Configuration Management

Drupal Camp Alpe-Adria 2014

Fabian Bircher

NUVOLE

Agenda

- Introduction
- Configuration Management for site builders
- From Drupal 7 to Drupal 8
- A Closer Look at CMI
- CMI for Developers

Fabian Bircher

- Developer at Nuvole
  - Drupal company
  - Brussels, Parma, Prague
  - Pioneers of Code Driven Development:
    configuration management for Drupal 6 and Drupal 7.

@fabianbircher

@nuvoleweb

Configuration Management

An overview and practical introduction for site builders

Configuration Management Initiative

- “Site configuration information in D8 is **stored in files** in your library’s directory, making it much simpler to **transport configuration changes** such as new content types, fields, or views from development to production.”

- The initial goal, with some implementation differences, was reached. CMI already works in the Drupal 8 branch.

Reference Use Case

- Modify the configuration of a production site:
  - Keeping the site online all the time.
  - Developing/testing the new configuration on a development copy.
  - Exporting the configuration changes from development and importing them in production.

Step 1 of 6: Clone Site to Dev

PRODUCTION

- Copy:
  - Database
  - Full Drupal tree
  - Files

DEVELOPMENT

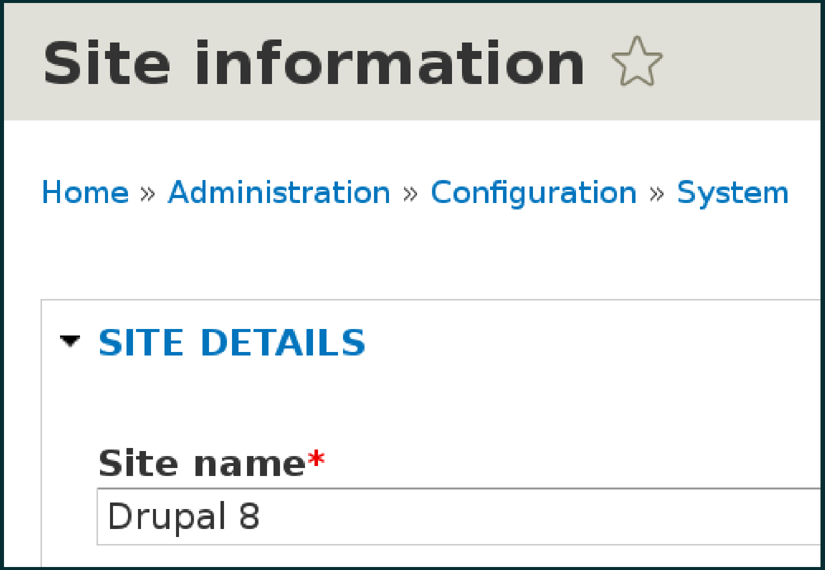
- Restore the copy

Step 2 of 6: Modify Configuration

PRODUCTION

- Site operates normally:
  - new users
  - new content

DEVELOPMENT


Site information

Home » Administration » Configuration » System

SITE DETAILS

Site name*

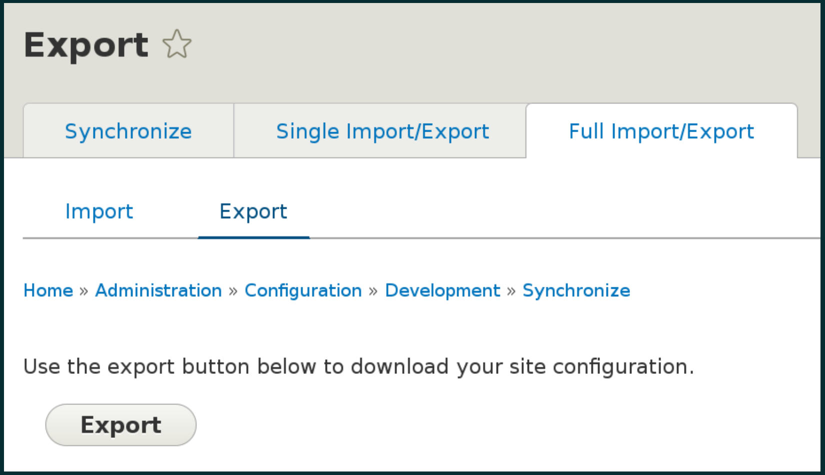
Drupal 8

Step 3 of 6: Export Configuration

PRODUCTION

- Site operates normally:
  - new users
  - new content

DEVELOPMENT

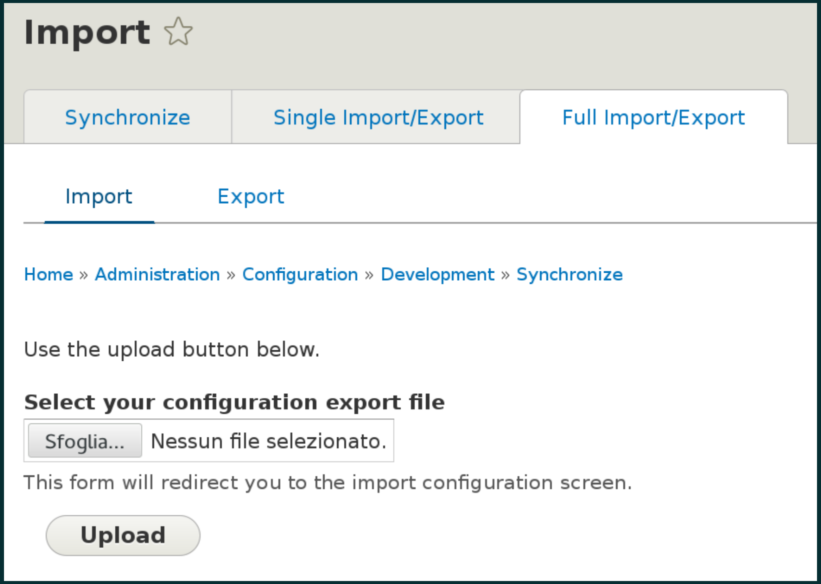


Use the export button below to download your site configuration.

Step 4 of 6: Import in Staging

PRODUCTION

DEVELOPMENT



Use the upload button below.

Select your configuration export file

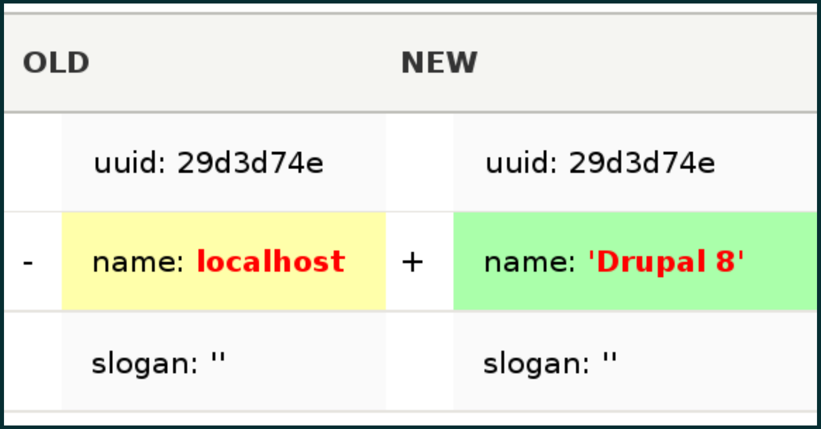
Nessun file selezionato.

This form will redirect you to the import configuration screen.

Step 5 of 6: Review Changes

PRODUCTION

DEVELOPMENT

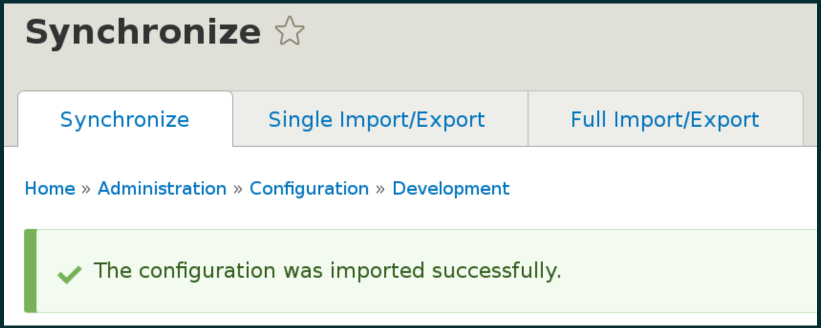


	OLD		NEW
	uuid: 29d3d74e		uuid: 29d3d74e
-	name: **localhost**	+	name: **'Drupal 8'**
	slogan: ''		slogan: ''

Step 6 of 6: Apply Changes

PRODUCTION

DEVELOPMENT



The configuration was imported successfully.

There's Much More...

- This is only the site builder's user experience.
- CMI is still under active development.
- Significant changes can still occur before Drupal 8 is released.

From Drupal 7 to Drupal 8

What changed. What improved. What's still missing.

D7 to D8: Configuration is defined

- Are vocabularies configuration?
- Are taxonomy terms configuration?

- If it's in the config, it's configuration!

D7 to D8: Configuration Storage

- Database

- Text files
  (well... text files stored in DB by default)

D7 to D8: Uniform Approach

- Variables
- DB Tables
- CTools
- Features
- Black magic...

- Text files in YAML format

D7 to D8: Staging Configuration

7

- Through Features,
- revert to apply

8

- Native

D7 to D8: Interface to Developers

- “Exportables”
- from CTools and bag of (incompatible) tricks

- “Configurables”
- as PHP classes (entities)

D7 to D8: Comparison

	D7 core	D7 core + features	D8 core
Export full site config (no content)	NO	NO	YES
Export selected config items	NO	YES	YES
Track config changes (full site)	NO	NO	YES
Track config changes (selected items)	NO	YES	YES
Stage configuration	NO	YES	YES
Package configuration	NO	YES	NO
Reuse configuration in other projects	NO	YES	NO
Collaborate on the same project	NO	YES	NO*

Is CMI “Features Done Right”?

- **No.**
- It is a nice way to replace and improve one use case for Features: making configuration exportable into text files.

Is CMI “Features Done Wrong”?

- **No.**

- It is a huge step forward to developers and it paves the way for additional modules that could offer the same functionality of Drupal 7 + Features in a much cleaner and more reliable way.

A Closer Look at CMI

Under the hood. Caveats and pending developments.

Key-Value store

- Concept
  - Unique key
  - Potentially complex data
    - Declarative (no logic)
- Implementation
  - Files
  - Two columns in a database
  - …

Two Levels of Configuration

- Active store
  - The real site configuration. If you only configure “D7-style”, you'll use this one and never see CMI.
- Staging store
  - Temporary area for configuration files that are to be reviewed and imported.

Configuration in settings.php

```
$config_directories['active'] =
'sites/default/files/config_al6ppw6/active';
$config_directories['staging'] =
'sites/default/files/config_al6ppw6/staging';
```

- The random string is for extra security.
- The two directories can be out of the Drupal root.

A Look at the Active Store

```
$ ls sites/default/files/config_al6ppw6/active
$
```

- Empty?

Files are in Database

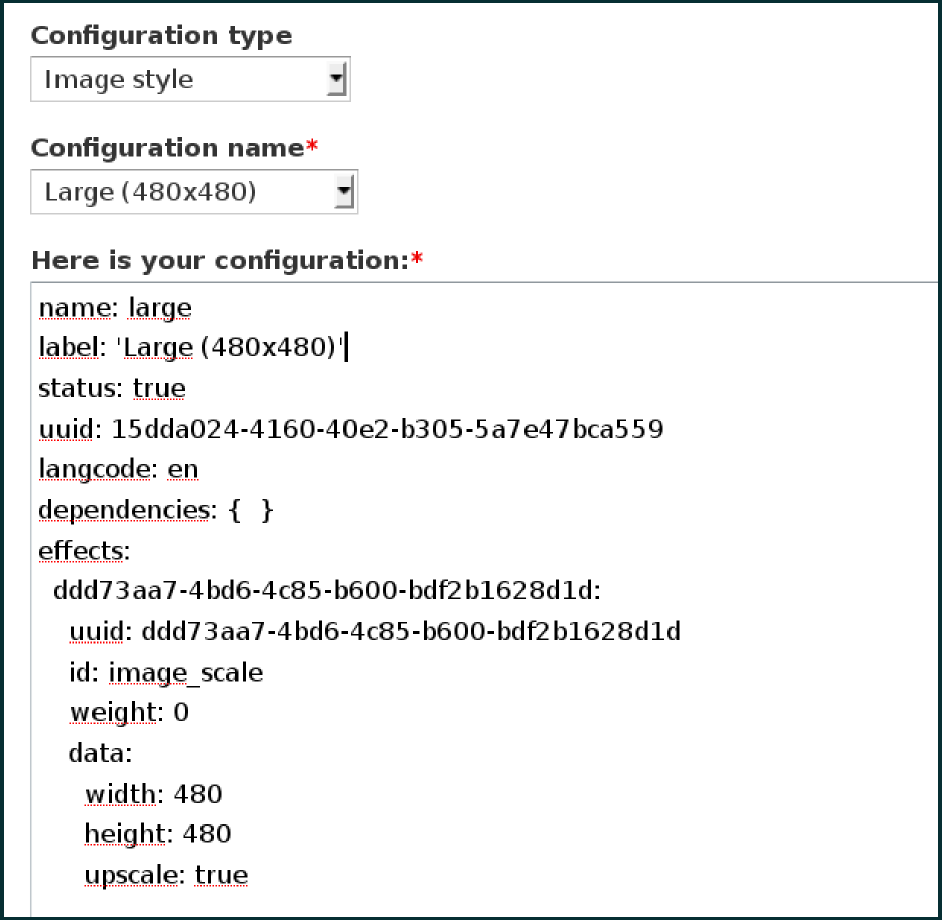
```
MySQL> select name FROM config;
+-----------------------------------+
| name                              |
+-----------------------------------+
| bartik.settings                   |
  ...
| views.view.who_s_new              |
| views.view.who_s_online           |
+-----------------------------------+
166 rows in set
```

- Files are binary blobs in DB (backend pluggable)

Why the Database?

- Performance
  - It's faster than reading/parsing files.
- Safety
  - No temptation to edit files “because you can”.
  - Changes ALWAYS need to be imported!
- Security
  - Less likely to leave read access accidentally open.

YAML Example: in Drupal



Configuration type

Image style

Configuration name*

Large (480x480)

Here is your configuration:*

```
name: large
label: 'Large (480x480)'
status: true
uuid: 15dda024-4160-40e2-b305-5a7e47bca559
langcode: en
dependencies: {  }
effects:
  ddd73aa7-4bd6-4c85-b600-bdf2b1628d1d:
    uuid: ddd73aa7-4bd6-4c85-b600-bdf2b1628d1d
    id: image_scale
    weight: 0
    data:
      width: 480
      height: 480
      upscale: true
```

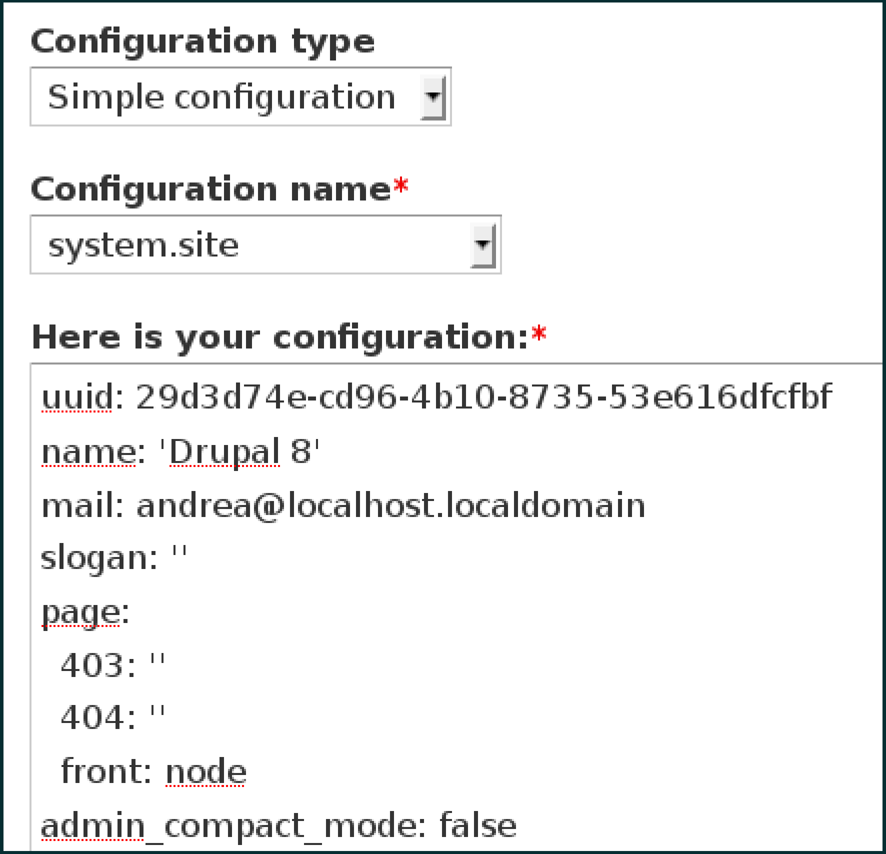
YAML Example: in Exported Config

```
$ cat image.style.large.yml
name: large
label: 'Large (480x480)'
status: true
uuid: 15dda024-4160-40e2-b305
langcode: en
dependencies: {  }
effects:
  ddd73aa7-4bd6-4c85-b600:
    uuid: ddd73aa7-4bd6-4c85-b600
    id: image_scale ...
```

YAML Example: in Default Config

```
$ pwd
.../core/modules/system/config/install
$ cat system.site.yml
uuid: ''
name: Drupal
mail: ''
slogan: ''
page:
  403: ''
  404: ''
  front: user
admin_compact_mode: false
```

YAML Example: After Install



Configuration type

Simple configuration

Configuration name*

system.site

Here is your configuration:*

```
uuid: 29d3d74e-cd96-4b10-8735-53e616dfcfbf
name: 'Drupal 8'
mail: andrea@localhost.localdomain
slogan: ''
page:
  403: ''
  404: ''
  front: node
admin_compact_mode: false
```

UUIDs Everywhere

- Good
  - Prevent any possible conflicts.
- Bad
  - “Universally Unique” clashes with “Reusable” (especially with relations).

Caveat: CMI is for the Same Site

- No portability, by design
  - The export/import cycle is assumed to be between multiple version (dev, production) of the same Drupal project.

Caveat: CMI is not Atomic

```
$ cat user.role.authenticated.yml
id: authenticated
label: 'Authenticated user'
weight: 1
permissions:
  - 'use text format plain_text'
  - 'access content'
  - 'use text format basic_html'
  - 'access comments'
```

- Can't package a “feature” reusing whole files. Need for an intermediate layer.

Caveat: Import Can Break Things

```
$ [export config]
$ rm node.type.page.yml
$ [import config]
```

- Existing pages are orphaned and unusable; always make a backup dump or review changes carefully!

Caveat: No Consistency Check

```
$ [export config]
$ vim system.theme.yml
$ cat system.theme.yml # Typo in name!
admin: seven
default: barTtik
$ [import config]
```

- An invalid configuration is imported and applied; style is broken.

Pending: Install from Existing Config

- Issue 1613424

- Idea: create a clean copy (all config, no content) of a production site.
- Theoretically, possible by providing an “install fom existing config” choice as if it were an installation profile (see issue for discussion).

Pending: Robust Synchronization

- Issue 2121751

- Idea: Synchronize sites in a stable way, handling possible conflicts.
- Use case: handle gracefully the case where the same View was created independently on production and development (see issue for discussion).

CMI for Developers

Configurables. How to work with configuration.

Configurables

- As everything in Drupal 8, object-oriented interface:
- `abstract class ConfigEntityBase`

- Namespace
- `Drupal\Core\Config\Entity`

Configurables are Entities

```
class ConfigEntityBase extends Entity { ...
}

Drupal\Core\Entity\Entity
|- Drupal\Core\Config\Entity\ConfigEntityBase
\- Drupal\Core\Entity\ContentEntityBase
```

Two namespaces: one for config, one for content.

Retrieving Configuration Objects

```
Drupal::config($name)
```

Where **$name** is the configuration object (filename of the YAML file, without the **.yml** extension).

Example:

```
$site_name = Drupal::config('system.site')
               ->get('name');
```

Modifying Configuration Objects

```
Drupal::config('system.site')
        ->set('name', 'Drupal 8 test')
        ->save();
```

- The active store is immediately modified. The staging store is not used.

Drush 7.x Integration

```
config-list (cli)
    List config names by prefix.
config-export (cex)
    Export config from the active store.
config-import (cim)
    Import config from a config dir.
config-get (cget)
    Display a config value, or a whole
    configuration object.
config-set (cset)
    Set a config value directly in
    the active configuration.
```

Configuration EVERYTHING?

- variable_xxx() is dead!
- How is the time of the last cron run saved?
- State API to the rescue!
  - Environment specific settings

- Separation of configuration, state and content
  - Content staging not implemented yet

Settings overrides

- Configuration can be locally overridden
  - CSS & JS aggregation
  - TWIG debugging

- settings.local.php

```
$config['system.performance']['css']
['preprocess'] = FALSE;
```

CSV workflow

PRODUCTION

- Clone site
- don’t touch config
- Pull from git
- Import configuration

DEVELOPMENT

- Insall clone
- Configure
- Export configuration into staging
- Commit&push to git

More…

- Jam’s virtual drupal camp featuring Rob Bayliss

https://www.youtube.com/watch?v=St9s7DqFR7o

Thank you for your attention