

# Laravel 4 on a shared host

Sometimes you're stuck on shared hosting. Be it client reasons or company policies. Even though Laravel is structured in a very specific way we'll go over some steps to see how we can get Laravel working on a shared hosting environment with minimal requirements.

I'd like to begin by saying that while most solutions below might suit your needs, the ultimate solution is always just to move to a webhost which suits your needs to host a PHP framework like Laravel. [Fortrabb.it](http://fortrabb.it) (<http://fortrabb.it>) is one such solution but definitely not the only one around.

Remember that the absolute minimum requirements always include the need for the MCrypt PHP Extension.

We'll use the following folder structure as a base for this tutorial, with the `www` folder as our DocumentRoot. Your own folder structure on your own shared hosting may differ from the one below.

```
config/  
logs/  
www/
```

Thanks to the following sources and people:

- [Laravel 4 - Easily Extended](http://laravel.io/topic/39/laravel-4-easily-extended) (<http://laravel.io/topic/39/laravel-4-easily-extended>) by [Rob Clancy](http://twitter.com/robboclancy) (<http://twitter.com/robboclancy>)
- [Laravel 4 in Shared Hosting](http://crynobone.com/posts/3/laravel-4-in-shared-hosting) (<http://crynobone.com/posts/3/laravel-4-in-shared-hosting>) by [Mior Muhammad Zaki](https://twitter.com/crynobone) (<https://twitter.com/crynobone>)
- [Understanding the Laravel PUBLIC folder](http://forums.laravel.io/viewtopic.php?pid=10023#p10023) (<http://forums.laravel.io/viewtopic.php?pid=10023#p10023>) by [Shawn](#)

# No DocumentRoot Access

The DocumentRoot is the folder to where your domain points to on your hosting. Laravel requires you to map your domain to the public folder so your application core stays out of access from the outside. But what if you couldn't map your domain to Laravel's public folder? Most shared hosting don't offer this option. There are a couple of ways to solve this, one better than the other.

## 1. Rename the public folder

One thing you could do, if your hosting allowed it is to upload the application to the webroot or a specific folder on your hosting and rename your public folder to the folder's name to which your shared hosting's DocumentRoot points to. Let's apply this to the folder structure for this tutorial.

First we'll create a folder called `laravel` on our shared hosting and we'll upload our application structure to this folder.

```
config/  
laravel/  
  app/  
  bootstrap/  
  vendor/  
  ...  
logs/  
www/
```

After that we'll upload our public folder's content into the `www` folder.

```
config/  
laravel/  
    app/  
    bootstrap/  
    vendor/  
    ...  
logs/  
www/  
    packages/  
    .htaccess  
    index.php  
    ...
```

Now that we've relocated the public folder we should adjust the path structure in the `bootstrap/paths.php` file.

```
# Change this...  
'public' => __DIR__.'/../public',  
  
# ... into this.  
'public' => __DIR__.'/../../www',
```

We should also change the paths in the `www/index.php` file to locate the new `laravel` folder.

```
# These two lines should be changed...  
require __DIR__.'/../bootstrap/autoload.php';  
$app = require_once __DIR__.'/../bootstrap/start.php';  
  
# ... into these two lines.  
require __DIR__.'/../laravel/bootstrap/autoload.php';  
$app = require_once __DIR__.'/../laravel/bootstrap/start.php';
```

We've now correctly relocated our public folder and our Laravel application should work like a charm. You could also have uploaded the Laravel core into the base `/` directory but moving it in a separate folder keeps your folder structure clean.

Another way could have been to symlink your public folder to the `www` folder but there's a small chance that you have access to this ability on a shared hosting.

## 2. Using `.htaccess` with `mod_rewrite`

Before we begin I'd like to mention that the following is highly discouraged. By moving everything into your DocumentRoot you expose your application to the outside which leaves you vulnerable to malicious attacks against your website. At this point you should seriously consider switching to a different webhost which is more suited to host PHP frameworks like Laravel.

That being said, I know first hand that sometimes you're stuck on a webhost because of client reasons, etc. In contrast to the previous method, we're going to upload the entire framework into the DocumentRoot folder. Again we're using the folder structure for this tutorial. We're going to use `.htaccess` to our benefit to redirect all the requests to the public folder.

First, let's move the entire application into the `www` folder (the DocumentRoot).

```
config/  
logs/  
www/  
  app/  
  bootstrap/  
  public/  
    packages/  
    .htaccess  
    index.php  
    ...  
  vendor/  
  ...
```

Now inside your `www` folder, place the following `.htaccess` file.

```
<IfModule mod_rewrite.c>
    RewriteEngine on
    RewriteCond %{REQUEST_URI} !^public
    RewriteRule ^(.*)$ public/$1 [L]
</IfModule>
```

This should redirect all the requests to your public folder. Requests to your public folder to, for example, asset files will still be accepted.

### 3. Move everything into the DocumentRoot

Just like the previous method we're moving the entire application into the DocumentRoot folder. But this time we're going to move the public folder's contents into the DocumentRoot as well.

First, let's move the entire application into the `www` folder (the DocumentRoot).

```
config/
logs/
www/
  app/
  bootstrap/
  public/
    packages/
    .htaccess
    index.php
    ...
  vendor/
  ...
```

Now move the contents of the public folder into the document root as well and remove the public folder.

```
config/  
logs/  
www/  
    app/  
    bootstrap/  
    packages/  
    vendor/  
    .htaccess  
    index.php  
    ...
```

We'll have to adjust the public folder path in the `bootstrap/paths.php` file.

```
# Change this...  
'public' => __DIR__.'/../public',  
  
# ... into this.  
'public' => __DIR__.'/../',
```

And lastly we should also change the paths in the `index.php` file.

```
# These two lines should be changed...  
require __DIR__.'/../bootstrap/autoload.php';  
$app = require_once __DIR__.'/../bootstrap/start.php';  
  
# ... into these two lines.  
require __DIR__.'/bootstrap/autoload.php';  
$app = require_once __DIR__.'/bootstrap/start.php';
```

Your Laravel application should now work.

## ~~Downgrading To PHP 5.3.2~~

~~While Laravel 4 requires at least PHP 5.3.7 because of Bcrypt, there is a way to downgrade to PHP 5.3.2 if you're stuck with that. [Rob Clancy](https://twitter.com/robboclancy) (<https://twitter.com/robboclancy>) has written a great tutorial (<http://laravel.io/topic/39/laravel-4-easily-extended>) about this on how to downgrade to the absolute minimum PHP requirement.~~

**Update:** Laravel 4.2 will require at least PHP 5.4 so you will always have to get webhost with at least these requirements.

# No SSH Access

Most shared hosting don't offer version control support so your deployment options are limited to FTP mostly. This is obviously a bit of a hassle because deployment over FTP is highly discouraged. Which files were edited that should be uploaded? What if I accidentally overwrite some crucial data?

Luckily there are some deployment tools out there which can help.

- [FTPlay](http://ftploy.com/) (<http://ftploy.com/>)
- [Beanstalk](http://beanstalkapp.com/) (<http://beanstalkapp.com/>)
- [Deploy](http://deployhq.com/) (<http://deployhq.com/>)
- [Wercker](http://wercker.com/) (<http://wercker.com/>)
- [Dploy](http://dpjoy.io/) (<http://dpjoy.io/>)
- [Codeship](https://www.codeship.io/) (<https://www.codeship.io/>)

I'm obviously not going to list every single one out there but these should get you started.

No SSH access also means no access to composer, artisan and other command line tools. Once again, FTP is probably your only friend. Should you choose not to use a deployment tool and rather just deploy over FTP, I recommend doing at least the following:

Do a composer install with the `--no-dev` argument before uploading your files. This makes sure your dependencies are optimized for production without the unnecessary development packages. Overwrite your vendor folder with the updated dependencies when you upload through FTP.

## Conclusion

I hope some of these tips helped you on getting Laravel 4 installed on your shared hosting. If you have more tips please share them in the comments and I'll add them to this article.

