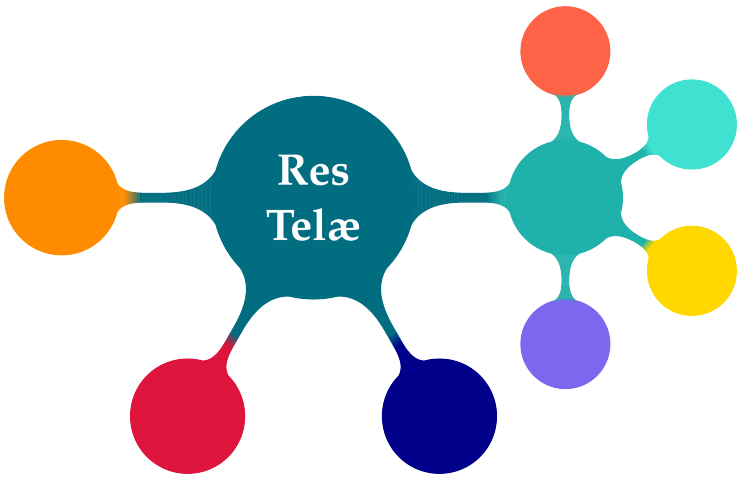
Composer pour Drupal 8

Felip Manyer i Ballester



15 mars 2017

À propos de moi

- Drupalien depuis fin 2009, centralien de Lyon (2008).
- Exerce en indépendant à Perpignan sous le nom « Res Telæ ».
- Vient aux meetups pour donner libre cours à sa logorrhée, rencontrer des gens comme lui.
- Libriste fermement opposé à la confiscation et la minitellisation d'Internet.
- Loisirs : natation, cyclisme, sports de montagne, piano, sciences naturelles, linguistique, OpenStreetMap, etc.

Plan

Motivation

☞ Comment gérer un projet Drupal ?

...Avec Git bien sûr ! Mais encore ?

Intégralité de la plate-forme

- Suivre tous les fichiers, dont le cœur, les modules et les bibliothèques contribués.
- Plus « simple », pas de dépendance externe.
- Téléchargements et mises à jour manuels.
- Commits moins clairs.

Ne gérer que son propre code

- Dépendance de l'extérieur.
- Étape de construction.
- Commits plus clairs et concis.
- Téléchargements et mises à jour facilités.
- Gestion avancée des « patchs ».

Drush make (makefiles), ancêtre de Composer

- Format de fichier ($\sim$ dosini, puis yaml) permettant de décrire une plate-forme Drupal, à la version près, et de la reconstituer (de manière *reproductible*) grâce à Drush.
- Récupération automatique du cœur, des modules, thèmes, bibliothèques, traductions, via HTTP, SVN, Git... éventuellement patchés (proprement !) à la volée.

Un exemple de makefile

```
core = 7.x
api = 2

; Modules
projects[addressfield][version] = "1.2"
projects[admin_menu][version] = "3.0-rc5"
projects[advanced_link][version] = "1.3"
projects[auto_nodequeue][version] = "2.1"

; Libraries
libraries[ckeditor][download][type] = "file"
libraries[ckeditor][download][url] = "http://download.cksource.com/
  CKEditor/CKEditor/CKEditor%204.4.7/ckeditor_4.4.7_standard.zip"
libraries[ckeditor][download][sha1] = "fa6cc00cc5422b92bbe61f4ea84b
  197bccdf332f"
libraries[ckeditor][directory_name] = "ckeditor"
libraries[ckeditor][type] = "library"
```

Un exemple de makefile (yaml)

```
core: 7.x
api: 2
defaults:
  projects:
    subdir: 'contrib'
projects:
  drupal:
    type: core
    version: '7.53'
  addressfield:
    version: 1.0-beta5
    patch:
      2192381:
      'https://www.drupal.org/files/issues/addressfield-
  entityreference_autofill_support-2192381-4.patch'
  admin_menu:
    version: 3.0-rc4
```

Apports de Composer

- Capable de reconstruire uniquement les changements.
- Intégré à l'ensemble du monde PHP, et au-delà.
- Capable à la fois de télécharger, mettre à jour ou supprimer **et** de modifier le fichier `composer.json` en une seule commande.

Historique

- Gestionnaire de dépendances utilisé dans le monde PHP.
- Écrit par Nils Adermann et Jordi Boggiano, première version en mars 2012.
- S'inspire d'autres gestionnaires de paquetages tels que npm (node.js) et bundler (Ruby).
- Désormais très populaire dans le monde PHP.
- Entièrement supporté par Drupal 8 (voir Composer Manager pour Drupal 7).

« Getting off the island. »

Composants

Exécutable à appeler en ligne de commande : `composer`.

Fichiers `composer.json` (description de la plate-forme) et `composer.lock` (permet de figer les versions). Les deux doivent être sous gestion de version.

Installer Composer

- Depuis vos dépôts (distribution « rolling release »).
- En suivant les instructions sur `https://getcomposer.org/download/`.

Exemple de fichier composer.json

```
{
    "name": "restelae-drupal/setsivelles",
    "description": "Set Sivelles",
    "type": "drupal-profile",
    "require-dev": {
        "drupal/devel": "^1.0@beta",
        "drupal/console": "~1.0"
    },
    "require": {
        "composer/installers": "^1.0",
        "drupal-composer/drupal-scaffold": "^2.0.0",
        "cweagans/composer-patches": "^1.5.0",
        "drupal/core": "~8.2.0",
        "drupal/admin_toolbar": "^1.18",
        "drupal/pathauto": "^1.0@beta",
        "drupal/token": "^1.0@beta",
    },
}
```

Démarrer un projet

En utilisant un *template*

```
composer create-project drupal-composer/drupal-project :8.x-dev
my_site_name-dir –stability dev –no-interaction
```

En partant d'un fichier composer.json existant

```
composer install
```

Téléchargement et suppression d'un paquetage

Téléchargement

composer require drupal/pathauto

Suppression

composer remove drupal/pathauto

Composer n'interagit pas avec la base (installation, mise à jour, désinstallation). Drush (voire Console) restent nécessaires.

Contraintes de version

Exemple

composer require drupal/pathauto^1.0

Télécharge la dernière version de Pathauto < 2.0.
Facultatif, Composer ajoute de lui-même cette contrainte en fonction des réglages dans `composer.json`.

Quelques contraintes (tirées de la documentation)

^1.2.3	>=1.2.3 <2.0.0
~1.3	>=1.3.0.0-dev <2.0.0.0-dev
1.4.*	>=1.4.0.0-dev <1.5.0.0-dev

Installation et mise à jour

Installation

composer install [-n] [–no-dev]

Installer les dépendances aux versions figées dans `composer.lock` (ou les déterminer automatiquement et le créer).

Mise à jour

composer update [-n] [–no-dev]

Mettre à niveau les versions dans `composer.lock` (cf. contraintes de versions dans `composer.json`) et télécharger les paquets correspondants.

Dépôt

Au départ, on utilisait Packagist. Nous disposons désormais de nos propres dépôts :

```
"repositories": {
    "drupal": {
        "type": "composer",
        "url": "https://packages.drupal.org/8"
    }
}
```

Dépendances nécessaires

composer/installers	Définir répertoire d'installation en fonction du type (module, bibliothèque, thème).
drupal-composer/drupal-scaffold	S'adapter à la structure particulière de Drupal et fichiers robots.txt, .htaccess...
cweagans/composer-patches	Permettre l'application automatique et propre des *patchs*.

Répertoire d'installation

Exemple :

```
"extra": {
    "installer-paths": {
        "docroot/core": [
            "drupal/core"
        ],
        "docroot/profiles/rs2017/modules/contrib/{$name}": [
            "type:drupal-module"
        ],
        "docroot/profiles/rs2017/themes/contrib/{$name}": [
            "type:drupal-theme"
        ],
        "docroot/libraries/{$name}": [
            "type:drupal-library"
        ]
    },
```

drupal-scaffold

Exemple :

```
"extra": {
    "drupal-scaffold": {
        "excludes": [
            ".csslintrc",
            ".editorconfig",
            ".eslintignore",
            ".eslintrc",
            ".gitattributes",
            "web.config"
        ]
    }
}
```

Appliquer un patch

```
"extra": {
    "patches": {
        "drupal/geocoder": {
            "2747193: Implement GPX File Parsing Provider":
            "https://www.drupal.org/files/issues/geocoder-add_gpx...
        },
        "drupal/superfish": {
            "2825106: HTTP 500 Internal Server Error after...":
            "https://www.drupal.org/files/issues/fixInternalLinks...
        },
        "drupal/leaflet_more_maps": {
            "2848895: Update Thunderforest maps":
            "https://www.drupal.org/files/issues/update_thunderfor...
        }
    }
}
```

Définir un dépôt manuellement

Parfois, certaines dépendances ne sont pas connues de Composer. C'est le cas d'une bibliothèque comme Leaflet :

```
"repositories": {
    "leaflet": {
        "type": "package",
        "package": {
            "name": "leaflet/leaflet",
            "version": "1.0.2",
            "type": "drupal-library",
            "dist": {
                "url": "https://github.com/Leaflet/Leaflet/archive/
                  v1.0.2.zip",
                "type": "zip"
            }
        }
    }
```

Merci !

Références

- *Site de Composer*, `https://getcomposer.org/`
- *Using Composer to manage Drupal site dependencies*, `https://www.drupal.org/docs/develop/using-composer/using-composer-to-manage-drupal-site-dependencies`