Composer - Introduction

Presentation by:
PHP as a Social Activity
Sam Likins

4th of April 2013

What is Composer?

Composer is a project **dependency manager**,

not a ~~*package manager*~~.

Inspired by **Ruby Bundler** and **Node.JS npm**

Requires >= **PHP 5.3.2** and

optionally ***Subversion***, ***GIT***, ***Mercurial***

Cross-platform compatible (Windows, Linux, OSX)

Released under the **MIT license**

Developed by: **Nils Adermann** and **Jordi Boggiano**

What Problems are Solved?

- Manages dependent libraries

  - Your project needs other projects

- Handles cascading dependencies

  - These other projects need additional projects

- Declarative dependency

  - Specify the project needs in writing

- Install and update dependencies

  - Download and maintain projects

- Autoload dependencies

  - Configure projects for use in you project

Installing

On Linux and OSX

- **Locally** (within your project)

```
$ curl -sS https://getcomposer.org/installer | php
```

- **Globally** (system wide)

```
$ curl -sS https://getcomposer.org/installer | php
$ sudo mv composer.phar /usr/local/bin/composer
```

Installing

On Windows (Globally)

- **Installer**

Download and run **Composer-Setup.exe**

- **Manually**

```
C:\> cd C:\bin

C:\bin> php -r "eval('?>'. file_get_contents

('https://getcomposer.org/installer'));"

C:\bin> echo @php "%~dp0composer.phar" %*>composer.bat
```

Basic Usage

- Locally

```
$ php composer.phar
```

- Globally

```
$ composer
```

- Display Composer Version

```
$ composer -V
composer version 692015c12b4299dcee92bded7968b21a4b345891
```

Project Setup

Composer uses a **composer.json** file in the root of your project.

To create this file, you can write it manually or run **initialization**.

```
$ composer init

   Welcome to the Composer config generator

 This command will guide you through creating your composer.json
 config.
 Package name (<vendor>/<name>) [vendor/package]:
 Description []: Description of your project
 Author [User Name <name@example.com>]:
 Minimum Stability []:
 License []: GPL-3.0+
 Define your dependencies.
 Would you like to define your dependencies (require) interactively
 [yes]? no
 Would you like to define your dev dependencies (require-dev)
 interactively [yes]? no
```

Project Setup

```
{
    "name": "vendor/package",
    "description": "Description of your project",
    "license": "GPL-3.0+",
    "authors": [
        {
            "name": "User Name",
            "email": "user@example.com"
        }
    ],
    "require": {

    }
}

Do you confirm generation [yes]?
```

A **composer.json** file is now in the root directory of your project.

Packages & Libraries

- A directory with a **composer.json** file is a **package**
- A **package** with a name is a library
- A **library** can be required by another **package**

```
{
    "name": "vendor/package",
    "require": {

    }
}
```

Require

- Vender / Package **Name**
- Package **Version**

Exact Version

```
"require": {
    "vendor/project": "1.2.3"
}
```

Wildcard

```
"require": {
    "vendor/project": "1.2.*"
}
```

Equivalency: **>=1.2, <1.3**

Range

```
"require": {
    "vendor/project": ">=1.2"
}
```

Valid Values: **>**, **>=**, **<**, **<=**, **!=**

Tilde

```
"require": {
    "vendor/project": "~1.2.3"
}
```

Equivalency: **>=1.2.3, <1.3**

Repository

- Package Source Types
  - Composer
  - Version Control Systems
  - PEAR
  - Package
- Declared in **composer.json**
- Specified order is important
- First match is used
- Packagist is checked last

```
{
    "name": "vendor/project",
    "require": {
        "": ""
    },
    "repositories": [
        {
            "type": "",
            "url": ""
        }
    ]
}
```

Packagist

- Default **Composer** repository type
- Enabled by default

```
"repositories": [
    {
        "packagist": false
    }
]
```

- Suggested central repository
- Publish your packages
- Personal Packagist with **Satis**

Version Control System

- Repository type to fetch **GIT**, **Subversion**, and **Mercurial**
- Package Version refers to VCS **Branches** and **Tags**

```
"require": {
    "vendor/package": "dev-master"
},
"repositories": [
    {
        "type": "vcs",
        "url": "https://bitbucket.org/vendor/package.git"
    }
]
```

Platform Packages

Virtual packages are components not installable by **Composer**

- **php** – Allows you to apply constraints on the *PHP version*

```
"require": {
    "php": ">=5.3.2"
}
```

- **ext-<name>** – Allows requiring of *PHP extensions*

```
"require": {
    "ext-curl": "*"
}
```

- **lib-<name>** – Allows version constraints of *PHP libraries*

```
"require": {
    "lib-pcre": ">=7.8"
}
```

Autoloading

Their are currently 3 autoloader types supported

- **PSR-0**
  - Define mapping from **Namespace** to **Path**
- **Classmap**
  - Create collection of **Classname** to **Filename** mappings
- **Files**
  - Require specified files to **EagerLoad** on every request

```
"autoload": {
    "psr-0": { "Vendor\\Namespace\\": "src/" },
    "classmap": [ "lib/", "ext/library.php" ],
    "files": [ "ext/functions.php" ],
}
```

Install & Update

```
$ composer install                  $ composer update
```

- Reads **composer.lock** from current directory if exists
- Else reads **composer.json**
- Resolves dependencies
- Creates / Updates **composer.lock** with exact version info
- Installs / Updates packages in vendor directory

The vendor directory defaults to **vendor** in the root of the project

You can change this behavior with the **vendor-dir** option

```
{
    "vendor-dir": "packages"
}
```

Self Updating

Composer has a simple and straightforward update process.

```
$ composer self-update
```

If you installed globally on Linux or OSX you need to use root privileges.

```
$ sudo composer self-update
Updating to version 692015c12b4299dcee92bded7968b21a4b345891.
    Downloading: 100%
```

Composer is now updated to the latest version.

Additional Resources

- **Aliases** - Alias branch names to versions
- **Custom Installers** - Modify how package types are installed
- **Handling Private Packages** - Host your own repository
- **Scripts** - Callbacks called during package install
- **Trubleshooting** - Solve common pitfalls
- **Vendor Binaries** - CLI scripts from packages

FAQs

- How Do I Install A Package To A Custom Path For My Framework
- Should I Commit The Dependencies In My Vendor Directory
- Why Are Version Constraints Combining Comparisons And Wildcards A Bad Idea
- Why Can't Composer Load Repositories Recursively