Instalación y uso de Composer en Windows

Vamos a ver paso a paso **como instalar Composer**, el **gestor de dependencias PHP** por excelencia, y algunos ejemplos de cómo usarlo para crear proyectos o instalar librerías de forma ágil.

Requisitos del sistema

Antes de instalar nada debemos comprobar que nuestro equipo será compatible con Composer. Los requisitos mínimos del sistema son:

* Tener instalado PHP 5.3.2 o superior. Si tu instalación de PHP es más reciente que dos años es casi seguro que cumplirá esta condición, pero si aún tienes dudas, puedes comprobar la versión de distintas formas:
  1. Ir a la consola de nuestro sistema, escribir *php -v* (WINDOWS con la variable PATH configurado) *php --version* (LINUX)
  2. Otra forma es ejecutar la función de php *phpversion()* en un script cualquiera php.
  3. Si tienes instalado WAMP en Windows en la ruta de instalación de php, normalmente *C:\wamp\bin\php* dispondrás de una carpeta con el nombre phpX.X.X con la versión de php.
* También se necesitan algunas directivas de configuración de php concretas, pero la instalación de Composer te avisará de cualquier incompatibilidad.

Cómo instalar Composer en Windows

Usando el instalador de Windows

Esta es la forma más sencilla de instalar Composer en Windows.

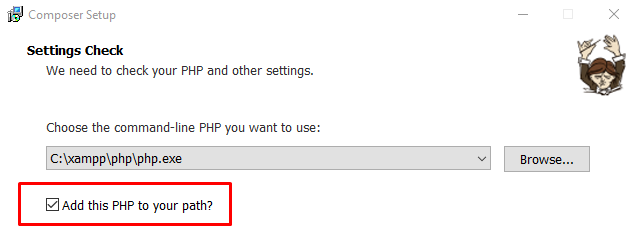
Primero debes descargar el instalador de Composer oficial desde su web: [**instalador composer.exe**](https://getcomposer.org/Composer-Setup.exe)

Después debes ejecutar el fichero descargado. Este instalador lo que hará es instalar la última versión de Composer disponible y establecer la variable de entorno (PATH) para que puedas llamar al comando especifico (*Composer*) desde cualquier lugar de tu PC en la línea de comandos de Windows.

Nota: Si tenías el terminal (cmd) de Windows abierto ciérralo y vuelve a abrirlo. Solo se carga la información de la variable de entorno PATH

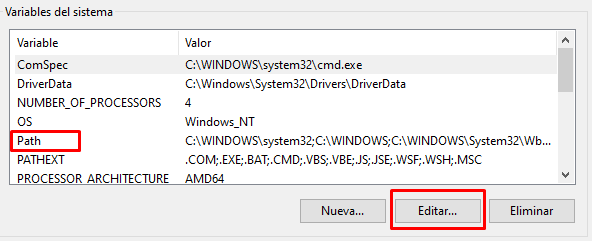
Instalación manual en Windows

En la instalación deberemos señalar la opción del PATH



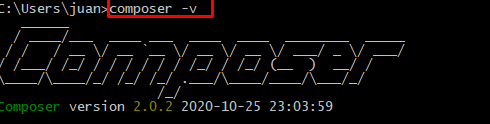
En caso contrario deberemos posteriormente añadirle de forma manual en modo consola o a través de la pantalla de las variables del sistema de Windows.

SET PATH=%PATH%;c:\Composer\



Para comprobar que nuestra instalación es correcta debemos cerrar CMD y volver a abrirlo. Escribimos el siguiente comando y **si se ejecuta con éxito ya lo tenemos listo**:

composer -V



Crear nuevos proyectos con Composer

**Una de las funcionalidades de composer es la instalación de *frameworks* PHP** como symfony, laravel, cakephp, etc. Con Composer podemos descargarnos toda la estructura de carpetas y dependencias del *framework* deseado dejándolo listo el proyecto para empezar a programar.

Para una **instalación limpia de symfony** por ejemplo, podríamos utilizar el siguiente comando:

composer create-project symfony/framework-standard-edition my\_project\_name

o para una versión concreta del framework:

composer create-project symfony/framework-standard-edition my\_project\_name "2.8.\*"

Comandos básicos para trabajar con PHP composer

**composer init**  
Lo ejecutamos en la raíz del directorio en el que se encuentra nuestro proyecto PHP. Al ejecutarlo nos pedirá una serie de información básica sobre nuestro proyecto, nombre del paquete, descripción, dependencias. Las dependencias podemos agregarlas en cualquier momento editando nuestro archivo composer.json el cual es generado de manera automática con este comando.  
**composer install**  
Ejecuta la instalación de las **dependencias definidas** en nuestro fichero composer.json  
**composer require**  
Con este comando instalamos la **dependencia requerida** y la agregamos de manera automática al archivo composer.json  
**Composer update**  
En caso que ocupes descargar la versión más reciente de una dependencia lo podemos realizar ejecutando este comando.

Descargar librerías y dependencias con Composer

Demostraremos la utilidad de composer.json creando un proyecto de muestra.

Nuestro proyecto es un temporizador PHP simple, para desarrolladores que quieren saber cuánto tiempo tarda en ejecutarse un fragmento de código específico. Esto es muy útil para fines de depuración y optimización.

Para hacer esto, siga estos pasos:

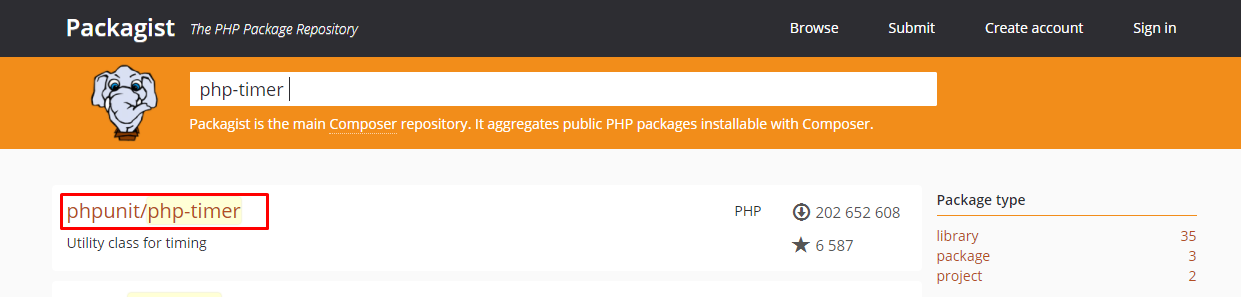
1. Cree un nuevo directorio para el proyecto (dentro del directrio root del servidor que eses utilizando). Dado que nuestro proyecto es un temporizador, vamos a nombrarlo, sin imaginación, **phptimer**. Para hacer esto, abra la terminal y escriba el siguiente comando:



1. Ingrese el directorio recién creado usando el comando:



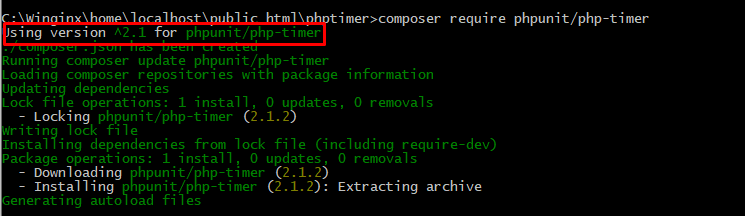
1. Busque un paquete o biblioteca para el proyecto. El mejor lugar para encontrar paquetes de Composer es [Packagist](https://packagist.org/), donde encontrará una serie de bibliotecas para ayudarlo en el desarrollo de su proyecto. Para este tutorial, necesitamos un **paquete de temporizador**. Para obtenerlo, simplemente escriba **Temporizador** en la barra de búsqueda: como puede ver, hay varios temporizadores disponibles para los desarrolladores en el sitio web. Cada paquete tiene un nombre y una breve descripción de lo que hace. En este ejemplo, elegimos el **phpunit / php-timer** ya que tiene más descargas y más estrellas en GitHub.



1. Especifique el paquete deseado para que Composer lo agregue a su proyecto:

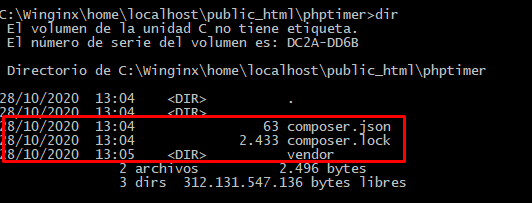
composer require phpunit/php-timer

La salida mostrará la versión de phpunit / php-timer:

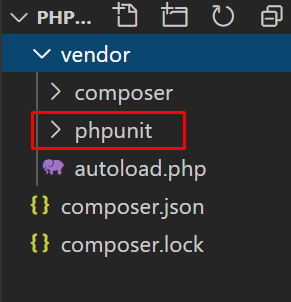


El signo de intercalación (**^)** se define como la opción para una mayor interoperabilidad. Esto significa que **Composer siempre actualizará el paquete** hasta que cierta versión lo rompa de alguna manera.

En nuestro caso, el rango de actualización es **> = 1.0.9 <2.0.0**, ya que la versión 2.0.0 romperá la compatibilidad **al revés**. Para obtener información más detallada sobre las versiones de Composer, visite [**página de documentación**](https://getcomposer.org/doc/articles/versions.md).

Después de ejecutar el comando anterior, Composer creará dos archivos nuevos: **composer.json** y **composer.lock** – en el directorio del proyecto, 

además de una nueva carpeta llamada **vendor**. El directorio de **vendor** es donde Composer almacenará todos sus paquetes y dependencias.



CÓMO UTILIZAR EL SCRIPT DE CARGA AUTOMÁTICA

Su proyecto está casi terminado y lo único que queda por hacer es cargar la dependencia en su script PHP. Afortunadamente, el archivo de carga automática de Composer lo ayudará a completar este proceso de una manera más fácil.

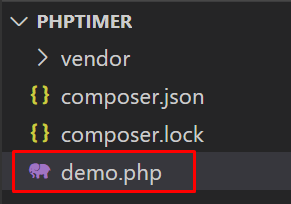
Para usar la carga automática, escriba la siguiente línea antes de declarar o instanciar nuevas variables en su secuencia de comandos:

composer require 'vendor/autoload.php'

Tomemos un ejemplo para ayudarlo a comprenderlo mejor.

Digamos que queremos probar nuestro proyecto. **phptimer**:

1. Abra el editor de texto VSCode y cree un script llamado **demo.php.**



Luego pegue las siguientes líneas en su archivo:

Dirección : https://github.com/sebastianbergmann/php-timer

require \_\_DIR\_\_ . '/vendor/autoload.php';

use SebastianBergmann\Timer\Timer;

Timer::start();

foreach (range(0, 100000) as $i) {

    //echo $i;

}

$duracion = Timer::stop();

var\_dump($duracion);

print Timer::secondsToTimeString($duracion);

1. Ejecute el script:

php demo.php

El terminal deberá mostrar una salida similar a lo siguiente:



CÓMO ACTUALIZAR LAS DEPENDENCIAS DE SU PROYECTO

Todo lo que queda es que sepa cómo actualizar sus paquetes cuando sea necesario. Esto se puede hacer de dos maneras:

* **Actualización universal**. Para verificar e instalar actualizaciones para todos sus paquetes y dependencias a la vez, escriba el siguiente comando en la terminal:

composer update

* **Actualización específica del paquete**. Ejecute este comando para buscar actualizaciones de uno o más paquetes específicos:

composer update vendor/package vendor2/package2

Recuerde reemplazar el **proveedor / paquete** por el nombre del paquete que desea actualizar.

Al ejecutar el comando **actualizar**, Los **Compositor**también actualiza archivos **composer.json** y **composer.lock** para que se correspondan con el estado actual de las instalaciones del proyecto.

EXTRA: ¿Para qué sirve el fichero composer.lock?

Tras la instalación de las dependencias establecidas en el fichero *composer.json* Composer escribirá **las versiones exactas instaladas en un fichero llamado *composer.lock***. Este fichero establece las versiones específicas que usa el proyecto.

Es recomendable incluir el fichero *composer.lock*, junto al *composer.json*, en tus sistema de control de versiones preferido (como GIT).

Cuando realizamos un *install* con Composer se comprueba si existe el fichero *.lock*, si es así, se descargan las versiones exactas indicadas en su interior. Esto significa que, cualquiera que monte el proyecto con Composer tendrá las mismas versiones de las librerías descargadas.

Vamos a instalar otra librería

En este caso la librería Carbón que es una extensión de DataTime

<https://packagist.org/packages/nesbot/carbon>

para ello instalamos la dependencia

composer require nesbot/carbón

require \_\_DIR\_\_ . '/vendor/autoload.php';

use SebastianBergmann\Timer\Timer;

use Carbon\Carbon;

Timer::start();

foreach (range(0, 100000) as $i) {

    //echo $i;

}

$duracion = Timer::stop();

var\_dump($duracion);

print Timer::secondsToTimeString($duracion);

echo '<br>';

$tomorrow = Carbon::now()->addDay();

echo $tomorrow->locale('es')->isoFormat('dddd, MMMM Do YYYY, h:mm');

printf("Now: %s", Carbon::now());

CONCLUSIÓN

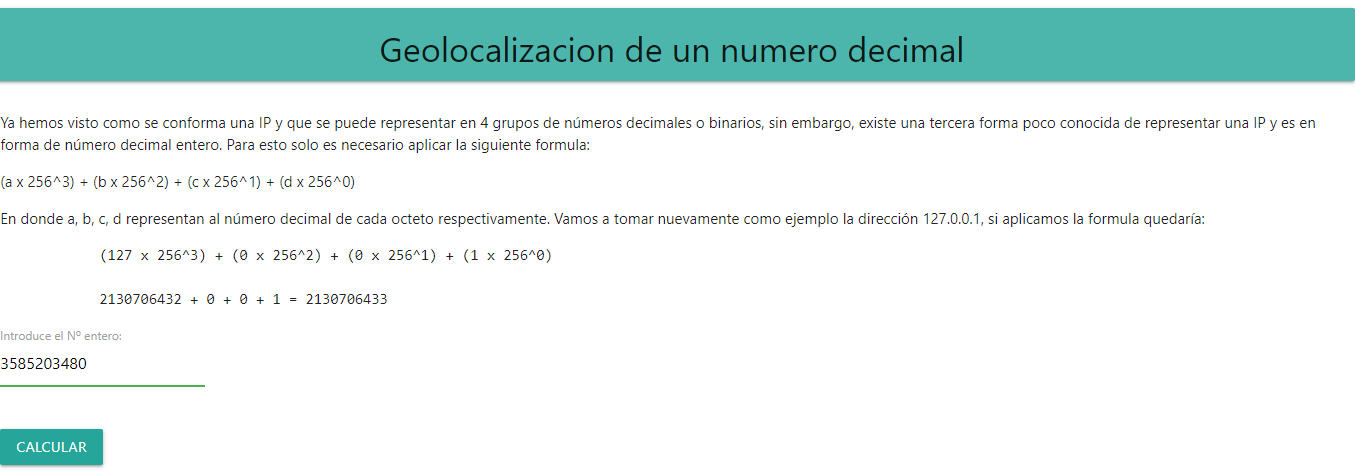
**Composer** ayuda a los desarrolladores a gestionar las dependencias de sus proyectos PHP. Gracias a este software, pueden integrar y administrar fácilmente archivos de código abierto en un solo lugar.

Y lo bueno es que Composer también puede manejar dependencias individualmente para cada proyecto. Con eso, los desarrolladores pueden controlar los paquetes para cada proyecto y mantener el tamaño del proyecto bajo control

Práctica Obligatoria: Composer

Queremos que a partir de un numero decimal nos geolocalize donde se encuentra dicho número.

Para ello deberás utilizar una las librerías de Composer que consideres necesarias. No debes de programar tu ni la conversión de decimal a ipv4 ni la geolocalización de esa ip, solo utilizar las librerías que se encuentran en el [Package Repository de PHP](https://packagist.org/).



El resultado de esto debería ser

