



COMPOSER

Le gestionnaire de dépendances pour PHP



UN PEU DE CONTEXTE

C'est quoi en fait ?



C'est quoi ?

- Composer est un gestionnaire de dépendances pour PHP
- C'est un système qui va nous permettre de définir et de gérer les différentes librairies dont à besoin une application PHP pour fonctionner
- Composer se chargera d'aller automatiquement télécharger ces librairies pour nous
- De plus composer dispose d'un autoloader qui permet de charger automatiquement les librairies dont on a besoin



C'est quoi ?

- Ces librairies seront installées dans un dossier « vendor » situé dans votre projet
- On retrouve ce concept dans d'autres langages comme NPM ou YARN en Js
- Prenons un exemple :
 - *Vous avez un projet qui dépend de librairies*
 - *Certaines de ces librairies dépendent elles mêmes d'autres librairies*
 - *Composer va vous permettre :*
 - De déclarer les librairies dont vous dépendez
 - Automatiquement télécharger les versions correspondantes dans votre projet

Installation

- Il est possible [d'installer Composer](#) sur des systèmes base Unix ou Windows à partir du moment où PHP est installé sur votre machine
- Pour Windows il existe un .exe qui fera tout pour vous et installera Composer globalement
- Pour les systèmes Unix, il va falloir mettre un peu les mains dans le cambouis.
Sachez cependant qu'il est possible d'installer composer localement (à l'échelle de votre projet seulement) ou globalement (à l'échelle de votre système)
- Direction le terminal pour ensuite contrôler Composer

```
# Si Composer est installé  
# localement  
php composer.phar --help  
  
# Si Composer est installé  
# globalement  
composer --help
```

UTILISER COMPOSER

Pour se servir de ses librairies dans nos projets



Init

- Pour définir les différentes dépendants, Composer utilise un fichier : **composer.json** (qui est totalement le jumeau de package.json de NPM)
- Pour le créer, on peut le faire à la main, mais le plus simple est d'utiliser la commande « **composer init** » qui va vous poser plein de questions pour générer ce fichier json
- Pour Composer, votre app est un package comme un autre (comme ceux que vous allez importer) il faut donc lui donner un nom comportant <votre nom>/<nom de l'app>

```
jean-francois@macbook-air-de-jean-francois composer-demo % cd src
jean-francois@macbook-air-de-jean-francois src % composer init
```

```
Welcome to the Composer config generator
```

```
This command will guide you through creating your composer.json config.
```

```
Package name (<vendor>/<name>) [jean-francois/src]: jean-francois/demo
Description []: Une rapide démo de composer
Author [John-Bob-DIRIENZO <jf.dirienzo@gmail.com>, n to skip]:
Minimum Stability []:
Package Type (e.g. library, project, metapackage, composer-plugin) []:
License []:
```

Define your dependencies.

Would you like to define your dependencies (require) interactively [yes]?

Search for a package: markdown

Found 15 packages matching markdown

```
[0] league/commonmark
[1] erusev/parsedown
[2] michelf/php-markdown
[3] league/html-to-markdown
[4] graham-campbell/markdown
[5] cebe/markdown
[6] erusev/parsedown-extra
[7] binarytorch/larecipe
[8] twig/markdown-extra
[9] knplabs/knp-markdown-bundle
[10] aptoma/twig-markdown
[11] dflydev/markdown
[12] s9e/text-formatter
[13] qoraiche/laravel-mail-editor
[14] pixel418/markdownify
```

Enter package # to add, or the complete package name if it is not listed: 2

Enter the version constraint to require (or leave blank to use the latest version):

Using version ^1.9 for michelf/php-markdown

Search for a package:

Would you like to define your dev dependencies (require-dev) interactively [yes]? no

Add PSR-4 autoload mapping? Maps namespace "JeanFrancois\Demo" to the entered relative path. [src/, n to skip]:

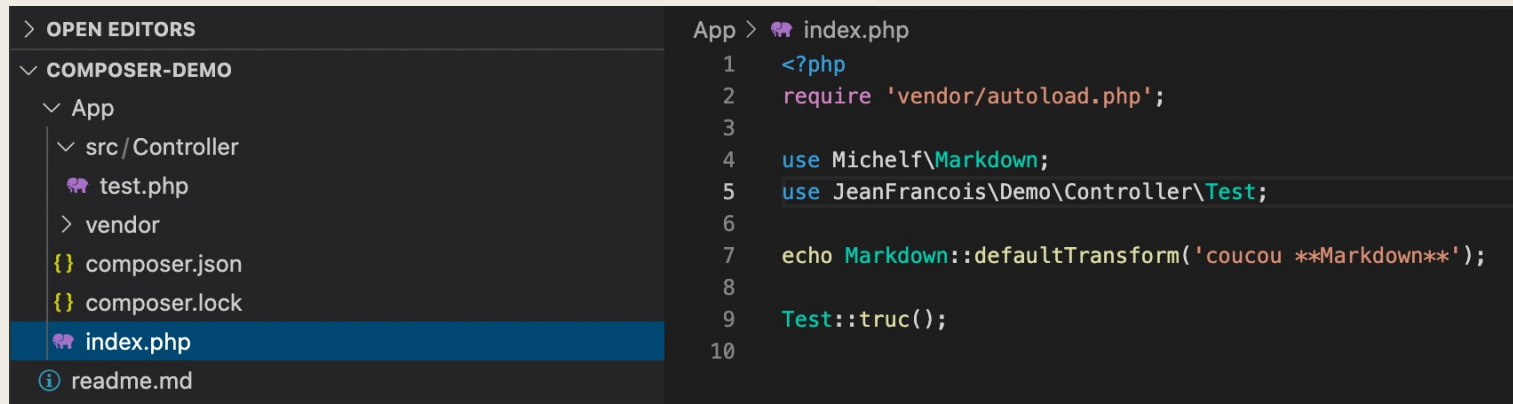
```
{
  "name": "jean-francois/demo",
  "description": "Une rapide démo de composer",
  "require": {
    "michelf/php-markdown": "^1.9"
  },
  "autoload": {
    "psr-4": {
      "JeanFrancois\\Demo\\": "src/"
    }
  },
  "authors": [
    {
      "name": "John-Bob-DIRIENZO",
      "email": "jf.dirienzo@gmail.com"
    }
  ]
}
```

Init

- Ensuite, l'initialisateur va vous demander si vous voulez déclarer des dépendances de suite, dans mon exemple je vais chercher une librairie qui fait du [markdown](#)
- La CLI vous demande si vous avez des dépendances spécifiques au développement à inclure (des librairies de test par exemple)
- Enfin, on vous demande comment vous voulez mapper votre application avec l'autoload. Comme ça, l'autoload de composer prendra aussi en charge les namespaces de votre application (ici « JeanFrancois\Demo » pointera vers /src)

Autoload PSR-4

- Voilà ce que ça donne
- Ma classe Test est déclarée dans le namespace « JeanFrancois\Demo\Controller »
- Vous remarquerez aussi que comme j'ai installé ma dépendance de Markdown, je peux aussi l'utiliser en déclarant juste son namespace, l'autoload se charge du reste



The screenshot shows an IDE interface. On the left, the 'EXPLORER' sidebar displays a project structure under 'COMPOSER-DEMO'. The 'App' directory is expanded, showing 'src/Controller' with 'test.php' and 'vendor' with 'composer.json' and 'composer.lock'. The 'index.php' file is selected and highlighted. On the right, the 'index.php' file is open in the editor, showing the following PHP code:

```
App > index.php
1  <?php
2  require 'vendor/autoload.php';
3
4  use Michelf\Markdown;
5  use JeanFrancois\Demo\Controller\Test;
6
7  echo Markdown::defaultTransform('coucou **Markdown**');
8
9  Test::truc();
10
```

Autoload PSR-4

```
App > {} composer.json > ...
1  {
2      "name": "jean-francois/demo",
3      "description": "Une rapide démo de composer",
4      "require": {
5          "michelf/php-markdown": "^1.9"
6      },
7      "autoload": {
8          "psr-4": {
9              "App\\": "src/"
10         }
11     },
12     "authors": [
13         {
14             "name": "John-Bob-DIRIENZO",
15             "email": "jf.dirienzo@gmail.com"
16         }
17     ]
18 }
```

- C'est super pratique tout ça mais le namespace qui a été choisi est un peu long... Je vais changer ça directement dans mon composer.json
- Je vais spécifier dans la partie « psr-4 » que le namespace « App » sera la racine
- Pour que ces changements prennent effet, je vais devoir taper la commande « composer update » afin que compose régénère tous les fichiers avec les bonnes informations
- Je vais pouvoir maintenant déclarer des namespaces plus court

Les commandes de base

- La première commande à connaître est « **composer require <package name>** ». C'est la commande qui va ajouter une dépendance à notre fichier `composer.json` et l'installer
- Juste avant nous parions de « **composer update** », cette commande fait deux choses :
 - Elle va prendre toutes les informations dans `composer.json` et aller inscrire les versions exactes utilisées dans `composer.lock` (on en reparle après)
 - Une fois ceci fait, elle va run « `composer install` » qui va installer toutes les dépendances avec les versions listées dans `composer.lock` (que nous venons d'updater)
- Une autre commande utile sera forcément « **composer remove <package name>** » qui va... retirer un package et le supprimer du dossier `/vendor`

Composer.lock

- C'est le fichier chargé de « locker » votre projet à une certaine version
- Il est important de commit ce fichier avec Git car il va permettre de s'assurer que toutes les personnes qui utiliseront votre projet le feront avec les bonnes versions des dépendances (donc une version stable)
- Il est aussi important de ne pas confondre « **composer install** » qui va aller installer les dépendances aux versions de composer.lock avec « **composer update** » qui va updater les versions dans composer.lock avant d'exécuter « **composer install** »

Mémo

- [Composer](#) utilise deux fichiers :
 - *Composer.json* : Liste des packages et config de Composer
 - *Composer.lock* : Verrouille les versions précises de vos dépendances pour garder un projet stable
- Les dépendances seront stockées dans le dossier /vendor (à mettre dans votre .gitignore pour éviter de surcharger votre projet inutilement)
- « composer init » pour créer avec CLI votre fichier composer.json
- « composer require <package> » / « composer remove <package> » pour installer / désinstaller des packages
- « composer install » pour installer les dépendances présentes dans composer.json aux versions indiquées dans composer.lock
- « composer update » pour updater composer.lock puis mettre à jour les dépendances de votre projet
- Pour trouver des packages : packagist.org