

10 Things You Didn't Know You Could Do With Composer



Patrick Schwisow

ZendCon 2016

@PSchwisow

10 Things You Already Knew...

1. Dependency Resolution
2. Install Libraries
3. Check PHP Version
4. Check for Extensions
5. Updating Libraries
6. Works with Packagist
7. Autoloading
8. Lock Library Versions
9. Minimum Stability
10. Use Named Branches



1. Optimize Autoloader

- PSR-0 / PSR-4 autoloading may require many calls to `file_exists` (SLOW).
- Classmaps are faster, but require manual updates.
- Composer to the rescue!!!



1. Optimize Autoloader

- Composer can generate a classmap for all PSR-0/4 classes.
- If not in classmap, fallback to PSR-0/4 autoloader.

```
composer dump-autoload --optimize
```

```
composer install --optimize-autoloader
```



1. Optimize Autoloader

- To turn off PSR-0/4 fallback:

```
composer dump-autoload --classmap-authoritative
```

```
composer install --classmap-authoritative
```



2. Override Autoloading

- Autoloader is built from a sorted list of autoload mappings from `composer.json` files in the project and all dependencies.
- Autoload mappings in main `composer.json` take precedence.

Example 2-1: Override Autoloading in composer.json

```
1 {  
2     "require": {  
3         "phergie/phergie-irc-bot-react": "~2",  
4         "pschwisow/irc-plugin-react-karma": "~2"  
5     },  
6     "autoload": {  
7         "psr-4": {  
8             "PSchwisow\\Plugin\\Karma\\": "/code/karma/src"  
9         }  
10    }  
11 }
```

@PSchwisow



Example 2-2: vendor/composer/autoload_psr4.php

```
1 <?php
2 // autoload_psr4.php @generated by Composer
3
4 $vendorDir = dirname(dirname(__FILE__));
5 $baseDir = dirname($vendorDir);
6
7 return array(
8     'Phergie\\Irc\\Bot\\React\\' => array(
9         $vendorDir . '/phergie/phergie-irc-bot-react/src'
10    ),
11    'PSchwisow\\Plugin\\Karma\\' => array(
12        $baseDir . '/../karma/src',
13        $vendorDir . '/pschwisow/irc-plugin-react-karma/src'
14    ),
15 );
16
```

@PSchwisow





3. Inline Aliases

- Inline aliases tell Composer to use one version of a package as if it were another version.

Example 3-1: Before Inline Alias

```
1 {  
2   "require": {  
3     "phergie/phergie-irc-bot-react": "dev-bugfix",  
4     "pschwisow/irc-plugin-react-karma": "~2"  
5   }  
6 }
```

7
8 Your requirements could not be resolved to an installable set of
9 packages.

10 Problem 1

11 - pschwisow/irc-plugin-react-karma 2.0.0 requires phergie/
12 phergie-irc-bot-react ~2 -> satisfiable by phergie/phergie-irc-
13 bot-react [2.0.0, 2.0.1] but these conflict with your requirements
14 or minimum-stability.

15 - Installation request for pschwisow/irc-plugin-react-karma
16 2.0.0 -> satisfiable by pschwisow/irc-plugin-react-karma [2.0.0].

@PSchwisow

Example 3-2: With Inline Alias

```
1 {  
2     "require": {  
3         "phergie/phergie-irc-bot-react": "dev-bugfix as 2.0.1",  
4         "pschwisow/irc-plugin-react-karma": "~2"  
5     }  
6 }
```

Use most recent version
listed in composer.lock

@PSchwisow





4. Beyond Require

- `require-dev` – Do not install if `--no-dev` flag is set.

Example:

```
"require-dev": {  
    "phpunit/phpunit": "4.5.*",  
    "phake/phake": "2.0.0-beta2",  
    "friendsofphp/php-cs-fixer": "~1"  
},
```



4. Beyond Require

- conflict – This package cannot be used together with the specified package.

Example: Phergie client now handles “pong” functionality internally. If you use it together with plugin, things break.

```
"conflict": {  
    "phergie/phergie-irc-plugin-react-pong": "*"   
},
```



4. Beyond Require

- `replace` – This package can be used in place of the specified package.

Example: You fork a dependency to fix bugs, but you need other dependencies to treat it as the original.

```
"replace": {  
    "other/library": "1.2.3"  
},
```



4. Beyond Require

- **provide** – This package provides an implementation of the specified package.

Example: PSR-3 defines a logger interface. Packages can require psr/log-implementation. Multiple loggers could satisfy this requirement.

```
"provide": {  
    "psr/log-implementation": "1.0.0"  
},
```



4. Beyond Require

- suggest – When installing, provide suggested additional packages.
 - Values are text, rather than version constraints.

Example:

```
"suggest": {  
    "monolog/monolog": "Allows more advanced logging of  
the application flow",  
    "ext-xml": "Needed to support XML format in class Foo"  
},
```




5. Non-Packagist Repositories

- Repository types:
 - Composer – Satis or Toran Proxy
 - VCS – git, Subversion, Mercurial, Fossil
 - PEAR
 - Package – Supports libraries without their own composer.json

Example 5-1: VCS Repositories

```
1 {
2     "require": {
3         "vendor/my-private-repo": "dev-master",
4         "vendor/project-a": "~2"
5     },
6     "repositories": [
7         {
8             "type": "vcs",
9             "url": "git@bitbucket.org:vendor/my-private-repo.git"
10        },
11        {
12            "type": "vcs",
13            "url": "http://svn.example.org/projectA/",
14        }
15    ]
16 }
```

@PSchwisow



Example 5-2: Package Repositories

```
1 {
2     "require": { "smarty/smarty": "3.1.*" },
3     "repositories": [ {
4         "type": "package",
5         "package": {
6             "name": "smarty/smarty",
7             "version": "3.1.7",
8             "dist": {...},
9             "source": {
10                "url": "http://smarty-php.googlecode.com/svn/",
11                "type": "svn",
12                "reference": "tags/Smarty_3_1_7/distribution/"
13            },
14            "autoload": { "classmap": ["libs/"] }
15        }
16    } ]
}
```

@PSchwisow





6. dist vs. source

- Dist – a packaged version (usually a ZIP)
- Source – clone / checkout from VCS
- Packages supply one or both of these.
- If both are available, default behavior:
 - For stable versions, use dist
 - Otherwise, use source



6. dist vs. source

- Pros for dist:
 - Faster installs
 - Archives are cached

```
composer install --prefer-dist  
composer update --prefer-dist
```



6. dist vs. source

- Pros for source:
 - Faster updates (git fetch + git checkout)
 - Vendors are git repos, can be edited

```
composer install --prefer-source  
composer update --prefer-source
```



7. Vendor Binaries

- **Vendor binary** – “Any command line script that a Composer package would like to pass along to a user...”
- Composer creates symlinks to these files in `vendor/bin` directory.



7. Vendor Binaries

Example:

- phergie/phergie-irc-bot-react contains a script “phergie” in its bin directory.
- Its `composer.json` contains:

```
"bin": [  
    "bin/phergie"  
]
```
- My installation of Phergie requires the bot package.
- Composer install creates a symlink:

```
vendor/bin/phergie ->  
    vendor/phergie/phergie-irc-bot-react/bin/phergie
```




8. Scripts and Extra

- **Script** – PHP static method or CLI command executed by composer during its work.
- **Extra** – arbitrary data that can be used by scripts.



8. Scripts and Extra

- Commonly-used events:
 - pre-install-cmd / post-install-cmd
 - pre-update-cmd / post-update-cmd
 - pre-autoload-dump / post-autoload-dump
 - post-create-project-cmd
 - pre-dependencies-solving / post-dependencies-solving
- Complete list:
<https://getcomposer.org/doc/articles/scripts.md#event-names>

Example 8-1: Calling Scripts

```
1 {
2     "scripts": {
3         "post-install-cmd": [
4             "MyVendor\\MyClass::warmCache",
5             "phpunit -c app/"
6         ],
7         "post-autoload-dump": [
8             "MyVendor\\MyClass::postAutoloadDump"
9         ],
10        "post-create-project-cmd": [
11            "php -r \"copy('config/local-example.php', 'config/
12 local.php');\""
13        ],
14    },
15    "extra": { "foo": "bar" }
16 }
```

@PSchwisow



Example 8-2: Script Definitions

```
1 <?php
2 namespace MyVendor;
3 use Composer\Script\Event;
4
5 class MyClass
6 {
7     public static function postAutoloadDump(Event $event)
8     {
9         $extra = $event->getComposer()->getPackage()->getExtra();
10        $vendorDir = $event->getComposer()
11            ->getConfig()->get('vendor-dir');
12        require $vendorDir . '/autoload.php';
13
14        some_function_from_an_autoloaded_file($extra['foo']);
15    }
16 }
```

@PSchwisow



9. Informational Tools

- search – keyword search through current repositories (usually Packagist)

```
$ composer search phergie
```

```
phergie/phergie-irc-connection Data structure for  
containing information about an IRC client connection
```

```
phergie/phergie-irc-client-react IRC client library built  
on React
```

```
phergie/phergie-irc-bot-react IRC bot built on React
```

```
phergie/phergie-irc-plugin-react-command Phergie plugin  
for parsing commands issued to the bot
```



9. Informational Tools

- `show` – Displays currently installed packages.

```
composer show
```

Lists all packages.

```
composer show pschwisow/*
```

Lists all packages under pschwisow vendor.

```
composer show pschwisow/foo
```

Shows detailed information about one package.



9. Informational Tools

- `why (depends)` – Shows which packages depend on the specified package.

```
$ composer why doctrine/lexer
```

```
doctrine/annotations v1.2.7 requires doctrine/lexer (1.*)
```

```
doctrine/common      v2.6.1 requires doctrine/lexer (1.*)
```



9. Informational Tools

- why --tree – Show a recursive tree of dependencies.

```
$ composer why psr/log --tree
```

```
psr/log 1.0.0 Common interface for logging libraries
|- aboutyou/app-sdk 2.6.11 (requires psr/log 1.0.*)
|  `-- __root__ (requires aboutyou/app-sdk ^2.6)
|- monolog/monolog 1.17.2 (requires psr/log ~1.0)
|  `-- laravel/framework v5.2.16 (requires monolog/monolog ~1.11)
|     `-- __root__ (requires laravel/framework ^5.2)
`-- symfony/symfony v3.0.2 (requires psr/log ~1.0)
    `-- __root__ (requires symfony/symfony ^3.0)
```




9. Informational Tools

- why-not (prohibits) – Shows which packages block specified version of a package.

```
$ composer why-not symfony/symfony 3.1
```

```
laravel/framework v5.2.16 requires symfony/var-dumper  
(2.8.*|3.0.*)
```

- -t or --tree also works for why-not.



9. Informational Tools

- `status -v` – Checks for local modifications in your dependencies.

```
$ composer status -v
```

```
You have changes in the following dependencies:  
vendor/seld/jsonlint:
```

```
    M README.mdown
```



9. Informational Tools

- `validate` – Checks if your `composer.json` is valid.

```
$ composer validate
```

`./composer.json` is valid for simple usage with composer but has strict errors that make it unable to be published as a package:
See <https://getcomposer.org/doc/04-schema.md> for details on the schema

description : The property description is required

No license specified, it is recommended to do so. For closed-source software you may use "proprietary" as license.

`require.pschwisow/phergie-irc-plugin-react-karma` : unbound version constraints (`dev-master`) should be avoided



10. Configuration

- Composer can be configured through the `config` section of `composer.json` (root level only).

Example 10-1: Path Configuration

```
1 {
2     "config": { // default values shown below
3         // directories that are generally inside your project
4         "vendor-dir": "vendor", // where package are installed
5         "bin-dir": "vendor/bin", // symlinks to vendor binaries
6
7         // “global” directories
8         // $HOME or ~ can be used to represent your home directory
9         "cache-dir": "$HOME/.composer/cache",
10        "cache-files-dir": "$cache-dir/files", // cache for dist
11        "cache-vcs-dir": "$cache-dir/vcs" // cache of git repos
12    }
13 }
14
15 // P.S. You can't really use comments in JSON.
16
```

@PSchwisow



Example 10-2: More Configuration

```
1 {
2     "config": { // default values shown below
3         // Autoloader searches PHP include path
4         "use-include-path": false,
5
6         // Always optimize autoloader (see --optimize-autoloader)
7         "optimize-autoloader": false,
8
9         // Use only classmap (see --classmap-authoritative)
10        "classmap-authoritative": false,
11
12        // Add to this list if you have GitHub Enterprise
13        "github-domains": ["github.com"]
14    }
15 }
16 // P.S. You can't really use comments in JSON.
```

@PSchwisow



Who Am I?

Feedback / Contact / Slides

- Software Engineer at [Shutterstock](#)
- Zend Certified Engineer – PHP 5 and Zend Framework
- Founder / Organizer of [Lake / Kenosha PHP](#)
- Email: patrick.schwisow@gmail.com
- Twitter: [@PSchwisow](#)
- Slides: github.com/PSchwisow/Miscellaneous/
- Joind.in: <https://joind.in/talk/fde58>
- Composer logo was created by <http://wizardcat.com/>