Dependency Management in WordPress with Composer

Ryan Plas

Back-end Developer @ CrowdFavorite

@WordPlas

Overview

1. What is Composer?
2. Why should you use Composer?
3. How can you get started with Composer?
4. How can you use Composer with WordPress?
5. What are some Issues you can run into

What is Composer?

What is Composer?

Composer is a tool for dependency management in PHP. It allows you to declare the libraries your project depends on and it will manage (install/update) them for you.
– Composer Documentation

What is Composer?

Other Package/Dependency Managers

- NPM (JavaScript)
- RubyGems (Ruby)
- pip (Python)

Composer Basics

init

Initialize a composer.json file with prompts.

Composer Basics

update

Get the latest versions of dependencies and update the composer.lock file.

Composer Basics

install

Reads the composer.json file from the current directory, resolves the dependencies, and installs them into vendor.

Composer Basics

require

Adds new packages to the composer.json file from the current directory and installs them.

Why Should You Use Composer?

Why Should You Use Composer?

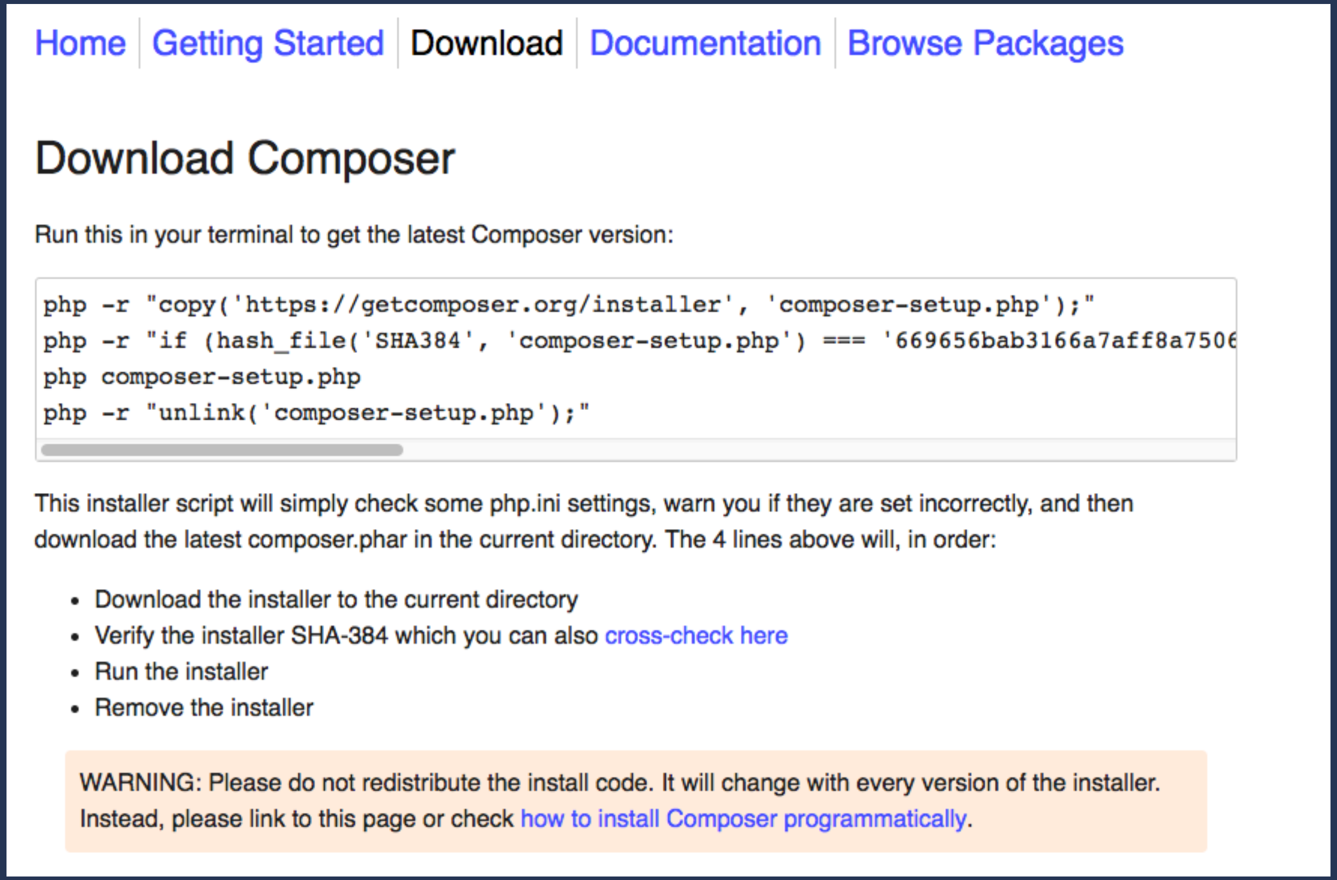
Composer is useful for:

1. Managing **third-party** plugins/dependencies
2. **Autoloading** classes
3. Creating/running custom **install/build/[x]_** scripts

How Can You Get Started With Composer?

How can you get started with Composer?

https://getcomposer.org/download/ (Local):



Download Composer

Run this in your terminal to get the latest Composer version:

```
php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"
php -r "if (hash_file('SHA384', 'composer-setup.php') === '669656bab3166a7aff8a7506
php composer-setup.php
php -r "unlink('composer-setup.php');"
```

This installer script will simply check some php.ini settings, warn you if they are set incorrectly, and then download the latest composer.phar in the current directory. The 4 lines above will, in order:

- Download the installer to the current directory
- Verify the installer SHA-384 which you can also cross-check here
- Run the installer
- Remove the installer

WARNING: Please do not redistribute the install code. It will change with every version of the installer. Instead, please link to this page or check how to install Composer programmatically.

How can you get started with Composer?

Homebrew (Global):

```
brew update
brew tap homebrew/dupes
brew tap homebrew/php
brew install composer
```

How can you get started with Composer?

Use Local:

```
php composer.phar init
```

Use Global:

```
composer init
```

DEMO

How can you get started with Composer?

- Package Name: vendor/name (wordplas/cool-library)
- Description: What does your package do?
- Author: Who are you? (FirstName LastName <email@email.com>)
- Minimum Stability: defaults to stable
- Define Dependencies

SO EASY!!!

How Can You Use Composer With WordPress?

How can you use Composer with WordPress?

1. Plugin/Theme Development
2. Manage your WordPress Install

Plugin/Theme Development

1. Class Autoloading
2. Manage PHP dependencies

Class Autoloading

1. PSR-4
2. PSR-0
3. classmap
4. file

Class Autoloading

PSR4

```json
{
    "autoload": {
        "psr-4": {
            "CoolCalendar\\": "src/"
        }
    }
}
```

Class Autoloading

PSR4

```
├──composer.json
├──cool-calendar.php
└──src
    └──Taxonomy
        └──EventTaxonomy.php
```

Class Autoloading

PSR4

`src/Taxonomy/EventTaxonomy.php`

```php
<?php

namespace CoolCalendar\Taxonomy;

class EventTaxonomy{
    public function do_something() {

    }
}
```

Class Autoloading
PSR4

cool-calendar.php

```php
<?php
// Stuff about my plugin

use CoolCalendar\Taxonomy\EventTaxonomy;

require __DIR__ . '/vendor/autoload.php';

$taxonomy = new EventTaxonomy();
$taxonomy->do_something();
```

DEMO

Class Autoloading
Pros

- Only one require
- Organized code

Class Autoloading

Cons

- PSR-4 directly contradicts WordPress Coding Standards

Manage PHP Dependencies

```
php composer.phar require guzzlehttp/guzzle:~6.0
```

```
{
    "require": {
        "guzzlehttp/guzzle": "~6.0"
    }
}
```

Manage PHP Dependencies

```php
<?php

use GuzzleHttp\Client;

$client = new GuzzleHttp\Client(['base_uri' => 'https://foo.com/api/']);
$response = $client->request('GET', 'test');
```

DEMO

Manage PHP Dependencies

- Cleaner repo
- Easier to manage updates
- Easier to keep everyone on the same version

Manage your WordPress Install

WordPress in a Sub-directory

```
{
    "require": {
        "johnpbloch/wordpress": "^4.7",
        "composer/installers": "~1.2.0"
    },
    "extra": {
        "wordpress-install-dir": "wp"
    }
}
```

Manage your WordPress Install
Manage Plugins as Dependencies

WordPress Packagist

```
{
    "repositories":[
        {
            "type":"composer",
            "url":"https://wpackagist.org"
        }
    ]
}
```

Manage your WordPress Install

Manage Plugins as Dependencies

```
composer require wpackagist-plugin/contact-form-7
```

Manage your WordPress Install

Manage Plugins as Dependencies

Custom install directories

```
{
    extra": {
        "installer-paths": {
            "wp-content/plugins/{$name}/": ["type:wordpress-plugin"],
            "wp-content/mu-plugins/{$name}/": ["type:wordpress-muplugin"],
            "wp-content/themes/{$name}/": ["type:wordpress-theme"]
        }
    }
}
```

DEMO

Manage your WordPress Install

Next Steps:

- Set up wp-config and server config to recognize new folder structure
- See: Bedrock by Roots for an example of a full setup.

What Are Some Issues You Can Run Into?

Issues

Version conflicts

- Plugin 1 requires FooSDK v1
- Plugin 2 requires FooSDK v2
- Plugin 1 is loaded first with FooSDK v1
- Plugin 2 runs into problems

OH NO!!!

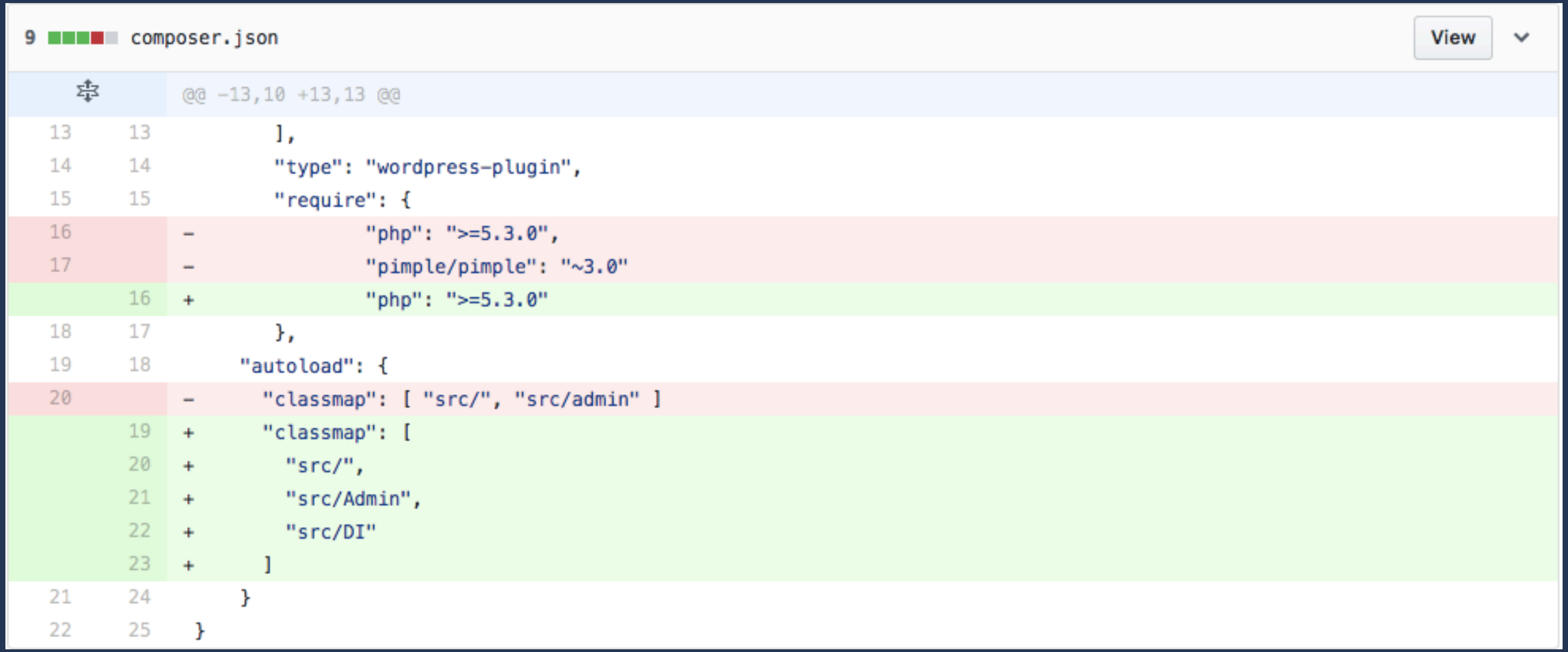
WHAT CAN WE DO???

Options

- Manually rename packages
- Automatically rename packages

Manually Rename Packages

- Copy the class files into a folder
- Add the folder to the autoloader section of composer.json



```diff
@@ -13,10 +13,13 @@
         ],
         "type": "wordpress-plugin",
         "require": {
-                "php": ">=5.3.0",
-                "pimple/pimple": "~3.0"
+                "php": ">=5.3.0"
         },
      "autoload": {
-       "classmap": [ "src/", "src/admin" ]
+       "classmap": [
+         "src/",
+         "src/Admin",
+         "src/DI"
+       ]
      }
 }
```

Automatically Rename Packages

https://github.com/coenjacobs/mozart

```json
{
    "require-dev": {
        "coenjacobs/mozart": "^0.2.0"
    },
    "extra": {
        "mozart": {
            "dep_namespace": "WordPlas\\MozartTest\\Dependencies\\",
            "dep_directory": "/src/Dependencies/",
            "classmap_directory": "/classes/dependencies/",
            "classmap_prefix": "WPMZ_",
            "packages": [
                "pimple/pimple"
            ]
        }
    },
    "scripts": {
        "post-install-cmd": [
            "\"vendor/bin/mozart\" compose"
        ],
        "post-update-cmd": [
            "\"vendor/bin/mozart\" compose"
        ]
    },
    "autoload": {
        "psr-4": {
            "WordPlas\\MozartTest\\Dependencies\\": "src/Dependencies"
        }
    }
}
```

Automatically Rename Packages

```php
<?php

use WordPlas\MozartTest\Dependencies\Pimple\Container;

require __DIR__ . '/vendor/autoload.php';

$container = new Container();

var_dump($container);
```

DEMO

Issues

Version conflicts

https://deliciousbrains.com/dependency-management-wordpress-proposal/

Questions?