Advanced Configuration Management In Drupal 8
Software Architect

Mike Potter
Email: mpotter@phase2technology.com
Drupal.org: mpotter

- Primary maintainer of Features, Features Override, Config Actions modules
- Architect of Open Atrium 2 distribution
REVIEW OF
CONFIG MANAGEMENT
Config vs Content

Drupal Database

- Pages
- Articles
- Comments
- Users
Config vs Content

Drupal Database

- Pages
- Articles
- Comments
- Users
- Content Types
- Fields
- Views
- Site Info

CONTENT
Config vs Content

Drupal Database

CONFIG
- Site Info
- Views
- Fields
- Content Types

CONTENT
- Users
- Comments
- Articles
- Pages
The Dilemma...

**CONFIG**

- Config comes from Developers

  Dev → Staging/QA → Production

**CONTENT**

- Content comes from Production Users

  Dev ← Staging/QA ← Production
The Old D7 way

Drupal Database

CONFIG

CONTENT

Features

Site Info
Views
Fields
Content Types
Users
Comments
Articles
Pages

Code

CONTENTS

Features

Drupal Database

CONFIG

CONTENT

Features

Site Info
Views
Fields
Content Types
Users
Comments
Articles
Pages

Code
The New D8 way

Drupal Database

CONFIG
- Site Info
- Views
- Fields
- Content Types

CONTENT
- Users
- Comments
- Articles
- Pages

config export

config/sync config.yml files
CORE D8 CONFIG WORKFLOW
D8 Normal Deployment

Export config

Import config

config/sync directory
D8 Normal Deployment

Dev
Staging/QA
Production

Export config
Import config

config/sync directory
D8 Core Config Deployment

- Modify location of config sync in settings.php
  
  ```
  $config_directories['sync'] = 'config/default';
  ```

- Ensure config/default is added to your git repo

- On **DEV**: Full Export config
  `drush config-export`

- On **DEV**: Review Changes, Commit and Push to your develop branch

- On **QA**: Pull develop, Review and import config
  `drush config-import`
Headaches...

• Can only import into the same SITE UUID
  - Clean install creates new random uuid

• Config items have their own UUID
  - Creating new config creates random uuid

• config-import can delete config
  - will complain if content exists
  - e.g. Shortcut links in standard profile
USEFUL DRUSH COMMANDS
Export and Import

- **drush config-export <label>**
  - Has several options for add, commit, push
  - Probably better to just use git commands.
  - Label is optional name of config location (defaults to "default")

- **drush config-import <label>**
  - **--preview=diff** to see changes
  - **--partial ??**
What about "config-import --partial"

- Need drush 8.1.0+
- Will not delete config in DB that is missing from config/sync folder
- Be careful after doing a full config-export that any unwanted config doesn't get added to git repo.

**DO NOT USE "git add -A"!!!!!!**

- MUST carefully review changes from config-export

NOT BEST PRACTICE
Getting and Setting config

- `drush config-get <config-id> <key>`
  - Show active config from DB
  - `--include-overridden` to show overrides.

- `drush config-set <config-id> <key> <value>`
  - Set a new value for config item

- `drush config-edit <config-id>`
  - Edit active config item using editor (vi or other)
Other commands

- drush config-list <prefix>
  - List config names starting with prefix

- drush config-delete <config-id>
  - Delete config from active DB
  - Be CAREFUL. Can break site.
INSTALLING WITH CONFIG
One-time Clean Site Install

- Install site with profile normally
  - drush site-install profile-name
- Confirm install_profile and config/sync in settings.php
- drush config-export
- git add config/sync
- git commit and push config to repo
- Now, How to install site using this config?
Creating a custom profile and using a core 8.3.x patch (below):

- `drush site-install profile-name`

Either put config in profile/config/config/sync or, specify location in profile.info:

`config_install: config/sync`

Will set the site uuid and config uuids the **same** on every site using the profile.

[https://www.drupal.org/node/2788777](https://www.drupal.org/node/2788777)
Installing site from config

Using Drush `si --config-dir`

- OR,
  `drush site-install profile-name --config-dir=config/default`

- Could have an issue with entity schema changes after cron
  
  [https://github.com/acquia/lightning/issues/387](https://github.com/acquia/lightning/issues/387)

  So, avoid creating config in installer!

[https://www.drupal.org/node/1613424](https://www.drupal.org/node/1613424)
Installing site from config

What about config_installer

- Profile to install site from config/sync
  - Prompts for tar.gz or config/sync directory in installer UI.
  - drush site-install config-installer
  - Doesn't seem to work with other profiles, such as Lightning
  - Hasn't been updated in February, maybe in favor of some similar core patches

NOT BEST PRACTICE
Setting the Site ID

- Look in config/default/system.site.yml

  ```yaml
  uuid: 0e1861a0-8e9a-469b-97d7-33317d99d785
  name: My Site Name
  mail: admin@example.com
  ...
  ```

- Copy the uuid value to clipboard

- Use drush config-set to set the site ID in the database:

  ```bash
  # drush config-set system.site uuid <paste-uuid-here>
  Do you want to update uuid key in system.site config? (y/n): y
  ```
Running UPDATES

- Run updates **BEFORE** import
  - git pull
  - composer install
  - drush updb
  - drush cim

- Will be enforced in core

https://www.drupal.org/node/2628144
MULTIPLE ENVIRONMENTS
• Different environments need different configuration
  - local
  - stage
  - prod

• Use **Config Split** module
  - splits different config into different config/sync directories
  - config-import merges directories
Config Split

- Must create sync directory for each environment
- Create splits in Config Split UI
  - Disable each split by default
- Export the full config
- Enable the Split Environment to use in settings.php (based on ENV var)
- `config-import` will merge the sync
- To export the split, use `drush csex` instead of `drush cex`

https://blog.liip.ch/archive/2017/04/07/advanced-drupal-8-cmi-workflows.html
Deployment with Config Split

Diagram:

- config-import
- config/default directory
- config/local directory
- Dev
- Staging/QA
- Production
Deployment with Config Split

config/export

config/default
directory

config/local
directory

Dev

Staging/QA

Production
Deployment with Config Split

config/default directory

config-import

Dev

config/local directory

Staging/QA

Production

config/stage directory
Deployment with Config Split

- **Dev**
- **Staging/QA**
- **Production**

- **config/default** directory
- **config-export**

- **config/local** directory
- **config/stage** directory
Deployment with Config Split

- **config/default** directory
- **config-import**
- **dev** directory
- **config/local** directory
- **config/stage** directory
- **config/prod** directory
- Staging/QA
- Production

- **filestack**
  - Dependencies:
    - **config**:
      - field.storage.node.body
      - node.type.page
    - **module**:
      - text
  - **id**:
    - node.page.body
  - **field_name**:
    - body
  - **entity_type**:
    - node
  - **bundle**:
    - page
  - **label**:
    - Body
  - **description**:
    - 
  - **required**:
    - false
  - **translatable**:
    - true
  - **default_value**:
    - 
  - **default_value_callback**:
    - 
  - **settings**:
    - display_summary: true
    - field_type: text_with_summary
Deployment with Config Split

- **config/default directory**
  - Dev
  - config/local directory
  - config/development directory
  - config/stage directory
  - config/production directory
  - Production

- Staging/QA
- config/stage directory

- Production
- config/prod directory
OVERRIDDING CONFIG
When do Overrides Apply?

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

- Returns "Immutable" READ-ONLY config.
- Can be overridden.

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

- Returns "Mutable" READ/WRITE config.
- Overrides are ignored.
"Runtime" Overrides

- Can override $config[] in settings.php
  - Cannot add new config, cannot delete config.

- **Config Override module** supports "runtime" overrides in custom modules
  - Extends core ConfigFactoryOverrideInterface

- Overrides only affect Immutable config (doesn't affect import/export)

- UX challenge of editing config that is overridden
FEATURES ??
What About Features?

- Features was designed to **Package configuration into modules**.

- Why?
  - Reuse functionality across sites
  - Organize your config
  - In D7 it was the only way
But Features has Problems

- Tricky to figure out how to split up configuration
  - New assignment plugins tried to help, but are complex
- Features headaches with "unmet dependencies"
- Installing a Features export module will create new config uuids
  - causes issues with config-sync
- Features aren't actually reusable
STOP USING FEATURES!

- Core D8 Configuration already works!
  - Puts config into YAML files
  - Allows normal version control
  - Captures ALL the config
  - Clean export/import workflow

- Config Split is handling environment-specific config
- Do you really need config organization?
TEMPLATES & ACTIONS
Re-usable Config

- Features were supposed to be re-usable
- But they are NOT!
  - Config contains machine names

```
field.field.node.project_blog.project_image.yml

langcode: en
status: true
dependencies:
  config:
    - field.storage.node.project_image
    - node.type.project_blog
module:
  - text
id: node.project_blog.project_image
field_name: project_image
entity_type: node
bundle: project_blog
label: 'Image Field'
description: ''
required: false
translatable: true
default_value: {}
default_value_callback: ''
settings:
  display_summary: true
field_type: text_with_summary
```
Templates

- How could we make a feature reusable?
- What if we created a "config template"

```yaml
field_template.yml
langcode: en
status: true
dependencies:
  config:
    - field.storage.node.@field_name@
    - node.type.@bundle@
  module:
    - text
id: node.@bundle@.@field_name@
field_name: @field_name@
entity_type: node
bundle: @bundle@
label: 'Image Field'
description: ''
required: false
translatable: true
default_value: {}
default_value_callback: ''
settings:
  display_summary: true
field_type: text_with_summary
```
Then created a way to "use" the template

source: field_template.yml

@bundle@: "project_blog"
@field_name@: "project_image"

dest: field.field.node.@bundle@.@field_name@
Overrides

• Then override some values:

```yaml
source: field_template.yml
@bundle@: "project_blog"
@field_name@: "project_image"
dest: field.field.node.@bundle@.@field_name@

value:
  label: "My Project Image Field"
  description: "Project-specific description"
  required: true
```
• drupal.org/project/config_actions

• Pluggable framework for manipulating configuration:
  - variable replacements
  - change values
  - add values
  - delete config
  - include templates
  - read/write yml files or config entities

• http://config-actions.readthedocs.io/

• Beta, release planned soon
Config Templates Module

- Uses Config Actions
- Provide common config templates
- Plugin for Features to export templates and overrides from existing config
- Can use the same Features workflow we are used to, without the downsides
- Still in active development (no release yet)
When should you use templates?

- Reuseable functionality
  - Code packaged with its config

- Manage project-specific overrides

- Master template for project config
  - Create multiple content-types from single common project template
  - Field templates with project settings

- Define Content Model in a single module
  - Actions are the "content schema"
  - Build automated tests from schema
  - Easily update IA as model changes
Summary

1. Stop using Features in D8
2. Use "drush config-export"
3. Use "drush config-import"
4. Be aware of --partial option
5. Core patch for installing config via profile
6. Manage changes via git
7. Use Config Split for environment-specific config
8. Try Config Actions and Config Templates for reusable functionality instead of Features