PHP is a great language for building web applications. It is essentially a server-side scripting language that is also used for general-purpose programming. PHP 7 is the latest version, providing major backward-compatibility breaks and focusing on high performance and speed.

This fast-paced introduction to PHP 7 will improve your productivity and coding skills. The concepts covered will allow you, as a PHP programmer, to improve the performance standards of your applications. We will introduce you to the new features in PHP 7 and then run through the concepts of object-oriented programming (OOP) in PHP 7. Next, we will shed some light on how to improve your PHP 7 applications’ performance and database performance. Through this book, you will be able to improve the performance of your programs using the various benchmarking tools discussed.

At the end, the book discusses some best practices in PHP programming to help you improve the quality of your code.

Who this book is written for
This book is for those who have basic experience in PHP programming; if you are developing performance-critical applications, then this book is for you.

What you will learn from this book
- Setup high performance development and production environment for PHP 7
- Discover new OOP features in PHP 7 to achieve high performance
- Improve your PHP applications’ performance
- Attain improved database performance
- Benchmark PHP applications to optimize them
- Write quality code by learning to improve code reusability, simplicity, and expressiveness
- Get rid of the bottlenecks in your PHP 7 applications by writing PHP code optimally
- Tackle issues related to web applications, such as high user dependency and large datasets
In this package, you will find:

- The author biography
- A preview chapter from the book, Chapter 1 'Setting Up the Environment'
- A synopsis of the book’s content
- More information on Learning PHP 7 High Performance
About the Author

Altaf Hussain is a fullstack web and mobile app developer with over 6 years of experience in PHP development. He received his degree in electrical engineering and specialized in computer and communications from Pakistan. Altaf is an electrical engineer on paper and a software engineer by heart.

He worked as a system programmer in his team, developing control software for small test robots using assembly languages and C. After this, Altaf got interested in web technologies and never looked back. He has worked with numerous PHP frameworks, including Zend, Laravel, and Yii, and open source systems such as Drupal, WordPress, PrestaShop, and Magento. Altaf designed and built two proprietary CMS systems with full support for multiple languages and models, permissions, and translations, as well as different kinds of multilingual content management. Now, he works in the fashion industry as the head of IT at shy7lo.com, where his role is to manage the development team on the premises and abroad, in order to manage Magento and Laravel applications development and the deployment life cycle. Besides web apps, Altaf has worked on iOS and Android applications, including building APIs in Lumen. He is a big fan of service-oriented architecture (SOA) and successfully uses it in different applications.

Altaf actively researches on website performance and has deployed the latest technologies, such as PHP 7, NGINX, Redis, Varnish, and others, in production environments for high-speed and scalable applications. He is a Debian lover and uses it for all of his web application deployments.

When not working, Altaf writes articles for programmingtunes.com and techyoccean.com. He has reviewed several books for Packt Publishing, including Learning Phalcon PHP, Mastering jQuery Mobile, and PrestaShop Module Development.
The PHP community faced a huge problem over decades: performance. No matter how powerful hardware they got, in the end, PHP was a bottleneck in itself. With PHP 5.4.x, 5.5.x, and 5.6.x, PHP's performance started to improve, but still it was a huge problem in high-load applications. The community developed caching tools such as Alternative PHP Cache (APC) and Zend OpCache, which cached the opcode for high performance, and these tools had a good effect on the performance.

To get rid of the performance issues of PHP, Facebook built their own open source tool called HHVM (HipHop Virtual Machine). According to their official website, HHVM uses the Just In Time (JIT) compilation to achieve superior performance while maintaining the development flexibility that PHP provides. HHVM had great performance compared to PHP, and it is widely used in production for heavy applications such as Magento.

PHP went to war with HHVM using PHP Next Generation (PHPNG). The whole purpose of PHPNG is to increase performance and focus on the rewriting and optimization of the Zend engine memory allocation and PHP data types. People around the world started benchmarking PHPNG and HHVM, and according to them, PHPNG was outperforming HHVM.

At last, PHPNG was merged with the master branch of PHP, and after a tremendous amount of optimization and complete rewriting, PHP 7 was released with huge performance improvements. PHP 7 is still not JIT, but its performance is great and similar to HHVM. This is a huge performance increase from the older versions of PHP.
What this book covers

Chapter 1, Setting Up the Environment, covers how to set up different development environments, including the installation of NGINX, PHP 7, and Percona Server on Windows, different Linux distros, and setting up the Vagrant virtual machine for development purposes.

Chapter 2, New Features in PHP 7, covers the major new features introduced in PHP 7, including Type Hints, Group use Declarations, Anonymous classes, and new operators, such as Spaceship operator, Null Coalesce operators, and the Uniform variable syntax.

Chapter 3, Improving PHP 7 Application Performance, covers different techniques to increase and scale a PHP 7 application's performance. In this chapter, we cover optimization of NGINX and Apache, CDN, and CSS/JavaScript, such as merging and minifying them, full-page caching, and installing and configuring Varnish. At last, we discuss an ideal infrastructure setup for application development.

Chapter 4, Improving Database Performance, covers techniques to optimize MySQL and Percona Server configuration for high performance. Also, it covers different tools to monitor the performance of a database. It also covers Memcached and Redis for caching objects.

Chapter 5, Debugging and Profiling, covers debugging and profiling techniques including the use of Xdebug for debugging and profiling, debugging with Sublime Text 3 and Eclipse, and the PHP Debugbar.

Chapter 6, Stress/Load Testing PHP Applications, covers different tools to stress and load test the application. It covers Apache JMeter, ApacheBench, and Siege for load testing. It also covers how to load test different open source systems such as Magento, Drupal, and WordPress on PHP 7 and PHP 5.6, and compares their performance on PHP 7 and PHP 5.6.

Chapter 7, Best Practices in PHP Programming, covers a few best practices for producing quality standard code. It covers coding styles, design patterns, service-oriented architecture, test-driven development, Git, and deployments.

Appendix A, Tools to Make Life Easy, discusses three of these tools in much more detail. The tools we will discuss are Composer, Git, and Grunt watch.

Appendix B, MVC and Frameworks, covers MVC design patterns and the most popular frameworks that are used in PHP development, which include Laravel, Lumen, and Apigility.
Chapter 1

Setting Up the Environment

PHP 7 has finally been released. For a long time, the PHP community was talking about it and has still not stopped. The main improvement in PHP 7 is its performance. For a long time, the PHP community faced performance issues in large-scale applications. Even some small applications with high traffic faced performance issues. Server resources were increased, but it did not help much because in the end the bottleneck was PHP itself. Different caching techniques were used, such as APC, and this helped a little. However, the community still needed a version of PHP that could boost the application's performance at its peak. And this is where PHPNG comes in.

PHPNG stands for **PHP next generation**. It is a completely separate branch and is mainly targeted for performance. Some people thought that PHPNG is **JIT (Just In Time)** compilation, but in reality, PHPNG is based on a refactored **Zend Engine**, which was highly optimized for performance. PHPNG is used as a base for PHP 7 development, and according to the official PHP wiki page, the PHPNG branch is now merged into the master branch.

Before starting to build an application, the development environment should be finalized and configured. In this chapter, we will discuss setting up the development environment on different systems, such as Windows and different flavors of Linux.

We will cover the following topics:

- Setting up Windows
- Setting up Ubuntu or Debian
- Setting up CentOS
- Setting up Vagrant

All other environments can be skipped, and we can set up the environment that we will use.
Setting up Windows

There are many tools available that have Apache, PHP, and MySQL bundled for Windows, provide easy installation, and are very easy to use. Most of these tools already provide support for PHP 7 with Apache, such as through XAMPP, WAMPP, and EasyPHP. EasyPHP is the only one that also provides support for **NGINX** and provides easy steps to change webserver from NGINX to Apache or Apache to Nginx.

XAMPP is also available for Linux and Mac OS X. However, WAMP and EasyPHP are only available for Windows. Any of these three can be used for this book, but we recommend EasyPHP as it supports NGINX, and for this book, we mostly use NGINX.

Any of the three tools can be used, but we require more control over every element of our web server tools, so we will also install NGINX, PHP 7, and MySQL individually and then connect them together.

NGINX Windows binaries can be downloaded from [http://nginx.org/en/download.html](http://nginx.org/en/download.html). We recommend using a stable version, though there is no problem with using a mainline version. PHP Windows binaries can be downloaded from [http://windows.php.net/download/](http://windows.php.net/download/). Download either 32-bit or 64-bit binaries of the *non-thread safe* version according to your system.

Perform the following steps:

1. Download NGINX and PHP Windows binaries mentioned in the information box. Copy NGINX to a suitable directory. For example, we have a completely separate D drive for development purposes. Copy NGINX to this development drive or any other directory. Now, copy PHP either to the NGINX directory or to any other secure folder location.

2. In the PHP directory, there will be two `.ini` files, `php.ini-development` and `php.ini-production`. Rename either one of them to `php.ini`. PHP will be using this configuration file.

3. Hold the *Shift* key and right click in the PHP directory to open the command-line window. The command-line window will be opened in the same location path. Issue the following command to start PHP:

   ```bash
   php-cgi -b 127.0.0.1:9000
   ```
The `-b` option starts PHP and binds to the path for external FastCGI servers. The preceding command binds PHP to loop back the `127.0.0.1` IP on port `9000`. Now, PHP is accessible on this path.

4. To configure NGINX, open the nginx_folder/conf/nginx.conf file. The first thing to do is to add root and index to the server block, as follows:

```conf
server {
    root html;
    index index.php index.html index.htm;
}
```

**Downloading the example code**

You can download the example code files for this book from your account at http://www.packtpub.com. If you purchased this book elsewhere, you can visit http://www.packtpub.com/support and register to have the files e-mailed directly to you.

You can download the code files by following these steps:

- Log in or register to our website using your e-mail address and password.
- Hover the mouse pointer on the SUPPORT tab at the top.
- Click on Code Downloads & Errata.
- Enter the name of the book in the Search box.
- Select the book for which you’re looking to download the code files.
- Choose from the drop-down menu where you purchased this book from.
- Click on Code Download.

Once the file is downloaded, please make sure that you unzip or extract the folder using the latest version of:

- WinRAR / 7-Zip for Windows
- Zipeg / iZip / UnRarX for Mac
- 7-Zip / PeaZip for Linux

5. Now, we need to configure NGINX to use PHP as FastCGI on the path mentioned before on which it is started. In the nginx.conf file, uncomment the following location block for PHP:

```conf
location ~ \.php$ {
    fastcgi_pass   127.0.0.1:9000;
}
```
Setting Up the Environment

```bash
fastcgi_param    SCRIPT_FILENAME complete_path_webroot_
folder$fastcgi_script_name;
include    fastcgi_params;
}
```

Note the `fastcgi_param` option. The highlighted `complete_path_webroot_folder` path should be the absolute path to the HTML directory inside the `nginx` folder. Let's say that your NGINX is placed at the `D:\nginx` path; then, the absolute path to the `HTML` folder will be `D:\nginx\html`. However, for the preceding `fastcgi_param` option, `\` should be replaced by `/`.

6. Now, restart NGINX by issuing the following command in the root of the NGINX folder:

   ```bash
   nginx –s restart
   ```

7. After NGINX is restarted, open your browser and enter the IP or hostname of your Windows server or machine, and we will see the NGINX welcome message.

8. Now, to verify the PHP installation and its working with NGINX, create an `info.php` file in webroot and enter the following code in it:

   ```php
   <?php
   phpinfo();
   ?>
   ```

9. Now, in the browser, access `your_ip/info.php`, and we will be presented with a page full of PHP and server information. Congratulations! We configured NGINX and PHP to work perfectly together.

On Windows and Mac OS X, we recommend that you use a virtual machine installed with all the tools on a Linux flavor to get the best performance out of the server. It is easy to manage everything in Linux. There are vagrant boxes available that have everything ready to use. Also, a custom virtual machine configuration with all the tools, including NGINX, Apache, PHP 7, Ubuntu, Debian, or CentOS, and other great ones, can be made at [https://puphpet.com](https://puphpet.com), which is an easy-to-use GUI. Another nice tool is Laravel Homestead, which is a Vagrant box with great tools.
Setting up Debian or Ubuntu

Ubuntu is derived from Debian, so the process is the same for both Ubuntu and Debian. We will use Debian 8 Jessie and Ubuntu 14.04 Server LTS. The same process can be applied to desktop versions for both.

First, add the repositories for both Debian and Ubuntu.

**Debian**

As of the time we're writing this book, Debian does not provide an official repository for PHP 7. So, for Debian, we will use dotdeb repositories to install NGINX and PHP 7. Perform the following steps:

1. Open the `/etc/apt/sources.list` file and add the following two lines at the end of the file:
   
   ```
   deb http://packages.dotdeb.org jessie all
   deb-src http://packages.dotdeb.org jessie all
   ```

2. Now, execute the following commands in the terminal:
   
   ```
   wget https://www.dotdeb.org/dotdeb.gpg
   sudo apt-key add dotdeb.gpg
   sudo apt-get update
   ```

   The first two commands will add dotdeb repo to Debian and the last command will refresh the cache for sources.

**Ubuntu**

As of the time of writing this book, Ubuntu also does not provide PHP 7 in their official repos, so we will use a third-party repo for the PHP 7 installation. Perform the following steps:

1. Run the following commands in the terminal:
   
   ```
   sudo add-apt-repository ppa:ondrej/php
   sudo apt-get update
   ```

2. Now, the repositories are added. Let's install NGINX and PHP 7.

   The rest of the process is mostly the same for both Debian and Ubuntu, so we won't list them separately, as we did for the adding repositories section.
3. To install NGINX, run the following command in the terminal (Debian and Ubuntu):

   ```
   sudo apt-get install nginx
   ```

4. After the installation is successful, it can be verified by entering the hostname and IP of the Debian or Ubuntu server. If we see something similar to the following screenshot, then our installation is successful:

   ![Welcome to nginx on Debian!]
   
   If you see this page, the nginx web server is successfully installed and working on Debian. Further configuration is required.

   For online documentation and support please refer to nginx.org

   Please use the reportbug tool to report bugs in the nginx package with Debian. However, check existing bug reports before reporting a new bug.

   Thank you for using debian and nginx.

The following is a list of three useful NGINX commands:

- `service nginx start`: This starts the NGINX server
- `service nginx restart`: This restarts the NGINX server
- `service nginx stop`: This stops the NGINX server

5. Now, it's time to install PHP 7 by issuing the following command:

   ```
   sudo apt-get install php7.0 php7.0-fpm php7.0-mysql php7.0-mcrypt php7.0-cli
   ```

   This will install PHP 7 along with the other modules mentioned. Also, we installed PHP CLI for the command-line purpose. To verify whether PHP 7 is properly installed, issue the following command in the terminal:

   ```
   php -v
   ```

6. If it displays the PHP version along with some other details, as shown in the following screenshot, then PHP is properly installed:
7. Now, we need to configure NGINX to work with PHP 7. First, copy the NGINX default configuration file `/etc/nginx/sites-available/default` to `/etc/nginx/sites-available/www.packt.com.conf` using the following command in the terminal:

```bash
cd /etc/nginx/sites-available
sudo cp default www.packt.com.conf
```

First, we copied the default configuration file, created another virtual host configuration file, `www.packt.com.conf`, and then created a symbolic link file to this virtual host file in the sites-enabled folder.

It is good practice to create a configuration file for each virtual host by the same name as of the domain so that it can easily be recognized by any other person.

8. Now, open the `/etc/nginx/sites-available/www.packt.com.conf` file and add or edit the highlighted code, as shown here:

```bash
server {
    server_name your_ip:80;
    root /var/www/html;
    index index.php index.html index.htm;
    location ~ \.php$ {
        fastcgi_pass unix:/var/run/php/php7.0-fpm.sock;
        fastcgi_index index.php;
        include fastcgi_params;
    }
}
```

The preceding configuration is not a complete configuration file. We copied only those configuration options that are important and that we may want to change.

In the preceding code, our webroot path is `/var/www/html`, where our PHP files and other application files will be placed. In the index config option, add `index.php` so that if no file is provided in the URL, NGINX can look for and parse `index.php`.

We added a location block for PHP that includes a `fastcgi_pass` option, which has a path to the PHP7 FPM socket. Here, our PHP runs on a Unix socket, which is faster than that of TCP/IP.
9. After making these changes, restart NGINX. Now, to test whether PHP and NGINX are properly configured, create an info.php file at the root of the webroot folder and place the following code in it:

```php
<?php
    phpinfo();
?
```

10. Now, in the browser, type server_ip/info.php, and if you see a PHP configuration page, then congratulations! PHP and NGINX are both properly configured.

If PHP and NGINX run on the same system, then PHP listens to the loopback IP at port 9000. The port can be changed to any other port. In case, we want to run PHP on the TCP/IP port, then in fastcgi_pass, we will enter 127.0.0.1:9000.

Now, let's install **Percona Server**. Percona Server is a fork of MySQL and is optimized for high performance. We will read more about Percona Server in Chapter 3, *Increasing PHP 7 Application Performance*. Now, let's install Percona Server on Debian/Ubuntu via the following steps:

1. First, let's add the Percona Server repository to our system by running the following command in the terminal:

   ```bash
   sudo wget https://repo.percona.com/apt/percona-release_0.1-3.$(lsb_release -sc)_all.deb
   sudo dpkg -i percona-release_0.1-3.$(lsb_release -sc)_all.deb
   ```

   The first command will download the repo packages from the Percona repo. The second command will install the downloaded packages and will create a percona-release.list file at /etc/apt/sources.list.d/percona-release.list.

2. Now, install Percona Server by executing the following command in the terminal:

   ```bash
   sudo apt-get update
   ```

3. Now, issue the following command to install Percona Server:

   ```bash
   sudo apt-get install percona-server-5.5
   ```

   The installation process will start. It will take a while to download it.
For the purpose of this book, we will install Percona Server 5.5. Percona Server 5.6 is also available, which can be installed without any issues.

During the installation, the password for the root user will be asked, as shown in the following screenshot:

![Screenshot of password input](image)

It is optional but recommended to enter the password. After entering the password, re-enter the password on the next screen. The installation process will continue.

4. After the installation is complete, the Percona Server installation can be verified by using the following command:
   
   ```bash
   mysql --version
   ```
   
   It will display the version of Percona Server. As mentioned before, Percona Server is a fork of MySQL, so all the same MySQL commands, queries, and settings can be used.

### Setting up CentOS

CentOS is a fork of Red Hat Enterprise Linux (RHEL) and stands for Community Enterprise Operating System. It is a widely used OS on servers specially used by hosting companies to provide shared hosting.
Let's start by configuring CentOS for our development environment. Perform the following steps:

**Installing NGINX**

1. First, we need to add NGINX RPM to our CentOS installation because CentOS does not provide any default repository for NGINX. Issue the following command in your terminal:

   ```bash
   sudo rpm -Uvh http://nginx.org/packages/centos/7/noarch/RPMS/nginx-release-centos-7-0.el7.ngx.noarch.rpm
   ```

   This will add the NGINX repo to CentOS.

2. Now, issue the following command to see which versions of NGINX are available to install:

   ```bash
   sudo yum --showduplicates list Nginx
   ```

   This will show you the latest stable releases. In our case, it displays NGINX 1.8.0 and NGINX 1.8.1.

3. Now, let's install NGINX using the following command:

   ```bash
   sudo yum install Nginx
   ```

   This will install NGINX.

4. On CentOS, NGINX won't start automatically after installation or restarting. So, first, we will enable NGINX to autostart after a system restarts using the following command:

   ```bash
   systemctl enable Nginx.service
   ```

5. Now, let's start NGINX by issuing the following command:

   ```bash
   systemctl start Nginx.service
   ```

6. Then, open your browser and enter the IP of the CentOS server or host name. If you see the same welcome screen as we saw in the figure earlier in the chapter for Debian, then NGINX is installed successfully.

   To check which version of NGINX is installed, issue the following command in the terminal:

   ```bash
   Nginx -v
   ```

   On our server, the NGINX version installed is 1.8.1.

   Now, our web server is ready.
Installing PHP 7

1. The next step is to install PHP 7 FPM and configure both NGINX and PHP 7 to work together. As of the time of writing this book, PHP 7 is not packaged in official CentOS repositories. So, we have two choices to install PHP 7: either we build it from source, or we use third-party repositories. Building from source is a little bit difficult, so let's go the easy way and use third-party repositories.

   For this book, we will use webtatic repos for the PHP 7 installation as they provide quick updates for the new versions. There are some more repositories, and it is just the reader's choice to use any repository as long as it works.

2. Now, let's add a webtatic repository to our CentOS repo by issuing the following command:

   ```bash
   rpm -Uvh https://mirror.webtatic.com/yum/el7/webtatic-release.rpm
   ```

3. After the repos are added successfully, issue the following command to see which version is available for installation:

   ```bash
   sudo yum --showduplicates list php70w
   ```

   In our case, PHP 7.0.3 is available to install.

4. Now, issue the following command to install PHP 7 along with some modules that may be required:

   ```bash
   sudo yum install php70w php70w-common php70w-cli php70w-fpm php70w-mysql php70w-opcache php70w-mcrypt
   ```

5. This will install core PHP 7 and some modules available for PHP 7. If any other module is required, it can be installed easily; however, first, search to check whether it is available or not. Issue the following command in the terminal to see all the available modules for PHP 7:

   ```bash
   sudo yum search php70w-
   ```

   We will see a long list of all the available modules for PHP 7.

6. Now, let's say that we want to install the PHP 7 gd module; issue the following command:

   ```bash
   sudo yum install php70w-gd
   ```
This will install the gd module. Multiple modules can be installed using the same command and separating each module by a space, as we did in the initial installation of PHP.

Now, to check which version of PHP is installed, issue the following command:

```
php -v
```

In our case, PHP 7.0.3 is installed.

7. To start, stop, and restart PHP, issue the following commands in the terminal:

```
sudo systemctl start php-fpm
sudo systemctl restart php-fpm
sudo systemctl stop php-fpm
```

8. Now, let's configure NGINX to use PHP FPM. Open the default NGINX virtual host file located at `/etc/Nginx/conf.d/default.conf` using either `vi`, `nano`, or any other editor of your choice. Now, make sure that two options are set in the server block, as follows:

```
server {
    listen 80;
    server_name localhost;
    root /usr/share/nginx/html;
    index index.php index.html index.htm;
}
```

The `root` option indicates the web document root where our website source code files will be placed. Index indicates the default files that will be loaded along with extensions. If any of these files are found, they will be executed by default, regardless of any file mentioned in the URLs.

9. The next configuration in NGINX is a location block for PHP. The following is the configuration for PHP:

```
location ~ \.php$ {
    try_files $uri =404;
    fastcgi_split_path_info ^(.+\.php)(/.+)$;
    fastcgi_pass 127.0.0.1:9000;
    fastcgi_index index.php;
    fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
    include fastcgi_params;
}
```
Chapter 1

The preceding block is the most important configuration as it enables NGINX to communicate with PHP. The line `fastcgi_pass 127.0.0.1:9000` tells NGINX that PHP FPM can be accessed on the 127.0.0.1 loopback IP on port 9000. The rest of the details are the same as those we discussed for Debian and Ubuntu.

10. Now, to test our installation, we will create a file named `info.php` with the following contents:

```php
<?php
    phpinfo();
?>
```

After saving the file, type `http://server_ip/info.php` or `http://hostname/info.php`, and we will get a page with complete information about PHP. If you see this page, congratulations! PHP runs alongside NGINX.

## Installing Percona Server

1. Now, we will install Percona Server on CentOS. The installation process is the same, except that it has a separate repository. To add the Percona Server repo to CentOS, execute the following command in the terminal:

   ```bash
   ```

   After the repo installation is completed, a message will be displayed stating the completion of the installation.

2. Now, to test the repo, issue the following command, and it will list all the available Percona packages:

   ```bash
   sudo yum search percona
   ```

3. To install Percona Server 5.5, issue the following command in the terminal:

   ```bash
   sudo yum install Percona-Server-server-55
   ```

   The installation process will start. The rest of the process is the same as for Debian/Ubuntu.

4. After the installation is completed, we will see a completion message.
Setting up Vagrant

Vagrant is a tool used by developers for development environments. Vagrant provides an easy command-line interface to set up virtual machines with all the tools required. Vagrant uses boxes called Vagrant Boxes that can have a Linux operating system and other tools according to this box. Vagrant supports both Oracle VM VirtualBox and VMware. For the purpose of this book, we will use VirtualBox, which we assume is installed on your machine as well.

Vagrant has several boxes for PHP 7, including Laravel Homestead and Rasmus PHP7dev. So, let’s get started by configuring the Rasmus PHP7dev box on Windows and Mac OS X.

We assume that both VirtualBox and Vagrant are installed on our machine. VirtualBox can be downloaded from https://www.virtualbox.org/wiki/Downloads, and Vagrant can be downloaded from https://www.vagrantup.com/downloads.html for different platforms. Details about Rasmus PHP7dev Vagrant Box can be found at https://github.com/rlerdorf/php7dev.

Perform the following steps:

1. Make a directory in one of the drives. For example, we made a php7 directory in our D drive. Then, open the command line in this specific folder directly by holding the Shift key, right-clicking, and then selecting Open command window here.

2. Now, issue the following command in the command window:

   ```
   vagrant box add rasmus/php7dev
   ```

   It will start downloading the Vagrant box, as shown in the following screenshot:
3. Now, when the download is completed, we need to initialize it so that the box is configured and added to VirtualBox for us. Issue the following command in the command window:

```
vagrant init rasmus/php7dev
```

This will start adding the box to VirtualBox and configuring it. When the process is completed, it will display a message, as in the following screenshot:

![Screenshot of Vagrant init command output]

4. Now, issue the following command, which will completely set up the Vagrant box and start it up and running:

```
vagrant up
```

This process will take a little bit of time. When it is completed, your box is ready and running and can be used.

5. Now, the first thing to do after it is up is to update everything. This box uses Ubuntu, so open the command window in the same php7dev directory and issue the following command:

```
vagrant ssh
```

It will connect us with the virtual machines through SSH.

In Windows, if SSH is not installed or not configured in the PATH variable, PuTTY can be used. It can be downloaded from [http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html](http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html). For PuTTY, the host will be 127.0.0.1, and the port will be 2222. Vagrant is both the username and password for SSH.

6. When we are logged in to the box OS, issue the following commands to update the system:

```
sudo apt-get update
sudo apt-get upgrade
```

This will update the core system, NGINX, MySQL, PHP 7, and other installed tools if new versions are available.
7. The box is now ready to use for development purposes. The box can be accessed in the browser by typing its IP address in the browser window. To find the IP address of the box, issue the following command in the SSH-connected command window:

```
sudo ifconfig
```

This will display some details. Find out the IPv4 details there and take the IP of the box.

**Summary**

In this chapter, we configured different environments for the purpose of development. We installed NGINX and PHP 7 on the windows machine. We also configured Debian/Ubuntu and installed NGINX, PHP, and Percona Server 5.5. Then, we configured CentOS and installed NGINX, PHP, and Percona Server 5.5. Lastly, we discussed how to configure Vagrant Box on a Windows machine.

In the next chapter, we will study new features in PHP 7, such as type hints, namespace groupings and declarations, the Spaceship operator, and other features.
Where to buy this book
You can buy Learning PHP 7 High Performance from the Packt Publishing website.
Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.
Click here for ordering and shipping details.