DUBLIN DRUPALCON

NUVOLE

Configuration Management

Theory and practice

Andrea Pescetti
andrea@nuvole.org

Fabian Bircher
fabian@nuvole.org

Antonio De Marco
antonio@nuvole.org

web: nuvole.org **twitter:** @nuvoleweb

Our Distributed Team

Nuvole: a **100% Drupal company** with a distributed team in:

Italy

Belgium

Czech Republic

Our Clients

- International organisations
- Institutions
- Fast delivery: several developers working simultaneously on the same project
- Frequent configuration changes: need for safe updates

Chapter 1

Can I deploy configuration?

- Can I develop/test configuration on a development copy and keep the production site online all the time?
- Can I export configuration changes from development and import them into production?

Configuration Management

- One of the most important improvements in Drupal 8.
- A new and complete solution to an ancient problem.
- Reference use case: export the configuration changes from development and import them into production.

The reference use case: step 1 of 6

Clone Site to Dev

Production

- Install Site
- Make a full backup
- Database
- Files

Development

- Clone production site by restoring the backup

...or the other way round.

The reference use case: step 2 of 6

Modify Configuration

Production

Site operates normally:

- new users.
- new content.

Development

- Configuration changes...

The reference use case: step 3 of 6

Export Configuration

Production

Site operates normally:

- new users.
- new content.

Development

```
$ drush cex
Collection  Config
system.site  update
The .yml files in your export
directory (config/sync) will
be deleted and replaced with
the active config. (y/n): y
```

Then

```
$ git commit && git push
```

The reference use case: step 4 of 6

Stage Configuration

Production

```
$ git pull
```

Development

- Development goes on normally.

The reference use case: step 5 of 6

Review Changes

Production

```
$ drush config-import \
        --preview=diff
...
uuid: d1329a45-acab-4599...
-name: 'localhost'
+name: 'Drupal 8'
mail: info@example.com
slogan: ''
page:
Import changes? (y/n):
```

Development

- Development goes on normally.

The reference use case: step 6 of 6

Apply Changes

Production

```
$ drush cim
Collection  Config
system.site  update
Import changes? (y/n): y
```

Development

- Development goes on normally.

Production deployment

- Pull code and configuration from master branch.
- Run updates: `$ drush updatedb`
- Import configuration: `$ drush cim`
- **Important**: the order must be respected. Issue #2628144 will enforce it.

Problem solved?

- Configuration Management works perfectly for its use case.
- But the reference use case scenario is very narrow.
- In real life we need to cover many more scenarios.
- ...And weren't we supposed to get rid of database dumps, by the way?

Chapter 2

Can I install a site from existing configuration?

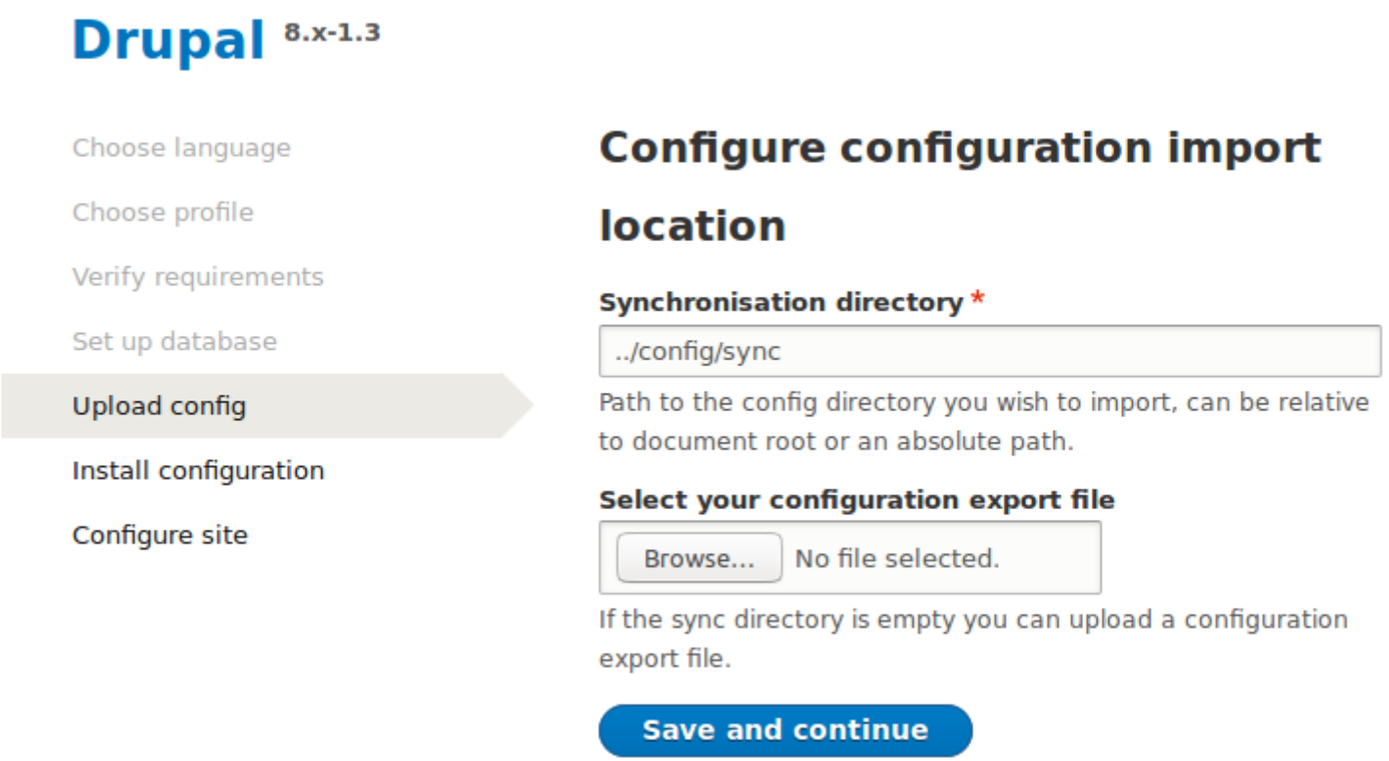
Bootstrapping production

- Deployment is nice but how do get production up and running for the first time?
- Did you say database dump?

Configuration Installer

- Usually running the installer creates a "new site".
- The Configuration Installer is an installation profile that takes over the Drupal installer and allows sites to be created from existing configuration.
- It is an installation profile and needs to be put in `/profiles` in order to work.
- Should be on every site (and in core)

Configuration Installer UI



Configuration Installer in core

Allow a site to be installed from existing configuration:

https://www.drupal.org/node/1613424

Chapter 3

Can I override local configuration?

- Can I have verbose error logging enabled on the development copy only?
- Can I customize API keys in development without committing them?

Overriding

- In development, it is convenient to have a different configuration than on the production site.
- Examples: different error reporting, different API keys for services, different site name or site mail.
- These customizations are **not** to be exported.
- Not covered by the reference use case.

Using `$config`

The `$config` array allows run-time overriding: configuration is still there, but it gets overridden.

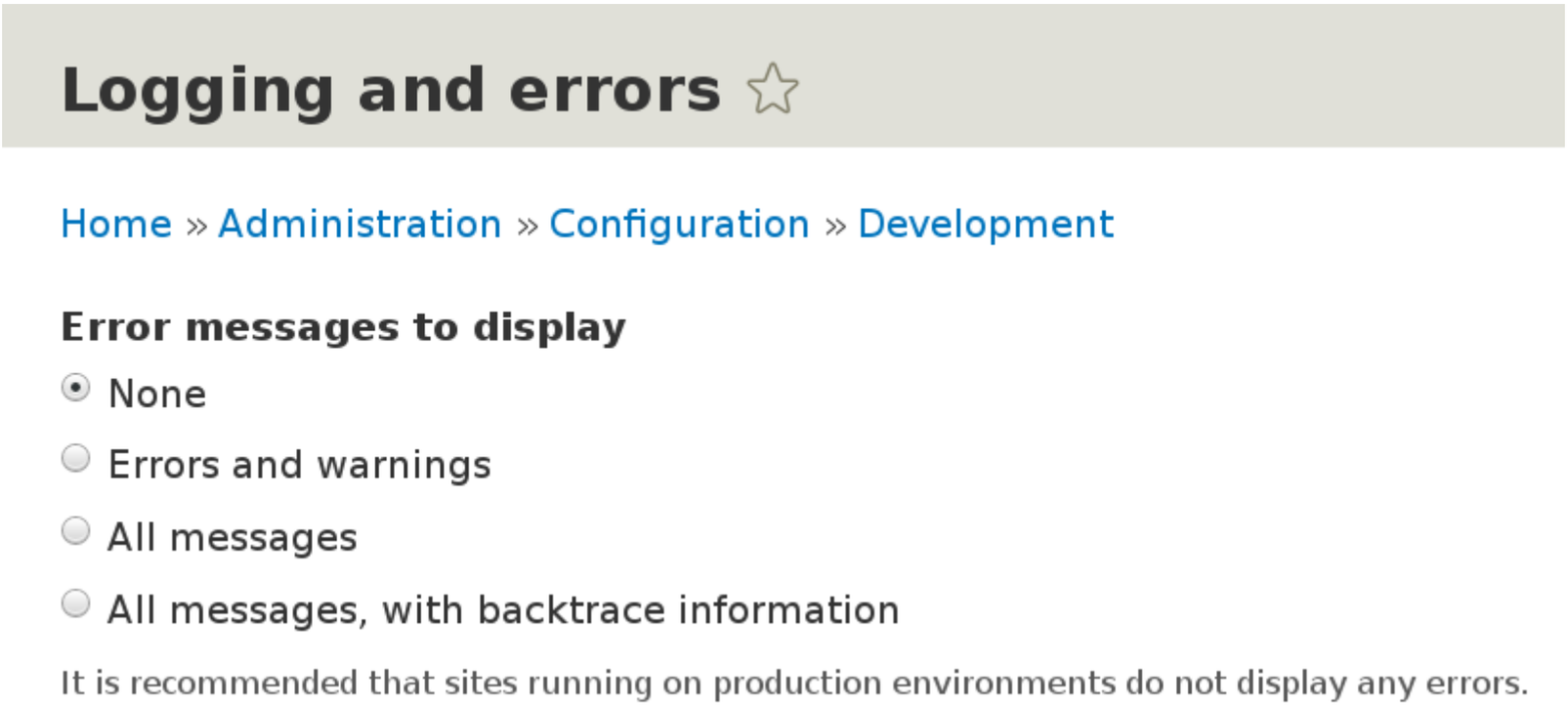
Example: add to `settings.php` (or `settings.local.php`) in the development enviroment:

```
$config['system.logging']['error_level'] = 'verbose';
```

This enables verbose error logging on that instance.

`$config` and editing

Even after the override `admin/config/development/logging` shows the original (non-overridden) settings. **This is wanted:** Drupal prevents you from submitting overridden configuration.



`$config` and exporting

- Even after the override, `$ drush cex` exports the original settings.
- This is wanted too: Drupal will not let the override slip into the export.

Mutable and immutable configuration

Drupal can retrieve configuration as mutable or immutable.

Immutable

Retrieved in read-only mode (to apply/display; overrides are considered)

```
\Drupal::config('system.site');
```

Mutable

Retrieved in read-write mode (to set/export values; overrides are ignored)

```
\Drupal::configFactory()
  ->getEditable('system.site');
```

Finding keys and values: mapping

Keys/values for `$config` are mapped from the YAML files.

```
--- a/sync/system.logging.yml
+++ b/sync/system.logging.yml
@@ -1,3 +1,3 @@
-error_level: hide
+error_level: verbose
 _core:
   default_config_hash: u3-njszl92FaxjrCMiq0yDcjAfcdx72
```

becomes:

```
$config['system.logging']['error_level'] = 'verbose';
```

A satisfactory solution?

- `$config` covers our need for differentiating configuration between environments but...
- You can only alter existing configuration.
- You can't add new configuration using `$config`
- You can't completely "unset" existing configuration using `$config`
- You can't override which modules are installed.
- You can't override the color of Bartik and other details.

Can I exclude modules from getting deployed?

- Can I have development modules enabled on a development environment but not deploy them to the production site?

Method: Drush

Drush can ignore the enabled states of extensions when synchronizing configuration.

```
drush cex --skip-modules=devel
drush cim --skip-modules=devel
```

Does still export configuration depending on devel.

This may result in inconsistent configuration being exported!

Method: `drush_cmi_tools`

https://github.com/previousnext/drush_cmi_tools

```
drush cexy --ignore-list=/path/to/config-ignore.yml
drush cimy --delete-list=/path/to/config-delete.yml
```

Operates with a list of configuration to ignore.

Method: Configuration split

- Configuration override not at runtime but at import/export time.
- Split off some configuration to dedicated folder
- Blacklist configuration
- Ignore set of configuration
- Configured by configuration entities

```
drush config-split-export
drush config-split-import
```

```
drupal config_split:export
drupal config_split:import
```

Method: Configuration split

Edit *Development config* ☆

Home » Administration » Configuration » Development » Synchronize » Configuration Split Setting

Label *

Development config

Machine name: development

Label for the Configuration Split Setting.

Folder

../config/dev

The folder, relative to the Drupal root, to which to save the filtered config.
Configuration related to the "filtered" items below will be split from the main configuration and exported to this folder by `drupal config_split:export`.

Modules

Devel ×

Select modules to filter.

Blacklist

system.menu.devel ×
views.view.development_view ×

Select configuration to filter.

Graylist

automated_cron.settings ×

Select configuration to ignore.

Weight

0

The weight to order the splits.

Save Delete

Chapter 5

Can I work in parallel with a colleague?

- Can two or more developers work simultaneously on the same project?
- How do I ensure that my work is not lost?
- Can I assume that Git will always do the right thing when merging?

Git to the rescue

- Configuration Management is based on having **two** instances of the same site (development and production).
- Multiple instances are not considered.
- On the other hand, configuration is exported to text files.
- And for text files we have Git!

Working as a

Team of developers

- Share a Git repository for both code and configuration.
- Install site starting from initial configuration.
- Adopt “A successful Git branching model” (cit.)

Project bootstrap

First developer:

- Installs site locally.
- Exports configuration to sync
- Commits and pushes to shared Git repository.

Other developers (and prod):

- Clone code.
- Have `config_installer` profile available.
- Install site starting from exported configuration.

Parallel development

First developer:

- Own branch:
  `checkout -b feature-a`
- (code, code, code...)
- Commits and pushes to shared Git repository.

Other developer(s):

- Own branch:
  `checkout -b feature-b`
- (code, code, code...)
- Commit and push to shared Git repository.

...but careless merge is dangerous and problematic.

Collaboration issues

A careless workflow may result in:

- Losing all uncommitted work.
- Accidentally overwrite work by others.
- A configuration that looks OK at first sight but that is actually invalid for Drupal.

The safe sequence

1. Export configuration: `drush cex`
2. Commit
3. Merge: `git pull`
4. Import configuration: `drush cim`
5. Push: `git push`

If you do it wrong...

- Import before Export: Deletes your work, no backup.
- Merge before Export: Export deletes previous work, solved by git.
- Merge before Commit: Manual labour on conflicts.
- Forgotten Import: Next export will not contain merged config, more difficult to solve in git.

Breaking configuration with Git

- Setup: Installed `standard` profile
- Developer A on branch `feature-a` deletes Tags from 'Article'.
  - Resulting configuration change: 2 files are removed (field instance and field storage)
- Developer B on branch `feature-b` adds Tags to 'Basic page'.
  - Resulting configuration change: 1 file is added (field instance)
- Git will happily merge `feature-a` and `feature-b` into `develop`
- The resulting configuration is invalid:
  - Tags has a field instance but no storage.

Takeaway: when merging check that the configuration is still valid by importing it.

Chapter 6

Can I package configuration and re-use it?

Features for Drupal 8

- The Features module: a configuration packager.
- Entirely new for Drupal 8, to take advantage of Configuration Management.
- Offers automatic packaging: Features analyzes your site and automatically packages up site configuration into a set of features.

Features for Drupal 8

- Focuses on packaging configuration for reuse purpose only.
- Is meant to be a development module: generated features do not depend on the Features module.
- The resulting features are modules: just enable to activate their configuration.

Using Features

- Enable the Features module - only in development, not needed in production.
- Enable the dedicated Features UI module too.
- Generate the Features at `admin/config/development/features`
- Enable the Features modules you generated; these are portable between sites.

Using Features: Example

Blog	blog	Provides Blog content type and related configuration. Included configuration	Not exported

Content type	Blog
Dependencies	Field, Menu UI, Node, Path, Text, User
Entity form display	node.blog.default
Entity view display	node.blog.default node.blog.teaser
Field	Body, Topic
Field storage	node.field_topic

Features workflow

- If you use Features 8.x for deployment: **You are doing it wrong.**™
- Re-use *partial* configuration between **different** sites.
- Use Features in development environments only.

Features TODO list

Features 8.x is now in beta. Major gaps left to fill:

- Overrides are not supported properly: update of a customized distribution are still problematic
- Menu links are not exportable.
- Permissions are not exportable (currently removed).

More information at:

https://www.drupal.org/docs/8/modules/features/gaps

Chapter 7

Can I handle a client messing with production configuration?

Changes on production

Imagine the ideal situation:

- Configuration is correctly exported, versioned and deployed
- Development team adopts a solid GIT branching model

BUT...

Configuration on production is changed
by your **Geeky Client**™ overnight, without notice.

Option 1

Lock configuration on production

Don't allow config changes on the production site if ever possible by installing the `config_readonly` module.

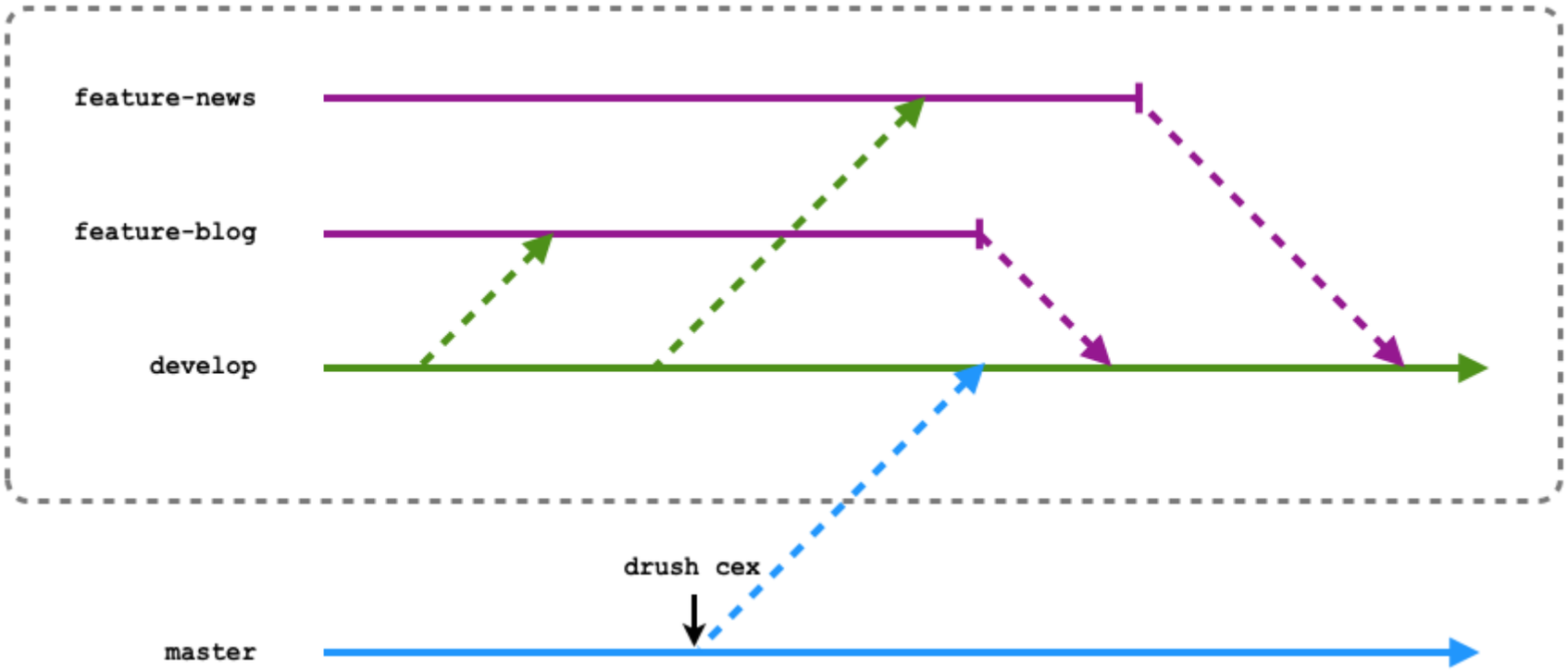
Note: add this to `settings.php` in production:

```
$settings['config_readonly'] = TRUE;
```

Option 2

Export to a dedicated branch

Have a person responsible for merging prod and dev configuration.



Option 3

Configuration split

- Review changes done by the client on production and choose what to keep
- Export production changes via `drush config-split-export` to `../config/client`
- Pull new configuration: business as usual
- Import previous configuration changes via `drush config-split-import`
- Configuration is imported from both `../config/sync` and `../config/client`

Chapter 8

Can I deploy the content my configuration depends upon?

- What about a view that depends on a taxonomy term?
- Can I have default content on my site?

Deploying content

- **Scenario**: Deploying configuration that depends on content such as nodes, blocks, taxonomy terms, etc.
- **Problem**: Content is not exported along with configuration.

Option 1

Handle missing content event

- When content required by configuration is missing Drupal fires a `ConfigEvents::IMPORT_MISSING_CONTENT` event.
- You could subscribe to that event via an event subscriber and, for example, perform a run-time migration using `Migrate` (now in core).
- If nothing happens Drupal will handle it for you on `\Drupal\Core\Config\Importer\FinalMissingContentSubscriber` which will just remove content dependencies, run-time.

Option 2

Use `default_content` module

- Any module that requires default content can provide hal+json versions of the entities inside `{module_name}/content/{entity_type}` folders.
- Entity references are respected by Drupal 8 core thanks to the HAL core module and content entities having `UUID` by default.
- Content is imported when the module is installed.

Option 2

Use `default_content` module

- content
  - block_content
  - menu_link_content
    - menu_link_content
      - 9adf04b3-1e43-4197-a141-657b3ab69f2b.json
      - 34ed8934-6149-4bb8-a150-80fcfe3f6402.json
      - 1749fb62-8b74-4b1f-9f00-c55b3c3de11f.json
      - b741cb00-86d0-4ba3-a000-b8ddf934b1a3.json
  - node
    - organisation
      - 0bf245d8-6e52-4e99-b530-649ea864c53b.json
      - 0c2acb8b-ace5-439a-b921-9299837f8d41.json
      - 0cbbfe21-97f9-4b1d-9500-d0be6098cb3e.json
      - 0d5df9ed-4c08-42e8-880f-156768603b2a.json
      - 0de59d17-8368-4a10-8707-97e9bf402fb7.json
      - 1be7101b-808b-4c33-9400-ff8a8d22c4dd.json
      - 1cfbc12a-da90-4b2b-92b8-db73dc6ca577.json
      - 1f69ce1b-56d1-4026-ad57-8706cb68836d.json

```
25      },
26      "uuid": [
27          {
28              "value": "0bf245d8-6e52-4e99-b530-649ea864c53b"
29          }
30      ],
31      "type": [
32          {
33              "target_id": "organisation"
34          }
35      ],
36      "langcode": [
37          {
38              "value": "en",
39              "lang": "en"
40          }
41      ],
42      "title": [
43          {
44              "value": "Drupal Association",
45              "lang": "en"
46          }
47      ],
```

Option 3

Use `deploy`

From the module's page at https://www.drupal.org/project/deploy

```
The Deploy module is designed to allow users
to easily stage and preview content for a Drupal site.
```

Working with content updates

In Drupal 8 there is a post update hook specifically designed to work with content.

```
/**
 * Executes an update which is intended to update data, like enti
 *
 * These updates are executed after all hook_update_N() implementa
 * At this stage Drupal is already fully repaired so you can use a
 * as you wish.
 */
function hook_post_update_NAME(&$sandbox) { }
```

State of the art

Can I deploy configuration?	☆☆☆☆☆
Can I install a site from existing configuration?	☆☆☆☆
Can I override local configuration?	☆☆☆☆
Can I exclude modules from getting deployed?	☆☆☆☆
Can I work in parallel with a colleague?	☆☆☆☆☆
Can I package configuration and re-use it?	☆☆☆
Can I handle a client messing with production configuration?	☆☆☆☆
Can I deploy the content my configuration depends upon?	☆☆☆☆

Related contrib projects

- `config_installer` install site from existing config.
- `config_readonly` locks any configuration changes via UI.
- `config_update` report changes between original and active config of a module.
- `config_devel` module helps with developing configuration.
- `config_split` import time config override.
- `config_tools` automatically commit config changes to git.
- `features` bundle config for re-use on different site, ideal for distributions.
- `config_sync`: provides methods for safely importing site configuration from updated modules and themes.


DUBLIN
DRUPALCON
WHAT DID YOU THINK?
Evaluate This Session
events.drupal.org/dublin2016/schedule
THANK YOU!