Sqitch
VCS-enabled database change management

Ronan Dunklau

Kozea

2013-02-01
Outline

1. The problem
   - Overview
   - Solutions

2. Sqitch: overview
   - Goals
   - Concepts

3. Practical sqitch
   - Getting started
   - Writing changes
   - Deploying changes
   - VCS-enabled, you said?
   - Templates
Different kind of changes:

- **Database Schema changes (DDL statements)**

  ```sql
  ALTER TABLE sometable ADD COLUMN somecolumn VARCHAR
  ```

- **Static data changes (DML statements)**

  ```sql
  INSERT INTO vat_rates (vat_code, vat_country, timespan)
  values ('BTW', 'BE', '[2012-01-01, infinity)')::daterange
  ```

- **Procedures, Triggers, Extensions...**

  ```sql
  CREATE OR REPLACE FUNCTION increment(integer) RETURNS integer as $$
  select $1 + 1;
  $$ LANGUAGE sql;
  ```
Common problems

- **Current state of the db**  What should I deploy to be up-to-date?
- **Versioning**  How can I store the different revisions?
- **Concurrent work**  How can I merge my work with my colleagues?
- **Revert**  Is there a way to revert to a previous state?
Ordered scripts - bis

- Hard to maintain
- Tend to evolve into a full home-grown (or not) solution: sql migrations
- Merges are hell
Ordered scripts - bis

- Hard to maintain
- Tend to evolve into a full home-grown (or not) solution: sql migrations
- Merges are hell

Example

```bash
-lctrh
01-initial.sql 02-table1.sql 10-table2.sql
15-table3.sql 20-add_function.sql
21-add_another_function.sql
```
Domain-Specific-Languages (DSLs)

- Examples: Liquibase, Rails migrations, Django south, Sqlalchemy’s alembic...
- Often tied to a language/framework
- ”Opinionated” about what you can and should do
Sqitch goals

Goals:
- Easy-to-use
- VCS integration
- No silly conventions
- Easy configuration
- No opinions
Sqitch concepts

Concepts

- Change
- Status
- Tag
- Plan
Changes

- Named transition from a state A to a state B, along with the reverse transition.
- **Uniquely identified** (think of git commit hash)
- Defines ”dependencies” and ”conflicts” with other changes (optional)
- Actually defines three scripts:
  - deploy
  - revert
  - verify
Status

- Current database status
- Defined by the set of applied change

Example

```
user@machine% sqitch status
# On database sqitchtest
# Project: sqitch-tuto
# Change: 0694e414dc474004803c150df5ff61b672b47e6
# Name: change_pass
# Deployed: 2013-01-29 08:38:21 +0100
# By: Ronan Dunklau <ronan.dunklau@kozea.fr>
#
Nothing to deploy (up-to-date)
```
Tags

- Similar to a VCS tag \texttt{ex: git tag}
- Allows to "tag" a specific change to identify a particular state \texttt{ex: V1.0.0}
Plan

- Set of every defined change and tag
- Materialized by a simple "sqitch.plan" text file

Example

```
%syntax-version=1.0.0-b2
$project=sqitch-tuto
$uri=sqitch-tuto
appschema 2012-10-15T12:30Z Ronan Dunklau <ronan.dunklau@kozea.fr> # App schema for all sqitch tuto objects.
users [appschema] 2012-10-11T07:42:04Z Ronan Dunklau <ronan.dunklau@kozea.fr> # Creates table to track our users.
insert_user [users appschema] 2012-10-11T07:47:22Z Ronan Dunklau <ronan.dunklau@kozea.fr> # insert_user function
change_pass [users appschema] 2012-10-11T07:47:27Z Ronan Dunklau <ronan.dunklau@kozea.fr> # change_pass function
@v1.0.0-dev1 2012-10-11T07:51:41Z Ronan Dunklau <ronan.dunklau@kozea.fr> # Tag v1.0.0-dev1
```
Preamble

Plan

Change name

Changes

Tags

Comments
plan

- Preamble
- Plan
- Change name

Changes
- Tags
- Comments

%syntax-version=1.0.0-b
%project=sqitch-tuto
%uri=sqitch-tuto

appschema 2012-10-10T15:30Z Author <author@domain.com> # Comments.

users [appschema] 2012-10-11T07:42:04Z Author <author@domain.com> # Comments.

@v1.0.0-dev1 2012-10-11T07:51:41Z Author <author@domain.com> # Tag v1.0.0-dev1
Preamble

Plan

Change name

Changes

Tags

Comments
plan

- Preamble
- Plan
- Change name

Changes
Tags
Comments

%syntax-version=1.0.0-b
%project=sqitch-tuto
%uri=sqitch-tuto

appschema 2012-10-10T15:30Z Author <author@domain.com> # Comments.

users [appschema] 2012-10-11T07:42:04Z Author <author@domain.com> # Comments.

@v1.0.0-dev1 2012-10-11T07:51:41Z Author <author@domain.com> # Tag v1.0.0-dev1
plan

- Preamble
- Plan
- Change name

%syntax-version=1.0.0-b
%project=sqitch-tuto
%uri=sqitch-tuto

appschema 2012-10-10T15:30Z Author <author@domain.com> # Comments.
users [appschema] 2012-10-11T07:42:04Z Author <author@domain.com> # Comments.
@v1.0.0-dev1 2012-10-11T07:51:41Z Author <author@domain.com> # Tag v1.0.0-dev1
The problem
Sqitch: overview
Goals
Concepts
Practical sqitch

Preamble

Plan

Change name

%syntax-version=1.0.0-b
%project=sqitch-tuto
%uri=sqitch-tuto

appschema 2012-10-10T15:30Z Author <author@domain.com>  # Comments.

users [appschema] 2012-10-11T07:42:04Z Author <author@domain.com>  # Comments.

@v1.0.0-dev1 2012-10-11T07:51:42Z Author <author@domain.com>  # Tag v1.0.0-dev1

Changes

Tags

Comments
Installation

- **Install from CPAN (comprehensive perl archive network)**: `cpan -f App::Sqitch`

- **Available from the Kozea gentoo overlay.**
  
  https://github.com/Kozea/Overlay.git
Configuration

- **Per-project**
  
  ~/myproject/sqitch.conf

  - Project specific configuration: dbengine, dbname...
  - Edition via a text editor, or the sqitch config command
    
    user@host:# sqitch config core.engine pg

- **Per-user**
  
  ~/.sqitch/sqitch.conf

  - User configuration: user name, user email...
  - Edition via a text editor, or the sqitch config --user command
    
    user@host:# sqitch config --user user.name "Ronan Dunklau"
    user@host:# sqitch config --user user.email "ronan.dunklau@kozea.fr"
Initializing a new project

```
user@host % mkdir myproject
user@host % cd myproject
user@host % sqitch --engine pg init myproject --uri http://myproject/
Created sqitch.conf
Created sqitch.plan
Created deploy/
Created revert/
Created verify/
```
Default sqitch.conf

```ini
[core]
engine = pg
# plan_file = sqitch.plan
# top_dir = .
# deploy_dir = deploy
# revert_dir = revert
# verify_dir = verify
# extension = sql

# [core "pg"]
# db_name =
# client = psql
# sqitch_schema = sqitch
# password =
# port =
# host =
# username =
```
Anatomy of a change

- Entry in the sqitch.plan file
- 3 scripts, all named by "change_name.sql"
  - deploy
  - revert
  - verify
Adding a change

squitch add mynewtable -n "Add my newtable"
Created deploy/mynewtable.sql
Created revert/mynewtable.sql
Created verify/mynewtable.sql
Added "mynewtable" to sqitch.plan
Adding a change

Sqitch add mynewtable -n "Add my newtable"
Created deploy/mynewtable.sql
Created revert/mynewtable.sql
Created verify/mynewtable.sql
Added "mynewtable" to sqitch.plan

Creates everything
Adding a change

- `sqitch add mynewtable -n "Add my newtable"
  Created deploy/mynewtable.sql
  Created revert/mynewtable.sql
  Created verify/mynewtable.sql
  Added "mynewtable" to sqitch.plan

- Creates everything
- Just need to fill in the blanks!
Sqitch: overview

Practical sqitch

Getting started
Writing changes
Deploying changes
VCS-enabled, you said?
Templates

%syntax-version=1.0.0-b2
%project=myproject
%uri=http://myproject/

mynewtable 2013-01-31T09:57:11Z Ronan Dunklau <ronan.dunklau@kozea.fr> # Add mynewtable
Deploy Script

-- Deploy mynewtable

BEGIN;

-- XXX Add DDLs here.

COMMIT;
Deploy Script

```
-- Deploy mynewtable

BEGIN;

CREATE TABLE mynewtable ( id serial primary key, label varchar );

COMMIT;
```
-- Revert mynewtable

BEGIN;

-- XXX Add DDLs here.

COMMIT;
Revert Script

```sql
-- Revert mynewtable

BEGIN;
    DROP TABLE mynewtable;
COMMIT;
```
Verify Script

-- Test mynewtable

BEGIN;

-- XXX Add tests here.

COMMIT;
-- Test mynewtable

BEGIN;
    SELECT 1 from mynewtable;
COMMIT;
3 commands:
- sqitch deploy [target]
- sqitch revert [target]
- sqitch rebase [revert-target] [deploy-target]
Examples

user@host % sqitch status
# On database sqitchtest
No changes deployed

user@host % sqitch deploy
Deploying changes to sqitchtest
  + mynewchange .. ok

user@host % sqitch revert
Revert all changes from sqitchtest? [Yes]
  - mynewchange .. ok
user@host % sqitch add mysecondtable --requires mynewtable -n "Add second tables"
Created deploy/mysecondtable.sql
Created revert/mysecondtable.sql
Created verify/mysecondtable.sql
Added "mysecondtable [mynewtable]" to sqitch.plan

%syntax-version=1.0.0-b2
%project=myproject
%uri=http://myproject/

mynewtable 2013-01-31T09:57:11Z Ronan Dunklau <ronan.dunklau@kozea.fr> # Add mynewtable
mysecondtable [mynewtable] 2013-01-31T10:04:25Z Ronan Dunklau <ronan.dunklau@kozea.fr> # Add second tables
Ready to release

user@host % sqitch tag v0.0.1 -n First release
Tagged "mysecondtable" with @v0.0.1
user@host % sqitch bundle
Bundling into bundle
Writing config
Writing plan
Writing scripts
  + mynewtable
  + mysecondtable @v0.0.1
Easy to merge

- Each change lives in its own set of files
- Only sqitch.plan may produce merge conflict
Easy to merge

- Each change lives in its own set of files
- Only sqitch.plan may produce merge conflict
  - Easy to resolve: merge=union
Each change lives in its own set of files

- Only sqitch.plan may produce merge conflict
  - Easy to resolve: merge=union
  - Simple sequence + dependencies
Rework

Rework = modify an existing script Ex: rewrite a function, or change a column data type.

- sqitch rework mynewtable -m "Add a column"
  Added "mynewtable [mynewtable@v0.0.1]" to sqitch.plan.
  Modify these files as appropriate:
  * deploy/mynewtable.sql
  * revert/mynewtable.sql
  * verify/mynewtable.sql

- Copy mynewtable.sql to mynewtable@v0.0.1.sql
Rework

Rework = modify an existing script. Ex: rewrite a function, or change a column data type.

- sqitch rework mynewtable -m "Add a column"
  Added "mynewtable [mynewtable@v0.0.1]" to sqitch.plan.
  Modify these files as appropriate:
  * deploy/mynewtable.sql
  * revert/mynewtable.sql
  * verify/mynewtable.sql

- Copy mynewtable.sql to mynewtable@v0.0.1.sql

Warning

Make sure everything is consistent! Deploy:
mynewtable@v0.0.1 -z mysecondtable -z v0.0.1 -z mynewtable
Revert: mynewtable -z v0.0.1 -z mysecondtable -z mynewtable@v0.0.1
Templates

- Custom templates creation for deploy / revert / verify scripts
- Deploy.templates configuration
Templates

- Custom templates creation for deploy / revert / verify scripts
- Deploy.templates configuration
- Available as a command-line options:
  - --template-directory
  - --deploy-template
  - --revert-template
Why use templates?

- Enhance productivity
- Enforce conventions
Example: template

```sql
-- Deploy [% change %]
[% FOREACH item IN requires -%]
-- requires: [% item %]
[% END -%]
[% FOREACH item IN conflicts -%]
-- conflicts: [% item %]
[% END -%]

BEGIN;

CREATE FUNCTION [% change %] () RETURNS VOID AS $$
-- XXX function body
$$ language sql;

COMMENT ON FUNCTION [% change %] IS ' [% comment %]';

COMMIT;
```
Example: Add change

user@host % sqitch add randomstring \ 
   --template-dir=./functiontemplates/ \ 
   --set comment="Returns a random string"
Created deploy/randomstring.sql
Created revert/randomstring.sql
Created verify/randomstring.sql
Added "randomstring" to sqitch.plan

-- Deploy randomstring

BEGIN;

CREATE FUNCTION randomstring () RETURNS VOID AS $$
   -- XXX function body
$$
language sql;

COMMENT ON FUNCTION randomstring IS 'Returns a random string';

COMMIT;
Questions

Thank you for your attention! Questions?