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: 86-13718877200, 王先生
- 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:

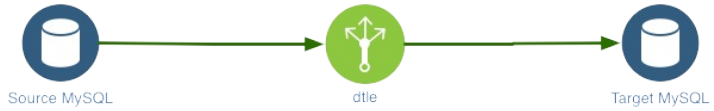
- 多种数据传输模式
  - 支持链路压缩
  - 支持同构传输和异构传输
  - 支持跨网络边际的传输
- 多种数据处理模式
  - 支持库/表/行级别 数据过滤
- 多种数据通道模式
  - 支持多对多的数据传输
  - 支持回环传输
- 多种源/目标端
  - 支持MySQL - MySQL的数据传输
  - 支持MySQL - Kafka的数据传输
- 集群模式
  - 提供可靠的元数据存储
  - 可进行自动任务分配
  - 支持自动故障转移

1.0 MySQL的单向复制/聚合/分散

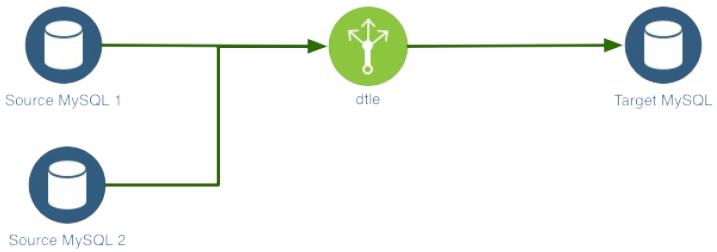
如下图, dtle 支持 MySQL 单向数据复制的常见场景如下:

- 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

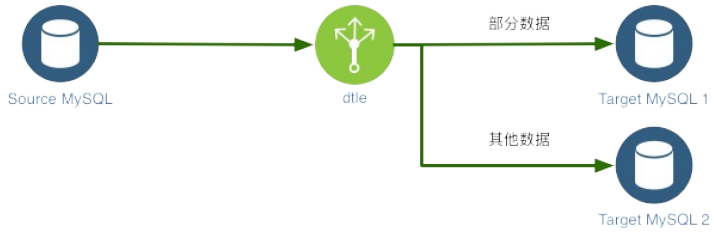
1-1复制



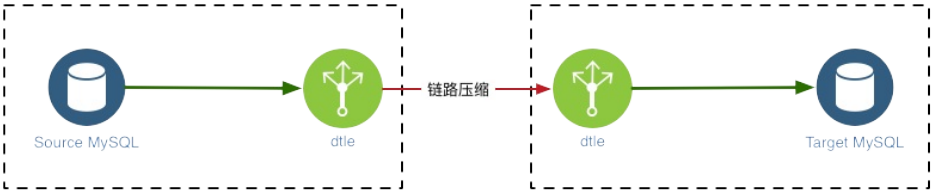
n-1汇聚



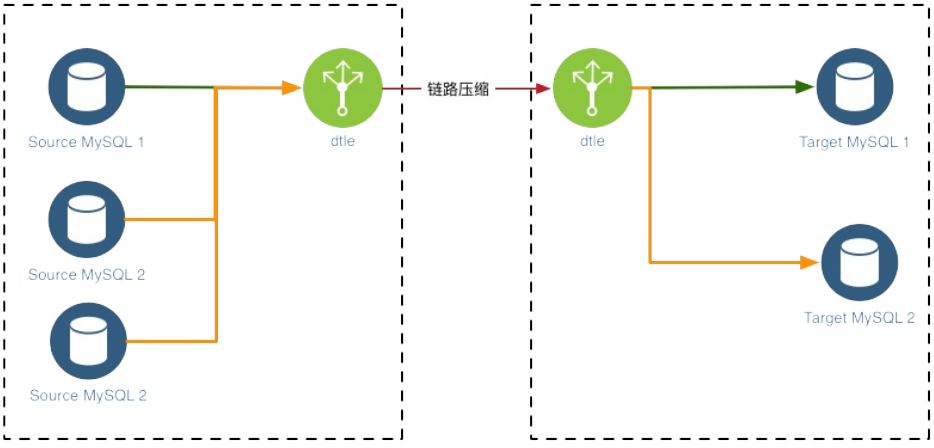
1-n拆分



跨网络边际的1-1复制

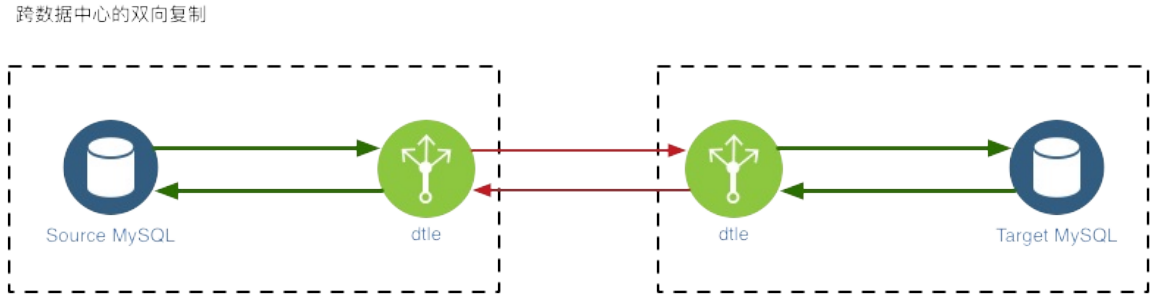


跨网络边际的多个复制通道



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)

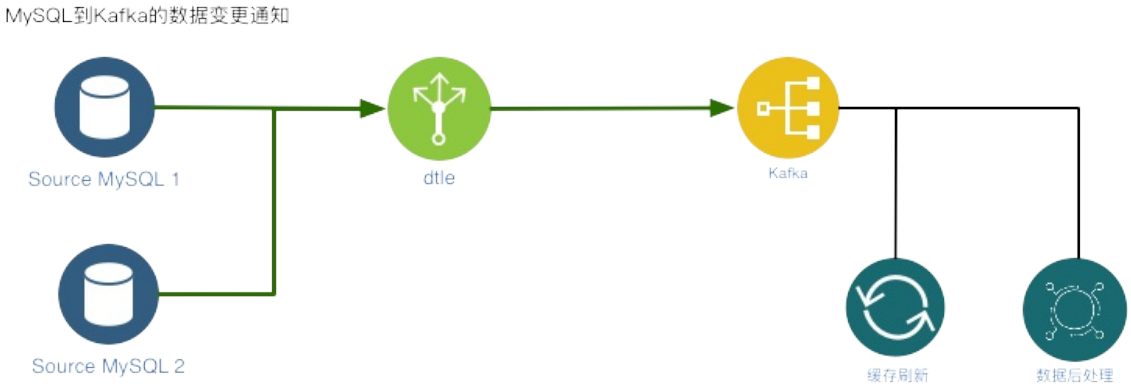
目前支持的公有云同步通道:

- 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



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"
              }]
            }],
            "ConnectionConfig": {
              "Host": "mysql-src",
              "Port": 3306,
              "User": "root",
              "Password": "pass"
            }
          }
        }]
      }, {
        "Name": "dest",
        "Tasks": [{
          "Name": "dest",
          "Driver": "dtle",
          "Config": {
            "ConnectionConfig": {
              "Host": "mysql-dst",
              "Port": 3306,
              "User": "root",
              "Password": "pass"
            }
          }
        }]
    }]
  }
}
```

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 --enforce-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 --enforce-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.d
emo_tbl(a int primary key)"
< ...

> mysql -h 127.0.0.1 -P 33061 -uroot -ppass -e "show master status\G" | grep "Executed_Gt
id_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.d
emo_tbl(a int primary key)"
< ...

> mysql -h 127.0.0.1 -P 33062 -uroot -ppass -e "show master status\G" | grep "Executed_Gt
id_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.d
emo_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"
              }]
            }],
            "ConnectionConfig": {
              "Host": "mysql-src1",
              "Port": 3306,
              "User": "root",
              "Password": "pass"
            }
          }
        }]
      }, {
        "Name": "dest",
        "Tasks": [{
          "Name": "dest",
          "Driver": "dtle",
          "Config": {
            "ConnectionConfig": {
              "Host": "mysql-dst",
              "Port": 3306,
              "User": "root",
              "Password": "pass"
            }
          }
        }]
    }]
  }
}
```

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"
              }]
            }],
            "ConnectionConfig": {
              "Host": "mysql-src2",
              "Port": 3306,
              "User": "root",
              "Password": "pass"
            }
          }
        }]
      }, {
        "Name": "dest",
        "Tasks": [{
          "Name": "dest",
          "Driver": "dtle",
          "Config": {
            "ConnectionConfig": {
              "Host": "mysql-dst",
              "Port": 3306,
              "User": "root",
              "Password": "pass"
            }
          }
        }]
    }]
  }
}
```

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 --enforce-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 --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.d
emo_tbl(a int primary key)"
< ...

> mysql -h 127.0.0.1 -P 33061 -uroot -ppass -e "show master status\G" | grep "Executed_Gt
id_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.d
emo_tbl(a int primary key)"
< ...

> mysql -h 127.0.0.1 -P 33063 -uroot -ppass -e "CREATE DATABASE demo; CREATE TABLE demo.d
emo_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"
              }]
            }],
            "ConnectionConfig": {
              "Host": "mysql-src",
              "Port": 3306,
              "User": "root",
              "Password": "pass"
            }
          }
        }]
      }, {
        "Name": "dest",
        "Tasks": [{
          "Name": "dest",
          "Driver": "dtle",
          "Config": {
            "ConnectionConfig": {
              "Host": "mysql-dst1",
              "Port": 3306,
              "User": "root",
              "Password": "pass"
            }
          }
        }]
    }]
  }
}
```

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"
              }]
            }],
            "ConnectionConfig": {
              "Host": "mysql-src",
              "Port": 3306,
              "User": "root",
              "Password": "pass"
            }
          }
        }]
      }, {
        "Name": "dest",
        "Tasks": [{
          "Name": "dest",
          "Driver": "dtle",
          "Config": {
            "ConnectionConfig": {
              "Host": "mysql-dst2",
              "Port": 3306,
              "User": "root",
              "Password": "pass"
            }
          }
        }]
    }]
  }
}
```

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

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-b322-0242
```

```
ac150002:1-5",
            "ReplicateDoDb": [{
              "TableSchema": "demo",
              "Tables": [{
                "TableName": "demo_tbl"
              }]
            }],
            "ConnectionConfig": {
              "Host":"mysql-dc2",
              "Port": 3306,
              "User": "root",
              "Password": "pass"
            }
          }
        }]
      }, {
        "Name": "dest",
        "Tasks": [{
          "Name": "dest",
          "Driver": "dtle",
          "Constraints": [{
            "LTarget": "${node.unique.name}",
            "RTarget": "nomad1",
            "Operand": "="
          }],
          "Config": {
            "ConnectionConfig": {
              "Host":"mysql-dc1",
              "Port": 3306,
              "User": "root",
              "Password": "pass"
            }
          }
        }]
    }]
  }
}
```

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, R
EFERENCES, INDEX, ALTER, CREATE TEMPORARY TABLES, LOCK TABLES, EXECUTE, REPLICATION SLAVE
, REPLICATION CLIENT, CREATE VIEW, SHOW VIEW, CREATE ROUTINE, ALTER ROUTINE, 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": []
            }],
            "ConnectionConfig": {
              "Host":"rm-xxxx.mysql.rds.aliyuncs.com",
              "Port":"3306",
              "User":"root",
              "Password":"Acti0ntech"
            }
          }
        }]
      }, {
        "Name": "dest",
        "Tasks": [{
          "Name": "dest",
          "Driver": "dtle",
          "Config": {
            "ConnectionConfig": {
              "Host":"mysql-cn-east-2-yyyy.public.jcloud.com",
              "Port":"3306",
              "User":"actiontech",
              "Password":"Acti0ntech"
            }
          }
        }]
    }]
  }
}
```

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.d
emo_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=dtl
e-net -d bitnami/zookeeper
docker run --name kafka-dst -p 9092:9092 -e KAFKA_ZOOKEEPER_CONNECT=kafka-zookeeper:2181
-e ALLOW_PLAINTEXT_LISTENER=yes --network=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/iss
ues
```

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"
              }]
            }],
            "ConnectionConfig": {
              "Host": "mysql-src",
              "Port": 3306,
              "User": "root",
              "Password": "pass"
            }
          }
        }]
      }, {
        "Name": "dest",
        "Tasks": [{
          "Name": "dest",
          "Driver": "dtle",
          "Config": {
            "KafkaConfig": {
              "Topic": "demo-topic",
              "Brokers": ["kafka-dst:9092"],
              "Converter": "json"
            }
          }
        }]
    }]
  }
}
```

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/iss
ues

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 --to
pic demo-topic.demo.demo_tbl --from-beginning
< ...
{"schema":{"type":"struct","optional":false,"fields":[{"type":"struct","optional":true,"f
ield":"before","fields":[{"type":"int32","optional":false,"field":"a"}],"name":"demo-topi
c.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":false,"field":"source","fields":[{"type":"string","optional":true,"f
ield":"version"},{"type":"string","optional":false,"field":"name"},{"type":"int64","optio
nal":false,"field":"server_id"},{"type":"int64","optional":false,"field":"ts_sec"},{"type
":"string","optional":true,"field":"gtid"},{"type":"string","optional":false,"field":"fil
e"},{"type":"int64","optional":false,"field":"pos"},{"type":"int32","optional":false,"fie
ld":"row"},{"type":"boolean","optional":true,"field":"snapshot"},{"type":"int64","optiona
l":true,"field":"thread"},{"type":"string","optional":true,"field":"db"},{"type":"string"
,"optional":true,"field":"table"}],"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},"source":{"version":"0.0.1","name":"demo-topic","server_id":0,"ts_sec":0,"gtid":
null,"file":"","pos":0,"row":1,"snapshot":true,"thread":null,"db":"demo","table":"demo_tb
l"},"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

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 。

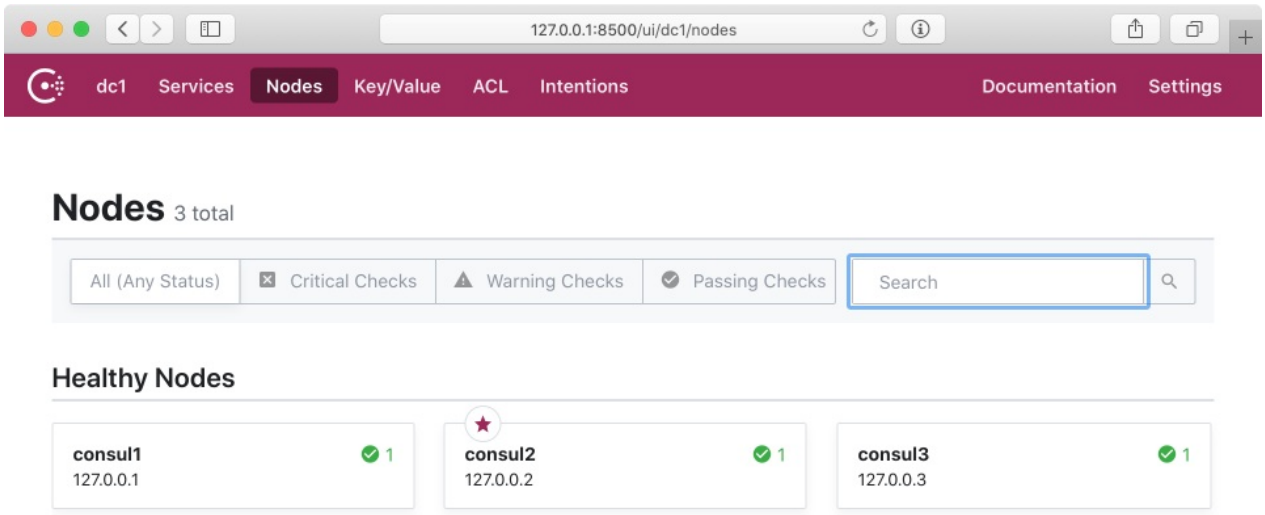
consul配置

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。

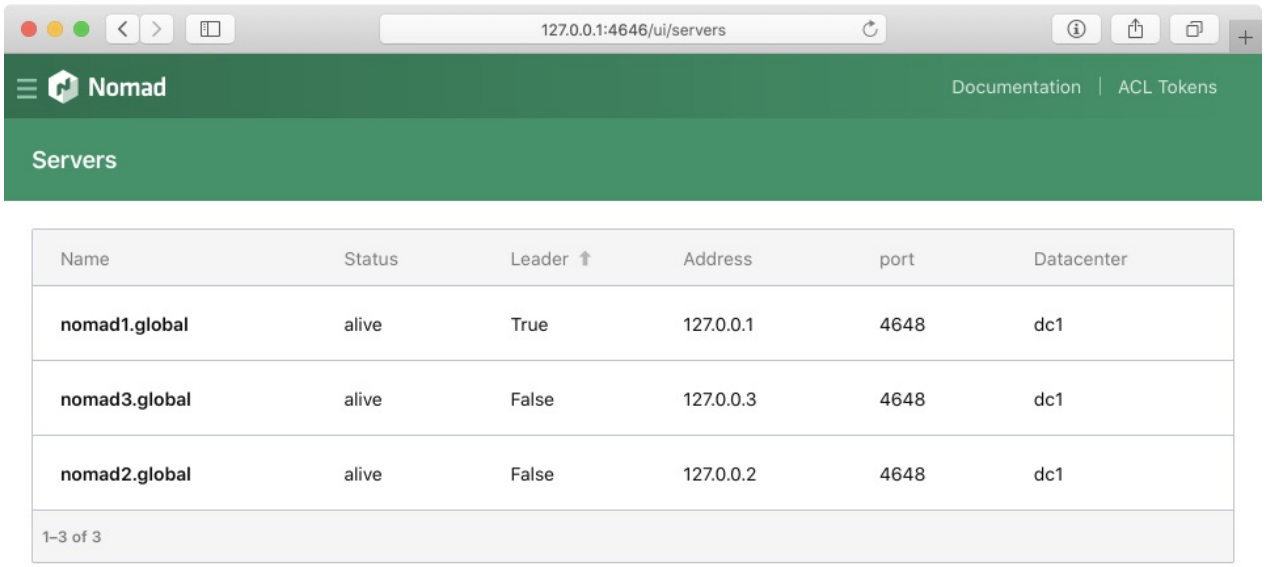


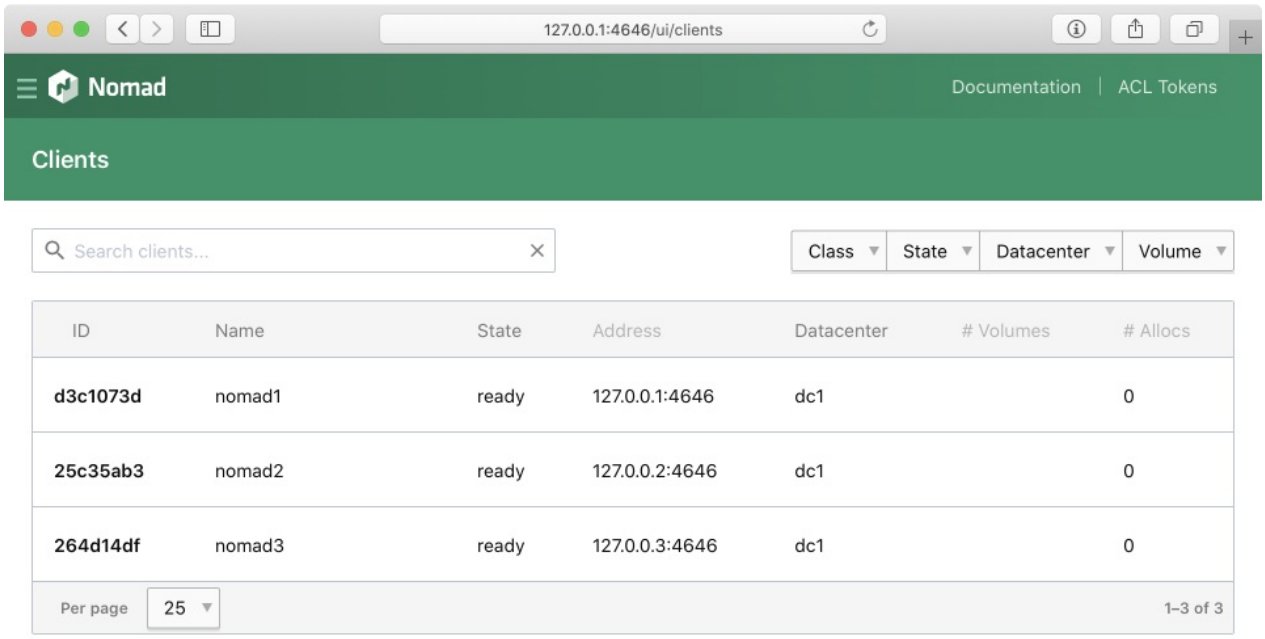
nomad配置

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
  - gbk
  - utf8
  - utf8mb4
  - binary
- 在latin1/2表中，不支持非latin字符（如中文）（#388）
- 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';

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])

其中延迟统计可见

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

Prometheus配置演示

使用Prometheus可直观查看监控项并记录历史值。

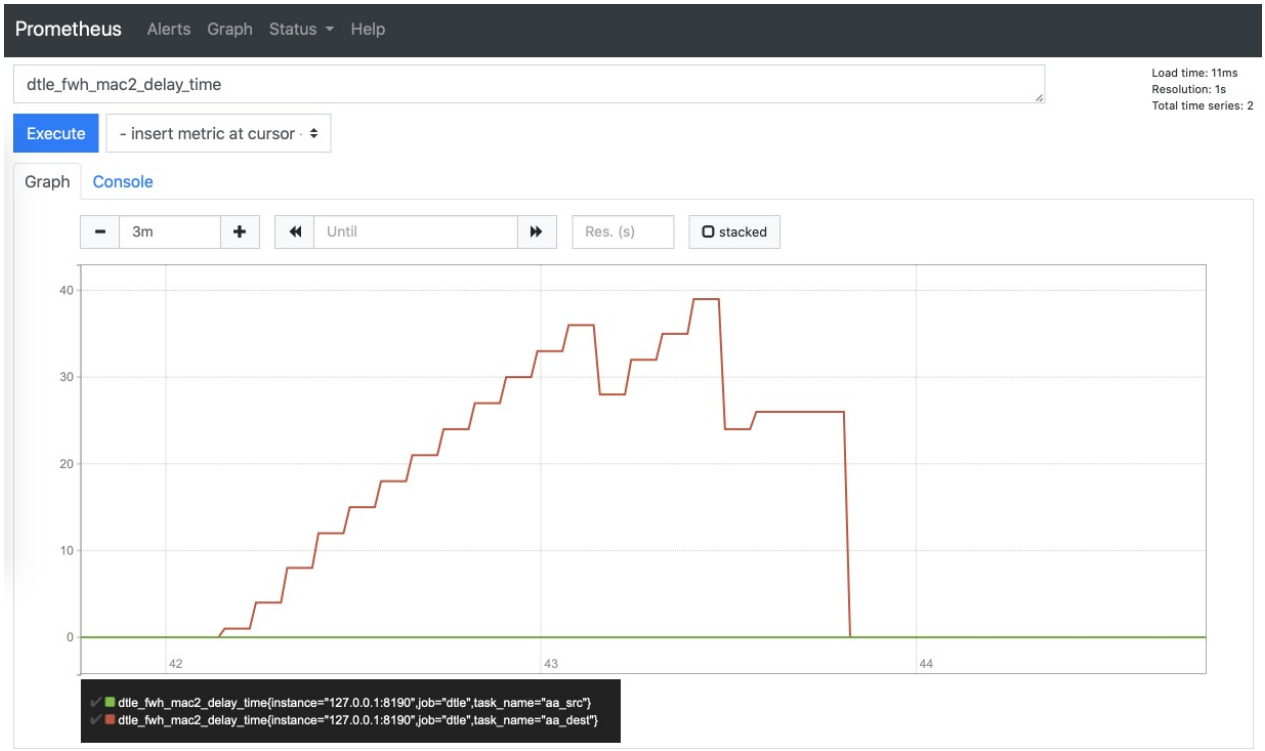
准备配置文件 `less prometheus.yml` ：

```
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。
```

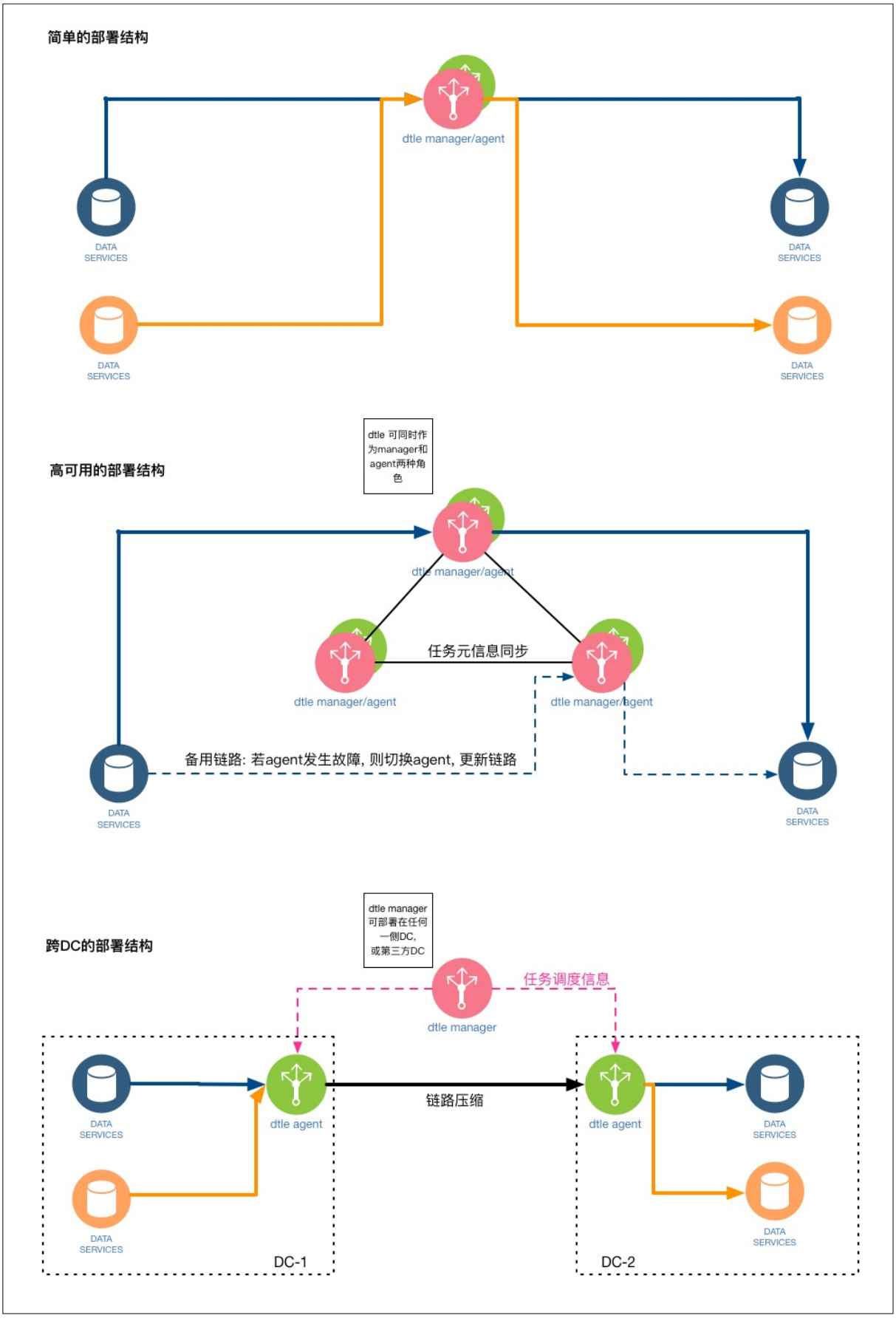
使用docker运行Prometheus:

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



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table为对象的DDL

DDL类型	**DDL**语句示例	是否支持
添加列	alter table sbtest1 add c_blob blob;	支持
删除列	alter table sbtest1 drop pad;	支持
更改表名	alter table sbtest1 rename rename test1;	支持
更改列名	alter table sbtest1 change k z int(10) DEFAULT NULL;	支持
更改列名和类型	alter table sbtest1 change k z bigint DEFAULT NULL;	支持
更改列类型	alter table sbtest1 modify k bigint DEFAULT NULL;	支持
添加index	alter table sbtest1 add index index_c(c);	支持
添加 unique index	alter table t_mod add constraint UK unique(m);	支持
删除 unique index	alter table t_mod drop index UK;	支持
添加primary key	alter table t_mod add constraint PK primary key(id);	支持
删除primary key	alter table t_mod drop primary key;	支持
添加fulltext index	alter table t_mod add fulltext index last_name_index(last_name)	支持
删除fulltext index	alter table t_mod drop index last_name_index;	支持
添加 foreign key	alter table t_add_foreign_key add constraint FK foreign key(id) references t_add_column(id_name);	支持
删除foreign key	alter table t_add_foreign_key drop foreign key FK;	支持
创建index	create index t_test on t_add_column(a);	支持
添加 partition	alter table part_tab add partition (partition p11 values less than (2012));	支持

- 其他对象：

对象	全量	增量	备注
procedure	不支持	支持create，alter，drop，但可能存在字符集不完全一致的情况#357	创建库级别迁移，"ExpandSyntaxSupport": true
function	不支持	支持create，alter，drop，但可能存在字符集不完全一致的情况#357	创建库级别迁移，"ExpandSyntaxSupport": true，mysql> `set global log_bin_trust_function_creators=1;`
view	不支持	不支持	
trigger	不支持	不支持	目前由于ddl解析问题，dtle不能正常跳过trigger ddl，源端执行的所有ddl都会被发送给目标端。为确保稳定性及数据一致性，应避免在源端执行任何trigger ddl，一般情况下也不建议在目标端执行trigger（#577）
event	（由于event的特殊性，全量中event虽然未迁移至目标端，但不影响src和dest数据一致，增量中出现的event可以迁移至dest）	支持create，alter，drop，但可能存在字符集不完全一致的情况#357	创建库级别迁移，"ExpandSyntaxSupport": true，在源端开启 mysql> `set global event_scheduler=1` ，目标端event_scheduler不应开启

job.json 配置样例：

```
{
    "Name":"test_function",
    "Tasks":[
        {
            "Type":"Src",
            "Config":{
                "Gtid":"",
                "ExpandSyntaxSupport":true,
                "ReplicateDoDb":[
                    {
                        "TableSchema":"functiontest",
                        "Tables":[]
                    }
                ],
                "ConnectionConfig":{ ... }
            }
        }, {
            "Type":"Dest",
            "Config":{
                "ConnectionConfig":{ ... }
            }
        }
    ]
}
```

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’

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：

```
{
    "Name":"test1",
    "Tasks":[
        {
            "Type":"Src",
            "Config":{
                "Gtid":"",
                "ExpandSyntaxSupport":true,
                "DropTableIfExists":false,
                "ReplicateDoDb":[],
                "ConnectionConfig":{...}
            }
        },
        {
            "Type":"Dest",
            "Config":{
                "ConnectionConfig":{...}
            }
        }
    ]
}
```

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 table

```
create table demo.colmap (id int primary key, val1 int, val2 int)
```

dst table

```
create table demo.colmap (val2 int, id int primary key)
```

使用 `ColumnMapFrom` 调整列顺序并忽略 `val1` 列。注意预先创建好表。

```
"ReplicateDoDb": [{
    "TableSchema":"demo",
    "Tables":[{
        "TableName": "colmap",
        "ColumnMapFrom": ["val1", "id"]
    }]
}],
"SkipCreateDbTable": true,
"DropTableIfExists": false,
```

暂不支持列重命名和正则匹配。

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": "",
```

若要在增量job（即指定Gtid）中使用BinlogRelay，则必须另外指定 `BinlogFile` 、 `BinlogPos` (见 `show master status`).

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。

延迟监控

开启 监控项（metrics） 后可查看延迟统计

延迟统计仅对增量（含Kafka输出）有效，其原理为：

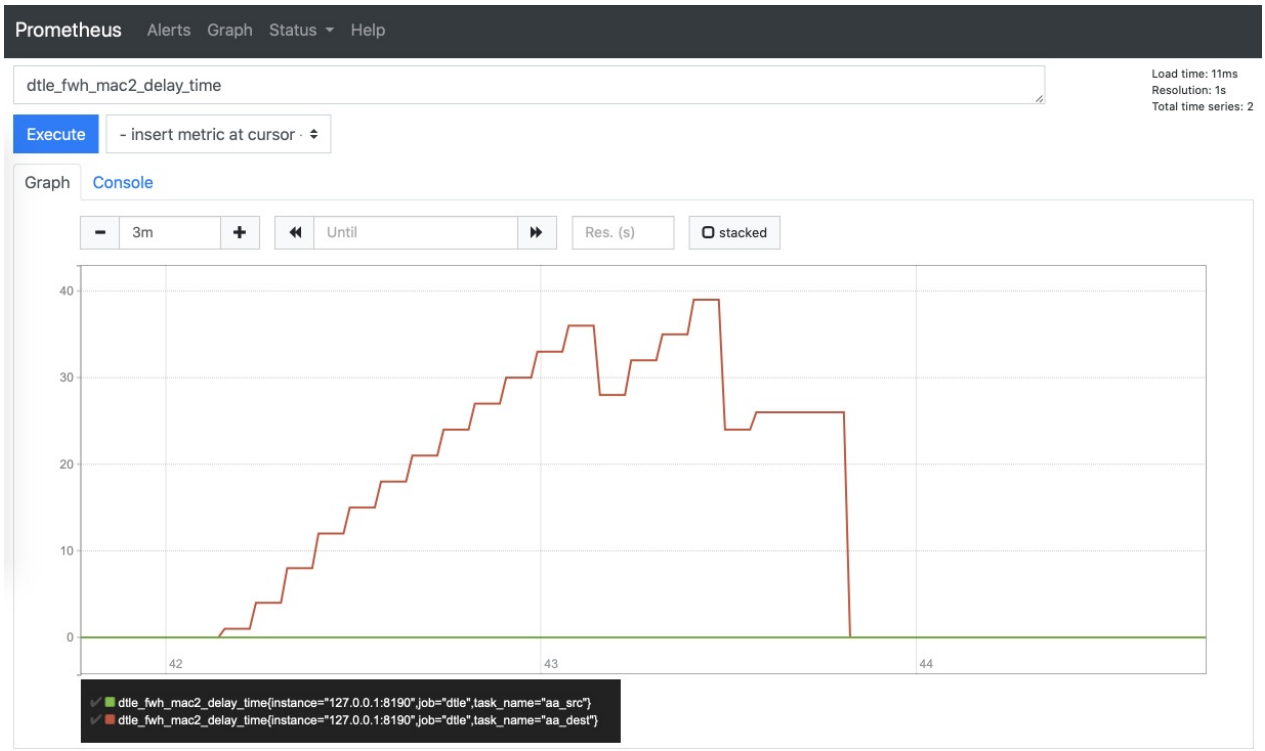
- 源端MySQL在执行事务时，binlog中记录了时间戳
- dtle在传输/回放事务时，取时间戳和当前时间的差值为延迟值
- 如果一段时间（15s）没有事务，则重置延迟值为0

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，可用Prometheus抓取dtle的监控项，方法见 监控项（metrics） 。

效果图：



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）
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）
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）

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）

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。如

```
group "src" {
  task "src" {
    driver = "dtle"
    config {
      ...
    }
  }
}
```

dtle专有配置有如下字段**:**

参数名	必填**?**	类型	源端/目标端可配	默认值	说明
Gtid	否	String	源端	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	源端		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	源端	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 (有bug，暂时禁用)	否	Bool	源端	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(指定GTID), 需要同时填写BinlogFile和BinlogPos.
BinlogFile	否	String	源端		增量任务开始的Binlog文件(即源端mysql上 `show master status` 的结果).
BinlogPos	否	Int	源端	0	增量任务开始的Binlog位置, 和BinlogFile配套使用.

ReplicateDoDb	否	Object 数组	源端	-	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	源端	-	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
ConnectionConfig	否	Object	两端	-	MySQL源/目标端信息, 见下方 ConnectionConfig 说明。和KafkaConfig二选一填写。
KafkaConfig	否	Object	目标端	-	Kafka目标端信息, 见下方 KafkaConfig 说明。和ConnectionConfig二选一填写。
DropTableIfExists	否	Bool	源端	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	源端	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	目标端	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	目标端	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	目标端	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。
ReplChanBufferSize	否	Int	源端	32	复制任务缓存的大小, 单位为事务数
ChunkSize	否	Int	源端	2000	全量复制时, 每次读取-传输-写入的行数
ExpandSyntaxSupport	否	Bool	源端	false	支持复制 用户权限/存储过程DDL/函数DDL
GroupMaxSize	否	int	源端	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	源端	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	源端	[]	是否跳过一些事件, 如 `["NoDML", "NoDMLDelete", "NoDMLInsert", "NoDMLUpdate", "NoDDL", "NoDDLAlterTableAddColumn", "NoDDLAlterTableDropColumn", "NoDDLAlterTableModifyColumn", "NoDDLAlterTableChangeColumn",`

					`"NoDDLAlterTableAlterColumn"]` (以上为所有的filter)

ReplicateDoDb 每个元素有如下字段:

参数名	必填?	类型	源端/目标端 可配	默认值	说明
TableSchema	否	String	源端	-	数据库名
TableSchemaRegex	否	String	源端	-	数据库映射正则表达式，可用于多个数据库重命名
TableSchemaRename	否	String	源端	-	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	源端	-	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	源端	-	表名
Table.Where	否	String	源端	-	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	源端	-	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	源端	-	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	源端	-	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	源端/目标端 可配	默认值	说明
TableSchema	是	String	源端	-	数据库名
Tables	否	Object数组	源端	-	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	源端	-	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
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.

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` ）。

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"
  }]
}]
```

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）。

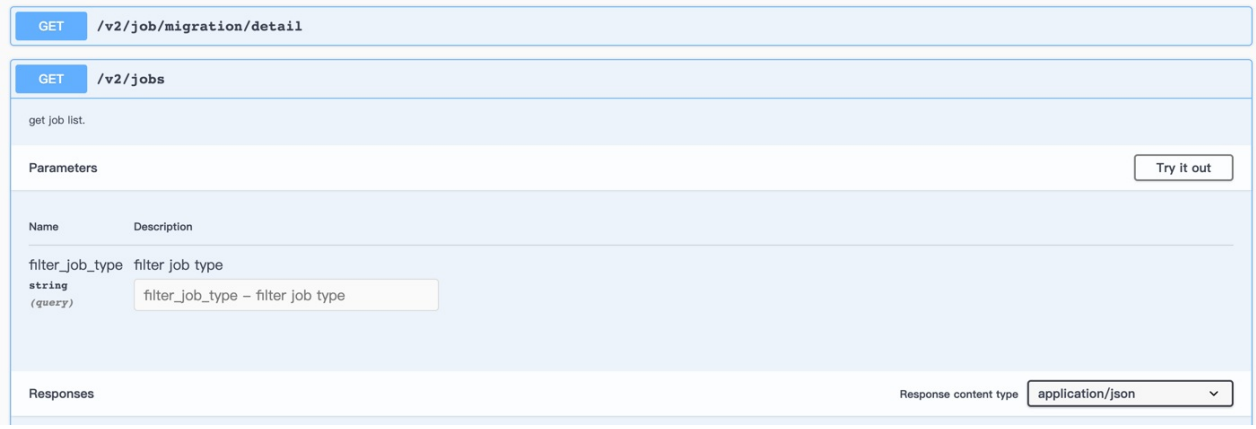
dtle 3.x HTTP API 说明

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：

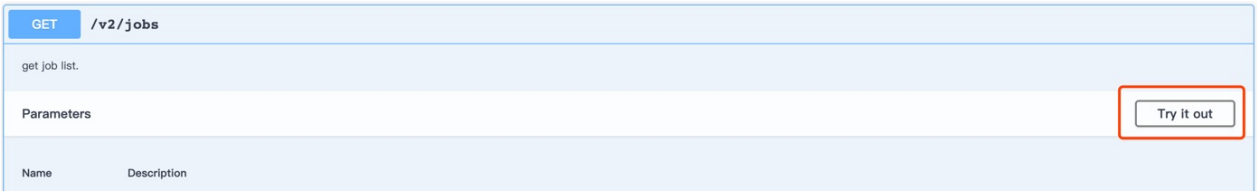


swagger UI 登录

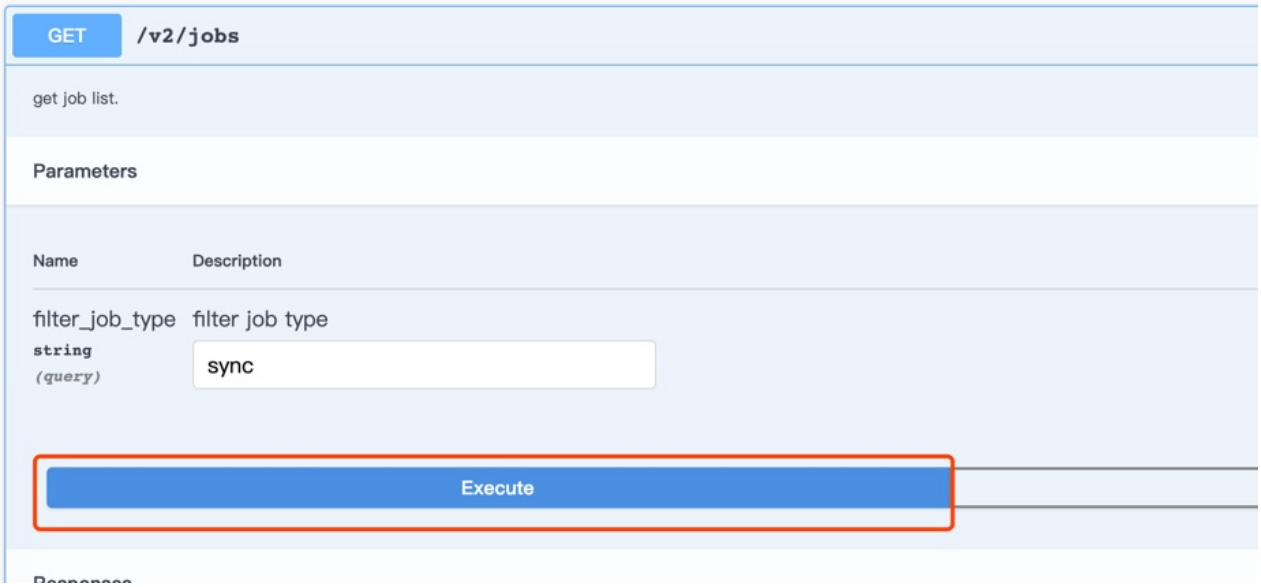
通过swagger UI调用API

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：

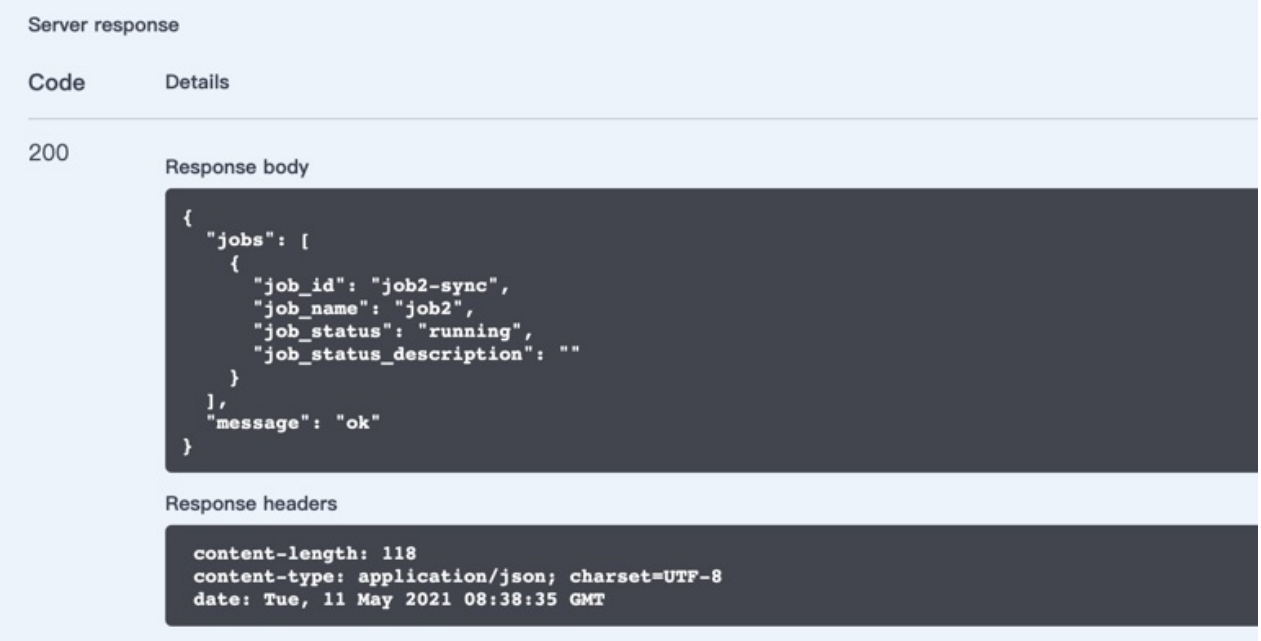
1 点击"Try it out"，进入调试模式



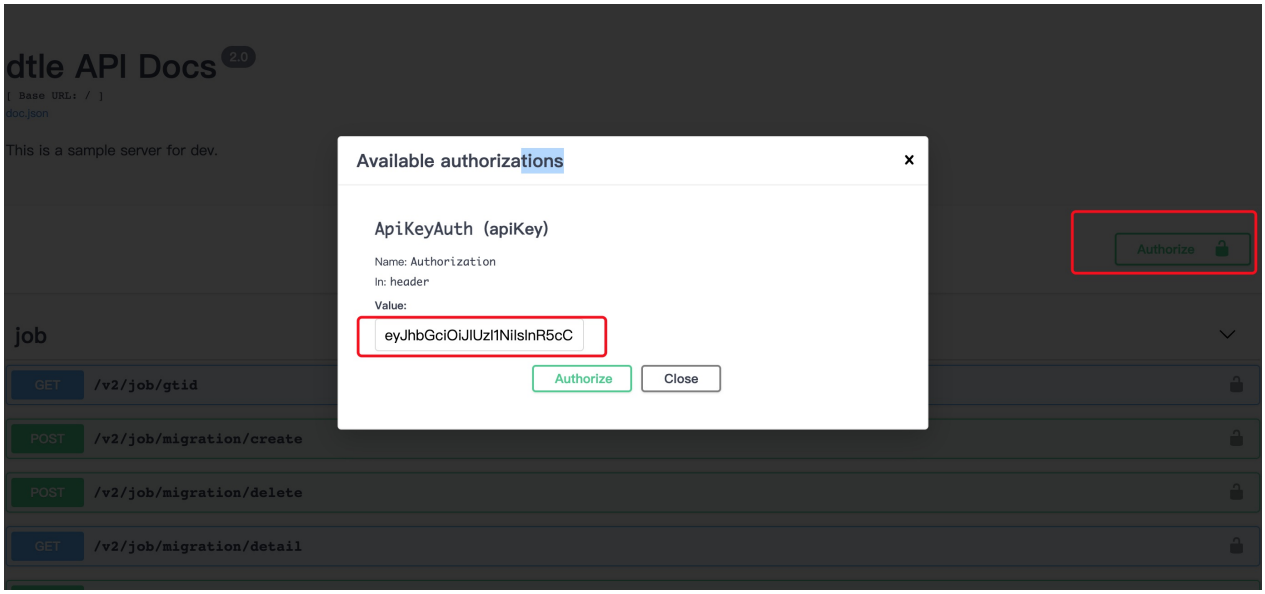
2 填写请求参数后点击"Execute"调用API



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



获取验证码

API: `GET /v2/login/captcha`

请求参数说明

参数名	必填?	类型	默认值	说明
captcha_type	是	String	"default"	获取校验码类型

响应参数说明

参数名	类型	说明
id	String	校验码id
data_scheme	String	Data URI scheme，复制到浏览器的地址栏，获取校验码

样例

response:

```
{
    "id":"BCOUfuFCdR2m6DRmCov8",
    "data_scheme":"
81BMVEUAAAAZR2oTQWQ8ao0eTG82ZIcALlF3pcgWRGcbSWwiUHMGNFdNe56HtdgxX4IBL1I1Y4ZDcZRgjrErWXwXR
WgmVHdNe54+bI8qWHtEcpV1o8YfTXCArtFejK9Bb5JunL+DsdRhj7JikLNlk7ZIdpksWn1+rM+Ittk6aIsOPF9wns
FEcpUtW35Mep06aIsGNFcJN1pXhagzYYQVQ2aDsdQ0YoVUgqUFM1YgTnFRf6J2pMeJt9pJd5oqWHt9q84VQ2Z9q84
7aYxdi65vncBejK8INll8qs16qMuFs9ZwnsFXhag1Y4ZMep2GtNcbSWxLeZx5p8oCljxgAAAAAXRSTlMAQObYZgAA
BgRJREFUeJzsXFlL60AUnlMRxeXBB1FciCIFFbUuoJVWqqgvhuv//zmXpMnM2SaZrLXe+wk2mfV8+c6cOUm05j8Wi
88OxrxsdbST8KY35U0+P9tnfHnZJuOTk2DGNzchjNXS50o2cbSs8E5w0wC+Op6fCxgP645aEzs74Yxro4jvsHfGPc
/H0Tff/+gHp4s2oAM8FtSdnv4+xo+PhYx7tKQvFPH95QD7828AHIL7HHZqUbeoQdccHi4p49SRaxBeFoU5pYzn71z
BcylZ2ZzubwMYGGheW9mPVbxnnwda5V1+8Np0mnAkpAaDgSAcoG7I/eb7+5zxwYHC+O4uY/z6WsK4iaMBGyhl7BZr
+AxBTxRaUVhYF4gs+AAusL6cHW/gKUpQ+4lCRQDaLKp0M45bPhAOwenxxsaGq/ohAKSwtMq7iaA+thcdACncSrhyu
FPK4sC+1AE5L28WCERgsI155/yo1c3IxiaEOA5mbGCWGAbKzgnkB1cAETjrpTSyh2i85migcGLUbDazZhGDgAJVON
MBfP5KFjcZboG5x5zizJ1i24HpSIITctf8TEskjVvZLqAtMoBp2mmVVOScIZxkn6nOTGqwPjDnW+u2sG2oTqxW0iw
iMzt9T8EcFS1oMO6Irvme6CnQwhSuU89y8lhhADjNm1CxwTpAdrrIOyUeQITf8mTYs1Ml42RPHpGT43F4aGhi9VX9
rjzVYE4sFHZ6snGSklO8P9MIx1dDfYsTvle1GeuWozQCaBWKPGovVJ3+jmzeqs1QG1d07ipdZTLJMmM3rFuXykbqS
OKtJ4oiGaRa2YapbTX7Ict5JVq8vm4G+N1SrrBk3Exh5QKGj1eksEscAIUi0J2SKfw1cH7NnLop3zMxCHj9WryJLF
E423sM3lGNugyBhqavwWBgs+ysQnGcikiHOzs70/zLaKMq75pLFcZ0HVF9EduTOP4a4JieTePO6/Gdf5ypdaCGr0K
FHygnW+vbil1pZPili2kcwzZVW3KOLNwW72hqNq8OlDV6eHjIS1CdcocLXM8oinxeTlt6A0CAjXB7e1va0BP/I3e4
61cY1LsadRFHssigkE1tgqNsWQuMClikCpdDHTkVZI7d3V1tDeerTzIOXcSuQCQdR0dHuZPQAUYjwRgNEegU6sgeh
d0cq8ZFPi2F9GxM/NrJB4Np9DqyIZWNMuITWU8rJrnC+hU2Zlamp6urqyjB8OSQ04L9zBWI+00Uw0aAygBHh/nhJD
Cir6ysKJbwnhNslGs5b7zKjBY07s10yhmLSWyMJ7dNVq7RaKRdI3s4mUwC3Tjne8zKcffJJGds8K2CWC66xPf392Z
qlBWr9bQpCQ8JI9TQo7CKdU/58fExmx1bP3FVIjAT0WVwMonCekzGfO0zAkyoRLMQD15f9zIWlhqu2Q29sqb4iQ8p
0VYsiU4AeI16nvLXgY+vBDAjIH15RVe4UFgSHrpgxC8NSdFwRq6FkZ6gKowsNfymP2aRaDgc2p0Ya5orr7xhswrbo
rWO2PlAaGLzpA5xHNMrYIZkcdAAjO+HLfbF2l1bq8ZYz8FqwxmuOV5MF6dwf2BPnEV42t/fF1NW5CtzsGZArw8Vxs
R+xf8FWT6E5FsVLfPFgcWTJLNoR2u4tIt8dZTjPvvc8NRjhWWl+MFVnOyPQJIaJchfx0vIt/mk1u+vKMUIt6f7P7H
xKjxFxwW5QdGLsOqi+v7EZlx5pMqYTqe0wGu9n3ANePiOLeM/zefwYBrcUlnBbcPx/VPKuLtLshCU8y2/JG0h6mui
YvTHN/ohjHvDv8Y3EDOl7EFr+Na9LX3A/iEUQv6kn+DtTWN83ooRm7LopZWBNTRS+Py8Dcabm4Lxy0t3jBthCRUux
tOC5l0Unp7CGIe8TVoOEL7XehswAe8L28NWu8PxR8Zm2x5dX6uMk5uU3viOzdZWq4zxS4EU29uIsd6nx0cJ4+G4R4
W7x0dZg/6/dqBTfHyUM84P9jq3pg+U8rXY21smxhfp72ZfidI9329RUvd/ly4uLsq+EuUH4PubMw777zQNbSjcPZo
r/DcAAP//7W9H1fSQTvcAAAAASUVORK5CYII=",
    "message":"ok"
}
```


用户登录

API: `POST /v2/login`

请求参数说明

参数名	必填?	类型	默认值	说明
username	是	String	""	用户名
tenant	是	String	""	租户名
password	是	String	""	密码
captcha_id	是	String	""	验证码id
captcha	是	String	""	验证码

响应参数说明

token 接口调用时候需要在header中携带

样例

params:

```
{
     "username":"admin",
     "password":"qrzOYInKluhHI2SkEQssfrNCB+ee9rLGzLEGZMXANPU4npTlFmS/hb0qs3ahn6S4tG/RvoUD
wsTdUj77Lbrt1hQ+lJT3JVnXCoj4bxQG2W2W8CAzDLMj9Dmzp9iOt7iEPCgonpwOuaMYwjXNnUN05IvI3hcCIaR02
EoW1NqR3Ow=",
     "captcha":"00036",
     "tenant":"platform",
     "captcha_id":"BCOUfuFCdR2m6DRmCov8"
 }
```

response:

```
{
    "message":"ok",
    "data":{
        "token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE2MzEwMDQ5NDQsImdyb3VwIjo
icGxhdGZvcm0iLCJuYW1lIjoiYWRtaW4ifQ.iZYXNXXZqY5FVmexPY3yD2ZiO4xjeOLmdTq6pxvU4sE"
    }
}
```

创建 一个迁移作业

API: `POST /v2/job/migration/create`

创建的job类型为 `migration`

请求参数说明

参数名	必填?	类型	默认值	说明
migration_job_config	是	JSON Object	""	创建job所需的配置

`migration_job_config` 字段说明:

参数名	必填?	类型	默认值	说明
job_id	是	String	""	创建job时会指定job id为 `"job_id-{job类型}"`
failover	否	bool	true	是否支持故障转移。设置failover=false时，该job的所有task不启用故障转移，要求同时填写task的 `node_id` 以指定每个task运行的节点，如果运行节点发生故障，不会将task转移到其它节点运行；设置failover=true时，启用故障转移。
task_step_name	否	String	""	任务步骤名。表示该任务为全量复制 job_stage_full，增量复制 job_stage_incr，增量加全量 all
is_mysql_password_encrypted	否	bool	false	MySQL密码是否经过加密。设置 is_mysql_password_encrypted=true时，认为所填写的 `mysql_password` 经过两次加密（先是rsa/none/pkcs1 padding 加密，再做base64编码），要求将rsa密钥以pem格式保存在文件中，再将文件绝对路径配置到nomad.hcl的 `rsa_private_key_path` 参数（配置方法见"节点配置"）；设置 is_mysql_password_encrypted=false 时， `mysql_password` 填写明文密码。
dest_task	是	JSON Object	""	目标端配置，参数说明参考"作业配置"
src_task	是	JSON Object	""	源端配置，参数说明参考"作业配置"
retry	否	Int	2	重试次数。参数说明参考restart字段 "作业配置"。
reverse	否	bool	true	该任务是否为反向复制标志。设置 reverse=true时，同时配合src_task字段中的task_name和wait_on_job，表示该任务是task_name的反向复制任务，需要再次调用启动反向任务接口才会运行当前任务并且停止原任务（task_name）

响应参数说明

返回请求的job配置

样例

params:

`migration_job_config` :

```
{
    "job_id":"demo",
    "src_task":{
        "mysql_connection_config":{
            "mysql_host":"10.186.63.18",
            "mysql_port":33061,
            "mysql_user":"root",
            "mysql_password":"tOVPvMXeG3ab9JX66sNXc/X4E+3fOTA4kWCvHrw7PGxUrlkMx+naHUQFDqP
v/WKym6BhjZ4kh5GNOsTk/rU4vc+LDn4eLauydTlLkJ0ZUHF5YF3OIjFO/4w3UDHOroUbEtizO/cXljmEu+J1jovQ
tsqQGE8mVfqq9iQsb2pB1EA="
        },
        "node_id":"85148787-a6ba-016f-d9d2-3bc3522a573a",
        "gtid":"",
        "drop_table_if_exists":true,
        "binlog_relay":false,
        "repl_chan_buffer_size":120,
        "group_max_size":1,
        "group_timeout":100,
        "chunk_size":2000,
        "skip_create_db_table":false,
        "task_name":"src",
        "replicate_do_db":[

        ]
    },
    "is_mysql_password_encrypted":true,
    "dest_task":{
        "mysql_connection_config":{
            "mysql_host":"10.186.63.18",
            "mysql_port":33062,
            "mysql_user":"root",
            "mysql_password":"pytsz1GOSL7jmW+U2PLJSenEPCZGFJqP0PP1DDe/r+D8n8n6pXtCPu8FlQp
CyMjdpWGV30Hv3RLmIH4MziLKeKJ/Lz/ZuoDVvAQEL/Hym7TSmriK8WJYQQKqgUKjmQLuBG7zQZo8nwD/+i0SBDvM
C+i0vp9InVw1U7zK/pAclhw="
        },
        "node_id":"85148787-a6ba-016f-d9d2-3bc3522a573a",
        "use_my_sql_dependency":true,
        "dependency_history_size":2500,
        "parallel_workers":1,
        "task_name":"dest"
    },
    "task_step_name":"job_stage_full",
    "failover":true,
    "retry":2
}
```


response:

```
{
    "job":{
        "job_id":"demo-migration",
        "task_step_name":"job_stage_full",
        "reverse":false,
        "failover":true,
        "is_mysql_password_encrypted":true,
        "src_task":{
            "task_name":"src",
            "node_id":"85148787-a6ba-016f-d9d2-3bc3522a573a",
            "gtid":"",
            "group_max_size":1,
            "chunk_size":2000,
            "drop_table_if_exists":true,
            "skip_create_db_table":false,
            "repl_chan_buffer_size":120,
            "replicate_do_db":[

            ],
            "replicate_ignore_db":null,
            "mysql_connection_config":{
                "mysql_host":"10.186.63.18",
                "mysql_port":33061,
                "mysql_user":"root",
                "mysql_password":"*"
            },
            "binlog_relay":false,
            "group_timeout":100,
            "wait_on_job":"",
            "auto_gtid":false
        },
        "dest_task":{
            "task_name":"dest",
            "node_id":"85148787-a6ba-016f-d9d2-3bc3522a573a",
            "parallel_workers":1,
            "mysql_connection_config":{
                "mysql_host":"10.186.63.18",
                "mysql_port":33062,
                "mysql_user":"root",
                "mysql_password":"*"
            },
            "use_my_sql_dependency":true,
            "dependency_history_size":2500
        },
        "retry":2
    },
    "eval_create_index":4798,
    "job_modify_index":4798,
    "message":"ok"
}
```

更新 一个迁移作业

API: `POST /v2/job/migration/update`

更新的job类型为 `migration`

请求参数说明

参数名	必填?	类型	默认值	说明
migration_job_config	是	JSON Object	""	创建job所需的配置

`migration_job_config` 字段说明:

参数名	必填?	类型	默认值	说明
job_id	是	String	""	创建job时会指定job id为 `"job_id-{job类型}"`
failover	否	bool	true	是否支持故障转移。设置failover=false时，该job的所有task不启用故障转移，要求同时填写task的 `node_id` 以指定每个task运行的节点，如果运行节点发生故障，不会将task转移到其它节点运行；设置failover=true时，启用故障转移。
task_step_name	否	String	""	任务步骤名。表示该任务为全量复制 job_stage_full，增量复制 job_stage_incr，增量加全量 all
is_mysql_password_encrypted	否	bool	false	MySQL密码是否经过加密。设置 is_mysql_password_encrypted=true时，认为所填写的 `mysql_password` 经过两次加密（先是rsa/none/pkcs1 padding 加密，再做base64编码），要求将rsa密钥以pem格式保存在文件中，再将文件绝对路径配置到nomad.hcl的 `rsa_private_key_path` 参数（配置方法见"节点配置"）；设置 is_mysql_password_encrypted=false 时， `mysql_password` 填写明文密码。
dest_task	是	JSON Object	""	目标端配置，参数说明参考"作业配置"
src_task	是	JSON Object	""	源端配置，参数说明参考"作业配置"
retry	否	Int	2	重试次数。参数说明参考restart字段 "作业配置"。
reverse	否	bool	true	该任务是否为反向复制标志。设置 reverse=true时，同时配合src_task字段中的task_name和wait_on_job，表示该任务是task_name的反向复制任务，需要再次调用启动反向任务接口才会运行当前任务并且停止原任务（task_name）

响应参数说明

返回请求的job配置

样例

params:

`migration_job_config` :


```
{
    "job_id":"demo-migration",
    "src_task":{
        "mysql_connection_config":{
            "mysql_host":"10.186.63.18",
            "mysql_port":33061,
            "mysql_user":"root",
            "mysql_password":"PCvXTB1/kgoxBgJUArJLsdeWo04FgFDi+xeAAgvo+yoz7KUwmHFGE51WRWG
rpUcWXgSQGnwuyuTzEwgQv6/ZDAkZZfO+T/F/67eiWT7QhZatn6XCiX0ElpC3rPi2YkOpsyL7pN0d52IfcdrkBrdR
h3vBe3uW9fYEAfSbHDET26Q="
        },
        "node_id":"85148787-a6ba-016f-d9d2-3bc3522a573a",
        "gtid":"",
        "replicate_ignore_db":[

        ],
        "drop_table_if_exists":true,
        "binlog_relay":false,
        "repl_chan_buffer_size":120,
        "group_max_size":1,
        "group_timeout":100,
        "chunk_size":2000,
        "skip_create_db_table":false,
        "task_name":"src",
        "replicate_do_db":[
            {
                "table_schema":"demo",
                "tables":[
                    {
                        "table_name":"demo_tbl"
                    }
                ]
            }
        ]
    },
    "is_mysql_password_encrypted":true,
    "dest_task":{
        "mysql_connection_config":{
            "mysql_host":"10.186.63.18",
            "mysql_port":33062,
            "mysql_user":"root",
            "mysql_password":"DRNmjvZs71Kv16acHXAS9SSqS5slxLQ8j48SU0lfH48QBVxi4tTAg2hp+go
fR+9NRT/HaZ5AraMmSpW9j0o379QymUaIHKXK25PB5goj8OBzg1NZD3sm865RU+VQFeEf87B2eyNqGBmOczS1s2ws
2JitEGhiothaCRRHbo52Lp4="
```

```
        },
        "node_id":"85148787-a6ba-016f-d9d2-3bc3522a573a",
        "use_my_sql_dependency":true,
        "dependency_history_size":2500,
        "parallel_workers":1,
        "task_name":"dest"
    },
    "task_step_name":"job_stage_full",
    "failover":true,
    "retry":2
}
```

response:

```
{
    "job":{
        "job_id":"demo-migration",
        "task_step_name":"job_stage_full",
        "reverse":false,
        "failover":true,
        "is_mysql_password_encrypted":true,
        "src_task":{
            "task_name":"src",
            "node_id":"85148787-a6ba-016f-d9d2-3bc3522a573a",
            "gtid":"",
            "group_max_size":1,
            "chunk_size":2000,
            "drop_table_if_exists":true,
            "skip_create_db_table":false,
            "repl_chan_buffer_size":120,
            "replicate_do_db":[
                {
                    "table_schema":"demo",
                    "table_schema_regex":"",
                    "table_schema_rename":"",
                    "tables":[
                        {
                            "table_name":"demo_tbl",
                            "table_regex":"",
                            "table_rename":"",
                            "column_map_from":null,
                            "where":""
                        }
                    ]
                }
            ],
            "replicate_ignore_db":[

            ],
            "mysql_connection_config":{
                "mysql_host":"10.186.63.18",
                "mysql_port":33061,
```

```
                "mysql_user":"root",
                "mysql_password":"*"
            },
            "binlog_relay":false,
            "group_timeout":100,
            "wait_on_job":"",
            "auto_gtid":false
        },
        "dest_task":{
            "task_name":"dest",
            "node_id":"85148787-a6ba-016f-d9d2-3bc3522a573a",
            "parallel_workers":1,
            "mysql_connection_config":{
                "mysql_host":"10.186.63.18",
                "mysql_port":33062,
                "mysql_user":"root",
                "mysql_password":"*"
            },
            "use_my_sql_dependency":true,
            "dependency_history_size":2500
        },
        "retry":2
    },
    "eval_create_index":4816,
    "job_modify_index":4816,
    "message":"ok"
}
```


创建 一个同步作业

API: `POST /v2/job/sync/create`

创建的job类型为 `sync`

请求参数说明

参数名	必填?	类型	默认值	说明
sync_job_config	是	JSON Object	""	创建job所需的配置

`sync_job_config` 字段说明:

参数名	必填?	类型	默认值	说明
job_id	是	String	""	创建job时会指定job id为 `"job_id-{job类型}"`
failover	否	bool	true	是否支持故障转移。设置failover=false时，该job的所有task不启用故障转移，要求同时填写task的 `node_id` 以指定每个task运行的节点，如果运行节点发生故障，不会将task转移到其它节点运行；设置failover=true时，启用故障转移。
task_step_name	否	String	""	任务步骤名。表示该任务为全量复制 job_stage_full，增量复制 job_stage_incr，增量加全量 all
is_mysql_password_encrypted	否	bool	false	MySQL密码是否经过加密。设置 is_mysql_password_encrypted=true时，认为所填写的 `mysql_password` 经过两次加密（先是rsa/none/pkcs1 padding 加密，再做base64编码），要求将rsa密钥以pem格式保存在文件中，再将文件绝对路径配置到nomad.hcl的 `rsa_private_key_path` 参数（配置方法见"节点配置"）；设置 is_mysql_password_encrypted=false 时， `mysql_password` 填写明文密码。
dest_task	是	JSON Object	""	目标端配置，参数说明参考"作业配置"
src_task	是	JSON Object	""	源端配置，参数说明参考"作业配置"
retry	否	Int	2	重试次数。参数说明参考restart字段 "作业配置"。
reverse	否	bool	true	该任务是否为反向复制标志。设置 reverse=true时，同时配合src_task字段中的task_name和wait_on_job，表示该任务是task_name的反向复制任务，需要再次调用启动反向任务接口才会运行当前任务并且停止原任务（task_name）

响应参数说明

返回请求的job配置

样例

params:

```
sync_job_config :
```

```
{
    "job_id":"demo",
    "src_task":{
        "mysql_connection_config":{
            "mysql_host":"10.186.63.18",
            "mysql_port":33061,
            "mysql_user":"root",
            "mysql_password":"OIjXGJYydZHFNN3TyPBAMFUfsfcIbXBW3sBKjf2c0wvgXrmqI44xldxr6e5
vqzBUwLpEMp+fNh/h80xwWbSJm4wjiJjQPSwNaj5lN/6+yDMM+LqiuJC0WqOG/L2Unk4AsfEvHNxeZO0sXVHDsM+m
VIJnS8D0dnYzZsF874m3nTs="
        },
        "node_id":"85148787-a6ba-016f-d9d2-3bc3522a573a",
        "gtid":"",
        "drop_table_if_exists":true,
        "binlog_relay":false,
        "repl_chan_buffer_size":120,
        "group_max_size":1,
        "group_timeout":100,
        "chunk_size":2000,
        "skip_create_db_table":false,
        "task_name":"src",
        "replicate_do_db":[

        ]
    },
    "is_mysql_password_encrypted":true,
    "dest_task":{
        "mysql_connection_config":{
            "mysql_host":"10.186.63.18",
            "mysql_port":33062,
            "mysql_user":"root",
            "mysql_password":"Kql8Oq2X0T0MS8SMBkOKsdt0CU2igpeVSLykQaoJL6NAW8L1hVRFUETRKlF
6kSEa33TTagjoQ6S8rKKLz4YWJnn69TV2RgOkDk98PNE6IFls4y0mm4BbtKOhKKSj6Pr7qcA7Y5OTBrc/Re2VTALL
6fdCw+0Zfw9Mk+vtPuODUoQ="
        },
        "node_id":"85148787-a6ba-016f-d9d2-3bc3522a573a",
        "use_my_sql_dependency":true,
        "dependency_history_size":2500,
        "parallel_workers":1,
        "task_name":"dest"
    },
    "task_step_name":"job_stage_incr",
    "failover":true,
    "retry":2
}
```


response:

```
{
    "job":{
        "job_id":"demo-sync",
        "task_step_name":"job_stage_incr",
        "reverse":false,
        "failover":true,
        "is_mysql_password_encrypted":true,
        "src_task":{
            "task_name":"src",
            "node_id":"85148787-a6ba-016f-d9d2-3bc3522a573a",
            "gtid":"",
            "group_max_size":1,
            "chunk_size":2000,
            "drop_table_if_exists":true,
            "skip_create_db_table":false,
            "repl_chan_buffer_size":120,
            "replicate_do_db":[

            ],
            "replicate_ignore_db":null,
            "mysql_connection_config":{
                "mysql_host":"10.186.63.18",
                "mysql_port":33061,
                "mysql_user":"root",
                "mysql_password":"*"
            },
            "binlog_relay":false,
            "group_timeout":100,
            "wait_on_job":"",
            "auto_gtid":false
        },
        "dest_task":{
            "task_name":"dest",
            "node_id":"85148787-a6ba-016f-d9d2-3bc3522a573a",
            "parallel_workers":1,
            "mysql_connection_config":{
                "mysql_host":"10.186.63.18",
                "mysql_port":33062,
                "mysql_user":"root",
                "mysql_password":"*"
            },
            "use_my_sql_dependency":true,
            "dependency_history_size":2500
        },
        "retry":2
    },
    "eval_create_index":4967,
    "job_modify_index":4967,
    "message":"ok"
}
```

更新 一个同步作业

API: `POST /v2/job/sync/update`

更新的job类型为 `sync`

请求参数说明

参数名	必填?	类型	默认值	说明
sync_job_config	是	JSON Object	""	创建job所需的配置

`sync_job_config` 字段说明:

参数名	必填?	类型	默认值	说明
job_id	是	String	""	创建job时会指定job id为 `"job_id-{job类型}"`
failover	否	bool	true	是否支持故障转移。设置failover=false时，该job的所有task不启用故障转移，要求同时填写task的 `node_id` 以指定每个task运行的节点，如果运行节点发生故障，不会将task转移到其它节点运行；设置failover=true时，启用故障转移。
task_step_name	否	String	""	任务步骤名。表示该任务为全量复制 job_stage_full，增量复制 job_stage_incr，增量加全量 all
is_mysql_password_encrypted	否	bool	false	MySQL密码是否经过加密。设置 is_mysql_password_encrypted=true时，认为所填写的 `mysql_password` 经过两次加密（先是rsa/none/pkcs1 padding 加密，再做base64编码），要求将rsa密钥以pem格式保存在文件中，再将文件绝对路径配置到nomad.hcl的 `rsa_private_key_path` 参数（配置方法见"节点配置"）；设置 is_mysql_password_encrypted=false 时， `mysql_password` 填写明文密码。
dest_task	是	JSON Object	""	目标端配置，参数说明参考"作业配置"
src_task	是	JSON Object	""	源端配置，参数说明参考"作业配置"
retry	否	Int	2	重试次数。参数说明参考restart字段 "作业配置"。
reverse	否	bool	true	该任务是否为反向复制标志。设置 reverse=true时，同时配合src_task字段中的task_name和wait_on_job，表示该任务是task_name的反向复制任务，需要再次调用启动反向任务接口才会运行当前任务并且停止原任务（task_name）

响应参数说明

返回请求的job配置

样例

params:

`sync_job_config` :


```
{
    "job_id":"demo-sync",
    "src_task":{
        "mysql_connection_config":{
            "mysql_host":"10.186.63.18",
            "mysql_port":33061,
            "mysql_user":"root",
            "mysql_password":"H66OYCpS4+ElyGFpUIpUp80v1dKF5aCkOEPfCKUrFpC3xddB0fDsENgI8A5
275ctsOn01O4J4UMw1OZ3AQlY6uwzeMq9T+ekx31v5D2/RjBe557GBtrS8ijKA8pAFfOhUxug/TB0qzBzRekw4KWE
L9PTNKRfX0I54J2IsFSXp48="
        },
        "node_id":"85148787-a6ba-016f-d9d2-3bc3522a573a",
        "gtid":"",
        "replicate_ignore_db":[

        ],
        "drop_table_if_exists":true,
        "binlog_relay":false,
        "repl_chan_buffer_size":120,
        "group_max_size":1,
        "group_timeout":100,
        "chunk_size":2000,
        "skip_create_db_table":false,
        "task_name":"src",
        "replicate_do_db":[
            {
                "table_schema":"demo",
                "tables":[
                    {
                        "table_name":"demo_tbl"
                    }
                ]
            }
        ]
    },
    "is_mysql_password_encrypted":true,
    "dest_task":{
        "mysql_connection_config":{
            "mysql_host":"10.186.63.18",
            "mysql_port":33062,
            "mysql_user":"root",
            "mysql_password":"yvgvYc7vYJkhrMMPmfdfeAzcZZttxB7RwXQiQJpEdAGtfhrqB/IkILYrESH
eVbC7Narf4iKImWpPKwL7baPNp+bNTl1Hw/ZRf5yZKCw6uQeynpwIXhL92RWpDtB9KBtaUBrUzwJmu8If50c2U+S7
S8usfKE/6Y8QM7OaI0wXjHI="
```

```
        },
        "node_id":"85148787-a6ba-016f-d9d2-3bc3522a573a",
        "use_my_sql_dependency":true,
        "dependency_history_size":2500,
        "parallel_workers":1,
        "task_name":"dest"
    },
    "task_step_name":"job_stage_incr",
    "failover":true,
    "retry":2
}
```

response:

```
{
    "job":{
        "job_id":"demo-sync",
        "task_step_name":"job_stage_incr",
        "reverse":false,
        "failover":true,
        "is_mysql_password_encrypted":true,
        "src_task":{
            "task_name":"src",
            "node_id":"85148787-a6ba-016f-d9d2-3bc3522a573a",
            "gtid":"",
            "group_max_size":1,
            "chunk_size":2000,
            "drop_table_if_exists":true,
            "skip_create_db_table":false,
            "repl_chan_buffer_size":120,
            "replicate_do_db":[
                {
                    "table_schema":"demo",
                    "table_schema_regex":"",
                    "table_schema_rename":"",
                    "tables":[
                        {
                            "table_name":"demo_tbl",
                            "table_regex":"",
                            "table_rename":"",
                            "column_map_from":null,
                            "where":""
                        }
                    ]
                }
            ],
            "replicate_ignore_db":[

            ],
            "mysql_connection_config":{
                "mysql_host":"10.186.63.18",
                "mysql_port":33061,
```

```
                "mysql_user":"root",
                "mysql_password":"*"
            },
            "binlog_relay":false,
            "group_timeout":100,
            "wait_on_job":"",
            "auto_gtid":false
        },
        "dest_task":{
            "task_name":"dest",
            "node_id":"85148787-a6ba-016f-d9d2-3bc3522a573a",
            "parallel_workers":1,
            "mysql_connection_config":{
                "mysql_host":"10.186.63.18",
                "mysql_port":33062,
                "mysql_user":"root",
                "mysql_password":"*"
            },
            "use_my_sql_dependency":true,
            "dependency_history_size":2500
        },
        "retry":2
    },
    "eval_create_index":4979,
    "job_modify_index":4979,
    "message":"ok"
}
```


创建 一个订阅作业

API: `POST /v2/job/subscription/create`

创建的job类型为 `subscription`

请求参数说明

参数名	必填?	类型	默认值	说明
subscription_job_config	是	JSON Object	""	创建job所需的配置

`subscription_job_config` 字段说明:

参数名	必填?	类型	默认值	说明
job_id	是	String	""	创建job时会指定job id为 `"job_id-{job类型}"`
task_step_name	否	String	""	任务步骤名。表示该任务为全量复制 job_stage_full，增量复制 job_stage_incr，增量加全量 all
failover	否	bool	true	是否支持故障转移。设置failover=false时，该job的所有task不启用故障转移，要求同时填写task的 `node_id` 以指定每个task运行的节点，如果运行节点发生故障，不会将task转移到其它节点运行；设置failover=true时，启用故障转移。
is_mysql_password_encrypted	否	bool	false	MySQL密码是否经过加密。设置is_mysql_password_encrypted=true时，认为所填写的 `mysql_password` 经过两次加密（先是rsa/none/pkcs1 padding 加密，再做base64编码），要求将rsa密钥以pem格式保存在文件中，再将文件绝对路径配置到nomad.hcl的 `rsa_private_key_path` 参数（配置方法见"节点配置"）；设置is_mysql_password_encrypted=false时， `mysql_password` 填写明文密码。
dest_task	是	JSON Object	""	目标端配置，参数说明参考"作业配置"
src_task	是	JSON Object	""	源端配置，参数说明参考"作业配置"
retry	否	Int	2	重试次数。参数说明参考restart字段 "作业配置"。

响应参数说明

返回请求的job配置

样例

params:

`subscription_job_config` :

```
{
    "job_id":"demo",
    "src_task":{
        "mysql_connection_config":{
            "mysql_host":"10.186.63.18",
            "mysql_port":33061,
            "mysql_user":"root",
            "mysql_password":"2ZT/i7938WoFXs+U8J4W6x7Qm6S6vH2hiq2q9TvM9seYCyedMy7ym1X2h9j
TATFqEkGwP7/kV7rm3G9wgkSH9GVKqR/amHWsgT7gWQejqq4UO2HtTwx3dZfQAt/wT6CDNn8cM3VnXeiPr4K7MuEM
iKb6VDJIcSHIhw8iKqTABzg="
        },
        "node_id":"85148787-a6ba-016f-d9d2-3bc3522a573a",
        "drop_table_if_exists":true,
        "binlog_relay":false,
        "repl_chan_buffer_size":120,
        "group_max_size":1,
        "group_timeout":100,
        "chunk_size":2000,
        "skip_create_db_table":false,
        "task_name":"src",
        "replicate_do_db":[

        ]
    },
    "is_mysql_password_encrypted":true,
    "dest_task":{
        "kafka_broker_addrs":[
            "10.186.63.18:9092"
        ],
        "kafka_topic":"demo",
        "node_id":"85148787-a6ba-016f-d9d2-3bc3522a573a",
        "parallel_workers":1,
        "task_name":"dest"
    },
    "task_step_name":"job_stage_full",
    "failover":true,
    "retry":2
}
```


response:

```
{
    "job":{
        "job_id":"demo-subscription",
        "task_step_name":"job_stage_full",
        "failover":true,
        "is_mysql_password_encrypted":true,
        "src_task":{
            "task_name":"src",
            "node_id":"85148787-a6ba-016f-d9d2-3bc3522a573a",
            "gtid":"",
            "group_max_size":1,
            "chunk_size":2000,
            "drop_table_if_exists":true,
            "skip_create_db_table":false,
            "repl_chan_buffer_size":120,
            "replicate_do_db":[

            ],
            "replicate_ignore_db":null,
            "mysql_connection_config":{
                "mysql_host":"10.186.63.18",
                "mysql_port":33061,
                "mysql_user":"root",
                "mysql_password":"*"
            },
            "binlog_relay":false,
            "group_timeout":100,
            "wait_on_job":"",
            "auto_gtid":false
        },
        "dest_task":{
            "task_name":"dest",
            "node_id":"85148787-a6ba-016f-d9d2-3bc3522a573a",
            "kafka_broker_addrs":[
                "10.186.63.18:9092"
            ],
            "kafka_topic":"demo",
            "message_group_max_size":1,
            "message_group_timeout":100
        },
        "retry":2
    },
    "eval_create_index":4997,
    "job_modify_index":4997,
    "message":"ok"
}
```


更新 一个订阅作业

API: `POST /v2/job/subscription/update`

更新的job类型为 `subscription`

请求参数说明

参数名	必填?	类型	默认值	说明
subscription_job_config	是	JSON Object	""	创建job所需的配置

`subscription_job_config` 字段说明:

参数名	必填?	类型	默认值	说明
job_id	是	String	""	创建job时会指定job id为 `"job_id-{job类型}"`
task_step_name	否	String	""	任务步骤名。表示该任务为全量复制 job_stage_full，增量复制 job_stage_incr，增量加全量 all
failover	否	bool	true	是否支持故障转移。设置failover=false时，该job的所有task不启用故障转移，要求同时填写task的 `node_id` 以指定每个task运行的节点，如果运行节点发生故障，不会将task转移到其它节点运行；设置failover=true时，启用故障转移。
is_mysql_password_encrypted	否	bool	false	MySQL密码是否经过加密。设置is_mysql_password_encrypted=true时，认为所填写的 `mysql_password` 经过两次加密（先是rsa/none/pkcs1 padding 加密，再做base64编码），要求将rsa密钥以pem格式保存在文件中，再将文件绝对路径配置到nomad.hcl的 `rsa_private_key_path` 参数（配置方法见"节点配置"）；设置is_mysql_password_encrypted=false时， `mysql_password` 填写明文密码。
dest_task	是	JSON Object	""	目标端配置，参数说明参考"作业配置"
src_task	是	JSON Object	""	源端配置，参数说明参考"作业配置"
retry	否	Int	2	重试次数。参数说明参考restart字段 "作业配置"。

响应参数说明

返回请求的job配置

样例

params:

`subscription_job_config` :

```
{
    "job_id":"demo",
    "src_task":{
        "mysql_connection_config":{
            "mysql_host":"10.186.63.18",
            "mysql_port":33061,
            "mysql_user":"root",
            "mysql_password":"2ZT/i7938WoFXs+U8J4W6x7Qm6S6vH2hiq2q9TvM9seYCyedMy7ym1X2h9j
TATFqEkGwP7/kV7rm3G9wgkSH9GVKqR/amHWsgT7gWQejqq4UO2HtTwx3dZfQAt/wT6CDNn8cM3VnXeiPr4K7MuEM
iKb6VDJIcSHIhw8iKqTABzg="
        },
        "node_id":"85148787-a6ba-016f-d9d2-3bc3522a573a",
        "drop_table_if_exists":true,
        "binlog_relay":false,
        "repl_chan_buffer_size":120,
        "group_max_size":1,
        "group_timeout":100,
        "chunk_size":2000,
        "skip_create_db_table":false,
        "task_name":"src",
        "replicate_do_db":[

        ]
    },
    "is_mysql_password_encrypted":true,
    "dest_task":{
        "kafka_broker_addrs":[
            "10.186.63.18:9092"
        ],
        "kafka_topic":"demo",
        "node_id":"85148787-a6ba-016f-d9d2-3bc3522a573a",
        "parallel_workers":1,
        "task_name":"dest"
    },
    "task_step_name":"job_stage_full",
    "failover":true,
    "retry":2
}
```


response:

```
{
    "job":{
        "job_id":"demo-subscription",
        "task_step_name":"job_stage_full",
        "failover":true,
        "is_mysql_password_encrypted":true,
        "src_task":{
            "task_name":"src",
            "node_id":"85148787-a6ba-016f-d9d2-3bc3522a573a",
            "gtid":"",
            "group_max_size":1,
            "chunk_size":2000,
            "drop_table_if_exists":true,
            "skip_create_db_table":false,
            "repl_chan_buffer_size":120,
            "replicate_do_db":[

            ],
            "replicate_ignore_db":null,
            "mysql_connection_config":{
                "mysql_host":"10.186.63.18",
                "mysql_port":33061,
                "mysql_user":"root",
                "mysql_password":"*"
            },
            "binlog_relay":false,
            "group_timeout":100,
            "wait_on_job":"",
            "auto_gtid":false
        },
        "dest_task":{
            "task_name":"dest",
            "node_id":"85148787-a6ba-016f-d9d2-3bc3522a573a",
            "kafka_broker_addrs":[
                "10.186.63.18:9092"
            ],
            "kafka_topic":"demo",
            "message_group_max_size":1,
            "message_group_timeout":100
        },
        "retry":2
    },
    "eval_create_index":4997,
    "job_modify_index":4997,
    "message":"ok"
}
```


暂停一个迁移/同步/订阅作业

API: `POST /v2/job/job_type/pause`

请求参数说明

参数名	必填?	类型	默认值	说明
job_id	是	String	""	要暂停的job id

响应参数说明

返回暂停结果

样例

params:

`job_id : demo-migration`

response:

```
{
  "message": "ok"
}
```

恢复一个迁移/同步/订阅作业

API: `POST /v2/job/job_type/resume`

请求参数说明

参数名	必填?	类型	默认值	说明
job_id	是	String	""	要恢复的job id

响应参数说明

返回恢复结果

样例

params:

`job_id : demo-migration`

response:

```
{
  "message": "ok"
}
```

快速创建一个迁移/同步反向复制作业

API: `POST /v2/job/job_type/reverse`

请求参数说明

参数名	必填?	类型	默认值	说明
job_id	是	String	""	反向复制的源job id
reverse_config	否	JSON	""	新任务的源端连接方式和目标端连接方式

响应参数说明

返回创建结果

样例

params:


```
{
    "job_id":"demo-migration",
    "reverse_config":{
        "src_user":"root",
        "src_pwd":"eI7vOGHFBeKiQraRyG3ELhBQV0PZZvW+yOTSbHTF9+8SJQ8Pr/EetkYm4WcG3Lx6KOhDBd
iou2QCikIppovUgcTDaEovA7cpeU+FzvLIj2VccQMSodQuAb+/oniols+On5QYNPlhVRM5xCn0FH9Zm57dcSoCDMe
OiCxj3APeMJA=",
        "dest_user":"root",
        "dst_pwd":"2a08zoW6VY3K5PQFgFfQdH7fOut0LnJT9prO4/bcYcAs2MlCS1JHANoul7yHPE7T4nKHQL
VLn0xheax6I7MJudtzbewuVSx6zt0t7kWvCayB4yQwwzT/2wKlEqxc+u9U0n0VtnmIzuiklFxYPOiECZxzFFOt3Tl
7EpioeBD6BvY=",
        "is_mysql_password_encrypted":true
    }
}
```


response:


```
{
  "message": "ok"
}
```


启动一个迁移/同步反向复制作业

API: `POST /v2/job/job_type/reverse_start`

请求参数说明

参数名	必填?	类型	默认值	说明
job_id	是	String	""	反向复制的源job id

响应参数说明

返回创建结果

样例

params:

`job_id : reverse-demo-migration`

response:

```
{
  "message": "ok"
}
```

删除一个迁移/同步/订阅作业

API: `POST /v2/job/job_type/delete`

请求参数说明

参数名	必填?	类型	默认值	说明
job_id	是	String	""	要删除的job id

响应参数说明

返回删除结果

样例

params:

`job_id : demo-migration`

response:

```
{
  "message": "ok"
}
```

列出迁移/同步/订阅下所有作业

API: `GET /v2/jobs/job_type`

请求参数说明

参数名	必填?	类型	默认值	说明
filter_job_name	否	String	""	过滤器，根据job名过滤请求结果
filter_job_status	否	String	""	过滤器，根据job状态过滤请求结果
order_by	否	String	""	排序

响应参数说明

参数名	类型	说明
jobs.job_id	String	job id
jobs.job_status	String	job运行状态，注意区分job状态和task状态，如果某个job下有task失败，此时获取到的job状态仍可能为"running"，因为job负责管理和调度task，它确实是处于正常运行状态，它可能正在对失败的task做重试、重新分配等工作
jobs.topic	String	订阅任务专用topic
jobs.job_create_time	String	任务创建时间
jobs.src_addr_list	[]String	源端地址
jobs.dst_addr_list	[]String	目标端地址
jobs.user	String	创建任务用户

样例

params:

`filter_job_name`:"" `filter_job_status`:"" `order_by`:""

response:


```
{
    "jobs":[
        {
            "job_id":"demo-migration",
            "job_status":"running",
            "topic":"",
            "job_create_time":"2021-09-06T03:26:41Z",
            "src_addr_list":[
                "10.186.63.18"
            ],
            "dst_addr_list":[
                "10.186.63.18"
            ],
            "user":"platform:admin",
            "job_steps":[
                {
                    "step_name":"job_stage_full",
                    "step_status":"start",
                    "step_schedule":0,
                    "job_create_time":"2021-09-06T03:26:41Z"
                }
            ]
        }
    ],
    "message":"ok"
}
```


获取某个迁移作业的详细信息

API: `GET /v2/job/migration/detail`

请求参数说明

参数名	必填?	类型	默认值	说明
job_id	是	String	""	要查询的job id

样例

params:

`job_id` : `demo-migration`

response:


```
{
    "basic_task_profile":{
        "job_base_info":{
            "job_id":"demo-migration",
            "subscription_topic":"",
            "job_status":"running",
            "job_create_time":"2021-09-06T03:26:41Z",
            "job_steps":[
                {
                    "step_name":"job_stage_full",
                    "step_status":"start",
                    "step_schedule":0,
                    "job_create_time":"2021-09-06T03:26:41Z"
                }
            ],
            "delay":0
        },
        "dtle_node_infos":[
            {
                "node_addr":"127.0.0.1",
                "node_id":"85148787-a6ba-016f-d9d2-3bc3522a573a",
                "data_source":"10.186.63.18:33061",
                "source":"src"
            },
            {
                "node_addr":"127.0.0.1",
                "node_id":"85148787-a6ba-016f-d9d2-3bc3522a573a",
                "data_source":"10.186.63.18:33062",
                "source":"dst"
            }
        ],
        "connection_info":{
            "src_data_base":{
                "mysql_host":"10.186.63.18",
                "mysql_port":33061,
                "mysql_user":"root",
                "mysql_password":"*"
            },
```

```
            "dst_data_base":{
                "mysql_host":"10.186.63.18",
                "mysql_port":33062,
                "mysql_user":"root",
                "mysql_password":"*"
            },
            "dst_kafka":{
                "task_name":"",
                "kafka_broker_addrs":null,
                "kafka_topic":"",
                "message_group_max_size":0,
                "message_group_timeout":0
            }
        },
        "configuration":{
            "binlog_relay":false,
            "fail_over":true,
            "retry_times":2,
            "parallel_workers":1,
            "repl_chan_buffer_size":120,
            "group_max_size":1,
            "chunk_size":2000,
            "group_timeout":100,
            "drop_table_if_exists":true,
            "skip_create_db_table":false,
            "use_my_sql_dependency":true,
            "dependency_history_size":2500
        },
        "replicate_do_db":[
            {
                "table_schema":"demo",
                "table_schema_regex":"",
                "table_schema_rename":"",
                "tables":[
                    {
                        "table_name":"demo_tbl",
                        "table_regex":"",
                        "table_rename":"",
                        "column_map_from":null,
                        "where":""
                    }
                ]
            }
        ],
        "replicate_ignore_db":[

        ]
    },
    "task_logs":[
        {
            "task_events":[
                {
                    "event_type":"Received",
```

```
                    "setup_error":"",
                    "message":"Task received by client",
                    "time":"2021-09-06T03:26:41Z"
                },
                {
                    "event_type":"Task Setup",
                    "setup_error":"",
                    "message":"Building Task Directory",
                    "time":"2021-09-06T03:26:41Z"
                },
                {
                    "event_type":"Started",
                    "setup_error":"",
                    "message":"Task started by client",
                    "time":"2021-09-06T03:26:41Z"
                },
                {
                    "event_type":"Signaling",
                    "setup_error":"",
                    "message":"Task being sent a signal",
                    "time":"2021-09-06T05:37:21Z"
                },
                {
                    "event_type":"Signaling",
                    "setup_error":"",
                    "message":"Task being sent a signal",
                    "time":"2021-09-06T05:42:23Z"
                }
            ],
            "node_id":"85148787-a6ba-016f-d9d2-3bc3522a573a",
            "allocation_id":"6a8fe2af-99fd-7880-0989-35a4d3fed0de",
            "address":"127.0.0.1",
            "target":"src"
        },
        {
            "task_events":[
                {
                    "event_type":"Driver",
                    "setup_error":"",
                    "message":"job_stage_incr",
                    "time":"2021-09-06T03:20:08Z"
                },
                {
                    "event_type":"Terminated",
                    "setup_error":"",
                    "message":"Exit Code: 2, Exit Message: \"DstPutNats: NatsAddr changed
to wait. will restart dst\"",
                    "time":"2021-09-06T03:26:41Z"
                },
                {
                    "event_type":"Restarting",
                    "setup_error":"",
                    "message":"Task restarting in 18.388197111s",
```

```
                    "time":"2021-09-06T03:26:41Z"
                },
                {
                    "event_type":"Started",
                    "setup_error":"",
                    "message":"Task started by client",
                    "time":"2021-09-06T03:26:59Z"
                },
                {
                    "event_type":"Driver",
                    "setup_error":"",
                    "message":"job_stage_full",
                    "time":"2021-09-06T03:26:59Z"
                },
                {
                    "event_type":"Driver",
                    "setup_error":"",
                    "message":"job_stage_incr",
                    "time":"2021-09-06T03:26:59Z"
                },
                {
                    "event_type":"Signaling",
                    "setup_error":"",
                    "message":"Task being sent a signal",
                    "time":"2021-09-06T05:37:21Z"
                },
                {
                    "event_type":"Signaling",
                    "setup_error":"",
                    "message":"Task being sent a signal",
                    "time":"2021-09-06T05:42:23Z"
                },
                {
                    "event_type":"Driver",
                    "setup_error":"",
                    "message":"job_stage_full",
                    "time":"2021-09-06T05:42:23Z"
                },
                {
                    "event_type":"Driver",
                    "setup_error":"",
                    "message":"job_stage_incr",
                    "time":"2021-09-06T05:42:23Z"
                }
            ],
            "node_id":"85148787-a6ba-016f-d9d2-3bc3522a573a",
            "allocation_id":"f9d36de1-b309-6252-e203-e342d3dcfe80",
            "address":"127.0.0.1",
            "target":"dst"
        }
    ],
    "message":"ok"
}
```

获取某个同步作业的详细信息

API: `GET /v2/job/sync/detail`

请求参数说明

参数名	必填?	类型	默认值	说明
job_id	是	String	""	要查询的job id

样例

params:

`job_id` : `demo-sync`

response:


```
{
    "basic_task_profile":{
        "job_base_info":{
            "job_id":"demo-sync",
            "subscription_topic":"",
            "job_status":"running",
            "job_create_time":"2021-09-06T05:28:55Z",
            "job_steps":[
                {
                    "step_name":"job_stage_incr",
                    "step_status":"start",
                    "step_schedule":0,
                    "job_create_time":"2021-09-06T05:28:55Z"
                }
            ],
            "delay":0
        },
        "dtle_node_infos":[
            {
                "node_addr":"127.0.0.1",
                "node_id":"85148787-a6ba-016f-d9d2-3bc3522a573a",
                "data_source":"10.186.63.18:33061",
                "source":"src"
            },
            {
                "node_addr":"127.0.0.1",
                "node_id":"85148787-a6ba-016f-d9d2-3bc3522a573a",
                "data_source":"10.186.63.18:33062",
                "source":"dst"
            }
        ],
        "connection_info":{
            "src_data_base":{
                "mysql_host":"10.186.63.18",
```

```
                "mysql_port":33061,
                "mysql_user":"root",
                "mysql_password":"*"
            },
            "dst_data_base":{
                "mysql_host":"10.186.63.18",
                "mysql_port":33062,
                "mysql_user":"root",
                "mysql_password":"*"
            },
            "dst_kafka":{
                "task_name":"",
                "kafka_broker_addrs":null,
                "kafka_topic":"",
                "message_group_max_size":0,
                "message_group_timeout":0
            }
        },
        "configuration":{
            "binlog_relay":false,
            "fail_over":true,
            "retry_times":2,
            "parallel_workers":1,
            "repl_chan_buffer_size":120,
            "group_max_size":1,
            "chunk_size":2000,
            "group_timeout":100,
            "drop_table_if_exists":true,
            "skip_create_db_table":false,
            "use_my_sql_dependency":true,
            "dependency_history_size":2500
        },
        "replicate_do_db":[
            {
                "table_schema":"demo",
                "table_schema_regex":"",
                "table_schema_rename":"",
                "tables":[
                    {
                        "table_name":"demo_tbl",
                        "table_regex":"",
                        "table_rename":"",
                        "column_map_from":null,
                        "where":""
                    }
                ]
            }
        ],
        "replicate_ignore_db":[

        ]
    },
    "task_logs":[
```

```
        {
            "task_events":[
                {
                    "event_type":"Received",
                    "setup_error":"",
                    "message":"Task received by client",
                    "time":"2021-09-06T05:27:16Z"
                },
                {
                    "event_type":"Task Setup",
                    "setup_error":"",
                    "message":"Building Task Directory",
                    "time":"2021-09-06T05:27:16Z"
                },
                {
                    "event_type":"Started",
                    "setup_error":"",
                    "message":"Task started by client",
                    "time":"2021-09-06T05:27:16Z"
                },
                {
                    "event_type":"Killing",
                    "setup_error":"",
                    "message":"Sent interrupt. Waiting 5s before force killing",
                    "time":"2021-09-06T05:28:55Z"
                },
                {
                    "event_type":"Terminated",
                    "setup_error":"",
                    "message":"Exit Code: 0",
                    "time":"2021-09-06T05:28:55Z"
                },
                {
                    "event_type":"Killed",
                    "setup_error":"",
                    "message":"Task successfully killed",
                    "time":"2021-09-06T05:28:55Z"
                }
            ],
            "node_id":"85148787-a6ba-016f-d9d2-3bc3522a573a",
            "allocation_id":"369abffd-c20f-02a7-d59c-8ba1972fc7da",
            "address":"127.0.0.1",
            "target":"src"
        },
        {
            "task_events":[
                {
                    "event_type":"Received",
                    "setup_error":"",
                    "message":"Task received by client",
                    "time":"2021-09-06T05:28:55Z"
                },
                {
```

```
                    "event_type":"Task Setup",
                    "setup_error":"",
                    "message":"Building Task Directory",
                    "time":"2021-09-06T05:28:55Z"
                },
                {
                    "event_type":"Started",
                    "setup_error":"",
                    "message":"Task started by client",
                    "time":"2021-09-06T05:28:55Z"
                }
            ],
            "node_id":"85148787-a6ba-016f-d9d2-3bc3522a573a",
            "allocation_id":"3fba0699-48ad-2265-30e3-8c767fbeed25",
            "address":"127.0.0.1",
            "target":"src"
        },
        {
            "task_events":[
                {
                    "event_type":"Received",
                    "setup_error":"",
                    "message":"Task received by client",
                    "time":"2021-09-06T05:27:16Z"
                },
                {
                    "event_type":"Task Setup",
                    "setup_error":"",
                    "message":"Building Task Directory",
                    "time":"2021-09-06T05:27:16Z"
                },
                {
                    "event_type":"Started",
                    "setup_error":"",
                    "message":"Task started by client",
                    "time":"2021-09-06T05:27:16Z"
                },
                {
                    "event_type":"Driver",
                    "setup_error":"",
                    "message":"job_stage_full",
                    "time":"2021-09-06T05:27:16Z"
                },
                {
                    "event_type":"Driver",
                    "setup_error":"",
                    "message":"job_stage_incr",
                    "time":"2021-09-06T05:27:17Z"
                },
                {
                    "event_type":"Terminated",
                    "setup_error":"",
                    "message":"Exit Code: 2, Exit Message: \"DstPutNats: NatsAddr changed
```

```
  to wait. will restart dst\"",
                    "time":"2021-09-06T05:28:55Z"
                },
                {
                    "event_type":"Restarting",
                    "setup_error":"",
                    "message":"Task restarting in 18.343194092s",
                    "time":"2021-09-06T05:28:55Z"
                },
                {
                    "event_type":"Started",
                    "setup_error":"",
                    "message":"Task started by client",
                    "time":"2021-09-06T05:29:13Z"
                },
                {
                    "event_type":"Driver",
                    "setup_error":"",
                    "message":"job_stage_full",
                    "time":"2021-09-06T05:29:13Z"
                },
                {
                    "event_type":"Driver",
                    "setup_error":"",
                    "message":"job_stage_incr",
                    "time":"2021-09-06T05:29:13Z"
                }
            ],
            "node_id":"85148787-a6ba-016f-d9d2-3bc3522a573a",
            "allocation_id":"25830ed3-52a1-52df-e2b5-b9d0db4f0249",
            "address":"127.0.0.1",
            "target":"dst"
        }
    ],
    "message":"ok"
}
```


获取某个订阅作业的详细信息

API: `GET /v2/job/subscription/detail`

请求参数说明

参数名	必填?	类型	默认值	说明
job_id	是	String	""	要查询的job id

样例

params:

`job_id : demo-subscription`

response:


```
{
    "basic_task_profile":{
        "job_base_info":{
            "job_id":"demo-subscription",
            "subscription_topic":"subTopick",
            "job_status":"running",
            "job_create_time":"2021-09-06T09:03:05Z",
            "job_steps":[
                {
                    "step_name":"job_stage_full",
                    "step_status":"start",
                    "step_schedule":0,
                    "job_create_time":"2021-09-06T09:03:05Z"
                }
            ],
            "delay":0
        },
        "dtle_node_infos":[
            {
                "node_addr":"127.0.0.1",
                "node_id":"ab0a1cf9-6100-fb9e-6e9a-ca2e898b4d57",
                "data_source":"10.186.63.18:33061",
                "source":"src"
            },
            {
                "node_addr":"127.0.0.1",
                "node_id":"ab0a1cf9-6100-fb9e-6e9a-ca2e898b4d57",
                "data_source":"10.186.63.18:9092",
                "source":"dst"
            }
        ],
        "connection_info":{
            "src_data_base":{
                "mysql_host":"10.186.63.18",
                "mysql_port":33061,
                "mysql_user":"root",
                "mysql_password":"*"
            },
            "dst_data_base":{
                "mysql_host":"",
                "mysql_port":0,
                "mysql_user":"",
                "mysql_password":""
            },
            "dst_kafka":{
                "task_name":"dest",
                "kafka_broker_addrs":[
                    "10.186.63.18:9092"
                ],
                "kafka_topic":"subTopick",
                "message_group_max_size":1,
```

```
                "message_group_timeout":100
            }
        },
        "configuration":{
            "binlog_relay":false,
            "fail_over":true,
            "retry_times":2,
            "parallel_workers":0,
            "repl_chan_buffer_size":120,
            "group_max_size":1,
            "chunk_size":2000,
            "group_timeout":100,
            "drop_table_if_exists":true,
            "skip_create_db_table":false,
            "use_my_sql_dependency":false,
            "dependency_history_size":0
        },
        "replicate_do_db":[

        ],
        "replicate_ignore_db":[

        ]
    },
    "task_logs":[
        {
            "task_events":[
                {
                    "event_type":"Received",
                    "setup_error":"",
                    "message":"Task received by client",
                    "time":"2021-09-06T09:03:05Z"
                },
                {
                    "event_type":"Task Setup",
                    "setup_error":"",
                    "message":"Building Task Directory",
                    "time":"2021-09-06T09:03:05Z"
                },
                {
                    "event_type":"Started",
                    "setup_error":"",
                    "message":"Task started by client",
                    "time":"2021-09-06T09:03:05Z"
                }
            ],
            "node_id":"ab0a1cf9-6100-fb9e-6e9a-ca2e898b4d57",
            "allocation_id":"5bce902f-c824-2a95-5b0e-523f3a11c312",
            "address":"127.0.0.1",
            "target":"src"
        },
        {
            "task_events":[
```

```
                {
                    "event_type":"Received",
                    "setup_error":"",
                    "message":"Task received by client",
                    "time":"2021-09-06T09:03:05Z"
                },
                {
                    "event_type":"Task Setup",
                    "setup_error":"",
                    "message":"Building Task Directory",
                    "time":"2021-09-06T09:03:05Z"
                },
                {
                    "event_type":"Started",
                    "setup_error":"",
                    "message":"Task started by client",
                    "time":"2021-09-06T09:03:05Z"
                }
            ],
            "node_id":"ab0a1cf9-6100-fb9e-6e9a-ca2e898b4d57",
            "allocation_id":"b3aa6a5c-8635-d9aa-3c03-6aa6e7322bf7",
            "address":"127.0.0.1",
            "target":"dst"
        }
    ],
    "message":"ok"
}
```


校验作业配置

API: `GET /v2/validation/job`

注意：目前只支持对MySQL-to-MySQL的job校验

请求参数说明

参数名	必填?	类型	默认值	说明
job_config	是	JSON Object	""	参考 `POST /v2/job/migration` 的 `migration_job_config` 配置

响应参数说明

参数名	类型	说明
driver_config_validated	bool	是否校验driver
job_validation_error	String	校验job配置的错误信息
job_validation_warning	String	校验job配置的警告信息
mysql_task_validation_report	JSON Array	task的校验结果
mysql_task_validation_report.task_name	String	task名称
mysql_task_validation_report.binlog_validation	JSON Object	binlog校验结果
mysql_task_validation_report.connection_validation	JSON Object	dtle和数据库的连接校验结果
mysql_task_validation_report.gtid_mode_validation	JSON Object	gtid mode校验结果
mysql_task_validation_report.privileges_validation	JSON Object	用户权限校验结果
mysql_task_validation_report.server_id_validation	JSON Object	MySQL server ID校验结果

样例

params:

`job_config`:

```
  "is_mysql_password_encrypted": true,
  "failover": true,
  "dest_task": {
      "mysql_connection_config": {
      "mysql_host": "10.186.63.28",
      "mysql_password": "vo0+U/SoBFC2QHE1KSunmAIZGpkxpXRWyDJVMNZ3HaQGQRcrVl2w/VuT2gfdD76P
j/Te+24LMBDeoWyNzc8w1NBNqL2+BD7uuLl2aAHPrarhY07LvI2X3NC8n59c0ozN/3+uJPvX7YQ5w8jMjm8xD4qht
I99PkBtC+hfjP1WY38=",
      "mysql_port": 3308,
      "mysql_user": "test"
    },
    "task_name": "dest"
  },
  "job_name": "job1",
  "src_task": {
      "mysql_connection_config": {
      "mysql_host": "10.186.63.28",
      "mysql_password": "vo0+U/SoBFC2QHE1KSunmAIZGpkxpXRWyDJVMNZ3HaQGQRcrVl2w/VuT2gfdD76P
j/Te+24LMBDeoWyNzc8w1NBNqL2+BD7uuLl2aAHPrarhY07LvI2X3NC8n59c0ozN/3+uJPvX7YQ5w8jMjm8xD4qht
I99PkBtC+hfjP1WY38=",
      "mysql_port": 3307,
      "mysql_user": "test"
    },
    "replicate_do_db": [
      {
        "table_schema": "db1",
        "tables": [
          {
            "table_name": "tb1"
          }
        ]
      }
    ],
    "task_name": "src"
  }
}
```


response:

```
{
  "driver_config_validated": true,
  "mysql_task_validation_report": [
    {
      "task_name": "src",
      "connection_validation": {
        "validated": true,
        "error": ""
      },
      "privileges_validation": {
        "validated": true,
        "error": ""
      },
      "gtid_mode_validation": {
        "validated": true,
        "error": ""
      },
      "server_id_validation": {
        "validated": true,
        "error": ""
      },
      "binlog_validation": {
        "validated": true,
        "error": ""
      }
    },
    {
      "task_name": "dest",
      "connection_validation": {
        "validated": true,
        "error": ""
      },
      "privileges_validation": {
        "validated": true,
        "error": ""
      },
      "gtid_mode_validation": null,
      "server_id_validation": null,
      "binlog_validation": null
    }
  ],
  "job_validation_error": "",
  "job_validation_warning": "",
  "message": "ok"
}
```


查看某个任务执行的状态

API: `GET /v2/monitor/task`

请求参数说明

参数名	必填?	类型	默认值	说明
allocation_id	是	String	""	要查询的任务的allocation id
task_name	是	String	""	要查询的任务名称
nomad_http_address	否	String	""	dtle内部使用，可以忽略

响应参数说明

参数名	类型	说明
file	String	正在复制的binlog文件名
gtid_set	String	已复制的GTID集合
delay_count	JSON Object	延时统计
progress_PCT	String	全量复制进度
ETA	String	全量复制预估剩余时间
nats_message_status	JSON Object	Nats网络吞吐数据
stage	String	复制阶段
timestamp	Timestamp	当前时间

样例

params:

```
allocation_id : 0d3939ec-a499-90e3-6bfe-f857448712f9 task_name : src
```

response:

```
{
  "tasks_status": {
    "current_coordinates": {
      "file": "mysql-bin.000001",
      "position": 154,
      "gtid_set": "00003307-1111-1111-1111-111111111111:1-24",
      "relay_master_log_file": "",
      "read_master_log_pos": 0,
      "retrieved_gtid_set": ""
    },
    "delay_count": {
      "num": 0,
      "time": 0
    },
    "progress_PCT": "100.0",
    "exec_master_row_count": 4,
    "exec_master_tx_count": 958859,
    "read_master_row_count": 4,
    "read_master_tx_count": 958859,
    "ETA": "0s",
    "backlog": "0/128",
    "throughput_status": null,
    "nats_message_status": {
      "in_messages": 5,
      "out_messages": 5,
      "in_bytes": 0,
      "out_bytes": 1021,
      "reconnects": 0
    },
    "buffer_status": {
      "binlog_event_queue_size": 0,
      "extractor_tx_queue_size": 0,
      "applier_tx_queue_size": 0,
      "send_by_timeout": 0,
      "send_by_size_full": 0
    },
    "stage": "Master has sent all binlog to slave; waiting for more updates",
    "timestamp": 1620882952091441700
  },
  "message": "ok"
}
```


列出所有节点

API: `GET /v2/nodes`

请求参数说明

没有请求参数

响应参数说明

参数名	类型	说明
nodes	JSON Array	所有节点信息
nodes.node_name	String	节点的hostname
nodes.node_address	String	节点的IP
nodes.node_id	String	节点ID
nodes.node_status	String	节点状态
nodes.node_status_description	String	节点状态描述
nodes.datacenter	String	节点所在的datacenter
nodes.dtle_version	String	节点的dtle版本
nodes.nomad_version	String	节点的nomad版本

样例

response:

```
{
    "nodes":[
        {
            "node_address":"127.0.0.1",
            "node_name":"nomad0",
            "node_id":"ab0a1cf9-6100-fb9e-6e9a-ca2e898b4d57",
            "node_status":"ready",
            "node_status_description":"",
            "datacenter":"dc1",
            "nomad_version":"1.1.2",
            "dtle_version":"3.21.08.0-3.21.08.x-e56bde2"
        }
    ],
    "message":"ok"
}
```

获取**MySQL**实例的**schemas**

API: `GET /v2/mysql/schemas`

请求参数说明

参数名	必填?	类型	默认值	说明
mysql_host	是	String	""	MySQL所在主机IP
mysql_port	是	String	""	MySQL端口
mysql_user	是	String	""	MySQL连接用户
mysql_password	是	String	""	MySQL连接密码
mysql_character_set	否	String	"utf8mb4"	MySQL字符集
is_mysql_password_encrypted	否	String	false	MySQL密码是否经过加密。设置is_mysql_password_encrypted=true时，认为所填写的 `mysql_password` 经过两次加密（先是rsa/none/pkcs1 padding加密，再做base64编码），要求将rsa密钥以pem格式保存在文件中，再将文件绝对路径配置到nomad.hcl的 `rsa_private_key_path` 参数（配置方法见"节点配置"）；设置is_mysql_password_encrypted=false时， `mysql_password` 填写明文密码。

响应参数说明

参数名	类型	说明
schemas	JSON Array	在MySQL实例上执行 `show databases` 获取到的schema
schemas.schema_name	String	schema名称
schemas.tables	JSON Array	在某个schema上执行 `show tables` 获取到的table
schemas.tables.table_name	String	table名称

样例

response:

```
{
  "schemas": [
    {
      "schema_name": "db1",
      "Tables": [
        {
          "table_name": "tb1"
        }
      ]
    },
    {
      "schema_name": "dbtest",
      "Tables": []
    },
    {
      "schema_name": "test",
      "Tables": []
    }
  ],
  "message": "ok"
}
```

验证**MySQL**实例的连通性

API: `GET /v2/mysql/instance_connection`

请求参数说明

参数名	必填?	类型	默认值	说明
mysql_host	是	String	""	MySQL所在主机IP
mysql_port	是	String	""	MySQL端口
mysql_user	是	String	""	MySQL连接用户
mysql_password	是	String	""	MySQL连接密码
is_mysql_password_encrypted	否	String	false	MySQL密码是否经过加密。设置is_mysql_password_encrypted=true时，认为所填写的 `mysql_password` 经过两次加密（先是rsa/none/pkcs1 padding 加密，再做base64编码），要求将rsa密钥以pem格式保存在文件中，再将文件绝对路径配置到nomad.hcl的 `rsa_private_key_path` 参数（配置方法见"节点配置"）；设置is_mysql_password_encrypted=false时， `mysql_password` 填写明文密码。

响应参数说明

联通性结果

样例

params:

```
mysql_host=10.186.63.18&mysql_port=33061&mysql_user=root&mysql_password=ztPqAK1J5uExymUNJvDGMfbNGs7uHrNS%2BTyZodvaR3R24cgOf94AF4w2qp2xqj%2FqC%2FcVtKwWi%2ByhhViWP8UepNjl5hxQMi%2BnYiwCjRHercSRacdX21YulXElhZMp%2F%2BU7KYwWjIznsViVYZkinoXyJShKpmXczyb9UBTpoelqu0w%3D&is_mysql_password_encrypted=true
```

response:

```
{
  "message": "ok"
}
```

获取**MySQL**实例指定**schema**下的**table**的所有**columns**

API: `GET /v2/mysql/columns`

请求参数说明

参数名	必填?	类型	默认值	说明
mysql_host	是	String	""	MySQL所在主机IP
mysql_port	是	String	""	MySQL端口
mysql_user	是	String	""	MySQL连接用户
mysql_password	是	String	""	MySQL连接密码
mysql_schema	是	String	""	MySQL指定库
mysql_table	是	String	""	MySQL指定表
mysql_character_set	否	String	"utf8mb4"	MySQL字符集
is_mysql_password_encrypted	否	String	false	MySQL密码是否经过加密。设置is_mysql_password_encrypted=true时，认为所填写的 `mysql_password` 经过两次加密（先是rsa/none/pkcs1 padding加密，再做base64编码），要求将rsa密钥以pem格式保存在文件中，再将文件绝对路径配置到nomad.hcl的 `rsa_private_key_path` 参数（配置方法见"节点配置"）；设置is_mysql_password_encrypted=false时， `mysql_password` 填写明文密码。

响应参数说明

参数名	类型	说明
columns	String Array	在MySQL实例上执行 `use mysql_schema;select COLUMN_NAME from information_schema.columns where table_name = '?'` 获取到的列数据

样例

response:

```
{
  "columns": [
    "id",
    "c_bit_1"
  ],
  "message": "ok"
}
```

修改日志等级

通过该API可以动态修改日志等级，修改步骤如下： 如要将日志等级变更为 `DEBUG` 1 修改nomad.hcl配置文件中的nomad及dtle日志等级为 `DEBUG` ，如

```
# nomad.hcl
...
log_level = "DEBUG"
...

plugin "dtle" {
  config {
...
    log_level = "DEBUG"
...
  }
}
```

2 调用API，请求参数 `dtle_log_level` 传入 `DEBUG` 3 查看日志，检查内容等级是否变更

API: `POST /v2/log/level`

请求参数说明

参数名	必填?	类型	默认值	说明
dtle_log_level	是	String	""	要修改的日志等级，支持 `TRACE, DEBUG, INFO, WARN, ERROR`

响应参数说明

参数名	类型	说明
dtle_log_level	String	修改后的日志等级

样例

params:

`dtle_log_level` : `DEBUG`

response:

```
{
  "dtle_log_level": "DEBUG",
  "message": "ok"
}
```

用户列表

API: `GET /v2/user/list`

请求参数说明

参数名	必填?	类型	默认值	说明
filter_username	否	String	""	过滤器，根据用户名过滤请求结果
filter_tenant	否	String	""	过滤器，根据租户名过滤请求结果

响应参数说明

参数名	类型	说明
user_list.username	String	用户名
user_list.tenant	String	租户名
user_list.role	String	用户所属角色
user_list.password	String	用户密码（*）
user_list.create_time	time.time	创建时间
user_list.remark	String	备注信息

样例

params：

`filter_username` : "" `filter_tenant` : ""

response:

```
{
    "user_list":[
        {
            "username":"admin",
            "tenant":"platform",
            "role":"admin",
            "password":"*",
            "create_time":"2021-09-06T07:02:37Z",
            "remark":""
        }
    ],
    "message":"ok"
}
```


创建新用户

API: `POST /v2/user/create`

请求参数说明

参数名	必填?	类型	默认值	说明
user_name	是	String	""	用户名/
tenant	是	String	""	租户名
role	是	String	""	用户所属角色
remark	否	String	""	备注
pass_word	是	String	""	用户密码

响应参数说明

返回创建结果

样例

params:


```
{
    "pass_word":"RXBljXmt8grSr9i9OT9URgrlOTSZQoG/qRrC9uc3NE6AEFkGuKorh1TCAlfQtOfC/SvykczwAX7aIcN1ajCsv8mvkeeXDmFNxmr9eOrzV0mOy4jhqnzJys7uzoKGvXzzFZ/g4M+Q/uzmZSea24M7KhwBBimFTMP3s9Ahn5dsitk=",
    "role":"admin",
    "tenant":"platform",
    "username":"test",
    "remark":"*"
}
```


response:

```
{
    "message":"ok"
}
```

更新用户信息

API: `POST /v2/user/update`

请求参数说明

参数名	必填?	类型	默认值	说明
user_name	是	String	""	用户名/
tenant	是	String	""	租户名
role	是	String	""	用户所属角色
remark	否	String	""	备注

响应参数说明

返回创建结果

样例

params:

```
{
    "remark":"railen*188*****5229",
    "role":"admin",
    "tenant":"platform",
    "username":"test"
}
```

response:

```
{
    "message":"ok"
}
```

重置用户密码

API: `POST /v2/user/reset_password`

请求参数说明

参数名	必填?	类型	默认值	说明
user_name	是	String	""	用户名/
tenant	是	String	""	租户名
current_user_password	是	String	""	当前用户密码
pass_word	是	String	""	用户密码

响应参数说明

返回创建结果

样例

params:

```
{
    "current_user_password":"eUa9jNysWv6a0lpzkDoH9tS5T2zoetoG6qwkbtPo99j6dLd1NdWMWeH15qwBJl9VSrMceGZhTsw9NxwgT2XC8MVITSA2VDwqyEwACmkVKu2PcVPJVkT9G1uhGeH7GBgkHJsIZ50b3K16DSpCfYOfbPsUXcCAKAlLmG8BmITM3z4=",
    "password":"xF3ieRGiVj8NQV7+bj5Cclwd+lBA3peUpK3xVlswrUn5qpHp4OBmq6dZq4VluMGAyXHeWCMy9Ffcehb0gf6ISCP4axDP7FdTf+fiMOIlCkVM0O8DJOQKN9QMtca9uD3xv3uacbrEft0N5+TDnIGRhzo1gXttd169xUIR7mNWFEQ=",
    "tenant":"platform",
    "username":"test"
}
```

response:

```
{
    "message":"ok"
}
```

删除用户

API: `POST /v2/user/delete`

请求参数说明

参数名	必填?	类型	默认值	说明
user_name	是	String	""	用户名/
tenant	是	String	""	租户名

响应参数说明

返回创建结果

样例

params:

```
   "tenant":"platform",
   "username":"test"
```

response:

```
{
    "message":"ok"
}
```

租户列表

API: `GET /v2/tenant/list`

请求参数说明

无

响应参数说明

参数名	类型	说明
user_list.username	String	用户名
user_list.tenant	String	租户名
user_list.role	String	用户所属角色
user_list.password	String	用户密码（*）
user_list.create_time	time.time	创建时间
user_list.remark	String	备注信息

样例

response:

```
{
    "tenant_list":[
        "platform"
    ],
    "message":"ok"
}
```

获取当前用户信息

API: `GET /v2/user/current_user`

请求参数说明

无

样例

response:

```
{
    "current_user":{
        "username":"admin",
        "tenant":"platform",
        "role":"admin",
        "password":"*",
        "create_time":"2021-09-06T07:02:37Z",
        "remark":""
    },
    "message":"ok"
}
```

列出当前用户动作权限

API: `GET /v2/user/list_action`

请求参数说明

没有请求参数

响应参数说明

参数名	类型	说明
authority	JSON Array	当前用户的所有菜单权限
menu_item.name	String	菜单名
menu_item.text_cn	String	菜单中文翻译
menu_item.text_en	String	菜单英文翻译
menu_item.menu_level	String	菜单层级
menu_item.menu_url	String	菜单路由
menu_item.id	String	菜单id
menu_item.parent_id	String	菜单的父级id
menu_item.operations	JSON Array	菜单下的所有按钮权限
menu_item.admin_only	String	是否仅管理员可见

样例

response:

```
{
```

```
    "authority":[
        {
            "name":"service",
            "text_cn":"服务",
            "text_en":"service",
            "menu_level":1,
            "menu_url":"",
            "id":1,
            "parent_id":0,
            "operations":[

            ],
            "admin_only":false
        },
        {
            "name":"migration",
            "text_cn":"数据迁移",
            "text_en":"migration",
            "menu_level":2,
            "menu_url":"/migration",
            "id":2,
            "parent_id":1,
            "operations":[
                {
                    "action":"migration-list",
                    "uri":"/v2/jobs/migration",
                    "text_cn":"迁移任务列表",
                    "text_en":"migration job list"
                },
                {
                    "action":"migration-create",
                    "uri":"/v2/job/migration/create",
                    "text_cn":"迁移创建任务",
                    "text_en":"create migration job"
                },
                {
                    "action":"migration-pause",
                    "uri":"/v2/job/migration/pause",
                    "text_cn":"暂停任务",
                    "text_en":"pause migration job"
                },
                {
                    "action":"migration-resume",
                    "uri":"/v2/job/migration/resume",
                    "text_cn":"重启迁移任务",
                    "text_en":"resume migration job"
                },
                {
                    "action":"migration-delete",
                    "uri":"/v2/job/migration/delete",
                    "text_cn":"销毁迁移任务",
                    "text_en":"delete migration job"
                },
```

```
            {
                "action":"migration-reverse",
                "uri":"/v2/job/migration/reverse",
                "text_cn":"创建迁移反向复制任务",
                "text_en":"reverse migration job"
            },
            {
                "action":"migration-reverse_start",
                "uri":"/v2/job/migration/reverse_start",
                "text_cn":"启动迁移反向任务",
                "text_en":"start migration reverse job "
            },
            {
                "action":"migration-update",
                "uri":"/v2/job/migration/update",
                "text_cn":"修改迁移任务",
                "text_en":"update migration job"
            },
            {
                "action":"migration-detail",
                "uri":"/v2/job/migration/detail",
                "text_cn":"查看迁移任务详情",
                "text_en":"migration detail job"
            }
        ],
        "admin_only":false
    },
    {
        "name":"sync",
        "text_cn":"数据同步",
        "text_en":"sync",
        "menu_level":2,
        "menu_url":"/sync",
        "id":3,
        "parent_id":1,
        "operations":[
            {
                "action":"sync-list",
                "uri":"/v2/jobs/sync",
                "text_cn":"同步任务列表",
                "text_en":"sync job list"
            },
            {
                "action":"sync-create",
                "uri":"/v2/job/sync/create",
                "text_cn":"创建同步任务",
                "text_en":"create sync job"
            },
            {
                "action":"sync-pause",
                "uri":"/v2/job/sync/pause",
                "text_cn":"暂停同步任务",
                "text_en":"pause sync job"
```

```
            },
            {
                "action":"sync-resume",
                "uri":"/v2/job/sync/resume",
                "text_cn":"重启同步任务",
                "text_en":"resume sync job"
            },
            {
                "action":"sync-delete",
                "uri":"/v2/job/sync/delete",
                "text_cn":"销毁同步任务",
                "text_en":"delete sync job"
            },
            {
                "action":"sync-reverse",
                "uri":"/v2/job/sync/reverse",
                "text_cn":"反向复制同步任务",
                "text_en":"reverse sync job"
            },
            {
                "action":"sync-reverse_start",
                "uri":"/v2/job/sync/reverse_start",
                "text_cn":"启动同步反向任务",
                "text_en":"reverse start sync job"
            },
            {
                "action":"sync-update",
                "uri":"/v2/job/sync/update",
                "text_cn":"修改同步任务",
                "text_en":"update sync job"
            },
            {
                "action":"sync-detail",
                "uri":"/v2/job/sync/detail",
                "text_cn":"查看同步任务详情",
                "text_en":"sync job detail "
            }
        ],
        "admin_only":false
    },
    {
        "name":"subscription",
        "text_cn":"数据订阅",
        "text_en":"subscription",
        "menu_level":2,
        "menu_url":"/subscribe",
        "id":4,
        "parent_id":1,
        "operations":[
            {
                "action":"subscription-list",
                "uri":"/v2/jobs/subscription",
                "text_cn":"订阅任务列表",
```

```
                    "text_en":"subscription job list"
                },
                {
                    "action":"subscription-create",
                    "uri":"/v2/job/subscription/create",
                    "text_cn":"创建订阅任务",
                    "text_en":"create subscription job"
                },
                {
                    "action":"subscription-pause",
                    "uri":"/v2/job/subscription/pause",
                    "text_cn":"暂停订阅任务",
                    "text_en":"pause subscription job"
                },
                {
                    "action":"subscription-resume",
                    "uri":"/v2/job/subscription/resume",
                    "text_cn":"重启订阅任务",
                    "text_en":"resume subscription job"
                },
                {
                    "action":"subscription-delete",
                    "uri":"/v2/job/subscription/delete",
                    "text_cn":"销毁订阅任务",
                    "text_en":"delete subscription job"
                },
                {
                    "action":"subscription-update",
                    "uri":"/v2/job/subscription/update",
                    "text_cn":"修改订阅任务",
                    "text_en":"update subscription job"
                },
                {
                    "action":"subscription-detail",
                    "uri":"/v2/job/subscription/detail",
                    "text_cn":"查看订阅任务详情",
                    "text_en":"subscription job detail"
                }
            ],
            "admin_only":false
        },
        {
            "name":"platform",
            "text_cn":"平台管理",
            "text_en":"platform",
            "menu_level":1,
            "menu_url":"",
            "id":5,
            "parent_id":0,
            "operations":[

            ],
            "admin_only":false
```

```
        },
        {
            "name":"node",
            "text_cn":"DTLE节点",
            "text_en":"dtle nodes",
            "menu_level":2,
            "menu_url":"/node",
            "id":6,
            "parent_id":5,
            "operations":[
                {
                    "action":"node-list",
                    "uri":"/v2/nodes",
                    "text_cn":"获取节点列表",
                    "text_en":"get node list"
                }
            ],
            "admin_only":false
        },
        {
            "name":"users",
            "text_cn":"用户管理",
            "text_en":"user manage",
            "menu_level":2,
            "menu_url":"/users",
            "id":7,
            "parent_id":5,
            "operations":[
                {
                    "action":"user-list",
                    "uri":"/v2/user/list",
                    "text_cn":"查看用户列表",
                    "text_en":"user list"
                },
                {
                    "action":"user-list_tenant",
                    "uri":"/v2/tenant/list",
                    "text_cn":"获取租户列表",
                    "text_en":"get tenants"
                },
                {
                    "action":"user-create",
                    "uri":"/v2/user/create",
                    "text_cn":"创建用户",
                    "text_en":"create user"
                },
                {
                    "action":"user-delete",
                    "uri":"/v2/user/delete",
                    "text_cn":"删除用户",
                    "text_en":"delete"
                },
                {
```

```
                    "action":"user-update",
                    "uri":"/v2/user/update",
                    "text_cn":"修改用户",
                    "text_en":"update user"
                }
            ],
            "admin_only":true
        },
        {
            "name":"auth",
            "text_cn":"权限配置",
            "text_en":"rights profile",
            "menu_level":2,
            "menu_url":"/auth",
            "id":8,
            "parent_id":5,
            "operations":[
                {
                    "action":"auth-list",
                    "uri":"/v2/role/list",
                    "text_cn":"查看角色列表",
                    "text_en":"role list"
                },
                {
                    "action":"auth-create",
                    "uri":"/v2/role/create",
                    "text_cn":"创建角色",
                    "text_en":"create"
                },
                {
                    "action":"auth-delete",
                    "uri":"/v2/role/delete",
                    "text_cn":"删除角色",
                    "text_en":"delete"
                },
                {
                    "action":"auth-update",
                    "uri":"/v2/role/update",
                    "text_cn":"修改角色",
                    "text_en":"update"
                }
            ],
            "admin_only":true
        }
    ],
    "message":"ok"
}
```


角色列表

API: `GET /v2/role/list`

请求参数说明

参数名	必填?	类型	默认值	说明
filter_tenant	否	String	""	过滤器，根据租户名过滤请求结果

响应参数说明

参数名	类型	说明
role_list.name	String	角色名
role_list.tenant	String	租户名
role_list.authority	String	租户拥有的菜单/按钮权限
role_list.object_users	String	当前角色可操作对象(用户和用户创建的job)
role_list.object_type	String	所有对象都可操作(all),仅可操作自身(self),指定操作对象（designate）

样例

params:

`filter_tenant`:""

response:

```
{
    "role_list":[
        {
            "tenant":"platform",
            "name":"admin",
            "object_users":null,
            "object_type":"all",
            "authority":"[{\"name\":\"service\",\"text_cn\":\"服务\",\"text_en\":\"service
\",\"menu_level\":1,\"menu_url\":\"\",\"id\":1,\"parent_id\":0,\"operations\":[]},{\"name
\":\"migration\",\"text_cn\":\"数据迁移\",\"text_en\":\"migration\",\"menu_level\":2,\"menu
_url\":\"/migration\",\"id\":2,\"parent_id\":1,\"operations\":[{\"action\":\"migration-li
st\",\"uri\":\"/v2/jobs/migration\",\"text_cn\":\"迁移任务列表\",\"text_en\":\"migration jo
b list\"},{\"action\":\"migration-create\",\"uri\":\"/v2/job/migration/create\",\"text_cn
\":\"迁移创建任务\",\"text_en\":\"create migration job\"},{\"action\":\"migration-pause\",\
"uri\":\"/v2/job/migration/pause\",\"text_cn\":\"暂停任务\",\"text_en\":\"pause migration j
ob\"},{\"action\":\"migration-resume\",\"uri\":\"/v2/job/migration/resume\",\"text_cn\":\
"重启迁移任务\",\"text_en\":\"resume migration job\"},{\"action\":\"migration-delete\",\"ur
i\":\"/v2/job/migration/delete\",\"text_cn\":\"销毁迁移任务\",\"text_en\":\"delete migratio
n job\"},{\"action\":\"migration-reverse\",\"uri\":\"/v2/job/migration/reverse\",\"text_c
n\":\"创建迁移反向复制任务\",\"text_en\":\"reverse migration job\"},{\"action\":\"migration-
reverse_start\",\"uri\":\"/v2/job/migration/reverse_start\",\"text_cn\":\"启动迁移反向任务\"
,\"text_en\":\"start migration reverse job \"},{\"action\":\"migration-update\",\"uri\":\
"/v2/job/migration/update\",\"text_cn\":\"修改迁移任务\",\"text_en\":\"update migration job
\"},{\"action\":\"migration-detail\",\"uri\":\"/v2/job/migration/detail\",\"text_cn\":\"查
看迁移任务详情\",\"text_en\":\"migration detail job\"}]},{\"name\":\"sync\",\"text_cn\":\"数
据同步\",\"text_en\":\"sync\",\"menu_level\":2,\"menu_url\":\"/sync\",\"id\":3,\"parent_id
\":1,\"operations\":[{\"action\":\"sync-list\",\"uri\":\"/v2/jobs/sync\",\"text_cn\":\"同
```

步任务列表\",\"text_en\":\"sync job list\"},{\"action\":\"sync-create\",\"uri\":\"/v2/job/sync/create\",\"text_cn\":\"创建同步任务\",\"text_en\":\"create sync job\"},{\"action\":\"sync-pause\",\"uri\":\"/v2/job/sync/pause\",\"text_cn\":\"暂停同步任务\",\"text_en\":\"pause sync job\"},{\"action\":\"sync-resume\",\"uri\":\"/v2/job/sync/resume\",\"text_cn\":\"重启同步任务\",\"text_en\":\"resume sync job\"},{\"action\":\"sync-delete\",\"uri\":\"/v2/job/sync/delete\",\"text_cn\":\"销毁同步任务\",\"text_en\":\"delete sync job\"},{\"action\":\"sync-reverse\",\"uri\":\"/v2/job/sync/reverse\",\"text_cn\":\"反向复制同步任务\",\"text_en\":\"reverse sync job\"},{\"action\":\"sync-reverse_start\",\"uri\":\"/v2/job/sync/reverse_start\",\"text_cn\":\"启动同步反向任务\",\"text_en\":\"reverse start sync job\"},{\"action\":\"sync-update\",\"uri\":\"/v2/job/sync/update\",\"text_cn\":\"修改同步任务\",\"text_en\":\"update sync job\"},{\"action\":\"sync-detail\",\"uri\":\"/v2/job/sync/detail\",\"text_cn\":\"查看同步任务详情\",\"text_en\":\"sync job detail \"}]},{\"name\":\"subscription\",\"text_cn\":\"数据订阅\",\"text_en\":\"subscription\",\"menu_level\":2,\"menu_url\":\"/subscribe\",\"id\":4,\"parent_id\":1,\"operations\":[{\"action\":\"subscription-list\",\"uri\":\"/v2/jobs/subscription\",\"text_cn\":\"订阅任务列表\",\"text_en\":\"subscription job list\"},{\"action\":\"subscription-create\",\"uri\":\"/v2/job/subscription/create\",\"text_cn\":\"创建订阅任务\",\"text_en\":\"create subscription job\"},{\"action\":\"subscription-pause\",\"uri\":\"/v2/job/subscription/pause\",\"text_cn\":\"暂停订阅任务\",\"text_en\":\"pause subscription job\"},{\"action\":\"subscription-resume\",\"uri\":\"/v2/job/subscription/resume\",\"text_cn\":\"重启订阅任务\",\"text_en\":\"resume subscription job\"},{\"action\":\"subscription-delete\",\"uri\":\"/v2/job/subscription/delete\",\"text_cn\":\"销毁订阅任务\",\"text_en\":\"delete subscription job\"},{\"action\":\"subscription-update\",\"uri\":\"/v2/job/subscription/update\",\"text_cn\":\"修改订阅任务\",\"text_en\":\"update subscription job\"},{\"action\":\"subscription-detail\",\"uri\":\"/v2/job/subscription/detail\",\"text_cn\":\"查看订阅任务详情\",\"text_en\":\"subscription job detail\"}]},{\"name\":\"platform\",\"text_cn\":\"平台管理\",\"text_en\":\"platform\",\"menu_level\":1,\"menu_url\":\"\",\"id\":5,\"parent_id\":0,\"operations\":[]},{\"name\":\"node\",\"text_cn\":\"DTLE节点\",\"text_en\":\"dtle nodes\",\"menu_level\":2,\"menu_url\":\"/node\",\"id\":6,\"parent_id\":5,\"operations\":[{\"action\":\"node-list\",\"uri\":\"/v2/nodes\",\"text_cn\":\"获取节点列表\",\"text_en\":\"get node list\"}]},{\"name\":\"users\",\"admin_only\":true,\"text_cn\":\"用户管理\",\"text_en\":\"user manage\",\"menu_level\":2,\"menu_url\":\"/users\",\"id\":7,\"parent_id\":5,\"operations\":[{\"action\":\"user-list\",\"uri\":\"/v2/user/list\",\"text_cn\":\"查看用户列表\",\"text_en\":\"user list\"},{\"action\":\"user-list_tenant\",\"uri\":\"/v2/tenant/list\",\"text_cn\":\"获取租户列表\",\"text_en\":\"get tenants\"},{\"action\":\"user-create\",\"uri\":\"/v2/user/create\",\"text_cn\":\"创建用户\",\"text_en\":\"create user\"},{\"action\":\"user-delete\",\"uri\":\"/v2/user/delete\",\"text_cn\":\"删除用户\",\"text_en\":\"delete\"},{\"action\":\"user-update\",\"uri\":\"/v2/user/update\",\"text_cn\":\"修改用户\",\"text_en\":\"update user\"}]},{\"name\":\"auth\",\"admin_only\":true,\"text_cn\":\"权限配置\",\"text_en\":\"rights profile\",\"menu_level\":2,\"menu_url\":\"/auth\",\"id\":8,\"parent_id\":5,\"operations\":[{\"action\":\"auth-list\",\"uri\":\"/v2/role/list\",\"text_cn\":\"查看角色列表\",\"text_en\":\"role list\"},{\"action\":\"auth-create\",\"uri\":\"/v2/role/create\",\"text_cn\":\"创建角色\",\"text_en\":\"create\"},{\"action\":\"auth-delete\",\"uri\":\"/v2/role/delete\",\"text_cn\":\"删除角色\",\"text_en\":\"delete\"},{\"action\":\"auth-update\",\"uri\":\"/v2/role/update\",\"text_cn\":\"修改角色\",\"text_en\":\"update\"}]}]"
        }
    ],
    "message":"ok"
}

创建新角色

API: `POST /v2/user/create`

请求参数说明

参数名	必填?	类型	默认值	说明
name	是	String	""	角色名/
tenant	是	String	""	租户名
object_users	是	String	""	当前角色可操作对象(用户和用户创建的job)
authority	true	String	""	租户拥有的菜单/按钮权限
object_type	是	String	""	所有对象都可操作(all),仅可操作自身(self),指定操作对象（designate）

响应参数说明

返回创建结果

样例

params:

```
{
    "tenant":"platform",
    "name":"user",
    "operation_object_type":"designate",
    "operation_users":[
        "platform:admin"
    ],
    "authority":"[{\"name\":\"service\",\"text_cn\":\"服务\",\"text_en\":\"service\",\"menu_level\":1,\"menu_url\":\"\",\"id\":1,\"parent_id\":0,\"operations\":[],\"admin_only\":false},{\"name\":\"platform\",\"text_cn\":\"平台管理\",\"text_en\":\"platform\",\"menu_level\":1,\"menu_url\":\"\",\"id\":5,\"parent_id\":0,\"operations\":[],\"admin_only\":false},{\"name\":\"node\",\"text_cn\":\"DTLE节点\",\"text_en\":\"dtle nodes\",\"menu_level\":2,\"menu_url\":\"/node\",\"id\":6,\"parent_id\":5,\"operations\":[{\"action\":\"node-list\",\"uri\":\"/v2/nodes\",\"text_cn\":\"获取节点列表\",\"text_en\":\"get node list\"}],\"admin_only\":false}]"
}
```

response:

```
{
    "message":"ok"
}
```

删除角色

API: `POST /v2/role/delete`

请求参数说明

参数名	必填?	类型	默认值	说明
name	是	String	""	角色名/
tenant	是	String	""	租户名

响应参数说明

返回创建结果

样例

params:

```
   "tenant":"platform",
   "username":"test"
```

response:

```
{
    "message":"ok"
}
```

更新角色信息

API: `POST /v2/user/update`

请求参数说明

参数名	必填?	类型	默认值	说明
name	是	String	""	用户名/
tenant	是	String	""	租户名
operation_object_type	是	String	""	可操作对象类型
operation_users	否	String Array	""	当可操作对象为designate，该字段为指定用户的列表
authority	否	string	""	菜单和按钮权限

响应参数说明

返回创建结果

样例

params:

```
{
    "tenant":"platform",
    "name":"user",
    "operation_object_type":"self",
    "authority":"[{\"name\":\"service\",\"text_cn\":\"服务\",\"text_en\":\"service\",\"menu_level\":1,\"menu_url\":\"\",\"id\":1,\"parent_id\":0,\"operations\":[],\"admin_only\":false},{\"name\":\"platform\",\"text_cn\":\"平台管理\",\"text_en\":\"platform\",\"menu_level\":1,\"menu_url\":\"\",\"id\":5,\"parent_id\":0,\"operations\":[],\"admin_only\":false},{\"name\":\"node\",\"text_cn\":\"DTLE节点\",\"text_en\":\"dtle nodes\",\"menu_level\":2,\"menu_url\":\"/node\",\"id\":6,\"parent_id\":5,\"operations\":[{\"action\":\"node-list\",\"uri\":\"/v2/nodes\",\"text_cn\":\"获取节点列表\",\"text_en\":\"get node list\"}],\"admin_only\":false},{\"name\":\"sync\",\"text_cn\":\"数据同步\",\"text_en\":\"sync\",\"menu_level\":2,\"menu_url\":\"/sync\",\"id\":3,\"parent_id\":1,\"operations\":[{\"action\":\"sync-list\",\"uri\":\"/v2/jobs/sync\",\"text_cn\":\"同步任务列表\",\"text_en\":\"sync job list\"},{\"action\":\"sync-create\",\"uri\":\"/v2/job/sync/create\",\"text_cn\":\"创建同步任务\",\"text_en\":\"create sync job\"},{\"action\":\"sync-pause\",\"uri\":\"/v2/job/sync/pause\",\"text_cn\":\"暂停同步任务\",\"text_en\":\"pause sync job\"},{\"action\":\"sync-resume\",\"uri\":\"/v2/job/sync/resume\",\"text_cn\":\"重启同步任务\",\"text_en\":\"resume sync job\"},{\"action\":\"sync-delete\",\"uri\":\"/v2/job/sync/delete\",\"text_cn\":\"销毁同步任务\",\"text_en\":\"delete sync job\"},{\"action\":\"sync-reverse\",\"uri\":\"/v2/job/sync/reverse\",\"text_cn\":\"反向复制同步任务\",\"text_en\":\"reverse sync job\"},{\"action\":\"sync-reverse_start\",\"uri\":\"/v2/job/sync/reverse_start\",\"text_cn\":\"启动同步反向任务\",\"text_en\":\"reverse start sync job\"},{\"action\":\"sync-update\",\"uri\":\"/v2/job/sync/update\",\"text_cn\":\"修改同步任务\",\"text_en\":\"update sync job\"},{\"action\":\"sync-detail\",\"uri\":\"/v2/job/sync/detail\",\"text_cn\":\"查看同步任务详情\",\"text_en\":\"sync job detail \"}],\"admin_only\":false}]"
}
```


response:


```
{
    "message":"ok"
}
```

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

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

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/运行状态信息：执行 `kill -SIGTTIN {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
- 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

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:

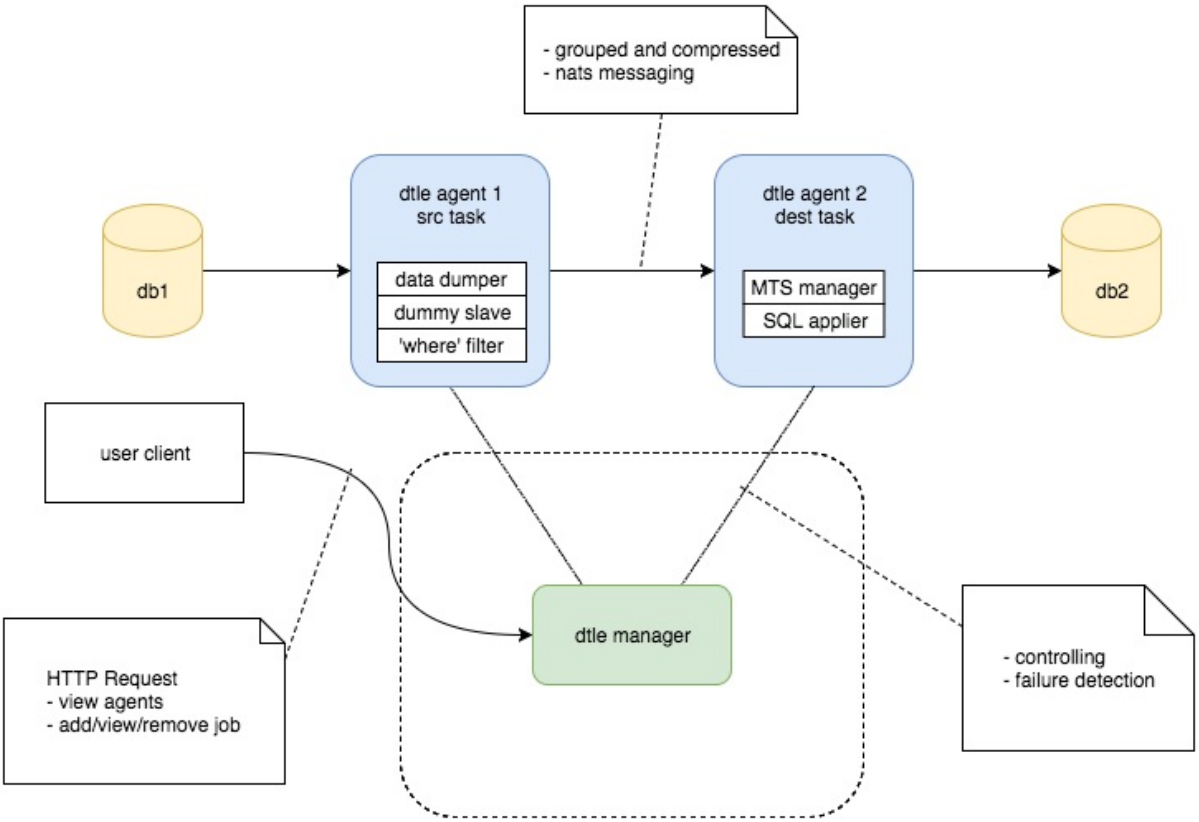
- RowSize为数据行的平均大小 (字节)
- ChunkSize为配置项
- QueueSize为传输队列长度, 硬编码为24
- 内存占用系数 测量约为 常量3.2

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/docs/tutorial/#viewing-the-change-eventsl

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

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
  }
}
```


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

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

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

- 支持更多种类的公有云间的数据迁移
- 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)
- 对于 MySQL分布式中间件 (如dble) 提供数据扩容方案
- 对链路提供限流参数
- 提供告警功能
- 复制到Kafka的数据格式支持Avro
- 列名变换
- 数据变换
- 线路加密
- 免一致性快照事务的全量复制
  - 全量复制也可断点续传
- 支持MGR Primary切换
- 动态增减同步对象
- 支持2G级别的大事务

已完成

- 一致性DDL元数据 (2.19.03.0+ #321)
- 库.表名变换 (2.19.11.0+)
- 列选择、列顺序变换 (3.20.08.0+)