Learn Drupal 8

01/10/2018

Plan

Plan

Plan

1 Introduction

- Who am I? (-> didier.boff@smile.fr)
- What is Drupal?
- History
- Advantages/Drawbacks
- What's new in Drupal 8?
- What you will learn

Who am I? (-> didier.boff@smile.fr)

- Open Source developer and contributor since 2012 (`https://github.com/B2F`, `https://www.drupal.org/u/b2f`)
- Drupal 6, 7, 8

Who am I? (-> didier.boff@smile.fr)

- Open Source developer and contributor since 2012 (`https://github.com/B2F`, `https://www.drupal.org/u/b2f`)
- Drupal 6, 7, 8
- Companies I worked for
  - Open Web Solution: 2012 - 2014
  - Alter Way: 2014
  - Smile: 2015 - 2018

Who am I? (-> didier.boff@smile.fr)

- Speaks given:
  - 2013: Conference at Drupal Camp Paris: Automatic functionnal testing with Behat
  - 2014: Drupal 7 for CGI (7p) , TV5 Monde (2p)
  - 2016: Drupal 8 for Smile Lille (6p), Drupal 7 for Sarthe department (5p)

Who am I? (-> didier.boff@smile.fr)

- Examples of Drupal websites I worked on:
  - Intranets: TV5 Monde, Renmans
  - Book sellers: Nathan Livres / Nathan Jeux / Dalloz, Bordas
  - Collectivies (regions, departments): Charente Maritime, Sarthe
  - Job boards: Synergie, Aile Medicale, Reso Emploi

Who am I? (-> didier.boff@smile.fr)

- Send me an email:

dibof@smile.fr

Plan

Introduction

- An Open Source Content Management System

Introduction

- An Open Source Content Management System
- A programmation framework

Introduction

- An Open Source Content Management System
- A programmation framework
- Now based on Symfony components (huge leap forward from D7!)

Introduction

- An Open Source Content Management System
- A programmation framework
- Now based on Symfony components (huge leap forward from D7!)
- Drupal is built to be extended using modules, themes etc

Introduction

- An Open Source Content Management System
- A programmation framework
- Now based on Symfony components (huge leap forward from D7!)
- Drupal is built to be extended using modules, themes etc
- OOP knowledges are needed to develop with Drupal 8

Plan

History

- Originally written by Dries Buytaert (Belgium) as a message board
- Drupal became an open source project in 2001
- Dries Buytaert found Acquia in 2007
- Acquia is not editor but a sponsor
- Drupal 8.0.0 release: 19th November 2015
- Last release: Drupal 8.6.1 Sep 10 2018

Plan

Advantages/Drawbacks

- Advantages
  - Popularity (safe choice)
    - As of now, more than 1 000 000 sites used Drupal (170K+ in Drupal 8)
    - https://www.drupal.org/project/usage/drupal
  - Active community
  - Many modules available
  - Very extendable

Advantages/Drawbacks

- Advantages
  - Popularity (safe choice)
    - As of now, more than 1 000 000 sites used Drupal (170K+ in Drupal 8)
    - https://www.drupal.org/project/usage/drupal
  - Active community
  - Many modules available
  - Very extendable
- Drawbacks
  - Too much modules, different qualities make it difficult to choose
    - See active install module rate from a module page
  - Some concepts are not so obvious (taxonomy, hooks, entities)
  - Some coding standards and architecture patterns are old (like hooks)

Plan

- Type of field (in HTML5)
  - Date
  - Entity reference (natively)
  - Telephone
  - Email
  - Link
- Media entities
- Form modes
- Blocks entities: one class to many instances
- Comments entities: not only for nodes now
- Responsive BackOffice

- Symfony components are integrated
  - HTTPKernel + HTTPFoundation
  - Annotation
  - DependencyInjection
  - EventDispatcher
  - ClassLoader (autoloader)
  - Routing
  - YAML
- OOP

- Hooks are going to disappear
- RESTful Web Services
- New cache system
- Configuration management
- Composer
- New Libraries management
  - GuzzleHTTP (HTTP Client)

- HTML 5
- End of IE 8 support: Drupal 8.0.0
- End of IE 10 support: from Drupal 8.4.0 (Change record)
- JS Libraries
  - Jquery 2.x: Drupal 8.0.0
  - Jquery 3.x: from Drupal 8.4.0 (Change record)
  - Modernizr (browser features detection)
  - Normalize.css (make the default browser css iso)
  - Jquery touch punch (touch gesture management)
- Template engine: Twig instead of PHP

Plan

Introduction

Learn to know where and what to search (:source code, official documentations)

Practice from exercises (keep the given document for reference)

Get solutions from a demo Git repositories

Do not hesitate to interupt and ask questions !

Thanks for your attention :=)

Plan

2 Installation and Architecture

- Development environments
- Practice
- Production environements
- Server Architectures
- Practice
- Security advices

Plan

Development environments

- Shared containers (recommended)
- Optionnal container technologies listed from lower to higher level:
  - LXC
  - Docker
  - Virtual Box
  - Vagrant
- Apache 2.2.x, Nginx 1.4.x or higher
- PHP 5.5.9 or higher
- Database
  - MySQL 5.5.3/MariaDB/Percona or higher
  - PostgreSQL 8.3 or higher
  - SQLite 3.3.7 or higher

`https://www.drupal.org/requirements`

Plan

Practice

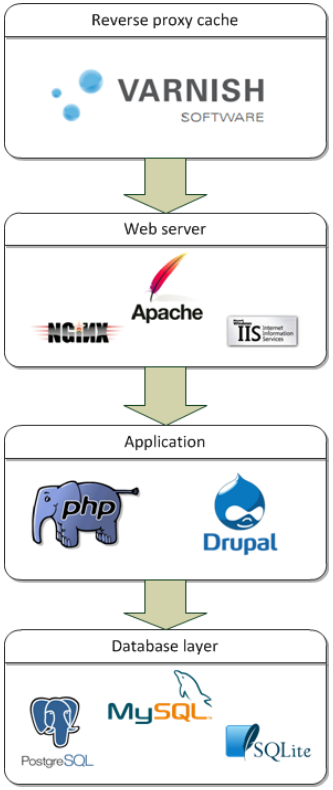
- Install Drupal from given instructions

Note

If you have a memory limit error, increase temporarily your php cli `memory_limit` to 1500M

Plan

Production environements

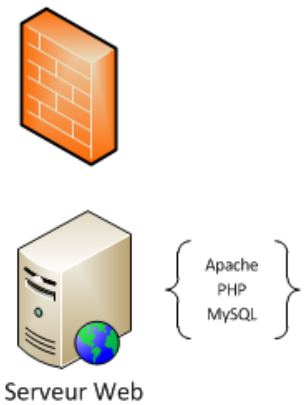


- Reverse Proxy Varnish (Recommended) and/or Nginx (required for SSL)
- Apache 2.2.x, Nginx 1.4.x or higher
- PHP 5.5.9 or higher
- APC or OPCache (OPCache is native from PHP 5.5+)
- Database
  - MySQL 5.5.3/MariaDB/Percona or higher
  - PostgreSQL 8.3 or higher
  - SQLite 3.3.7 or higher
- Cache backend (supported via modules)
  - Redis
  - Memcached

`https://www.drupal.org/requirements`

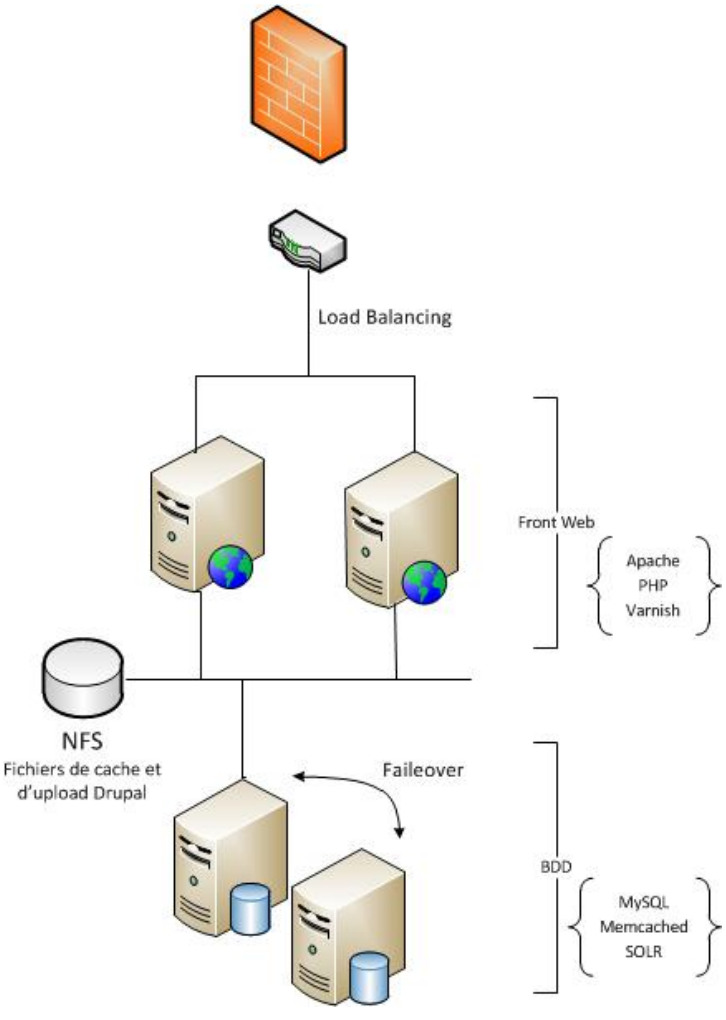
Plan

Server Architectures: Minimal



- 1 Front server
  - Apache
  - PHP, APC
  - MySQL
- Good CPU
- Good RAM

Server Architectures: Multi Front



- 1 Load balancer
- n Front servers (CPU Needed)
  - Varnish
  - Apache
  - PHP, APC
- 1 NFS
- 2 Database servers (Master/Slave) (RAM Needed)
  - MySQL/Percona
  - Memcached/Redis
  - SolR

Server Architectures

- Use Varnish on anonymous pages
- Redis or Memcached are alternative to native cache in database.
  - It allows to store sessions
- APC/OPCache is an OPCode cache
  - Improve up to 20% the PHP performances
  - Easy to install

Plan

Practice

- Install Varnish and Memcache from given instructions

Plan

Security advices

- Performance
  - Keep minimal packages on target server to prevent security holes
    - Commonly, GIT and composer are not installed on the production server
  - Always use SSH key to connect on remote server, disable password login
  - Update your packages frequently to most recent versions

Plan

3 Continuous Integration

- Development
- Packaging
- Deployment
- Monitoring

Plan

Development

- Keep your source code under a version control system.
  - Example of a Git Workflow
    - Make branches related to environments: prod, preprod, integ, develop
    - Make features branches for long developments
    - Rebase features branches with develop and deleted after merge
  - Code reviews with your peers
  - ... Or limit branch access with Gitlab to force merge reviews.
  - Never use –force (especially for rebase) ... unless you're a Git Jedi !

Plan

Packaging

- Packaging
  - Create an archive file package (zip, tar.gz etc...) from source repositories
    - git clone
    - composer install
    - front assets (Gulp, Grunt, Bower, NPM or webpack).
  - tag the release package with Git.
  - /! Important: Never copy local sources to deploy a version

Plan

Deployment

- Deployment
  - Deploy by unpacking release, then play deployment tasks
    - Update symlinks: site settings, NFS, backups directories (current vs old releases)
    - Clear all caches
    - Revert configuration (Features)
  - Bash Script, Phing, Capistrano, or Ansible
  - Ansible: script ssh tasks ("playbooks") similarly across different linux distributions

Plan

Monitoring

- Monitoring
  - Use Jenkins to monitor server status and add tasks after deployments
  - Launch automated tests and verify coding standard to block releases

Plan

4 Extending Drupal 8

- Coding standards
- OOP Concepts
- Namespaces and autoload
- Practice
- General concepts
- Themes
- Modules
- Installation profiles

Plan

4 Extending Drupal 8

- Coding standards
- OOP Concepts
- Namespaces and autoload
- Practice
- General concepts
- Themes
- Modules
- Installation profiles

General

Drupal 8 follows PSR-1 and most of the drupal 7 coding standards

- 2 spaces indent with no tabs (PSR-2 = 4 spaces)
- Space before and after operators (+, -, =, etc)
- Control structures (respect spaces, end of line bracket, etc)

```
  //
}
elseif ($condition3 && $condition4) {
  //
}
else {
  //
}
```

- String Concatenations with space before and after the "."
- More at `https://www.drupal.org/coding-standards`

Drupal follows common PHP conventions for OO code but there are Drupal-specific considerations.

- Use UpperCamel case for Class name (no underscore)
  - `LocaleEvent`

Drupal follows common PHP conventions for OO code but there are Drupal-specific considerations.

- Use UpperCamel case for Class name (no underscore)
  - `LocaleEvent`
- Use lowerCamel case for Methods and class properties
  - `getLangCodes()`
  - `$langCodes`

Drupal follows common PHP conventions for OO code but there are Drupal-specific considerations.

- Use UpperCamel case for Class name (no underscore)
  - `LocaleEvent`
- Use lowerCamel case for Methods and class properties
  - `getLangCodes()`
  - `$langCodes`
  - Hooks variables and properties of configuration entities are exceptions and can use underscores
    - `$entity_type_class`

Drupal follows common PHP conventions for OO code but there are Drupal-specific considerations.

- Use UpperCamel case for Class name (no underscore)
  - `LocaleEvent`
- Use lowerCamel case for Methods and class properties
  - `getLangCodes()`
  - `$langCodes`
  - Hooks variables and properties of configuration entities are exceptions and can use underscores
    - `$entity_type_class`
- Interface have suffix "Interface"
  - `EntityInterface`
- Protected or private properties and methods should not use an underscore prefix

Drupal follows common PHP conventions for OO code but there are Drupal-specific considerations.

- Use UpperCamel case for Class name (no underscore)
  - `LocaleEvent`
- Use lowerCamel case for Methods and class properties
  - `getLangCodes()`
  - `$langCodes`
  - Hooks variables and properties of configuration entities are exceptions and can use underscores
    - `$entity_type_class`
- Interface have suffix "Interface"
  - `EntityInterface`
- Protected or private properties and methods should not use an underscore prefix
- More at `https://www.drupal.org/node/608152`

Plan

4 Extending Drupal 8

- Coding standards
- OOP Concepts
- Namespaces and autoload
- Practice
- General concepts
- Themes
- Modules
- Installation profiles

Drupal 8 is based on Symfony 2 components and some conventions are inherited from it.
Here are some conventions extracted from the Drupal.org website:

- Each class must be in its own file

Drupal 8 is based on Symfony 2 components and some conventions are inherited from it.
Here are some conventions extracted from the Drupal.org website:

- Each class must be in its own file
- Classes must be namespaced (see after)

Drupal 8 is based on Symfony 2 components and some conventions are inherited from it.
Here are some conventions extracted from the Drupal.org website:

- Each class must be in its own file
- Classes must be namespaced (see after)
- Class location must follow the PSR-4 autoloader (see after)

Plan

4 Extending Drupal 8

- Coding standards
- OOP Concepts
- Namespaces and autoload
- Practice
- General concepts
- Themes
- Modules
- Installation profiles

Namespaces

Classes must be namespaced

- If a module defines a class, the namespace must start with `\Drupal\module_name`
- If it is defined by Drupal Core for use across many modules, the namespace should be
  - `\Drupal\Core`
  - `\Drupal\Component`
  - Exception of the global class `\Drupal`
- See `https://www.drupal.org/node/1353118` for more about namespaces

Autoload

In order for the PSR-4-based class auto-loader to find the class, it must be located in a directory corresponding to the namespace.

- `\Drupal\module_name\foo\bar` `my_module_directory/src/foo/bar`
- `\Drupal\Core\foo\bar` `core/lib/Drupal/Core/foo/bar`

Ref

- `https://github.com/php-fig/fig-standards/blob/master/accepted/PSR-4-autoloader.md`
- `https://www.drupal.org/node/2156625` for more info about PSR-4

Plan

4 Extending Drupal 8

- Coding standards
- OOP Concepts
- Namespaces and autoload
- Practice
- General concepts
- Themes
- Modules
- Installation profiles

Practice

- Install and configure PHPCS and PHPMD for your IDE

Plan

4 Extending Drupal 8

- Coding standards
- OOP Concepts
- Namespaces and autoload
- Practice
- General concepts
- Themes
- Modules
- Installation profiles

Extending Drupal 8 - General concepts

Drupal's core behavior can be extended and altered via these basic types of extension

- Modules
- Themes
- Installation profiles

Plan

4 Extending Drupal 8

- Coding standards
- OOP Concepts
- Namespaces and autoload
- Practice
- General concepts
- Themes
- Modules
- Installation profiles

Extending Drupal 8 - Themes

- Themes alter the appearance of Drupal sites
- Can include template files, which alter the HTML markup and other raw output of the site
- CSS/JS
- Theme system and render API topic and `https://www.drupal.org/theme-guide/8`

Plan

4 Extending Drupal 8

- Coding standards
- OOP Concepts
- Namespaces and autoload
- Practice
- General concepts
- Themes
- Modules
- Installation profiles

- Modules add or alter the behavior and functionality of Drupal
  - Hooks
  - Services
  - Routing
  - Events
  - Plugins
  - Entities
- See `https://www.drupal.org/developing/modules/8`

Note

Hooks must be avoided since it's there mainly for legacy reason. Events must be prefered.

Plan

4 Extending Drupal 8

- Coding standards
- OOP Concepts
- Namespaces and autoload
- Practice
- General concepts
- Themes
- Modules
- Installation profiles

Extending Drupal 8 - Installation profiles

- Installation profiles can be used to create distributions including out of box
  - additional modules
  - themes
  - data structure
  - ...
- See

  `https://www.drupal.org/developing/distributions`