# Silverstripe 4 and Nginx

Written by Brett Tasker

Using Nginx as your web server for a Silverstripe 4 CMS website requires a bit of tweaking on both the infrastructure and application in order to get working. These changes are currently required due to Silverstripe 4 asset module's default dependency on the functionality of an Apache web server.

Silverstripe 4 assets module, by default, relies on Apache to handle mismanaged configurations, and to provide "secure assets" via `.htaccess` functionality. As Nginx does not support this functionality, we need to ensure we configure our application to expect this.

## Move /.protected outside of /public

Silverstripe CMS uses a common endpoint for both public and private assets, `ASSET_PATH` (Commonly '`/public`'). It uses a '`/.protected`' folder within this directory to hold the private assets. It relies on the content of the .htaccess file to protect this directory from public access by forcing the request through PHP.

To ensure static content requests from Nginx are unable to access "secure assets" we need to move these files outside of the '`/public`' directory.

Silverstripe 4 provides the ability to do this via the `SS_PROTECTED_ASSETS_PATH` environment variable.

**`.env`**

```
SS_PROTECTED_ASSETS_PATH=/sites/myapp/protected
```

Now we can expose the `/public` directory as a static content directory within Nginx.

## Use `/public` as the root directory in Nginx

Nginx's static content is served from the root directory. To ensure that static file requests are not able to access source files or protected content outside of the `/public` directory, we need to set the Nginx root directory to `/public`.

```
root /sites/myapp/public;
```

## Example Nginx configuration with PHP-fpm

*Example Nginx configuration file*

```
upstream php {
        #server unix:/tmp/php-cgi.socket;
        server 127.0.0.1:9000;
```

```
}

server {
    include mime.types;
    default_type application/octet-stream;

    server_name example.com;
    root /sites/myapp/public;
    index index.php;

    location / {
        try_files $uri $uri/ /index.php?$args;
    }

    location ~ \.php$ {
        #NOTE: You should have "cgi.fix_pathinfo = 0;" in php.ini
        include fastcgi_params;
        fastcgi_intercept_errors on;
        fastcgi_pass php;
        fastcgi_param  SCRIPT_FILENAME $document_root$fastcgi_script_name;
    }

}
```

## Conclusion

Now you can use Nginx to handle your web traffic without the requirement of an Apache layer to handle `.htaccess` file protections.