Advanced Configuration Management with Config Split et al.

Fabian Bircher fabian@nuvole.org

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

Our Distributed Team

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

Italy

Belgium

Czech Republic

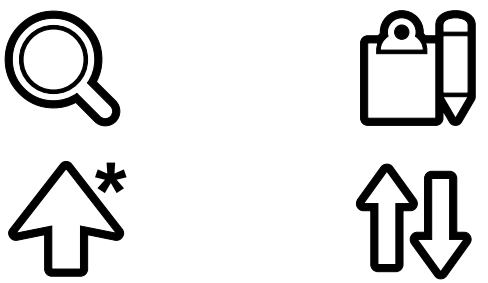
Our Clients

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

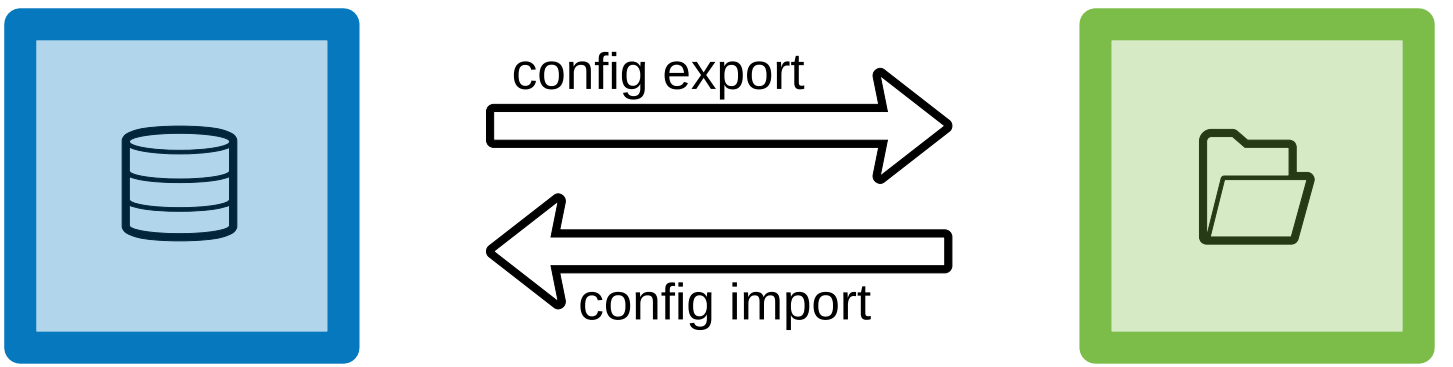
CM in core

- 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?

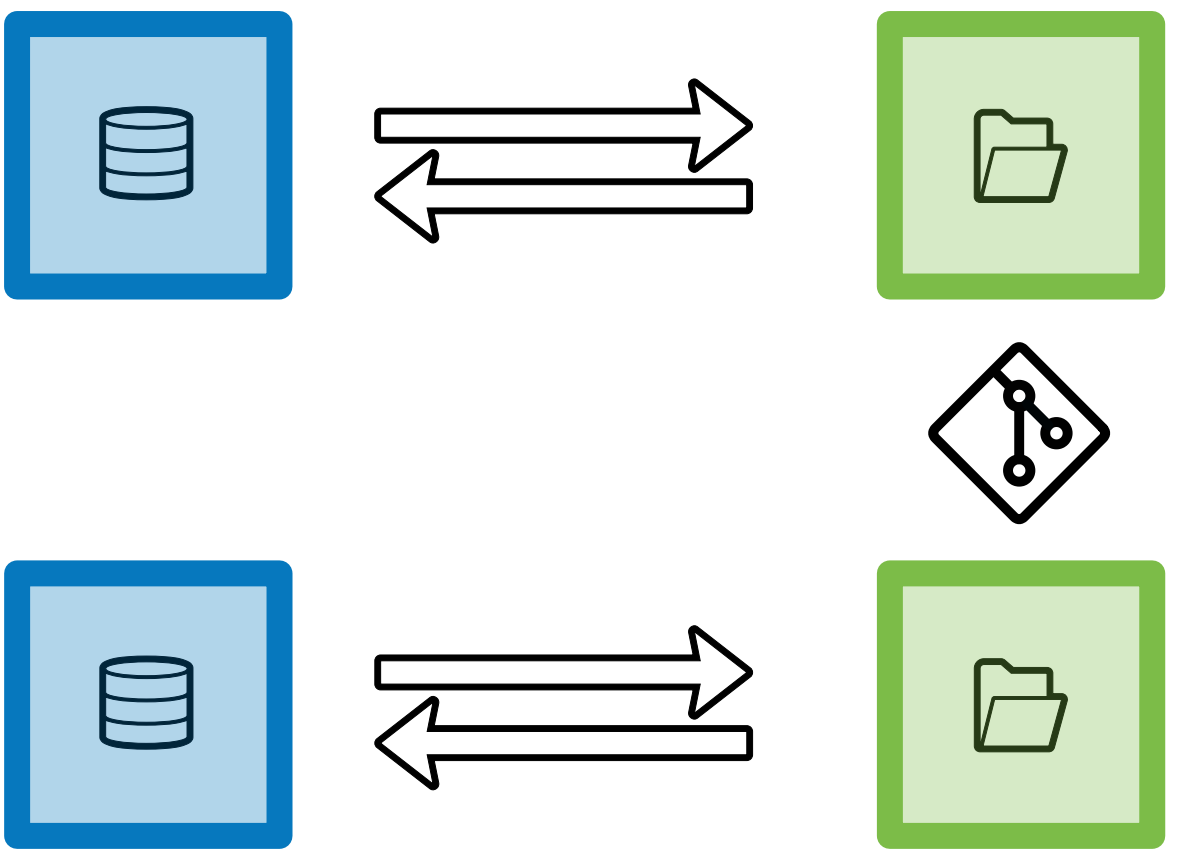
Active Configuration Storage



Synchronise Configuration



Deploy Configuration



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.

Chapter 2

Install a site from existing configuration

Bootstrapping production

- Deployment is nice but how do get production up and running for the first time?

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

Allow a profile to be installed from existing config:

https://www.drupal.org/node/2788777

Chapter 3

local configuration override

- Can I have verbose error logging enabled on the development copy only?
- Can I customize API keys in production 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`

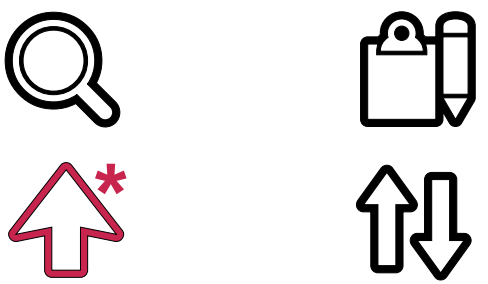
The `$config` array allows run-time overriding: configuration is still there, but it gets overridden.

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 Override



A satisfactory solution?

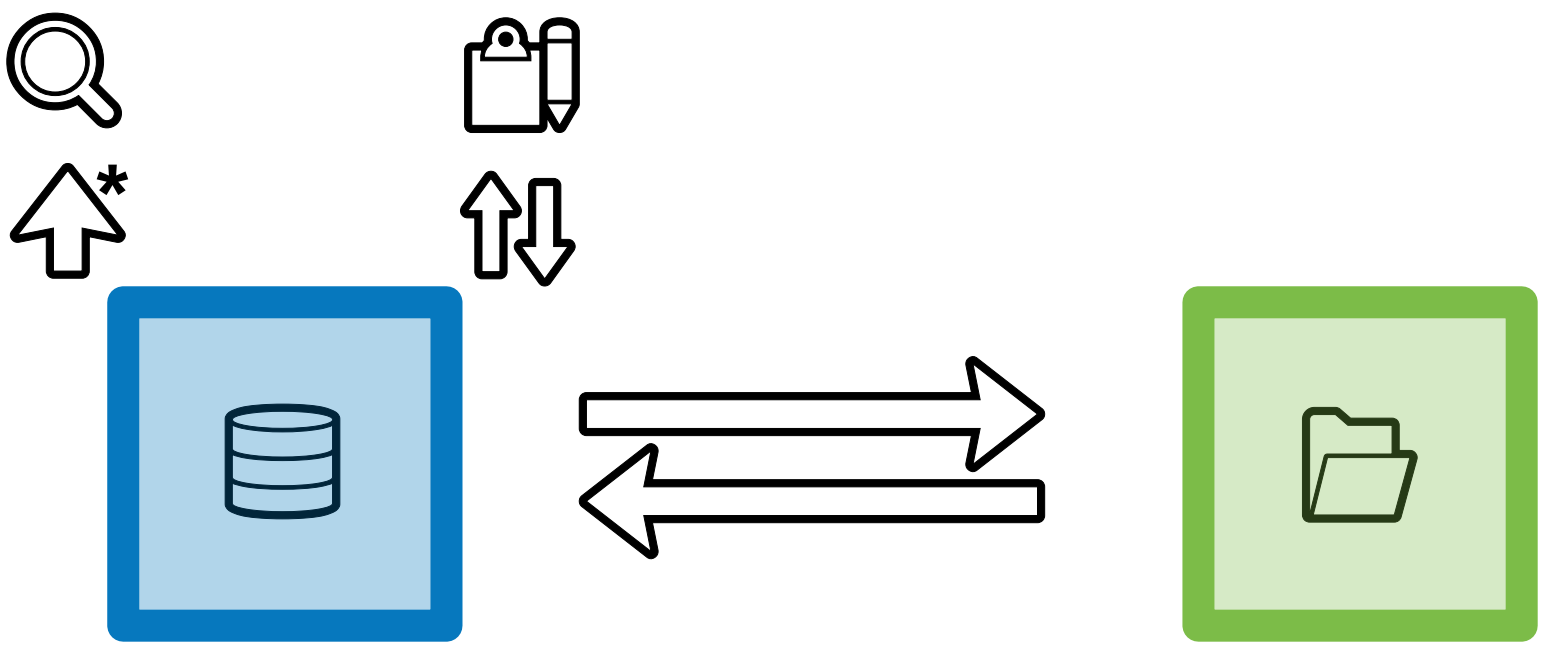
- `$config` covers our need for differentiating configuration between environments but...
- You can only alter existing configuration.
- You can't add new configuration using `$config`
- You can't completely "unset" existing configuration using `$config`
- You can't override which modules are installed.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

Chapter 4

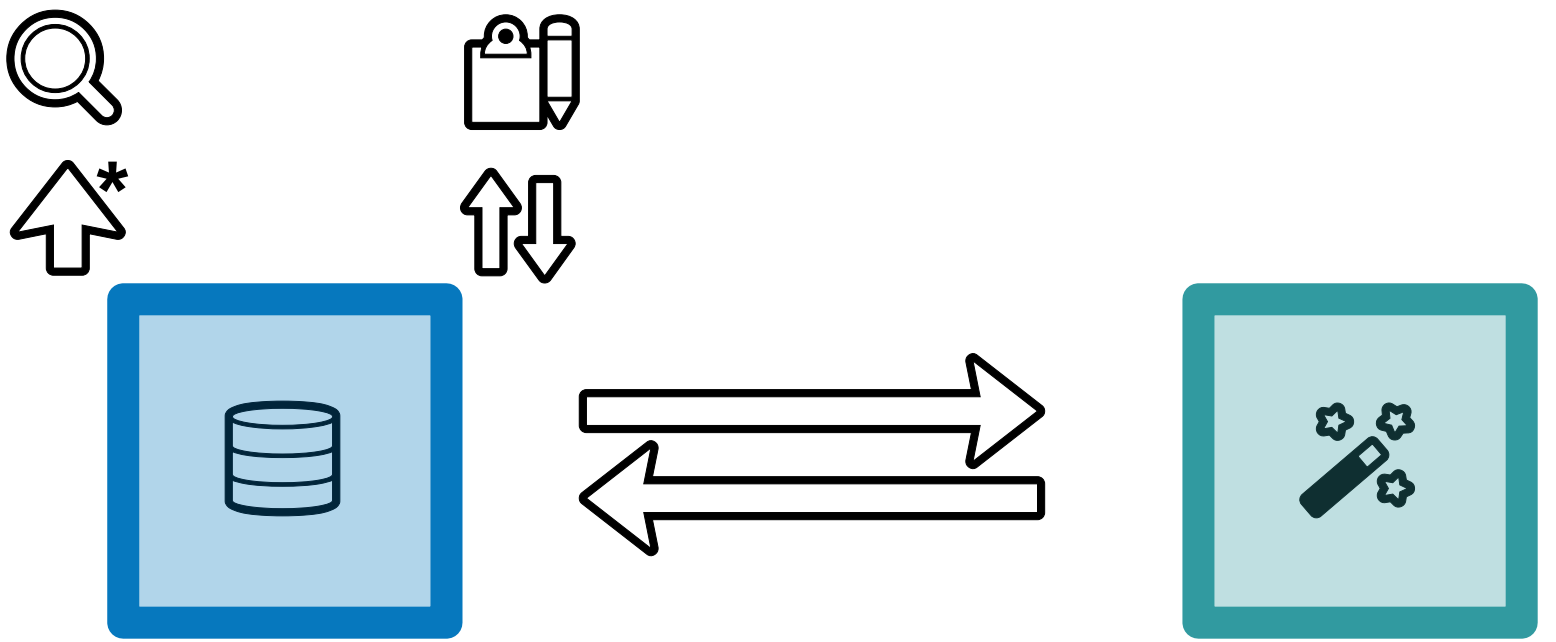
Config Filter

- How can we do more than configuration overrides?

CM in Core



with Config Filter

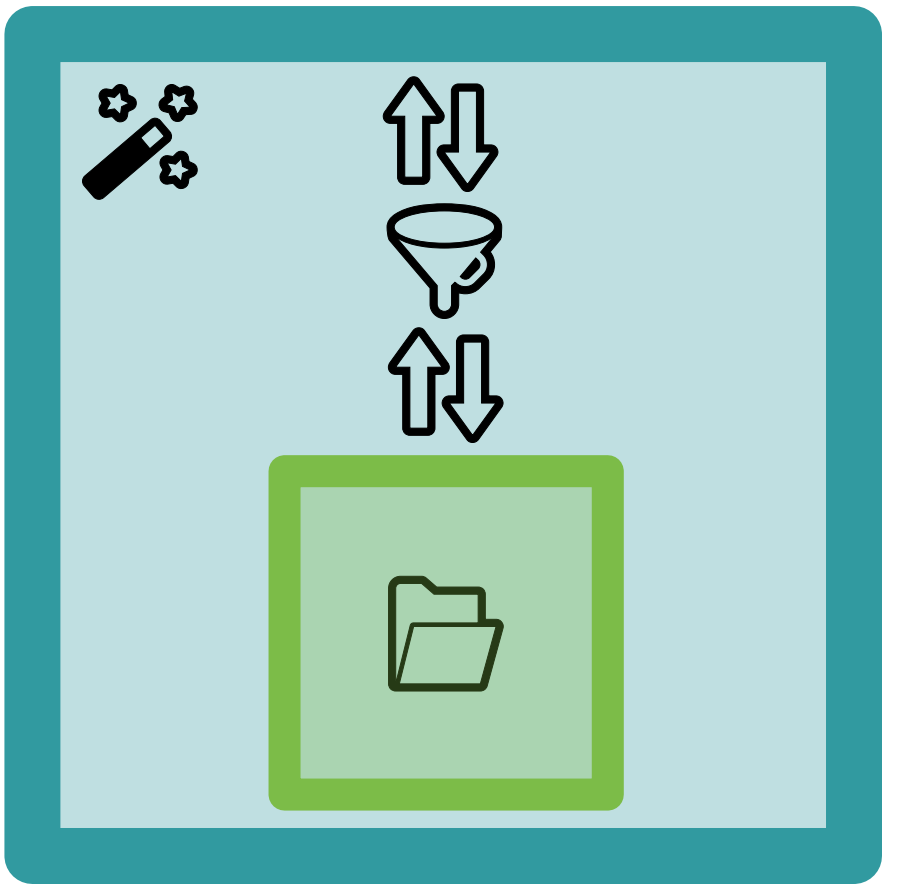


Configuration Filters

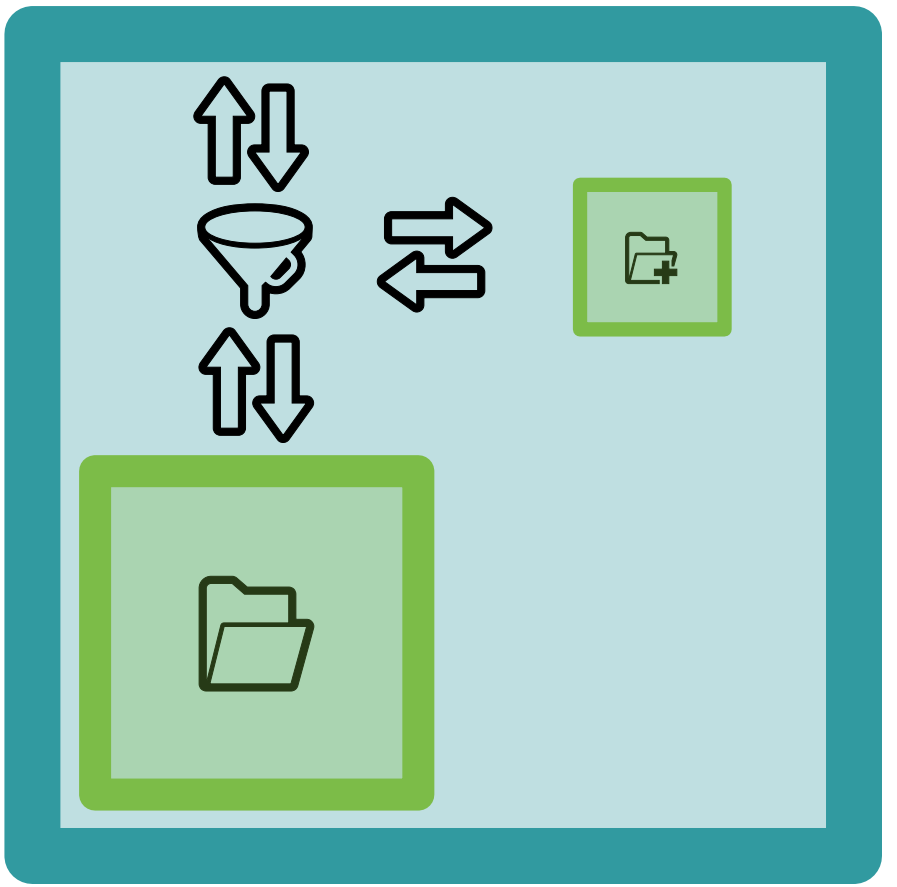
- Filters can modify the data for every operation.
- Filters are plugins
- Plugins are sorted by weight and applied one after the other
- Plugins can be inactive and skipped

6k+ installs, top 100 modules, 0* bugs

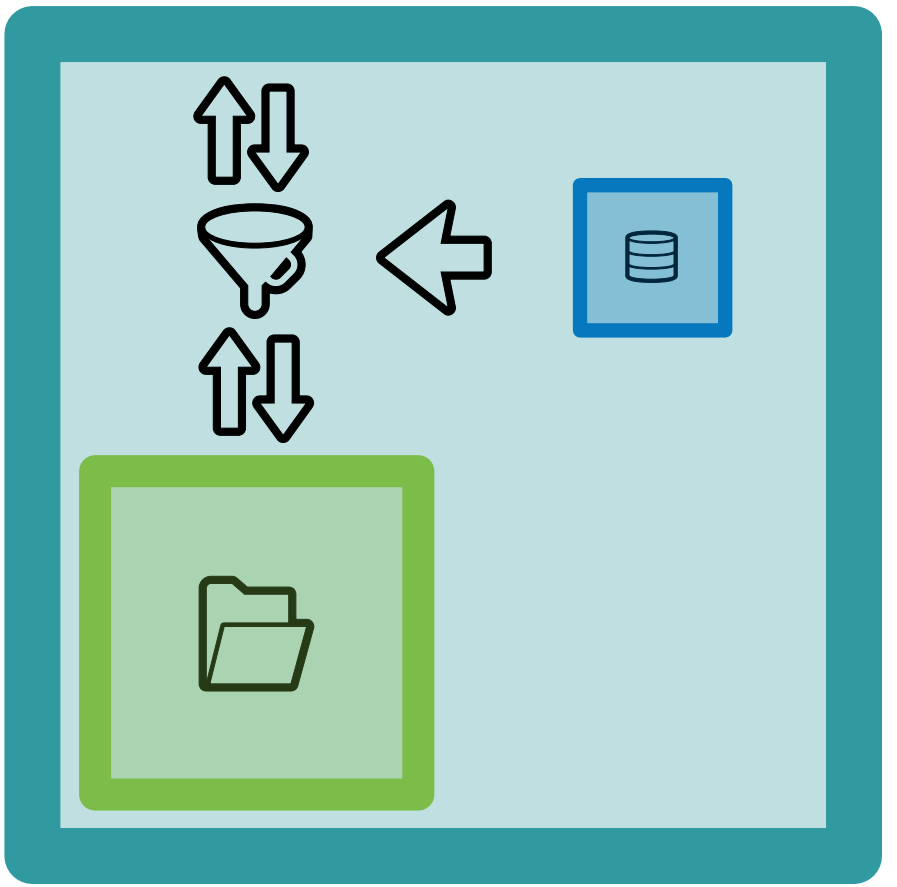
Config Filter



Config Split



Config Ignore



Chapter 5

Config Split configuration

- What do the different configuration options do?
- What is the difference between a complete split and a conditional split?

Static settings

- Folder: Path to the secondary config storage
- Weight: Determines the order in daisy-chained filters
- Active: To use the split or not to use the split.

Complete Split (blacklist)

- Modules: will be removed from core.extensions when exporting
- Config items: automatically includes configuration which depends on modules
- Additional config: text area for use with * wildcards

Conditional Split (graylist)

- Config items: select the configuration

  will not be deleted on export
- Dependent config: add config that depends on the listed ones
- split when different: useful when using wildcards

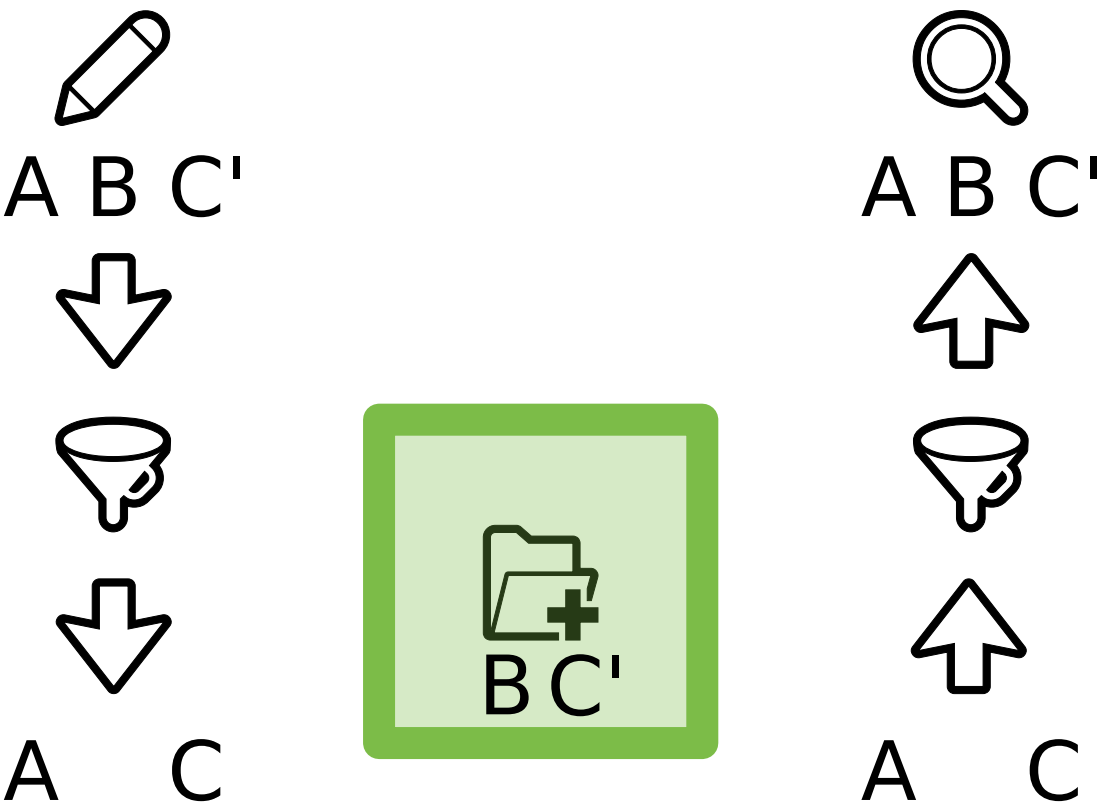
CLI commands

- `csim/csex`
- without argument: replacement for drush < 8.1.10 and console
- with split machine name: import/export only that specific split

Example

- Not listed: **A**
- Complete Split: **B**
- Conditional Split: **C**

Example



Chapter 6

Environment specific modules/config

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

Configuration split

- List modules to split off
- Add environment specific configuration
- Override per environment to make split active

```
$config['config_split.config_split.dev']['status'] = TRUE;
```

Environment specific permissions

- Use Config Role Split
- Config Filter Plugin
- Add/remove permissions during import/export
- Role Split can be overwritten split or ignored per environment

Chapter 7

Configuration Management with git

- 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 designed to share configuration between different environments.
- 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:

- Initialise repository.
- 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 for sharing

1. Export configuration: `drush cex`
2. Commit: `git add && git commit`
3. Merge: `git pull`
4. Update dependencies: `composer install`
5. Run updates: `drush updb`
6. Import configuration: `drush cim`
7. 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.
- No `updb` or after `cim`, will be disallowed, database might be broken.
- No composer install, may not have all the updated code.
- Merge before Commit: Manual labour on conflicts.
- Forgotten Import: Next export will not contain merged config, more difficult to solve in git.

The safe sequence for updating

1. Update code: `composer update`
2. Run updates: `drush updb`
3. Export updated config: `drush cex`
4. Commit: `git add && git commit`
5. Push: `git push`

Update DB before config import

update hooks are for fixing the database. See #2762235

New with proof of concept module Config Import N:

```
function hook_pre_config_import_NAME(&$sandbox) { }
function hook_post_config_import_NAME(&$sandbox) { }
```

Or in core: #2901418

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.

Chapter 8

Config changes on production

- How to deal with changes to configuration on the production site?

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

Configuration Split

- Review changes done by the client on production and agree what to keep ➜ conditional split.
- Export production changes via `drush config-split-export client` to `../config/client`
- Pull new configuration: business as usual
- Deploy configuration changes via `drush cim`: business as usual
- Configuration is imported from both `../config/sync` and `../config/client`

Option 3

Config Ignore

- Add config names/keys allowed to change to ignore config.
- Pull new configuration: business as usual
- Deploy configuration changes via `drush cim`: business as usual
- Configuration is imported from both `../config/sync` and the active config.

Chapter 9

Shared configuration

- Re-using configuration?
- Multisite configuration?

Features

- Bundle configuration for re-use on other sites
- Site then owns configuration, feature update unsolved.
- Great for starter-kit to optionally add more features

Config Split for multisite

- Shared splits or shared sync with site specific splits
- Join BOF on Thursday

DRUPALCON VIENNA

JOIN US FOR CONTRIBUTION SPRINT

Friday, 29 September, 2017

Mentored Core Sprint

9:00-18:00
Room: Stolz 2

First time Sprinter Workshop

9:00-12:00
Room: Lehar 1 - Lehar 2

General Sprint

9:00-18:00
Room: Mall

#drupalsprints

DRUPALCON VIENNA

WHAT DID YOU THINK?

Locate this session at the DrupalCon Vienna website:

http://vienna2017.drupal.org/schedule

Take the survey!

https://www.surveymonkey.com/r/drupalconvienna