Demystifying Composer

David Hernandez
FFW, ffwagency.com

Drupal.org: davidhernandez

FFW

Full Tutorial

https://github.com/david-hernandez/composer-tutorial

Why Composer?

- Builds modern PHP applications
- https://packagist.org/
- Drupal 8 uses it to build the application
- Everyone can use it to build their own application

Getting Started

- Command-line tool. Install if your OS doesn't already have it
- Dependant on your command-line version of PHP
  - Most OSes come with PHP 5.
- **Composer.json** file defines the project
- **Composer.lock** file defines the result of composer.json
- Packages are downloaded into a **vendor** directory

`composer` Command

https://getcomposer.org/doc/03-cli.md

- Finding info and adding dependencies
- Results write **composer.json** and **composer.lock**
- Runs using the PHP default in your shell

$ composer require vendor/package

`composer init`

https://getcomposer.org/doc/03-cli.md#init

- Wizard install
- Writes **composer.json**
- Optional

```
{
  "name": "david-hernandez/myproject",
  "description": "Using the init command to create a new project.",
  "type": "project",
  "authors": [
    {
      "name": "David Hernandez",
      "email": "david@example.com"
    }
  ],
  "minimum-stability": "dev",
  "require": {}
}
```

`composer install`

https://getcomposer.org/doc/03-cli.md#install

- Reads **composer.json** and builds the project
- Writes **composer.lock**
- Writes **vendor** directory
- Commands that gather data and install dependencies are memory hogs

composer require

https://getcomposer.org/doc/03-cli.md#require

```
composer require [vendor]/[package_name]:[version]
```

```
$ composer require drupal/drupal
```

- Adds info to the **require** section of **composer.json**

```
{
    "name": "david-hernandez/myproject",
    "description": "Using the init command to create a new project.",
    "type": "project",
    "authors": [
        {
            "name": "David Hernandez",
            "email": "david@example.com"
        }
    ],
    "minimum-stability": "dev",
    "require": {
        "drupal/drupal": "8.6.x-dev"
    }
}
```

`composer require` Dev

https://getcomposer.org/doc/04-schema.md#require-dev

```
$ composer require drupal/drupal --dev
```

- Adds info to the **require-dev** section of **composer.json**
- Installing with development dependencies is the default
- Use **--no-dev** with the install command to avoid

```
{
    ...
    "require": {
        "drupal/drupal": "8.6.x-dev"
    },
    "require-dev": {
        "drupal/console": "dev-master"
    }
}
```

Version Constraints

https://getcomposer.org/doc/articles/versions.md

Exact	8.5.1	8.5.1
Wildcard	8.5.*	8.5.0, 8.5.9, etc
Range	>=8.5 <8.6	8.5.0, 8.5.9, etc
Stability constraint	8.6.x-dev	Unstable(dev) branch 8.6.x
Increment last digit	~8.5.0	8.5.0, 8.5.9, etc
Increment last two	^8.5.0	8.5.x, 8.6.x, 8.7.x, etc

Implementing Versions

```
"require-dev": {
    "drupal/console": "^1.0.0"
}
```

```
$ composer require drupal/console:^1.0.0 --dev
```

Minimum Stability

https://getcomposer.org/doc/04-schema.md#minimum-stability

```
"minimum-stability": "dev"
```

- Defines the least stable version you find acceptable
- Project-wide or per package
- **dev**, **alpha**, **beta**, **RC**, or **stable**
- Works with version constraints

Minimum Stability

```
"minimum-stability": "dev"

$ composer require drupal/drupal

"drupal/drupal": "8.6.x-dev"
```

```
"minimum-stability": "stable"

$ composer require drupal/drupal

"drupal/drupal": "8.5.6"
```

More on Stability

```
"prefer-stable": true
```

- Alpha, beta, etc versions can be specified but...
- Composer will prefer a stable version if available

Things Needed for Drupal

Drupal projects can’t be built the Composer way without some magic

Why?

- Drupal is an already built product
- It expects its directory structure to be a certain way
- Modules are not PHP packages
- Drupal expects them to be put in the right place
- If you do **composer require drupal/drupal** it will put Drupal into the **vendor** directory

Adding Repositories

https://getcomposer.org/doc/04-schema.md#repositories

- Out-of-the-box Composer will look to packagist.org
- Drupal can be retrieved from packagist.org but not themes and modules
- Neither will packages that are not publicly listed
- You need to tell Composer the location

Adding Repositories

```
"repositories": [
    {
        "type": "composer",
        "url": "https://packages.drupal.org/8"
    }
]
```

Once added to **composer.json** Composer will know where to get Drupal things

Scripts

https://getcomposer.org/doc/articles/scripts.md

- Scripts can be run pre-install, post-install, pre-update, post-update, etc

Scripts

https://github.com/drupal-composer/drupal-scaffold

- For Drupal, the most necessary is a scaffold script
- This script will build a proper webroot and ensure the **vendor** directory, modules, themes, etc, can go in the right place

Scripts

```
"require": {
    ...
    "drupal-composer/drupal-scaffold": "^2.4"
},
"scripts": {
    "drupal-scaffold":"DrupalComposer\\DrupalScaffold\\Plugin::scaffold"
}
```

This will prevent Drupal from ending up in the **vendor** directory

Extra

https://getcomposer.org/doc/04-schema.md#extra

- Used for additional metadata
- Used by the scaffolder to know where to put things
- Added to **composer.json**

Extra

```
"extra": {
    "installer-paths": {
        "docroot/core": ["type:drupal-core"],
        "docroot/libraries/{$name}": ["type:drupal-library"],
        "docroot/modules/contrib/{$name}": ["type:drupal-module"],
        "docroot/profiles/contrib/{$name}": ["type:drupal-profile"],
        "docroot/themes/contrib/{$name}": ["type:drupal-theme"],
        "drush/contrib/{$name}": ["type:drupal-drush"]
    }
}
```

drupal/drupal vs. drupal/core

https://github.com/drupal/core

drupal/drupal

- Complete copy of Drupal 8
- Essentially webroot
- Don’t use

drupal/core

- Subtree split of just the /core directory
- Use with scaffold
- Use this

Other Stuff

Updating

https://getcomposer.org/doc/03-cli.md#update

```
$ composer update

$ composer update drupal/console

$ composer update drupal/ctools drupal/pathauto
```

Patches

https://github.com/cweagans/composer-patches

```
"extra": {
  ...
  "patches": {
    "drupal/some_module": {
      "Text label": "https://www.drupal.org/files/issues/some_patch.patch"
    }
  }
}
```

Path to the patch can be a url or local or within the project

`create-project` Command

https://getcomposer.org/doc/03-cli.md#create-project

- Clone an existing project
- Acts as a starting point
- Must go into an empty directory
  - Think of it no different than **git clone**
- Composer will then run **composer install**

https://github.com/drupal-composer/drupal-project

```
$ composer create-project drupal-composer/drupal-project:8.x-dev
. --stability dev --no-interaction
```

```
composer create-project

drupal-composer/drupal-project:8.x-dev

. (or some directory name)

--stability dev

--no-interaction
```

Things to Remember

- Composer will not install modules for you
  - This isn’t drush
- Be mindful of the **composer.lock**
  - It will contain exactly what happened
- Think ahead how you will manage Composer as a team
- Read the messages
  - It will likely tell you why something didn’t work
- It can be slow