

Cours Laravel 6 – les bases – un environnement de développement

 laravel.sillo.org/cours-laravel-6-les-bases-un-environnement-de-developpement/

bestmomo

August 28,
2019

Pour fonctionner Laravel a besoin d'un certain environnement :

- un serveur,
- PHP,
- MySQL,
- Node,
- Composer...

Il y a de plus en plus d'environnements disponibles dans le cloud avec des options gratuites comme chez [c9](#). Mais rien ne vaut un bon système local : c'est rapide, sûr et on peut tout gérer. Mais évidemment il faut se le construire !

Heureusement il existe des solutions toutes prêtes, par exemple pour PHP + MySQL : [wampserver](#), [xampp](#), [easyphp](#)...

Ces solutions sont intéressantes mais pour ce cours je vous conseille plutôt [Laragon](#). Il est simple, rapide, convivial, non intrusif, complet, et en plus pensé pour Laravel ! Mais il ne fonctionne que sur Windows.

Pour les utilisateurs de Linux il faut se tourner vers l'une des solutions évoquées ci-dessus ou alors Homestead qui est l'environnement officiel de Laravel. Pour son installation il suffit de suivre la procédure décrite dans la documentation. Par contre je déconseille aux utilisateurs de Windows d'installer [Homestead](#), sauf s'ils ont envie de passer de longues heures à configurer leur système.

Pour les inconditionnel de Mac il faut se tourner vers [Valet](#).

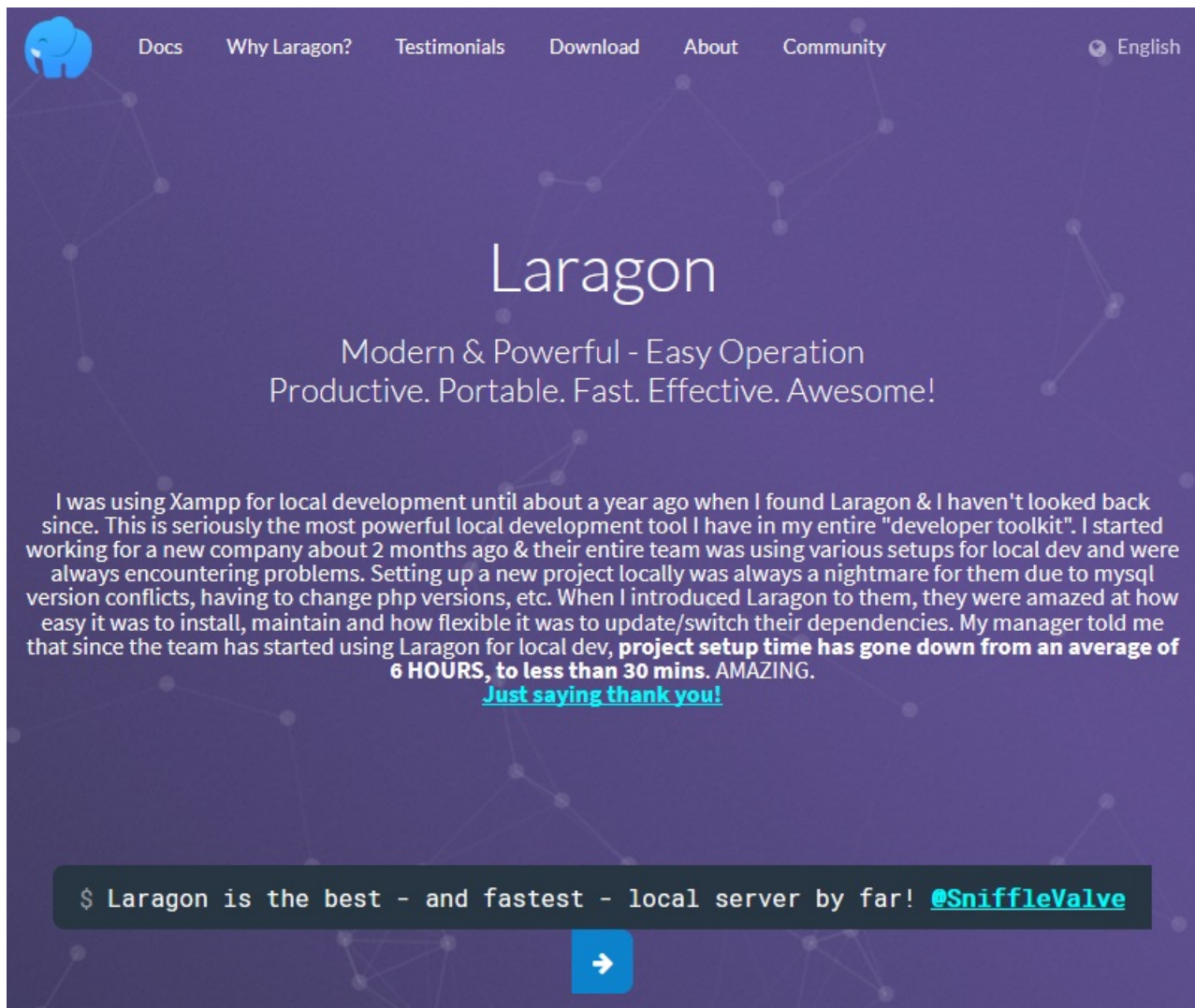
Donc en résumé :

- pour Windows : Laragon,
- pour Linux ou Mac : Homestead,
- pour Mac : Valet.

Laragon

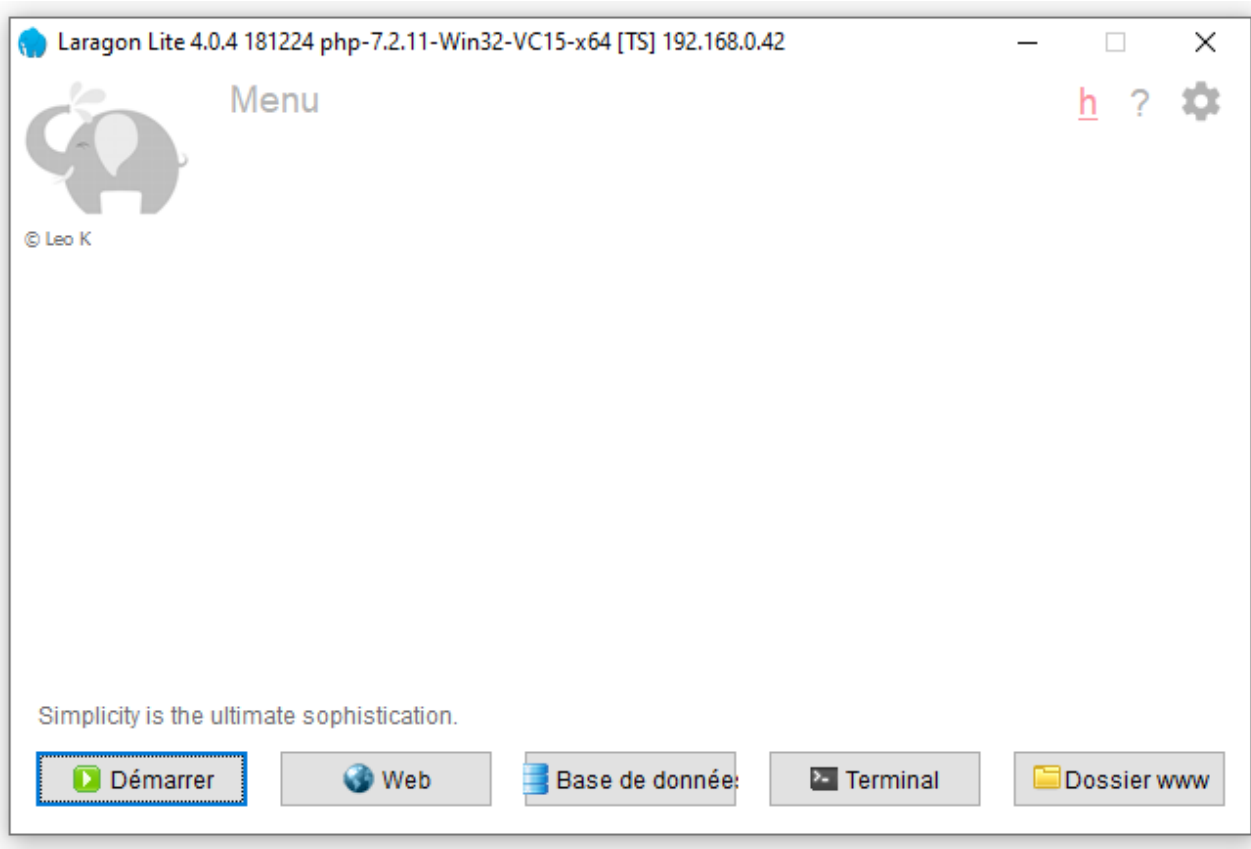
Installation

Pour récupérer Laragon il faut se rendre sur le site :

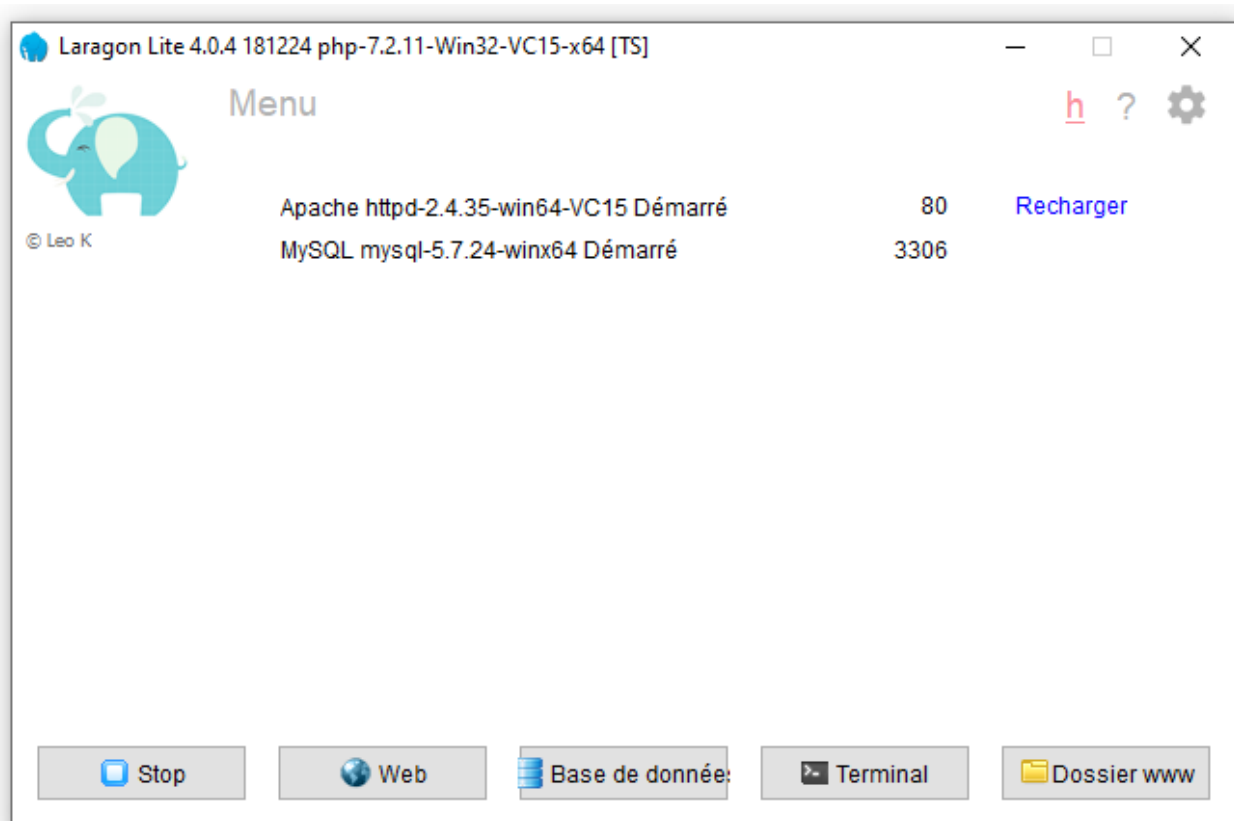


Ne cherchez pas trop de documentation (mais ne vous en faites pas, il n'y en a pas vraiment besoin), il y a surtout un bouton pour aller sur le forum de discussion ainsi qu'un autre pour le téléchargement. Vous verrez que vous avez plusieurs possibilités pour le téléchargement, choisissez celle qui vous convient, a priori Laragon full qui comporte Apache 2.4, Nginx, MySQL 5.7, PHP 7.2, Redis, Memcached, Node.js 11, npm, yarn, git...

Une fois l'installation effectuée vous ouvrez Laragon :



Vous n'avez plus qu'à cliquer sur « Tout démarrer » pour créer et lancer le serveur :



Vous avez maintenant à disposition :

- un serveur Apache
- PHP 7.2

- MySQL 5.7
- le terminal Cmder (une console améliorée par rapport à l'horrible de Windows !)
- Notepad++
- composer
- nodejs
- putty
- memcached
- git
- redis...

Avec Homestead on obtient pratiquement les mêmes possibilités.

Ne vous inquiétez pas si vous ne connaissez pas la moitié de ces outils ! D'une part nous ne les utiliserons pas tous, d'autre part je détaillerai l'utilisation de ceux qui vous seront nécessaires.

Un grand avantage de Laragon c'est qu'il est très simple à compléter. Par exemple ajouter une version de PHP est d'une simplicité enfantine.

Hôte virtuel

Cerise sur le gâteau : Laragon crée automatiquement des hôtes virtuels pour les dossiers qui se trouvent sur le serveur (**www**). Pour ceux qui ne savent pas de quoi il s'agit voici un exemple avec justement le cas de Laravel. Le fichier de démarrage de Laravel est placé en **www/monsite/public/index.html**. Donc à partir de **localhost** il faut entrer : **localhost/monsite/public**. Ce n'est ni pratique ni élégant. Un hôte virtuel permet de définir une adresse simplifiée. Par exemple ici Laragon va définir automatiquement **monsite.oo**. Avouez que c'est quand même mieux !

Mais ce n'est pas seulement une histoire d'esthétique ou d'économie de clavier. Le fait de disposer d'un hôte virtuel permet d'avoir en local exactement le même comportement que sur le serveur de production. Par exemple si vous avez sur une page HTML une image avec ce genre de référence : **/images/bouton.png**, vous serez tout à fait heureux d'avoir un hôte virtuel pour que l'image s'affiche !

Créer manuellement un hôte virtuel avec Windows n'est pas difficile mais un peu laborieux. Il faut modifier le fichier **hosts** de Windows et **httpd-vhosts.conf** de Apache. Autant laisser Laragon s'en charger !

Composer

Présentation

Je vous ai dit dans le précédent chapitre que Laravel utilise des composants d'autres sources. Plutôt que de les incorporer directement, il utilise un gestionnaire de dépendances : **composer**. D'ailleurs pour le coup les composants de Laravel sont aussi traités comme des dépendances. Mais c'est quoi un gestionnaire de dépendances ?

Imaginez que vous créez une application PHP et que vous utilisez des composants issus de différentes sources : **Carbon** pour les dates, **Redis** pour les données... Vous pouvez utiliser la méthode laborieuse en allant chercher tout ça de façon manuelle, et vous allez être confronté à des difficultés :

- télécharger tous les composants dont vous avez besoin et les placer dans votre structure de dossiers,
- traquer les éventuels conflits de nommage entre les librairies,
- mettre à jour manuellement les librairies quand c'est nécessaire,
- prévoir le code pour charger les classes à utiliser...

Tout ça est évidemment faisable mais avouez que s'il était possible d'automatiser les procédures ce serait vraiment génial. C'est justement ce que fait un gestionnaire de dépendances !

JSON

Pour comprendre le fonctionnement de composer, il faut connaître le format **JSON** qui est l'acronyme de **JavaScript Object Notation**. Un fichier JSON a pour but de contenir des informations de type étiquette-valeur. Regardez cet exemple élémentaire :

```
{
  "nom": "Durand",
  "prénom": "Jean"
}
```

Les étiquettes sont « nom » et « prénom » et les valeurs correspondantes « Durand » et « Jean ». Les valeurs peuvent être aussi des tableaux ou des objets. Regardez ce second exemple :

```
{
  "identité1" : {
    "nom": "Durand",
    "prénom": "Jean"
  },
  "identité2" : {
    "nom": "Dupont",
    "prénom": "Albert"
  }
}
```

Composer a besoin d'un fichier **composer.json** associé. Ce fichier contient les instructions pour composer : les dépendances, les classes à charger automatiquement... Par exemple :

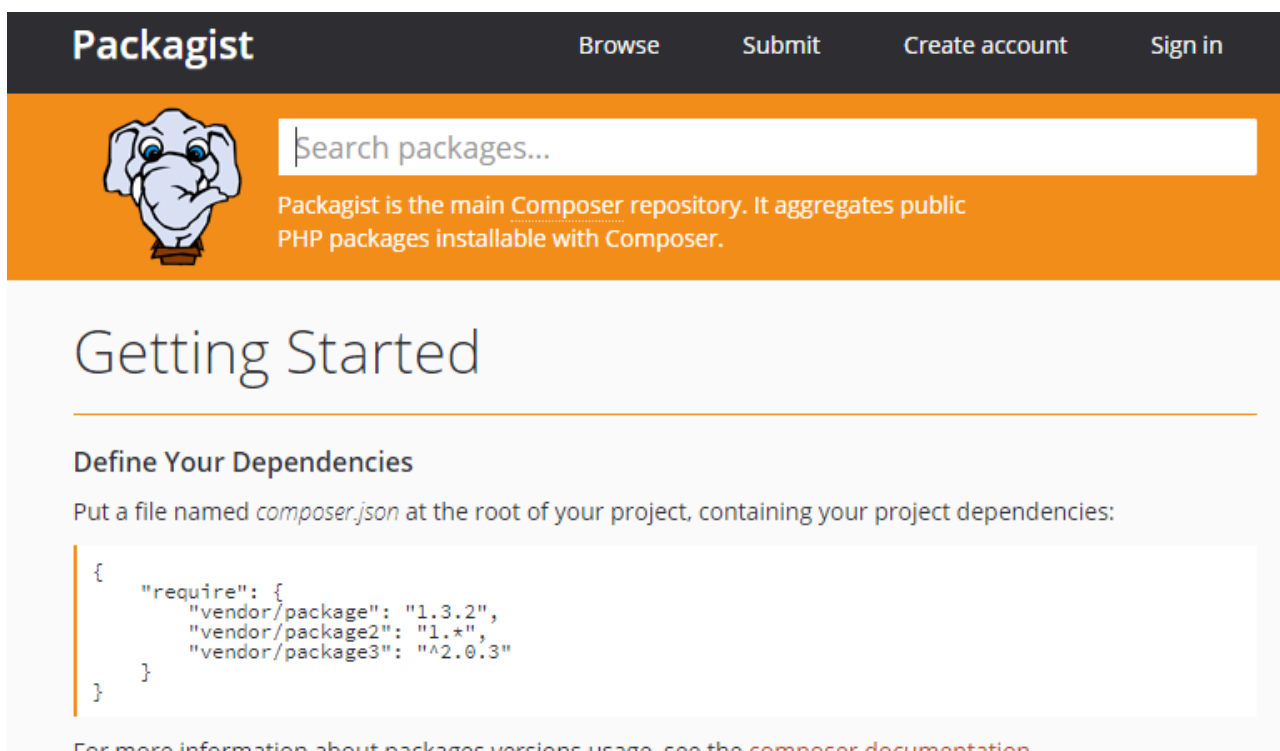
```
"require": {  
    "php": ">=7.2.0",  
    "laravel/framework": "6.0.*",  
},
```

Ici on dit qu'on veut (**require**) que PHP soit au moins en version 7.2.0 et on veut également charger le composant « laravel/framework ».

On verra comment se présente le fichier **composer.json** de Laravel dans le prochain chapitre.

Packagist

Tous les composants disponibles se trouvent sur le site [Packagist](https://packagist.org) :



The screenshot shows the Packagist website. At the top is a dark navigation bar with the 'Packagist' logo and links for 'Browse', 'Submit', 'Create account', and 'Sign in'. Below this is an orange banner featuring a cartoon elephant logo on the left. To the right of the logo is a search bar with the placeholder text 'Search packages...'. Below the search bar, text states: 'Packagist is the main Composer repository. It aggregates public PHP packages installable with Composer.' The main content area has a light gray background and is titled 'Getting Started' in a large font. Underneath, a section titled 'Define Your Dependencies' instructs users to 'Put a file named *composer.json* at the root of your project, containing your project dependencies:'. A code block shows a sample *composer.json* structure:

```
{  
    "require": {  
        "vendor/package": "1.3.2",  
        "vendor/package2": "1.*",  
        "vendor/package3": "^2.0.3"  
    }  
}
```

At the bottom of the code block, a note says: 'For more information about packages versions usage, see the [composer documentation](#)'.

Par exemple il me faut un composant pour l'envoi d'email, j'entre ceci dans la zone de recherche :



email

Packagist is the main Composer repository. It aggregates public PHP packages installable with Composer.

egulias/email-validator

A library for validating emails against several RFCs

PHP ↓ 49 673 937

★ 7 458

swiftmailer/swiftmailer

Swiftmailer, free feature-rich PHP mailer

PHP ↓ 123 886 597

★ 8 416

tijssverkoyen/css-to-inline-styles

CssToInlineStyles is a class that enables you to convert HTML-pages/files into HTML-pages/files with inline styles. This is very useful when you're sending emails.

PHP ↓ 45 503 130

★ 4 958

J'obtiens une liste assez longue et je n'ai plus qu'à fouiller un peu pour trouver ce que je cherche.

Packalyst

Le site Packalyst est spécialisé dans les composants conçus pour Laravel :

packalyst



Looking to hire Laravel developers? Try **LaraJobs**

Search

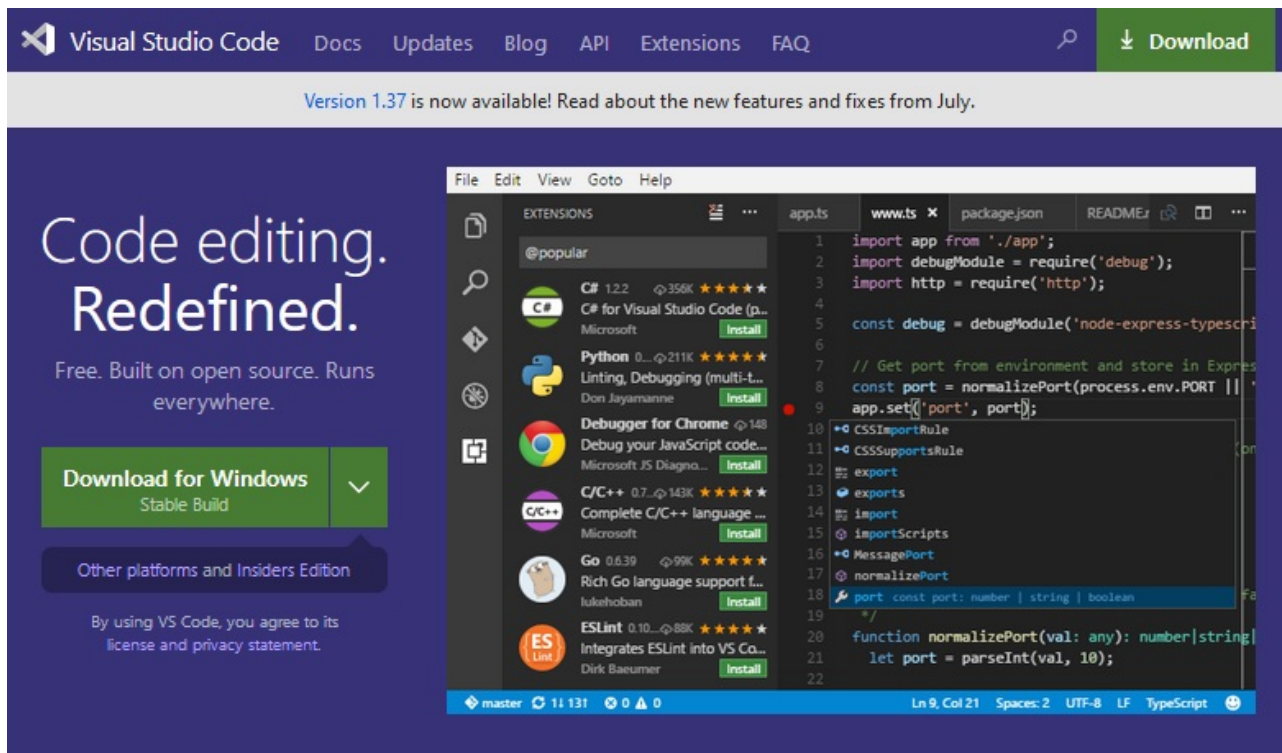


***Packalyst** is a directory of Packages for your Laravel projects*

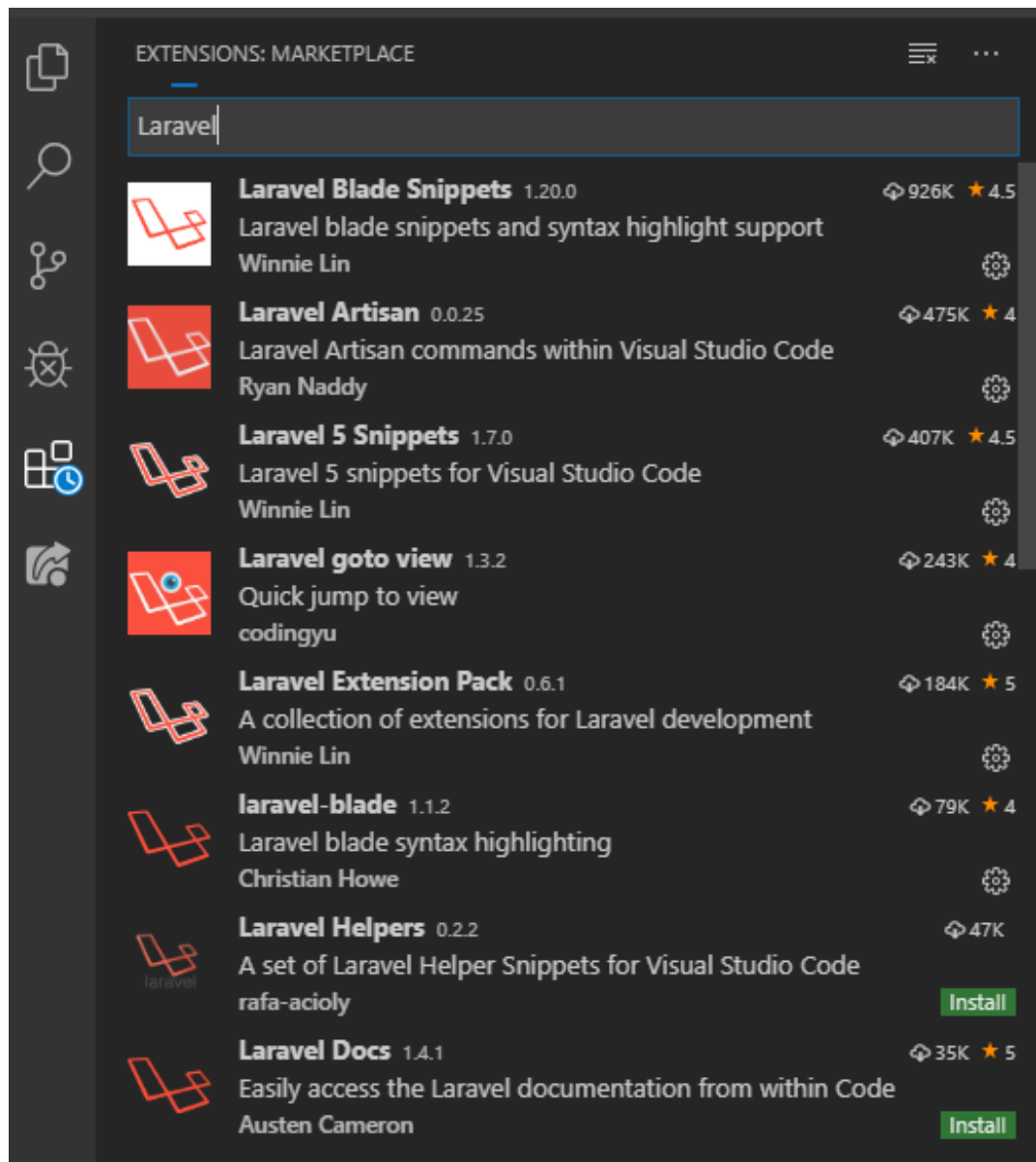
Là vous êtes sûr que le composant va fonctionner directement dans Laravel ! Il faut toutefois vérifier qu'il correspond au numéro de version que vous utilisez.

Les éditeurs de code

Choisir un éditeur de code n'est pas évident, les critères sont nombreux. j'aime bien Visual Studio Code qui jouit d'une grande popularité :



Son grand avantage est de proposer de très nombreux plugins pour étendre ses possibilités, et il y en a pas mal pour Laravel. Vous pouvez installer tous les plugins que vous voulez. Pour trouver ceux qui concernent Laravel vous avez une zone de recherche :



Dans la communauté Laravel l'IDE qui a le plus de succès est PhpStorm. Il est vraiment puissant et complet mais il n'est pas gratuit (sauf pour les étudiants qui ont une carte internationale à jour).

En résumé

- Laravel a besoin d'un environnement de développement complet : PHP, MySQL, composer...
- Il existe des solutions toutes prêtes : Homestead pour Linux, Valet pour Mac et Laragon pour Windows.
- Composer est le gestionnaire de dépendances utilisé par Laravel.
- Visual Studio Code est l'éditeur de code le plus utilisé.
- L'IDE préféré des utilisateurs de Laravel est PhpStorm.