bin);	支持	支持	
		null	CREATE TABLE [IF NOT EXISTS] tbl_name (col_name column_definition NULL);	支持	支持	
		not null	CREATE TABLE [IF NOT EXISTS] tbl_name (col_name column_definition NOT NULL);	支持	支持	
		default	CREATE TABLE [IF NOT EXISTS] tbl_name (c CHAR(20) DEFAULT '');	支持	支持	
		auto_increment	CREATE TABLE [IF NOT EXISTS] tbl_name (id INT(11) AUTO_INCREMENT PRIMARY KEY);	支持	支持	
		comment	CREATE TABLE [IF NOT EXISTS] tbl_name (id INT(11) COMMENT '');	支持	支持	
		generated always	CREATE TABLE [IF NOT EXISTS] tbl_name (sidea DOUBLE,sideb DOUBLE,sidec DOUBLE GENERATED ALWAYS AS (SQRT(sidea * sidea + sideb * sideb)));	支持	支持	建表语句支持，但是数据 https://github.com/actionte
		check	CREATE TABLE [IF NOT EXISTS] tbl_name (id INT(11) CHECK(expr));	支持	支持	
		primary key	CREATE TABLE [IF NOT EXISTS] tbl_name (id INT(11) AUTO_INCREMENT PRIMARY KEY); CREATE TABLE [IF NOT EXISTS] tbl_name (id INT(11) AUTO_INCREMENT, PRIMARY KEY (id)); CREATE TABLE [IF NOT EXISTS] tbl_name (id INT(11) AUTO_INCREMENT, c CHAR(20), PRIMARY KEY(id, c));	支持	支持	
		key	CREATE TABLE [IF NOT EXISTS] tbl_name (id INT(11) AUTO_INCREMENT KEY); CREATE TABLE [IF NOT EXISTS] tbl_name (id INT(11) AUTO_INCREMENT, KEY index_name (key_part)); CREATE TABLE [IF NOT EXISTS] tbl_name (id INT(11) AUTO_INCREMENT, c	支持	支持	

			CHAR(20) DEFAULT 't7', KEY (key_part, key_part));			
		index	CREATE TABLE [IF NOT EXISTS] tbl_name (id INT(11) AUTO_INCREMENT, INDEX index_name (key_part)); CREATE TABLE [IF NOT EXISTS] tbl_name (id INT(11) AUTO_INCREMENT, c CHAR(20) DEFAULT 't10', INDEX(key_part, key_part));	支持	支持	
	indexes and foreign keys	unique	CREATE TABLE [IF NOT EXISTS] tbl_name (id INT(11) AUTO_INCREMENT UNIQUE KEY); CREATE TABLE [IF NOT EXISTS] tbl_name (id INT(11) AUTO_INCREMENT, UNIQUE KEY index_name (key_part)); CREATE TABLE [IF NOT EXISTS] tbl_name (c CHAR(20), UNIQUE INDEX (key_part(prefix value))); CREATE TABLE [IF NOT EXISTS] tbl_name (id INT(11) AUTO_INCREMENT, c CHAR(20) DEFAULT 't10', CONSTRAINT UNIQUE INDEX (key_part, key_part));"	支持	支持	
		fulltext	CREATE TABLE [IF NOT EXISTS] tbl_name (c_varchar_1 varchar(255), c_varchar_2 varchar(255), FULLTEXT KEY index_name (key_part, key_part));	支持	支持	
		foreign key	CREATE TABLE [IF NOT EXISTS] tbl_name (id INT, parent_id INT, CONSTRAINT symbol FOREIGN KEY (col_name) REFERENCES tbl_name(key_part) ON DELETE reference_option ON UPDATE reference_option); CREATE TABLE [IF NOT EXISTS] tbl_name (id INT, parent_id INT, CONSTRAINT symbol FOREIGN KEY (col_name) REFERENCES tbl_name(key_part) ON DELETE reference_option ON UPDATE reference_option); CREATE TABLE [IF NOT EXISTS] tbl_name (id INT, parent_id INT, FOREIGN KEY index_name (col_name) REFERENCES tbl_name(key_part) ON DELETE reference_option ON UPDATE reference_option); CREATE TABLE [IF NOT	支持	支持	应满足一下配置，否则在 数据一致性 1.目标端数据库@@forei 1） 2.dtle job 中ForeignKeyC

			EXISTS] tbl_name (id INT, parent_id INT, FOREIGN KEY (col_name) REFERENCES tbl_name(key_part) ON DELETE reference_option ON UPDATE reference_option);			
CREATE TABLE	table options	engine	CREATE TABLE [IF NOT EXISTS] tbl_name (c CHAR(20)) ENGINE=InnoDB;	支持	支持	
		auto_increment	CREATE TABLE [IF NOT EXISTS] tbl_name (id INT(11) AUTO_INCREMENT PRIMARY KEY) AUTO_INCREMENT=100;	支持	支持	
		character set	CREATE TABLE [IF NOT EXISTS] tbl_name (c CHAR(20)) CHARACTER SET=utf8;	支持	支持	
		collate	CREATE TABLE [IF NOT EXISTS] tbl_name (c CHAR(20)) DEFAULT COLLATE=utf8_general_ci;	支持	支持	
		checksum	CREATE TABLE [IF NOT EXISTS] tbl_name (c CHAR(20)) CHECKSUM=1;	支持	支持	
		comment	CREATE TABLE [IF NOT EXISTS] tbl_name (id INT(11) AUTO_INCREMENT PRIMARY KEY) COMMENT='';	支持	支持	
		compression	CREATE TABLE [IF NOT EXISTS] tbl_name (id INT(11) AUTO_INCREMENT PRIMARY KEY) COMPRESSION='ZLIB';	支持	支持	
		hash	CREATE TABLE [IF NOT EXISTS] tbl_name (col1 INT, col2 CHAR(5)) PARTITION BY HASH(col1); CREATE TABLE [IF NOT EXISTS] tbl_name (col1 INT, col2 CHAR(5), col3 DATETIME) PARTITION BY HASH (YEAR(col3));	支持	支持	
		key	CREATE TABLE [IF NOT EXISTS] tbl_name (col1 INT, col2 CHAR(5), col3 DATE) PARTITION BY KEY(col3) PARTITIONS 4;	支持	支持	
		linear key	CREATE TABLE [IF NOT EXISTS] tbl_name (col1 INT, col2 CHAR(5), col3 DATE) PARTITION BY LINEAR KEY(col3) PARTITIONS 5;	支持	支持	
		range	CREATE TABLE [IF NOT EXISTS] tbl_name (year_col INT, some_data INT) PARTITION BY RANGE (year_col) (PARTITION p0 VALUES LESS THAN (1991), PARTITION p1 VALUES	支持	支持	

	table partitioning		LESS THAN (2020), PARTITION p5 VALUES LESS THAN MAXVALUE);			
		range columns	CREATE TABLE [IF NOT EXISTS] tbl_name (a INT NOT NULL, b INT NOT NULL) PARTITION BY RANGE COLUMNS(a,b) (PARTITION p1 VALUES LESS THAN (20,10), PARTITION p2 VALUES LESS THAN (50,MAXVALUE), PARTITION p3 VALUES LESS THAN (65,MAXVALUE), PARTITION p4 VALUES LESS THAN (MAXVALUE,MAXVALUE));	支持	支持	
		list	CREATE TABLE [IF NOT EXISTS] tbl_name (id INT, name VARCHAR(35)) PARTITION BY LIST (id) (PARTITION r0 VALUES IN (1, 5, 9, 13, 17, 21), PARTITION r1 VALUES IN (2, 6, 10, 14, 18, 22), PARTITION r2 VALUES IN (3, 7, 11, 15, 19, 23), PARTITION r3 VALUES IN (4, 8, 12, 16, 20, 24));	支持	支持	

	like statement	tracked to tracked	CREATE TABLE new_1 LIKE old_1;	支持	支持	新表在目标端创建，后续
						{
						"replicate_do_db": [
						{
						"table_schema": "action_db
						"tables": [
						{
						"table_name": "old_1"
						},
						{
						"table_name": "new_1"
						}
						]
						}
						]
						}

		tracked to not tracked	CREATE TABLE new_2 LIKE old_2;	支持	支持	新表不会在目标端创建
						
						{
						"replicate_do_db": [
						{
						"table_schema": "action_db
						"tables": [
						{
						"table_name": "old_2"
						}
						]
						}
						]
						}

						新表不会在目标端创建， old_3不在目标端，无法 { "replicate_do_db": [{ "table_schema": "action_db

		tracked	old_3;	持	持	"tables": [
						{
						"table_name": "new_3"
						}
						]
						}
						]
						}

		not tracked to not tracked	CREATE TABLE new_4 LIKE old_4;	支持	支持	新表不会在目标端创建
						
						{
						"replicate_do_db": [
						{
						"table_schema": "action_dl
						"tables"": [
						{
						"table_name": "old_1"
						}
						]
						}
						]
						}

	table options	engine	ALTER TABLE tbl_name ENGINE=InnoDB;	/	支持	
		auto_increment	ALTER TABLE tbl_name AUTO_INCREMENT=100;	/	支持	
		character set	ALTER TABLE tbl_name CHARACTER SET=utf8;	/	支持	
		collate	ALTER TABLE tbl_name COLLATE=utf8_general_ci;	/	支持	
		checksum	ALTER TABLE tbl_name CHECKSUM=1;	/	支持	
		comment	ALTER TABLE tbl_name COMMENT='';	/	支持	
		compression	ALTER TABLE tbl_name COMPRESSION='ZLIB';	/	支持	
	adding and dropping columns	add	ALTER TABLE tbl_name ADD d CHAR(20);	/	支持	
		add after	ALTER TABLE tbl_name ADD d CHAR(20) AFTER b;	/	支持	
		add first	ALTER TABLE tbl_name ADD d CHAR(20) FIRST;	/	支持	
		drop	ALTER TABLE tbl_name DROP b;	/	支持	
		add multiple	ALTER TABLE tbl_name ADD (d CHAR(20), e INT(11));	/	支持	
		drop multiple	ALTER TABLE tbl_name DROP b, DROP COLUMN c;	/	支持	
		change name	ALTER TABLE tbl_name CHANGE COLUMN b d INT(11) DEFAULT NULL;	/	支持	
		change type	ALTER TABLE tbl_name CHANGE b b BIGINT DEFAULT NULL;	/	支持	
		modify type	ALTER TABLE tbl_name MODIFY COLUMN b BIGINT	/	支	

	renaming, redefining, and reordering columns		DEFAULT NULL;		持	
		chang multiple	ALTER TABLE tbl_name CHANGE b d INT(11) DEFAULT NULL, CHANGE c e BIGINT DEFAULT NULL;	/	支持	
		modify multiple	ALTER TABLE tbl_name MODIFY b BIGINT DEFAULT NULL, MODIFY c BIGINT DEFAULT NULL;	/	支持	
		modify character set	ALTER TABLE tbl_name MODIFY b TEXT CHARACTER SET utf8;	/	支持	
	primary keys and indexes	add primary key	ALTER TABLE tbl_name ADD CONSTRAINT PRIMARY KEY (key_part); ALTER TABLE tbl_name ADD PRIMARY KEY (key_part, key_part);	/	支持	
		drop primary key	ALTER TABLE tbl_name DROP PRIMARY KEY;	/	支持	
		add key	ALTER TABLE tbl_name ADD KEY (key_part); ALTER TABLE tbl_name ADD INDEX index_name (key_part, key_part);	/	支持	
		drop key	ALTER TABLE tbl_name DROP PRIMARY KEY; ALTER TABLE tbl_name DROP INDEX index_name;	/	支持	
		add unique key	ALTER TABLE tbl_name ADD UNIQUE INDEX (key_part); ALTER TABLE tbl_name ADD CONSTRAINT UNIQUE KEY index_name (key_part, key_part);	/	支持	
		drop unique key	ALTER TABLE tbl_name DROP KEY id; ALTER TABLE tbl_name DROP INDEX index_name;	/	支持	
alter table		add foreign key	ALTER TABLE tbl_name ADD CONSTRAINT symbol FOREIGN KEY (col_name) REFERENCES tbl_name(key_part) ON DELETE reference_option ON UPDATE reference_option; ALTER TABLE tbl_name ADD CONSTRAINT symbol FOREIGN KEY (col_name) REFERENCES tbl_name(key_part) ON DELETE reference_option ON UPDATE reference_option; ALTER TABLE tbl_name ADD CONSTRAINT FOREIGN	/	支持	应满足一下配置，否则在 数据一致性 1.目标端数据库@@forei

	foreign keys		REFERENCES tbl_name(key_part) ON DELETE reference_option ON UPDATE reference_option; ALTER TABLE tbl_name ADD FOREIGN KEY (col_name) REFERENCES tbl_name(key_part) ON DELETE reference_option ON UPDATE reference_option; ALTER TABLE tbl_name ADD FOREIGN KEY (col_name) REFERENCES tbl_name(key_part);			2.dtle job 中ForeignKeyC
		drop foreign key	ALTER TABLE tbl_name DROP FOREIGN KEY fk_child_5_1;	/	支持	无法和MTS同时使用：https://github.com/actionte 961786003
	changing the character set	modify character set	ALTER TABLE tbl_name MODIFY b TEXT CHARACTER SET utf8;	/	支持	
		convert to	ALTER TABLE tbl_name CONVERT TO CHARACTER SET utf8;	/	支持	
	rename	rename index	ALTER TABLE tbl_name RENAME INDEX index_name_old to index_name_new;	/	支持	
		rename key	ALTER TABLE tbl_name RENAME KEY index_name_old toindex_name_new;	/	支持	

		tracked to tracked	ALTER TABLE old_1 RENAME TO new_1;	/	支持	重命名成功，后续数据
						
						{
						"replicate_do_db": [
						{
						"table_schema": "action_d
						"tables": [
						{
						"table_name": "old_1"
						},
						{
						"table_name": "new_1"
						}
						]
						}
						]
						}

		tracked to not tracked	ALTER TABLE old_2 RENAME AS new_2;	/	支持	重命名成功，后续数不
						
						{
						"replicate_do_db": [
						{
						"table_schema": "action_d
						"tables": [
						{
						"table_name": "old_2"
						}
						]
						}
						]
						}

						}

		not tracked to tracked	ALTER TABLE old_3 RENAME AS new_3;	/	支持	因目标端没有改名前的表
						old_3不在目标端，无法
						
						{
						"replicate_do_db": [
						{
						"table_schema": "action_db
						"tables": [
						{
						"table_name": "new_3"
						}
						]
						}
						]
						}

		not tracked to not tracked	ALTER TABLE old_4 RENAME TO new_4;	/	支持	目标端不会有重命名后的
						
						{
						"replicate_do_db": [
						{
						"table_schema": "action_db
						"tables": [
						{
						"table_name": "old_1"
						}
						]
						}
						]
						}

	table partitioning	ADD PARTITION	ALTER TABLE tbl_name PARTITION BY HASH(expr) PARTITIONS num; ALTER TABLE tbl_name ADD PARTITION (PARTITION partition_names VALUES LESS THAN (MAXVALUE));	/	支持	
		DROP PARTITION	ALTER TABLE tbl_name DROP PARTITION partition_names;	/	支持	
drop table		basic	DROP TABLE tbl_name;	/	支持	
		if exists	DROP TABLE IF EXISTS tbl_name RESTRICT; DROP TABLE IF EXISTS not_exists_tbl_name CASCADE;	/	支持	
		drop multiple	DROP TABLE IF EXISTS not_exists_tbl_name CASCADE; DROP TABLE IF EXISTS tbl_name_1, tbl_name_2, not_exists_tbl_name;	/	支持	

以INDEX为对象的DDL

DDL类型		DDL语句示例	全量	增量	备注
CREATE INDEX	basic	CREATE INDEX key_t1_1 ON t1 (id);	支持	支持	
	prefix key	CREATE INDEX key_t2_1 ON t2 (a(2)); CREATE UNIQUE INDEX key_t3_1 ON t3 (a(3));	支持	支持	
	unique	CREATE UNIQUE INDEX key_t4_1 ON t4 (id);	支持	支持	
	unique not null	CREATE UNIQUE INDEX key_t5_1 ON t5 (id);	支持	支持	
	unique multiple	CREATE UNIQUE INDEX key_t6_1 ON t6 (a, b);	支持	支持	
	fulltext	CREATE FULLTEXT INDEX full_t7_1 ON t7 (a);	支持	支持	
	index_type	CREATE INDEX key_t8_1 ON t8 (id) USING BTREE;	支持	支持	
	comment	CREATE INDEX key_t9_1 ON t9 (id) COMMENT 'test comment';	支持	支持	
DROP INDEX		DROP INDEX key_t1_1 ON t1;	支持	支持	

其他对象**DDL**

DDL类型		DDL语句示例	全量	增量	备注

RENAME	tracked to tracked	RENAME TABLE old_1 to new_1;	/	支持	重命名成功，后续数据传输正常
					
					{
					"replicate_do_db": [
					{
					"table_schema": "action_db_1",
					"tables": [
					{
					"table_name": "old_1"
					},
					{
					"table_name": "new_1"
					}
					]
					}
					]
					}"

	tracked to not tracked	RENAME TABLE old_2 to new_2;	/	支持	重命名成功，后续数不应据传到目标端
					
					{
					"replicate_do_db": [
					{
					"table_schema": "action_db_1",
					"tables": [
					{
					"table_name": "old_2"
					}
					]
					}
					]
					}

	not tracked to tracked	RENAME TABLE old_3 to new_3;	/	支持	因目标端没有改名前的表，会有ERROR log
					old_3不在目标端，无法执行该语句
					
					{
					"replicate_do_db": [
					{
					"table_schema": "action_db_1",
					"tables": [
					{
					"table_name": "new_3"
					}
					]
					}
					]
					}

	not tracked to not tracked	RENAME TABLE old_4 to new_4;	/	支持	目标端不会有重命名后的表
					
					{
					""replicate_do_db"": [
					{
					""table_schema"": ""action_db_1"",
					""tables"": [
					{
					""table_name"": ""old_1""
					}
					]
					}
					]
					}

	rename multiple	RENAME TABLE old_5 to new_5, old_6 to new_6;	/	支持	
TRUNCATE		TRUNCATE tbl_name; TRUNCATE TABLE tbl_name;	/	支持	
VIEW	CREATE VIEW	CREATE VIEW view_name AS select_statement; CREATE OR REPLACE VIEW view_name AS select_statement;"	不支持	不支持	
	ALTER VIEW	ALTER VIEW view_name (column_list) AS select_statement;	/	不支持	
	DROP VIEW	DROP VIEW view_name RESTRICT; DROP VIEW IF EXISTS view_name RESTRICT; DROP VIEW IF EXISTS view_name CASCADE; DROP VIEW view_name_1, view_name_2;	/	不支持	
	CREATE FUNCTION	CREATE FUNCTION sp_name() RETURNS type characteristic routine_body; CREATE FUNCTION sp_name(func_parameters) RETURNS type characteristic RETURN routine_body;	不支持	支持	1. [MySQL 5]目标端账户需要CREATE ROUTINE, SUPER权限

FUNCTION	ALTER FUNCTION	ALTER FUNCTION sp_name SQL SECURITY DEFINER; ALTER FUNCTION sp_name COMMENT ';	/	支持	CREATE ROUTINE, SET_USER_ID, SYSTEM_USER 权限, SET GLOBAL log_bin_trust_function_creators = ON; 3. job配置 ExpandSyntaxSupport=true 4. 全量不复制function
	DROP FUNCTION	DROP FUNCTION sp_name; DROP FUNCTION IF EXISTS sp_name; DROP FUNCTION IF EXISTS not_exists_sp_name;	/	支持	
PROCEDURE	CREATE PROCEDURE	CREATE PROCEDURE sp_name() characteristic routine_body; CREATE PROCEDURE sp_name(proc_parameters) characteristic routine_body;	不支持	支持	1. [MySQL 5]目标端账户需要 CREATE ROUTINE, SUPER权限 2. [MySQL 8] 目标端账户需要 CREATE ROUTINE, SET_USER_ID, SYSTEM_USER 权限, SET GLOBAL log_bin_trust_function_creators = ON; 3. job配置 ExpandSyntaxSupport=true 4. 全量不复制创建procedure
	ALTER PROCEDURE	ALTER PROCEDURE sp_name SQL SECURITY DEFINER; ALTER PROCEDURE sp_name COMMENT ';	/	支持	
	DROP PROCEDURE	DROP PROCEDURE sp_name; DROP PROCEDURE IF EXISTS sp_name; DROP PROCEDURE IF EXISTS not_exists_sp_name;	/	支持	
EVENT	CREATE EVENT	CREATE EVENT event_name ON SCHEDULE schedule COMMENT '' DO event_body;	不支持	不支持	1.不支持复制event 2.源端event产生的数据会复制到目标端
	ALTER EVENT	ALTER EVENT event_name RENAME TO event_name_new;	/	不支持	
	DROP EVENT	DROP EVENT IF EXISTS not_exists_event_name; DROP EVENT IF EXISTS event_name; DROP EVENT event_name;	/	不支持	
	CREATE TRIGGER	CREATE TRIGGER trigger_name BEFORE INSERT on old FOR EACH ROW trigger_body; CREATE TRIGGER trigger_name AFTER INSERT on old FOR EACH ROW trigger_body; CREATE TRIGGER trigger_name BEFORE UPDATE on old FOR EACH ROW trigger_body; CREATE TRIGGER	不支持	不支持	

TRIGGER		CREATE TRIGGER trigger_name AFTER UPDATE on old FOR EACH ROW trigger_body; CREATE TRIGGER trigger_name BEFORE DELETE on old FOR EACH ROW trigger_body; CREATE TRIGGER trigger_name AFTER DELETE on old FOR EACH ROW trigger_body;	持	持	1.不支持复制trigger 2. 源端trigger产生的数据会复制到目标端
	DROP TRIGGER	DROP TRIGGER trigger_name; DROP TRIGGER IF EXISTS trigger_name; DROP TRIGGER schema_name.trigger_name;	/	不支持	

关于不支持 **Trigger**、**Event**

由于Trigger或Event可能会更改表数据，目标端存在Trigger或Event时，存在二次触发的问题（源端已经触发过一次），会引起数据不一致，故dtle不复制Trigger和Event。源端Trigger/Event变更的数据，会被写进binlog，并由dtle复制到目标端。

DCL支持度

条件及限制

- 创建实例级别迁移
- "ExpandSyntaxSupport": true
- 增量部分DCL的操作会被支持
- 全量部分是否需要支持？即，创建job前,源端已存在的用户是否需要被迁移至目标端？#358
- 若需要执行grant和revoke，则回放用户需要有‘grant option’,回放用户需要有被赋权的权限

DCL类型	语句示例	是否支持
CREATE	create user ...identified by ...	支持
ALTER	alter user ...identified by ...	支持
RENAME	rename user ... to ...	支持
SET PASSWORD	set password for ...='...';	支持
GRANT	grant all on . to 'test'@'%';	支持
REVOKE	revoke insert on . from 'test'@'%';	支持

实例级别job.json配置样例：

```
{
  "job_id": "dcl_expand_syntax_support_true",
  "src_task": {
    "task_name": "src",
    "mysql_src_task_config": {
      "expand_syntax_support": true
    },
    "replicate_do_db": [],
    ...
  },
  "dest_task": {
    "task_name": "dest",
    ...
  }
}
```

dtle mapping

在job配置文件中，Table字段增加若干参数，详情参考4.3 作业配置，使用方法如下

schema mapping

单库**mapping**

job.json中ReplicateDoDb配置：

```
  "ReplicateDoDb":[
                    {
                        "TableSchema":"demo",
                        "TableSchemaRename":"demoRename"
                    }
                ],
```

单库**mapping**结果

```
src : demo
dest: demoRename
```

多库**mapping**

job.json中ReplicateDoDb配置：

```
  "ReplicateDoDb":[
                    {
                        "TableSchemaRegex":"(\\w*)src(\\w*)",
                        "TableSchemaRename":"rename${1}",
                    }
                ],
```

多库**mapping**结果

```
src : test1src,test2src,test3src,cust
dest: renametest1, renametest2, renametest3
```

table mapping

单表**mapping**

job.json中ReplicateDoDb配置：

```
"ReplicateDoDb":[
                    {
                        "TableSchema":"demo",
                        "Tables":[
                            {
                                "TableName":"testDemo",
                                "TableRename":"renameDemo"
                            }
                        ]
                    }
                ],
```

单表**mapping**结果

```
src : demo.testDemo
dest: demo.renameDemo
```

多表**mapping**

job.json中ReplicateDoDb配置:

```
"ReplicateDoDb":[
                    {
                        "TableSchema":"demo",
                        "Tables":[
                            {
                                "TableRegex":"(\\w*)Shard(\\w*)",
                                "TableRename":"${1}Rename"
                            }
                        ]
                    }
                ],
```

多表**mapping**结果

```
src : demo.test1Shard,demo.test2Shard,demo.customer,demo.test3Shard
dest: demo.test1Rename,demo.test2Rename,demo.test3Rename
```

列**mapping**

src tables

```
create table demo.a (id int primary key, a int);
create table demo.b (id int primary key, b int);
```

dst table

```
create table demo.colmap (id int primary key auto_increment, val int);
```

使用 `ColumnMapFrom` 和 `ColumnMapTo` 参数，将表a和表b合并到表colmap。忽略原id，使用新的自增id作为主键。

注意：不支持自动创建目标表，需预先手动创建。

```
"ReplicateDoDb": [{
    "TableSchema":"demo",
    "Tables":[{
        "TableName": "a",
        "TableRename": "colmap",
        "ColumnMapFrom": ["a"],
        "ColumnMapTo": ["val"]
    }, {
        "TableName": "b",
        "TableRename": "colmap",
        "ColumnMapFrom": ["b"],
        "ColumnMapTo": ["val"]
    }]
}],
"SkipCreateDbTable": true,
"DropTableIfExists": false,
```

参数说明

- ColumnMapFrom: 从源表中, 依照指定的顺序, 提取全部列或部分列.
- ColumnMapTo: 写入目标表时, 指定写入的列. 可为目标表全部列或部分列.
- From和To的列数量必须相等.
  - 只填写ColumnMapFrom的用法现已deprecated.
- 对于TwoWaySync双向任务, 反向任务会交换正向任务的ColumnMapFrom/ColumnMapTo

暂不支持使用正则表达式匹配列。

Binlog Relay (中继)

背景

- 某些MySQL部署会定期清除binlog
- dtle增量复制依赖binlog，如果binlog被清除则复制会出错
  - dtle全量标记增量开始位置, 若全量耗时较长, 开始增量时binlog极有可能被清除
- 需要在开始全量时将MySQL binlog暂存到dtle本地

使用

在job.json源端任务配置中将 `BinlogRelay` 设为 `true`

```
    "Type": "Src",
    "Config": {
        "BinlogRelay": true,
        "Gtid": "",
```

对于纯增量job，开启BinlogRelay时，必须用Gtid指定复制起点（进度），不能使用BinlogFile/Pos。

参数说明详见作业配置.

影响

binlog储存位置为 `nomad_data_dir/binlog/job_name/mysql_server_uuid` 。一般情况job被删除时会自动清除binlog目录。若未清除则需手动清除。

consul 上的 job 数据管理

dtle 3.x作为nomad插件运行，并且需要consul伴随执行。

部分job信息储存在了consul上。其中最重要的是进度，即 Gtid 或 BinlogFile & Pos。

查看方法（以默认consul地址为例）：

```
# 使用raw查看原始值
$ curl -XGET "127.0.0.1:8500/v1/kv/dtle/job_name/Gtid?raw"
acd7d195-06cd-11e9-928f-02000aba3e28:1-143934

$ curl -XGET "127.0.0.1:8500/v1/kv/dtle/job_name/BinlogPos?raw"
bin.000075//dtle//11909
```

- 注意Gtid可能有多行，需要完整记录。
- BinlogFile & Pos 使用 `//dtle//` 分割。

为了使用户能够记录进度，job删除后，dtle不会自动删除consul上的信息.

重建Job时，若consul上已有进度，则会使用consul上的进度（而非job配置中的起点）。

已删除的Job需要自行删除consul上的信息:

```
# 使用recurse删除job_name下所有项目
$ curl -XDELETE "127.0.0.1:8500/v1/kv/dtle/job_name?recurse"
```

或者使用浏览器访问 127.0.0.1:8500, 使用Web UI管理。

CDC场景

全量复制

- 任务启动时间点开始，将指定库表结构数据传输到目标端

全量流程

1. 获取当前所需同步的库/表,从服务器上的redo日志获取当前系统改变号(SCN)的位置
2. 获取同步表的ROW SHARE LOCK，以防止在创建快照期间表结构发生变化
3. 获取同步的库/表的结构信息,同步到目标端
4. 释放ROW SHARE LOCK
5. 依据步骤3读取的SCN位置，全表扫描所有相关数据库表和schema

```
例:
SELECT * FROM SCHEMA.TABLE AS OF SCN 123  where ROWNUM <= 2000
minus
SELECT * FROM SCHEMA.TABLE AS OF SCN 123  where ROWNUM < 1
```

6. 传输完所有的表数据，继续增量同步

限制

全量同步过程，表结构同步完成前，不支持对同步的表做DDL操作

增量复制

- 根据SCN节点开启增量复制
- 从任务启动时间开启增量复制

DML支持

DML类型

DML类型	option	Oracle SQL	MySQL SQL	
INSERT		INSERT INTO TEST.CHARACTER_256_COLUMNS VALUES (4, NULL);	`replace into` `TEST`.`CHAR_256_COLUMNS`(`COL1`, `COL2`)values(?, ?)	args=[0,]
UPDATE		UPDATE TEST.CHAR_20000_COLUMNS SET COL2='a a b ' WHERE COL1=2;	`update` `TEST`.`CHAR_256_COLUMNS` set`COL1`=?, `COL2`=? where((`COL1` = ?) and (`COL2` = ?)) limit 1	args=[3, "a a", 3, "a
DELETE		DELETE FROM TEST.CHAR_256_COLUMNS WHERE COL1 = 5;	`delete from` `TEST`.`CHAR_256_COLUMNS` where((`COL1` = ?) and (`COL2` = ?)) limit 1	args=[5, "ABCDEFGHIJKL "]

函数支持

函数名	是否支持	其他
EMPTY_BLOB	是	函数支持解析为NULL
EMPTY_CLOB	是	函数支持解析为NULL
CHR	是	
HEXTORAW	是	
DATE	是	
TO_DATE	是	
TO_DSINTERVAL	是	
TO_YMINTERVAL	是	
RAWTOHEX	是	
UNISTR	是	
RAWTOHEX(CHR(34))	是	
TO_TIMESTAMP	是	
LOCALTIMESTAMP	是	
CURRENT_TIMESTAMP	是	
SYSTIMESTAMP	是	

DDL支持

SQL类型	Option	Oracle SQL	转化后MySQL SQL	语法支持
CREATE TABLE	不带约束	CREATE TABLE "test"."CaseInsensitive" ("firstName" VARCHAR(15) NOT NULL,lastName VARCHAR2(45) NOT NULL)	`CREATE TABLE `test`.`CaseInsensitive` (`firstName` VARCHAR(15) NOT NULL,`LASTNAME` VARCHAR(45) NOT NULL) DEFAULT CHARACTER SET = UTF8MB4`	支持
CREATE TABLE	带约束	CREATE TABLE TEST.employees_demo(employee_id NUMBER(6), last_name VARCHAR2(25) CONSTRAINT emp_last_name_nn_demo NOT NULL, CONSTRAINT emp_id_uk_demo UNIQUE (employee_id))	`CREATE TABLE `TEST`.`EMPLOYEES_DEMO` (`EMPLOYEE_ID` INT,`LAST_NAME` VARCHAR(25) NOT NULL,UNIQUE `EMP_ID_UK_DEMO`(`employee_id`)) DEFAULT CHARACTER SET = UTF8MB4`	不支持外键约束
ALTER TABLE	AddColumnClase	alter table TEST.ADDCOLUMN add (author_last_published date);	`ALTER TABLE `TEST`.`ADDCOLUMN` ADD COLUMN (`AUTHOR_LAST_PUBLISHED` DATETIME)`	支持
ALTER TABLE	ModifyColumnClause	ALTER TABLE test."MODIFYCOLUMN" MODIFY (alter_new_name1 CHAR (13)) MODIFY (alter_name2 VARCHAR (66))	`ALTER TABLE `TEST`.`MODIFYCOLUMN` MODIFY COLUMN `ALTER_NEW_NAME1` CHAR(13), MODIFY COLUMN `ALTER_NAME2` VARCHAR(66)`	支持
ALTER TABLE	DropColumnClause	alter table TEST.DROPCOLUMN1 drop column COL1	`ALTER TABLE `TEST`.`DROPCOLUMN1` DROP COLUMN `TEST`.`DROPCOLUMN1`.`COL1`	支持
ALTER TABLE	RenameColumnClase	alter table TEST.RENAMECOLUMN RENAME COLUMN COL1 TO COLNEW1	`ALTER TABLE `TEST`.`RENAMECOLUMN` RENAME COLUMN `TEST`.`RENAMECOLUMN`.`COL1` TO `TEST`.`RENAMECOLUMN`.`COLNEW1``	当前仅支持8.0语法
DROP TABLE		DROP TABLE TEST.DROPTABLE	`DROP TABLE `TEST`.`DROPTABLE``	
create schema/create user				实现为执行create table 前先执行create schema if not exists，保持库同步 #840

下个版本支持功能

- [ ] 支持 索引同步
- [ ] 同步LOB_WRITE，LOB_TRIM， LOB_ERASE，SEL_LOB_LOCATOR 事件
- [ ] 支持PDB（多租户，oracle 12开始支持）
- [ ] DTLE Oracle extractor 通过 SQL driver 轮询读取的间隔目前写死的5秒，优化为动态数值
- [ ] DTLE Oracle extractor 通过 SQL driver 轮询的SCN区间目前写死的100000，优化为动态数值

安装步骤

> 从 dtle 3.x 版本开始，dtle更改了架构，作为nomad插件运行（而非此前的单一二进制文件），并需要运行 consul 以储存任务元数据。
>
> dtle docker image 已包含nomad。consul可使用其官方image。
>
> 标准rpm安装包已集成 consul 和 nomad 及启动脚本和参考配置。

基于容器使用

```
docker pull consul:latest
docker pull actiontech/dtle:latest
```

使用方法参见 快速开始 一节

容器的版本列表参看docker hub

基于**rpm**包的安装

从此处下载dtle的 rpm 安装包, 并执行以下命令可安装dtle

```
rpm -ivh --prefix /opt/dtle dtle-<version>.rpm
```

配置文件位于

- `/opt/dtle/etc/dtle/`

服务启动命令:

```
systemctl start dtle-consul dtle-nomad
systemctl enable dtle-consul dtle-nomad # 开机自动启动
```

日志文件位于 `/opt/dtle/var/log/nomad/`

节点配置

安装包默认将参考配置装在了如下位置(安装时未设置--prefix的情况)

- /etc/consul
- /etc/nomad

使用多节点部署时，请注意更改 `node_name` 、 `data_dir` 、各类地址和端口, 避免冲突。

默认的启动脚本（systemd）使用单节点配置。

- consul 全部配置 https://www.consul.io/docs/agent/options.html#configuration_files
- nomad（本体）全部配置 https://www.nomadproject.io/docs/configuration/

nomad 分为 server 和 client。一个nomad进程可以同时作为server和client，也可以只担任一个角色。 dtle 插件运行在 nomad client 中。

nomad 中 **dtle** 插件的配置

参考样例配置中这一段

```
plugin "dtle" {
  config {
    ...
  }
}
```

配置项	类型	默认值	强制要求	说明
log_level	string	"INFO"	否	日志级别（由于dtle plugin无法获取nomad日志级别，此处需额外设置）
log_file	string	"/var/log/dtle/dtle.log"	否	从4.22.09.0开始, dtle单独生成日志，不再和nomad合并。 每512MB进行rotate和压缩，生成文件形如dtle-2022-11-04T06-46-39.502.log.gz
big_tx_max_jobs	int	取决于启动时的可用内存	否	允许同时处理大事务的job数量。默认值：启动时可用内存/2G。该值至少为1。如有6G可用内存，则该值默认为3
nats_bind	string	"0.0.0.0:8193"	否	Nats (dtle使用的传输协议) 地址
nats_advertise	string	127.0.0.1:8193	否	Nats Advertise 地址, 其他节点使用此地址连接本节点。跨公网传输需要设成上层路由器地址并设置网络穿透
api_addr	string	"" (参考配置中开启)	否	兼容层地址，可以在此地址使用dtle 2.x的HTTP API。参考值："0.0.0.0:8190"。为空则关闭兼容层。
nomad_addr	string	"127.0.0.1:4646"	否	nomad 地址. 由于nomad插件API限制, dtle 无法自动获取该地址, 需要用户手动重复填写一遍.
consul	string	"127.0.0.1:8500"	否	consul的地址, 同nomad本体配置中的. 应填写和最近nomad server关联的consul地址. dtle插件需要consul以储存任务信息
data_dir	string	"/var/lib/nomad"	否	数据目录。目前用于存放binlog（job配置中BinlogRelay=true时）
stats_collection_interval	int	15	否	收集监控项的周期（秒）
publish_metrics	bool	false	否	是否输出监控项
rsa_private_key_path	string	""	否	指定rsa私钥文件的绝对路径，目前只在HTTP api中用于对mysql密码解码。（具体用法见dtle 3.x HTTP API 说明）
cert_file_path	string	""	否	指定证书文件的绝对路径
key_file_path	string	""	否	指定私钥文件的绝对路径

关于 (Bind) Address 和 Advertise Address

- bind address为，需要是本地网卡配置的地址
- advertise addr为对外告知连接用的地址
  - 对于跨网段的nomad集群，需要配置上层路由地址并在各级路由配置NAT（端口映射）

修改日志级别

从4.22.09.0开始，动态修改日志级别直接调用API即可生效(不需要事先修改配置文件或重启dtle)。

```
curl -XPOST http://127.0.0.1:8190/v2/log/level -d "dtle_log_level=INFO"
```

注：dtle后续重启时，仍然使用配置文件中的日志级别。

命令说明

dtle二进制文件仅作为nomad插件使用。各项功能通过 `nomad` 二进制执行。

启动**nomad**节点

```
nomad agent -config=/path/to/nomad.hcl
```

集群相关

```
# 查看管理（server）节点
nomad server members
nomad server members -address=http://127.0.0.1:4646

# 查看执行（client）节点，即运行dtle插件的节点
nomad node status
nomad node status -address=http://127.0.0.1:4646

# 查看某个节点的状态
nomad node status <node ID>
```

此时nomad命令作为HTTP客户端连接nomad agent, 如果agent不在默认地址，则需要指定 `-address=...` , 下同。

job相关

```
# 增加
nomad job run job.hcl
nomad job run -address="http://127.0.0.1:4646" job.hcl

# 删除
nomad job stop -purge <job name>

# 查看所有
nomad job status

# 查看某个
nomad job status <job name>
nomad job status -address=http://127.0.0.1:4646 <job name>
```

查看版本

查看nomad本体版本

```
nomad version
```

查看某一节点的dtle插件版本

```
nomad node status -verbose <node ID> | grep dtle
```

输出

```
dtle         true      true     Healthy   2020-10-09T14:05:00+08:00
driver.dtle               = 1
driver.dtle.full_version  = 9.9.9.9-binlog-provider-7d5a0766
driver.dtle.version       = 9.9.9.9
```

作业(job)配置

作业配置一般采用 json (HTTP API 提交)或 hcl (nomad 命令行工具提交)文件。样例配置在 `/usr/share/dtle/scripts/` 中。

nomad job 的完整配置参考 https://www.nomadproject.io/docs/job-specification/

nomad job 有group/task层级，一个group中的tasks会被放在同一个节点执行。dtle要求src和dest task分别放在src 和 dest group. task 中指定 driver = "dtle", 在config段落中填写dtle专有配置。

从4.22.11.0开始，dtle配置发生变化

- 所有常规配置填在源端任务(src task)
  - 原两端的 `ConnectionConfig` 的分别重命名为 `SrcConnectionConfig` 和 `DestConnectionConfig`
- 目标端固定填写一个配置项 `DestType`

```
group "dest" {
  task "dest" {
    driver = "dtle"
    config {
      DestType = "mysql" # 或"kafka"
    }
  }
}
```

从3.x ~ 4.22.07.x升级到4.22.11后, 可使用 `/usr/share/dtle/scripts/dtle-7to11.py` 更新现有job配置格式.

```
./dtle-7to11.py 'http://127.0.0.1:4646'
```

dtle 源端任务有如下配置项:

参数名	必填?	类型	默认值	说明
Gtid	否	String	默认为全量+增量 任务	MySQL的GTID集合(区间), 可取值: 1. 默认为空, 则为 <全量+增量> 复制任务 2. 已复制的GTID集合(不是点位), 将从未复制的GTID开始增量复制
GtidStart	否	String		增量复制开始的 GTID 点位. (将自动求差集获取上述 GTID 集合.) 需要保持 Gtid 为空
AutoGtid	否	Bool	false	设为 true 后自动从当前 GTID 开始增量任务. 需要保持 Gtid 和 GtidStart 为空.
BinlogRelay	否	Bool	false	是否使用Binlog Relay(中继)机制. 即先将源端mysql binlog读到本地, 避免源端清除binlog导致任务失败. 注意: 如果使用带有BinlogRelay的纯增量复制, 必须用Gtid指定复制起点，不能使用BinlogFile/Pos。
BinlogFile	否	String		增量任务开始的Binlog文件(即源端mysql上 `show master status` 的结果).
BinlogPos	否	Int	0	增量任务开始的Binlog位置, 和BinlogFile配套使用.
ReplicateDoDb	否	Object数组	-	如为空 `[]`, 则复制整个数据库实例. 可填写多元素. 元素内容见下方说明
ReplicateIgnoreDb	否	Object数组	-	指定要忽略的库表，优先级高于ReplicateDoDb。如为空 `[]`, 则完全执行ReplicateDoDb配置. 可填写多元素. 元素内容见下方说明
SrcConnectionConfig	否	Object	-	MySQL源端信息, 见下方 ConnectionConfig 说明。和 OracleConfig 二选一填写。

DestConnectionConfig	否	Object	-	MySQL目标端信息, 见下方 ConnectionConfig 说明。和 KafkaConfig 二选一填写。
SrcOracleConfig	否	Object	-	Oracle源端信息, 见下方 OracleConfig 说明。和 SrcConnectionConfig 二选一填写。
KafkaConfig	否	Object	-	Kafka目标端信息, 见下方 KafkaConfig 说明。和 DestConnectionConfig 二选一填写。
DropTableIfExists	否	Bool	false	全量复制时, 在目标端删除参与复制的表, 之后由dtle自动创建表结构 (相关参数: `SkipCreateDbTable`). 如果开启此选项, 目标端数据库用户需要有相应表的 `DROP` 权限.
SkipCreateDbTable	否	Bool	false	不为目标库创建复制库和复制表. 如果关闭此选项, 目标端数据库用户需要有相应表的 `CREATE` 权限.
ParallelWorkers	否	Int	1	回放端的并发数. 当值大于1时, 目标端会进行并行回放
UseMySQLDependency	否	Bool	true	默认使用MySQL的并行回放事务依赖关系检测。如果不能开启源端MySQL的WRITESET追踪，可将此设为false，使用dtle的依赖检测。
DependencyHistorySize	否	Int	2500	使用dtle并行复制计算事务依赖时，保存的行数。增大可以潜在地增加并行度，但会更消耗内存。
ForeignKeyChecks	否	Bool	true	3.21.10.0+. 默认开启目标端MySQL连接上的 `@@foreign_key_checks`
ReplChanBufferSize	否	Int	32	复制任务缓存的大小, 单位为事务组数。事务组大小和GroupMaxSize/GroupTimeout有关。
ChunkSize	否	Int	2000	全量复制时, 每次读取-传输-写入的行数
DumpEntryLimit	否	Int	67108864 (64M)	复制时, 读取后分块发送的分块大小。空闲内存较小时需适当调小。适用于大全量/增量大事务
ExpandSyntaxSupport	否	Bool	false	支持复制 用户权限/存储过程DDL/函数DDL
GroupMaxSize	否	Int	1	源端发送数据时, 等待数据包达到一定大小 (`GroupMaxSize` 字节)后发送该包. 单位为字节. 默认值1表示即刻发送数据
GroupTimeout	否	Int	100	源端发送数据时, 等待数据包达到超时时间 (`GroupTimeout` 毫秒)发送该包. 单位为毫秒.
SqlFilter	否	String 数组	[]	是否跳过一些事件, 如 `["NoDMLDelete", "NoDDLDropSchema", "NoDDLDropTable", "NoDDLDropIndex", "NoDDLTruncate"]` 。详见下文。
SlaveNetWriteTimeout	否	Int	28800 (8 小时)	调整MySQL slave线程的超时时间。MySQL默认值为60，太短可能导致断连。太长则会导致异常连接回收不及时。
BulkInsert1	否	Int	4	批量插入第一级数量。见性能调优
BulkInsert2	否	Int	8	批量插入第二级数量。
BulkInsert3	否	Int	128	批量插入第三级数量。
SetGtidNext	否	Bool	false	目标端执行事务前执行 `set gtid_next = ...` , 使源端目标端MySQL事务gtid相同。可用以避免循环复制。需要 `REPLICATION_APPLIER` (MySQL 8.0)或 `SUPER` 权限
TwoWaySync	否	Bool	false	开启双向任务。
TwoWaySyncGtid	否	String	""	反向任务使用的Gtid。当值为"auto"时，从当前 GTID 开始增量。
RetryTxLimit	否	Int	3	当执行发生某些错误时（如：deadlock），重试事务的次数

ReplicateDoDb 每个元素有如下字段:

参数名	必填?	类型	默认值	说明
TableSchema	否	String	-	数据库名
TableSchemaRegex	否	String	-	数据库映射正则表达式，可用于多个数据库重命名
TableSchemaRename	否	String	-	重命名后的数据库名称，当进行多数据库重命名时，支持正则表达式，使用见demo
Tables	否	Object数组	-	可配置多张表, 类型为Table. 若不配置, 则复制指定数据库中的所有表
Table.TableName	否	String	-	表名
Table.Where	否	String	-	只复制满足该条件的数据行. 语法为SQL表达式, 返回值应为布尔值. 可以引用表中的列名.
Table.TableRegex	否	String	-	表名映射匹配正则表达式，用于多个表同时重命名.
Table.TableRename	否	String	-	重命名后的表名，当进行多表重命名时，支持支持正则表达，见demo
Table.ColumnMapFrom	否	String数组	-	列映射（暂不支持正则表达式）。见demo
Table.ColumnMapTo	否	String数组	-	列映射（暂不支持正则表达式）。见demo

注：hcl格式中 `${SOME_TEXT}` 会被认为是变量引用。正则替换中输入此类文字时，则需使用双$符号： `$${SOME_TEXT}` 。

ReplicateIgnoreDb 每个元素有如下字段:

参数名	必填?	类型	默认值	说明
TableSchema	是	String	-	数据库名
Tables	否	Object数组	-	可配置多张表, 类型为Table. 若不配置, 则忽略指定数据库中的所有表
Table.TableName	否	String	-	表名

ConnectionConfig 有如下字段:

参数名	必填?	类型	默认值	说明
Host	是	String	-	数据源地址
Port	是	String	-	数据源端口
User	是	String	-	数据源用户名
Password	是	String	-	数据源密码
Charset	否	String	utf8mb4	数据源的字符集

KafkaConfig 有如下字段:

参数名	必填?	类型	默认值	说明
Topic	是	String	-	Kafka Topic
SchemaChangeTopic	否	String	"schema-changes.Topic"	Schema change (DDL) 消息使用的topic
TopicWithSchemaTable	否	Bool	true	默认最终topic为 `指定的Topic.库名.表名` , 如果不需要追加库表名，请设为false
Brokers	是	String数组	-	Kafka Brokers, 如 `["127.0.0.1:9192", "..."]`
Converter	否	String	json	Kafka Converter。目前仅支持json
MessageGroupMaxSize	否	int	1	目标端向kafka发送消息时, 等待MySQL事务数据包达到一定大小(MessageGroupMaxSize字节)后将该包序列化并发送. 单位为字节. 默认值1表示即刻发送数据
MessageGroupTimeout	否	int	100	目标端向kafka发送消息时, 等待数据包达到超时时间(MessageGroupTimeout毫秒)发送该包. 单位为毫秒.
User	否	String	-	Kafka SASL.User
Password	否	String	-	Kafka SASL.Password

OracleConfig 有如下字段:

参数名	必填?	类型	默认值	说明
Host	是	String	-	数据源地址
Port	是	String	-	数据源端口
User	是	String	-	数据源用户名
Password	是	String	-	数据源密码
ServiceName	否	String	XE	数据源服务名
Scn	否	int	0	同步起点

SqlFilter注意事项

全部的filter:

- NoDML
- NoDMLInsert, NoDMLDelete, NoDMLUpdate
- NoDDL
- NoDDLCreateSchema, NoDDLCreateTable
- NoDDLDropSchema, NoDDLDropTable, NoDDLDropIndex, NoDDLTruncate
- NoDDLAlterTable
- NoDDLAlterTableAddColumn, NoDDLAlterTableDropColumn
- NoDDLAlterTableModifyColumn, NoDDLAlterTableChangeColumn, NoDDLAlterTableAlterColumn

SqlFilter只能简单过滤相关语句。不会自动转换后续语句。例如

```
-- SqlFilter = ["NoDDLDropTable"]

/** 源端 **/
-- 已有 table a.a (id int primary key)
drop table a.a;
create table a.a (id int primary key, val int);
insert into a.a values (1, 11);

/** 目标端 **/
-- 已有 table a.a (id int primary key)
-- drop table 语句被过滤
create table a.a (id int primary key, val int);
-- 执行错误，目标表已存在
insert into a.a values (1, 11);
-- 执行错误，列数目不对
```

用户需自行确保在发生过滤的情况下，后续DML/DDL能正确执行。

nomad job 常用通用配置

constraint

job、group 或 task 级配置。配置后该job/group/task会绑定在指定的节点上执行

```
constraint {
  attribute = "${node.unique.name}"
  value = "nomad3"
}
```

完整参考

- https://www.nomadproject.io/docs/job-specification/constraint
- https://www.nomadproject.io/docs/runtime/interpolation#interpreted_node_vars

resources

task级配置，src/dest task需各自重复。默认值为 `cpu=100`， `memory=300`。 以默认值建立大量轻量级任务，会导致资源不够而pending，可适当调小。

任务的内存消耗和每行大小、事物大小、队列长度有关。注意真实资源消耗，避免OOM。

```
task "src" {
  resources {
    cpu    = 100 # MHz
    memory = 300 # MB
  }
}
```

restart & reschedule

nomad job 默认有如下 restart 和 reschedule 配置

```
restart { # group or task level
  interval = "30m"
  attempts = 2
  delay    = "15s"
  mode     = "fail" # "fail" or "delay"
                    # "delay" 意味着interval过后继续尝试
                    # "fail" 则不再尝试
}
reschedule { # job or group level
 delay          = "30s"
 delay_function = "exponential"
 max_delay      = "1h"
 unlimited      = true
}
```

- 当task报错时，会根据restart配置，30分钟内在同一节点上重启最多两次
  - 即使失败的job被 `stop -purge` 再重新添加，也需要根据restart参数重启
- 2次重启均失败后，会根据reschedule配置，在其他节点上执行

为了避免无限reschedule带来的问题，dtle安装包提供的样例job配置中(`<prefix>/usr/share/dtle/scripts/example.job.*`)，限制reschedule为每半小时1次:

```
reschedule {
  attempts = 1
  interval = "30m"
  unlimited = false
}
# 或json格式
"Reschedule": {
  "Attempts": 1,
  "Interval": 1800000000000,
  "Unlimited": false
}
```

性能调优

部分参数可能影响复制性能。

nomad constraint

限制 task 在某个nomad client节点上执行。当源端目标端MySQL之间网络延迟很大时，应在各个主机/机房设立nomad client，并限制 task 在本地节点上执行，以充分利用dtle的压缩传输。

ReplChanBufferSize

默认60，增量事物队列数量。增大可以降低可能的空等，但同时会占用更多内存。

ChunkSize

默认2000。全量复制时每次选取的行数。增大可以增加吞吐量，但同时会占用更多内存。

GroupMaxSize & GroupTimeout

GroupMaxSize默认值1，即每个事物立刻发送。增大后将等待数据量达到设定值再打包发送多个事务。 可增加传输时压缩率，适合低带宽网络。

设定GroupTimeout可避免数据量不足时等待过久。默认值100(毫秒)。一般设成略小于 ping RTT 的时间值。

增量的并行回放（**MTS**）相关

推荐使用MySQL 5.7.22+ 和 MySQL 8.0 GA 后引入的 WriteSet MTS。在源端MySQL设置

```
set global transaction_write_set_extraction = XXHASH64;
set global binlog_transaction_dependency_tracking = WRITESET;
-- will take effect for new session
```

此后MySQL生成的binlog中将附带TX依赖信息，dtle回放时可以利用依赖信息进行调度。

在dtle dest task config中设置ParallelWorkers，控制增量并行回放线程数。参考值为8～64。

如果因版本和权限问题，不能在源端MySQL上设置WriteSet Tracking，则可以使用dtle的依赖计算功能（`UseMySQLDependency = false`）。

批量插入（**bulk insert**）

当源端使用批量插入，即 `insert into ... values (),(), ..., ()` 时，dtle会在目标端使用批量插入。

dtle会使用两个固定数量（行数）的批量插入PreparedStatement，默认为4和8。超过8的会被分到下一批。小于4的会单独插入。

可用 `BulkInsert1` 和 `BulkInsert2` 调整批量插入使用的数量。

Job 示例

复制整个实例的所有数据库

job.hcl 中ReplicateDoDb配置:

```
ReplicateDoDb = []
```

复制指定数据库

```
ReplicateDoDb = [{
  TableSchema = "action_db_1"
}]
```

复制一个库中的多个表

job.hcl 中ReplicateDoDb配置:

```
ReplicateDoDb = [{
  TableSchema = "action_db_1"
  Tables = [{
    TableName = "sbtest1"
  }, {
    TableName = "sbtest2"
  }, {
    TableName = "sbtest3"
  }]
}]
```

复制多个库中的多个表

job.hcl 中ReplicateDoDb配置:

```
ReplicateDoDb = [{
  TableSchema = "action_db_1"
  Tables = [{
    TableName = "sbtest1"
  }, {
    TableName = "sbtest2"
  }, {
    TableName = "sbtest3"
  }]
}, {
  TableSchema = "action_db_2"
  Tables = [{
    TableName = "sbtest1"
  }, {
    TableName = "sbtest2"
  }, {
    TableName = "sbtest3"
  }]
}]
```

带**where**条件复制任务

参考2.2.MySQL 的数据分散

使用正则挑选复制库表

参考3.8.dtle mapping 支持

忽略指定的库

job.hcl通过以下配置忽略表db1及db1内所有的表

```
ReplicateDoDb = []
ReplicateIgnoreDb = [{
  TableSchema = "db1"
}]
```

job.hcl通过以下配置在ReplicateDoDb指定的范围内忽略表db1和db1下的所有表，最终效果是没有要复制的库表

```
ReplicateDoDb = [{
  TableSchema = "db1"
  Tables = [{
    TableName = "tb1"
  }]
}]
ReplicateIgnoreDb = [{
  TableSchema = "db1"
}]
```

忽略指定的表

job.hcl通过以下配置在ReplicateDoDb指定的范围内忽略db1.tb1，最终复制库db1下除了tb1以外的表

```
ReplicateDoDb = [{
  TableSchema = "db1"
}]
ReplicateIgnoreDb = [{
  TableSchema = "db1"
  Tables = [{
    TableName = "tb1"
  }]
}]
```

job.hcl通过以下配置在ReplicateDoDb指定的范围内忽略db1.tb1，最终只复制库db1结构，但不复制db1下的任何表

```
ReplicateDoDb = [{
  TableSchema = "db1"
  Tables = [{
    TableName = "tb1"
  }]
}]
ReplicateIgnoreDb = [{
  TableSchema = "db1"
  Tables = [{
    TableName = "tb1"
  }]
}]
```

限定故障转移域

源端任务和目标端任务在指定 `datacenter` 上故障转移
dtle配置文件：

```
name = "dtle-1" # rename for each node
datacenter = "shanghai"
...
```

job示例：

```
job "test_constraint" {
  # 此处表示该job可以运行在datacenter为"shanghai"和"beijing"的节点上
  datacenters = ["shanghai", "beijing"]

  group "Src" {
    constraint {
      attribute  = "${node.datacenter}"
      operator   = "="
      value      = "shanghai"
    }
    task "src" {
      driver = "dtle"
      config {
        ReplicateDoDb = [{
          TableSchema = "test"
        }]
        ConnectionConfig = {
            ...
        }
      }
    }
  }
  group "Dest" {
    constraint {
      attribute  = "${node.datacenter}"
      operator   = "="
      value      = "beijing"
    }
    task "dest" {
      driver = "dtle"
      config {
        ConnectionConfig = {
            ...
        }
      }
    }
  }
}
```

HTTP API 说明

(适用dtle 3.x nomad 插件)

nomad 默认开启一个web服务，可使用curl工具向其发送HTTP请求。

作业管理

完整可参考

- https://www.nomadproject.io/api-docs/jobs
- https://www.nomadproject.io/api-docs/allocations

常用如下:

前置知识：**nomad** 中 **job**、**task**、**alloc**的概念

job包含多个task。一个dtle job有src和dest两个task。

task在nomad节点上的执行，称为allocation。同一个task的多次执行（如失败-重试）会创建多个allocation。

列出所有**job**

```
curl -XGET 127.0.0.1:4646/v1/jobs | jq
```

添加**job**

```
curl -XPOST -data @job.json 127.0.0.1:4646/v1/jobs | jq
```

job.json的内容说明参看 作业(job)配置

获取某个**job**信息

```
curl -XGET 127.0.0.1:4646/v1/job/<job_name> | jq
```

列出某**job**的所有**allocation**

```
curl -XGET "127.0.0.1:4646/v1/job/<job_name>/allocations | jq
```

查看某个**allocation**的执行状态

```
curl -XGET "127.0.0.1:4646/v1/allocation/<alloc_id>" | jq
```

区别任务处于全量还是增量状态

```
curl -XGET "127.0.0.1:4646/v1/job/<jobId>/allocations" | jq '.' | grep job_stage
```

```
            "DisplayMessage": "job_stage_full",
            "DriverMessage": "job_stage_full",
            "DisplayMessage": "job_stage_incr",
            "DriverMessage": "job_stage_incr",
```

- 当结果中 只 出现 job_stage_full 时，任务处于全量阶段
- 当结果出现 job_stage_incr 时，任务处于增量阶段

停止（删除）**job**

```
curl -XDELETE 127.0.0.1:4646/v1/job/my-job
# DELETE后job信息仍会在nomad上保留一段时间供查询，直到nomad自动回收（gc）
# 指定purge可立刻删除
curl -XDELETE 127.0.0.1:4646/v1/job/my-job?purge=true
```

dtle 3.x 移除了暂停/恢复job的功能. 可使用删除/添加job来达成相同的效果。

- 注意保留添加job时使用的job配置文件
- job删除后进度（Gtid）仍然保存在consul kv中
  - 位置: `dtle/<job_name>/Gtid`
  - 再次添加job时，consul中保存的Gtid优先于job配置中的项目

如果要后续添加同名job，并且不想从consul保存的位置继续（而是从job配置中指定的位置开始），则 需要删除consul重的数据。见 consul 上的 job 数据管理。

节点管理

更多可参考

- https://www.nomadproject.io/api-docs/nodes
- https://www.nomadproject.io/api-docs/status

列出所有节点：

```
curl -XGET "127.0.0.1:4646/v1/nodes" | jq
[{
    "Address": "127.0.0.1",
    "Datacenter": "dc1",
    "Drivers": {
      "dtle": {
        "Attributes": {
          "driver.dtle": "true",
          "driver.dtle.version": "9.9.9.9",
          "driver.dtle.full_version": "9.9.9.9-master-eeb399e9"
        },
        "Detected": true,
        "Healthy": true,
      }
    },
    "ID": "0e70636d-b274-c139-185e-e37dcf7a4bca",
    "Name": "nomad0",
    "Status": "ready",
    "Version": "0.11.2"
}]
```

可以查看节点名、节点ID和dtle插件信息（部分项省略）。

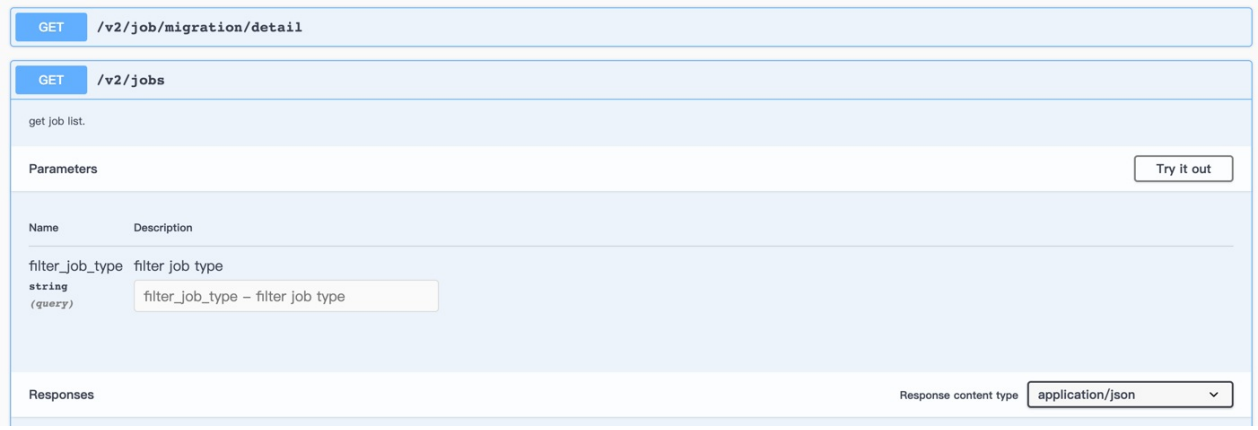
dtle 3.x HTTP API 说明

dtle 3.x 根据业务功能提供了一套HTTP API（开启方式见"节点配置"，api_addr），可与dtle UI配套使用。

本节API示例默认使用swagger UI调用。

通过swagger UI查看接口文档

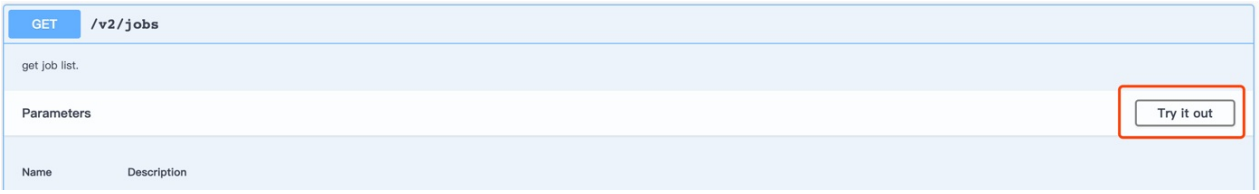
访问 `http://{dtle ip}:8190/swagger/index.html` 通过swagger UI查看接口文档，打开界面如下：



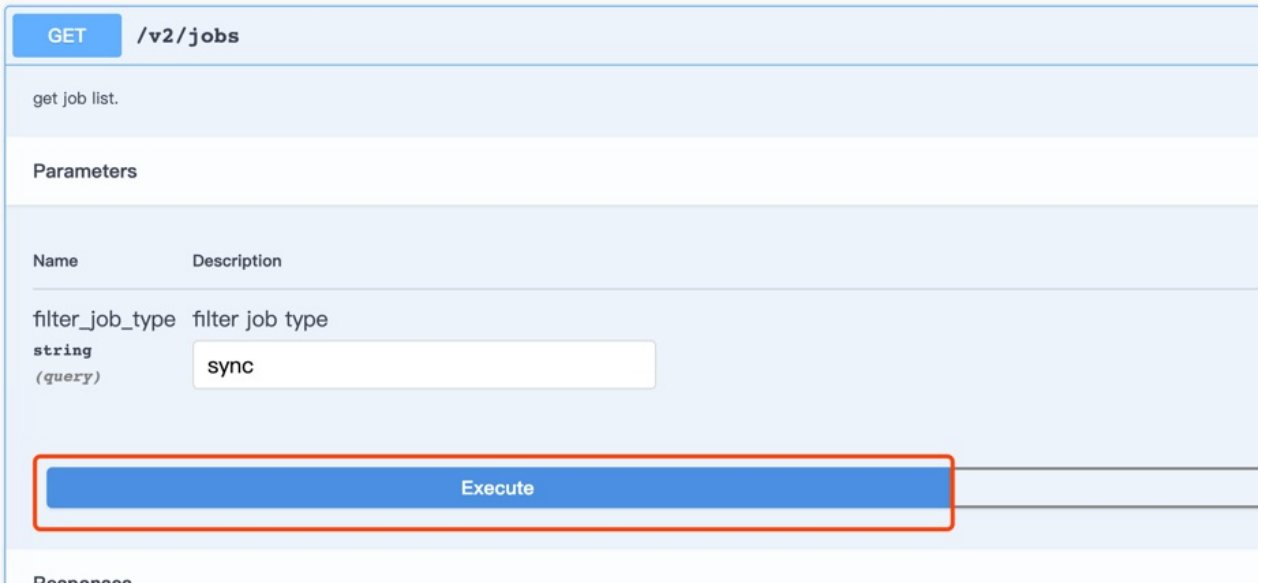
通过swagger UI调用API

除了使用curl命令外，还可以通过swagger UI界面调用API，具体步骤如下：

1 点击"Try it out"，进入调试模式

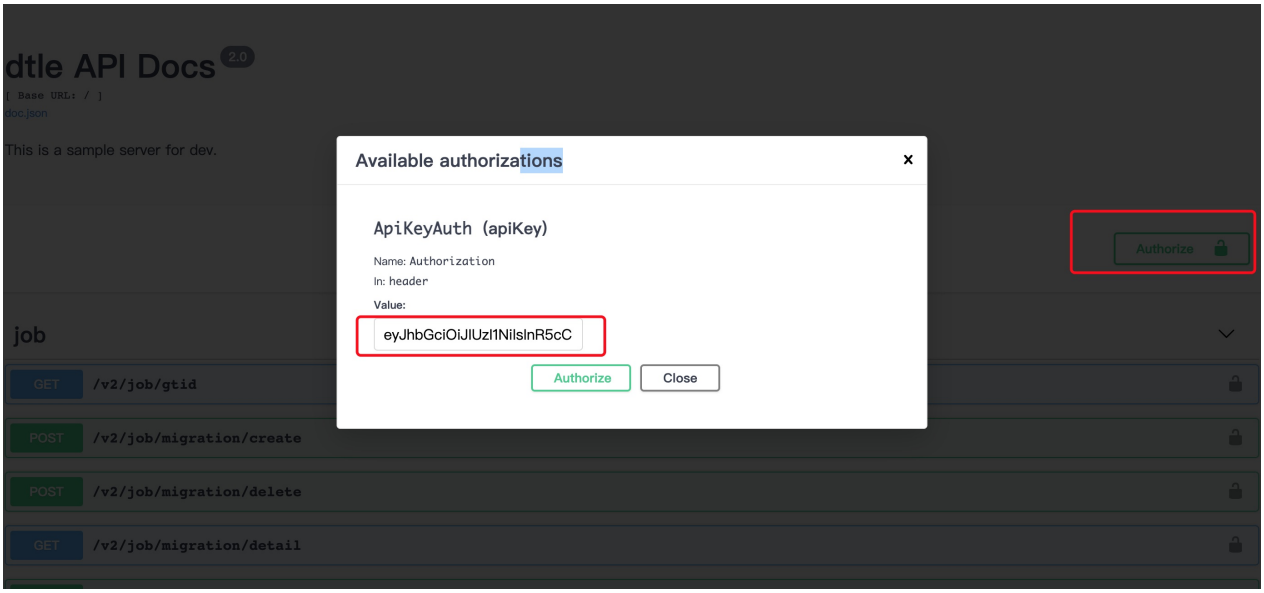


2 填写请求参数后点击"Execute"调用API



3 查看响应:

4 由于用户校验功能限制，大多数接口调用需要在header中携带登录成功返回的token，在swagger页面可点击swagger页面顶部的Authorize按钮，将token填入Value的文本框中，swagger页面中其他接口即可正常使用



dtle API Docs

This is a sample server for dev.

Version: 2.0

Security

ApiKeyAuth

apiKey	*API Key*
In	header
Name	Authorization

/v2/database/columns

GET

Description:

list columns of database source instance.

Parameters

Name	Located in	Description	Required	Schema
host	query	database host	Yes	string
port	query	database port	Yes	integer
user	query	database user	Yes	string
password	query	database password	Yes	string
database_type	query	database_type	Yes	string
service_name	query	database service_name	No	string
schema	query	database schema	Yes	string
table	query	database table	Yes	string
character_set	query	database character set	No	string
is_password_encrypted	query	indecate that database password is encrypted or not	No	boolean

Responses

Code	Description	Schema
200	OK	models.ListColumnsRespV2

Security

Security Schema	Scopes
ApiKeyAuth	

/v2/database/instance_connection

GET

Description:

connect to database instance.

Parameters

Name	Located in	Description	Required	Schema
host	query	database host	Yes	string
port	query	database port	Yes	integer
user	query	database user	Yes	string
password	query	database password	Yes	string
database_type	query	database_type	Yes	string
service_name	query	database service_name	No	string
is_password_encrypted	query	indecate that database password is encrypted or not	No	boolean

Responses

Code	Description	Schema
200	OK	models.ConnectionRespV2

Security

Security Schema	Scopes
ApiKeyAuth	

/v2/database/schemas

GET

Description:

list schemas of database source instance.

Parameters

Name	Located in	Description	Required	Schema
database_type	query	database_type	Yes	string
host	query	database host	Yes	string
port	query	database port	Yes	integer
user	query	database user	Yes	string
password	query	database password	Yes	string
service_name	query	database service_name	No	string
character_set	query	database character set	No	string
is_password_encrypted	query	indecate that database password is encrypted or not	No	boolean

Responses

Code	Description	Schema
200	OK	models.ListSchemasRespV2

Security

Security Schema	Scopes
ApiKeyAuth	

/v2/job/gtid

GET

Description:

get src task current gtid.

Parameters

Name	Located in	Description	Required	Schema
job_id	query	job id	Yes	string

Responses

Code	Description	Schema
200	OK	models.JobGtidResp

Security

Security Schema	Scopes
ApiKeyAuth	

/v2/job/migration/create

POST

Description:

create migration job.

Parameters

Name	Located in	Description	Required	Schema
migration_job_config	body	migration job config	Yes	models.CreateOrUpdateMysqlToMysqlJobParamV2

Responses

Code	Description	Schema
200	OK	models.CreateOrUpdateMysqlToMysqlJobRespV2

Security

Security Schema	Scopes
ApiKeyAuth	

/v2/job/migration/delete

POST

Description:

delete migration job.

Parameters

Name	Located in	Description	Required	Schema
job_id	formData	job id	Yes	string

Responses

Code	Description	Schema
200	OK	models.DeleteJobRespV2

Security

Security Schema	Scopes
ApiKeyAuth	

/v2/job/migration/detail

GET

Description:

get migration job detail.

Parameters

Name	Located in	Description	Required	Schema
job_id	query	job id	Yes	string

Responses

Code	Description	Schema
200	OK	models.MysqlToMysqlJobDetailRespV2

Security

Security Schema	Scopes
ApiKeyAuth	

/v2/job/migration/pause

POST

Description:

pause migration job.

Parameters

Name	Located in	Description	Required	Schema
job_id	formData	job id	Yes	string

Responses

Code	Description	Schema
200	OK	models.PauseJobRespV2

Security

Security Schema	Scopes
ApiKeyAuth	

/v2/job/migration/resume

POST

Description:

resume migration job.

Parameters

Name	Located in	Description	Required	Schema
job_id	formData	job id	Yes	string

Responses

Code	Description	Schema
200	OK	models.ResumeJobRespV2

Security

Security Schema	Scopes
ApiKeyAuth	

/v2/job/migration/reverse

POST

Description:

reverse migration Job

Parameters

Name	Located in	Description	Required	Schema
reverse_config	body	reverse config config	Yes	models.ReverseJobReq

Responses

Code	Description	Schema
200	OK	models.ReverseJobResp

Security

Security Schema	Scopes
ApiKeyAuth	

/v2/job/migration/reverse_start

POST

Summary:

start reverse-init job

Description:

Start Reverse Job.

Parameters

Name	Located in	Description	Required	Schema
job_id	formData	job id	Yes	string

Responses

Code	Description	Schema
200	OK	models.ReverseStartRespV2

Security

Security Schema	Scopes
ApiKeyAuth	

/v2/job/migration/update

POST

Description:

update migration job.

Parameters

Name	Located in	Description	Required	Schema
migration_job_config	body	migration job config	Yes	models.CreateOrUpdateMysqlToMysqlJobParamV2

Responses

Code	Description	Schema
200	OK	models.CreateOrUpdateMysqlToMysqlJobRespV2

Security

Security Schema	Scopes
ApiKeyAuth	

/v2/job/reverse_start

POST

Summary:

start reverse-init job

Description:

Finish Job.

Parameters

Name	Located in	Description	Required	Schema
job_id	formData	job id	Yes	string

Responses

Code	Description	Schema
200	OK	models.ReverseStartRespV2

Security

Security Schema	Scopes
ApiKeyAuth	

/v2/job/subscription/create

POST

Description:

create subscription job.

Parameters

Name	Located in	Description	Required	Schema
subscription_job_config	body	subscription job config	Yes	models.CreateOrUpdateMysqlToKafkaJobParamV2

Responses

Code	Description	Schema
200	OK	models.CreateOrUpdateMysqlToKafkaJobRespV2

Security

Security Schema	Scopes
ApiKeyAuth	

/v2/job/subscription/delete

POST

Description:

delete subscription job.

Parameters

Name	Located in	Description	Required	Schema
job_id	formData	job id	Yes	string

Responses

Code	Description	Schema
200	OK	models.DeleteJobRespV2

Security

Security Schema	Scopes
ApiKeyAuth	

/v2/job/subscription/detail

GET

Description:

get subscription job detail.

Parameters

Name	Located in	Description	Required	Schema
job_id	query	job id	Yes	string

Responses

Code	Description	Schema
200	OK	models.MysqlToKafkaJobDetailRespV2

Security

Security Schema	Scopes
ApiKeyAuth	

/v2/job/subscription/pause

POST

Description:

pause subscription job.

Parameters

Name	Located in	Description	Required	Schema
job_id	formData	job id	Yes	string

Responses

Code	Description	Schema
200	OK	models.PauseJobRespV2

Security

Security Schema	Scopes
ApiKeyAuth	

/v2/job/subscription/resume

POST

Description:

resume subscription job.

Parameters

Name	Located in	Description	Required	Schema
job_id	formData	job id	Yes	string

Responses

Code	Description	Schema
200	OK	models.ResumeJobRespV2

Security

Security Schema	Scopes
ApiKeyAuth	

/v2/job/subscription/update

POST

Description:

update subscription job.

Parameters

Name	Located in	Description	Required	Schema
subscription_job_config	body	subscription job config	Yes	models.CreateOrUpdateMysqlToKafkaJobParamV2

Responses

Code	Description	Schema
200	OK	models.CreateOrUpdateMysqlToKafkaJobRespV2

Security

Security Schema	Scopes
ApiKeyAuth	

/v2/job/sync/create

POST

Description:

create sync job.

Parameters

Name	Located in	Description	Required	Schema
sync_job_config	body	sync job config	Yes	models.CreateOrUpdateMysqlToMysqlJobParamV2

Responses

Code	Description	Schema
200	OK	models.CreateOrUpdateMysqlToMysqlJobRespV2

Security

Security Schema	Scopes
ApiKeyAuth	

/v2/job/sync/delete

POST

Description:

delete sync job.

Parameters

Name	Located in	Description	Required	Schema
job_id	formData	job id	Yes	string

Responses

Code	Description	Schema
200	OK	models.DeleteJobRespV2

Security

Security Schema	Scopes
ApiKeyAuth	

/v2/job/sync/detail

GET

Description:

get sync job detail.

Parameters

Name	Located in	Description	Required	Schema
job_id	query	job id	Yes	string

Responses

Code	Description	Schema
200	OK	models.MysqlToMysqlJobDetailRespV2

Security

Security Schema	Scopes
ApiKeyAuth	

/v2/job/sync/pause

POST

Description:

pause sync job.

Parameters

Name	Located in	Description	Required	Schema
job_id	formData	job id	Yes	string

Responses

Code	Description	Schema
200	OK	models.PauseJobRespV2

Security

Security Schema	Scopes
ApiKeyAuth	

/v2/job/sync/resume

POST

Description:

resume sync job.

Parameters

Name	Located in	Description	Required	Schema
job_id	formData	job id	Yes	string

Responses

Code	Description	Schema
200	OK	models.ResumeJobRespV2

Security

Security Schema	Scopes
ApiKeyAuth	

/v2/job/sync/reverse

POST

Description:

reverse sync Job

Parameters

Name	Located in	Description	Required	Schema
reverse_config	body	reverse config config	Yes	models.ReverseJobReq

Responses

Code	Description	Schema
200	OK	models.ReverseJobResp

Security

Security Schema	Scopes
ApiKeyAuth	

/v2/job/sync/reverse_start

POST

Summary:

start reverse-init job

Description:

Start Reverse Job.

Parameters

Name	Located in	Description	Required	Schema
job_id	formData	job id	Yes	string

Responses

Code	Description	Schema
200	OK	models.ReverseStartRespV2

Security

Security Schema	Scopes
ApiKeyAuth	

/v2/job/sync/update

POST

Description:

update sync job.

Parameters

Name	Located in	Description	Required	Schema
sync_job_config	body	sync job config	Yes	models.CreateOrUpdateMysqlToMysqlJobParamV2

Responses

Code	Description	Schema
200	OK	models.CreateOrUpdateMysqlToMysqlJobRespV2

Security

Security Schema	Scopes
ApiKeyAuth	

/v2/jobs/migration

GET

Description:

get job list.

Parameters

Name	Located in	Description	Required	Schema
filter_job_id	query	filter job id	No	string
filter_job_src_ip	query	filter job src ip	No	string
filter_job_src_port	query	filter job src port	No	string
filter_job_dest_ip	query	filter job dest ip	No	string
filter_job_dest_port	query	filter job dest port	No	string
filter_job_status	query	filter job status	No	string
order_by	query	order by	No	string

Responses

Code	Description	Schema
200	OK	models.JobListRespV2

Security

Security Schema	Scopes
ApiKeyAuth	

/v2/jobs/subscription

GET

Description:

get subscription job list.

Parameters

Name	Located in	Description	Required	Schema
filter_job_id	query	filter job id	No	string
filter_job_src_ip	query	filter job src ip	No	string
filter_job_src_port	query	filter job src port	No	string
filter_job_dest_ip	query	filter job dest ip	No	string
filter_job_dest_port	query	filter job dest port	No	string
filter_job_status	query	filter job status	No	string
order_by	query	order by	No	string

Responses

Code	Description	Schema
200	OK	models.JobListRespV2

Security

Security Schema	Scopes
ApiKeyAuth	

/v2/jobs/sync

GET

Description:

get sync job list.

Parameters

Name	Located in	Description	Required	Schema
filter_job_id	query	filter job id	No	string
filter_job_src_ip	query	filter job src ip	No	string
filter_job_src_port	query	filter job src port	No	string
filter_job_dest_ip	query	filter job dest ip	No	string
filter_job_dest_port	query	filter job dest port	No	string
filter_job_status	query	filter job status	No	string
order_by	query	order by	No	string

Responses

Code	Description	Schema
200	OK	models.JobListRespV2

Security

Security Schema	Scopes
ApiKeyAuth	

/v2/log/level

POST

Description:

reload log level dynamically.

Parameters

Name	Located in	Description	Required	Schema
dtle_log_level	formData	dtle log level	Yes	string

Responses

Code	Description	Schema
200	OK	models.UpdataLogLevelRespV2

Security

Security Schema	Scopes
ApiKeyAuth	

/v2/login

POST

Summary:

user loginV2

Description:

user login

Parameters

Name	Located in	Description	Required	Schema
user	body	user login request	Yes	models.UserLoginReqV2

Responses

Code	Description	Schema
200	OK	models.GetUserLoginResV2

/v2/login/captcha

POST

Summary:

create base64Captcha

Description:

create base64Captcha

Parameters

Name	Located in	Description	Required	Schema
captcha_type	formData	captcha type	Yes	string

Responses

Code	Description	Schema
200	OK	models.CaptchaRespV2

/v2/loginWithoutVerifyCode

POST

Summary:

user LoginWithoutVerifyCodeV2

Description:

user login Without Verify Code

Parameters

Name	Located in	Description	Required	Schema
user	body	user login request	Yes	models.LoginWithoutVerifyCodeReqV2

Responses

Code	Description	Schema
200	OK	models.GetUserLoginResV2

/v2/monitor/task

GET

Description:

get progress of tasks within an allocation.

Parameters

Name	Located in	Description	Required	Schema
allocation_id	query	allocation id	Yes	string
task_name	query	task name	Yes	string
nomad_http_address	query	nomad_http_address is the http address of the nomad that the target dtle is running with. ignore it if you are not sure what to provide	No	string

Responses

Code	Description	Schema
200	OK	models.GetTaskProgressRespV2

Security

Security Schema	Scopes
ApiKeyAuth	

/v2/mysql/columns

GET

Description:

list columns of mysql source instance.

Parameters

Name	Located in	Description	Required	Schema
mysql_host	query	mysql host	Yes	string
mysql_port	query	mysql port	Yes	string
mysql_user	query	mysql user	Yes	string
mysql_password	query	mysql password	Yes	string
mysql_schema	query	mysql schema	Yes	string
mysql_table	query	mysql table	Yes	string
mysql_character_set	query	mysql character set	No	string
is_mysql_password_encrypted	query	indecate that mysql password is encrypted or not	No	boolean

Responses

Code	Description	Schema
200	OK	models.ListColumnsRespV2

Security

Security Schema	Scopes
ApiKeyAuth	

/v2/mysql/instance_connection

GET

Description:

connect to mysql instance.

Parameters

Name	Located in	Description	Required	Schema
mysql_host	query	mysql host	Yes	string
mysql_port	query	mysql port	Yes	string
mysql_user	query	mysql user	Yes	string
mysql_password	query	mysql password	Yes	string
is_mysql_password_encrypted	query	indecate that mysql password is encrypted or not	No	boolean

Responses

Code	Description	Schema
200	OK	models.ConnectionRespV2

Security

Security Schema	Scopes
ApiKeyAuth	

/v2/mysql/schemas

GET

Description:

list schemas of mysql source instance.

Parameters

Name	Located in	Description	Required	Schema
mysql_host	query	mysql host	Yes	string
mysql_port	query	mysql port	Yes	string
mysql_user	query	mysql user	Yes	string
mysql_password	query	mysql password	Yes	string
mysql_character_set	query	mysql character set	No	string
is_mysql_password_encrypted	query	indecate that mysql password is encrypted or not	No	boolean

Responses

Code	Description	Schema
200	OK	models.ListSchemasRespV2

Security

Security Schema	Scopes
ApiKeyAuth	

/v2/nodes

GET

Description:

get node list.

Responses

Code	Description	Schema
200	OK	models.NodeListRespV2

Security

Security Schema	Scopes
ApiKeyAuth	

/v2/role/create

POST

Description:

create Role.

Parameters

Name	Located in	Description	Required	Schema
Role	body	Role info	Yes	models.CreateRoleReqV2

Responses

Code	Description	Schema
200	OK	models.CreateRoleRespV2

Security

Security Schema	Scopes
ApiKeyAuth	

/v2/role/delete

POST

Description:

delete Role.

Parameters

Name	Located in	Description	Required	Schema
tenant	formData	tenant	Yes	string
name	formData	role name	Yes	string

Responses

Code	Description	Schema
200	OK	models.DeleteRoleRespV2

Security

Security Schema	Scopes
ApiKeyAuth	

/v2/role/list

GET

Description:

get role list.

Parameters

Name	Located in	Description	Required	Schema
filter_tenant	query	filter tenant	No	string

Responses

Code	Description	Schema
200	OK	models.RoleListResp

Security

Security Schema	Scopes
ApiKeyAuth	

/v2/role/update

POST

Description:

update Role info.

Parameters

Name	Located in	Description	Required	Schema
Role	body	Role info	Yes	models.UpdateRoleReqV2

Responses

Code	Description	Schema
200	OK	models.UpdateRoleRespV2

Security

Security Schema	Scopes
ApiKeyAuth	

/v2/tenant/list

GET

Description:

get tenant list.

Responses

Code	Description	Schema
200	OK	models.TenantListResp

Security

Security Schema	Scopes
ApiKeyAuth	

/v2/user/create

POST

Description:

create user.

Parameters

Name	Located in	Description	Required	Schema
user	body	user info	Yes	models.CreateUserReqV2

Responses

Code	Description	Schema
200	OK	models.CreateUserRespV2

Security

Security Schema	Scopes
ApiKeyAuth	

/v2/user/current_user

GET

Description:

get current user.

Responses

Code	Description	Schema
200	OK	models.CurrentUserResp

Security

Security Schema	Scopes
ApiKeyAuth	

/v2/user/delete

POST

Description:

delete user.

Parameters

Name	Located in	Description	Required	Schema
tenant	formData	tenant	Yes	string
username	formData	user name	Yes	string

Responses

Code	Description	Schema
200	OK	models.DeleteUserRespV2

Security

Security Schema	Scopes
ApiKeyAuth	

/v2/user/list

GET

Description:

get user list.

Parameters

Name	Located in	Description	Required	Schema
filter_username	query	filter user name	No	string
filter_tenant	query	filter tenant	No	string

Responses

Code	Description	Schema
200	OK	models.UserListResp

Security

Security Schema	Scopes
ApiKeyAuth	

/v2/user/list_action

GET

Description:

list user action.

Responses

Code	Description	Schema
200	OK	models.ListActionRespV2

Security

Security Schema	Scopes
ApiKeyAuth	

/v2/user/reset_password

POST

Description:

reset user password.

Parameters

Name	Located in	Description	Required	Schema
user	body	reset user password	Yes	models.ResetPasswordReqV2

Responses

Code	Description	Schema
200	OK	models.ResetPasswordRespV2

Security

Security Schema	Scopes
ApiKeyAuth	

/v2/user/update

POST

Description:

update user info.

Parameters

Name	Located in	Description	Required	Schema
user	body	user info	Yes	models.UpdateUserReqV2

Responses

Code	Description	Schema
200	OK	models.UpdateUserRespV2

Security

Security Schema	Scopes
ApiKeyAuth	

/v2/validation/job

POST

Description:

validate job config.

Parameters

Name	Located in	Description	Required	Schema
job_config	body	validate job config	Yes	models.ValidateJobReqV2

Responses

Code	Description	Schema
200	OK	models.ValidateJobRespV2

Security

Security Schema	Scopes
ApiKeyAuth	

Models

common.JobListItemV2

Name	Type	Description	Required
dst_addr_list	[string]		No
dst_database_type	string		No
job_create_time	string		No
job_id	string		No
job_status	string		No
job_steps	[common.JobStep]		No
src_addr_list	[string]		No
src_database_type	string		No
topic	string		No
user	string		No

common.JobStep

Name	Type	Description	Required
job_create_time	string		No
step_name	string		No
step_schedule	number		No
step_status	string		No

common.Role

Name	Type	Description	Required
authority	string		No
name	string		No
object_type	string		No
object_users	[string]		No
tenant	string		No

common.User

Name	Type	Description	Required
create_time	string		No
password	string		No
remark	string		No
role	string		No
tenant	string		No
username	string		No

models.BasicTaskProfile

Name	Type	Description	Required
configuration	models.Configuration		No
connection_info	models.ConnectionInfo		No
dtle_node_infos	[models.DtleNodeInfo]		No
job_base_info	models.JobBaseInfo		No
replicate_do_db	[models.DataSourceConfig]		No
replicate_ignore_db	[models.DataSourceConfig]		No

models.BinlogValidation

Name	Type	Description	Required
error	string	Error is a string version of any error that may have occured	No
validated	boolean		No

models.BufferStat

Name	Type	Description	Required
applier_tx_queue_size	integer		No
binlog_event_queue_size	integer		No
extractor_tx_queue_size	integer		No
send_by_size_full	integer		No
send_by_timeout	integer		No

models.ButtonItem

Name	Type	Description	Required
action	string		No
text_cn	string		No
text_en	string		No
uri	string		No

models.CaptchaRespV2

Name	Type	Description	Required
data_scheme	string		No
id	string		No
message	string		No

models.Configuration

Name	Type	Description	Required
dst_config	models.DstConfig		No
fail_over	boolean		No
retry_times	integer		No
src_config	models.SrcConfig		No

models.ConnectionInfo

Name	Type	Description	Required
dst_data_base	models.DatabaseConnectionConfig		No
dst_kafka	models.KafkaDestTaskConfig		No
src_data_base	models.DatabaseConnectionConfig		No

models.ConnectionRespV2

Name	Type	Description	Required
message	string		No

models.ConnectionValidation

Name	Type	Description	Required
error	string	Error is a string version of any error that may have occured	No
validated	boolean		No

models.CreateOrUpdateMysqlToKafkaJobParamV2

Name	Type	Description	Required
dest_task	models.KafkaDestTaskConfig		Yes
failover	boolean	failover default:true	No
is_password_encrypted	boolean		No
job_id	string		Yes
retry	integer		No
src_task	models.SrcTaskConfig		Yes
task_step_name	string		No

models.CreateOrUpdateMysqlToKafkaJobRespV2

Name	Type	Description	Required
dest_task	models.KafkaDestTaskConfig		Yes
eval_create_index	integer		No
failover	boolean	failover default:true	No
is_password_encrypted	boolean		No
job_id	string		Yes
job_modify_index	integer		No
message	string		No
retry	integer		No
src_task	models.SrcTaskConfig		Yes
task_step_name	string		No

models.CreateOrUpdateMysqlToMysqlJobParamV2

Name	Type	Description	Required
dest_task	models.DestTaskConfig		Yes
failover	boolean		No
is_password_encrypted	boolean		No
job_id	string		Yes
retry	integer		No
reverse	boolean		No
src_task	models.SrcTaskConfig		Yes
task_step_name	string		No

models.CreateOrUpdateMysqlToMysqlJobRespV2

Name	Type	Description	Required
dest_task	models.DestTaskConfig		Yes
eval_create_index	integer		No
failover	boolean		No
is_password_encrypted	boolean		No
job_id	string		Yes
job_modify_index	integer		No
message	string		No
retry	integer		No
reverse	boolean		No
src_task	models.SrcTaskConfig		Yes
task_step_name	string		No

models.CreateRoleReqV2

Name	Type	Description	Required
authority	string		No
name	string		No
operation_object_type	string		No
operation_users	[string]		No
tenant	string		No

models.CreateRoleRespV2

Name	Type	Description	Required
message	string		No

models.CreateUserReqV2

Name	Type	Description	Required
pass_word	string		Yes
remark	string		No
role	string		Yes
tenant	string		Yes
username	string		Yes

models.CreateUserRespV2

Name	Type	Description	Required
message	string		No

models.CurrentCoordinates

Name	Type	Description	Required
file	string		No
gtid_set	string		No
position	integer		No
read_master_log_pos	integer		No
relay_master_log_file	string		No
retrieved_gtid_set	string		No

models.CurrentUserResp

Name	Type	Description	Required
current_user	common.User		No
message	string		No

models.DataSourceConfig

Name	Type	Description	Required
table_schema	string		No
table_schema_regex	string		No
table_schema_rename	string		No
tables	[models.TableConfig]		No

models.DatabaseConnectionConfig

Name	Type	Description	Required
database_type	string		Yes
host	string		Yes
password	string		Yes
port	integer		Yes
service_name	string		No
user	string		Yes

models.DelayCount

Name	Type	Description	Required
num	integer		No
time	integer		No

models.DeleteJobRespV2

Name	Type	Description	Required
message	string		No

models.DeleteRoleRespV2

Name	Type	Description	Required
message	string		No

models.DeleteUserRespV2

Name	Type	Description	Required
message	string		No

models.DestTaskConfig

Name	Type	Description	Required
connection_config	models.DatabaseConnectionConfig		Yes
database_type	string		No
mysql_dest_task_config	models.MysqlDestTaskConfig		No
node_id	string		No
task_name	string		Yes

models.DstConfig

Name	Type	Description	Required
mysql_dest_task_config	models.MysqlDestTaskConfig		No

models.DtleNodeInfo

Name	Type	Description	Required
data_source	string		No
node_addr	string		No
node_id	string		No
source	string		No

models.GetTaskProgressRespV2

Name	Type	Description	Required
message	string		No
tasks_status	models.TaskProgress		No

models.GetUserLoginResV2

Name	Type	Description	Required
data	models.UserLoginResV2		No
message	string		No

models.GtidModeValidation

Name	Type	Description	Required
error	string	Error is a string version of any error that may have occured	No
validated	boolean		No

models.JobBaseInfo

Name	Type	Description	Required
delay	integer		No
job_create_time	string		No
job_id	string		No
job_status	string		No
job_steps	[common.JobStep]		No
subscription_topic	string		No

models.JobGtidResp

Name	Type	Description	Required
gtid	string		No
message	string		No

models.JobListRespV2

Name	Type	Description	Required
jobs	[common.JobListItemV2]		No
message	string		No

models.KafkaDestTaskConfig

Name	Type	Description	Required
kafka_broker_addrs	[string]		Yes
kafka_topic	string		Yes
message_group_max_size	integer		No
message_group_timeout	integer		No
node_id	string		No
task_name	string		Yes

models.ListActionRespV2

Name	Type	Description	Required
authority	[models.MenuItem]		No
message	string		No

models.ListColumnsRespV2

Name	Type	Description	Required
columns	[string]		No
message	string		No

models.ListSchemasRespV2

Name	Type	Description	Required
message	string		No
schemas	[models.SchemaItem]		No

models.LoginWithoutVerifyCodeReqV2

Name	Type	Description	Required
password	string		Yes
tenant	string		Yes
username	string		Yes

models.MenuItem

Name	Type	Description	Required
admin_only	boolean		No
id	integer		No
menu_level	integer		No
menu_url	string		No
name	string		No
operations	[models.ButtonItem]		No
parent_id	integer		No
text_cn	string		No
text_en	string		No

models.MysqlDestTaskConfig

Name	Type	Description	Required
dependency_history_size	integer		No
parallel_workers	integer		No
use_my_sql_dependency	boolean		No

models.MysqlSrcTaskConfig

Name	Type	Description	Required
auto_gtid	boolean		No
binlog_relay	boolean		No
expand_syntax_support	boolean		No
gtid	string		No
wait_on_job	string		No

models.MysqlTaskValidationReport

Name	Type	Description	Required
binlog_validation	models.BinlogValidation		No
connection_validation	models.ConnectionValidation		No
gtid_mode_validation	models.GtidModeValidation		No
privileges_validation	models.PrivilegesValidation		No
server_id_validation	models.ServerIDValidation		No
task_name	string		No

models.MysqlToKafkaJobDetailRespV2

Name	Type	Description	Required
basic_task_profile	models.BasicTaskProfile		No
message	string		No
task_logs	[models.TaskLog]		No

models.MysqlToMysqlJobDetailRespV2

Name	Type	Description	Required
basic_task_profile	models.BasicTaskProfile		No
message	string		No
task_logs	[models.TaskLog]		No

models.NatsMessageStatistics

Name	Type	Description	Required
in_bytes	integer		No
in_messages	integer		No
out_bytes	integer		No
out_messages	integer		No
reconnects	integer		No

models.NodeListItemV2

Name	Type	Description	Required
datacenter	string		No
dtle_version	string		No
leader	boolean		No
member	boolean		No
node_address	string		No
node_id	string		No
node_name	string		No
node_status	string		No
node_status_description	string		No
nomad_version	string		No

models.NodeListRespV2

Name	Type	Description	Required
message	string		No
nodes	[models.NodeListItemV2]		No

models.OracleSrcTaskConfig

Name	Type	Description	Required
scn	integer		No

models.PauseJobRespV2

Name	Type	Description	Required
message	string		No

models.PrivilegesValidation

Name	Type	Description	Required
error	string	Error is a string version of any error that may have occured	No
validated	boolean		No

models.ResetPasswordReqV2

Name	Type	Description	Required
current_user_password	string		Yes
password	string		Yes
tenant	string		Yes
username	string		Yes

models.ResetPasswordRespV2

Name	Type	Description	Required
message	string		No

models.ResumeJobRespV2

Name	Type	Description	Required
message	string		No

models.ReverseConfig

Name	Type	Description	Required
dest_user	string		No
dst_pwd	string		No
is_mysql_password_encrypted	boolean		No
src_pwd	string		No
src_user	string		No

models.ReverseJobReq

Name	Type	Description	Required
job_id	string		Yes
reverse_config	models.ReverseConfig		No

models.ReverseJobResp

Name	Type	Description	Required
message	string		No

models.ReverseStartRespV2

Name	Type	Description	Required
message	string		No

models.RoleListResp

Name	Type	Description	Required
message	string		No
role_list	[common.Role]		No

models.SchemaItem

Name	Type	Description	Required
schema_name	string		No
tables	[models.TableItem]		No

models.ServerIDValidation

Name	Type	Description	Required
error	string	Error is a string version of any error that may have occured	No
validated	boolean		No

models.SrcConfig

Name	Type	Description	Required
chunk_size	integer		No
drop_table_if_exists	boolean		No
group_max_size	integer		No
group_timeout	integer		No
mysql_src_task_config	models.MysqlSrcTaskConfig		No
oracle_src_task_config	models.OracleSrcTaskConfig		No
repl_chan_buffer_size	integer		No
skip_create_db_table	boolean		No

models.SrcTaskConfig

Name	Type	Description	Required
chunk_size	integer		No
connection_config	models.DatabaseConnectionConfig		Yes
drop_table_if_exists	boolean		No
group_max_size	integer		No
group_timeout	integer		No
mysql_src_task_config	models.MysqlSrcTaskConfig		No
node_id	string		No
oracle_src_task_config	models.OracleSrcTaskConfig		No
repl_chan_buffer_size	integer		No
replicate_do_db	[models.DataSourceConfig]		No
replicate_ignore_db	[models.DataSourceConfig]		No
skip_create_db_table	boolean		No
task_name	string		Yes

models.TableConfig

Name	Type	Description	Required
column_map_from	[string]		No
table_name	string		No
table_regex	string		No
table_rename	string		No
where	string		No

models.TableItem

Name	Type	Description	Required
table_name	string		No

models.TaskEvent

Name	Type	Description	Required
event_type	string		No
message	string		No
setup_error	string		No
time	string		No

models.TaskLog

Name	Type	Description	Required
address	string		No
allocation_id	string		No
node_id	string		No
target	string		No
task_events	[models.TaskEvent]		No

models.TaskProgress

Name	Type	Description	Required
ETA	string		No
backlog	string		No
buffer_status	models.BufferStat		No
current_coordinates	models.CurrentCoordinates		No
delay_count	models.DelayCount		No
exec_master_row_count	integer		No
exec_master_tx_count	integer		No
nats_message_status	models.NatsMessageStatistics		No
progress_PCT	string		No
read_master_row_count	integer		No
read_master_tx_count	integer		No
stage	string		No
throughput_status	models.ThroughputStat		No
timestamp	integer		No

models.TenantListResp

Name	Type	Description	Required
message	string		No
tenant_list	[string]		No

models.ThroughputStat

Name	Type	Description	Required
num	integer		No
time	integer		No

models.UpdataLogLevelRespV2

Name	Type	Description	Required
dtle_log_level	string		No
message	string		No

models.UpdateRoleReqV2

Name	Type	Description	Required
authority	string		No
name	string		No
operation_object_type	string		No
operation_users	[string]		No
tenant	string		No

models.UpdateRoleRespV2

Name	Type	Description	Required
message	string		No

models.UpdateUserReqV2

Name	Type	Description	Required
remark	string		No
role	string		Yes
tenant	string		Yes
username	string		Yes

models.UpdateUserRespV2

Name	Type	Description	Required
message	string		No

models.UserListResp

Name	Type	Description	Required
message	string		No
user_list	[common.User]		No

models.UserLoginReqV2

Name	Type	Description	Required
captcha	string		Yes
captcha_id	string		Yes
password	string		Yes
tenant	string		Yes
username	string		Yes

models.UserLoginResV2

Name	Type	Description	Required
token	string		No

models.ValidateJobReqV2

Name	Type	Description	Required
dest_task	models.DestTaskConfig		Yes
is_password_encrypted	boolean		No
job_id	string		Yes
src_task	models.SrcTaskConfig		Yes

models.ValidateJobRespV2

Name	Type	Description	Required
driver_config_validated	boolean	DriverConfigValidated indicates whether the agent validated the driver	No
job_validation_error	string		No
job_validation_warning	string		No
message	string		No
mysql_task_validation_report	[models.MysqlTaskValidationReport]	config	No

MySQL 用户权限说明

dtle配置的MySQL用户, 在使用不同功能时, 需具有以下权限

源端用户

权限	功能说明
select	全量复制时, 对目标表需要 `select` 权限
replication client	全量/增量复制时, 需执行 `show master status` 获取binlog信息
replication slave	增量复制时, 需要模拟 MySQL 复制

目标端用户

权限	功能说明
alter	复制时处理DDL语句
create	复制时处理DDL语句; 自动创建表结构功能; 自动创建目标端的GTID元数据表
drop	复制时处理DDL语句
index	复制时处理DDL语句
references	复制时处理DDL语句
insert	复制时处理DML语句; 修改目标端的GTID元数据表
delete	复制时处理DML语句; 修改目标端的GTID元数据表
update	复制时处理DML语句
select	查询目标端的GTID元数据表
trigger	进行目标端触发器检查

如果job中设置 `SetGtidNext=true` , 则需要 replication_applier (MySQL 8.0) 或 super 权限。

从**dtle 2.x**升级到**dtle 3.x**

dtle 2.x 和 3.x 的数据文件不兼容，直接升级无法保留进行中的job。（若无需保留的job，则可直接升级。）

升级步骤

1. 确保dtle 2.x运行中。
2. 使用导出脚本(点此下载)将现有job导出
   - 如 `./dtle-job-2to3.py 127.0.0.1:8190`
   - 将在当前目录得到一系列job json文件。需手动填写文件里的密码。
   - 导出脚本主要意义在于保存复制进度。
3. 卸载dtle 2.x并删除数据目录。
4. 安装dtle 3.x。配置/etc/dtle/nomad.hcl, 开启兼容层(设定 `api_addr` 、 `nomad_addr`)
5. 运行 dtle 3.x。将导出的job配置提交到 dtle 兼容层端口
   - 如 `curl -XPOST -d @job1.json 127.0.0.1:8190`
   - 不要提交到nomad原生端口

dtle 3.x 和 **2.x**的显著差异

- 作为nomad插件运行
- 需要另外启动consul
- job.json格式差异
- 默认端口不同
- 可使用hcl格式job配置文件
- 查询任务进度
- "暂停/恢复job"被"删除/添加job"代替
  - 恢复需要根据之前的job.json(或hcl)添加job
  - 会自动从consul中储存的Gtid继续复制
- 如果要重建同名job（并放弃进度），除了在nomad上删除，还需要在consul上删除

`allocation/<alloc_id>/stats` 接口变更

由于nomad没有提供合适的API #5863 ，我们暂且借用nomad alloc signal接口返回的错误信息来传递stats。

```
$ nomad alloc signal -s stats b0a227c1 # 或使用curl访问HTTP API
$ curl -XPOST -d '{"Signal": "stats" }' 127.0.0.1:4646/v1/client/allocation/b0a227c1-b910-0eb1-2bb9-b8bfe7607adc/signal

Error signalling allocation: Unexpected response code: 500 (1 error occurred:
    * Failed to signal task: Dest, err: rpc error: code = Unknown desc = {
    "CurrentCoordinates":{"File":"bin.000075","Position":18716,
    "GtidSet":"acd7d195-06cd-11e9-928f-02000aba3e28:1-143962",
    "RelayMasterLogFile":"","ReadMasterLogPos":0,"RetrievedGtidSet":""},
    "TableStats":null,"DelayCount":null,"ProgressPct":"0.0","ExecMasterRowCount":0,
    "ExecMasterTxCount":0,"ReadMasterRowCount":0,"ReadMasterTxCount":0,"ETA":"N/A",
    "Backlog":"","ThroughputStat":null,"MsgStat":{"InMsgs":2,"OutMsgs":2,"InBytes":299,
    "OutBytes":0,"Reconnects":0},"BufferStat":{"ExtractorTxQueueSize":0,
    "ApplierTxQueueSize":0,"ApplierGroupTxQueueSize":0,"SendByTimeout":0,
    "SendBySizeFull":0},"Stage":"Waiting for slave workers to process their queues",
    "Timestamp":1599130915717858000}
```

问题诊断 **FAQ**

通用问题

1. dtle.gtid_executed 表中是乱码

该表用uuid以binary储存以提升性能。注意查询方式gtid_executed表

协助诊断

遇到问题，首先确认使用了最新稳定版dtle。

将以下内容提供给爱可生工程师，我们将帮助您诊断故障。

通用

- job配置
- 复制阶段(全量/增量)
- 日志（请用gzip压缩）
- 堆栈/内存/运行状态/pprof信息：执行 `kill -TTIN {dtle_pid}` ，dtle会自动生成信息文件，存放在 `/tmp/dtle_dump_[date-time]` 目录下

服务无法启动,无日志输出，使用如下命令查看**std**日志

- `journalctl _SYSTEMD_UNIT=dtle-consul.service`
- `journalctl _SYSTEMD_UNIT=dtle-nomad.service`

复制停顿、不开始

- 任务有无报错
- 修改日志级别为Debug

性能低、延迟大

- 确认日志级别为Info。Debug日志会大幅降低性能。
- 网络(带宽/延迟)
- 监控项: 队列
- 数据产生量
- 部署结构(节点、dtle/mysql所在)

数据不一致

- 不一致的具体表现、特征
- consul中保存的dtle进度(gtid)
- 目标端 dtle.gtid_executed 表的内容 方法参考
- 源端 show master status 结果
- 表结构、是否有无PK表
- 复制过程中是否有DDL
- 解析源端binlog, 查找不一致数据出现的位置
- 如为双向复制，需确保业务上无数据冲突

binlog purged

即类似如下报错

> ERROR 1236 (HY000): The slave is connecting using CHANGE MASTER TO MASTER_AUTO_POSITION = 1, but the master has purged binary logs containing GTIDs that the slave requires.

- 目标端 dtle.gtid_executed 表的内容 方法参考
- consul中储存的job gtid
- MySQL `show master status;` 、 `show binary logs;` 和 `select @@gtid_purged;` 的结果

时间/资源估算

ETA (预计完成时间) 估算

源端

- 全量过程, 公式为:

```
总时间 = 已用时间 / 发送到目标端的行数 * 总行数
    其中，总行数 = (select count(*) ...)
预计完成时间 = 总时间 - 已用时间
即：预计完成时间 = 剩余行数 / 当前发送速率
```

- 增量过程, ETA 一直为 `0s`

目标端

- 全量过程. 公式为:

```
总时间 = 已用时间 / 已写入目标端的行数 * 总行数
预计完成时间 = 总时间 - 已用时间
即：预计完成时间 = 剩余行数 / 当前写入速率
```

- 增量过程, ETA 一直为 `0s`

内存占用估算

```
内存占用估算 = RowSize * ChunkSize * QueueSize * 内存占用系数
```

其中:

- RowSize为数据行的平均大小 (字节)
- ChunkSize为配置项
- QueueSize为传输队列长度, 硬编码为24
- 内存占用系数 测量约为 常量3.2

关于大事务

大事务指传输、处理数据量较大的事务，一般由DML组成。DDL事务（QueryEvent）不会太大，尽管某些DDL需要较长的执行时间。

对于一个多行的大事务，dtle会按行分开处理、传输并执行（但在目标端仍作为一个事务提交）。

当一个job处理大事务时，需要等待该段数据在目标端执行完毕才会获取下一批数据。

当同时处理大事务的job数量达到 `big_tx_max_jobs` 时，所有job都会进入等待模式。

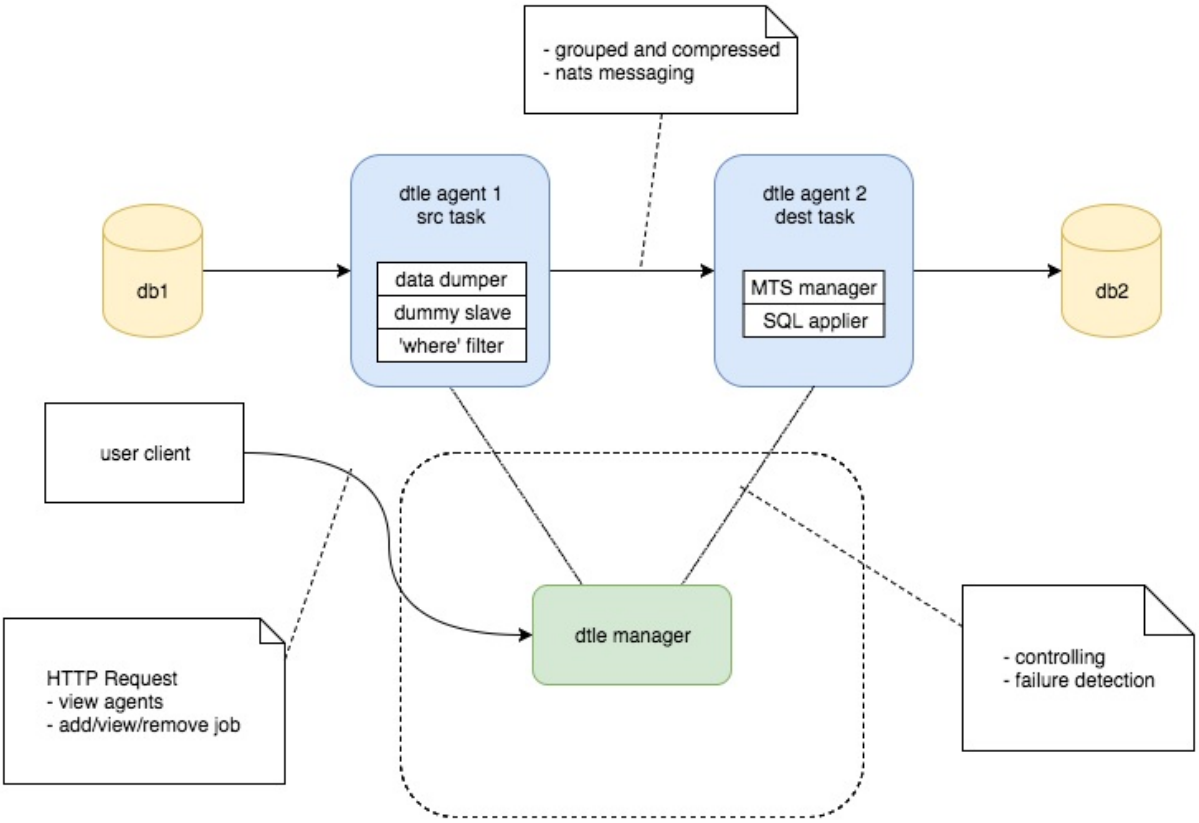
为了避免MySQL源端超时，等待时间的上限是 `@@net_write_timeout / 2`

dtle 架构

nomad角色分为 server、client.

- manager数量应为1、3或5个
- agent数量不限
- 至少需要1个manager和1个agent
- 一个nomad进程可同时扮演 server 和 client

任务分为源端任务和目标端任务, 各由agent执行. 通过网络压缩传输数据.



Kafka 消息格式

dtle Kafka 输出, 消息格式兼容 Debezium

其消息格式具体可参考 https://debezium.io/documentation/reference/1.8/tutorial.html

此处概要说明

- 每行数据变更会有一个消息
- 每个消息分为key和value
  - key是该次变更的主键
  - value是该次变更的整行数据
- key和value各自又有schema和payload
  - payload是具体的数据
  - schema指明了数据的格式, 即payload的解读方式, 可以理解为“类定义”
    - 注意和SQL schema含义不同
    - 表结构会包含在 Kafka Connect schema 中

DML

Key

以下是一个消息的key. 只是简单的包含了主键.

```
{
  "schema": {
    "type": "struct",
    "name": "dbserver1.inventory.customers.Key"
    "optional": false,
    "fields": [
      {
        "field": "id",
        "type": "int32",
        "optional": false
      }
    ]
  },
  "payload": {
    "id": 1004
  }
}
```

Value

以下是一个消息的value, 其类型为 `topic.schema.table.Envelope` , 拥有5个字段

- `before` , 复杂类型 `topic.schema.table.Value` , 为该表的表结构.
- `after` , 复杂类型, 同上
- `source` , 复杂类型, 为该次变更的元数据
- `op : string` . 用"c", "d", "u" 分别表达操作类型: 增、删、改
- `ts_ms : int64` . dtle 处理该行变更的时间.

```
{
  "schema": {
    "type": "struct",
    "fields": [
      {
        "type": "struct",
        "fields": [
```

```
        {
          "type": "int32",
          "optional": false,
          "field": "id"
        },
        {
          "type": "string",
          "optional": false,
          "field": "first_name"
        },
        {
          "type": "string",
          "optional": false,
          "field": "last_name"
        },
        {
          "type": "string",
          "optional": false,
          "field": "email"
        }
      ],
      "optional": true,
      "name": "dbserver1.inventory.customers.Value",
      "field": "before"
    },
    {
      "type": "struct",
      "fields": [
        {
          "type": "int32",
          "optional": false,
          "field": "id"
        },
        {
          "type": "string",
          "optional": false,
          "field": "first_name"
        },
        {
          "type": "string",
          "optional": false,
          "field": "last_name"
        },
        {
          "type": "string",
          "optional": false,
          "field": "email"
        }
      ],
      "optional": true,
      "name": "dbserver1.inventory.customers.Value",
      "field": "after"
    },
    {
      "type": "struct",
      "fields": [
        {
          "type": "string",
          "optional": true,
          "field": "version"
        },
        {
          "type": "string",
          "optional": false,
          "field": "name"
        },
        {
          "type": "int64",
          "optional": false,
          "field": "server_id"
        },
        {
```

```
            "type": "int64",
            "optional": false,
            "field": "ts_sec"
          },
          {
            "type": "string",
            "optional": true,
            "field": "gtid"
          },
          {
            "type": "string",
            "optional": false,
            "field": "file"
          },
          {
            "type": "int64",
            "optional": false,
            "field": "pos"
          },
          {
            "type": "int32",
            "optional": false,
            "field": "row"
          },
          {
            "type": "boolean",
            "optional": true,
            "field": "snapshot"
          },
          {
            "type": "int64",
            "optional": true,
            "field": "thread"
          },
          {
            "type": "string",
            "optional": true,
            "field": "db"
          },
          {
            "type": "string",
            "optional": true,
            "field": "table"
          }
        ],
        "optional": false,
        "name": "io.debezium.connector.mysql.Source",
        "field": "source"
      },
      {
        "type": "string",
        "optional": false,
        "field": "op"
      },
      {
        "type": "int64",
        "optional": true,
        "field": "ts_ms"
      }
    ],
    "optional": false,
    "name": "dbserver1.inventory.customers.Envelope",
    "version": 1
  },
  "payload": {
    "before": null,
    "after": {
      "id": 1004,
      "first_name": "Anne",
      "last_name": "Kretchmar",
      "email": "annek@noanswer.org"
    },
```

```
    "source": {
      "version": "0.8.3.Final",
      "name": "dbserver1",
      "server_id": 0,
      "ts_sec": 0,
      "gtid": null,
      "file": "mysql-bin.000003",
      "pos": 154,
      "row": 0,
      "snapshot": true,
      "thread": null,
      "db": "inventory",
      "table": "customers"
    },
    "op": "c",
    "ts_ms": 1486500577691
  }
}
```

DDL (SchemaChangeTopic)

dtle会将DDL写入SchemaChangeTopic。该topic值可配置.

Schema change消息中，key永远为 `null` , 仅 value部分有值:

```
{
  "source" : {
    "server" : "mysql2"
  },
  "position" : {
    "ts_sec" : 1641807976,
    "file" : "bin.000022",
    "pos" : 439,
    "gtids" : "acd7d195-06cd-11e9-928f-02000aba3e28:1-175",
    "snapshot" : true
  },
  "databaseName" : "a",
  "ddl" : "CREATE TABLE `a` (\n  `id` int(11) NOT NULL AUTO_INCREMENT,\n  PRIMARY KEY (`id`)\n) ENGINE=InnoDB AUTO_INCREMENT=4
 DEFAULT CHARSET=latin1",
  "tableChanges" : [ {
    "type" : "CREATE",
    "id" : "\"a\".\"a\"",
    "table" : {
      "defaultCharsetName" : "latin1",
      "primaryKeyColumnNames" : [ "id" ],
      "columns" : [ {
        "name" : "id",
        "jdbcType" : 4,
        "typeName" : "INT",
        "typeExpression" : "INT",
        "charsetName" : null,
        "length" : 11,
        "position" : 1,
        "optional" : false,
        "autoIncremented" : true,
        "generated" : true
      } ]
    }
  } ]
}
```

其中:

- `position.snapshot==true` 表明这是全量初始化时的表结构（通过 `show create table` 等生成）。
- `position.snapshot==false` 则表明：这是增量过程中执行的DDL。

注: `tableChanges` 结构在dtle中尚未实现。

MySQL数据类型到 “Kafka Connect schema types”的转换

见 https://debezium.io/docs/connectors/mysql/#data-types

Oracle MySQL 字段映射

字段类型

Oracle	MySQL	全量支持	增量支持	限制	后期是否考虑优化/支持
BINARY_FLOAT	float	否	否(insert支持)	mysql 不支持Inf/-Inf/Nan数据,MySQL float类型无法精确匹配)	是
BINARY_DOUBLE	float	是	是	mysql 不支持Inf/-Inf/Nan数据	
CHAR(n), CHARACTER(n)	CHAR(n), CHARACTER(n)	是	是		
DATE	datetime	是	是	MySQL 最大长度限制为6，Oracle为9	
DECIMAL(p,s), DEC(p,s)	DECIMAL(p,s), DEC(p,s)	是	是		
DOUBLE PRECISION	DOUBLE PRECISION	否	是		
FLOAT(p)	DOUBLE	是	是		
INTEGER, INT	INT	是	是	极值问题	
INTERVAL YEAR(p) TO MONTH	VARCHAR(30)	是	是	部分结果异常	是
INTERVAL DAY(p) TO SECOND(s)	VARCHAR(30)	是	是	同步结果以纳秒保存	是
NCHAR(n)	NCHAR(n)/NVARCHAR(n)	是	是		
NCHAR VARYING(n)	NCHAR VARYING(n)	是	是		
NUMBER(p,0), NUMBER(p)	TINYINT/SMALLINT/INT/BIGINT/DECIMAL(p)	是	是		
NUMBER(p,s)	DECIMAL(p,s)	是	是		
NUMBER, NUMBER(*)	DOUBLE	是	是		
NUMERIC(p,s)	NUMERIC(p,s)	是	是		
NVARCHAR2(n)	NVARCHAR(n)	是	是		
RAW(n)	VARBINARY(n)	是	是		

REAL	DOUBLE	是	是		
ROWID	CHAR(100)	是	是		
SMALLINT	DECIMAL(38)	是	是		
TIMESTAMP(p)	datetime	是	是		
VARCHAR(n)	VARCHAR(n)	是	是		
VARCHAR2(n)	VARCHAR(n)	是	是		
BLOB	BLOB	否	否	当前解析逻辑无法获取完整数据	
CLOB	CLOB	否	否	当前解析逻辑无法获取完整数据	
LONG	LONGTEXT	否	否(insert支持)	不支持minus查询	是
LONG RAW	LONGBLOB	否	否(insert支持)		是
NCLOB	NCLOB	否	否	当前解析逻辑无法获取完整数据	是
TIMESTAMP(p) WITH TIME ZONE	datetime	否	否	时区未解析	是
BFILE		否	否	logminer不支持	
UROWID(n)		否	否	logminer解析异常	否
XMLTYPE		否	否	logminer不支持	否

待支持

Oracle	**MySQL**	是否支持	不支持原因	后期是否考虑支持

不支持

Oracle	是否支持	不支持原因
BFILE	否	logminer不支持
UROWID(n)	否	logminer解析异常
XMLTYPE	否	logminer不支持

如何参与

提交缺陷

可直接在github issues页面 新建 issue, 选择 `Bug Report` 模板, 按格式填写完成后提交即可

提交功能

可直接在github issues页面 新建 issue, 选择 `Feature request` 模板, 按格式填写完成后提交即可

提交代码

按照github的pull request流程即可

如何全职参与

本项目的维护方(上海爱可生信息技术股份有限公司)一直在招聘 靠谱的研发工程师/靠谱的测试工程师. 如果通过dtle, 您对全职参与类似的项目有兴趣, 请联系我们的研发团队.

路线图

- 生态
  - [ ] 支持MGR Primary切换
  - [ ] 对于 MySQL分布式中间件 (如dble) 提供数据扩容方案
  - [ ] 复制到Kafka的数据格式支持Avro
  - [ ] 支持更多种类的公有云间的数据迁移
- ETL-E
  - [ ] WHERE 过滤条件 支持更丰富的函数 (目前仅支持关系符和简单函数)
  - [ ] *oracle*
  - [ ] 动态增减同步对象
- ETL-T
  - [ ] 列名变换
  - [ ] WHERE 过滤条件 支持更丰富的函数 (目前仅支持关系符和简单函数)
  - [ ] 数据变换
  - [x] 库.表名变换 (2.19.11.0+)
  - [x] 列选择、列顺序变换 (3.20.08.0+)
- ETL-L
  - [ ] 支持MGR Primary切换 actiontech/dtle#541
  - [x] MTS (actiontech/dtle#688)
- 链路管理
  - [ ] 对链路提供限流参
  - [ ] 支持2G级别的大事务
  - [ ] 限流
  - [ ] 加密
  - [ ] IPv6 actiontech/dtle#600
- 运维
  - [ ] 提供告警功能
  - [ ] DTLE 容灾
- 非技术指标
  - [ ] 支持2G级别的大事务
  - [x] 一致性DDL元数据 (2.19.03.0+ actiontech/dtle#321)
  - [ ] 免一致性快照事务的全量复制
    - [ ] 全量复制也可断点续传