Building and Maintaining a Distribution in Drupal 7 with Features

Antonio De Marco

Andrea Pescetti

http://nuvole.org

@nuvoleweb

Nuvole: Our Team

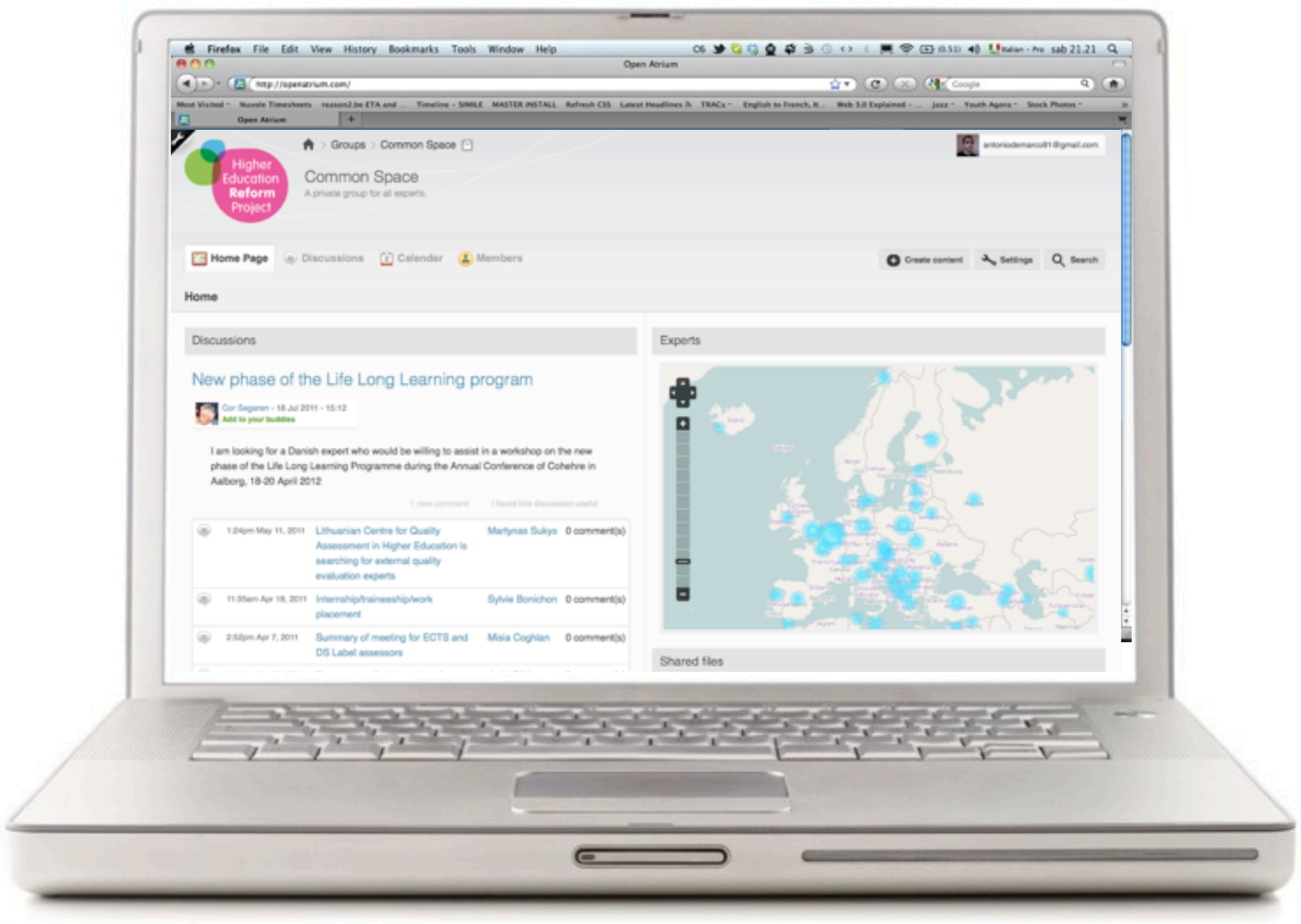
BELGIUM
Brussels

ITALY
Parma

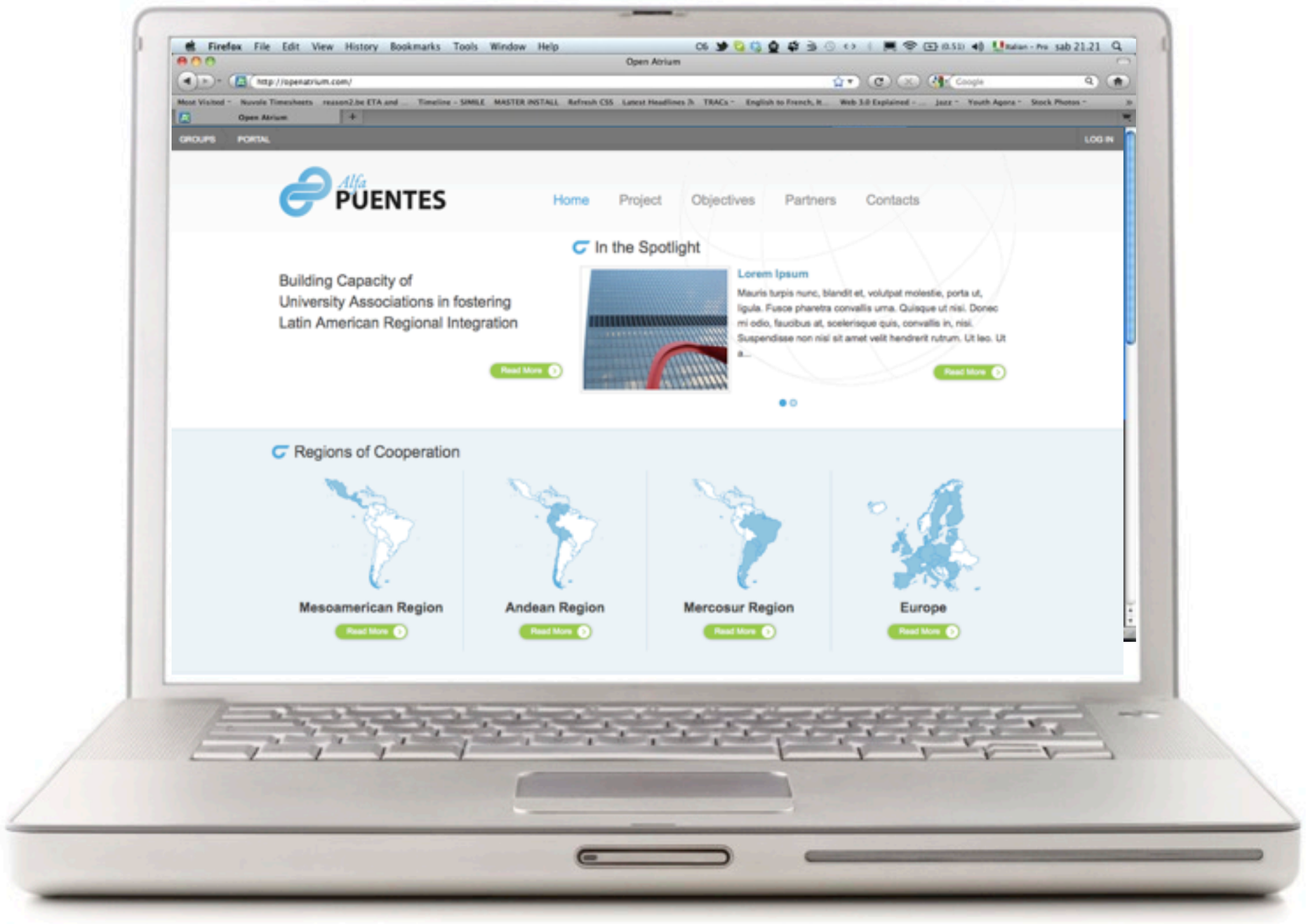
Clients in Europe and USA

Working with Drupal Distributions

Serving International Organizations



Serving International Organizations



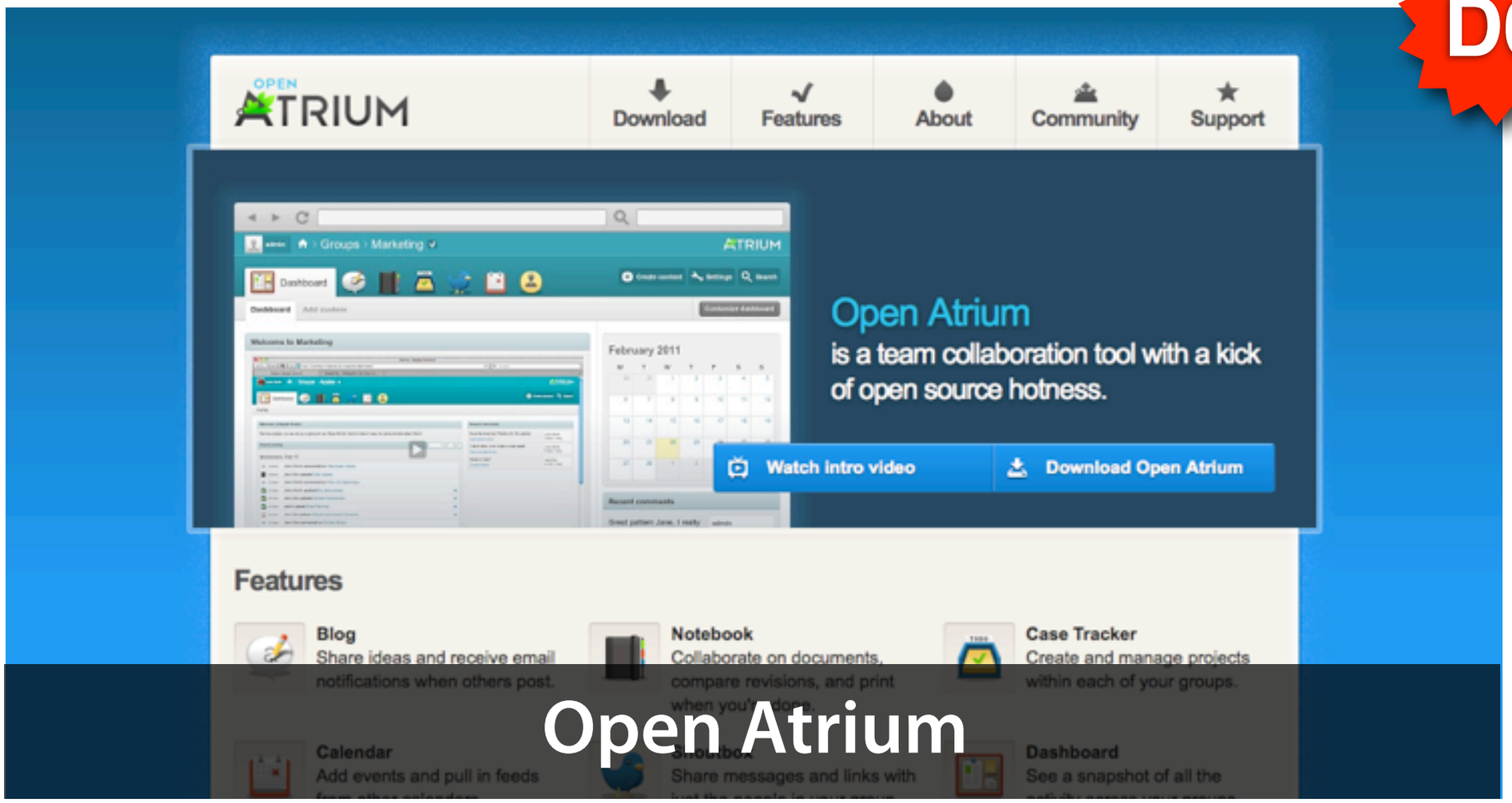
Drupal Distributions

Drupal Distribution

A pre-packaged Drupal installation meant to address a specific use case.

Popular Distributions

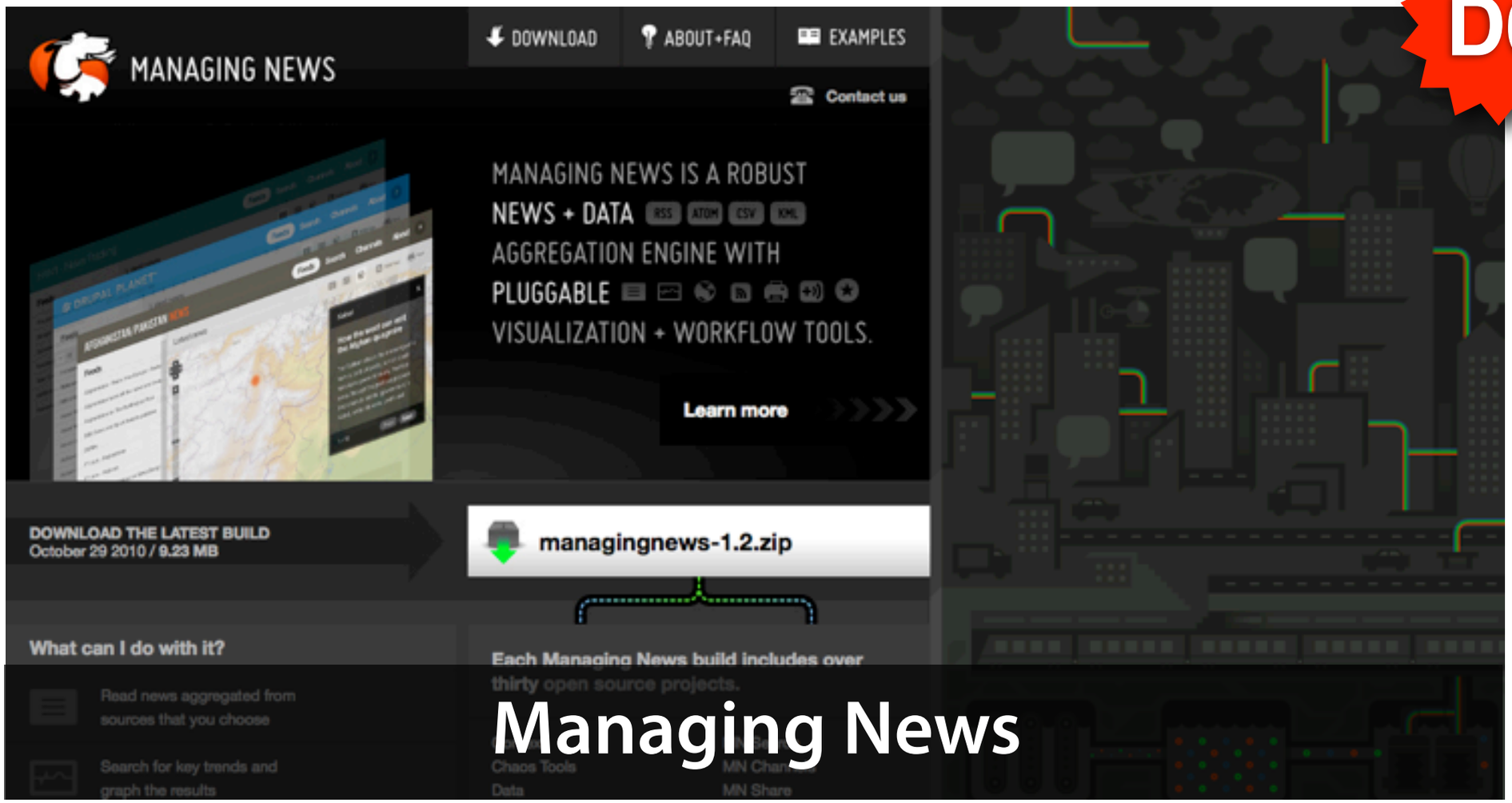
D6



Open Atrium

D6

Managing News

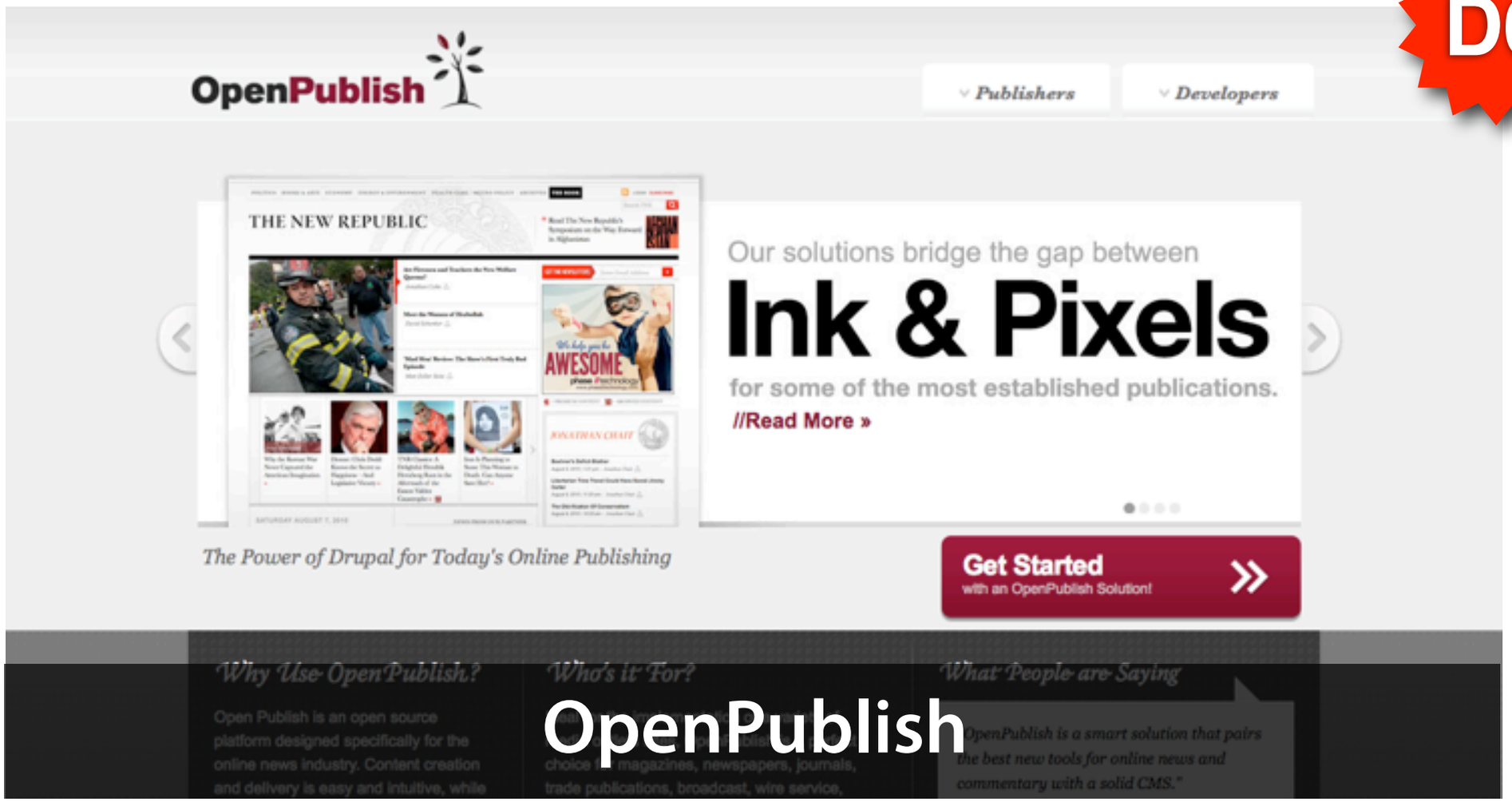



use COD!
Conference Organizing Distribution
COD is online software for conference and event sites, including registration and management
Home
Benefits
Download
Site showcase
News
Get support
Get involved
Conference and event sites built with COD
DrupalCon London 2011
MeeGo Conference 2010
Power Shift 2011
COD


Home
Hosted
Download
FAQ
Contact
videola
OPEN SOURCE IPTV
HOSTED VIDEOLA
Learn More
Download
VIDEOLA DISTRO

D6

Videola


D6
OpenPublish
Publishers
Developers
THE NEW REPUBLIC
Our solutions bridge the gap between
Ink & Pixels
for some of the most established publications.
//Read More »
The Power of Drupal for Today's Online Publishing
Get Started
with an OpenPublish Solution!
Why Use OpenPublish?
Who's it For?
What People are Saying
OpenPublish


OpenPublic
THE VISION
A Drupal CMS for government and policy groups
FEATURES
Secure, flexible, scalable, compliant, and intuitive
ROADMAP
Feature enhancements driven by our collaborators
COMMUNITY
Opportunities to strengthen OpenPublic for everyone
D7
BY THE PEOPLE
for
THE PEOPLE
OpenPublic is an open source content management system based on Drupal and tailored for building websites for government
DOWNLOAD


THE VISION
OpenPublic

Products are Preconfigured.

Distributions are Products

Products are Repeatable.

Distributions are Products

Products Must be Maintained.

Distributions are Products

Products Need to be Upgraded.

Distributions are Products

They make things

Minimally Preconfigured and Repeatable

What about Installation Profiles?

Select an installation profile

- **Choose profile**
- Choose language
- Verify requirements
- Set up database
- Install profile
- Configure site
- Finished

(•) Standard

Install with commonly used features pre-configured.

( ) Minimal

Start with only a few modules enabled.

Save and continue

Syntactically, no difference.

Not all Profiles are Distributions

Profiles

Ready to build upon

Focus on Developers

Distributions

Ready to use

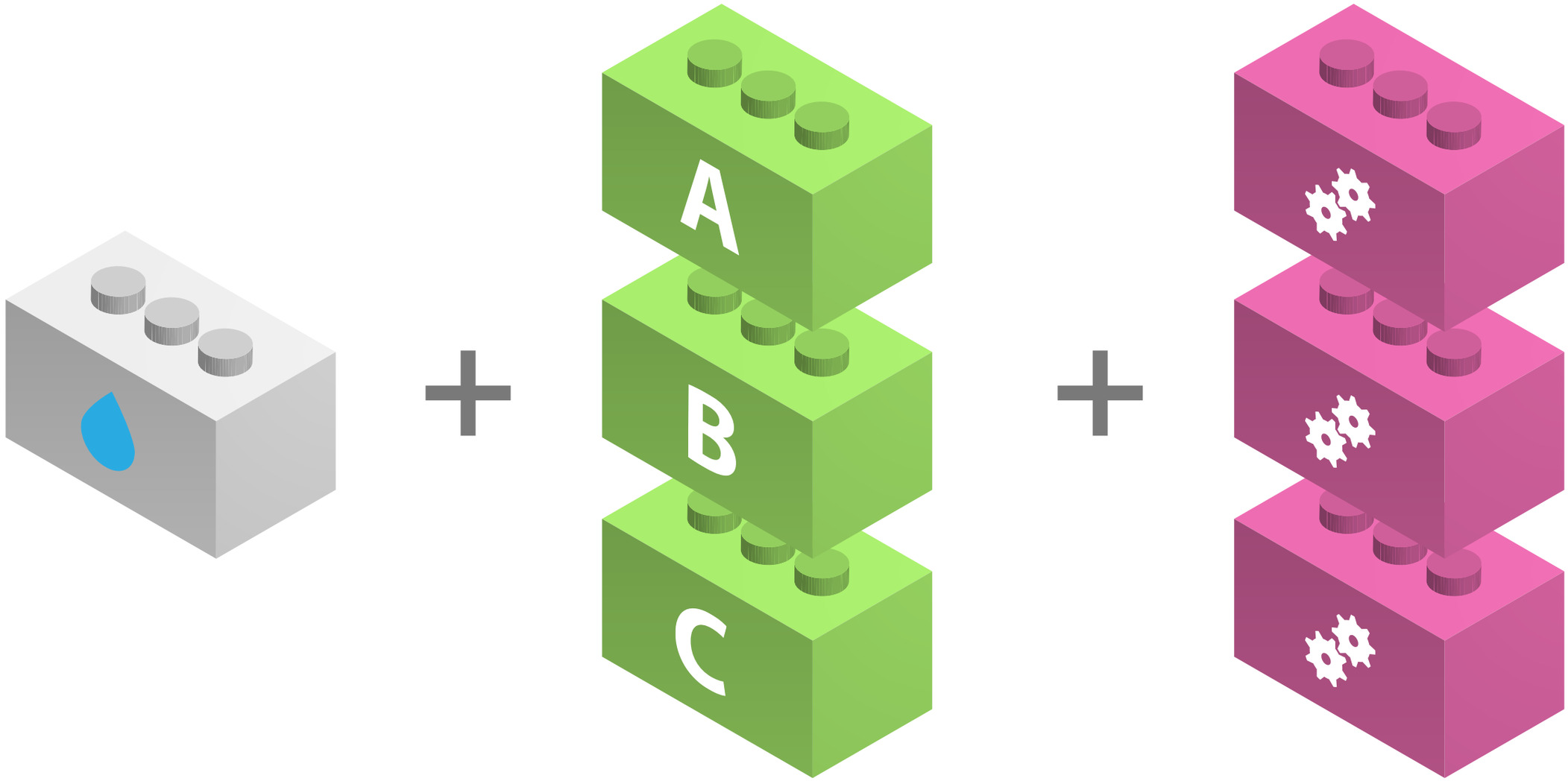
Focus on Final User

Core

Code

Modules, Themes, etc...

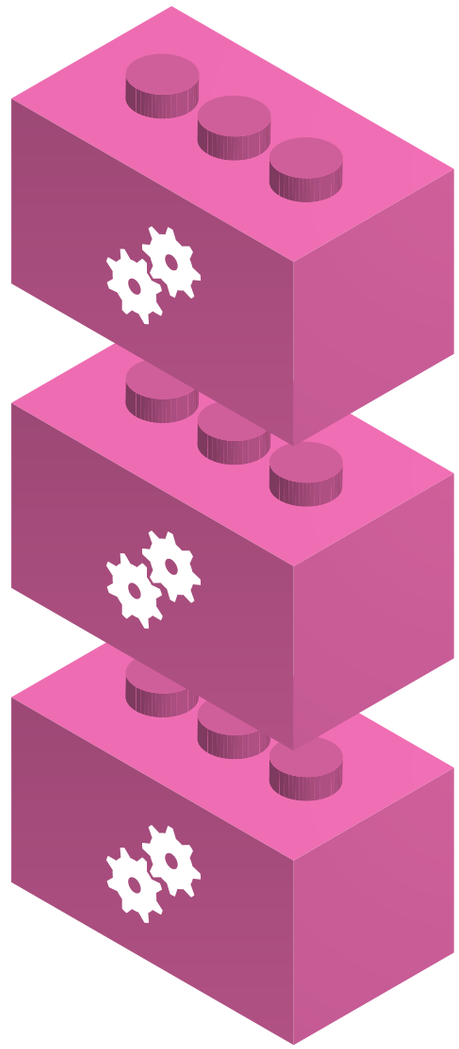
Configuration



Yet it is Drupal

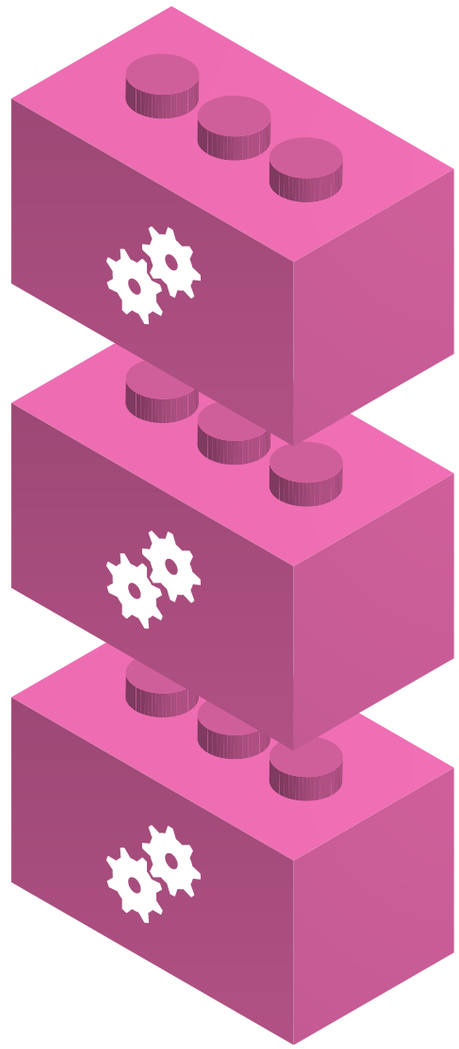
Storing Configuration

- Drupal traditionally stores configuration in database.
- But distributions cannot ship their default configuration this way.



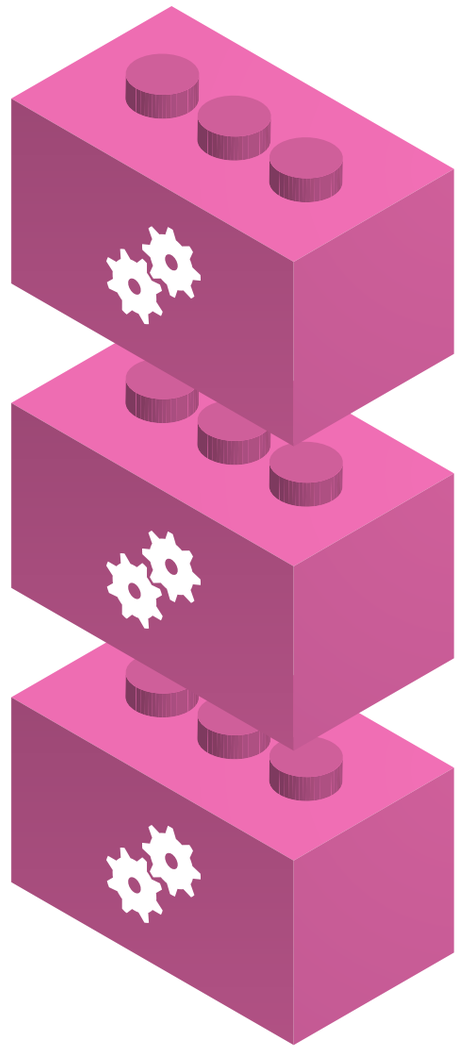
Configuration in Database: Major Drawbacks

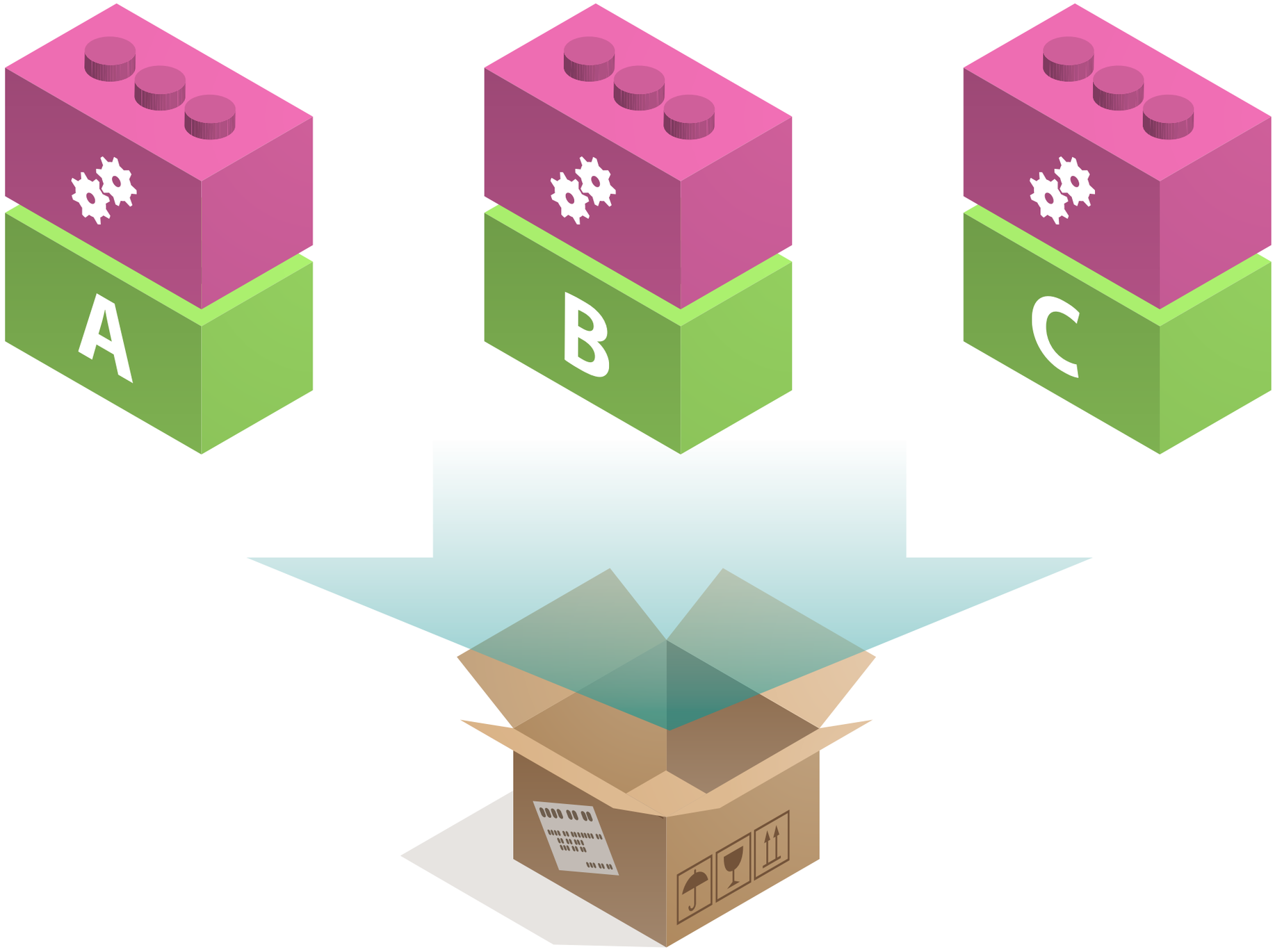
- Severe limits on repeatability
  - Database Engine
  - Table prefixes
  - “Dirty” state



Configuration in Database: Major Drawbacks

- Difficult to upgrade
- Content and configuration are mixed in database
- Hard to maintain for developers
  - Not ideal for a distributed team
  - Easy to lose control

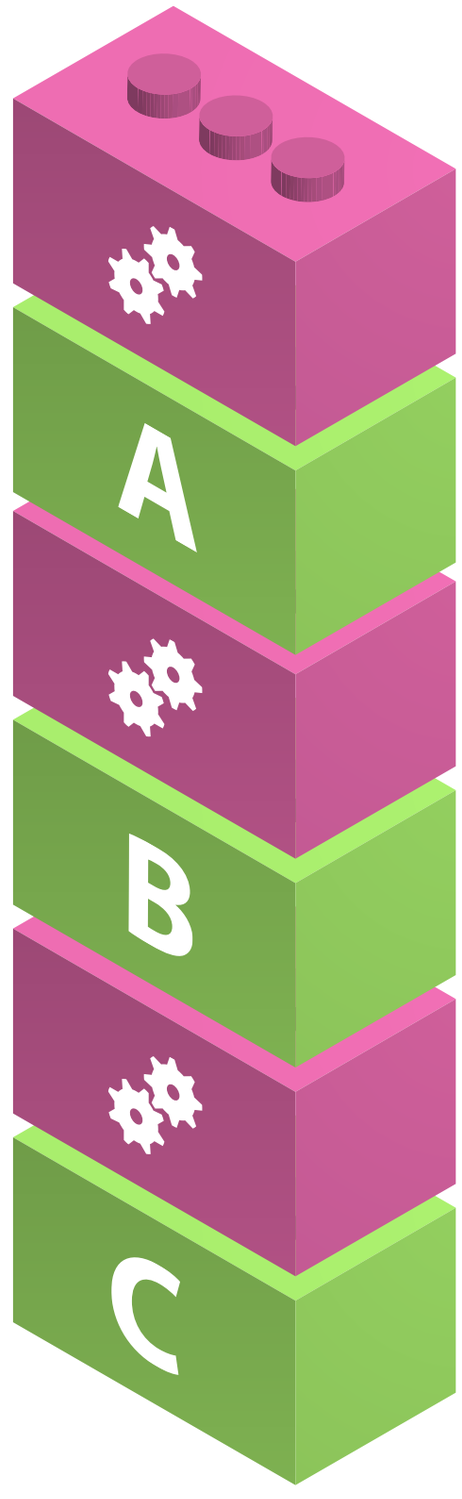




Store Configuration in Code

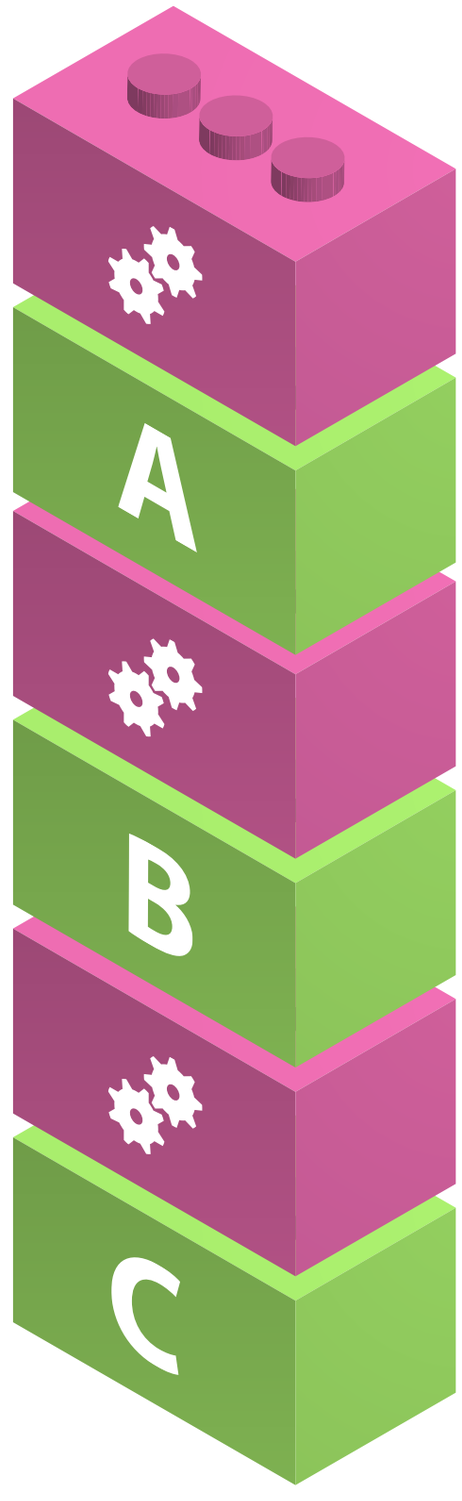
Configuration in Code: Major Benefits

- Configuration is generated via database-agnostic PHP code
- Clean installation procedure
- Clear upgrade path



Configuration in Code: Major Benefits

- Easier to maintain for developers
- Code can be versioned
- Conflicts can be solved
- Content and settings are separated



2

Building a Distribution

Packaging the code

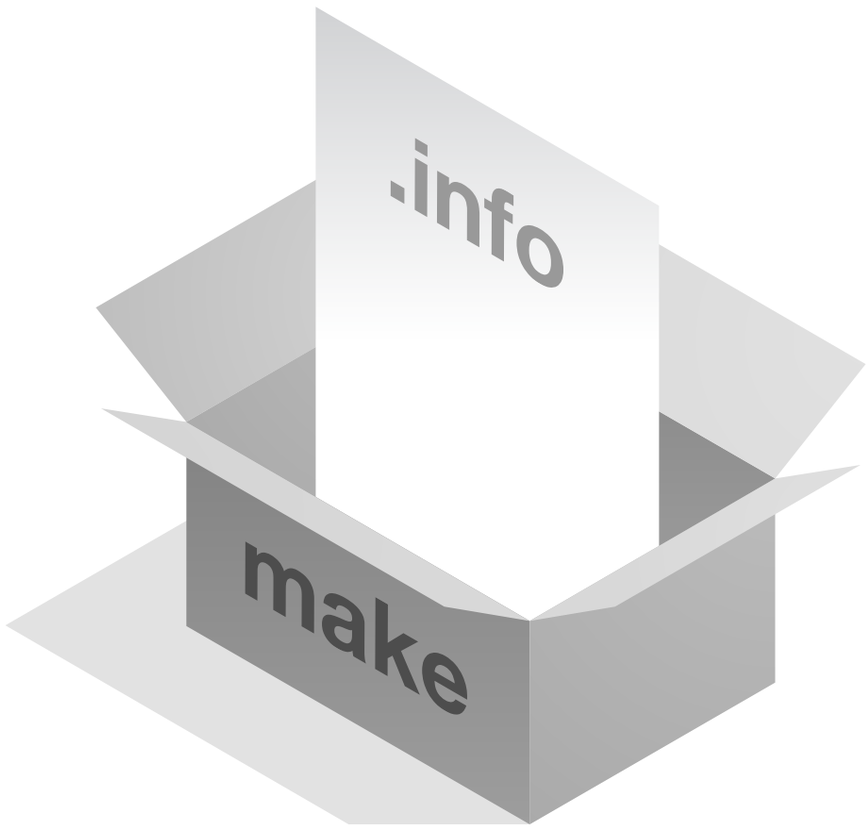
Meet...

Drupalissimo

Your Drupal Shop Distro

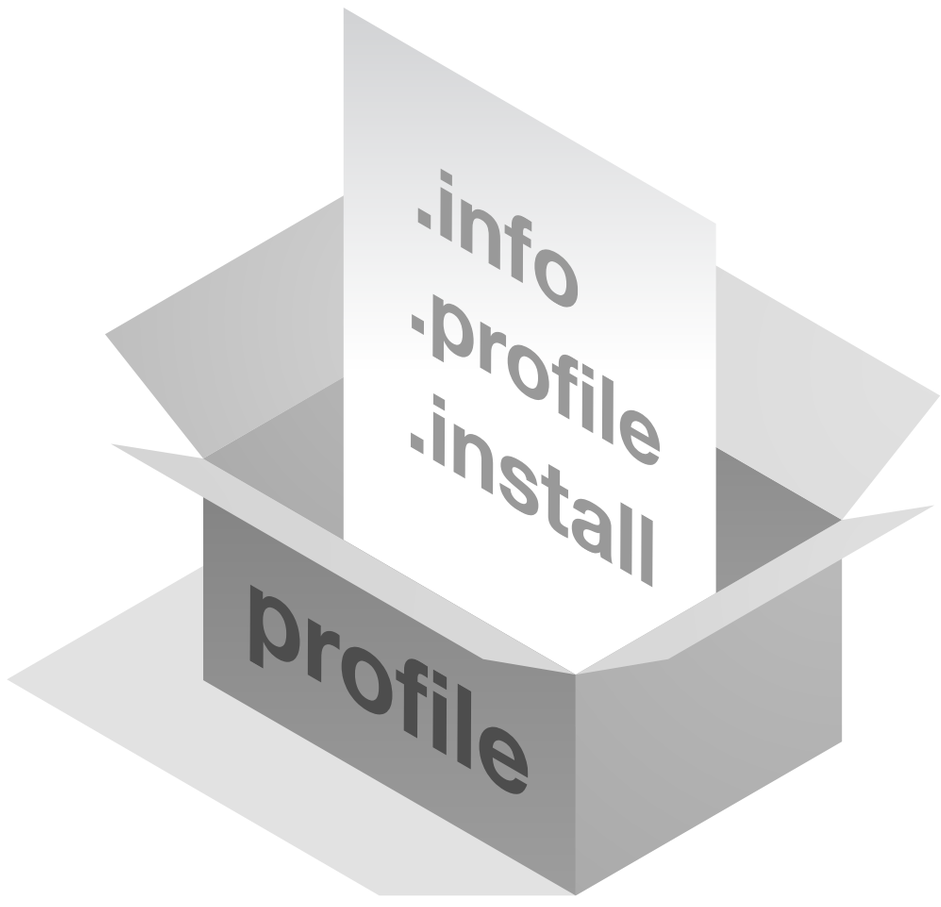
...your distro buddy!

Bootstrap



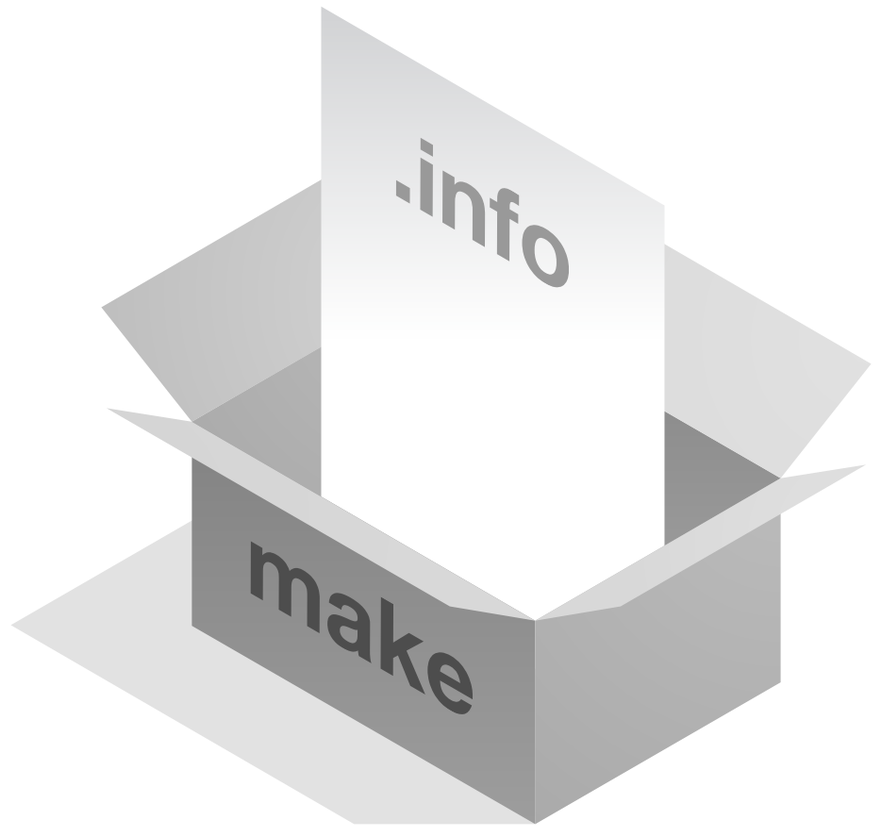
A Makefile to get your code

Makefile



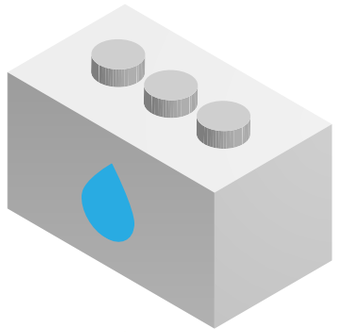
A Profile to manage the installation

Installation Profile



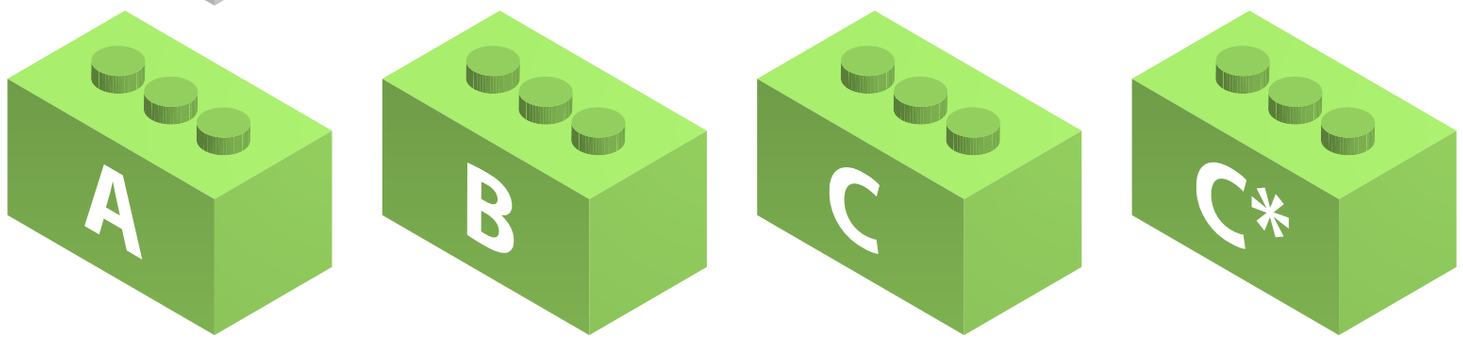
Makefile

Core

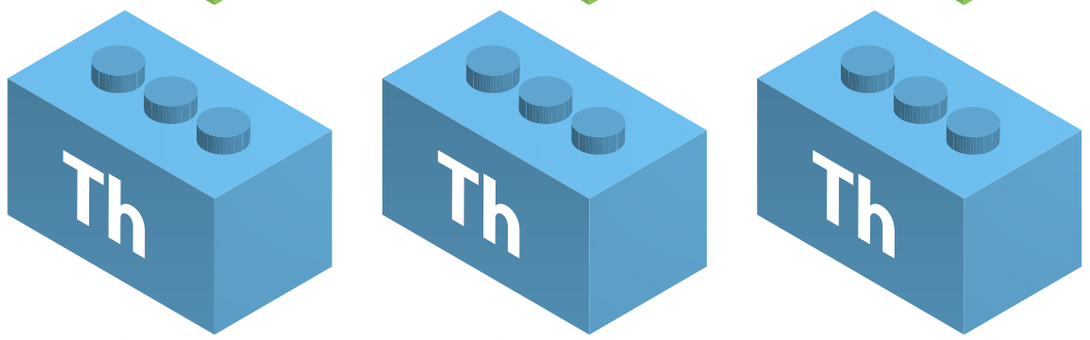


Modules

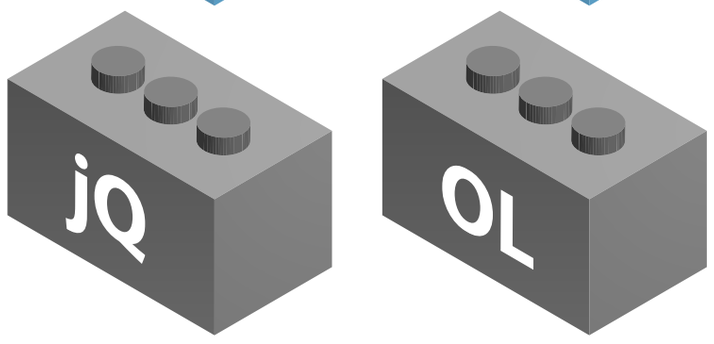
Contributed, Custom, Patched



Themes

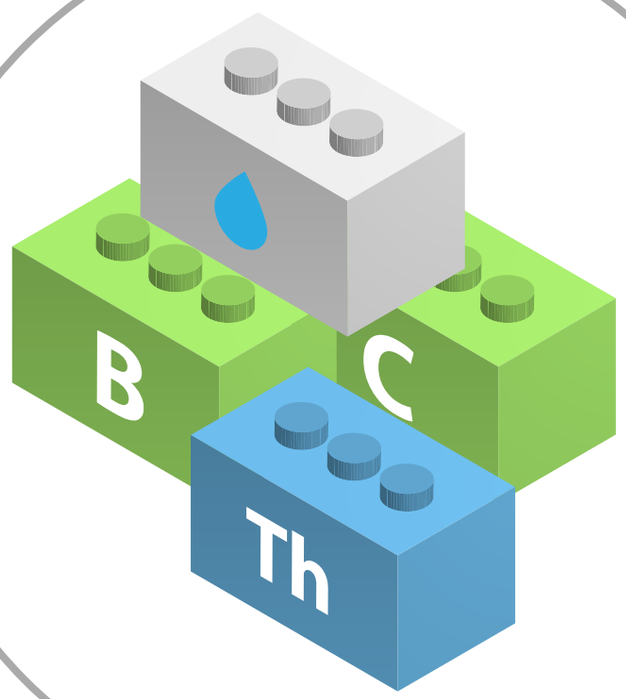


External Libraries

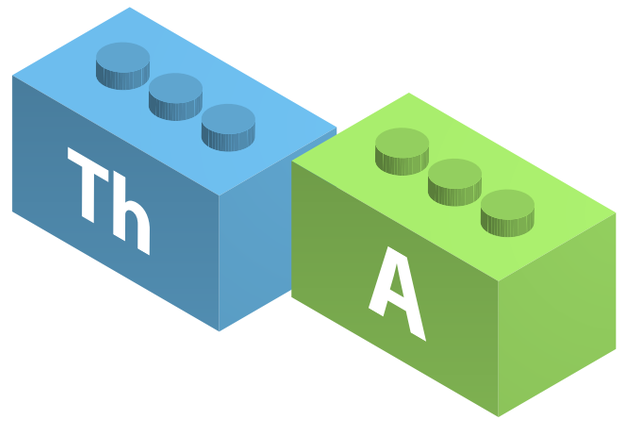


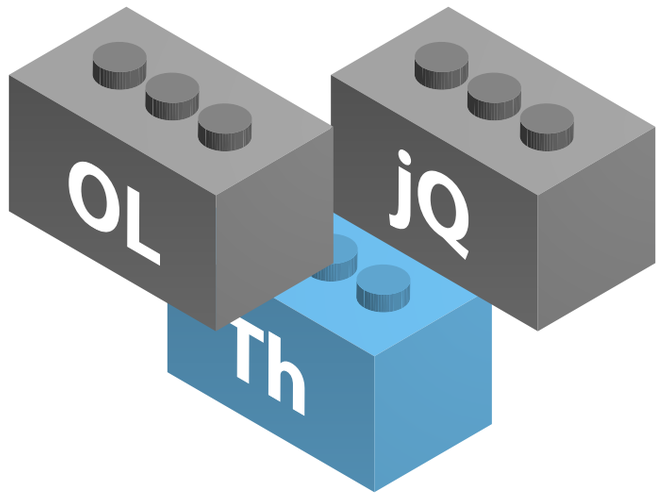
A distribution's building blocks

drupal.org



github.com





example.com

The best way to package code

Introducing Drush Make

Drush Make

View | Version control

Posted by dmitrig01 on *July 23, 2009 at 9:18pm*

Drush make is an extension to drush that can create a ready-to-use drupal site, pulling sources from various locations. It does this by parsing a flat text file (similar to a drupal `.info` file) and downloading the sources it describes. In practical terms, this means that it is possible to distribute a complicated Drupal distribution as a single text file.

- Getting plain `.tar.gz` and `.zip` files (particularly useful for libraries that can not be distributed directly with drupal core or modules).
- Fetching and applying patches.
- Fetching modules, themes, and installation profiles, but also external libraries.

Drush make does not turn modules on automatically: it only assembles Drupal directories -- it does not touch any database.

Usage

The `drush make` command can be executed from a path within a Drupal codebase or independent of any Drupal sites entirely.

Examples

1. Build a Drupal site at `example/` including Drupal core and any projects defined in the makefile:

   ```
   drush make example.make example
   ```

2. Build a tarball of the platform above as `example.tar.gz`:

   ```
   drush make --tar example.make example
   ```

3. Build an installation profile within an existing Drupal site:

   ```
   drush make --no-core --contrib-destination=. installprofile.make
   ```

Documentation & other resources

Maintainers for Drush Make

dmitrig01 - 358 commits

last: 47 weeks ago, first: 1 year ago

View all committers
View commits

Issues for Drush Make

To avoid duplicates, please search before submitting a new issue.

Search

Advanced search

All issues
75 open, 385 total

Bug reports
25 open, 164 total

Subscribe via e-mail
Oldest open issue: 18 Sep 09

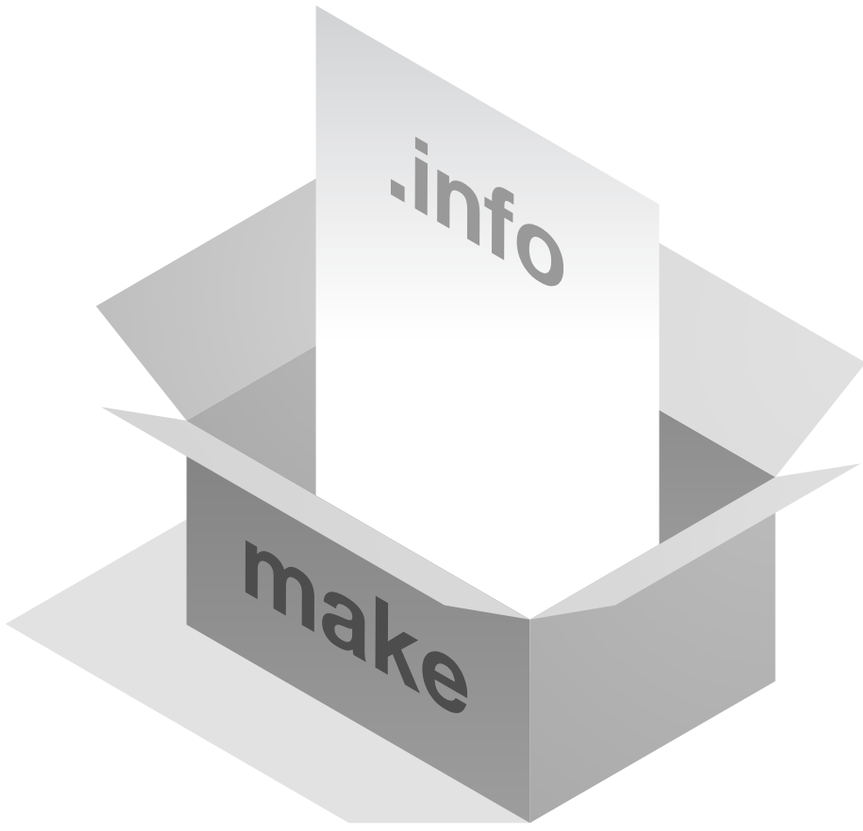
Related projects

jCarousel

Mailhandler

Introducing Drush Make

- A single .info file to describe modules, dependencies and patches
- A one-line command to download contrib and custom code: libraries, modules, themes, etc...
- A trustworthy reference index for your project



Makefile close-up

distro.make - drupalissimo.make

distro.make

Package Core and Profile

```
; distro.make
; Usage:
; $ drush make distro.make [directory]
;

api = 2
core = 7.x

projects[drupal][type] = core
projects[drupal][version] = "7.7"

; Make system directories configurable to allow tests in profiles/[name]/modules
; http://drupal.org/node/911354
projects[drupal][patch][911354] = http://drupal.org/files/issues/911354.43.patch

; Missing drupal_alter() for text formats and filters
; http://drupal.org/node/903730
projects[drupal][patch][903730] = http://drupal.org/files/issues/drupal.filter-c

; Use vocabulary machine name for permissions
; http://drupal.org/node/995156
projects[drupal][patch][995156] = http://drupal.org/files/issues/995156-5_portab
```

```
$ drush make distro.make drupalissimo

Project information for drupal retrieved.
drupal downloaded from http://ftp.drupal.org/files/projects/drupal-7.7.tar.gz.
drupal patched with 911354.43.patch.
drupal patched with drupal.filter-alter.82.patch.
drupal patched with 995156-5_portable_taxonomy_permissions.patch.
Generated PATCHES.txt file for drupal
```

Where is my installation profile?

```
...

api = 2
core = 7.x

projects[drupal][type] = core
projects[drupal][version] = "7.7"

; Make system directories configurable to allow tests in profiles/[name]/modules
; http://drupal.org/node/911354
projects[drupal][patch][911354] = http://drupal.org/files/issues/911354.43.patch

; Missing drupal_alter() for text formats and filters
; http://drupal.org/node/903730
projects[drupal][patch][903730] = http://drupal.org/files/issues/drupal.filter-c

; Use vocabulary machine name for permissions
; http://drupal.org/node/995156
projects[drupal][patch][995156] = http://drupal.org/files/issues/995156-5_portab

projects[drupalissimo][type] = profile
projects[drupalissimo][download][type] = git
projects[drupalissimo][download][url] = git://github.com/nuvoleweb/drupalissimo
```

```
$ git clone git@github.com:nuvoleweb/drupalissimo.git drupalissimo
...
$ ls -l drupalissimo
total 16
-rw-r--r--  1 ademarco  staff   1.1K Aug 15 07:59 README.txt
-rw-r--r--@ 1 ademarco  staff   1.2K Aug 15 07:58 distro.make
-rw-r--r--  1 ademarco  staff   2.1K Aug 15 07:59 drupalissimo.install
-rw-r--r--@ 1 ademarco  staff   498B Aug 15 07:58 drupalissimo.make
-rw-r--r--  1 ademarco  staff   3.2K Aug 15 07:59 drupalissimo.profile
```

drupalissimo.make

Package my distribution

```
api = 2
core = 7.x

; Modules ============================================================

projects[admin][subdir] = contrib
projects[admin][version] = 2.0-beta3

projects[context][subdir] = contrib
projects[context][version] = 3.0-beta1

...

; Features ===========================================================

projects[drupalissimo_core][type] = module
projects[drupalissimo_core][subdir] = features
projects[drupalissimo_core][download][type] = "git"
projects[drupalissimo_core][download][url] = "git://github.com/nuvoleweb/drupa

projects[drupalissimo_blog][type] = module
projects[drupalissimo_blog][subdir] = features
projects[drupalissimo_blog][download][type] = "git"
projects[drupalissimo_blog][download][url] = "git://github.com/nuvoleweb/drupa
```

Build Kit

Extendable distribution, reusable .make file

Build Kit

View | Version control

Posted by jmiccolis on *June 7, 2010 at 8:55pm*

Get started building with Drupal fast.

Build Kit is a basic distribution meant to capture elements that are generally useful and make building Drupal sites and Drupal distributions easier.

Build Kit helps site builders

- use install profiles and drush make for defining projects
- manage the dev > staging > live workflow problem using Features and exportables
- keep track of important upstream patches that are critical to Drupal distributions

Requirements

In addition to the standard Drupal requirements you will need the following to make use of Build Kit:

- drush 3.1 – http://drupal.org/project/drush
- drush make 2.0 beta 9 – http://drupal.org/project/drush_make
- git – http://git-scm.com

Getting started (7.x)

Build Kit for 7.x requires several patches to be applied to Drupal core. It provides a `distro.make` file for building a full Drupal distro including core patches as well as a copy of the `buildkit` install profile.

1. Download Build Kit (see this projects Git instructions) and grab the `distro.make` file and run:

   ```
   $ drush make distro.make [directory]
   ```

   or use its url on Drupal.org directly:

   ```
   $ drush make "http://drupalcode.org/project/buildkit.git/blob_plain/refs/heads
   /7.x-2.x:/distro.make" [directory]
   ```

2. Choose the "Build Kit" install profile when installing Drupal

Maintainers for Build Kit

langworthy – 24 commits
last: 5 days ago, first: 24 weeks ago

jmiccolis – 11 commits
last: 31 weeks ago, first: 1 year ago

alex_b – 19 commits
last: 36 weeks ago, first: 1 year ago

yhahn – 27 commits
last: 42 weeks ago, first: 1 year ago

Will White – 1 commit
last: 51 weeks ago, first: 51 weeks ago

View all committers
View commits

Issues for Build Kit

To avoid duplicates, please search before submitting a new issue.

Search

Advanced search

All issues
5 open, 14 total

Bug reports
0 open, 3 total

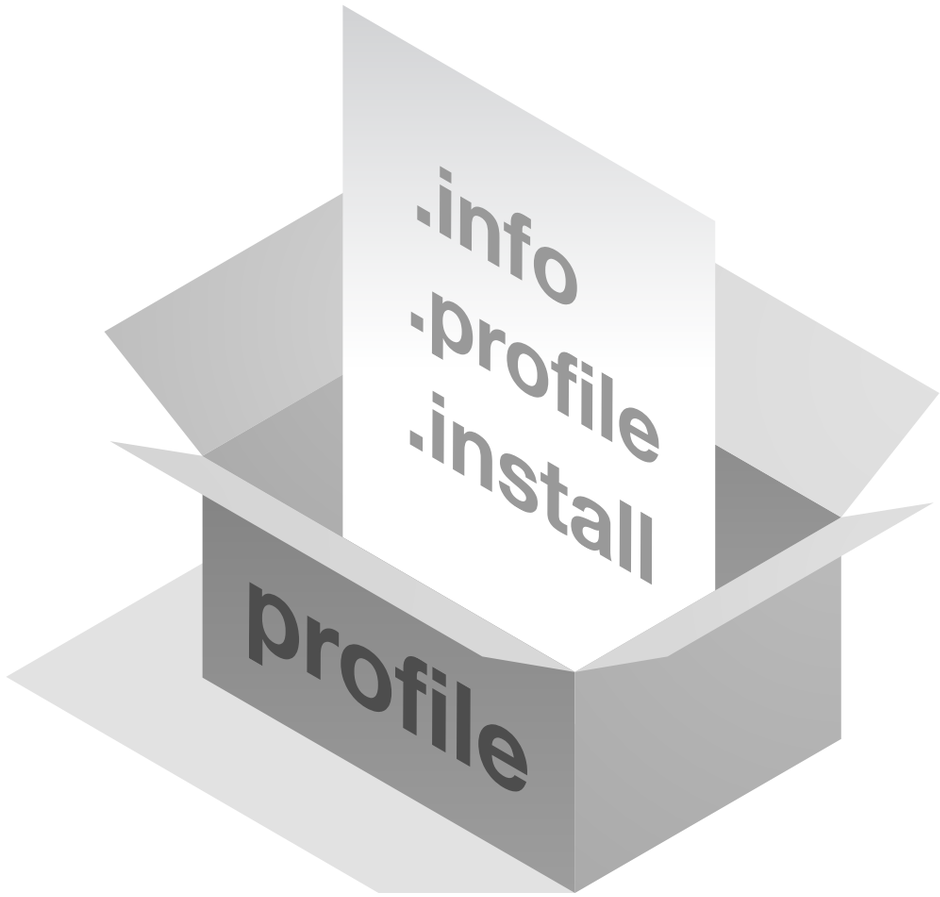
Subscribe via e-mail
Oldest open issue: 13 Jun 10

Resources

Read documentation

```
; distro.make
;
; $ drush make distro.make [directory]
;

api = 2
core = 7.x

; Include Build Kit distro makefile via URL
includes[] = http://drupalcode.org/project/buildkit.git/blob_plain/refs/heads/7

projects[drupalissimo][type] = profile
projects[drupalissimo][download][type] = git
projects[drupalissimo][download][url] = git://github.com/nuvoleweb/drupalissimo
```

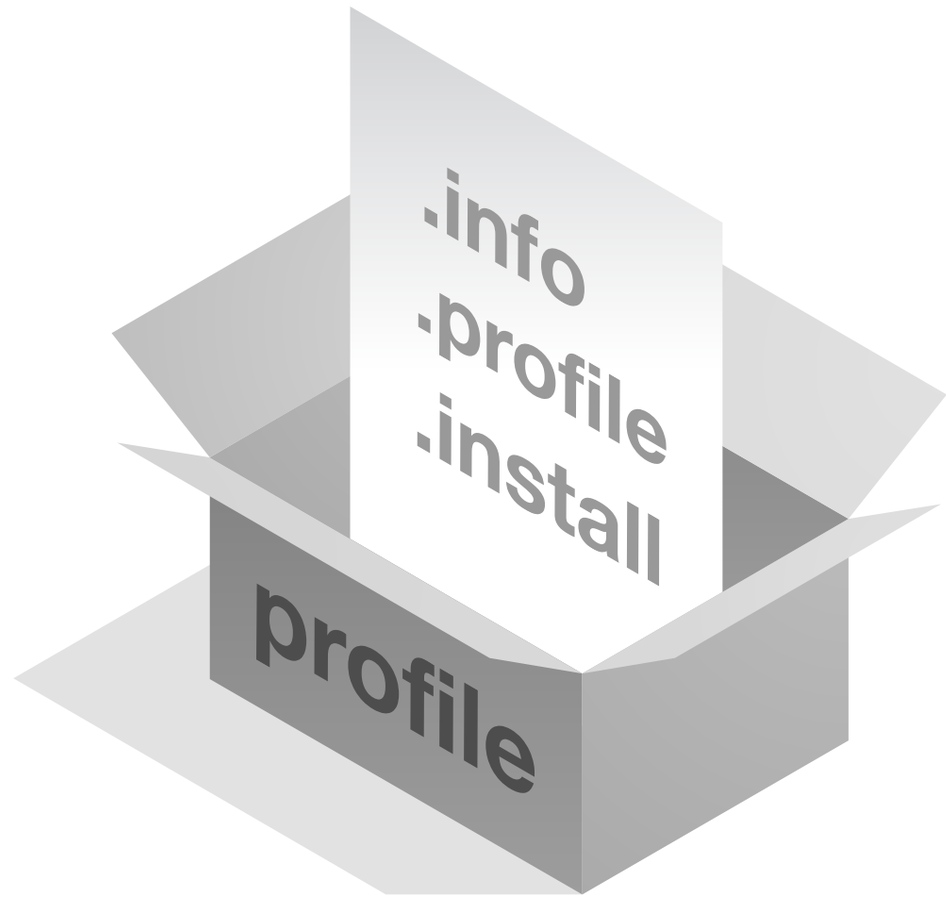
```
;
; drupalissimo.make
;

api = 2
core = 7.x

; Include Build Kit install profile makefile via URL
includes[] = http://drupalcode.org/project/buildkit.git/blob_plain/refs/heads/7

; Features ============================================================

projects[drupalissimo_core][type] = module
projects[drupalissimo_core][subdir] = features
projects[drupalissimo_core][download][type] = "git"
projects[drupalissimo_core][download][url] = "git://github.com/nuvoleweb/drupal

projects[drupalissimo_blog][type] = module
projects[drupalissimo_blog][subdir] = features
projects[drupalissimo_blog][download][type] = "git"
projects[drupalissimo_blog][download][url] = "git://github.com/nuvoleweb/drupal
```



Installation Profile

Profiles: just like modules

- An .info file to specify dependencies
- An .install file to perform installation tasks and upgrades
- Fully customizable via .profile files



drupalissimo.info

```
name = Drupalissimo
core = 7.x
description = Drupalissimo installation profile.

; Core
dependencies[] = book
dependencies[] = block
dependencies[] = contact
dependencies[] = dblog
dependencies[] = field_ui
dependencies[] = file
...

; Contrib
dependencies[] = admin
dependencies[] = colorbox
dependencies[] = ds
...

; Features
dependencies[] = drupalissimo_core
dependencies[] = drupalissimo_blog
```

drupalissimo.profile

A quick APIs overview

```php
/**
 * Implements hook_form_FORM_ID_alter().
 */
function drupalissimo_form_install_configure_form_alter(&$form, $form_state) {

  $form['site_information']['site_name']
       ['#default_value'] = 'Drupalissimo';
  $form['site_information']
       ['site_mail']['#default_value'] = 'info@drupalissimo.com';

  $form['admin_account']['account']
       ['name']['#default_value'] = 'admin';
  $form['admin_account']['account']
       ['mail']['#default_value'] = 'dev@nuvole.org';

  $form['update_notifications']
       ['update_status_module']['#default_value'] = array(1 => FALSE, 2 => FALS
}
```

Configure site

- ✓ Choose profile
- ✓ Choose language
- ✓ Verify requirements
- ✓ Set up database
- ✓ Install profile
- ▸ **Configure site**
- Create taxonomy terms
- Configure site features
- Finished

SITE INFORMATION

Site name *

Drupalissimo

Site e-mail address *

info@drupalissimo.com

Automated e-mails, such as registration information, will be sent from this address. Use an address ending in your site's domain to help prevent these e-mails from being flagged as spam.

SITE MAINTENANCE ACCOUNT

Username *

admin

Spaces are allowed; punctuation is not allowed except for periods, hyphens, and underscores.

E-mail address *

drv@nuvole.org

Password *

Password strength:

Confirm password *

SERVER SETTINGS

```php
/**
 * Implements hook_install_tasks()
 */
function drupalissimo_install_tasks() {
  return array(
    'drupalissimo_create_terms' => array(
      'display_name' => st('Create taxonomy terms'),
    ),
    'drupalissimo_configure_site_features' => array(
      'display_name' => st('Configure site features'),
    ),
  );
}
```

Configure site

- ✓ Choose profile
- ✓ Choose language
- ✓ Verify requirements
- ✓ Set up database
- ✓ Install profile
- ▸ **Configure site**
  - Create taxonomy terms
  - Configure site features
  - Finished

SITE INFORMATION

Site name *

Drupalissimo

Site e-mail address *

info@drupalissimo.com

Automated e-mails, such as registration information, will be sent from this address. Use an address ending in your site's domain to help prevent these e-mails from being flagged as spam.

SITE MAINTENANCE ACCOUNT

Username *

admin

Spaces are allowed; punctuation is not allowed except for periods, hyphens, and underscores.

E-mail address *

drv@nuvole.org

Password *

Password strength:

Confirm password *

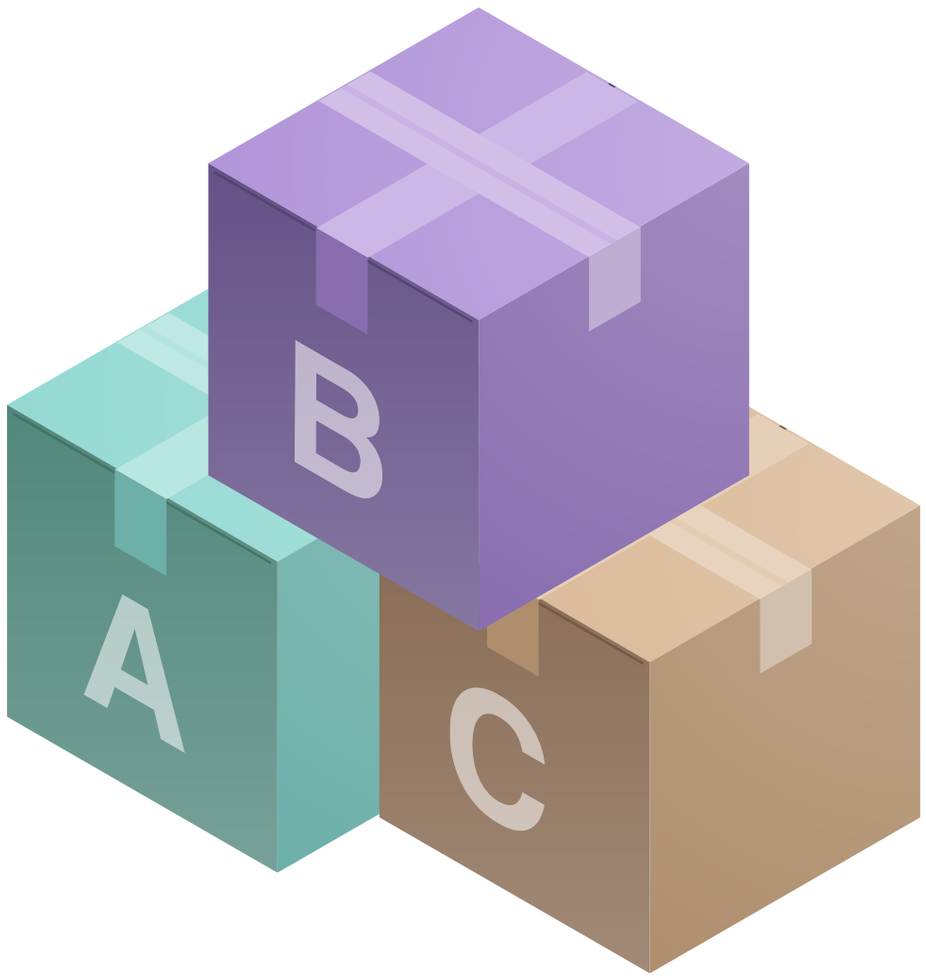
SERVER SETTINGS

```php
/**
 * Implements hook_install_tasks() callback
 */
function drupalissimo_create_terms() {
  $terms = array();
  $vocabulary = taxonomy_vocabulary_machine_name_load('category');

  $terms[] = 'Solution';
  $terms[] = 'Client';
  $terms[] = 'Use case';

  foreach ($terms as $name) {
    $term = new stdClass();
    $term->vid = $vocabulary->vid;
    $term->name = $name;
    taxonomy_term_save($term);
  }
}
```

```php
/**
 * Implements hook_install_tasks() callback
 */
function drupalissimo_configure_site_features() {

  // Create user roles
  $role = new stdClass();
  $role->name = 'editor';
  user_role_save($role);

  // Revert features
  $features = features_get_features();
  foreach ($features as $name => $feature) {
    if ($feature->status) {
      features_revert(array($name => array('variable', 'user_permission')));
    }
  }
  cache_clear_all();

  // Enable custom theme
  theme_enable(array('drupalissimo'));
  variable_set('theme_default', 'drupalissimo');
}
```

Building a Distribution

Packaging Configuration



Features

The best way to package configuration

Download & Extend

Features

Posted by febbraro on *March 13, 2009 at 10:02pm*

The features module enables the capture and management of features in Drupal. A feature is a collection of Drupal entities which taken together satisfy a certain use-case.

Features provides a UI and API for taking different site building components from modules with exportables and bundling them together in a single feature module. A feature module is like any other Drupal module except that it declares its components (e.g. views, contexts, CCK fields, etc.) in its `.info` file so that it can be checked, updated, or reverted programmatically.

Examples of features might be:

- A blog
- A pressroom
- An image gallery
- An e-commerce t-shirt store

Basic usage

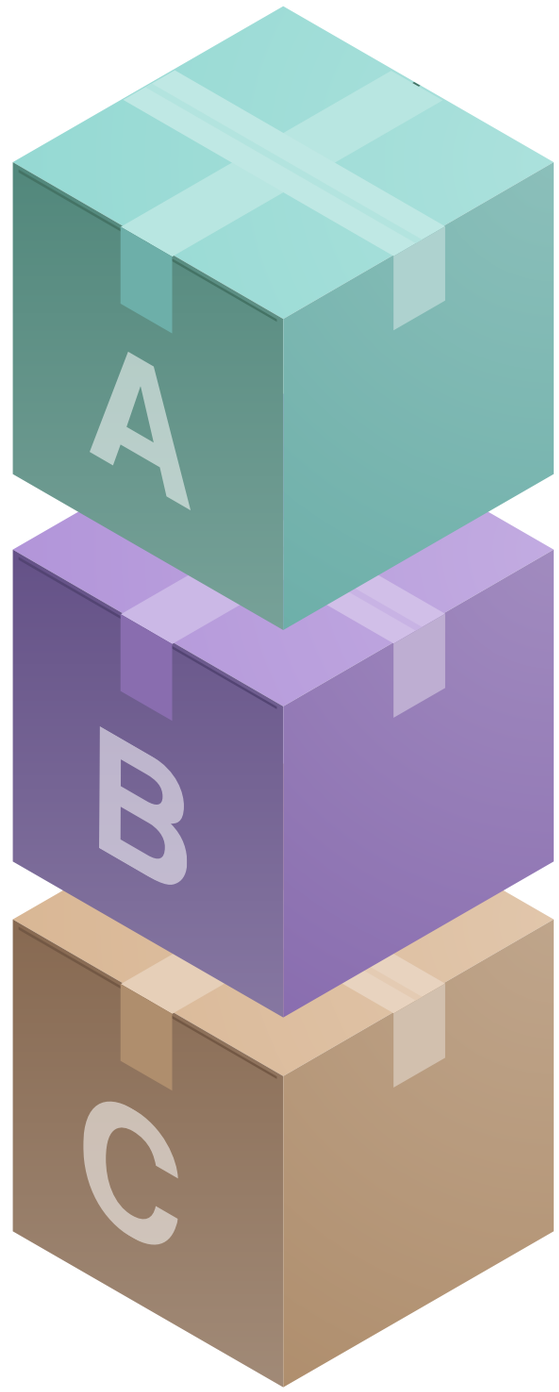
Features is geared toward usage by developers and site builders. It is not intended to be used by the general audience of your Drupal site. Features provides tools for accomplishing two important tasks:

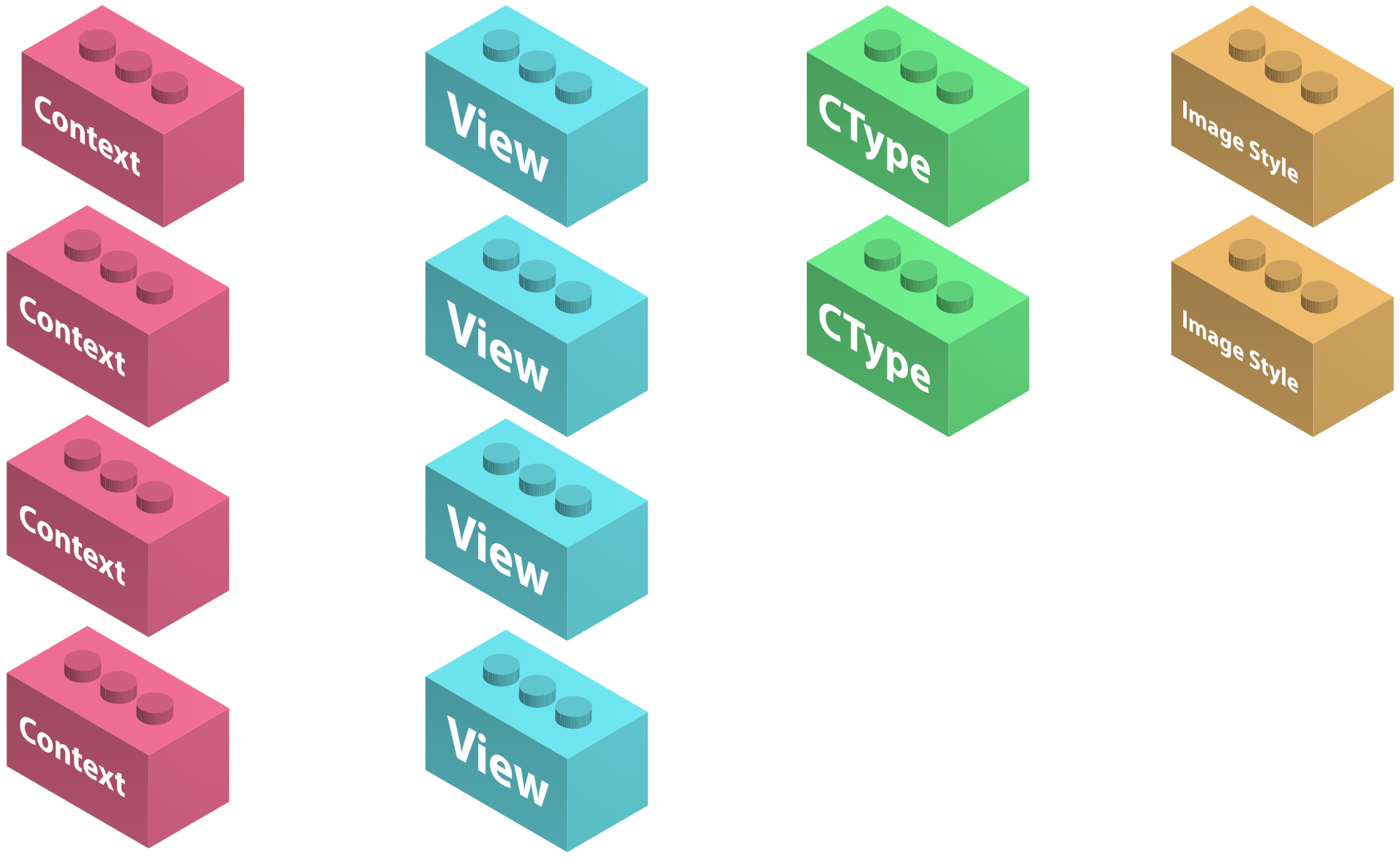
Task 1: Export features

You can build features in Drupal by using site building tools that are supported (see a short list

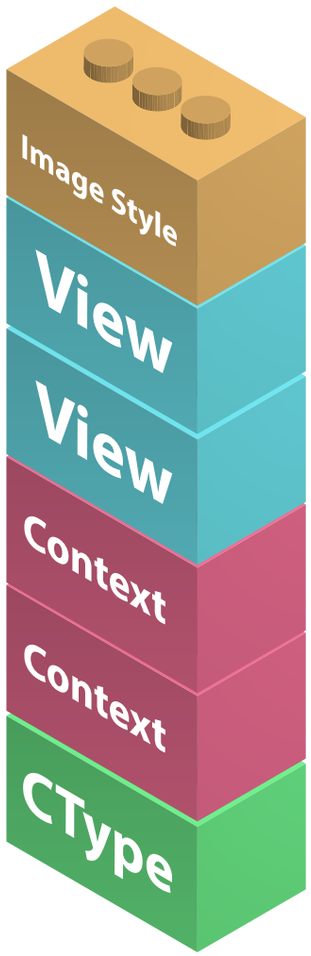
What is a feature?

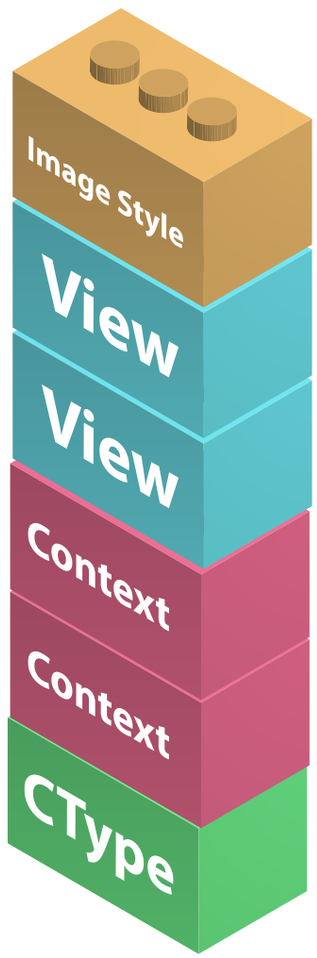
- A collection of Drupal elements which taken together satisfy a certain use-case.
- A modular piece of functionality for a Drupal site.
- A way to export configuration into PHP code, in the form of a module.
- http://drupal.org/project/features

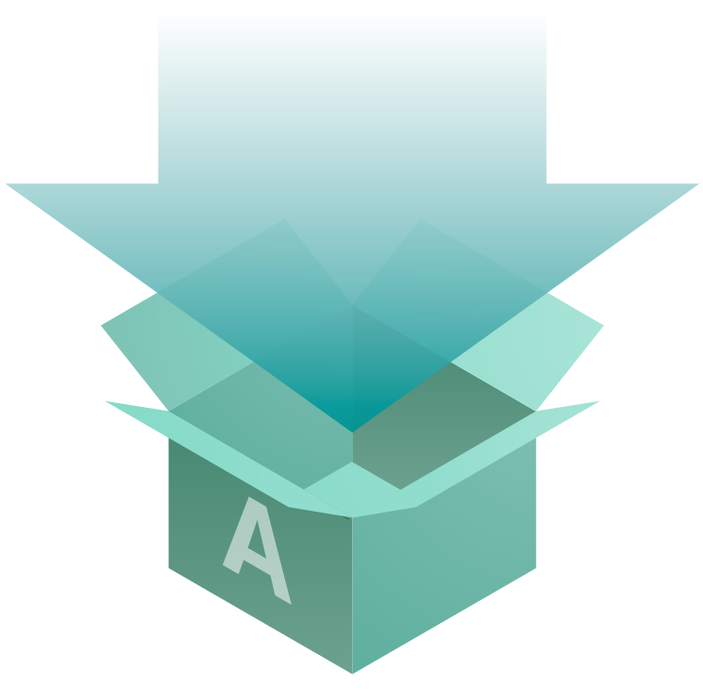


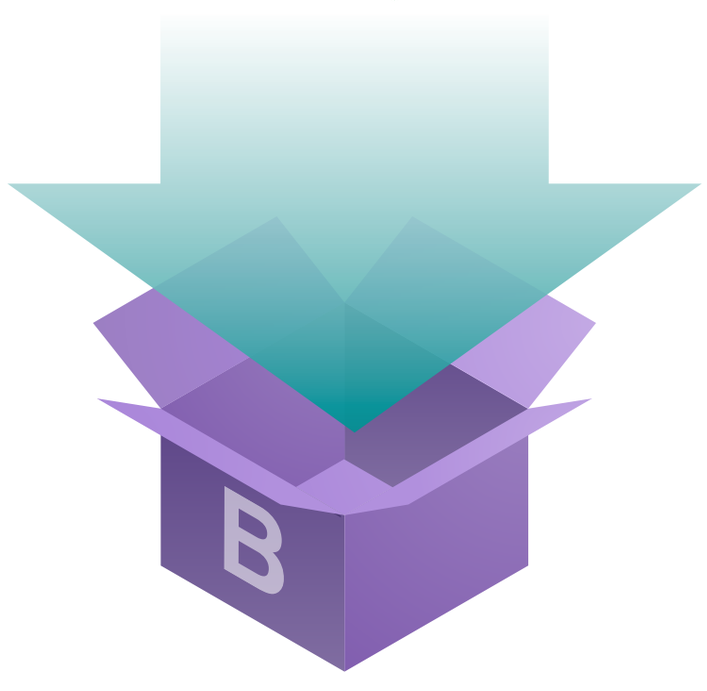


Configuration in Database









Packaged as Features

Creating a Feature

NAME *

Core

MACHINE-READABLE NAME *

feature_core

DESCRIPTION *

Core feature, stuff we need all the time.

VERSION

7.x-1.0-dev

URL OF UPDATE XML

EDIT COMPONENTS

CTools export API

ds:ds:1 strongarm:strongarm:1

Display suite view modes

core_small_teaser

Text formats

core_rich_text

Menus

main-menu

Menu links

main-menu:<front>

Permissions

access content use text format core_rich_text

Strongarm

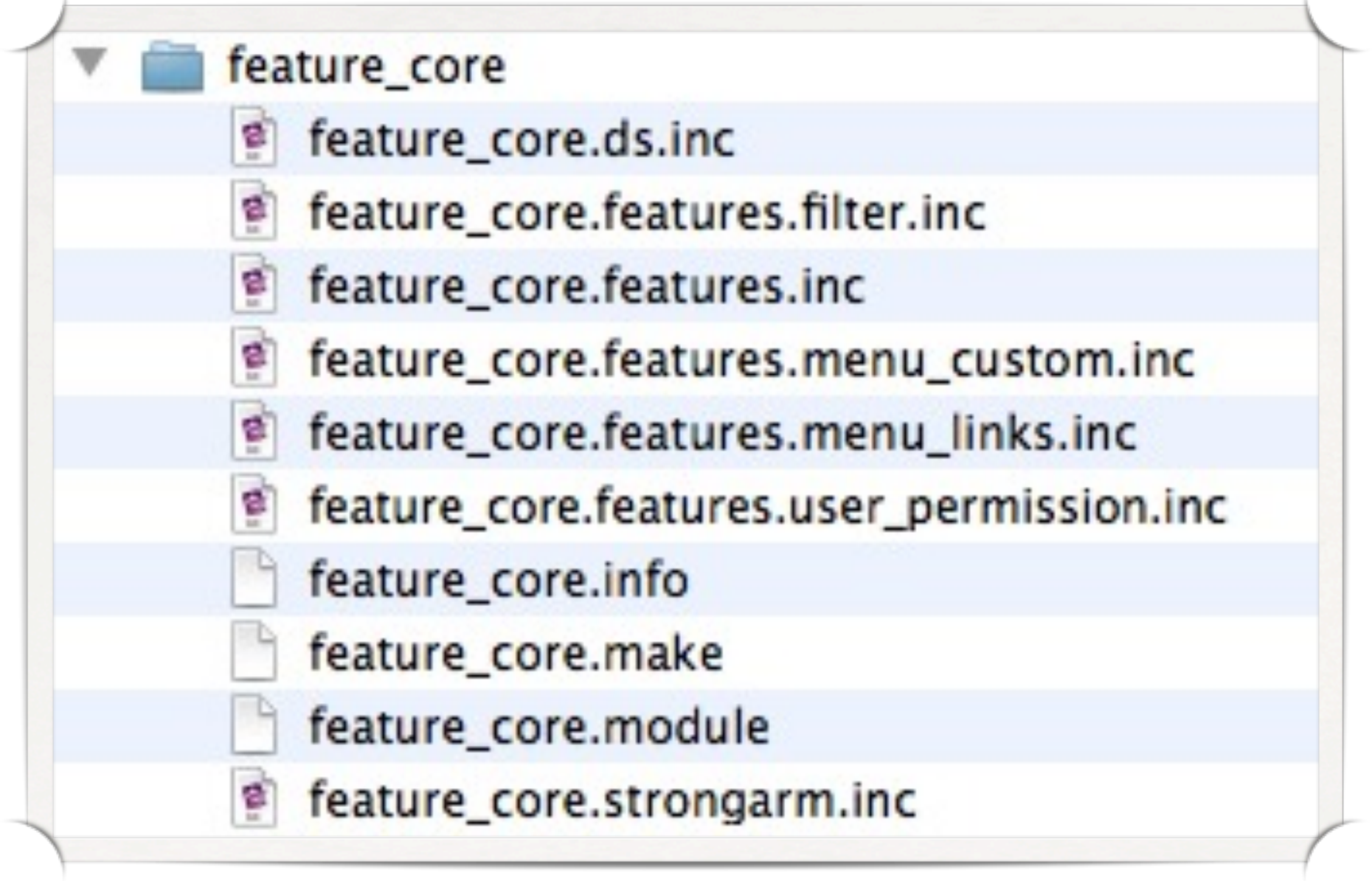
admin_toolbar date_format_long views_ui_show_advanced_column views_ui_show_sql_query views_ui_show_sql_query_where

Dependencies

colorbox ds features insert markdown menu pathauto strongarm token

Normal Auto-detected Provided by dependency

Download feature


feature_core
feature_core.ds.inc
feature_core.features.filter.inc
feature_core.features.inc
feature_core.features.menu_custom.inc
feature_core.features.menu_links.inc
feature_core.features.user_permission.inc
feature_core.info
feature_core.make
feature_core.module
feature_core.strongarm.inc

It’s all in the feature’s .info file

Features are Modules

```
core = "7.x"
description = "Core feature, stuff we need all the time."
dependencies[] = "colorbox"
dependencies[] = "ds"
dependencies[] = "features"
dependencies[] = "insert"
dependencies[] = "markdown"
dependencies[] = "menu"
dependencies[] = "pathauto"
dependencies[] = "strongarm"
dependencies[] = "token"
features[ctools][] = "ds:ds:1"
features[ctools][] = "strongarm:strongarm:1"
features[ds_view_modes][] = "core_small_teaser"
features[filter][] = "core_rich_text"
features[menu_custom][] = "main-menu"
features[menu_links][] = "main-menu:<front>"
features[user_permission][] = "access content"
features[user_permission][] = "use text format core_rich_text"
features[variable][] = "admin_toolbar"
features[variable][] = "date_format_long"
...
```

Features Boundaries

Package functionalities in a logical way

Defining a Feature boundary

- One site, one Feature?
- Well you could, but don't do it!
- Be modular: divide the site functionality into independent pieces
- Several Features work together to build a site

Avoiding transversal Features

- All site permissions into one Feature?
- No: better to bundle permissions with the functionality they belong to
  - Example: if you have a “Blog” Feature, associated permissions should be packaged with it
  - Miscellaneous permissions can be put in a site specific feature.

```
core = "7.x"
name = "Blog"
package = "Features"
description = "Blog for your site."
project = "drupalissimo_blog"

...

features[field][] = "node-blog-body"
features[field][] = "node-blog-field_blog_attachments"
features[field][] = "node-blog-field_blog_image"
features[image][] = "blog-m"
features[image][] = "blog-s"
features[node][] = "blog"
features[user_permission][] = "create blog content"
features[user_permission][] = "delete any blog content"
features[user_permission][] = "edit any blog content"
features[variable][] = "node_options_blog"
features[views_view][] = "blog"
php = "5.2.4"
version = "7.x-1.0-dev"
```

```
core = "7.x"
name = "Blog"
package = "Features"
description = "Blog for your site."
project = "drupalissimo_blog"

...

features[field][] = "node-blog-body"
features[field][] = "node-blog-field_blog_attachments"
features[field][] = "node-blog-field_blog_image"
features[image][] = "blog-m"
features[image][] = "blog-s"
features[node][] = "blog"
features[user_permission][] = "create blog content"
features[user_permission][] = "delete any blog content"
features[user_permission][] = "edit any blog content"
features[variable][] = "node_options_blog"
features[views_view][] = "blog"
php = "5.2.4"
version = "7.x-1.0-dev"
```

Naming conventions

Using Features Effectively

Kit

View | CVS instructions | Revisions

Kit is a specification, it is a set of guidelines that facilitates building compatible and interoperable Features.

- KIT Feature Specification (kitf 1.0-draft)
- KIT Theme Specification (kitt 1.0-draft)

If you are looking to get started with building Kit compliant Drupal sites try Build Kit.

Dedicated to Beth MacWright.

Project Information

Development status: Under active development

Module categories: Features Package

Maintenance status: Unknown
Last modified: September 30, 2010

Downloads

View all releases

Maintainers for Kit

eeks ago

eeks ago

alex_b - 18 commits
last: 39 weeks ago, first: 46 weeks ago

rsoden - 1 commit
last: 40 weeks ago, first: 40 weeks ago

View all committers

Issues for Kit

To avoid duplicates, please search before submitting a new issue.

Search

Advanced search

All issues
7 open, 14 total

Bug reports
0 open, 1 total

Subscribe via e-mail

Code namespace

A feature must, to the best of the creator's knowledge, use a unique code namespace.

Example: `drupalissimo_blog`, not `blog`

Machine name

A feature's machine name need not be unique.

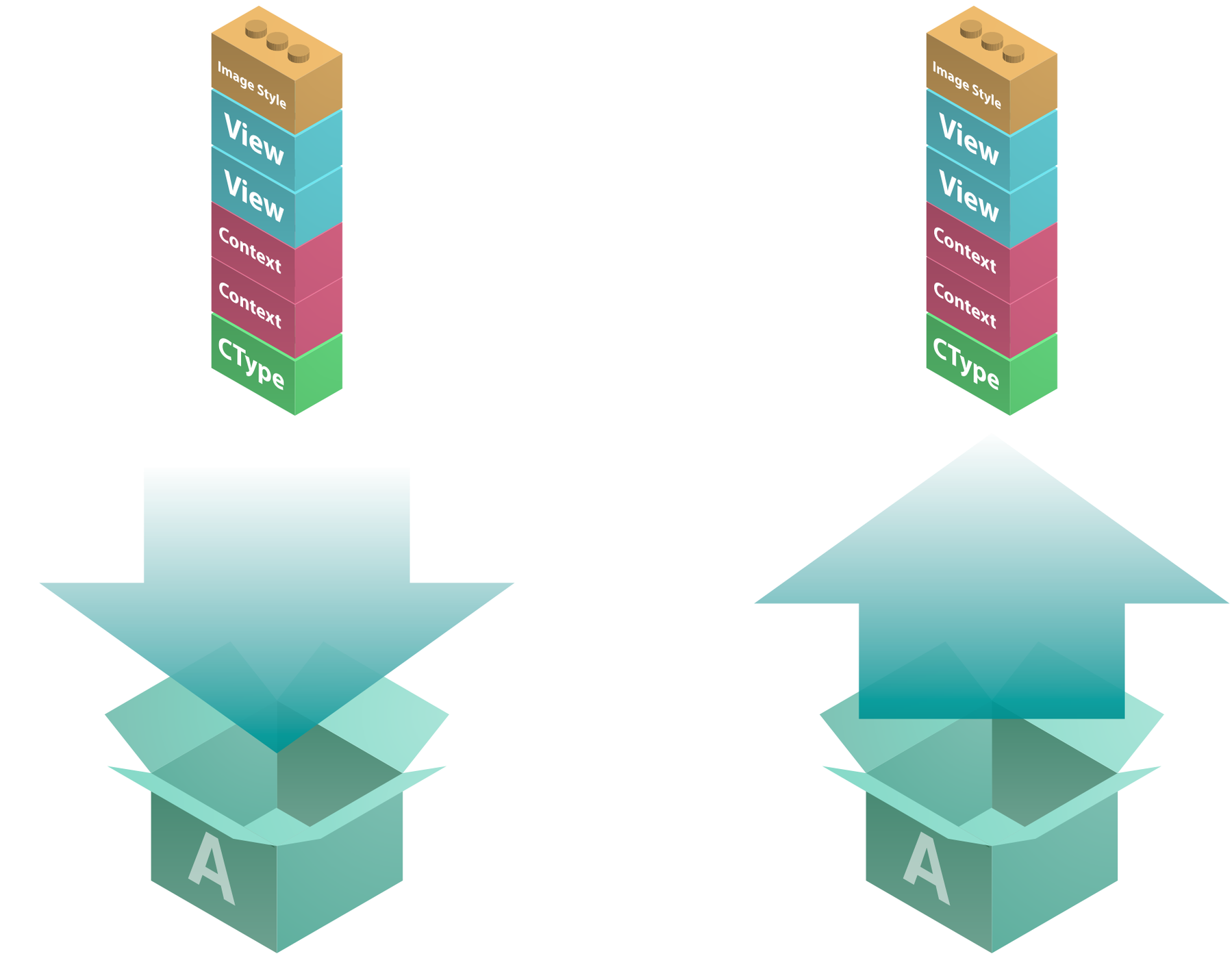
Example: `blog`, `gallery`, `timetracker`

Component namespace

A feature's component namespace need not be unique. Each component name should be prepended by the feature machine name whenever possible.

Example: `blog_listing`, not `recent_blog_posts`

Developing a Feature



Feature Update

Export the current configuration into PHP code

Feature Revert

Enforce the configuration from PHP code

How Features work

A closer look to the Code-Driven Development engine

A component can live in database, in code or both.
Features keeps track of it using a component state.

Component states

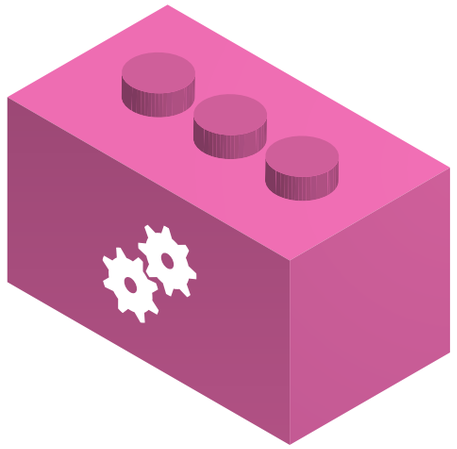
- **Default**: the object has no database entry or the database entry matches the state of the component in code.
- **Overridden**: the code is unchanged with respect to the latest check but the database object does not match the state of the component in code.
- **Needs review**: the previous code state, database state, and current code state all differ.

Features keeps a MD5 hash of:

- **Current code for the component:**
  the configuration as currently represented in code by a given feature.
- **The most recent prior code state that differs from the current code state:**
  if an svn update changes the configuration of a view, this contains a hash of the code as it was before the update.
- **The "normal" component state:**
  the configuration represented by the component as stored in the database or the default component, with any changes introduced by `drupal_alter()`, if no database override exists.

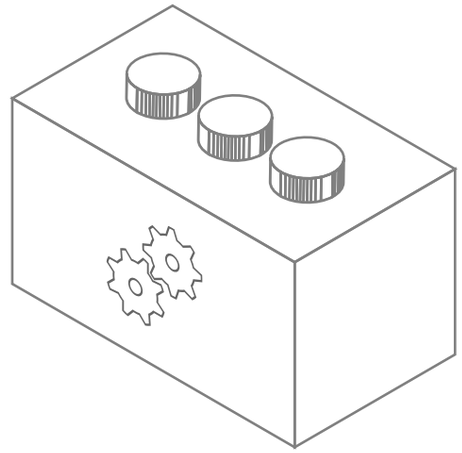
Hard and Soft configuration

What to package into features and what not.



Hard configuration

- Configuration that is under the distribution developer's control *(e.g., Views or Contexts)*
- Stored in Features
- Altering it results in an overridden feature
- Upgrade-unsafe



Soft configuration

- Configuration that is meant to be overridden by the site administrator *(e.g., default theme)*
- Stored in the installation Profile
- Altering it does not change the Features state
- Upgrade-safe

```php
/**
 * Implements hook_install_tasks() callback
 */
function drupalissimo_configure_site_features() {

  ...

  // Enable custom theme
  theme_enable(array('drupalissimo'));
  variable_set('theme_default', 'drupalissimo');
}
```

drupalissimo.profile

Soft configuration: set default theme

Beyond configuration

Taxonomy terms and other beasts.

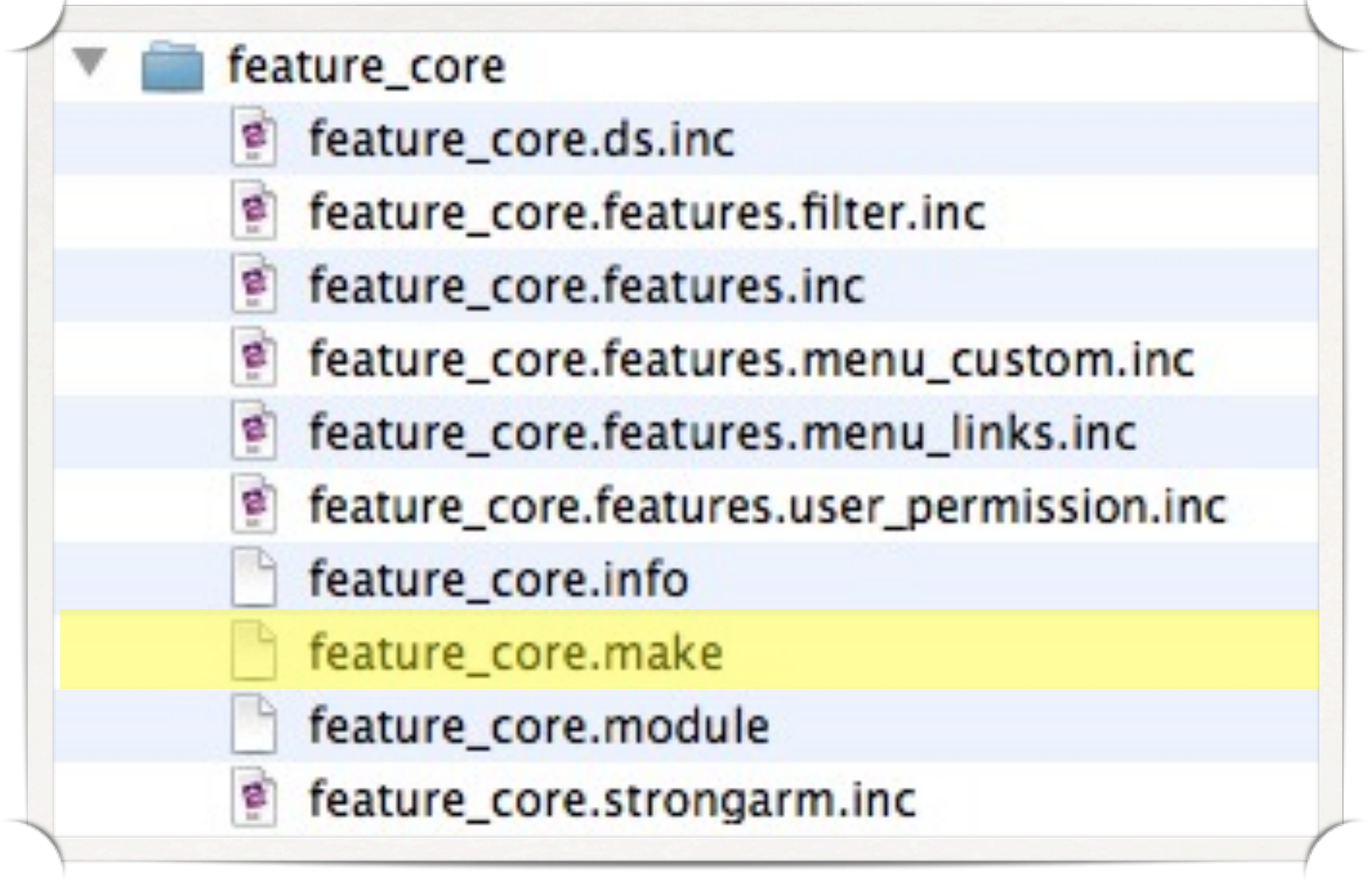
```php
/**
 * Implements hook_install_tasks() callback
 */
function drupalissimo_create_terms() {
  $terms = array();
  $vocabulary = taxonomy_vocabulary_machine_name_load('category');

  $terms[] = 'Solution';
  $terms[] = 'Client';
  $terms[] = 'Use case';

  foreach ($terms as $name) {
    $term = new stdClass();
    $term->vid = $vocabulary->vid;
    $term->name = $name;
    taxonomy_term_save($term);
  }
}
```

drupalissimo.profile

Soft configuration: create initial taxonomy terms

A feature can have a .make file too

Drush Make operates recursively


feature_core
feature_core.ds.inc
feature_core.features.filter.inc
feature_core.features.inc
feature_core.features.menu_custom.inc
feature_core.features.menu_links.inc
feature_core.features.user_permission.inc
feature_core.info
feature_core.make
feature_core.module
feature_core.strongarm.inc

```
api = 2
core = 7.x

; Modules ============================================================

projects[colorbox][subdir] = contrib
projects[colorbox][version] = 1.0-beta4

projects[insert][subdir] = contrib
projects[insert][version] = 1.1

; Libraries ==========================================================

libraries[colorbox_library][download][type] = "get"
libraries[colorbox_library][download][url] = "http://colorpowered.com/colorbox
libraries[colorbox_library][directory_name] = "colorbox"
libraries[colorbox_library][destination] = "libraries"
```

feature_core.make

A feature can specify where to find its own dependencies

Be picky in selecting contrib modules

- Prefer modules that can export their configuration in code.
- Otherwise, try and make configuration exportable.
  - Variables are no problem: Strongarm will help you.
  - If configuration is in tables (and no numeric IDs are around), CTools is your friend.

4

Maintaining a Distribution

Upgrading, the Drupal Way

- Focusing on migrating content is wrong: this is Drupal!
- Focus on upgrading the underlying platform, like a standard Drupal update would do.
- All hooks you need for a smooth upgrade are readily available in Drupal.

Updating Features

Features are modules.
Use `hook_update_N()` in the `.install` file of each feature to:

- Manage the feature's update process.
- Share changes with your development team.

Updating Distributions

Profiles behave like modules.
Use `hook_update_N()` in the `.install` file of the installation profile to:

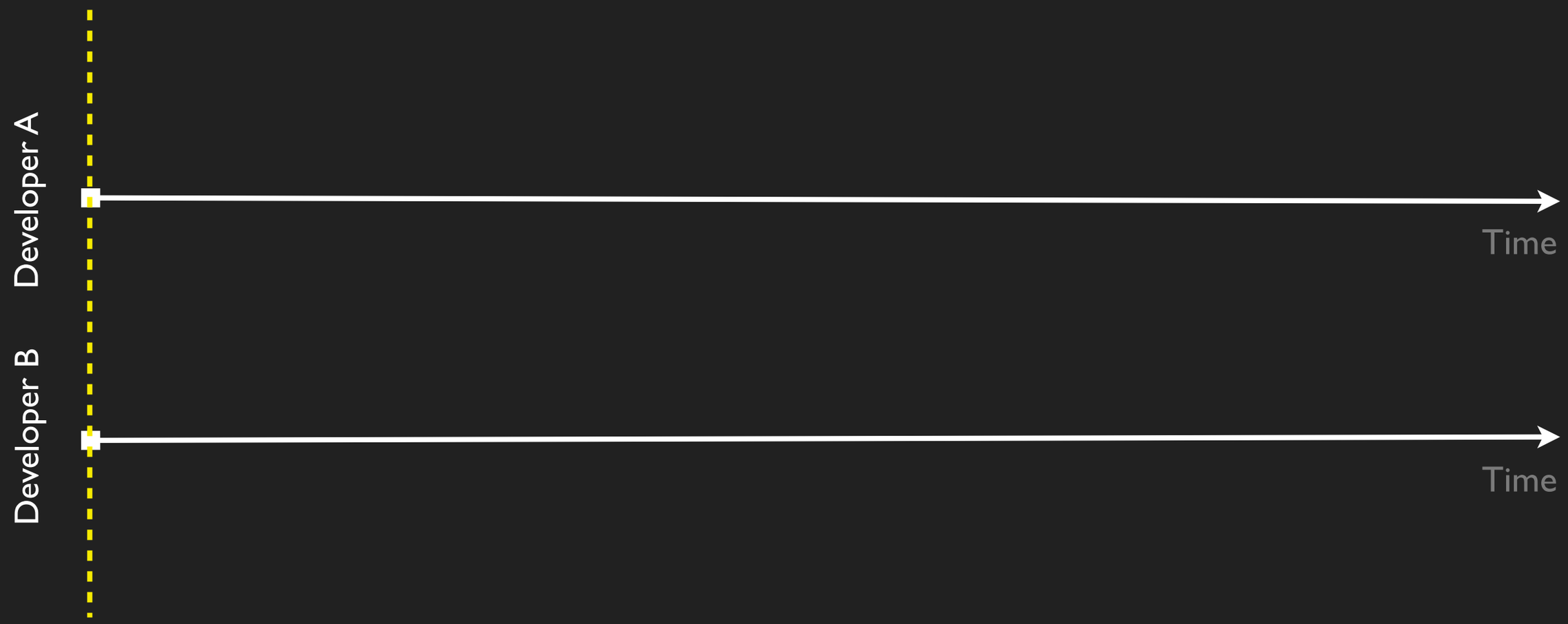
- Manage the distribution's update process.
- Enable new features you have added.
- Handle structural updates.
- Share changes with your development team.

The Code-Driven Workflow

Best practices for a distributed team

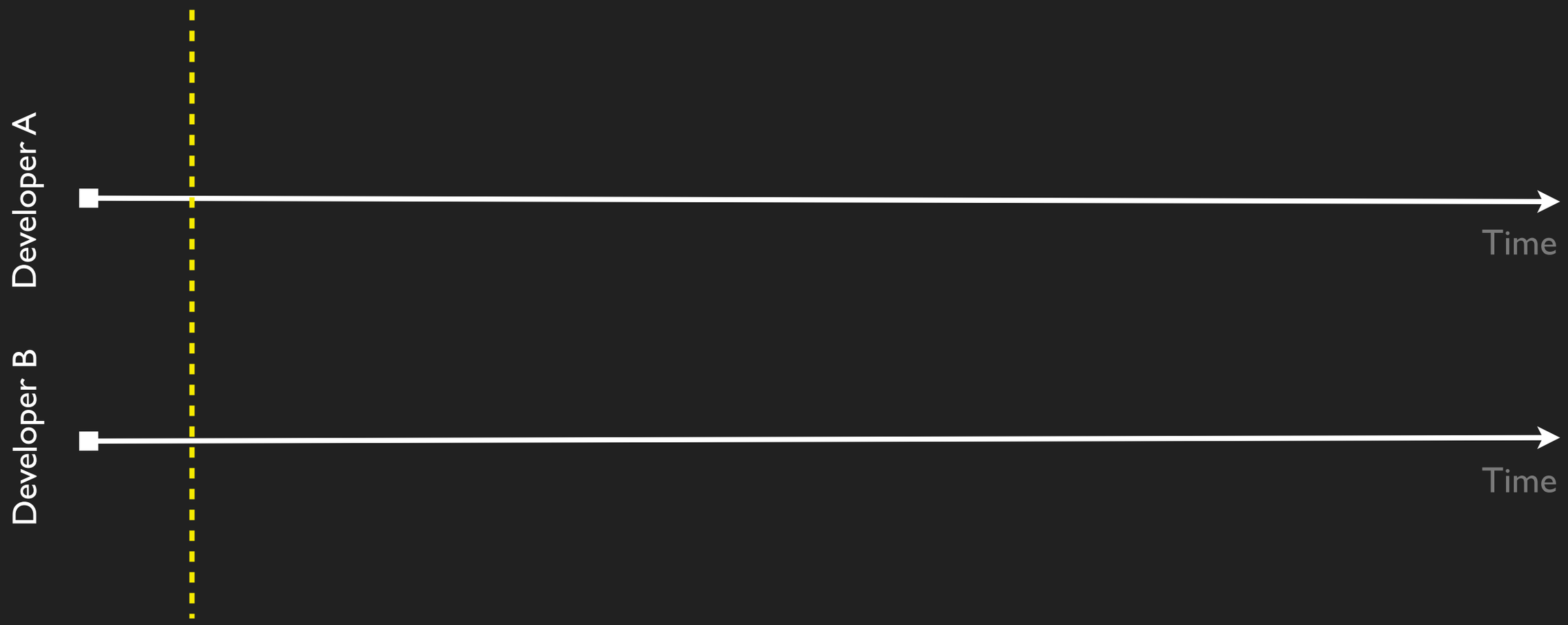
Your mantra: keep everything in code.

Code-driven workflow



Start: both Developers run installation.

Code-driven workflow



Developer A enables Blog feature,
Developer B does other work on the project.

Profiles behave like modules

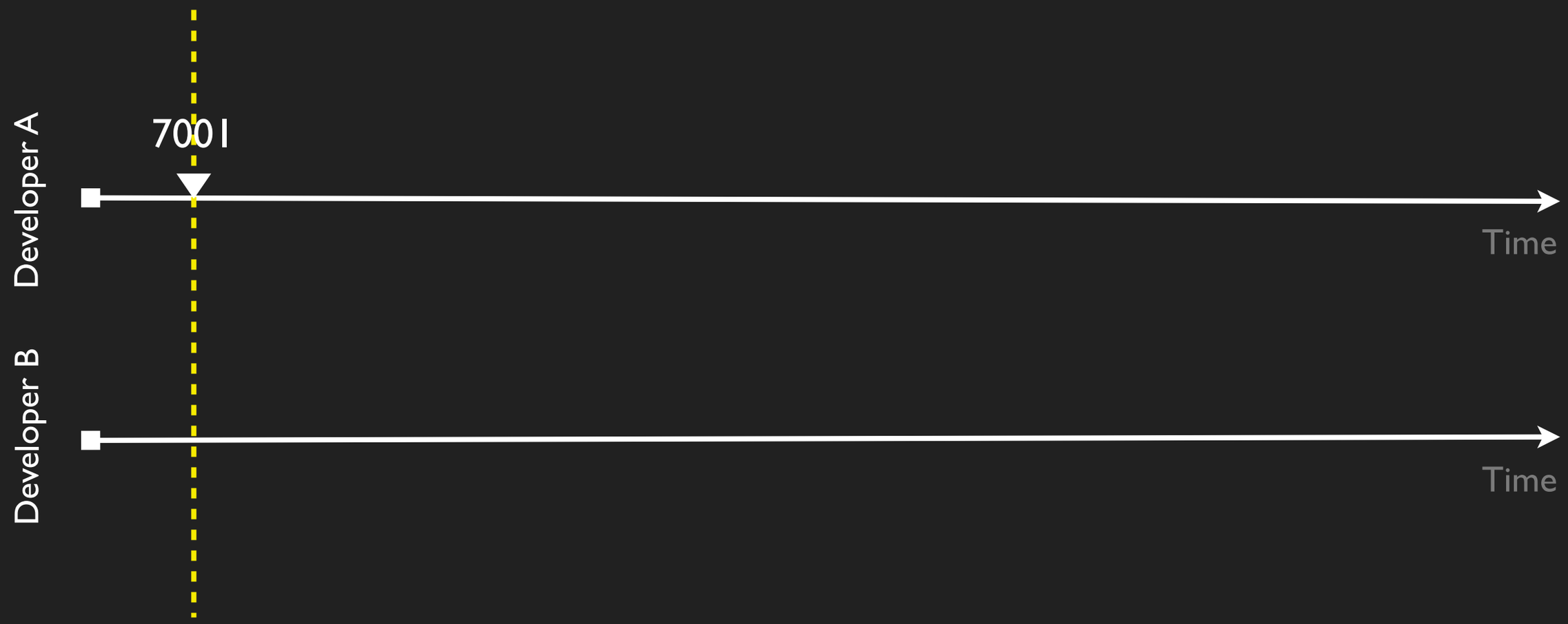
Modules can be updated via `hook_update_N()`

hook_update_N()

```php
/**
 * Enabling Blog feature
 */
function drupalissimo_update_7001() {
  module_enable(array('drupalissimo_blog'));
}
```

drupalissimo.install

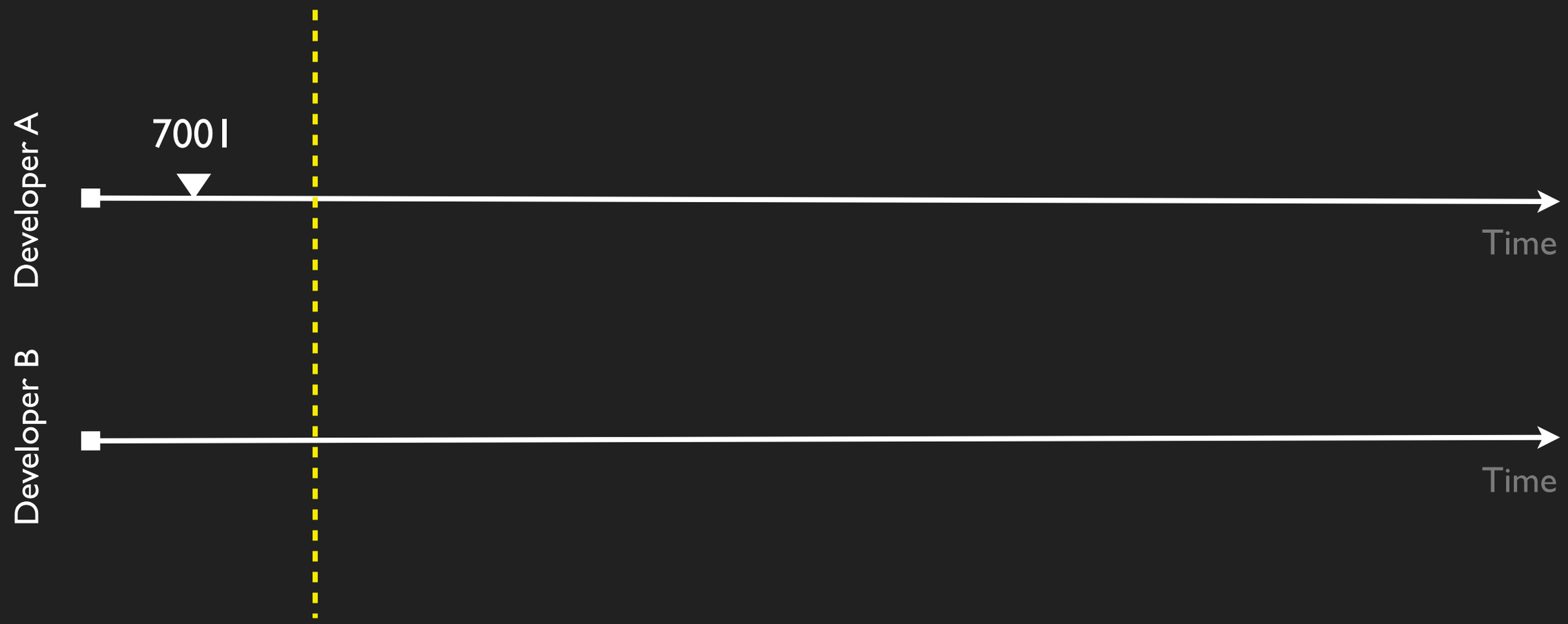
Code-driven workflow




```
DevA$ svn ci -m “Enable Blog feature.”
```

Developer A commits his changes.

Code-driven workflow




```
DevB$ svn up && drush updatedb -y && drush cc all
```

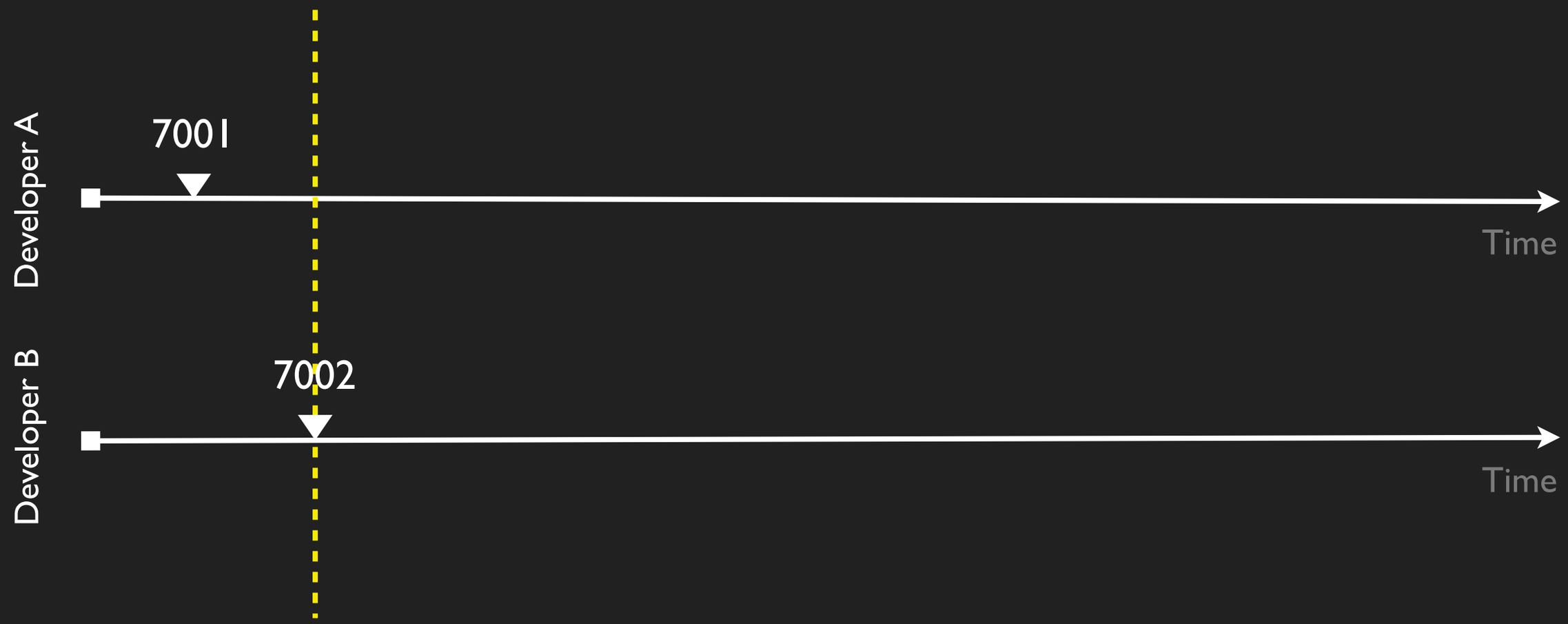
Developer B updates his local copy
and wants to remove a user role.

hook_update_N()

```php
/**
 * Remove editor role
 */
function drupalissimo_update_7002() {
  // Remove user role
  $role = new stdClass();
  $role->name = 'editor';
  user_role_delete($role);
}
```

drupalissimo.install

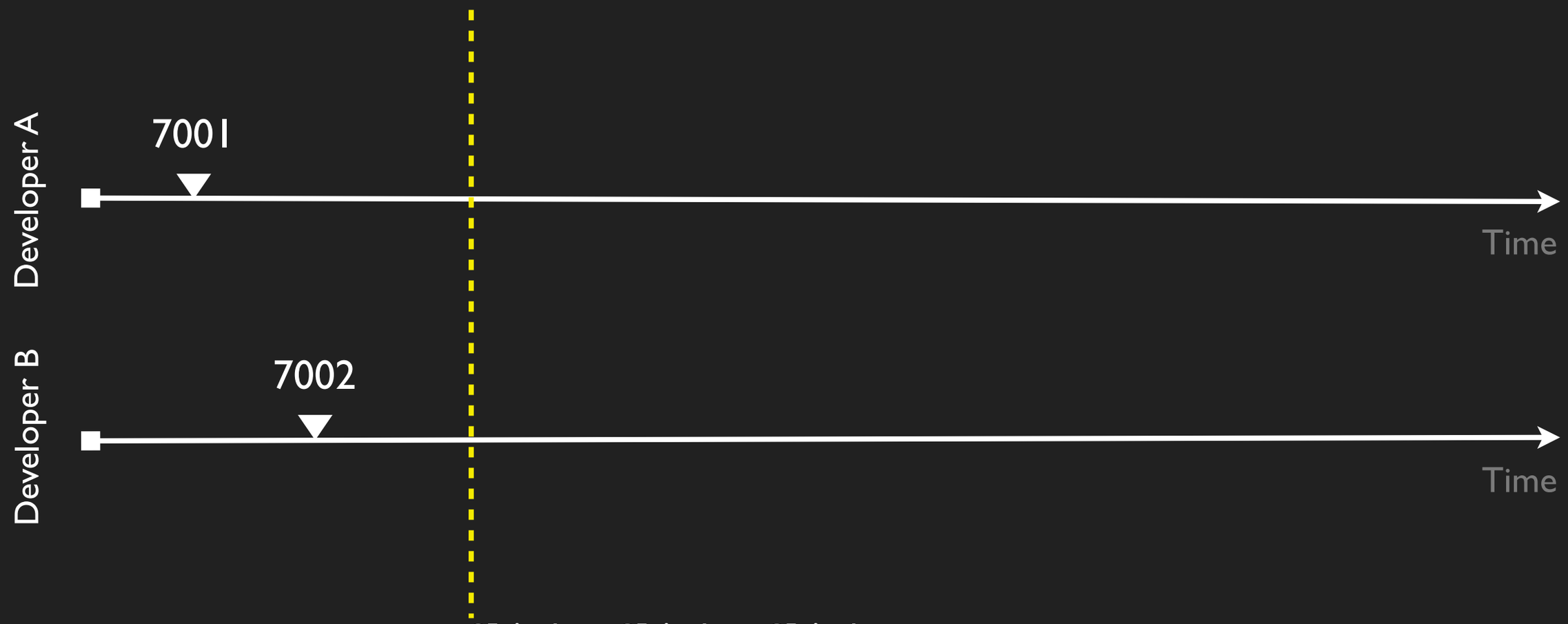
Code-driven workflow




```
DevB$ svn ci -m “Removing editor role.”
```

Developer B commits his changes.

Code-driven workflow




```
Click. Click. Click.
DevB$ patch -p0 < admin.patch
Click. Click. Click.
```

Developer B adds an OpenID account for admin and patches the Admin module.

hook_update_N()

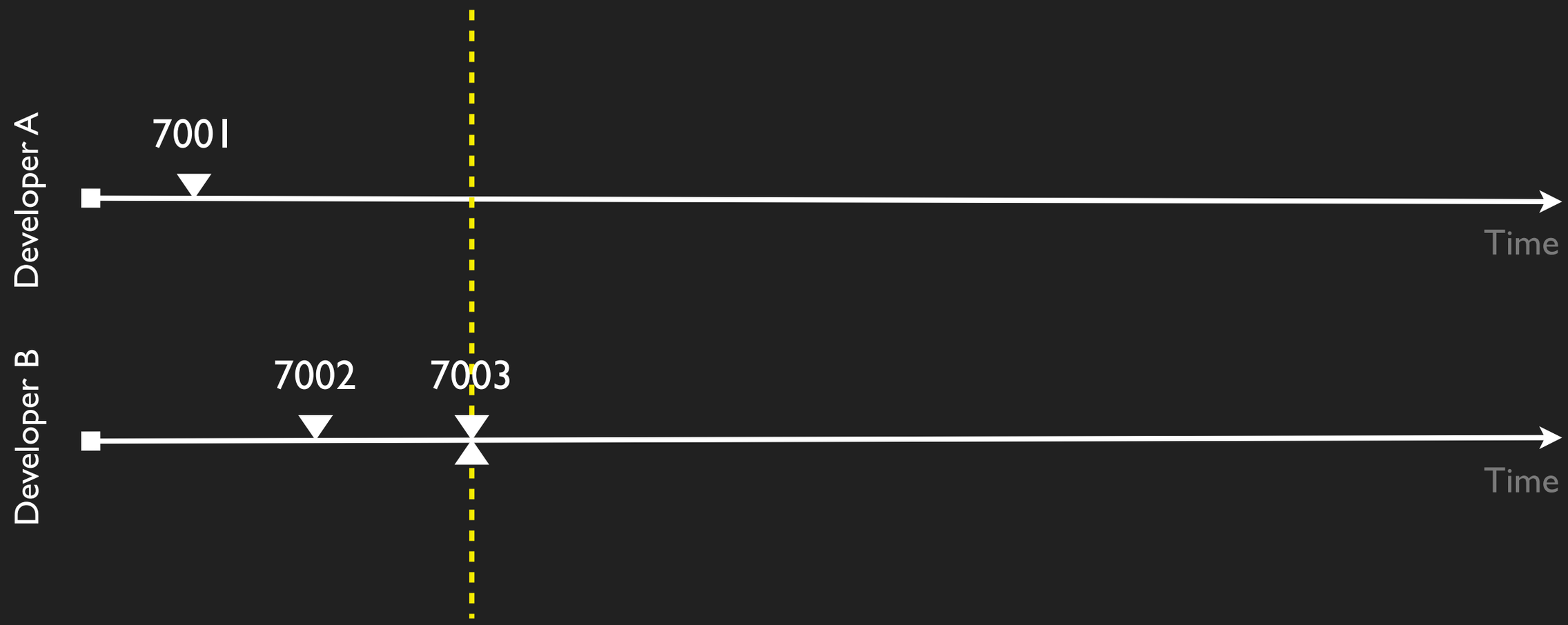
```php
/**
 * Adding Nuvole OpenID to admin account
 */
function drupalissimo_update_7003() {
  $claimed_id = 'http://admin.myopenid.com/';
  $return_to = url('user/1/openid', array('absolute' => TRUE));
  openid_begin($claimed_id, $return_to);
}
```

drupalissimo.install

drupalissimo.make

```
;
; Patches
;
projects[admin][patch][] = "http://drupal.org/files/issues/admin.patch"
```

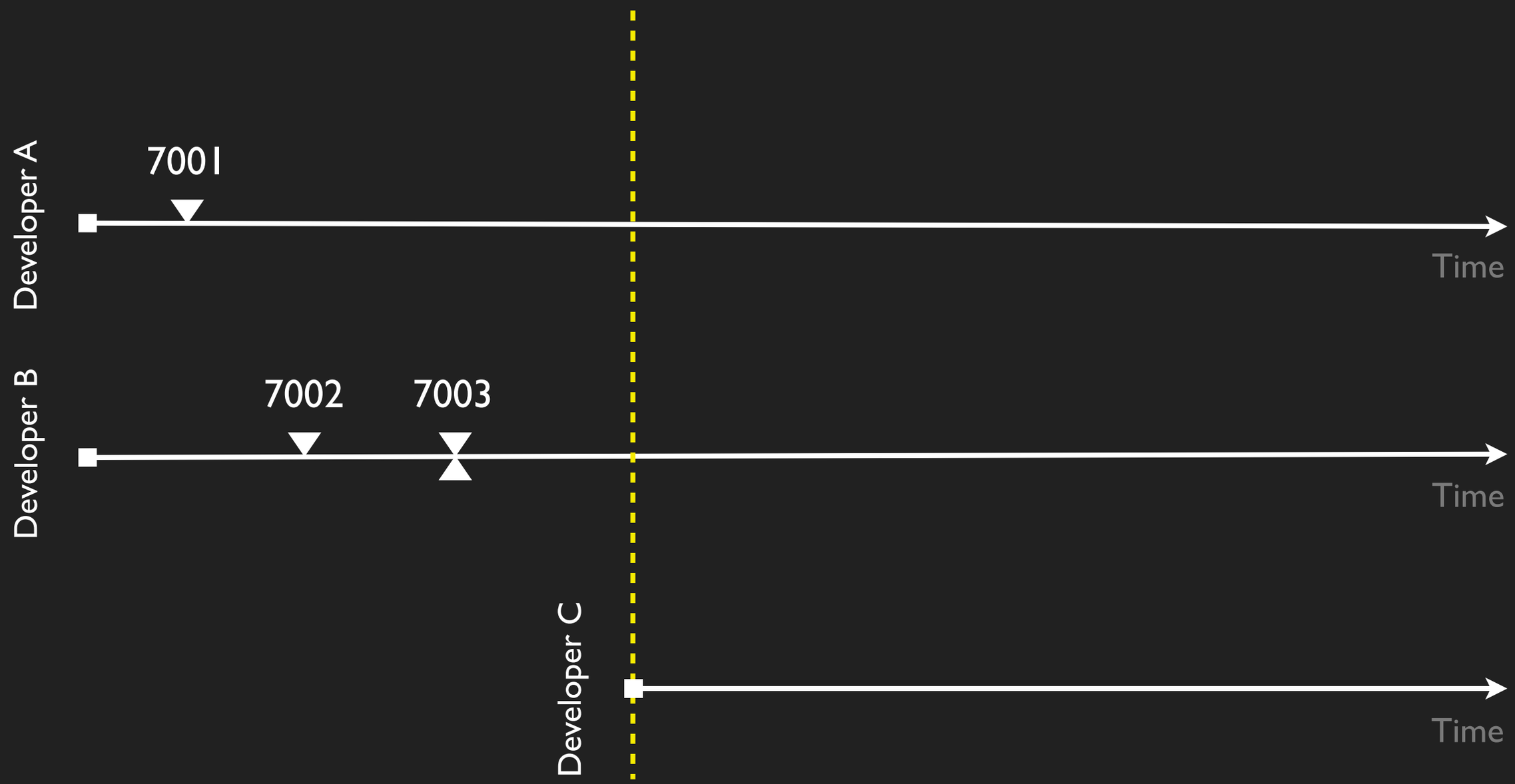
Code-driven workflow




```
DevB$ svn ci -m “Add OpenID for admin. Patching Admin”
```

Developer B commits his changes.

Code-driven workflow



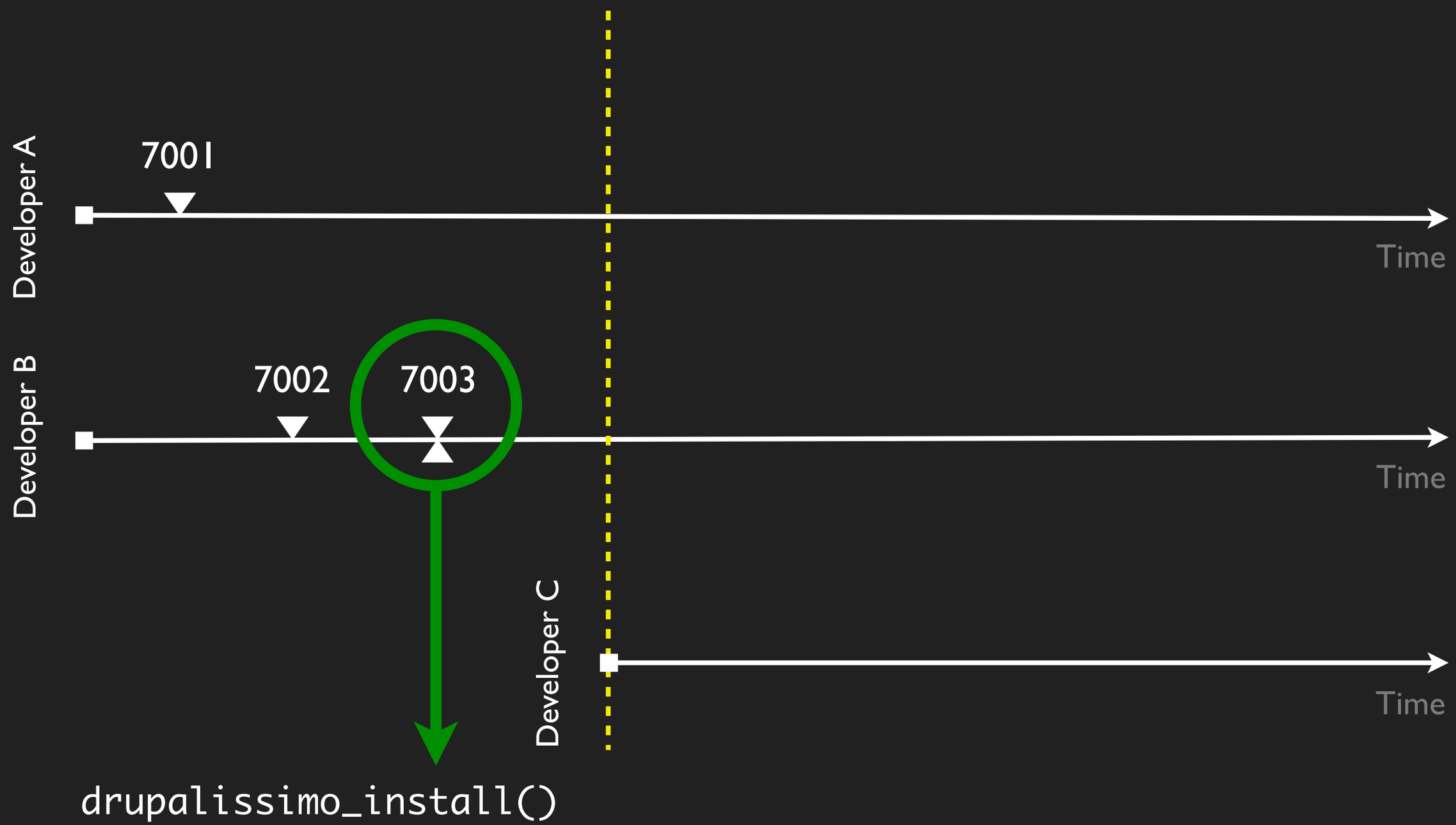

```
DevC$ svn co http://svn.nuvole.org/drupalissimo
```

Developer C joins.

`hook update N()` **is not enough**

Structural updates must go in `hook_install()` too

Code-driven workflow

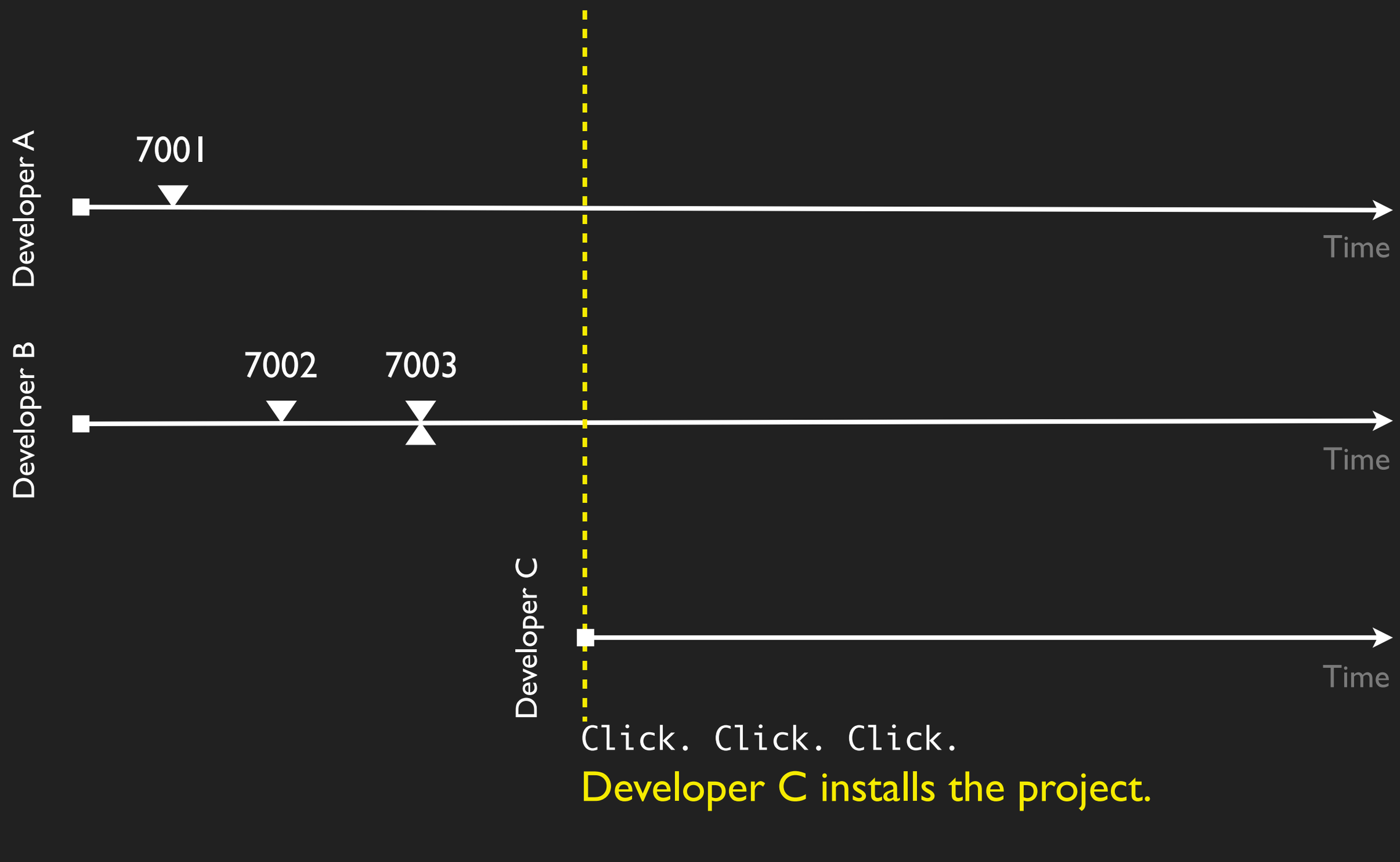


hook_install()

```php
/**
 * Implementation of hook_install()
 */
function drupalissimo_install() {
  // Add OpenID to admin user.
  $claimed_id = 'http://admin.myopenid.com/';
  $return_to = url('user/1/openid', array('absolute' => TRUE));
  openid_begin($claimed_id, $return_to);
}
```

drupalissimo.install

Code-driven workflow



Analysis of the upgrade process in popular distributions.

Practical examples from Open Atrium

```php
/**
 * Implementation of hook_install().
 */
function atrium_install() {
  // Add timestamp index to comments table.
  if (db_table_exists('comments')) {
    db_query("ALTER TABLE {comments} ADD INDEX(timestamp)");
  }
  // Add type,nid index to node table. Allows for more efficient joins t
  // og_ancestry when limiting a view by a certain node type.
  if (db_table_exists('node')) {
    db_query("ALTER TABLE {node} ADD KEY type_node (type, nid)");
  }
}
```

```php
/**
 * Update 6002: Enable new modules.
 */
function atrium_update_6002() {
  drupal_install_modules(array(
    'atrium_groups',
    'atrium_members',
  ));
  return array();
}

/**
 * Update 6003: Fix broken schema version for date_timezone.
 */
function atrium_update_6003() {
  $return = array();
  $status = db_result(db_query("SELECT status FROM {system} WHERE schema
  if ($status) {
    $return[] = update_sql("UPDATE {system} SET schema_version = 5999 WH
  }
  return $return;
}
```

The Code-Driven development cheatsheet

Drupal Code-Driven Development

The Cheat Sheet

NUVOLE

A **quick reference** to concepts and practices in Drupal Code-Driven Development.*

* Hungry for more? Check out http://nuvole.org/blog

Working with Drush

Rule Drupal from the command line. For more: http://drush.ws

```
$ drush fe feature_cool context:feature_cool_front
$ drush fu feature_cool
$ drush fr feature_cool
$ drush fd feature_cool
$ drush vget process
$ drush vset preprocess_css 1
$ drush fe | grep context
$ drush updatedb -y && drush cc all
$ drush dl views
$ drush en views
$ drush upwd admin --password="secr
$ drush sqlq "UPDATE {system} SE
schema_version='6001' WHERE fi
'%feature_cool_module%'"
```

Create a new f… with "feature_cool" in it.

Update a …

Revert … to code.

C…

Spe…

Clone…

```
$ cdd
```

Extending and Overriding

Override default settings. See more at http://bit.ly/featuresapi

```
function hook_user_default_roles_alter(&$roles)
function hook_user_default_permissions_alter(&$permissions)
function hook_content_default_fields_alter(&$fields)
```

Alter the default roles right before they are cached into the database.

Alter the default permissions right before they are cached into the database.

Alter the default cck fields right before they are cached into the database.

CTools also has a variety of `hook_COMPONENT_alter()`.

Naming Conventions

Check out http://drupal.org/project/klt

Component	Naming Convention	Example
View	[machine-name]_[view]	blog_list
Context	[machine-name]_[context]	blog_front
Content Type	[content-type]	blog
Imagecache	[content-type]-[descriptor]	blog-xl

Controller Feature

One feature to rule them all.

```
* Enabling features: feature
*/
```

Add all y…

More on Code-Driven development:

http://nuvole.org/blog

http://nuvole.org/trainings

What did you think?

Locate this session on the DrupalCon London website:

http://london2011.drupal.org/conference/schedule

Click the “Take the survey” link

THANK YOU!