How to test data migration scripts locally

Overview

While troubleshooting data migration script issues, I found myself looking for an easy way to verify these scripts without having to run a full blown FOLIO instance. Here's another option that requires only a database, and the ability to run the module under test locally.

Prerequisites

This is essentially one-time setup you'll need to do.

Database

First you'll need a database. Here's a how to setup postgres w/ docker:

```
# pull the desired postgres image
$ docker pull postgres:10

# start the db
$ docker run --rm --name pg_local -e POSTGRES_PASSWORD=mysecretpassword -p 5432:5432 -d postgres:10

# create role/db
$ docker exec -it pg_local /bin/bash
root@0728dc56d41e:/# su postgres
postgres@0728dc56d41e:/$ createuser -l -i -r -s -d -P folio (enter password when prompted)
postgres@0728dc56d41e:/$ createdb -O folio folio
postgres@0728dc56d41e:/$ exit
root@0728dc56d41e:/# exit

# create a config file for connecting to your db
$ cat <<EOF > /tmp/postgres-conf.json
{
  "host":"localhost",
  "port":5432,
  "username":"folio",
  "password":"letmein",
  "database":"folio"
}
EOF

# optional - create an image of postgres w/ your changes.  This will save you from having to do the work above
over and over again.
# next time you start your db, specify 'pg_folio' instead of 'postgres:10'
$ docker commit pg_local pg_folio
```

Clone and build modules

Now that you have a database, clone the git repo using the appropriate branch/tag and build with maven

```
$ git clone git@github.com:folio-org/mod-finance-storage -b v4.1.1 mod finance-storage-4.1.1
$ cd mod-finance-storage-4.1.1
$ mvn clean package

$ cd ..
$ git clone git@github.com:folio-org/mod-finance-storage -b v4.2.0 mod finance-storage-4.2.0
$ cd mod-finance-storage-4.2.0
$ mvn clean package
```

Run/Initialize the version you're upgrading from

Start the module up and hit the tenant API to initialize the database

```
$ cd mod-finance-storage-4.1.1

$ java -Dvertx.logger-delegate-factory-class-name=io.vertx.core.logging.SLF4JLogDelegateFactory -jar target/mod-
finance-storage-fat.jar -Dhttp.port=8082 -Dlog.level=debug db_connection=/tmp/postgres-conf.json

$ curl "http://localhost:8082/_/tenant" -H "X-Okapi-Tenant: testtenant" -H "X-Okapi-URL: http://localhost:8082"
-H "Content-type: application/json" -XPOST -d'
{
  "module_to": "mod-finance-storage-4.1.1",
  "parameters": [{
    "key": "loadSample",
    "value": "true"
  },{
    "key": "loadReference",
    "value": "true"
  }]
}'
```

Check that things are working

- via logging into the database and directly inspecting things
- making API calls to the module
- inspecting logs
- dump schemas for later comparison e.g. `pg_dump -hlocalhost -p5432 -Ufolio -dfolio -ntesttenant_mod_finance_storage -npublic > before.sql`
- etc.

Load additional data

This step is optional. If you want to add data to the system to exercise edge cases or see how things behave when migrating larger data sets, now is the time to do so.

- via API
- via directly loading them into the db, e.g. `psql \copy` or `INSERT`, etc.
- etc.

Run the version you're upgrading to and perform the upgrade

Stop the old module before proceeding

```
$ cd ../mod-finance-storage-4.2.0

$ java -Dvertx.logger-delegate-factory-class-name=io.vertx.core.logging.SLF4JLogDelegateFactory -jar target/mod-
finance-storage-fat.jar -Dhttp.port=8082 -Dlog.level=debug db_connection=/tmp/postgres-conf.json

$ curl "http://localhost:8082/_/tenant" -H "X-Okapi-Tenant: testtenant" -H "X-Okapi-URL: http://localhost:8082"
-H "Content-type: application/json" -XPOST -d'
{
  "module_from": "mod-finance-storage-4.1.1",
  "module_to": "mod-finance-storage-4.2.0",
  "parameters": []
}'
```

Validate upgrade and data migrations

- Inspect logs. However, only doing that is usually not good enough.
- Test data-migration scripts are run successfully by query the data to make sure the changes were applied
- Via logging into the database and directly inspecting things that you expect to be added or changed
    - Make sure the proper indexes are created by cross verify the new additions to schema.json with the indexes in the schema
    - All the required functions and triggers if any are updated according to the latest release
- Dump schemas and compare to previous, e.g. `pg_dump -hlocalhost -p5432 -Ufolio -dfolio -ntesttenant_mod_finance_storage -npublic > after.sql; diff before.sql after.sql`
- Making API calls to the module

Other considerations

- This can be tested on unreleased or snapshot versions of modules as well (e.g. HEAD of master, or even after making local changes), but keep in mind that the `module_to` version should align with the `fromModuleVersion` property in your module's `schema.json`
- If you find a problem and want to fix and repeat you can either "drop cascade" the schema and role for you tenant/module, or you can specify a different value for X-Okapi-Tenant when making the calls to the tenant API.
- If you prefer, an alternative to building the current/old module from source is to pull/run the official docker container from dockerhub, e.g. `docker pull folioorg/mod-finance-storage:4.1.1`
- Please help contribute here...