



Formation Drupal 8 Déploiement et Industrialisation

Animée par Romain JARRAUD

Stagiaires et formateur

- **Stagiaires**

- Nom et profil ?
- Comment avez-vous découvert Drupal ?
- Qu'attendez-vous de cette formation ?

- **Formateur Romain JARRAUD**

- Développeur web depuis 1998.
- A commencé par faire du **développement Drupal**, et aujourd'hui fait de l'**animation de formations et du consulting Drupal**.
- Contact : romain.jarraud@trainedpeople.com.

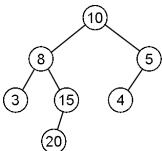
Objectifs de la formation

- **Identifier** les différents éléments impliqués dans un déploiement.
- **Comprendre** le système de gestion de la configuration de **Drupal 8** et ses limitations par rapport au déploiement.
- **Connaitre** les modules additionnels dédiés au déploiement.
- Savoir utiliser le module **Features**.
- Créer un workflow de déploiement/mise à jour avec **GIT**.
- Être capable de créer un script de déploiement.
- Mettre en place une synchronisation de contenus.

Bien commencer

Quelques points de repère

Connaissez-vous Drupal ?



Noeud (node)

0 €



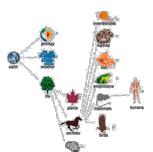
Module

2 001



Thème

+ 4 300



Taxonomie

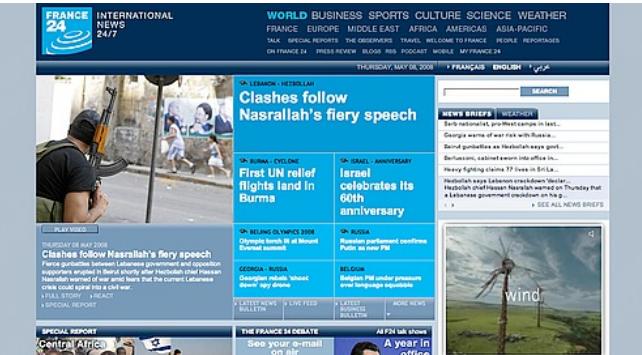
+ 100 000

Made in



2,5%

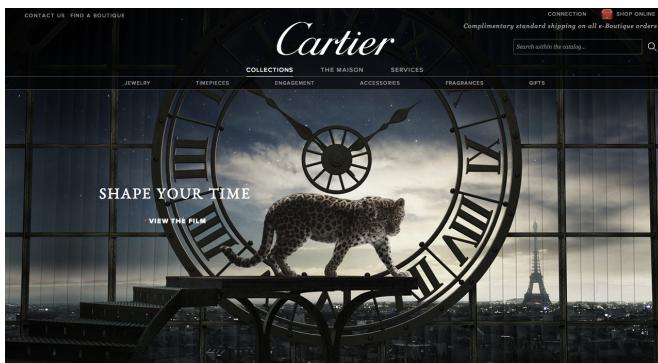
Utilisations de Drupal



Site de news



Site institutionnel



Site marchand



Site divers

Positionnement de Drupal

- **WordPress++**

- Création/gestion d'un site “**à la souris**”, sans développement.
- **Gestion de contenu** élaborée (types de contenu personnalisés, classification, workflow...).
- Écosystème de **modules** pour étendre ou modifier les fonctionnalités de base.
- Possibilité de **personnalisations avancées**, aussi bien graphiques que fonctionnelles.

- **Content Management Framework (CMF)**

- Un **cadre applicatif** permettant de créer des applications web de gestion de contenu.
- Un ensemble d'**APIs** (SGBD, formulaires, contrôle d'accès...).
- Une **architecture modulaire** extensible.

Boîte à outils

Navigateur



Extensions



Editeur



Module Devel



Installer Drupal

Prérequis serveur

- **Serveur Web :**

- Apache 2.x recommandé.
- Activer l'extension mod_rewrite pour les URLs simplifiées.

- **PHP :**

- PHP 5.5.9 au minimum pour D8.
- memory_limit >= 128M (dans php.ini).
- Extensions PHP recommandées : MySQL, librairie GD, XML...

- **Base de données :**

- MySQL 5.1.21 au minimum pour D8.

- **Plus d'infos :** drupal.org/requirements

Pour la formation, nous utiliserons le logiciel **XAMPP**, qui contient Apache + PHP + MySQL.

Récupérer les fichiers

- **Téléchargez** *d8.zip* depuis *trainedpeople.com/d8.zip* et *config_example.zip* depuis *trainedpeople.com/config_example.zip*. Cette archive contient tous les fichiers nécessaires à la formation.
- **Placez** *d8.zip* sur le bureau de Windows et décomptez-le (*clic droit > Winzip > Extraire ici*).
- Cela crée un répertoire *D8* sur le bureau de Windows. **N'y touchez plus !** A chaque fois que nous aurons besoin d'un fichier situé dans ce répertoire, son emplacement sera indiqué ainsi : *d8/repertoire/nom_du_fichier*

Installer XAMPP

- **Installez XAMPP :**
 - **XAMPP** est gratuit ; on peut le trouver sur www.apachefriends.org.
 - Installez **XAMPP** (conservez tous les réglages par défaut lors de la procédure d'installation).
 - Vérifiez que vous voyez bien la page d'accueil de **XAMPP** dans votre navigateur, à l'adresse localhost/.
- **Remarque :** avec **XAMPP**, l'emplacement par défaut des sites que vous créerez en local (sur votre disque dur) est C:\xampp\htdocs.

Préparation à l'installation

Les étapes suivantes ne sont pas obligatoires car **Drupal** les réalise automatiquement s'il en a les droits.

- **Créez une base de données vierge :**

- Lancez **phpMyAdmin** depuis l'interface de **XAMPP**.
- Créez une base de données appelée **drupal**, avec interclassement = utf8_general_ci.

- **Préparez le terrain :**

- Récupérez **D8/Logiciels/drupal.zip** qui contient *Drupal 8.x*.
- Décompressez *drupal.zip* et copiez le répertoire obtenu dans *C:\xampp\htdocs* (de façon à avoir *C:\xampp\htdocs\drupal*).
- Dans le répertoire */drupal/sites/default*, dupliquez le fichier *default.settings.php* et renommez cette copie en *settings.php*.
- Rendez le fichier *settings.php* accessible en écriture.
- Créez le répertoire */sites/default/files* et rendez-le accessible en écriture.

Installer Drupal

Lancer l'assistant d'installation :

- Dans votre navigateur, visitez localhost/drupal/ et lancez l'installation **Standard**.
- Saisissez les informations de **base de données** : Nom = **drupal** ; Utilisateur = **root** ; Mot de passe = **<<AUCUN>>**.
- Le titre du site est *Développement*.
- Définissez le **superutilisateur** (*Site Maintenance Account*) : Nom = **admin** ; Mot de passe = **admin**
- Une fois l'installation terminée, rétablissez les permissions par défaut sur le fichier **settings.php**.

Après l'installation...

- Vérifiez que **votre site s'affiche sans message d'erreur** à l'adresse *localhost/drupal/*.
- Vérifiez que **vous pouvez vous déconnecter/reconnecter** avec le compte **admin/admin**.
- Sur vos sites en développement **créez un hôte virtuel** pour pouvoir accéder à votre site via une URL locale telle que *monsite.local*.

Installer des modules

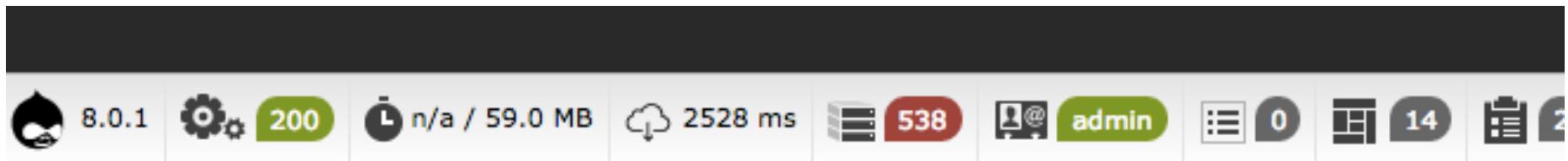
- Les modules sont présents physiquement dans le répertoire **/modules**. Il est possible de créer des sous-répertoires, par exemple **/modules/contrib** pour les modules téléchargés sur drupal.org et **/modules/custom** pour ceux que l'on développe.
- **Installez les modules suivants :**
 - Activer les modules **Devel**, **Devel Kint**, **Web Profiler** et **Devel Generate** (drupal.org/project/devel).
Contient des utilitaires d'aide au développement du site (raccourcis, création de contenus automatique...).
 - Installer les modules **Admin Toolbar** et **Admin Toolbar Extra Tools** (drupal.org/project/admin_toolbar)
Administration du back-office du site sous forme de menu déroulant.

Module *Devel*

- Le module ***devel*** (drupal.org/project/devel) propose un ensemble d'outils dédié au développement (vider les caches, réinstaller un module...) sous forme d'une suite de module :
 - ***Devel*** : module de base.
 - ***Devel generate*** : permet de générer du contenu, des utilisateurs, des termes de taxonomie... lors du développement. Tous les contenus peuvent ensuite être supprimés.
 - ***Devel Kint*** : Pour afficher dans la zone de messages le contenu d'une variable : ***kint(\$variable)***.
 - ***Web Profiler*** : fournit de nombreux outils d'analyse du site : temps d'affichage, nombre de requêtes, mémoire utilisée...
- ***Remarque*** : penser à désinstaller le module ***devel*** en Production.

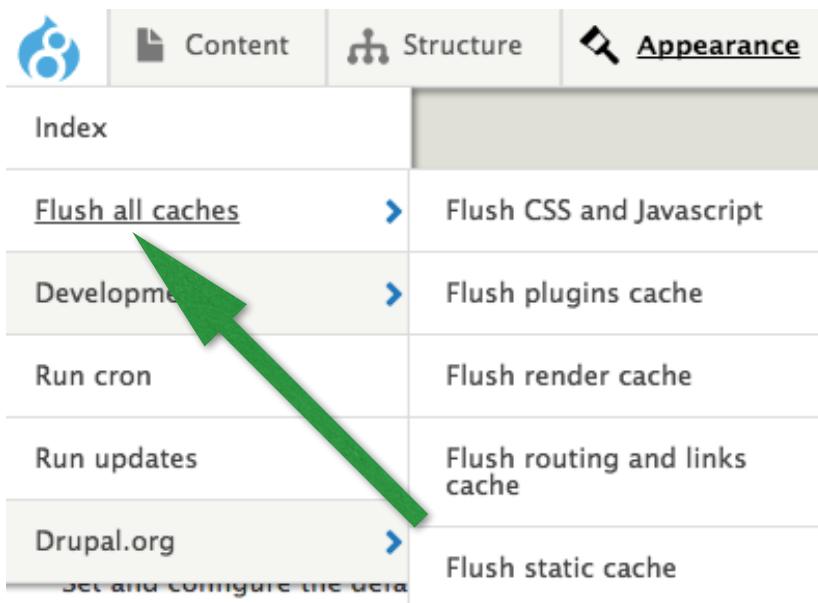
Module *Web Profiler*

- Le module *Web Profiler* fait maintenant partie du module *Devel*.
- Il regroupe un ensemble d'**outils d'analyse** du site : nombre de requêtes, mémoire, PHP info, routing, assets, utilisateur connecté, vues...
- Pour personnaliser les outils disponibles aller sur *Admin > Configuration > Développement > Devel settings > Webprofiler settings*.

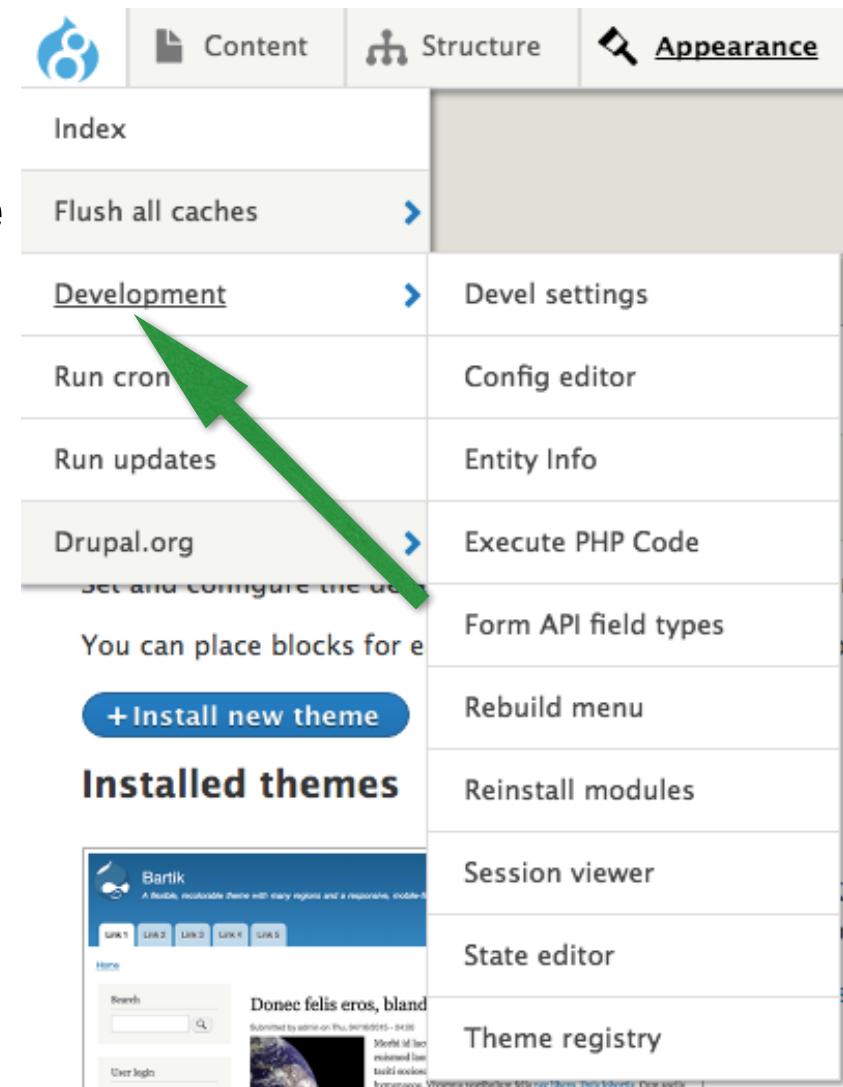


Module Admin Toolbar

Le module **Admin Toolbar** ajoute le menu d'administration déroulant et un certain nombre de liens dans ce menu afin de simplifier l'accès aux pages du back-office.



The screenshot shows a dropdown menu from the Admin Toolbar. It includes links such as "Flush all caches" (with sub-links for "Flush CSS and Javascript", "Flush plugins cache", "Flush render cache", "Flush routing and links cache", and "Flush static cache"), "Development" (with sub-links for "Devel settings", "Config editor", "Entity Info", "Execute PHP Code", "Form API field types", "Rebuild menu", "Reinstall modules", "Session viewer", "State editor", and "Theme registry"), "Run cron", "Run updates", and "Drupal.org". A green arrow points from the "Development" link in the main text to the "Development" link in the screenshot.



The screenshot shows the main Admin Toolbar with links for "Content", "Structure", and "Appearance". Below these are links for "Index", "Flush all caches", "Development" (which is underlined), "Run cron", "Run updates", "Drupal.org", and "Theme registry". A green arrow points from the "Development" link in the main text to the "Development" link in the screenshot. The "Development" link in the screenshot is highlighted with a blue box.

Déploiement

- Un site **Drupal** est l'association entre une base de données et des fichiers.
- Le déploiement est l'art de mettre à jour un site (staging, pré-production, production...) sans perte depuis un environnement local.
- Lors d'un déploiement, de nouvelles fonctionnalités et/ou contenus doivent être intégrés sur les différents environnements.
- Il est **impossible de faire des dumps** de base de données, car on perd alors tous les changements en production (ajout de contenus, création de configuration à chaud...). De plus nous allons voir que la configuration de chaque environnement est particulière.
- Le **déploiement** doit donc se faire à partir de **fichiers** que l'on peut versionner.

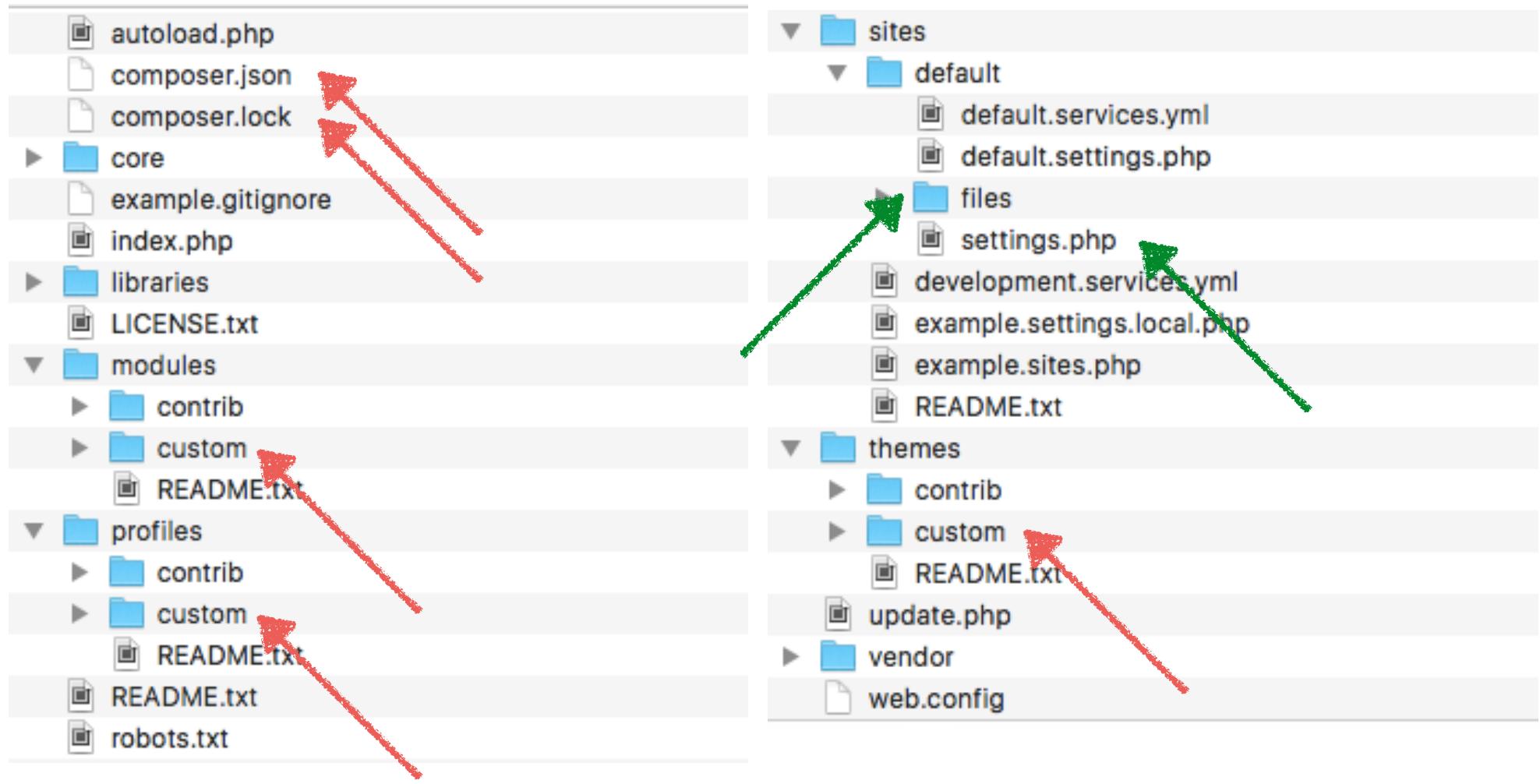
Déploiement - structure des fichiers d'un site

- On peut séparer les fichiers d'un site en plusieurs catégories :
 - cœur de *Drupal*.
 - fichiers de settings (*/sites/default/settings.php*...).
 - modules, thèmes et profiles contrib.
 - modules, thèmes et profiles custom.
 - librairies PHP externes.
 - librairies JS externes.
 - fichiers statiques de contenu (images, PDFs...).
- Seuls les fichiers de code custom et les fichiers statiques (uploadés par les utilisateurs) sont non récupérables depuis un dépôt de fichiers. Tous les autres peuvent re-téléchargés à n'importe quel moment.

Outils pour le déploiement

- **Drush (Drupal Shell)**: administration d'un site *Drupal* en ligne de commande. A installer sur tous les environnements.
- **GIT** : versionning de fichiers. A installer sur tous les environnements.
- **Composer** : gestionnaire de librairies PHP. Pas obligatoire, mais permet de récupérer tous les fichiers d'un projets à partir d'un fichier */composer.json*.
Composer est utile pour tous les environnements sauf celui de production (on évite de tester le code de nouveau). Dans le cas où l'on utilise **Composer**, il ne faut plus installer de modules via l'interface ou **Drush**. Tout doit être géré par le fichier */composer.json*.

Structure des fichiers



Fichiers propres au site

Fichiers locaux

Système de configuration

Configuration vs Contenu

- L'architecture de *Drupal 8* est largement basée sur la notion d'entité.
- Chaque type d'entité est une structure permettant d'organiser l'information.
- Les différents types d'entités sont regroupées en 2 familles :
 - **entité de contenu** : noeud, taxonomie, utilisateur...
 - **entité de configuration** : vue, style d'image, format de texte...
- Le déploiement entre différentes instances d'un même site (DEV, STAGING, PROD...) concerne principalement la configuration, même si certains contenus peuvent être également déployés.

Config API vs State API

- La configuration du site est soit :
 - exportable avec **Config API** :
 - *Configuration simple* utilisant un formulaire unique (avec éventuellement des traductions) : information de base du site (titre, email...), performance (activation du cache, activation de l'agrégation CSS et JS....)
 - *Configuration multiple* utilisant un formulaire (avec éventuellement des traductions) : les vues, les types de contenu, les vocabulaires de taxonomie...
 - non exportable avec **State API** :
 - date du dernier lancement des tâches planifiées, emplacement des fichiers en cache...
- La *State API* regroupe des configurations propres à un environnement qui ne nécessitent pas d'être déployées.

Gestion de la configuration

- Toute la **configuration** est **stockée en base**, pour des raisons de performances (table *config*).
- Chaque configuration du site correspond à un fichier YAML, par exemple les informations basiques du site sont contenues dans le fichier ***system.site.yml***.
- *Drupal 8* permet d'exporter/importer via le back-office n'importe quelle configuration :
 - Export/import unitaire fichier par fichier.
 - Export/import complet sous forme d'archive comprenant tous les fichiers de configuration du site.

Organisation des fichiers

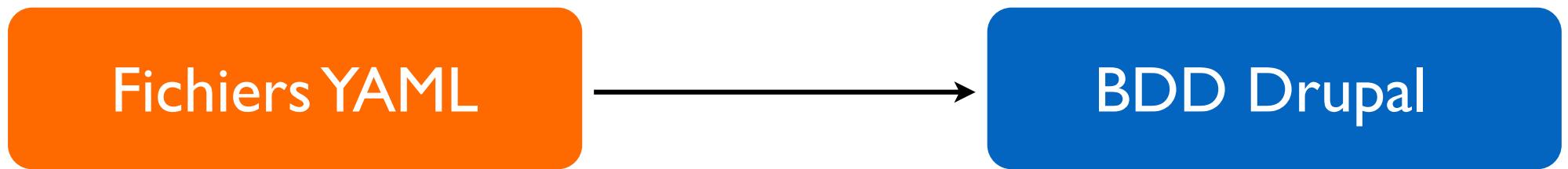
- Afin d'importer des changements de configuration, les fichiers correspondants doivent être placées dans le répertoire

/sites/default/files/config_XXX-sync

- Les fichiers sont donc « exposés », à condition de connaître leurs URLs.
- Il est conseillé de déplacer ce dossier en dehors de la racine du site, afin qu'il ne soit pas accessible de l'extérieur.
- Il est possible de configurer ce répertoire dans le fichier */sites/default/settings.php*.
- **Remarque** : il est possible également d'importer manuellement des configuration via le back-office.

Synchronisation de configuration

/sites/default/files/config_XXX-sync



Synchronisation
Admin > Configuration > Développement >
Synchronization de configurations

Synchronisation de configuration

- Attention la configuration présente dans le répertoire de synchronisation doit comporter 2 fichiers obligatoirement :
 - *core.extension.yml* (modules activés)
 - *system.site.yml* (informations de base du site comme le titre)
- Il est fortement conseillé de placer la totalité de la configuration dans le répertoire de synchronisation, car sinon lors de l'import la configuration en base correspondante aux fichiers manquants est supprimée.
- Cet outil de synchronisation de la configuration est dédié au **déploiement d'un même site** entre différentes instances.

Exporter toute la configuration d'un site

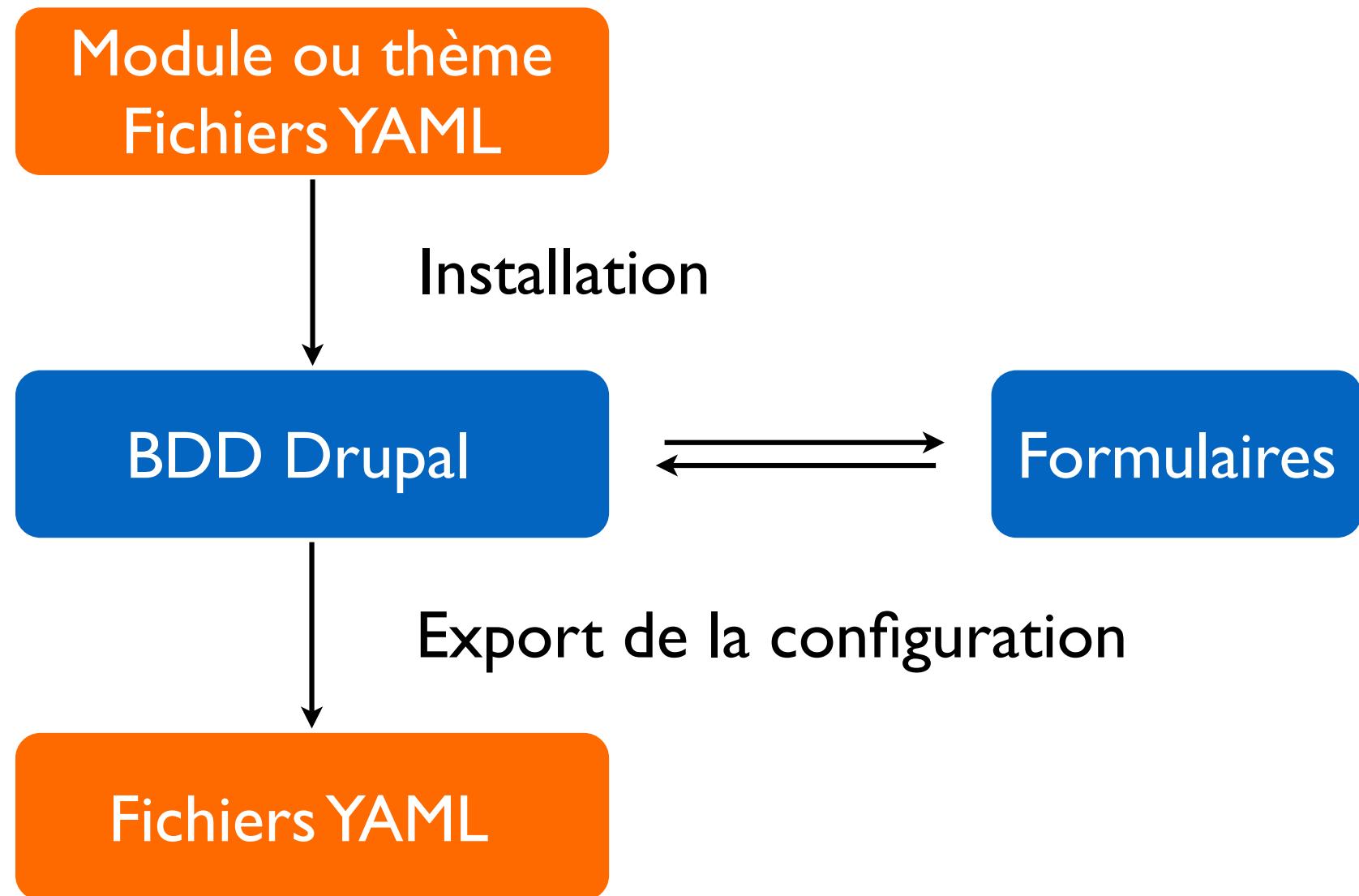
- Sur le site *Développement*, aller sur *Admin > Configuration > Développement > Synchronisation de configuration*, puis sur l'onglet *Export*. Exporter toute la configuration.
- Modifier le fichier **`/sites/default/settings.php`** afin de remonter le répertoire de synchronisation au niveau de la racine du serveur (**`../config`** par exemple). Vider le cache de votre site.
- Extraire l'archive obtenue et placer tous les fichiers dans le répertoire choisi pour la synchronisation.
- Modifier un des fichiers (par exemple le titre de la vue frontpage **`views.view.frontpage.yml`**) et synchroniser la modification sur *Admin > Configuration > Développement > Synchronisation de configurations*.
- Que remarquez-vous dans chacun des fichiers de configuration ?

Configuration dans un module ou un thème

Configuration dans un module ou un thème

- **Lors de l'installation** d'un module/thème, *Drupal* importe toutes les configurations présentes dans le dossier **/config/install** ou bien **/config/optional** du module.
- Le premier dossier sert à la configuration définie par le module/thème ou bien le cœur de **Drupal**, tandis que le second est destiné à embarquer des configurations externes au module/thème, c'est à dire qui sont définies par des modules non requis.
- La configuration d'un module/thème est supprimée automatiquement lors de sa dés-installation.

Configuration dans un module ou un thème



Configuration Update Base

- Par défaut **Drupal** ne peut pas mettre à jour une configuration embarquée dans un module. Les fichiers de configuration présents dans les répertoires `/config/install` et `/config/optional` ne sont pris en compte et chargés en base qu'au moment de l'installation.
- Le module **Configuration Update Base** permet de pallier cette limitation. Il est accompagné de son UI, **Configuration Update Reports**. Pour découvrir les mises à jour de configuration des modules et thèmes installés aller sur *Admin > Configuration > Développement > Synchronisation de configuration > Updates report*.
- Ce module dépend du module **Configuration Manager**.

The screenshot shows the 'Synchronize' tab selected in the top navigation bar. Below the navigation, there is a breadcrumb trail: Home » Administration » Configuration » Development » Synchronize. The main area is titled 'REPORT TYPE' and 'REPORT ON'. It includes dropdown menus for 'Configuration type' (set to 'All types'), 'Module' (set to 'Admin Toolbar'), 'Theme' (set to 'Bartik'), and 'Installation profile' (set to 'Standard').

Modifier la configuration d'un module

- Installer le module ***Config Example*** (fichiers dans l'archive *config_example.zip*).
- Installer les modules ***Configuration Update Reports*** (dépend des modules ***Configuration Manager*** et ***Configuration Update Base***) et ***Config Example***.
- Aller sur *Admin > Config > Config Example* et choisissez une des valeurs du select list.
- Modifier le fichier */modules/custom/config_example/config/install/config_example.settings.yml* en indiquant une valeur par défaut différente de celle du formulaire.
- Aller sur *Admin > Config > Développement > Synchronisation de configuration*, puis l'onglet *Updates report*. Vérifier que la mise à jour de la configuration par défaut est bien détectée.
- Faites-en sorte de revenir à la nouvelle valeur par défaut.

Commandes Drush

Drush

- Il est possible d'administrer un site **Drupal** en ligne de commande, par exemple *vider les caches, importer/exporter des configurations, installer des modules...*
- Pour installer **Drush** aller sur <http://docs.drush.org/en/master/install/> ou bien installer le via **Composer**.
- Pour la suite de la formation vous devez utiliser la version **8.1.11** au minimum.
- Pour afficher les commandes disponibles : **drush help**.
- La documentation est sur <https://drushcommands.com/>.

```
Drupal version          : 8.3.7
Site URI               : http://default
Database driver         : mysql
Database hostname       : localhost
Database port           : 3306
Database username        : root
Database name            : drupal8
Database                : Connected
Drupal bootstrap         : Successful
Drupal user              : bartik
Default theme            : seven
Administration theme     : /usr/bin/php
PHP executable           : Darwin
PHP configuration        : /Users/romainjarraud/Dropbox/h
PHP OS                  : 8.1.13
Drush script             : /tmp
Drush version            :
Drush temp directory     :
Drush configuration       :
Drush alias files         :
Install profile           :
Drupal root              : standard
Drupal Settings File      : /Users/romainjarraud/Dropbox/h
Site path                 : sites/default/settings.php
File directory path        : sites/default
Temporary file directory path : sites/default/files
Sync config path           : /Applications/MAMP/tmp/php
                                ..../config_deploy
```

Principales commandes

- Afficher les infos générales du site :

drush status

- Vider les caches du site :

drush cr

- Installer un module :

drush en MODULE_NAME

- Dés-installer un module :

drush pmu MODULE_NAME

- Afficher une configuration :

drush cget config.name

- Modifier la valeur d'une configuration :

drush cset config.name variable value

- Importer et exporter la configuration :

drush cim

drush cex

Utiliser *Drush*

- Installer *Drush* sur votre machine.
- Vider le répertoire de synchronisation en supprimant tous les fichiers de configuration **YAML**.
- Exporter la configuration avec la commande appropriée.
- Modifier le titre du site en éditant le fichier *system.site.yml*. Importer la configuration ainsi obtenue avec la commande appropriée.
- Vider le cache.
- Le titre du site est-il bien mis à jour ?

Composer

Composer

- Le cœur de Drupal utilise en permanence **Composer** pour charger les différentes librairies PHP dont il a besoin pour fonctionner, dont sa propre API.
- **Composer** est lui-même une librairie PHP présente dans le dossier `./vendor`. Elle n'est pas spécifique à **Drupal** et est utilisable dans d'autres projets PHP.
- Son utilisation directe par développeur (l'édition du fichier `/composer.json` et l'emploi des commandes de l'exécutable) n'est qu'optionnelle. Il peut avancer sur son projet en ignorant son fonctionnement jusqu'à un certain point.
- Il peut ne prendre la main dessus qu'à partir du moment où il analyse que cela lui est utile, par exemple si des librairies PHP additionnelles sont nécessaires. C'est alors la manière recommandée de les intégrer.
- **Composer** permet aussi de gérer la mise à jour des librairies PHP nécessaires au cœur (**Symfony**, **Guzzle**, etc.), comme le cœur lui-même et les modules contrib.

Composer : principales commandes

- Intégrer une nouvelle bibliothèque :

composer require vendor/library

- Mettre à jour l'ensemble des librairies :

composer update

- Installer toutes les librairies déclarées dans ***./composer.json*** ou ***./composer.lock*** :

composer install

- Par défaut, toutes les librairies gérées sont présentes dans le dossier ***./vendor*** à la racine du projet.

Configuration de Composer: composer.json

- Sa configuration est définie en premier lieu dans le fichier **/composer.json** qui est à la racine du projet.
Il ne fait pas partie du cœur et a vocation à être modifié pour les besoins du projet.
- D'autres fichiers *composer.json*, à divers endroits, viennent le compléter:
 - dans le cœur : **/core/composer.json**
 - dans les librairies déclarées dans le **/composer.json** et présentes dans **/vendor**.

Et éventuellement, si ceux-ci sont déclarés dans le
/composer.json

- dans les modules contrib et custom
- dans les thèmes contrib et custom
- dans les profiles contrib et custom

Composer : l'outil central de gestion des dépendances ?

- Dans l'absolu, *Composer* offre une solution de gestion des libraires plus complète que *Drush*, puisqu'il permet, en plus de gérer *Drupal*, ses thèmes et ses modules, de gérer des dépendances à des libraires PHP tierces. Il est en outre possible de lui faire gérer des librairies JS ou des exécutables «serveur», appliquer des patches, lancer des scripts, etc...
- Le développement d'applications web consistant de plus à plus en l'intégration de divers composants open source, *Composer* s'impose comme l'outil central idéal.
- Il facilite l'intégration de solutions comme ***Solr***, ***Elasticsearch***, ***Selenium***, ***PhantomJS***... et leur maintenance, qui nécessitent des dépendances de natures diverses pour fonctionner.
- Certains développeurs choisissent d'utiliser *Composer* pour gérer le code de l'ensemble de leur projet, cœur, modules et thèmes compris.
- Il est possible de limiter l'utilisation de *Composer* à la gestion de composants spécifiques (par ex. des libraires PHP additionnelles, certains modules de contribution) et continuer à utiliser *Drush* ou le back-office pour le reste, mais attention à la lisibilité et la cohérence du projet !

Composer : l'outil central de gestion des dépendances ?

- Certains développeurs choisissent d'utiliser **Composer** pour gérer le code de l'ensemble de leur projet.
- À noter que **Composer** ne fait que télécharger et mettre à jour les modules et thèmes Drupal. Les commandes ***drush pm-enable module*** et ***drush updatedb*** seront encore nécessaires pour activer les modules téléchargés et lancer les mises à jour associées en base de données.
- **Attention** : **Composer** et ses différentes commandes réclament d'être maîtrisées. Leur utilisation peut faire surgir de nouveaux problèmes.
- Son utilisation pourrait inutilement compliquer la gestion des projets simples.

Composer pour la gestion du cœur, modules et thèmes

- Pour gérer les modules et le cœur par **Composer**, il est nécessaire de retirer l'inclusion du *composer.json* du cœur (**/core/composer.json**) du **/composer.json** :

```
"extra": {  
    //...  
    "merge-plugin": {  
        "include": [  
            "core/composer.json"  
        ]  
    }  
}
```

- Puis de le déclarer comme tout autre dépendance:

```
"require": {  
    //...  
    "drupal/core": "8.*",  
    "composer/installers": "^1.0.21",  
    "wikimedia/composer-merge-plugin": "~1.3",  
    ...  
}
```

Composer pour la gestion du cœur, modules et thèmes

- Indiquer qu'une partie des dépendances sont à récupérer auprès d'un nouveau dépôt <https://packages.drupal.org/8>, plutôt que sur le dépôt par défaut **Packagist**.

```
"repositories": [  
    {  
        "type": "composer",  
        "url": "https://packages.drupal.org/8"
```

- Comme le dossier d'installation par défaut est **/vendor**, on indiquera que le cœur, modules et thèmes **Drupal**, étant d'un certain type, doivent être installés à des emplacements spécifiques:

```
"extra": {  
    //...  
    "installer-paths": {  
        "core": ["type:drupal-core"],  
        "modules/contrib/{$name}": ["type:drupal-module"],  
        "profiles/contrib/{$name}": ["type:drupal-profile"],  
        "themes/contrib/{$name}": ["type:drupal-theme"],  
        "drush/contrib/{$name}": ["type:drupal-drush"],  
        "modules/custom/{$name}": ["type:drupal-custom-module"],  
        "themes/custom/{$name}": ["type:drupal-custom-theme"]  
    }  
}
```

Composer et déploiement

- En local :

- Après l'ajout d'une nouvelle dépendance (**composer require**), la commande composer update est lancée dans l'environnement local. Elle réalise la re-génération d'un **/composer.lock** par la fusion de l'ensemble des fichiers **composer.json** des différentes librairies du projet, contrôlant que les différentes dépendances déclarées sont en cohérence. Les différentes librairies sont alors installées et/ou mises à jour.
- Le fichier **/composer.lock** généré pourra être commité, si, après tests, tout s'avère fonctionner correctement.

- En staging, preproduction, production :

- La commande **composer install** installe les dépendances sur la base du **/composer.lock**.
- Attention: la commande **composer update** ne doit pas être utilisée les instances de dev, staging, prod. Elle risquerait de créer de grosses anomalies.

Composer : que versionner/ packager?

- **/composer.lock** :

Il peut être versionné. Drupal n'en a pas besoin pour fonctionner. Mais il est indispensable si le contenu du dossier `./vendor` n'est pas versionné afin de télécharger les librairies et qu'on a choisi d'imposer de lancer un *build* dans chacun des environnements cibles (local, intégration, pre-prod, prod, etc.).

- **/composer.json** :

Il n'est en soi pas utile en pré-production et production, mais on peut considérer que c'est une source que doivent pouvoir éditer les collaborateurs, et à ce titre il peut être versionné.

- **/vendor** :

On peut considérer qu'il alourdit inutilement le dépôt. On peut à l'inverse considérer que le versionner limite le nombre d'erreurs qui peuvent survenir et réduit la longueur du processus d'update. De plus, l'ignorer implique de s'assurer que *composer* fonctionne parfaitement sur tous les environnements impliqués (firewall, etc.).

- **/modules/contrib, /themes/contrib** :

Mêmes considérations que pour le dossier **/vendor**.

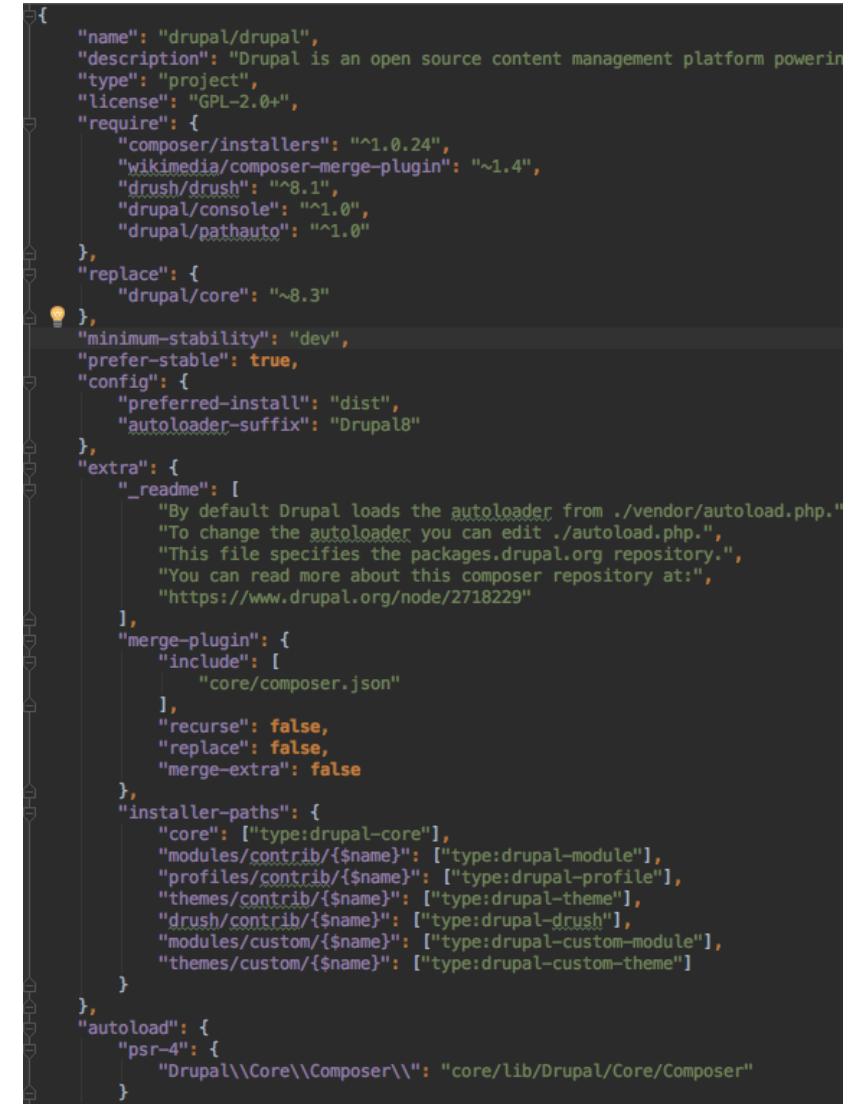
Composer - conclusion

- À vous de décider si vous voulez l'utiliser et avec quelle extension.
- Sur un «vrai» projet, commencez à l'utiliser pour un périmètre limité et allez-y progressivement, car l'outil demande d'être correctement maîtrisé par tous les collaborateurs!

Workflow avec GIT

Avec ou sans Composer ?

- Les fichiers que l'on versionne diffèrent en fonction des environnements et de l'utilisation ou non de **Composer**.
- Typiquement avec **Composer**, on ne versionne que les fichiers custom (modules, thèmes, profiles, librairies...). Tout le reste est récupérable via **Packagist** (ou autre dépôt compatible).
- Notons que **Composer** n'est pas utile en production car cela nécessiterait de refaire des tests.
- Par ailleurs lorsque l'on utilise **Composer**, il est déconseillé d'installer de nouveaux modules ou thèmes via le back-office ou via **Drush**. Toutes les librairies au sens large doivent être listées dans le fichier / **composer.json**.



```
{  
    "name": "drupal/drupal",  
    "description": "Drupal is an open source content management platform powerin",  
    "type": "project",  
    "license": "GPL-2.0+",  
    "require": {  
        "composer/installers": "^1.0.24",  
        "wikimedia/composer-merge-plugin": "~1.4",  
        "drush/drush": "^8.1",  
        "drupal/console": "^1.0",  
        "drupal/pathauto": "^1.0"  
    },  
    "replace": {  
        "drupal/core": "~8.3"  
    },  
    "minimum-stability": "dev",  
    "prefer-stable": true,  
    "config": {  
        "preferred-install": "dist",  
        "autoloader-suffix": "Drupal8"  
    },  
    "extra": {  
        "_readme": [  
            "By default Drupal loads the autoloader from ./vendor/autoload.php.",  
            "To change the autoloader you can edit ./autoload.php.",  
            "This file specifies the packages.drupal.org repository.",  
            "You can read more about this composer repository at:",  
            "https://www.drupal.org/node/2718229"  
        ],  
        "merge-plugin": {  
            "include": [  
                "core/composer.json"  
            ],  
            "recurse": false,  
            "replace": false,  
            "merge-extra": false  
        },  
        "installer-paths": {  
            "core": ["type:drupal-core"],  
            "modules/contrib/{$name}": ["type:drupal-module"],  
            "profiles/contrib/{$name}": ["type:drupal-profile"],  
            "themes/contrib/{$name}": ["type:drupal-theme"],  
            "drush/contrib/{$name}": ["type:drupal-drush"],  
            "modules/custom/{$name}": ["type:drupal-custom-module"],  
            "themes/custom/{$name}": ["type:drupal-custom-theme"]  
        },  
        "autoload": {  
            "psr-4": {  
                "Drupal\\Core\\Composer\\": "core/lib/Drupal/Core/Composer"  
            }  
        }  
    }  
}
```

Exemple de fichier *.gitignore* avec Composer ?

```
1 # Ignore directories generated by Composer
2 /drush/contrib/
3 /vendor/
4 /web/core/
5 /web/modules/contrib/
6 /web/themes/contrib/
7 /web/profiles/contrib/
8 /web/libraries/
9
10 # Ignore sensitive information
11 /web/sites/*/settings.php
12 /web/sites/*/settings.local.php
13
14 # Ignore Drupal's file directory
15 /web/sites/*/files/
16
17 # Ignore SimpleTest multi-site environment.
18 /web/sites/simpletest
19
20 # Ignore files generated by PhpStorm
21 /.idea/
```

Déploiement via GIT

Commandes sur l'environnement **local**

```
drush config-export  
git add .  
git commit -m 'Modification des configurations.'  
git push origin master
```

Commandes sur l'environnement **cible**

```
git pull origin master  
drush updatedb  
drush cache-rebuild  
drush config-import  
drush cache-rebuild
```

Cloner le site

- Dupliquer votre site *Développement* en copiant les fichiers et en exportant/important la base. Le site ainsi obtenu est appelé *Production*.
- Adapter le fichier `/sites/default/settings.php`. (informations de connexion à la base de données) du site *Production*.
- Modifier le nom du site *Production* (Titre = « Production ») sur *Admin > Configuration > Système > Information de base du site*.
- Vérifier que les 2 sites n'utilisent pas la même base de données.

Configuration globale et configuration locale

Configuration locale et configuration globale

- En pratique on dispose de plusieurs instances du même site avec des **configurations** possiblement **differentes**.
- Par exemple un développeur installe des modules en plus pour ses tests et/ou son développement (modules **Devel**, **Devel Kint**...). Il ne peut donc pas directement remonter la configuration de son site local vers une branche commune.
- Certains modules ne devrait pas être activés en Production, par exemple les « UI » modules (**Field UI**, **Views UI**...), **Update Manager**...
- Il faut pouvoir faire des exports partiels de la configuration locale, tout en s'assurant de ne rien perdre par rapport à la branche commune.

Configurations créées en Production

- Empêcher les changements en Production est une « bonne » idée, parfois malheureusement impossible à mettre en pratique.
- Dans certains cas, on est obligé d'autoriser la création de configuration directement en Production.
- C'est le cas des **Webforms**. On peut considérer un site où on laisse la possibilité au webmaster de créer régulièrement de nouveaux formulaires. Or, avec **Drupal 8**, ces formulaires sont enregistrés en tant que configurations.

Configurations créées en Production

- Il ne faut pas risquer de voir ces formulaires supprimés ou écrasés au prochain déploiement.
- Il s'agit de protéger les configurations concernées à l'étape de l'import des configurations sur le site de destination.
- Avant d'importer les configurations en Production, il faut exporter toutes les configurations (incluant les nouvelles). On peut alors les ajouter au git (ou pas), suivant que l'on souhaite ou non les partager sur d'autres environnements.

Configuration locale et configuration globale

Configuration
locale

Configuration
commune

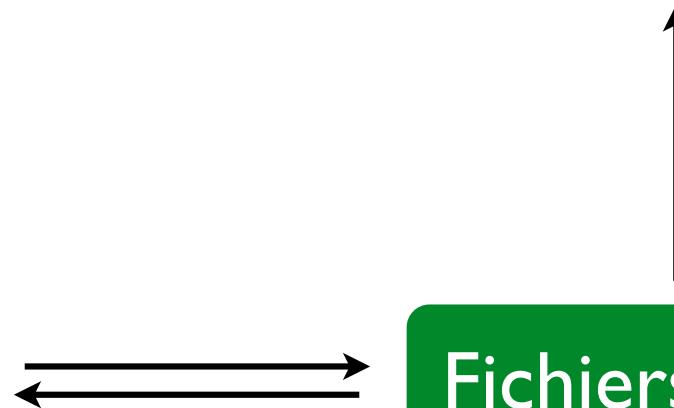
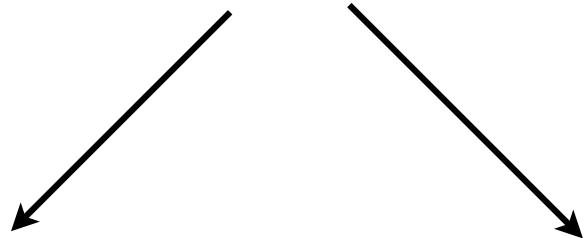
BDD Drupal

BDD Drupal

Fichiers YAML

Fichiers YAML

Fichiers YAML



Configuration locale et configuration globale

Configuration
commune

BDD Drupal

Configuration
Production

BDD Drupal

Fichiers YAML

Fichiers YAML

Fichiers YAML



Split et dossier d'export

- **Configuration Split** fonctionne sur la base du module **Config filter**, qui prend la main sur le canal normal d'export des configurations.
- Le dossier « sync » correspond à la configuration commune par défaut. C'est le dossier déclarer dans le fichier **/sites/default/settings.php** (`$config_directories['sync']`).
- Les dossiers d'export additionnels sont déclarés dans chaque *split*. Le dossier correspondant n'est utilisé que si le *split* est actif.
- Les commandes **drush config-export** (ou **drush cex**) et **drush config-import** (ou **drush cim**) prennent en compte nativement les filtres de *Config Filter*.

Split et dossier d'export

- Le module ***Configuration Split*** permet d'exporter la configuration totale de son site en direction de différents dossiers, en excluant certaines configurations.
- Il propose une interface permettant de créer des *split settings* définissant un dossier de destination et des règles d'exclusion :
 - liste des modules à ne pas activer.
 - liste des configurations à exclure de l'export principal.
- On a 2 types d'exclusion :
 - **configurations non partagées** : **blacklist**. Généralement utilisée pour les développements.
 - **configurations différentes** entre les environnements : **greylist**. Généralement utilisée pour la production.

Configuration des *Splits*

Exemple de *splits settings* possibles :

- *dev* pour les environnements locaux de développement.
- *sync* pour l'environnement de référence.
- *prod* pour l'environnement de Production.

Configuration Split Setting ★

Home » Administration » Configuration » Development » Synchronize

+ Add Configuration Split Setting

CONFIGURATION SPLIT SETTING	MACHINE NAME	STATUS	OPERATIONS
DEV	dev	active	Edit
PROD	prod	active	Edit

conf

 └── dev

 ├── devel.settings.yml

 ├── devel.toolbar.settings.yml

 └── webprofiler.config.yml

 └── prod

 ├── contact.form.feedback.yml

 ├── contact.form.personal.yml

 ├── webform.webform.custom_form_1.yml

 └── webform.webform.custom_form_2.yml

 └── sync

 ├── block.block.bartik_account_menu.yml

 ├── block.block.bartik_branding.yml

 └── block.block.bartik_breadcrumbs.yml

 └── block.block.bartik_content.yml

Split : *blacklist* (exclusion)

- La ***blacklist*** définit les configurations en place dans l'environnement local de développement qui ne doivent pas être transférées en production, donc à exclure de l'export

The screenshot shows the 'Blacklist' configuration screen in the Split interface. It has three main sections:

- Modules:** A dropdown menu containing several module names: 'Datetime', 'Database Logging', 'Devel', 'Devel generate', 'Internal Dynamic Page Cache', and 'Text Editor'. The 'Devel' option is currently selected.
- Blacklist:** A dropdown menu containing configuration keys: 'core.menu.static_menu_link_overrides', 'dblog.settings', 'devel.settings', 'devel.toolbar.settings', and 'editor.editor.basic_html'. The 'devel.settings' option is currently selected.
- Additional Blacklist:** A large text input field where additional configuration keys can be entered, one per line.

Below each section, there is a descriptive message: 'Select modules to filter. Configuration depending on the modules is automatically blacklisted too.' for the Modules section, 'Select configuration to filter.' for the Blacklist section, and 'Select additional configuration to filter. One configuration key per line.' for the Additional Blacklist section.

Split : *graylist* (ignore)

- La ***graylist*** définit les configurations en place dans la production qui ne doivent pas être écrasées à l'import
- Elles seront exportées et fusionnées avec les configurations partagées (ou *globales*) à l'import

GRAYLIST*

Graylist

`config_split.config_split.stage``contact.form.feedback``contact.form.personal``contact.settings``core.base_field_override.node.page.promote`

Select configuration to ignore.

Graylist

Select additional configuration to ignore. One configuration key per line.

Activation d'un *Split*

- Dans pour chacun des environnements, un *split* spécifique doit être activé.
- Souvent, il pourra être utile d'importer la configuration provenant de plusieurs dossiers/splits à la fois, en les « fusionnant ».
- L'activation d'un *split* ne doit jamais se faire via le back-office : lors du déploiement de la configuration correspondante, tous les environnement risqueraient de se voir attribués les mêmes *split*.
- On définit alors dans le fichier ***sites/default/settings.php*** spécifique à chaque environnement quel *split* est activé ou non.

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

Workflow basique

- Sur un environnement de dev, on **exporte** toute la configuration (dans 2 dossiers différents).
- Via **Git** on **partage** le dossier commun *sync* vers un environnement de staging.
- Sur la production, il faut tout d'abord exporter la configuration afin que les fichiers de *graylist* soient à jour et qu'aucun ne manque. Puis on récupère les fichiers du staging via **Git**. Seul le dossier *sync* est mis à jour. Enfin on **importe** toute la configuration (fichiers YAML communs et spécifiques à la production).

Export partiel de la configuration

- Installer le module ***Configuration split***.
- Créer 2 *splits* **Dev** et **Production** et leurs dossiers associés.
- Faire en sorte dans le *split* de Dev d'exclure les modules de développement et leurs configurations liées.
- Faire en sorte sur le split Production d'exclure les configurations liées aux formulaires de contact.
- Spécifiez grâce au fichier ***sites/default/settings.php*** de chaque environnement quel est le *split* actif.
- Tester les commandes *drush config-export* et *drush config-import*. Que constatez vous ?

Surcharger la configuration

Surcharger la configuration

- Pour contrôler l'état des configurations du site sur les différents environnements, on peut juger risqué de ne se fier qu'aux fichiers *YAML* comme source et leur correct filtrage par les *config split/filters*. Ceux-ci peuvent avoir été mal paramétrés ou modifiés par un des collaborateurs à l'export et la livraison des configurations.
- Aussi, il est possible, même si une variable de configuration a été importée qui n'aurait pas due l'être, de «forcer» sa valeur durant le temps d'exécution.

Surcharger la configuration

- Il est possible de surcharger n'importe quelle configuration dans le fichier **/sites/default/settings.php** (ou équivalent).
- Ces éventuelles surcharges n'apparaissent pas dans l'interface du site et ne sont pas exportées dans les fichiers de configuration YAML. Ce sont des configurations locales à l'environnement.

```
617      * configuration values in settings.php will not fire any of the configuration
618      * change events.
619  */
620  $config['system.site']['name'] = 'My Drupal site';
621  # $config['system.theme']['default'] = 'stark';
622  # $config['user.settings']['anonymous'] = 'Visitor';
623
624 /**
625  * Fast 404 pages:
626
```

Surcharger la configuration

- Le fichier **/sites/default/settings.php** est normalement proprement spécifique à chaque environnement, comportant des configurations uniques (ex: accès à la base de données).
- On peut néanmoins mettre à disposition des fichiers déclarant des surcharges de configurations spécifiques adaptées à chaque type d'environnement (local, preprod, prod, ect.), permettant aux collaborateurs d'y accéder et de les modifier à divers fins.
 - ▶ Quand le développement local sollicite un jeu de configurations complexes difficile à gérer individuellement (gestion des split, configuration de solutions de debug, endpoints de webservices «de test» différents de la prod, backend de cache spécifique, etc.). Ce partage sera d'autant puissant s'il est couplée à un environnement mutualisé par virtualisation ou de containerisation (**Vagrant** ou **Docker**).
 - ▶ Quand on veut gérer tout ou partie des configurations de la préproduction ou production depuis le dépôt GIT si un cherche le déploiement le plus automatique possible (attention, risqué!).

Surcharger la configuration

- Le dépôt GIT peut servir à versionner et rendre accessible différents fichiers ***settings.php***.
- Manuellement, dans chacun des environnements, il s'agira alors d'inclure un des fichiers à disposition dans le fichier **/sites/default/settings.php** principal :

/sites/default/settings.local-dev.php

/sites/default/settings.preprod-example.php

/sites/default/settings.prod-example.php

Exemple de surcharges

/sites/default/settings.php

```
'namespace' => 'Drupal\Core\Database\Driver\mysql',
'driver' => 'mysql',
);

# Ajout des configurations partagées
if (file_exists(__DIR__ . '/settings.local.php')) {
  include __DIR__ . '/settings.local-dev.php';
}

# Configurations spécifiques
$config['system.performance']['css']['preprocess'] = FALSE;
$config['system.performance']['js']['preprocess'] = FALSE;
```

/sites/default/settings.local-dev.php

```
$settings['twig_debug'] = TRUE;
$settings['twig_auto_reload'] = TRUE;
$settings['twig_cache'] = FALSE;
$settings['cache']['bins']['render'] = 'cache.backend.null';
$settings['cache']['bins']['dynamic_page_cache'] = 'cache.backend.null';
$settings['cache']['default'] = 'cache.backend.memcache_storage';
$settings['memcache_storage']['debug'] = FALSE;
$settings['memcache_storage']['extension'] = 'Memcached';

$config['system.performance']['cache']['page']['max_age'] = '20000';
$config['system.logging']['error_level'] = ERROR_REPORTING_DISPLAY_VERBOSE;
$config['system.performance']['css']['preprocess'] = FALSE;
$config['system.performance']['js']['preprocess'] = FALSE;
$config['devel.settings']['devel_dumper'] = 'kint';
$config['user.role.anonymous']['permissions'][10000] = 'access devel information';
$config['user.role.authenticated']['permissions'][10000] = 'access devel information';
$config['config_split.config_split.dev_split']['status'] = TRUE;
```

Surcharger la configuration

- Lorsque l'on manipule la configuration dans le code PHP, il faut faire attention à récupérer la valeur adéquate :
 - valeur surchargée (fichier `/sites/default/settings.php`).
 - valeur non surchargée (en base).
- C'est généralement la valeur « réelle » que l'on souhaite utiliser, correspondante à une éventuelle surcharge.

```
// Configuration surchargée par le fichier settings.php.  
$overridden_site_name = \Drupal::config('system.site')->get('name');  
  
// Configuration stockée en base.  
$site_name = \Drupal::configFactory()->getEditable('system.site')->get('name');
```

Configuration dynamique

- De base, le système de configuration suppose que les valeurs sont statiques et ne varient pas en fonction du contexte. Pourtant, la nécessité de définir des valeurs dynamiquement peut parfois survenir.
- Il est possible de surcharger la configuration depuis un module.
- On peut ainsi, par exemple, avoir des valeurs différentes en fonction du rôle de l'utilisateur ou de tout autre contexte.
- Il faut pour ce faire créer un tagged service de type *config.factory.override*, et la classe correspondante implémentant l'interface *ConfigFactoryOverrideInterface*.

/modules/custom/config_override/config_override.services.yml

```
services:
  config_override.override:
    class: \Drupal\config_override\Overrider
    tags:
      - { name: config.factory.override, priority: 5 }
```

/modules/custom/config_override/src/Overrider.php

```
<?php

namespace Drupal\config_override;

use Drupal\Core\Config\ConfigFactoryOverrideInterface;
use Drupal\Core\Config\StorageInterface;

/**
 * Class Overrider.
 *
 * @package Drupal\config_override
 */
class Overrider implements ConfigFactoryOverrideInterface{
    /**
     * Constructor.
     */
    public function loadOverrides($names) {
    }

    /**
     * Constructor.
     */
    public function createConfigObject($name, $collection = StorageInterface::DEFAULT_COLLECTION) {
    }

    /**
     * Constructor.
     */
    public function getCacheableMetadata($name) {
    }

    /**
     * Constructor.
     */
    public function getCacheSuffix() {
    }
}
```

Surcharge de configuration

- Créer le module ***Config Override*** (config_override). Ajouter un service permettant de surcharger la configuration (fichiers de services et de déclaration de la classe correspondante).
- Activer le module précédent.
- Faites en sorte d'avoir un titre de site différent en fonction du rôle de l'utilisateur :
 - pour tous les utilisateurs enregistrés : « Intranet Drupal 8 ».
 - pour les anonymes le titre doit être « Drupal 8 ».

Mode de lecture seule

Mode de lecture seule

- Lors d'un déploiement du site, il est vivement conseillé de mettre le site en mode de maintenance afin de s'assurer qu'il n'y a plus d'accès en écriture sur la base de données.
- En mode maintenance, le site n'est plus accessible (sauf de ceux ayant la permission *Accéder au site en mode maintenance*). Seule une page affichant un message d'avertissement est visible.
- Le module **Read Only Mode** permet de bloquer la création de contenu ou la modification de la configuration du site. Par défaut aucun formulaire n'est accessible.
- Le module **Configuration Read-only** permet lui de ne bloquer que la configuration du site. Aucun formulaire du back-office n'est accessible. Noter que cela nécessite une activation dans le fichier de settings **/sites/default/settings.php** :

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

Module *Read Only Mode*

- Pour passer un site en mode maintenance, aller sur *Admin > Configuration > Développement > Mode maintenance*.
- Le module ***Read Only Mode*** ajoute des options à ce formulaire.
- Le mode de lecture seule est indépendant du mode de maintenance. En pratique on utilise soit l'un soit l'autre.

The screenshot shows two side-by-side configuration pages for 'Maintenance mode'. Both pages have a header with 'Settings' and 'Translate system maintenance' buttons. Below the header, the breadcrumb navigation shows 'Home > Administration > Configuration > Development'. The left page displays a success message: 'Operating in maintenance mode. Go online.' It also contains a note: 'Use maintenance mode when making major updates, particularly modifying a theme, modifying content types, and making backups.' Below this note is a checked checkbox labeled 'Put site into maintenance mode' with the sub-note: 'Visitors will only see the maintenance mode message. Only user login page.' At the bottom of this section is a 'Save configuration' button. The right page shows the same basic structure but includes additional sections: 'Message to display when in maintenance mode' containing the text '@site is currently under maintenance. We should be back shortly.', and expanded sections for 'READ ONLY MODE', 'MESSAGES AND REDIRECTS', and 'ALLOWED FORMS'. Both pages end with a 'Save configuration' button at the bottom right.

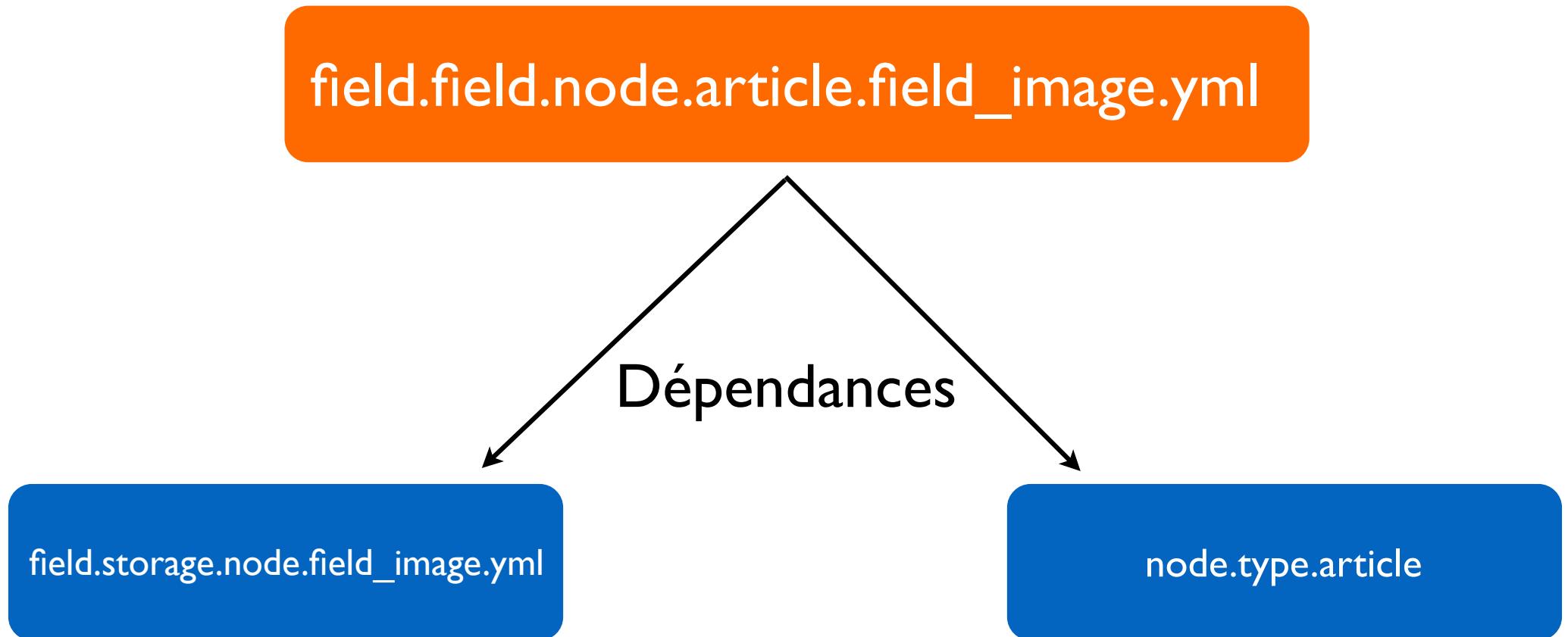
Dépendances de configurations

Dépendance de configuration

- Certaines configurations **dépendent** d'autres configurations.
- Par exemple, un type de contenu pour lequel on a ajouté des champs dépend donc des définitions et des instances de ces derniers. De la même manière, un champ faisant référence à un vocabulaire de taxonomie dépend de ce dernier.
- Les dépendances entre configurations peuvent être communes. Par exemple deux types de contenu ayant un champ en commun partagent la définition de base du champ en question.
- **Les dépendances sont déclarées dans les fichiers YAML correspondants.**
- La chaîne de dépendance peut se révéler être assez complexe.

```
status: true
dependencies:
  config:
    - field.storage.examination.field_candidat
  module:
    - certification
id: examination.examination.field_candidat
field_name: field_candidat
entity_type: examination
bundle: examination
```

Dépendance de configuration



Module Features

Module *Features*

- Le module **Features** a pour but de permettre d'exporter un ensemble de fonctionnalités (type de contenu, vue, rôle utilisateur...) sous forme de package réutilisable, par forcément sur des instances du même site (DEV, STAGING, PREProduction, Production...).
- Chaque fonctionnalité est un ensemble de fichiers de configuration regroupés dans un module. C'est le module **Features** qui génère le code de ce dernier.
- Le module **Features** prend en charge les dépendances entre configurations.
- Pour administrer les différentes features, aller sur *Admin > Configuration > Développement > Fonctionnalités*.

Drush et Features

Quelques commandes *drush* à connaître :

- drush features-list (fl)
- drush features-components (fc)
- drush features-export NAME (fe)
- drush features-update NAME (fu)
- drush features-update-all (fua)
- drush features-revert (fr)
- drush features-revert-all (fra)
- drush features-diff NAME (fd)

Création d'une feature

- Aller sur *Admin > Configuration > Développement > Fonctionnalités*.
- Ajouter une nouvelle feature, puis sélectionner les composants souhaités.
- Générer les fichiers correspondants en cliquant sur le bouton *Download Archive*.
- Vous obtenez alors une archive à dézipper dans le dossier **/modules/features**. Ce dossier est à créer. Il permet une distinction entre les différents type de modules (*contrib, custom, généré avec le module Features*).
- Installer le module correspondant via l'interface de **Drupal** ou via **drush**.
- Retourner sur *Admin > Configuration > Développement > Fonctionnalités*, et vérifier que votre feature est bien listée.

Créer une feature

- Installer les modules ***Features*** et ***Features UI***.
- Ajouter une nouvelle *feature* comprenant tous les composants liés au type de contenu *Article* (type de noeud, champ, permissions...).
- Télécharger cette *feature*, puis supprimer via le back-office les composants correspondants.
- Extraire l'archive dans le dossier **/modules/*features***, puis activer le module correspondant.
- Vérifier que vous avez bien récupérer tous les composants liés au type de contenu *Article*.

Cloner un site

Installer une nouvelle instance d'un site

- Lorsqu'un nouveau développeur rejoint un projet en cours, il est nécessaire qu'il travaille sur une nouvelle instance du site.
- Pour cloner un site, il suffit de récupérer la base de données et les fichiers et de les transférer vers un serveur. Cependant le clone contient tout le contenu du site original (noeuds, utilisateurs, fichiers statiques...). Ceci n'est pas forcément très pratique.
- On désire donc pouvoir dupliquer un site fonctionnellement sans pour autant avoir de contenus.
- Il est nécessaire également que chaque objet de configuration ait le même *UUID* entre les différentes instances du site.
- Il existe le profile d'installation ***Configuration installer*** permettant de créer un site strictement identique du point de vue de la configuration.

Cloner un site

- Dupliquer les fichiers du site principal en supprimant le répertoire **/sites/default/files** et le fichier **/sites/default/settings.php**.
- En allant sur *Admin > Configuration > Développement > Synchronisation de configuration*, exporter toute la configuration du site.
- Télécharger le profile d'installation **Configuration installer** sur *drupal.org* (*drupal.org/project/config_installer*) et extraire l'archive dans le répertoire **/profiles**.
- Installer le site en choisissant le profile et l'archive de configuration précédents.

Drupal 8.1.10

Choose language

Choose profile

Verify requirements

Set up database

Install site

Configure site

Select an installation profile

Standard
Install with commonly used features pre-configured.

Minimal
Build a custom site without pre-configured functionality.
Suitable for advanced users.

Configuration installer
Use configuration from another Drupal site to install a Drupal instance. Suitable for advanced users.

Save and continue

Drupal 8.x-1.3

Choose language

Choose profile

Verify requirements

Set up database

Upload config

Install configuration

Configure site

Configure configuration import location

Synchronisation directory *

sites/default/files/config_Z28BpuLi_rBmlC33y02honfoDb2l

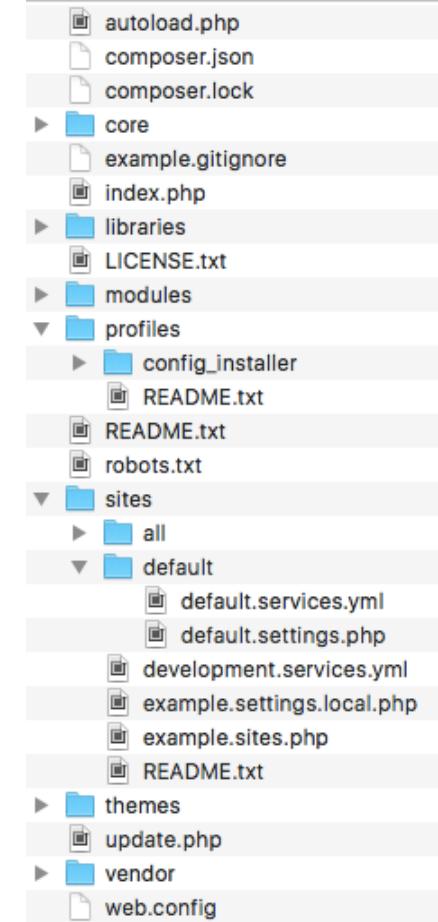
Path to the config directory you wish to import, can be relative to document root or an absolute path.

Select your configuration export file

Choisissez un fichier Aucun fichier choisi

If the sync directory is empty you can upload a configuration export file.

Save and continue



Traductions

Traductions

- Dans un projet non anglophone, en addition des traductions venant de [localize.drupal.org](https://translate.drupal.org), il est courant d'ajouter des traductions réalisées manuellement. Il a fallu traduire les nouvelles chaînes définies dans les modules *custom* ou des chaînes des modules *contrib* qui ne le sont pas.
- Ni le back-office ni Drush ne proposent nativement l'option d'exporter sous forme de fichiers.
- L'extension *Drush* fournie par le projet/module **Drush language** apporte de nouvelles commandes permettant d'export et d'importer les traductions sous forme de fichiers **.po** (un par langue), ceux-ci pouvant être transférés d'un environnement à l'autre.

```
drush language-export fr ./translations/fr.po  
drush language-export fr ./translations/fr.po
```

/translations est ici arbitrairement défini comme dossier d'export.
On peut le placer à la racine du projet ou au dossier supérieur.

Déploiement de contenu

Déploiement de contenus

- On souhaite pouvoir déployer d'un environnement à un autre le contenu, c'est à dire tout ce qui est Content Entity :
 - noeuds
 - termes de taxonomie
 - fichiers
 - commentaires
 - ...
- Comme pour la configuration, il est quasiment impossible de manipuler la base de données directement. Par contre si l'on a des fichiers plats décrivant le contenu, alors on peut les passer d'un environnement à un autre.

Hook_update

- Le ***hook_update_N()*** du fichier *.install* d'un module peut servir à déclarer l'ensemble des ajouts, suppressions et modifications d'entités de contenu (nœuds, termes de taxonomies, etc.) qu'il est nécessaire de déployer.
- Il est possible de répartir ces updates dans différents modules et/ou créer un module spécifique (ex: *project_updates*) où les divers updates sont centralisés.

/modules/custom/project_updates/project_updates.install

```
/**  
 * Implements hook_update_N  
 */  
function project_updates_update_8003() {  
    $em = \Drupal::entityTypeManager()->getStorage('taxonomy_term');  
    $new_terms = [  
        ['name' => 'Finance', 'vid' => 'categories'],  
        ['name' => 'Sport', 'vid' => 'categories'],  
    ];  
    foreach($new_terms as $new_term){  
        $newEntity = $em->create($new_term);  
        $em->save($newEntity);  
    }  
}
```

- Cette solution impose de coder chaque traitement, ce qui demande un travail assez fastidieux, surtout en cas d'entités ou règles de synchronisation nombreuses et complexes.

Déploiement de contenus

- On a intérêt à utiliser les outils s'appuyant sur les services de sérialisation de *Drupal*, permettant d'exporter et d'importer des entités vers et depuis des fichiers plats (ou web services). Les contenus y sont exposés au format *json* ou *xml*.
- Les modules *Deploy - Content Staging*, *Entity share* et *Default Content* fonctionnent sur ce mode opératoire.
- *Default Content for D8*, bien que récent, est celui qui répond au besoin de déployer les contenus créés localement. Il est la fois le plus élémentaire et le plus simple à mettre en place puisqu'il permet de packager les entités à synchroniser de façon semblable aux configurations.

Default Content Deploy

- Le module ***Default Content for D8*** propose une représentation *JSON* de chaque contenu : ainsi chaque entité correspond à un fichier.
- Ce module est destiné en réalité à installer un nouveau site avec du contenu par défaut (d'où son nom!). On crée un module qui contient des fichiers *JSON* de contenus et ceux-ci sont importés en base lors de son installation.
- En complément, le module ***Default Content Deploy*** permet d'avoir un répertoire dans l'arborescence du site contenant des fichiers *JSON* de contenus. Ces derniers peuvent alors être importés/exportés via des commandes ***Drush***.

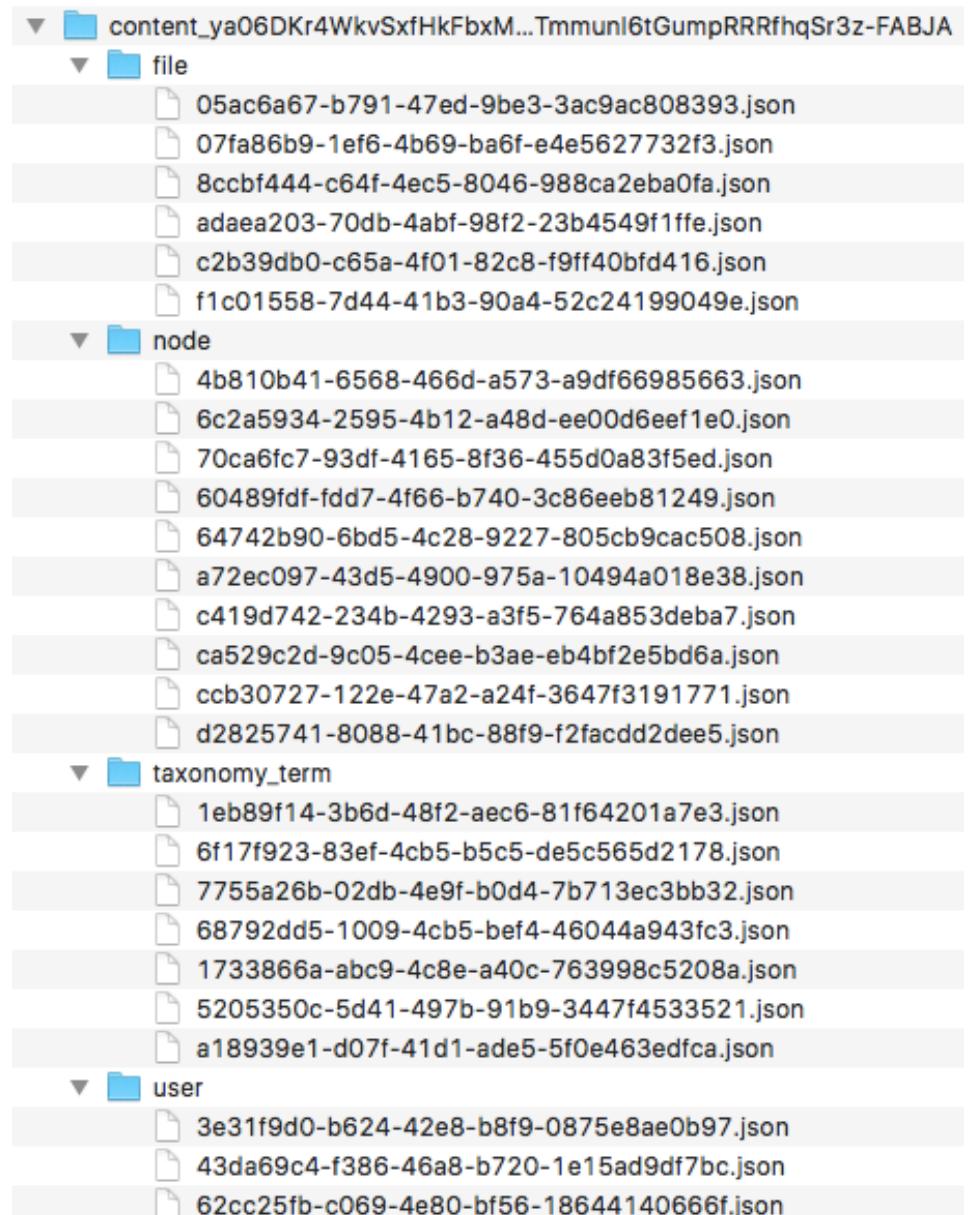
Module *Default Content Deploy*

Le répertoire contenant les fichiers JSON est défini dans le fichier **/sites/default/settings.php** (ou équivalent) comme suit :

```
$config['content_directory'] = '.../content';
```

Si aucun répertoire n'est spécifié alors par défaut c'est le dossier **/sites/default/files/content_ \$KEY** qui est utilisé, où **\$KEY** est la clé de Hash du site.

```
$settings['hash_salt'] = 'ya06DKr4WkvSxfHkFbxMF
```



Default Content Deploy - commandes Drush

- Les différentes actions de synchronisation sont effectuées en utilisant **Drush** :
 - Exporter toutes les entités de type *entity_type* : **drush dcde entity_type**
 - Exporter l'entité de type *entity_type* et d'identifiant 1 : **drush dcde entity_type 1**
 - Exporter toutes les entités de type *entity_type* dans le bundle *bundle_name* : **drush dcde entity_type –bundle=bundle_name**
 - Exporter toutes les entités de type *entity_type* sauf celles d'identifiant 1 et 2 : **drush dcde entity_type –skip_entities=1,2**
- Pour exporter toutes les références des entités également, il suffit d'utiliser la commande **drush dcder** avec les mêmes options que précédemment.
- On peut également exporter l'intégralité des entités de contenu d'un site avec la commande : **drush dcdes**.

Default Content Deploy - synchronisation

- La synchronisation (import) des entités de contenu se fait via ***Drush*** :

drush dc di –verbose –force-update

ou via le back-office sur *Admin > Configuration > Développement > Default Content Deploy - Import.*

- Les mises à jour de contenu sont basées sur la date de modification de l'entité.
- Quels sont les différents cas de figure ?
 - **UUID existant avec ID identique** : mise à jour automatique.
 - **UUID existant avec ID différent** : pas de mise à jour, sauf si l'option « Force update » est sélectionnée dans le back-office (par défaut). Dans ce cas il est indiqué qu'il s'agit d'une création (même UUID et nouvel ID).
 - **Nouvel UUID inexistant** : création de l'entité.
 - **Ancien UUID inexistant suite à une suppression** : l'entité est recréée.

Default Content Deploy - alias

- Il est possible également d'exporter/importer les alias, qui ne sont pas du contenu au sens de ***Drupal***. Les alias ne font pas partie intégrante des contenus.
- Les commandes ***drush*** correspondantes sont :
 - Export : ***drush dcdea***
 - Import : ***drush dcdia***

Déploiement de contenus

- Installer les modules ***Default Content Deploy*** et ***File Entity*** (et leurs dépendances).
- Déclarer dans le fichier de settings de chacun de vos 2 sites, le répertoire d'export de contenus ***../content_deploy***. Ce répertoire est donc commun aux différents environnements afin de simplifier les tests (pas besoin de copier les fichiers JSON).
- Sur votre site *Développement*, générer 5 noeuds, en supprimant tout le contenu existant.
- Vérifier sur le site *Production* qu'il n'y a pas de contenus.
- Utiliser ***Drush*** pour exporter les contenus du site *Développement*.
- Importer ces derniers sur le site *Production*. Comment sont déployer les images?
- Modifier un des contenus originaux (sur le site *Développement*). Synchroniser de nouveau avec le site *Production*.

Déploiement de contenus

- Les modules ***Deploy - Content Staging*** et ***Entity share*** fonctionnent sur un autre mode.
- Ils nécessitent l'activation de web services sur l'instance où est créé le contenu et sur la production.
- Ces modules sont plus adaptés quand les contenus sont créés sur un site de pré-production et qu'on souhaite dissocier leur déploiement en production de nouvelles «livraisons» de code.

Trained People

Trained People c'est aussi :

- des **formations Drupal spécialisées** (Webmaster, Développeur front, Développeur back, Sécurité & Performance, Déploiement & Industrialisation).
- des **certifications à Drupal** (Webmaster, Thèmeur/Intégrateur, Développeur et Expert).
- un programme de **e-learning Symfony/Drupal 8** en partenariat avec **SensioLabs**.
- de **l'accompagnement** durant vos projets (audit, régie...).
- des **recommandations** (freelances, agences, hébergement).



TRAINEDPEOPLE
SensioLabs
UNIVERSITY

Merci !