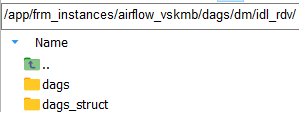
От команды vskmb

**DEV**

1. зайти на стенд через мобу, там располагаются рабочие и управляющие потоки.



Подробнее про потоки можно почитать здесь:

<https://wiki.corp.dev.vtb/pages/viewpage.action?pageId=1121405836> (1)

1. Нужно сделать управляющий (cf) и рабочий (wrk) поток, они представлены в виде DAG. Они лежат в папке ‘dags’
2. Для этих дагов должен быть создан yaml файл, который лежит в папке ‘dags\_struct’

Пример yaml файла (внимание при копировании, может потеряться нужная табуляция):

workflow:

name: WRK\_DM\_CUSTOMER\_PHONE

depends:

- slv\_wf\_name: WF\_DM\_CUSTOMER\_PHONE

target\_resource\_name: ceh.dm\_t\_diur\_skmb\_tech.dm\_customer\_phone

target\_table\_name: dm\_t\_diur\_skmb\_tech.dm\_customer\_phone

source\_resource\_names:

- ceh.bdm.counterparty

- ceh.bdm.counterparty\_contact

prepare\_db\_func\_name: fn\_qw\_customer\_phone\_stg\_load

load\_db\_func\_name: fn\_qw\_customer\_phone\_dlt\_load

schema\_name: dm\_t\_diur\_skmb\_tech

keys:

parent\_dag\_id: cf\_dm\_customer\_phone

prepare\_db\_func\_name и load\_db\_func\_name создаются в п.5

target\_resource\_name – расположение таргетной таблицы (витрины), создается в п.5

source\_resource\_names – таблицы (представления) из которых тянутся данные

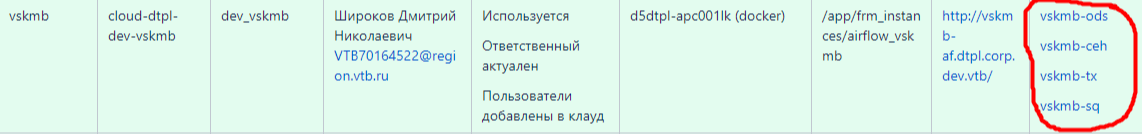
1. Для того что бы работать с какими-то данными (выгружать или загружать) они должны иметь «ресурсы» (мы не очень понимаем как они работают и для чего, но делаем по инструкции):

[https://wiki.corp.dev.vtb/pages/viewpage.action?pageId=1835780337#id-ПодготовкаресурсовфреймворкаIDL-ПодготовкаресурсовфреймворкаIDL-Проверка](https://wiki.corp.dev.vtb/pages/viewpage.action?pageId=1835780337#id-%D0%9F%D0%BE%D0%B4%D0%B3%D0%BE%D1%82%D0%BE%D0%B2%D0%BA%D0%B0%D1%80%D0%B5%D1%81%D1%83%D1%80%D1%81%D0%BE%D0%B2%D1%84%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA%D0%B0IDL-%D0%9F%D0%BE%D0%B4%D0%B3%D0%BE%D1%82%D0%BE%D0%B2%D0%BA%D0%B0%D1%80%D0%B5%D1%81%D) (2)

Это происходит через сервис, который называется uni provider Их 4 штуки, что бы на них попасть нужно перейти по ссылке:

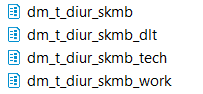
<https://wiki.corp.dev.vtb/display/DCEH/4.1.1+DEV> (3)

И найти такую строчку, связанную с вашей командой:



Перейдя по ссылке vskmb-ceh можно создать и проверить ресурсы. Но для таблиц источников ресурсы должны иметь состояние (про это все написано в гайде выше), которое заводится с помощью vskmb -ceh и vskmb -tx (довольно замороченный шаг, перед выполнением советую внимательно ознакомится с инструкцией).

1. Через DBeaver подключится к БД Greenplum в ней найти свои схемы:



Для одной витрины требуется создать таргетную таблицу и 2 функции, которые описываются по ссылке 1 (fn\_qw\_\*\_stg\_load и fn\_qw\_\*\_dlt\_load):

В схеме \_tech создаем вручную таблицу с нужными полями (например: dm\_customer\_phone)

в функции stg мы вытягиваем нужные представления из bdm в свою схему \_work и собираем витрину в этой же схеме.

В функции dlt создаем журнал дельты (за шаблон(template) берем таргетную таблицу из \_tech) и заполняем его. В журнале появляется дополнительный столбец record\_mode, при заполнении нужно это учесть и присваивать какое-либо значение (например «А»)

1. Зайти на стенд airflow и можно запускать сначала управляющий поток, затем рабочий.

**Preprod**

Есть 2 нужных репозитория: adgp, etl-scale

**Adgp**

[**https://bitbucket.region.vtb.ru/projects/DP/repos/adgp/browse**](https://bitbucket.region.vtb.ru/projects/DP/repos/adgp/browse) **(4)**

Довольно подробно описание дано в ридми, так же очень информативное видео от архитектора :

<https://disk.yandex.ru/d/Qnen3_59bsgolQ> (5)

В adgp хранятся yaml файлы по структуре похожей с базой данных. Эти файлы нужны для воспроизведения БД Greenplum локально или удаленно с помощью технологии Liquibase

0. Скачать все необходимое (перечисленно в ридми)

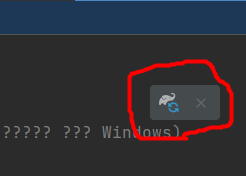
1. Для Начала нужно склонировать от ветки команды рабочую ветку с названием feature/''название команды''- ‘’номер задачи’’

1.1 Если делаете это в первый раз нужно создать и заполнить gradle.properties по инструкции в ридми, затем обновить изменения градла

В параметр runlist пишем

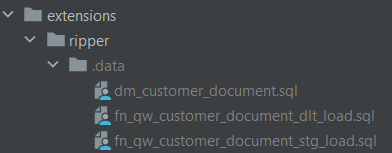
runlist = dm\_schemas, dm\_t\_diur\_skmb (имя вашей схемы)

вашей схемы может там не быть, тогда ее нужно создать в файле build.gradle аналогично другим



Далее начнут качаться библиотеки, в какой-то момент времени загрузка может зациклиться или идти очень долго тогда нужно отключится от токена.

2. в папку extension/ripper/.data закинуть sql скрипты.

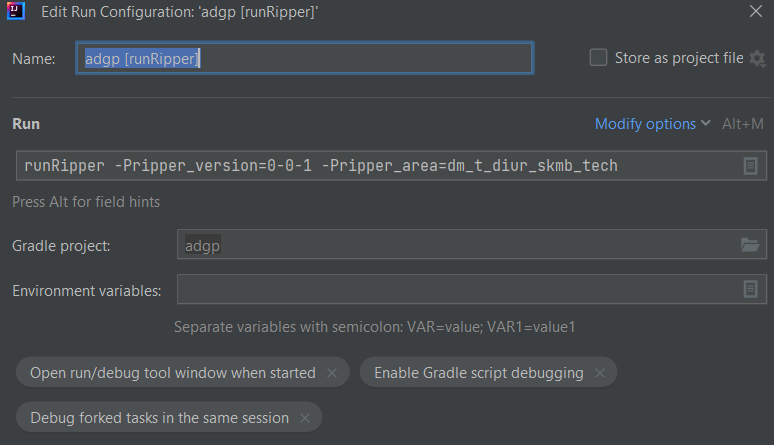
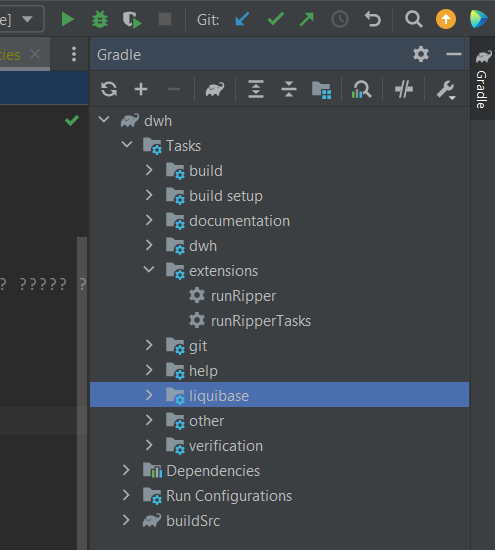


3. Справа найти скрипт runRipper, тыкнуть по нему ПКМ, выбрать modify run configuration и задать нужные переменные и запускаете.

По пути

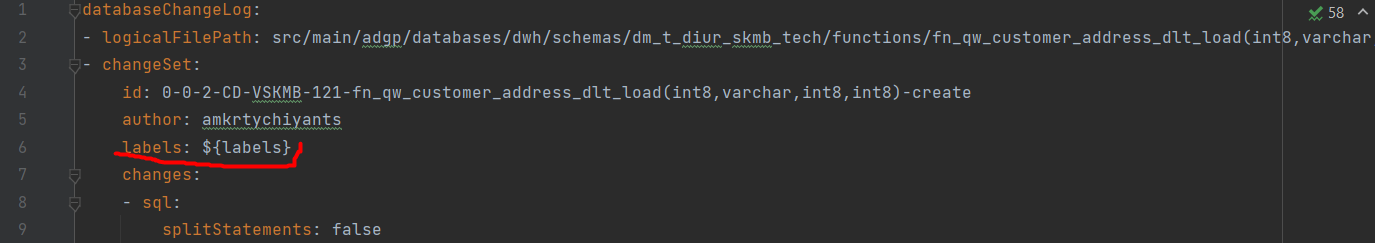
src/main/adgp/databases/dwh/schemas/ «ваша схема»

Создастся папка tables, functions в которых будут распологаться папки с готовыми yaml.



Цифру в версии меняете если вносите какие-то изменения в скрипте, которые уже были отправлены на препрод. Если заливать скрипт с прежней версией и id(он находится внутри полученных файлов-changelog) будет ошибка.

Так же стоит проверить в этих файлах, что в параметр labels передается переменная labels (мы каждый раз вручную перезаписываем)



4. Затем запускайте RunRipperTasks, он должен зафиксировать ваши изменения, и по пути

src/main/adgp/databases/dwh/changelogs/ «Ваша схема»/tasks

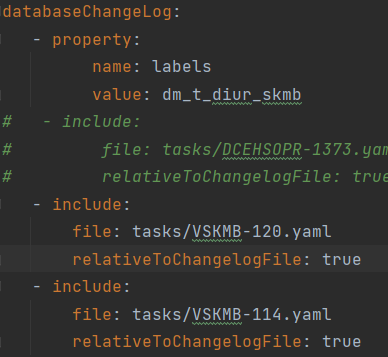
Будет создан yaml файл с названием вашей команды и номером задачи, который ссылается на все предыдущие yaml файлы.

Лично у меня не всегда этот скрипт срабатывает и приходится вручную создавать таску по этому пути

5. Нужно внести изменение в мастер файл который лежит по пути

src/main/adgp/databases/dwh/changelogs/ «Ваша схема»/master.yaml

и добавить в него ссылку на таску



6. По инструкции в ридми через докер развернуть локально БД и настроить соединение.

7. Для накатываний изменений нужно запустить градл скрипт в папке liquibase “UpdateSQL” и “update”.

Если падает ошибка парсинга, можно найти в каком файле произошла ошибка и скорее всего это будет связано с табуляцией или ошибкой в скрипте.

7.1 Если сделали какие-то изменения то можно вручную что-то удалить из БД для проверки, Так же перед “UpdateSQL” и “update” нужно вызвать функцию “dropAll”

8. Если все хорошо нактилось можно делать commit,push и создавать pull-request в битбакете.