Migratron

Release 2.1.0

Jampp

Nov 17, 2020

CONTENTS

CHAPTER

ONE

MIGRATRON

A database migration system to modify the database schema or to populate the tables information.

This will work with:

- PostgreSQL
- Hive
- PrestoSQL/PrestoDB

1.1 Installation

Install the system dependencies. For Debian based distributions, they are:

```
apt-get install -y gcc python3-dev libpq-dev
```

Note: Use python2-dev if you are running migratron when using Python 2.7 instead of Python 3.X

You can install `migratron` as another Python package:

```
pip install migratron
```

Or you can install it from the repo:

```
cd $WORKSPACE
git clone https://github.com/jampp/migratron.git
cd migratron
python setup.py install
```

You will require different setups based on the database against which you need to run the migrations.

1.1.1 PostgreSQL

When using PostgreSQL, you only need to need to install a system package to be able to use `psql`. On Debian based distributions:

```
apt-get install -y postgresql-contrib
```

1.1.2 Hive

You need to setup `beeline`. This shouldn't be done on a production environment or locally, but a **very** basic guide is:

```
apt-get install -y openjdk-11-jre curl

export HIVE_VERSION=1.2.1
export HADOOP_VERSION=2.5.1

cd /opt && \
    curl https://archive.apache.org/dist/hive/hive-$HIVE_VERSION/apache-hive-$HIVE_
↪VERSION-bin.tar.gz -o apache-hive-bin.tar.gz && \
    tar -xzf apache-hive-bin.tar.gz && \
    rm -rf apache-hive-bin.tar.gz

# Download apache-hadoop
cd /opt && \
    curl https://archive.apache.org/dist/hadoop/core/hadoop-$HADOOP_VERSION/hadoop-
↪$HADOOP_VERSION.tar.gz -o hadoop-$HADOOP_VERSION.tar.gz && \
    tar -xzf hadoop-$HIVE_VERSION.tar.gz && \
    rm -rf apache-hive-bin.tar.gz

export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64/
export HIVE_HOME=/opt/apache-hive-$HIVE_VERSION-bin
export HADOOP_HOME=/opt/hadoop-$HADOOP_VERSION
export PATH=$PATH:$HIVE_HOME/bin
```

1.1.3 PrestoDB or PrestoSQL

You need to setup `presto-cli`. This shouldn't be done on a production environment or locally, but a **very** basic guide is:

```
apt-get install -y openjdk-11-jre curl

# Download PrestoCli
mkdir /opt/presto-cli && \
    cd /opt/presto-cli && \
    curl https://repo1.maven.org/maven2/io/prestosql/presto-cli/$PRESTO_CLI_VERSION/
↪presto-cli-$PRESTO_CLI_VERSION-executable.jar -o presto-cli && \
    chmod +x presto-cli

export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64/
export PATH=$PATH:$HIVE_HOME/bin
```

1.2 Creating a new Migration

> **Warning:** Never edit an already executed migration. Just create a new one

First of all, there are 2 types of migrations:

PRE These migrations are the ones that are backwards compatible. For example, add a column or table, or solve an issue on a view.

POST These migrations are used to cleanup the database once that all the servers code has been updated. These migrations are the ones that drop a column or table.

This is to take into account that all the servers can't be updated at the same time (which would cause some downtime). For example, if we want to rename a column there should be a *PRE* and a *POST* migration:

- The *PRE* should create the new column and populate the value from the existing column
- The *POST* should delete the old column

The migration filename should follow the following template: `YYYYMMDD_index_(pre or post)_(small description).sql`. For example: `20160801_0_pre_add_foobar_column.sql`. This isn't optional because the migration system will fail if the migration doesn't follow that format. The index is used to take into account that the same date can have more than one migration so the index identifies the order on which those file should be executed

1.3 Running Migrations

`Migratron` is a very simple migration system:

Given a path where the migrations files are, it will check which have not been applied. For this, it has a custom table on the database called `db_migrations` which has the path of the executed files and when they were executed.

`Migratron` works by doing the following:

- You write a plain SQL file as specified on *Creating a new Migration*
- It will check in its internal database (check *First Time*) if a migration failed to execute. If that happens, you need to run the `cleanup` subcommand. Check more about rollbacking a migration on *What it won't do* section
- Check which migrations are missing, and which ones should be executed: *PRE*, *POST*, or any. In any case, the order of execution of the migrations is based on the filename of the migrations
- Execute the missing migrations. For each of these:

  a. Connect to the corresponding database using a command line tool

  - Hive: `beeline`
  - PrestoSQL/PrestoDB: `presto-cli`
  - PostgreSQL: `psql`

  and run one of the files. In any case, those commands are required to run the migrations. For more information on how to setup them, check the *Installation instructions*.

b. If the file runs successfully, update the status database. When using Hive and PrestoSQL/PrestoDB you need to have another PostgreSQL database that is used to store `migratron` internal state. When using PostgreSQL you might use the same database, or another one.

   It will also store the hash of the file, to check if there is some change on the file. It is ok to change the migration file when doing development, but never change the migration file once it has been executed in production.

c. If it failed, then it will store the information that the migration failed.

1.3.1 First time

If it is the first time you are running the migration against the database, make sure to run the migrations with the `--just-base-schema` to create the `db_migrations` table. Check *Connect to the Database*

```
migratron initialize --just-base-schema --migrations-path PATH_TO_MIGRATION_FILES
```

Once that the initial `migratron` setup has been done you could just run the migrations as usual

1.3.2 Running Missing Migrations

To update the database information, you must use:

```
migratron migrate --migrations-path PATH_TO_MIGRATION_FILES
```

Run:

```
migratron migrate --help
```

to get more information on the required parameters, or check *Connect to the Database*

1.3.3 Working With Branches

`Migratron` doesn't take into account the different branches. So there are two solutions:

1. Create a new database for the branch you are using
2. Work on the normal database, and in case that you have to return to `master` or rollback the migration, do the steps manually.

1.3.4 Connect to the Database

Note: When using Postgres, the recommeded option is that the `db-uri` and `state-db-uri` reference the same database

There is more than one way that `migratron` can connect to the PostgreSQL database:

- The PostgreSQL environment variables
- Specifing the `db-uri` argument

In both cases, you can read more information about this using the `--help` parameter. For example:

```
migratron migrate --help
```

For other Hive and PrestoDB, the `db-uri` argument is required because there is no way to use the environment variables. For example, when using Hive, you should use something like:

```
migratron migrate \
    --db-uri 'jdbc:hive2://localhost:10000/test' \
    --db-type hive \
    --state-db-uri postgres://foo:bar@localhost/test1'
```

1.3.5 What it won't do

This is a list of things that other systems do but `migratron` won't do:

- Rollback a migration. If you want to downgrade the last migration or if a migration failed to run, then you must rollback the changes manually.
- Adapt migrations to different databases. The migrations are plain SQL files, so they might not run on different databases.

1.4 Running Tests

Migratron tests can be executed `pytest`

```
pip install -r requirements-dev.txt
pytest
```

Some tests require a PostgreSQL and others a beeline connection. To run the tests that require PostgreSQL, you must set the `MIGRATIONS_DB_TESTS` environment variable. The database used must have `test` somewhere in the name, and it should be created.

For example:

```
export MIGRATIONS_DB_TESTS=postgres://username:password@localhost/test_db
pytest
```

To run the tests that use Hive, you should set the `MIGRATIONS_HIVE_TESTS` environment variable

```
export MIGRATIONS_HIVE_TESTS='jdbc:hive2://localhost:10000/test'
pytest
```

1.5 Change Log

All notable changes to this project will be documented here.

Sort subsections like so: Added, Bugfixes, Improvements, Technical tasks. Group anything an end user shouldn't care deeply about into technical tasks, even if they're technically bugs. Only include as "bugfixes" bugs with user-visible outcomes.

When major components get significant changes worthy of mention, they can be described in a Major section.

1.5.1 v2.1.0 - 2020-11-17

Added

- Official docker image

Bugfixes

- Document what migratron does
- Exception message is valid when using Python2 and Python3
- Make the subcommand required and print the help message if missing

Technical Tasks

- Improve the documentation on how to setup migratron locally
- Add documentation on how to run the tests
- Add documentation on how migratron works

1.5.2 v2.0.1 - 2020-09-29

Bugfixes

- Added VERSION file to the python package

1.5.3 v2.0.0 - 2020-08-28

Added

- Options to run the migration against Hive or Presto
- Created the documentation
- The project can be used on Python3
- Description to the README.rst

Changed

- Rename the package to migratron
- Changed the `--database-uri|type` parameters to `--db-uri|type`

Bugfixes

- Added requirements-dev.txt to setup.py package_data
- Missing requirements.txt when using pip to install package
- Use Python 2 and 3 compatible function to get user input

Technical Task

- Do not use unittest deprecated methods
- Generate coverage report when running the tests
- Added missing classifiers to setup.py
- Create a version file
- Format the code using black
- Add Makefile to run tests and generate documentation
- Solve typos and improve documentation
- Fix min required version for some dev dependencies
- Added readthedocs configuration file
- Added long description to the setup.py
- Use MANIFEST.ini to add non python files to the package<Paste>

1.6 Indices and tables

- genindex
- modindex
- search

INDEX

P