Composer Tools and Frameworks for Drupal

JULY 22-24, 2015

PANTHEON
promet source

NATIONAL INSTITUTES OF HEALTH CAMPUS

BETHESDA, MD

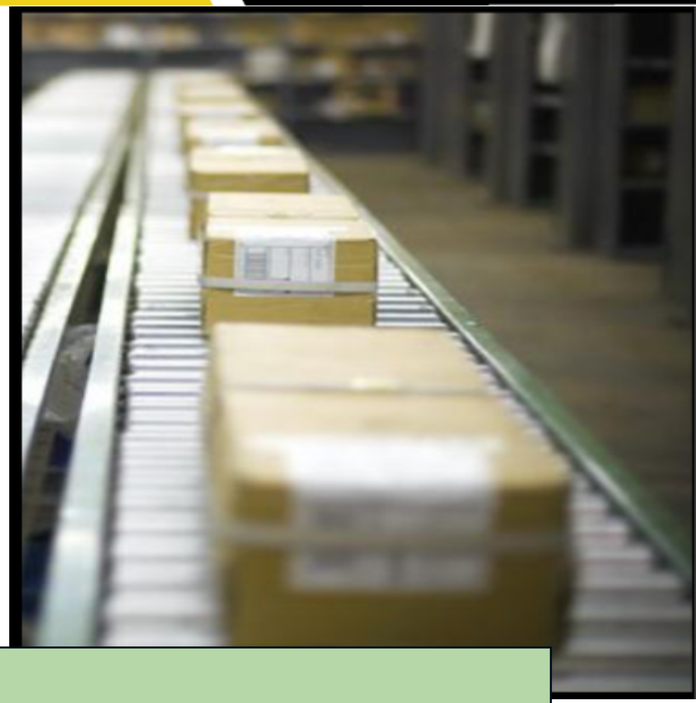
Why you should care about composer

PANTHEON
promet source

Today

- Dependency Management Hell
- Manually Include Classes at runtime
- Large unmanagable repository

With Composer



- Composer manages dependencies
- Autoloader lazy-includes class files
- Repository remains clean and lean

Composer Tools and Frameworks for Drupal

COMPOSER **DRUPAL** **DRUSH** **WIN**

Who Are We?


missing
Greg Anderson
PANTHEON
Website Management Platform
$$$
reward


missing
Doug Dobrzynski
$$$
reward

Who Are We?

Allan Chappell

Senior Solutions Architect

Agenda

- What is Composer?
- Installing Drupal 7 with Composer
  - `composer_vendor` + `custom-installer`
  - `drupal-tangler`
- Managing Your Project

PART ONE

WHAT IS COMPOSER?

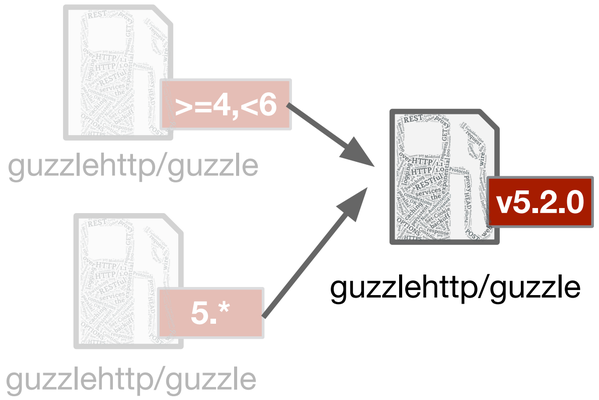
What Is Composer?

An Installer

```
{
  "require": {
      "php": ">=5.4.0",
      "symfony/browser-kit": "~2.1",
      "symfony/css-selector": "~2.1",
      "symfony/dom-crawler": "~2.1",
      "guzzlehttp/guzzle": ">=4,<6"
  },
  …
}
```

A Dependency Manager

Evaluate and select



An Autoloader

```
<?php

$client = new GuzzleHttp\Client();
```

composer.json

```
"autoload": {
   "psr-4": {
      "GuzzleHttp\\": "src/"
   }
},
```

Why Use Composer?

PANTHEON

prometsource

- Standard
- Easiest for developers
  - Dependency resolution
  - Code updates
  - Autoloading of classes
- Composer is being adopted everywhere

What Projects Are Using Composer?

PANTHEON
promet source

Using Composer

Drupal Modules


PHP APIs

Drush Extensions


fabpot/goutte

Not Using Composer

… and many others!

Drupal 7 and Composer

PANTHEON
promet source

Can we do this?

+

= ?

Composer Parts of the Whole

Composer

PHP dependency management software.

Packagist

A software repository manager.

composer.json

A structured file that defines a project.

autoload.php

A generated file that must be included by your app.

Custom Installer

An executable Composer component. Optional.

Repository Data for Drupal Projects

packagist.drupal-composer.org

A third-party repository containing data on projects from drupal.org.

Semantic Versioning

Drupal Module Version	Composer Version
7.x-1.5	7.1.5
DRUPAL-x.MODULE.MINOR	MAJOR.MINOR.PATCH

Versions are converted to semantic versioning before being published on packagist.drupal-composer.org.

Comparison with Drush Make

Drush Make

```
; Drush make file that uses guzzle
; API
api = 2

; Core
core = 7.x

; Drupal project.
projects[drupal][type] = core
projects[drupal][version] = 7.x
projects[drupal][download][type] = git
projects[drupal][download][branch] = 7.x

; Modules
projects[] = composer_manager
projects[] = aws_glacier
projects[] = devel
```

Drupal module that provides an autoload strategy (next).

composer.json

```
{
  "name": "organization/project",
  "description": "Drupal composer.json file",
  "repositories": [
    {
      "type": "composer",
      "url": "http://packagist.drupal-composer.org/"
    }
  ],
  "require": {
    "davidbarratt/custom-installer": "dev-master",
    "derhasi/composer-preserve-paths": "0.1.*",
    "drupal/drupal": "7.*",
    "drupal/composer_vendor": "7.1.*",
    "http/guzzle": "~5",
    "drupal/devel": "7.1.*",
  },
   …
}
```

Repository and custom installers (previously explained)

Autoloading in PHP

RUN TIME

php source file that calls GuzzleHTTP

```
<?php
include "vendor/autoload.php";

$client = new GuzzleHTTP\Client();
```

INSTALL TIME

composer.json from guzzlehttp/guzzle

```
"autoload": {
    "psr-4": {
        "GuzzleHttp\\": "src/"
    }
},
```

include $vendorDir . '/guzzlehttp/guzzle/src/Client.php';

Composer autoloader registration called from vendor/autoload.php

```
$map = require __DIR__ . '/autoload_psr4.php';
foreach ($map as $namespace => $path) {
    $loader->setPsr4($namespace, $path);
}
```

autoload_psr4.php generated by Composer via composer install

```
$vendorDir = dirname(dirname(__FILE__));

return array(
    'GuzzleHttp\\' => array($vendorDir .
'/guzzlehttp/guzzle/src'),
);
```

Saves one line of code per class - but it's a very important line!

Custom Installers for Composer + D7

davidbarratt/custom-installer

Allows composer.json files to specify where components get installed.

derhasi/composer-preserve-paths

Allows nested installation of composer libraries (e.g. Drupal modules inside of Drupal core).

OR

promet/drupal-tangler

Installs modules and themes to their standard location in the vendor directory, and symlinks to them from sites/all/modules/contrib or copies them to sites/all/modules/contrib.

OPTIONAL

generalredneck/drupal-libraries-installer-plugin

Allows composer.json files to specify where components get installed.

netresearch/composer-patches-plugin

Allows patch files to be applied to Drupal core or contrib modules after Composer installs them.

Adding Composer Support

To make this work, we just need to include `vendor/autoload.php`

PART TWO

INSTALLING DRUPAL 7 WITH COMPOSER

Composer Options for Drupal 7

PANTHEON
prometsource

composer_autoload

Searches for
autoload.php files
in every module
directory and loads
them individually.

composer_manager

Searches for
composer.json files
in every module
and dynamically
merges them.

Better Options for Drupal 7

PANTHEON
promet source

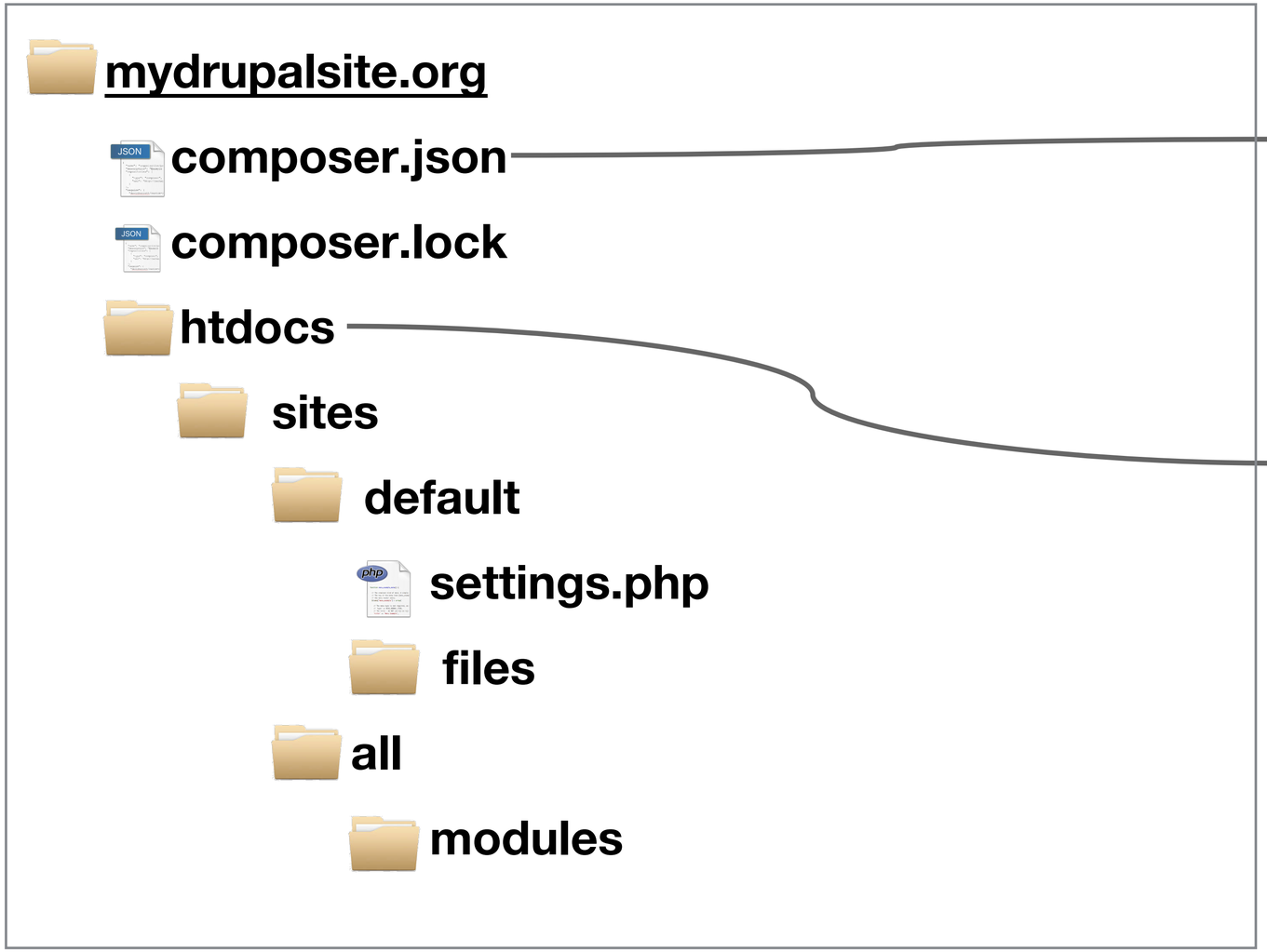
composer_vendor

Loads the `sites/all/vendor/autoload.php` file.

promet/drupal-tangler

Writes a `settings.php` file that loads the correct `autoload.php` file.

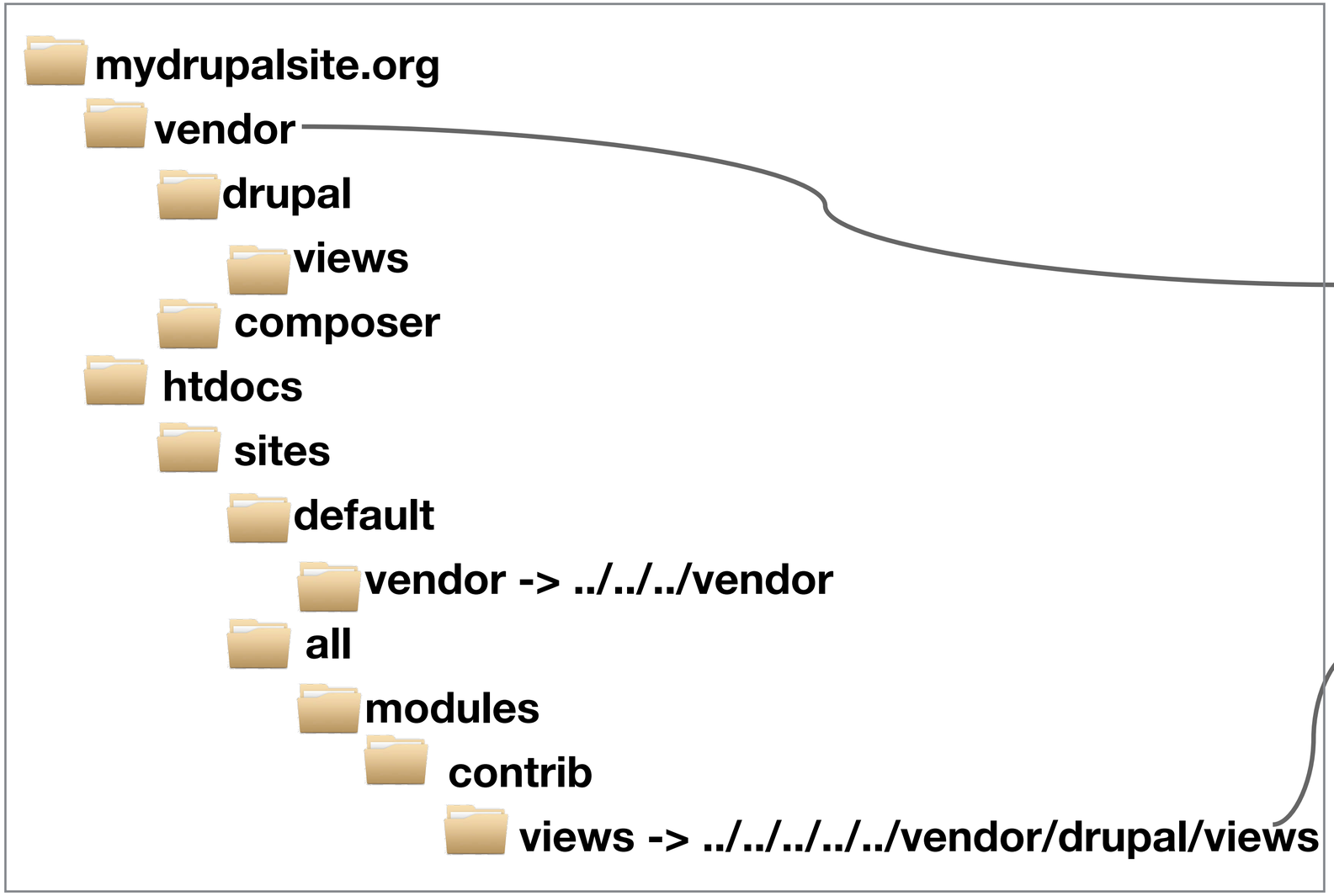
Directory Structure
With `drupal-tangler`



We create a new top-level directory for our project, because composer cannot manage dependencies in the same directory as composer.json.

We will put our Drupal root directory in `htdocs`. Tangler defaults to `www` but allows the option to specify a different name.

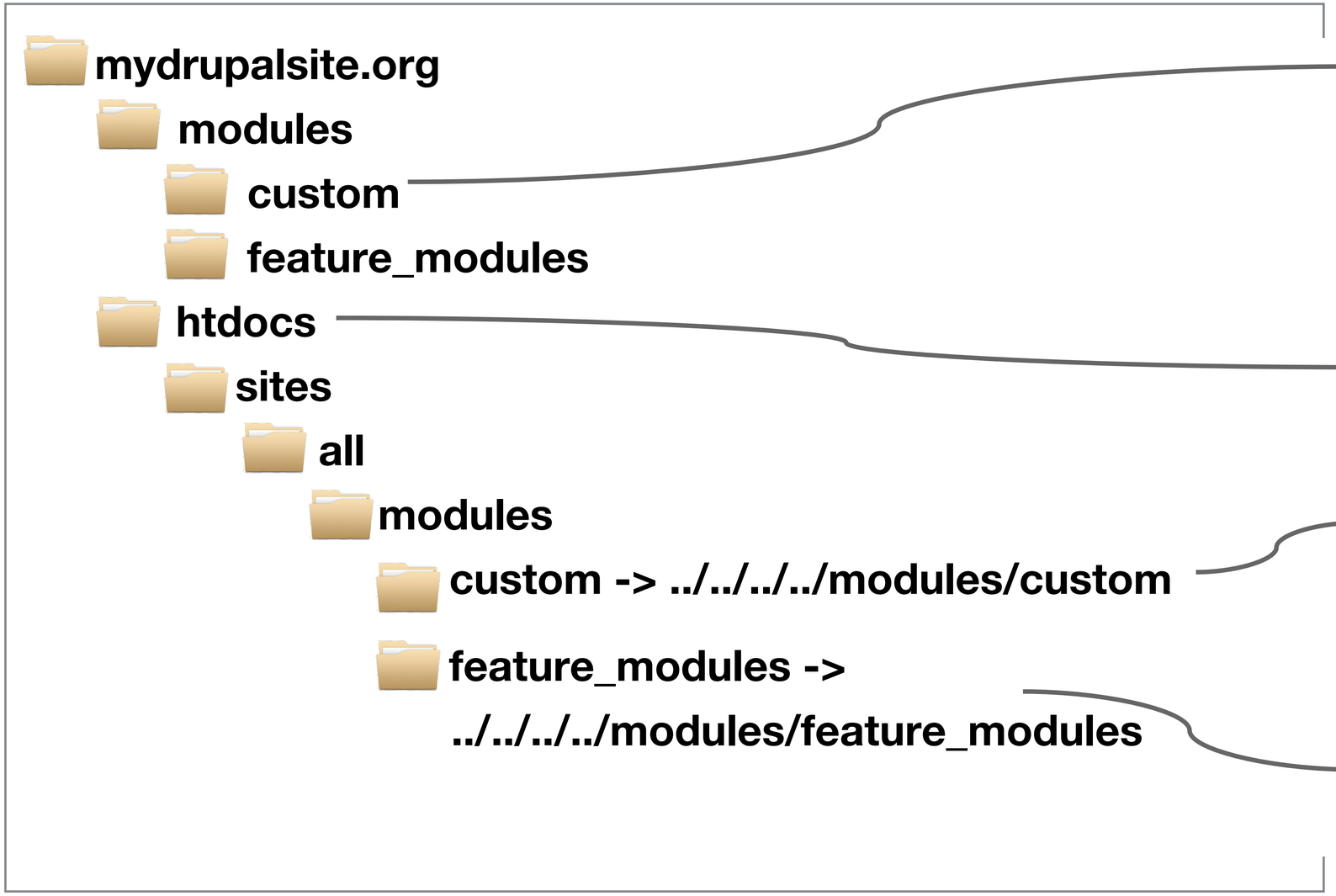
Directory Structure
With `drupal-tangler`



We put the `vendor` directory in root because that is the default location for the Composer installer. The directory is organized by vendor name.

Each contributed module directory is symlinked to the corresponding directory in vendor.

Directory Structure
With `drupal-tangler`



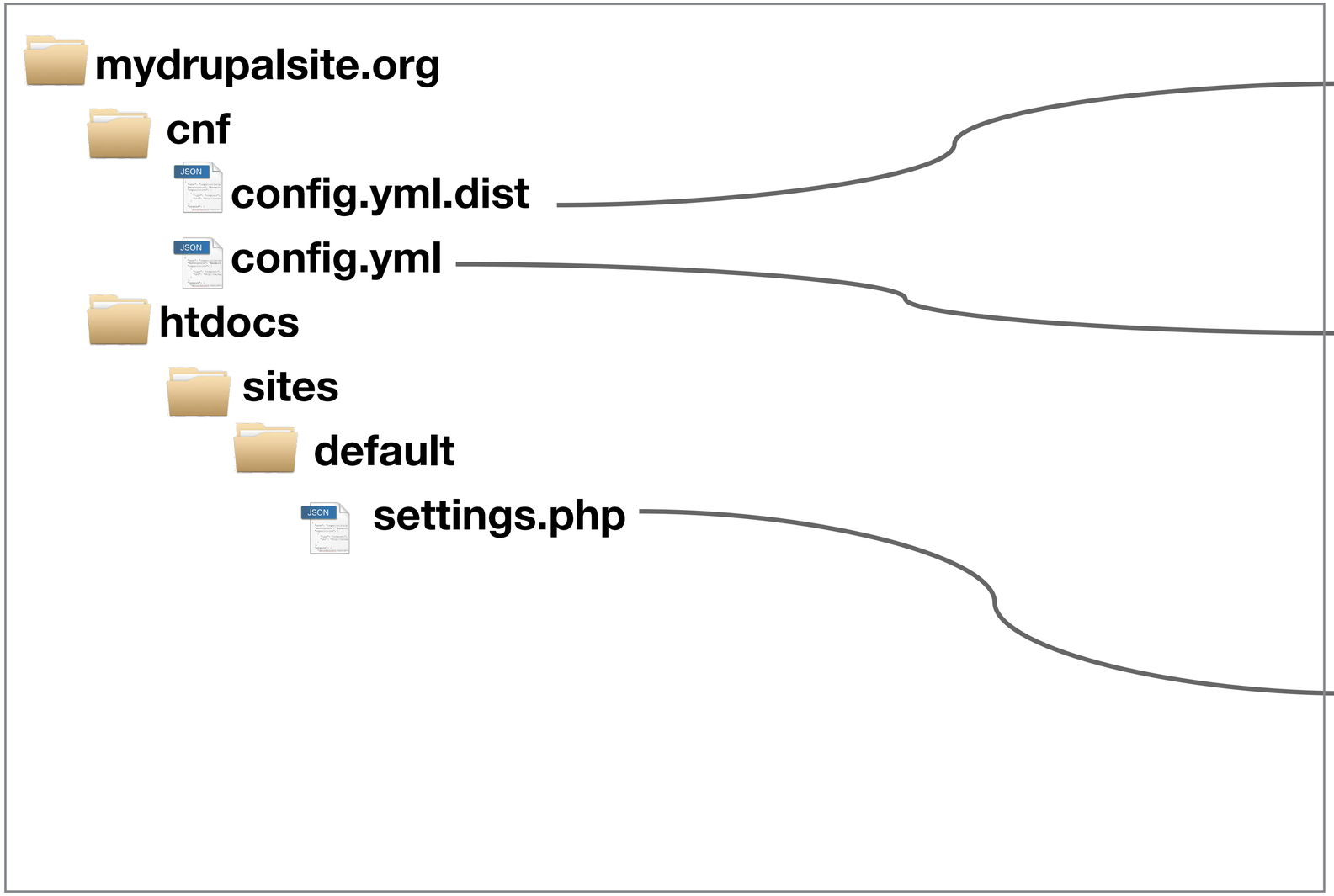
We put custom modules in the `custom` directory in the `modules` directory in root.

We will put our Drupal root directory in `htdocs`. Tangler defaults to `www` but allows the option to specify a different name.

The `custom` directory is symlinked to the `custom` directory in the `modules` directory in root.

The `feature_modules` directory is symlinked to the `feature_modules` directory in the `modules` directory in root.

Directory Structure
With `drupal-tangler`

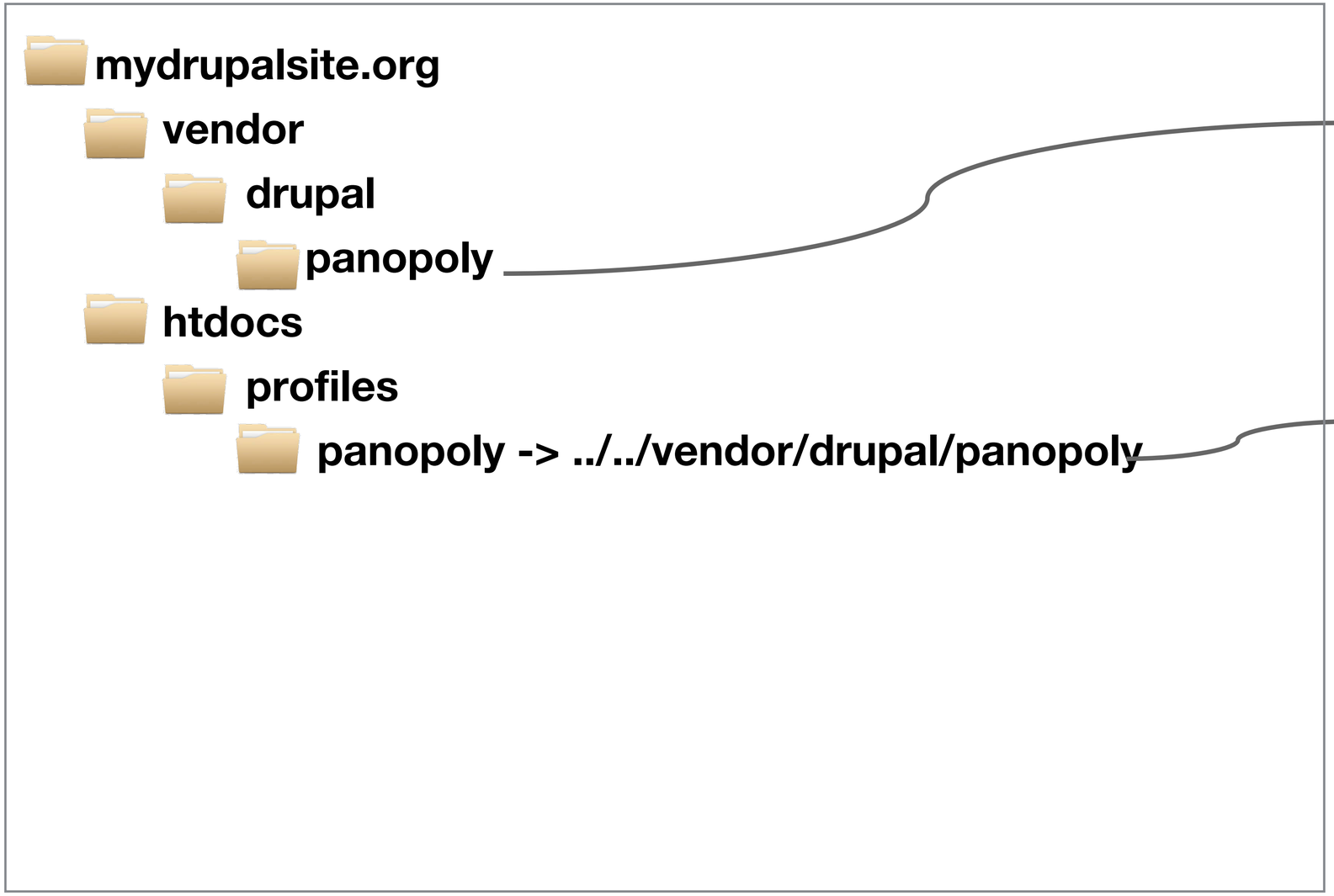


We put default configuration for settings in `config.yml.dist`.

We put default configuration for settings in `config.yml.dist`.

The settings.php file is generated by `settings_compile` from config.yml. If config.yml does not exist, it is generated from config.yml.dist.

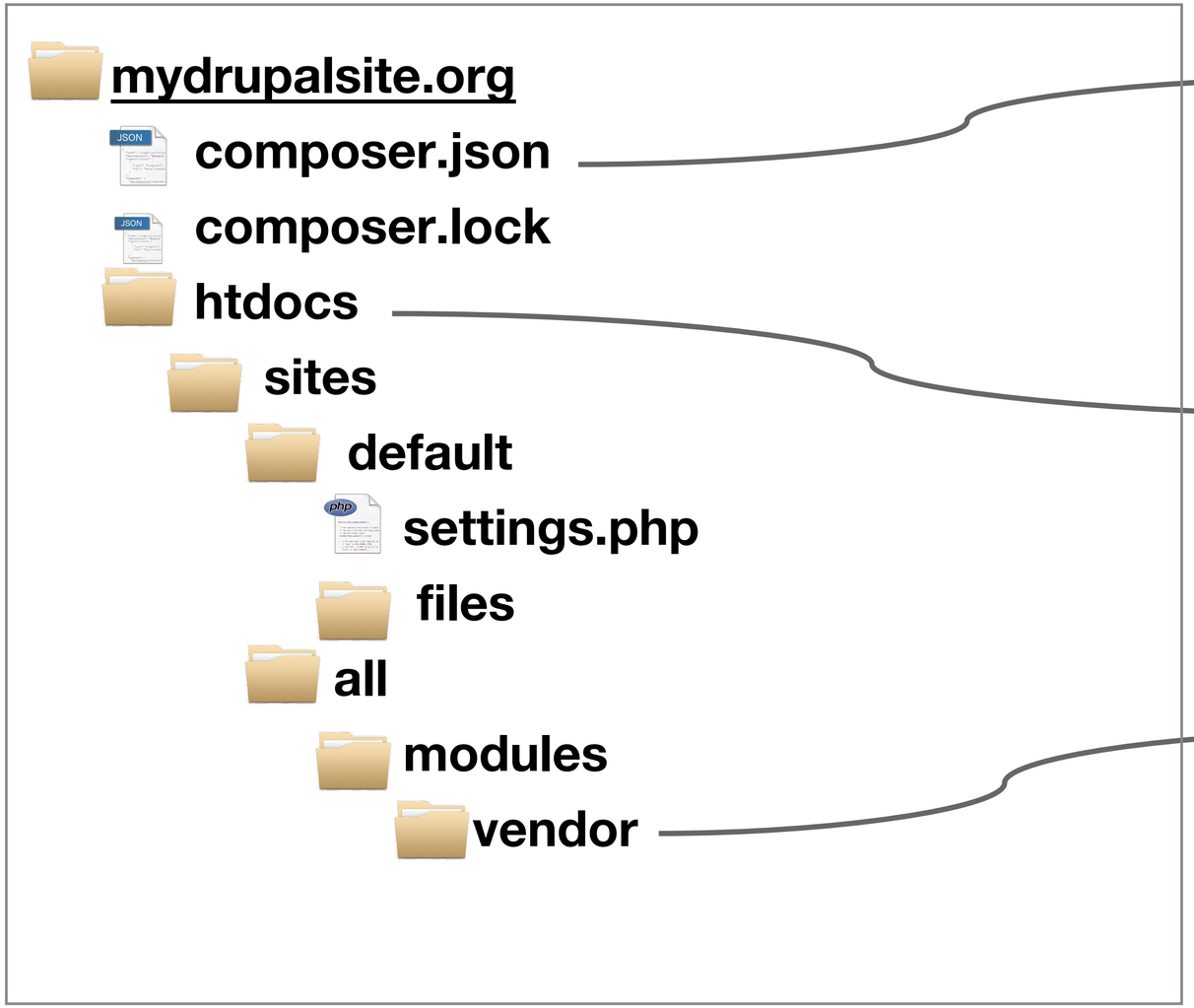
Use Installation Profiles
With `drupal-tangler`



Composer downloads the profile to the `drupal` directory in the `vendor` directory.

The `panopoly` directory is symlinked to the `panopoly` directory in the `vendor` directory.

Directory Structure
With `composer_vendor`



We create a new top-level directory for our project, because composer cannot manage dependencies in the same directory as composer.json.

We will put our Drupal root directory in htdocs.

We put the vendor directory in `sites/all/vendor` because that is where the `composer_vendor` project expects to find it.

Place the Vendor Directory

For `composer_vendor` with a `custom-installer`

composer.json

```
{
    "require": {
        …
    },
    "config": {
        "vendor-dir": "htdocs/sites/all/vendor"
    },
    …
}
```

Place Modules and Themes
For `composer_vendor` with a `custom-installer`

composer.json

```
{
  "require": {
    "davidbarratt/custom-installer": "dev-master",
    …
  },
  "extra": {
    "custom-installer": {
      "drupal-module": "htdocs/sites/all/modules/contrib/{$name}/",
      "drupal-theme": "htdocs/sites/all/themes/contrib/{$name}/"
    },
  },
  …
}
```

Use Installation Profiles

For `composer_vendor` with a `custom-installer`

composer_vendor

composer.json

```
{
  "require": {
    "davidbarratt/custom-installer": "dev-master",
    "drupal/panopoly": "7.1.*",
    …
  },
  "extra": {
    "custom-installer": {
      "drupal-profile": "htdocs/profiles/{$name}/"
    },
  },
  …
}
```

Downloading Modules

Drush

```
$ drush dl devel
Project devel (7.x-1.5) downloaded to
sites/all/modules/contrib/devel.
Project devel contains 3 modules:
devel_generate, devel, devel_node_access.
```

Composer

```
$ composer require drupal/devel '7.*'
./composer.json has been updated
Loading composer repositories with package
information
Updating dependencies (including require-dev)
```

Drush will select the right module major version, but `composer require` must be told which version to use.

`Composer require` will update the `composer.json` file before installing the module.

Install a Module from a Private Repository

For `composer_vendor` and `drupal-tangler`

PANTHEON

prometsource

composer.json

```
{
  "repositories": [
    {
      "type": "vcs",
      "url": "https://github.com/your-org/your-module"
    }
  ],
  "require": {
    "your-org/your-module": "dev-master"
  },
  …
}
```

https://knpuniversity.com/screencast/question-answer-day/create-composer-package

Add a Patch to a Module

For `composer_vendor` and `drupal-tangler`

PANTHEON

prometsource

composer.json

```
{
  "require": {
    "netresearch/composer-patches-plugin": "~1.0"
  },
  "extra": {
    "patches": {
      "drupal/features": {
        "7.2.2":[
          {
            "title": "Remove mtime from .info export (added by Drupal 7.33)",
            "url": "https://www.drupal.org/files/issues/2381739-features-mtime.patch"
          }
        ]
      }
    }
  },
  …
}
```

http://cambrico.net/drupal/using-composer-to-build-your-drupal-7-projects

Use a Composer Library from a Module

1. Add a `composer.json` and `require` the library

2. There is no step two!

VERY IMPORTANT - Never try to include an autoload.php file from a plug-in; always leave autoloader management to the application.

Use Composer from a Drush Command

1. Add a `composer.json` and `require` the library

2. Call `drush_autoload(__FILE__)` from your `hook_drush_init()`.

3. `Require` your Drush command in the `composer.json` of the Drupal site it is used in.

Updating a Site

Drush

```
$ drush pm-update
```

Composer

```
$ composer update
$ drush updatedb
```

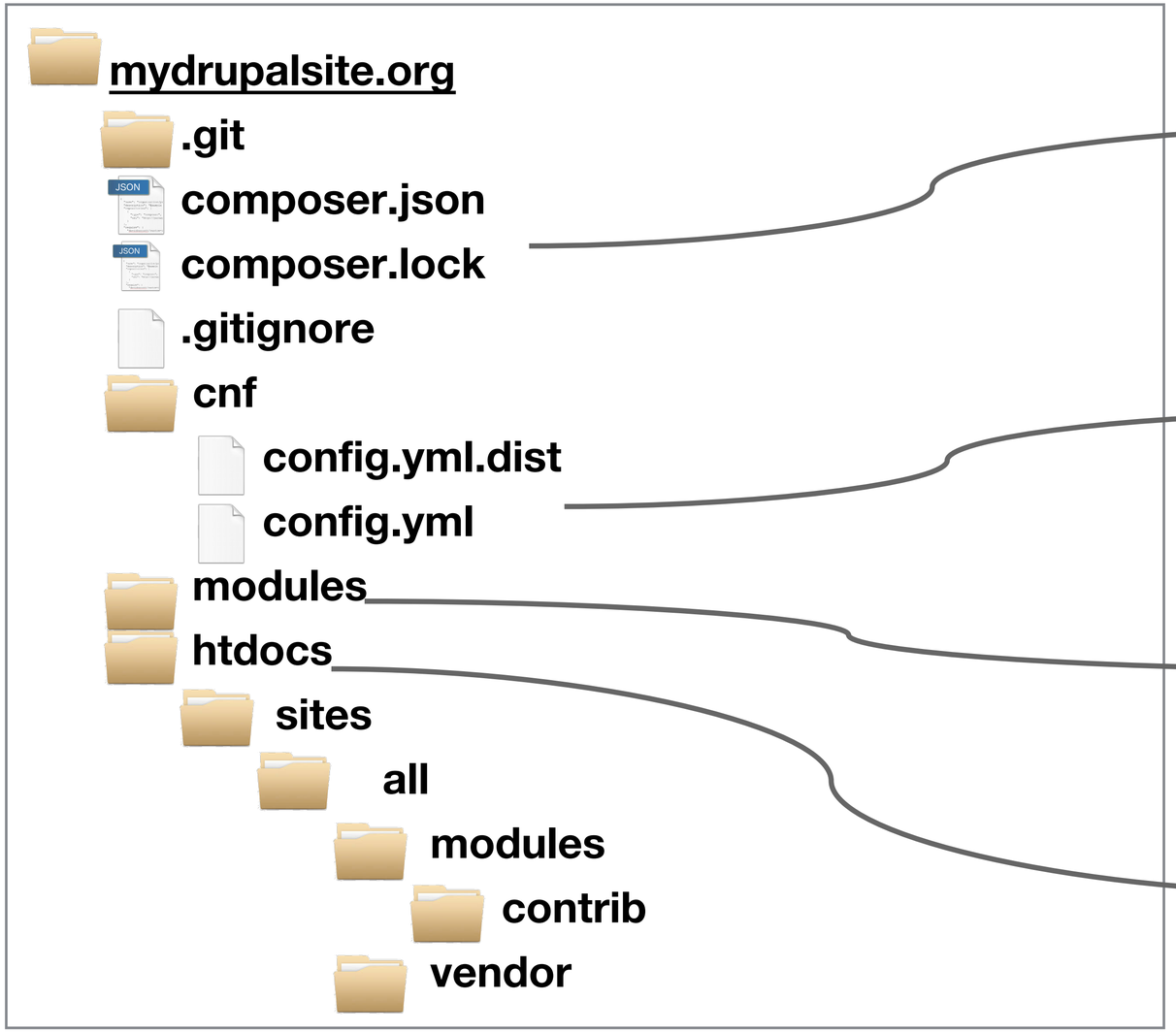
Tangler works the same

Remember - regardless of how you update your site, always do it on a copy first. Never update directly on the production site!

PART THREE

MANAGING YOUR PROJECT

Manage Project Code
With `drupal-tangler`



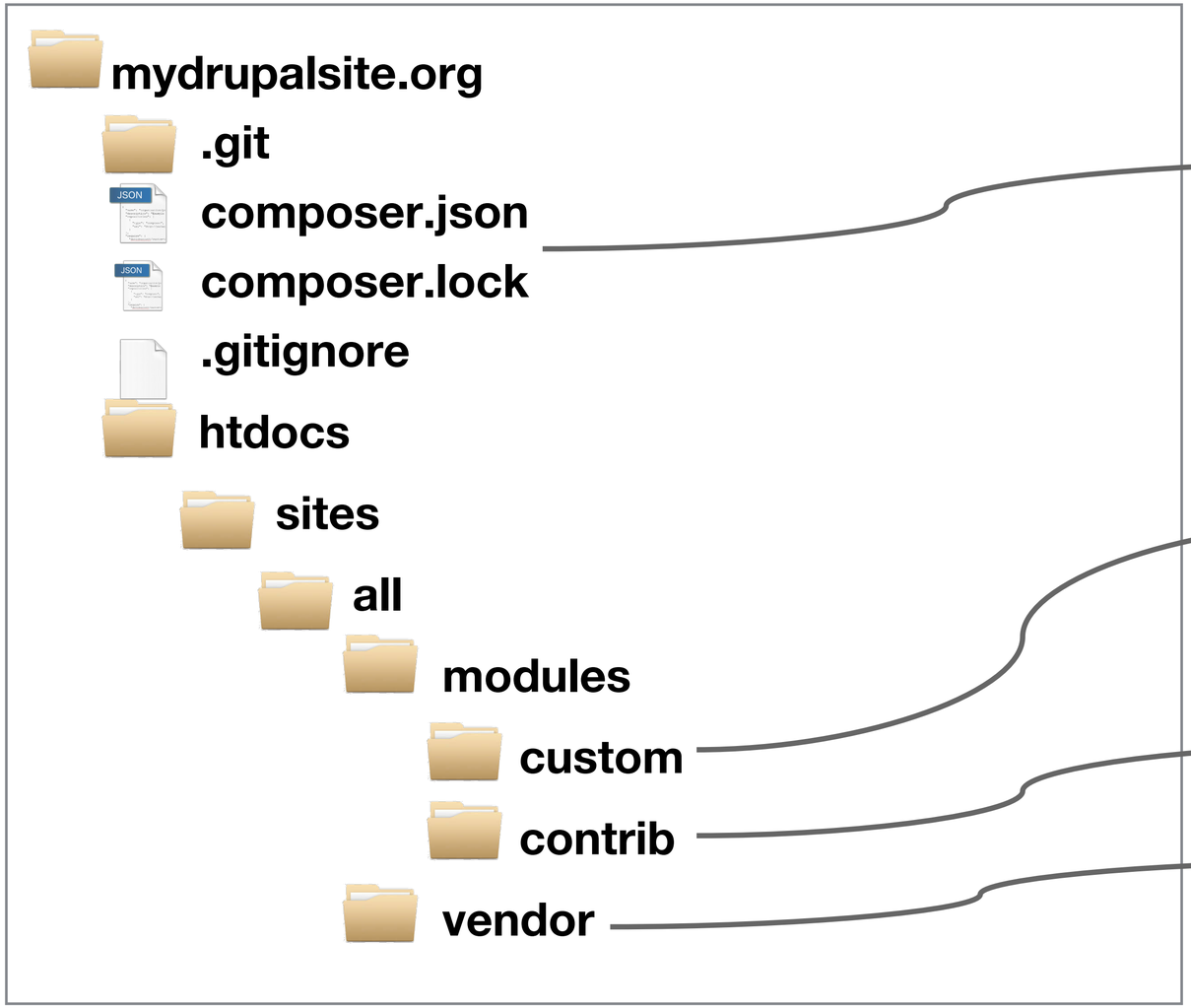
Commit composer.json. and composer.lock to the repository. composer.lock only changes when you run composer update.

Commit config.yml.dist.
Avoid committing config.yml.

If you have custom modules, you can commit them in the `modules` directory.

Avoid committing the `htdocs` directory, which is managed by `drupal-tangler`

Manage Project Code
With `composer_vendor`



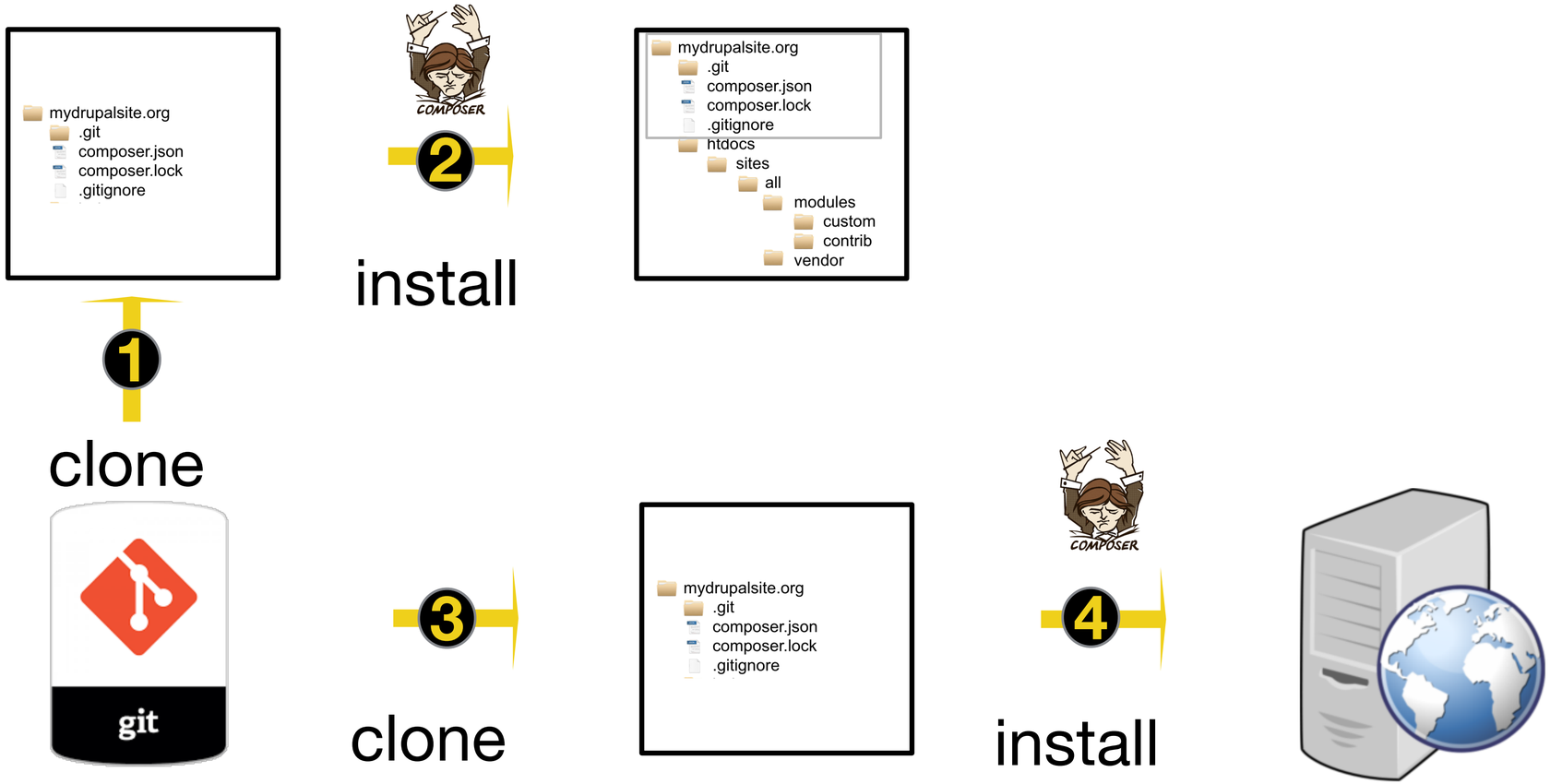
Commit `composer.json`. and `composer.lock` to the repository. `composer.lock` only changes when you run `composer update`.

If you have custom modules, you can commit them in the custom directory..

Avoid committing composer-managed directories, such as `sites/all/modules/contrib` and `sites/all/vendor`.

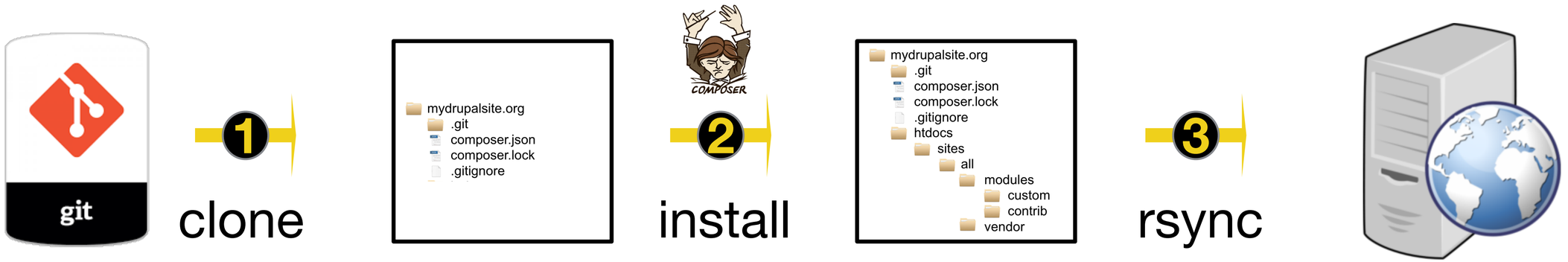
Deploy Code Using Composer

Local and Remote Dev Environments



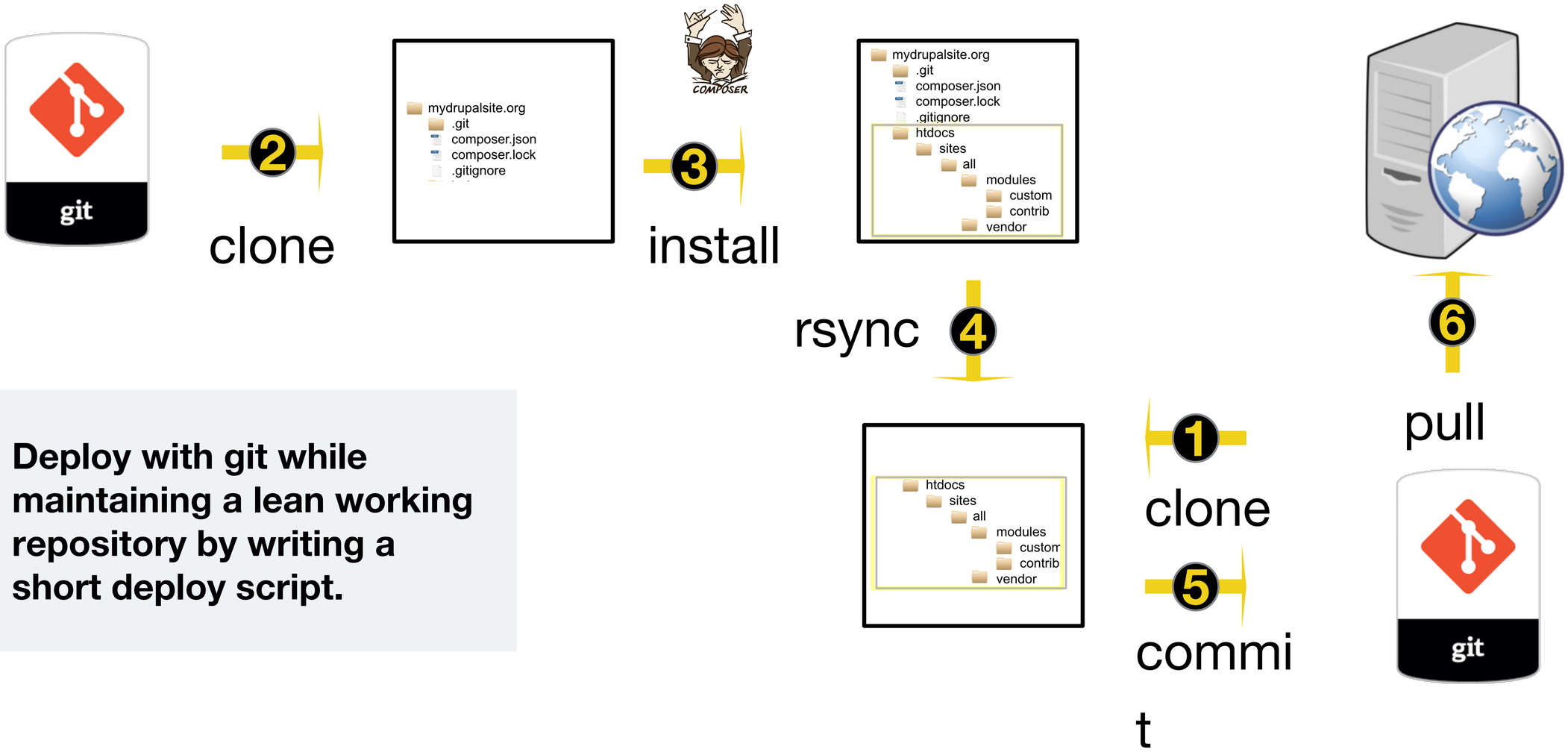
Deploy Code Using Rsync

Copy code from dev to stage or live



Deploy Code Using Two Repositories

Isolate the Provider's Repository



Deploy with git while maintaining a lean working repository by writing a short deploy script.

Converting an Existing Site

```
$ drush dl composer_generate
$ drush @site composer-generate > composer.json
$ composer install
# Set up settings.php, copy files…
$ drush site-install
```

Creating a New Site

drupal-composer/drupal-project

```
$ composer create-project drupal-
composer/drupal-project:7.x-dev dir
    --stability dev --no-interaction
# Set up settings.php, copy files…
$ drush site-install
```

Creating a New Site

promet/drupal7-framework

```
$ composer create-project promet/drupal7-
framework project_name
$ vagrant up --provision
```

Where Do We Go From Here?

https://getcomposer.org/

http://drupal-composer.org/

https://groups.drupal.org/composer

https://github.com/drupal-composer

@general_redneck

@greg_1_anderson

@dsdobrzynski