sink-agent

DRC-DA sink-agent 利用の流れ

1. IoT-Platform テストベッド利用契約を締結し，開通通知書及び DRC-DA index-node，sink-agent（drc-da.tar.gz）を受領する
    - 開通通知を受け取った時点で，DRC-DA directory-nodeは配備済みとなる
    - 下記の Setup 手順では，DRC-DA directory-node に接続する形で DRC-DA index-node を配備し，これに付随する生データ管理機能として sink-agent を配備する
    - 下記の Setup 手順は参考提供のdrc-da/modules/ddmgr_setup/pom.xml，drc-da/modules/sink-agent_setup/pom.xmlを使用するものとする
2. DRC-DA index-node 及び sink-agent を配備するホストを用意する．要求スペックは下記の通り
    - x86_64 アーキテクチャ
    - CentOS7
    - メモリ2GB以上
    - インターネット接続（NAT 可）
        - ただし NAT 環境下では，データを取得するアプリ自身も同 NAT 下にいる必要がある
        - 富士通クラウド上の IoT-Platform に http（port: 80），mqtt（port: 1883）でアクセス出来る
3. エージェント配備コマンドを発行するホスト（CentOS 環境設定用ホスト）を用意する
    - エージェントの配備は ansible を用いて行うため，配備対象ホストに対し SSH 接続できるホストを用意する

Setup

環境構築(ローカルのPCから対象のマシンを設定)

1. CentOS 環境設定用ホストにdrc-da.tar.gzをコピー
2. drc-da.tar.gzを展開

```
# tar zxvf drc-da.tar.gz
```

3. drc-da/配下に移動し，centos_setup.tar.gzを展開

```
# tar zxvf centos_setup.tar.gz
```

4. drc-da/centos_setup/setup/centos/inventoryを編集し[target]に対象のマシンのIPアドレスを設定する
5. drc-da/centos_setup/setup/centos/配下に移動し，下記のコマンドを実行

```
# chmod 400 {秘密鍵のパス}/{秘密鍵}　秘密鍵のpermissionを変更
# ansible-playbook -i inventory -s compose-setup.yml -u {SSHユーザ名} --private-key="
{秘密鍵のパス}/{秘密鍵}"
もしくは
# ansible-playbook -i inventory -s compose-setup.yml -u {SSHユーザ名} -k　実行後にSSH
のパスワード入力
```

配備

1. index-node , sink-agentを配備対象ホストにdrc-da.tar.gzをコピー
2. index-node , sink-agentを配備対象ホストにログイン
3. root権限に変更

```
# sudo -s
もしくは
# su 実行後にパスワード入力
```

4. drc-da.tar.gzを展開

```
# tar zxvf drc-da.tar.gz
```

5. drc-da/配下に移動し，modules.tar.gzを展開

```
# tar zxvf modules.tar.gz
```

6. modules/配下に移動し，start.sh，stop.shに実行権限追加

```
# chmod +x start.sh stop.sh
```

7. config.propertiesの設定内容を変更する
   以下は設定例．本configファイル以外にIoT Platformへのリソース，アクセスコードの設定が必要な点に留意（設定方法については別紙を参照のこと）.

```
LOG_LEVEL=DEBUG
OWN_ID=idx-XXX
OWN_DOMAIN=n/a
AS_DIRECTORY=false
AS_INDEX=true
IOTPF_HTTP_HOST=api.sys3.iot.jp.fujitsu.com
IOTPF_MQTT_HOST=sys3.iot.jp.fujitsu.com
IOTPF_MQTT_PORT=1883
IOTPF_USER=VTBXXX-001
IOTPF_PASSWORD=XXXXXXXX
IOTPF_RESOURCEROOT=v1/VTBXXX-001/testbeddir
IOTPF_BIN_RESOURCEROOT=v1/VTBXXX-001/_bin/testbeddir
IOTPF_TOKEN=XXXXXXXX
RABBITMQ_HOST=rabbit
RABBITMQ_PORT=5672
MONGODB_HOST=mongo
MONGODB_PORT=27017
MEMCACHED_HOST=sublist
MEMCACHED_PORT=11211
SD_MODE=default
DATA_CLEAR_PERIOD=172800000
SUBSCRIPTION_TTL=172800
DIR_NODE_ID=dir
```

実行環境に併せて，以下のパラメータを事前に変更する．

- OWN_ID -> index-node を一意に識別するためのID．idx-XXX（XXXは数値）の形式で記載．また，OWN_IDは重複しないように気をつける．
- IOTPF_USER -> 開通通知に記載されているテナントIDを記載．
- IOTPF_PASSWORD -> IoTPlatformにログイン後閲覧できるMQTTパスワードを記載．
  MQTTパスワードはIoTPlatformにログイン後「共通設定」->「Password」から確認できる．
    - **IoTPlatform Passwordは変更可能だが，変更した場合には富士通が運用する directory-node の設定も合わせて変更する必要がある．**
- IOTPF_RESOURCEROOT -> 開通通知書記載の値を記載．（v1/<テナント名>/<ディレクトリ>）
- IOTPF_BIN_RESOURCEROOT -> 開通通知書記載の値を記載．（v1/<テナント名>/_bin/<ディレクトリ>）
- IOTPF_TOKEN -> 開通通知書記載の値を記載．（＜リソースに設定されているアクセスコード＞）
- DIR_NODE_ID -> 開通通知書記載の値を記載．

8. index-node , sink-agentを配備する

```
# ./start.sh
```

削除

1. drc-da/modules/ 配下に移動し，下記のコマンドを実行

```
# ./stop.sh
```

備考

- drc-da(index-node)のportは下記の通り
  (※portを変更する際はdrc-da/modules/ddmgr_setup/setup/docker_compose/ddmgr_index/docker-compose.ymlを修正すること)
    - distributeddatamgr : 34001
    - RabbitMQ : 30011
    - MongoDB : 30001
    - memcached : 35001
- sink-agentのportは下記の通り
  (※portを変更する際はdrc-da/modules/sink-agent_setup/setup/docker_compose/sink-agent/docker-compose.ymlを修正すること)
    - sink-agent : 30013
    - MongoDB : 30002
    - memcached : 35002

API一覧

- インデックス登録
    - リクエストHeader
      Content-type: application/json
    - レスポンスHeader
      Content-type: application/json
    - リクエスト送信先
      POST http://xxx.xxx.xxx.xxx:30013/sink_agent/api/put_idx
    - リクエストボディ
      {"meta":{登録するインデックス(JSON)}, "sink_info":{"data_id":<daat_id>}
    - 応答
      {"result":true, "idx_id":UUID}
- インデックス検索取得
    - リクエストHeader
      Content-type: application/json
    - レスポンスHeader
      Content-type: application/json
    - リクエスト送信先
      POST http://xxx.xxx.xxx.xxx:30013/sink_agent/api/search_idx
    - リクエストボディ
      インデックス検索key(JSON)
    - 応答
      {"result":true, "idx":[...]}
- 生データ登録
    - リクエストHeader
      Content-type: application/octet-stream
    - レスポンスHeader
      Content-type: application/json
    - リクエスト送信先

POST http://xxx.xxx.xxx.xxx:30013/sink_agent/api/rawdata
    - リクエストボディ
      登録する生データ(binary)
    - 応答
      {"result":true, "data_id":UUID}

- 生データ取得

    - リクエストHeader
      Content-type: application/json
    - レスポンスHeader
      Content-type: application/octet-stream
    - リクエスト送信先
      GET http://xxx.xxx.xxx.xxx:30013/sink_agent/api/rawdata/{data_id}
    - 応答
      生データ(binary)

- 生データ削除

    - リクエストHeader
      Content-type: application/json
    - レスポンスHeader
      Content-type: application/json
    - リクエスト送信先
      DELETE http://xxx.xxx.xxx.xxx:30013/sink_agent/api/rawdata/{data_id}
    - 応答
      {"result":true}

Getting Started

前提

上記手順に従いセットアップが終了していることとする.
なお，ディレクトリ検索に関しては [drc-da README (https://github.labs.fujitsu.com/drc-da/distributeddatamgr)](https://github.labs.fujitsu.com/drc-da/distributeddatamgr) (API 及び Getting Started) を参照のこと.

準備

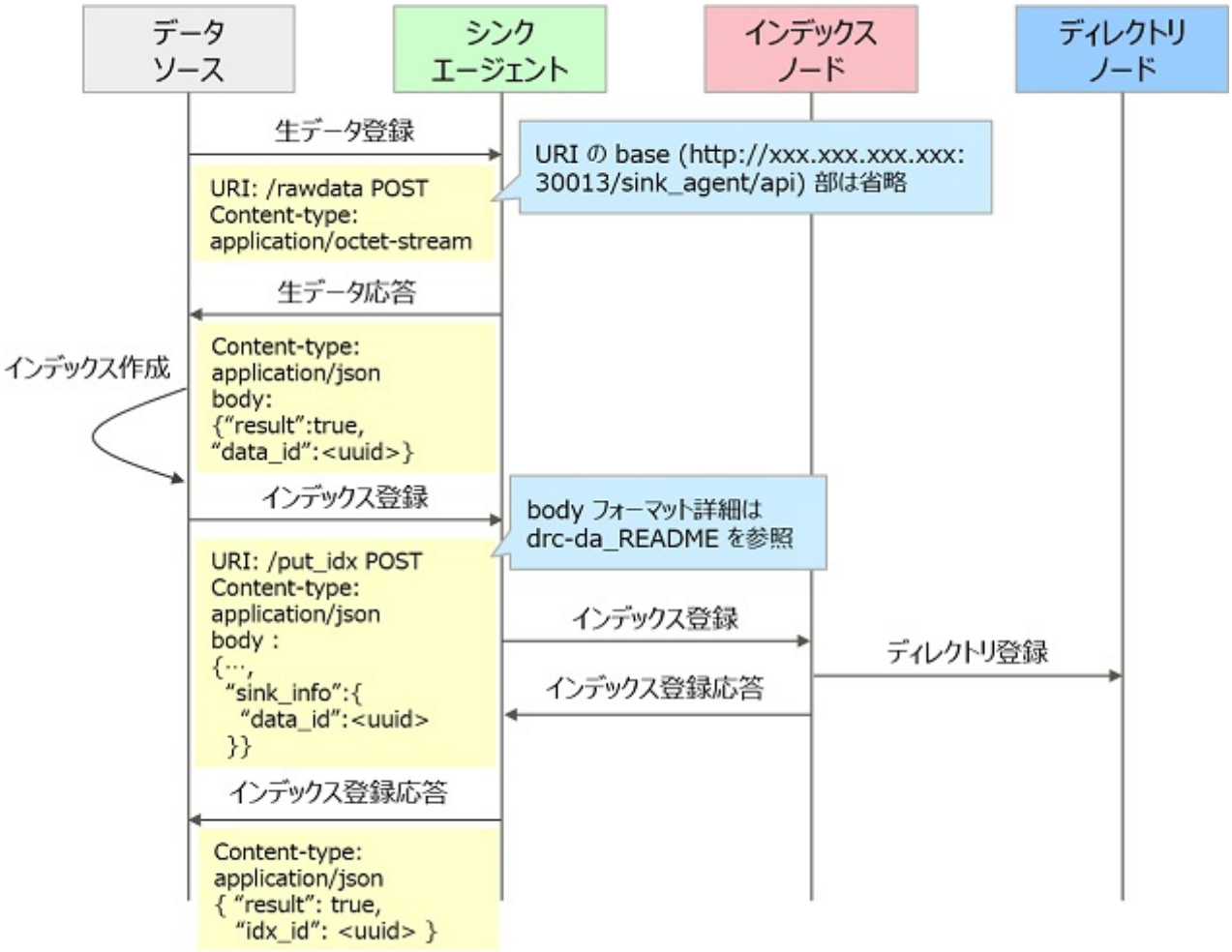
1. drc-da/scripts内の ruby スクリプトを展開する

```
# tar zxvf drc-da.tar.gz
# cd drc-da
# tar zxvf scripts.tar.gz
```

2. ruby (v2.3以降) 実行環境を構築する
3. ruby スクリプトに必要な gem をインストールする

```
# gem install httpclient
# gem install json
```

インデックス，生データ登録 (サンプル: rawdata.rb, put_idx.rb)



1. 必要ライブラリのインポート(全てのスクリプトに記述する)

```
require 'httpclient'
require 'json'
```

2. 生データの登録のIPアドレスを設定(rawdata.rb : 6行目)

```
base_uri = 'http://xxx.xxx.xxx.xxx:30013'
```

3. 生データを登録する(rawdata.rb : 10, 12行目)
   下記の例では適当なファイルを生データとして登録している

```
body = open('{path_to_file}/{file_name}','r')
res = client.post(uri, body, header)
```

4. 生データ登録時のレスポンス(data_id)を取得する(rawdata.rb : 13, 14行目)

```
puts "code=#{res.code}"
puts res.body
```

5. インデックス登録リクエスト送信先のIPアドレスを設定する(put_idx.rb : 6行目)

```
base_uri = 'http://xxx.xxx.xxx.xxx:30013'
```

6. 手順4で取得したdata_idを設定する(put_idx.rb : 11行目)

```
data_id = "<4. で取得した data_id>"
```

7. 登録するインデックスを作成，sink_info内にdata_idを必ず含めること(put_idx.rb : 13〜17行目)

```
payload = Hash.new
payload[:sink_info] = {:data_id => data_id}
payload[:dir_keys] = ["location","device_id"]
payload[:meta] = {:location => "loc-A", :device_id => "dev-01", :temp => 25.5, :humid
=> 60}
payload[:gen_time] = Time.now.to_i
```

8. インデックス登録リクエストを送信(put_idx.rb : 20行目)

```
res = client.post(uri, payload.to_json, header)
```

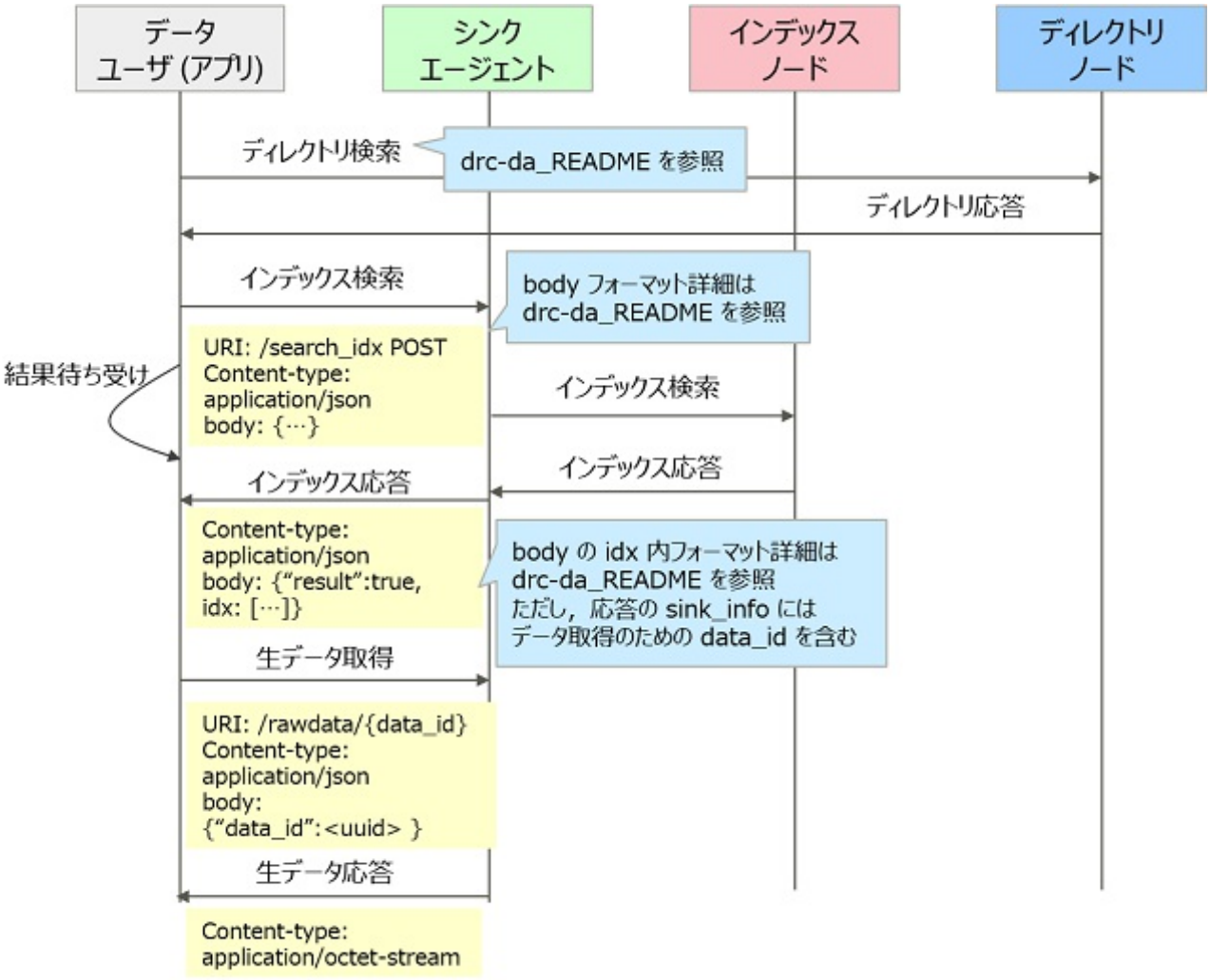
9. インデックス登録時のレスポンス(idx_id)を取得(put_idx.rb : 21, 22行目)

```
puts "code=#{res.code}"
puts res.body
```

スクリプトは下記のようなコマンドで実行可能

```
ruby rawdata.rb
ruby put_idx.rb
```

インデックス，生データ検索取得（サンプル: search_idx.rb, search_rawdata.rb)



1. 必要ライブラリのインポート(全てのスクリプトに記述する)

```
require 'httpclient'
require 'json'
```

2. インデックス検索リクエスト送信先のIPアドレスを設定(search_idx.rb : 6行目)

```
base_uri = 'http://xxx.xxx.xxx.xxx:30013'
```

3. インデックスの検索keyを作成，search_idx.rbではidx_idで検索している(search_idx.rb : 9行目)

```
payload = {'idx_id' => 'UUID'}
```

4. インデックス検索リクエスト送信(search_idx.rb : 13行目)

```
res = client.post(uri, payload.to_json, header)
```

5. インデックスの検索結果を取得

```
puts "code=#{res.code}"
puts res.body
```

6. 生データ検索リクエスト送信先のIPアドレスを設定(search_rawdata.rb : 6行目)

```
base_uri = 'http://xxx.xxx.xxx.xxx:30013'
```

7. 取得する生データのdata_idを指定する(search_rawdata.rb : 8行目)

```
data_id = "<5. で取得した data_id>"
```

8. 生データ検索リクエスト送信(search_rawdata.rb : 12行目)

```
res = client.get(uri, header)
```

9. 生データ検索結果を取得

```
puts "code=#{res.code}"
puts res.body
```

スクリプトは下記のようなコマンドで実行可能

```
ruby search_idx.rb
ruby search_rawdata.rb
```

【参考】利用手順(docker-compose)

以下は配備スクリプトを使用しない場合の手順です．(※通常使用することはありません．)
また，下記記述における URL はインターネットからアクセス出来ないため，drc-da.tar.gz 内にある同名のファイルに置き換えて実行する必要があります．

前提

- 事前にansibleをインストールしておくこと

環境構築(ローカルのPCから対象のマシンを設定)

1. sink-agent/setup/k5 をローカルのPCにコピーする(URL : https://github.labs.fujitsu.com/drc-da/sink-agent)
2. sink-agent/setup/k5/ 配下のinventryのIPアドレスを変更する(targetのマシンのIPアドレスにする)
3. sink-agent/setup/k5/ 配下で下記のコマンドを実行する

```
# ansible-playbook -i sink-agent/setup/k5/inventory -s sink-agent/setup/k5/centos-setup.yml
# ansible-playbook -i sink-agent/setup/k5/inventory -s sink-agent/setup/k5/compose-setup.yml
```

配備

1. sink-agent/setup/docker_files, sink-agent/setup/docker-compose, distributeddatamgr 配下のファイルを index-node, sink-agent を配備するマシン上にコピー
    - distributeddatamgr
        - https://github.labs.fujitsu.com/drc-da/distributeddatamgr
2. Dockerイメージの作成
    - sink-agent
        - sink-agent/配下で下記のコマンドを実行しライブラリを取得

```
# mvn dependency:copy-dependencies -
DoutputDirectory=./setup/docker_files/sink-agent/lib
```

- sink-agent/setup/docker_files/sink-agent配下にmeta-inf.tar.gz, web-inf.tar.gzをコピーする
  - meta-inf.tar.gz, web-inf.tar.gz：
    http://distpf3.png.flab.fujitsu.co.jp:8080/view/SDI/job/sink-agent/ws/
- sink-agent/setup/docker_files/sink-agent内で下記のコマンドを実行しdockerイメージを作成

```
# tar zxvf meta-inf.tar.gz
# tar zxvf web-inf.tar.gz
# cp -r lib/ WEB-INF/
# sudo docker build -t sink-agent .
```

- distributeddatamgr
  - distributeddatamgr配下で下記のコマンドを実行しライブラリを取得

```
# mvn dependency:copy-dependencies -
DoutputDirectory=./setup/docker_files/ddmgr/lib
```

  - distributeddatamgr/setup/docker_files/ddmgr配下にdistributeddatamgrをビルドしたjarファイルをコピーする
    - jarファイル：
      http://distpf3.png.flab.fujitsu.co.jp:8080/view/SDI/job/distribu
  - distributeddatamgr/setup/docker_files/ddmgr内で下記のコマンド実行dockerイメージを作成

```
# sudo docker build -t distributeddatamgr .
```

- rabbitmq
  - sink-agent/setup/docker_files/rabbitmq内で下記のコマンド実行dockerイメージを作成

```
# sudo docker build -t rabbitmq .
```

3. docker-compose実行
   - distributeddatamgr
     - sink-agent/setup/docker-compose/ddmgr_index/config.propertiesの設定内容を変更
     - sink-agent/setup/docker-compose/ddmgr_index内で下記のコマンド実行

```
# sudo /usr/local/bin/docker-compose up -d
```

   - sink-agent
     - sink-agent/setup/docker-compose/sink-agent内で下記のコマンド実行

```
# sudo /usr/local/bin/docker-compose up -d
```

削除

1. docker-compose実行
    - distributeddatamgr
        - sink-agent/setup/docker-compose/ddmgr_index内で下記のコマンド実行

```
# sudo /usr/local/bin/docker-compose down
```

    - sink-agent
        - sink-agent/setup/docker-compose/sink-agent内で下記のコマンド実行

```
# sudo /usr/local/bin/docker-compose down
```

【参考】利用手順(k8s環境)

以下はディレクトリ，index-node，sink-agentを全てk8s環境で動かすときの手順です(※通常使用することはありません)．
また，下記記述における URL はインターネットからアクセス出来ないため，drc-da.tar.gz 内にある同名のファイルに置き換えて実行する必要があります．

前提

DRC-DA index-node，directory-node を配備済みであること
https://github.labs.fujitsu.com/drc-da/distributeddatamgr

配備

1. sink-agent 配下のファイルをk8sのmasterにコピーする
2. Dockerイメージの作成
    - sink-agent/setup/docker_files/sink-agent配下にビルドしたwarファイル，agent.propertiesをコピーする
    - sink-agent/setup/docker_files/sink-agent内で下記のコマンドを実行

```
# sudo docker build -t sink-agent .
# sudo docker tag sink-agent localhost:5000/sink-agent
# sudo docker push localhost:5000/sink-agent
```

3. Podを配備
    - sink-agent/kube_yamls内で下記のコマンドを実行(namespaceはDRC-DAのnamespaceと合わせる)

```
# kubectl create --namespace=iot-da -f sink-agent.yml
# kubectl create --namespace=iot-da -f sink-agent-srv.yml
```

削除

1. Podの削除

- sink-agent/kube_yamls内で下記のコマンドを実行(namespaceはDRC-DAのnamespaceと合わせる)

```
# kubectl delete pod --namespace=iot-da sink-agent
# kubectl delete svc --namespace=iot-da sink-agent
```

Copyright について