Drush Rebuild Documentation

Release 7.x-1.x

Kosta Harlan

July 29, 2016

Contents

Drush Rebuild is a utility for rebuilding your local development environments. It uses your existing Drush aliases and an easy-to-read config file that defines the tasks for rebuilding your local environment.

CHAPTER 1

User Guide

1.1 Overview

1.1.1 Requirements

1. PHP 5.3
2. Drush 6 or later

Note: Drush Rebuild requires that you have installed Drush 6 or 7 using the composer install method.

1.1.2 Installation

The recommended install method is via Drush:

```
drush dl rebuild
# Project rebuild (7.x-1.x) downloaded to $HOME/rebuild.
```

Use the same method to update to the latest stable release.

For a development version, install via git:

```
cd ~/.drush
git clone --branch 7.x-1.x http://git.drupal.org/project/rebuild.git
```

1.1.3 Runnning the tests

Drush Rebuild contains a suite of tests. To run them, do:

```
cd ~/.drush/rebuild/tests
./runtests.sh
```

1.2 Usage

1.2.1 Getting Started

Drush Rebuild uses Drush Aliases to get things done. Don't know what Drush alias is? Read `drush docs-aliases` and come back to this page. It's just a shortcut for pointing to a site. It lets us conveniently run commands like `drush @example.prod core-status` to get status info from production site.

Let's consider a Drupal site with the following Drush alias file defined in `~/.drush/example.aliases.drushrc.php`:

```
$aliases['local'] = array(
  'root' => '/home/kosta/sites/example/live-docs',
  'uri' => 'http://local.example.com',
);

$aliases['stage'] = array(
  'uri' => 'http://stage.example.com',
  'root' => '/var/www/html/example.com/live-docs',
  'remote-host' => 'stage.example.com',
  'remote-user' => 'kosta',
);

$aliases['prod'] = array(
  'uri' => 'http://www.example.com',
  'root' => '/var/www/html/example.com/live-docs',
  'remote-host' => 'example.com',
  'remote-user' => 'kosta',
);
```

Pretty straightforward. Now, we want to rebuild our local environment based on the latest DB in production.

The traditional way might involve runninng `mysqldump` in production and downloading that, then importing via `mysql`. Or, more advanced users might do something like `drush sql-sync @example.prod @example.local` – but wait, we also have to change some variables, and enable `reroute_email`, etc. By the time you're done, you might have run a dozen or two commands.

Now try repeating the process or asking your colleague to do so. Or wait a couple months and come back to the project, and try to remember the steps you took. We need a way to simplify this process and make it repeatable. That's where Drush Rebuild comes in.

1.2.2 Configuring your aliases

As mentioned at the beginning, Drush Rebuild hooks into your existing aliases. So let's look at our local alias again:

```
$aliases['local'] = array(
  'root' => '/home/kosta/sites/example/live-docs',
  'uri' => 'http://local.example.com',
);
```

We're going to make a very simple change to this alias:

```
$aliases['local'] = array(
  'root' => '/home/kosta/sites/example/live-docs',
  'uri' => 'http://local.example.com',
  'path-aliases' => array(
    '%rebuild' => '/home/kosta/sites/example/resources/rebuild.yaml',
```

```
  ),
);
```

What we are doing is telling Drush Rebuild that a configuration file exists at `examples/resources/rebuild.yaml`.

1.2.3 Writing the configuration file

1.2.4 Commands

1.3 Configuration

The core of Drush Rebuild revolves around the configuration file. Consider the command `drush @example.local rebuild`. In this command, Drush Rebuild is going to rebuild the local development environment for `@example.local`, and it is going to do so based on the tasks defined in the configuration file.

To start with, you should know that the rebuild config has three main sections: `general`, `sync/site_install`, and `drupal`.

1.3.1 general

```
general:
  description: 'Rebuilds local development environment from remote destination'
  uli: true
  overrides: 'local.rebuild.yaml'
  default_source: '@example.prod'
  drush_scripts:
    pre_process: ['example.php', 'another.php']
    post_process: 'after_rebuild.php'
```

description

This key is self-explanatory. It is displayed to the user when they run `drush @example.local rebuild`.

uli

This key tells Drush to run `drush @example.local uli` after completing all the rebuild tasks. Set this to `false`, or leave it out entirely, if you don't want that.

overrides

This key is used to specify the path to a local overrides configuration file. For example, if your main configuration was at `/home/kosta/sites/example/resources/rebuild.yaml`, your overrides would be at `/home/kosta/sites/example/resources/local.rebuild.yaml`.

This is useful when a team is using the same rebuild file. You can define a local overrides file, and exclude it from version control, so that each team member can customize the rebuild process to their liking.

default_source

Drush Rebuild lets you rebuild your local environment based on any source defined in your drush alias, so you could rebuild based on `@staging` or `@prod` aliases for example. But more often than not, you want your local development environment to match one remote environment. By defining default_source you can save yourself some typing. You'll be able to run `drush @example.local rebuild` instead of `drush @example.local rebuild --source=@example.prod`.

drush_scripts

This section tells Drush to run `drush @example.local php-script` on the files specified in the `pre_process` and `post_process` sections. `pre_process` runs before any other step, while `post_process` comes at the very end. Note that returning `FALSE` from a Drush script will halt the rebuild process.

1.3.2 sync /site_install

The `sync` or `site_install` section tells Drush that we are either using `sql-sync` or `site-install` for rebuilding a local site. They are mutually exclusive; you can't have both in the same config.

sql_sync

This section lets you define options for syncing a remote database to your local environment.

```
sql_sync:
  create-db: 'TRUE'
  sanitize: sanitize-email
  structure-tables-key: common
```

If you just wanted database syncing without any additional options, you could write:

```
sync:
 sql_sync: true
```

Note that any option listed in `drush help sql-sync` can be defined in your rebuild config file.

site_install

If you are rebuilding by re-installing an install profile, you can set options like:

```
site_install:
  profile: 'standard'
  account-mail: 'admin@localhost'
  account-pass: 'admin'
  account-name: 'admin'
  site-name: 'Local install'
```

Any option listed in `drush help site-install` can be defined in the config file.

1.3.3 Review

Let's take a look at the entire file now:

```
general:
  description: 'Rebuilds local development environment from remote destination'
  uli: true
  overrides: 'local.rebuild.yaml'
  drush_scripts:
    pre_process: ['example.php', 'another.php']
    post_process: 'after_rebuild.php'

sync:
  default_source: '@example.prod'
  sql_sync:
    create-db: 'TRUE'
    sanitize: 'sanitize-email'
    structure-tables-key: 'common'
  rsync:
    files_only: 'TRUE'

drupal:
  variables:
    set:
      preprocess_js: 0
      preprocess_css: 0

  modules:
    enable:
      - devel
      - devel_node_access
      - dblog
      - views_ui
    disable:
      - overlay
      - syslog
    uninstall:
      - google_analytics

  permissions:
    anonymous user:
      grant:
        - access devel information
        - switch users
      revoke:
        - search content
    authenticated user:
      grant:
        - access devel information
      revoke:
        - search content
```