FastCGI

FastCGI is a deployment option on servers like nginx, lighttpd, and cherokee; see uWSGI and Standalone WSGI Containers for other options. To use your WSGI application with any of them you will need a FastCGI server first. The most popular one is flup which we will use for this guide. Make sure to have it installed to follow along.

Watch Out:

Please make sure in advance that any `app.run()` calls you might have in your application file are inside an `if __name__ == '__main__':` block or moved to a separate file. Just make sure it's not called because this will always start a local WSGI server which we do not want if we deploy that application to FastCGI.

Creating a *.fcgi* file

First you need to create the FastCGI server file. Let's call it *yourapplication.fcgi*:

```
#!/usr/bin/python
from flup.server.fcgi import WSGIServer
from yourapplication import app

if __name__ == '__main__':
    WSGIServer(app).run()
```

This is enough for Apache to work, however nginx and older versions of lighttpd need a socket to be explicitly passed to communicate with the FastCGI server. For that to work you need to pass the path to the socket to the **`WSGIServer`**:

```
WSGIServer(application, bindAddress='/path/to/fcgi.sock').run()
```

The path has to be the exact same path you define in the server config.

Save the `yourapplication.fcgi` file somewhere you will find it again. It makes sense to have that in `/var/www/yourapplication` or something similar.

Make sure to set the executable bit on that file so that the servers can execute it:

```
$ chmod +x /var/www/yourapplication/yourapplication.fcgi
```

Configuring Apache

The example above is good enough for a basic Apache deployment but your *.fcgi* file will appear in your application URL e.g. `example.com/yourapplication.fcgi/news/`. There are few ways to configure your application so that yourapplication.fcgi does not appear in the URL. A preferable way is to use the ScriptAlias and SetHandler configuration directives to route requests to the FastCGI server. The following example uses FastCgiServer to start 5 instances of the application which will handle all incoming requests:

```
LoadModule fastcgi_module /usr/lib64/httpd/modules/mod_fastcgi.so

FastCgiServer /var/www/html/yourapplication/app.fcgi -idle-timeout 300

<VirtualHost *>
    ServerName webapp1.mydomain.com
    DocumentRoot /var/www/html/yourapplication

    AddHandler fastcgi-script fcgi
    ScriptAlias / /var/www/html/yourapplication/app.fcgi/

    <Location />
        SetHandler fastcgi-script
    </Location>
</VirtualHost>
```

These processes will be managed by Apache. If you're using a standalone FastCGI server, you can use the FastCgiExternalServer directive instead. Note that in the following the path is not real, it's simply used as an identifier to other directives such as AliasMatch:

```
FastCgiServer /var/www/html/yourapplication -host 127.0.0.1:3000
```

If you cannot set ScriptAlias, for example on a shared web host, you can use WSGI middleware to remove yourapplication.fcgi from the URLs. Set .htaccess:

```
<IfModule mod_fcgid.c>
   AddHandler fcgid-script .fcgi
   <Files ~ (\.fcgi)>
       SetHandler fcgid-script
       Options +FollowSymLinks +ExecCGI
   </Files>
</IfModule>

<IfModule mod_rewrite.c>
   Options +FollowSymlinks
```

```
    RewriteEngine On
    RewriteBase /
    RewriteCond %{REQUEST_FILENAME} !-f
    RewriteRule ^(.*)$ yourapplication.fcgi/$1 [QSA,L]
</IfModule>
```

Set yourapplication.fcgi:

```
#!/usr/bin/python
#: optional path to your local python site-packages folder
import sys
sys.path.insert(0, '<your_local_path>/lib/python<your_python_version>/s

from flup.server.fcgi import WSGIServer
from yourapplication import app

class ScriptNameStripper(object):
    def __init__(self, app):
        self.app = app

    def __call__(self, environ, start_response):
        environ['SCRIPT_NAME'] = ''
        return self.app(environ, start_response)

app = ScriptNameStripper(app)

if __name__ == '__main__':
    WSGIServer(app).run()
```

Configuring lighttpd

A basic FastCGI configuration for lighttpd looks like that:

```
fastcgi.server = ("/yourapplication.fcgi" =>
    ((
        "socket" => "/tmp/yourapplication-fcgi.sock",
        "bin-path" => "/var/www/yourapplication/yourapplication.fcgi",
        "check-local" => "disable",
        "max-procs" => 1
    ))
)

alias.url = (
    "/static/" => "/path/to/your/static/"
)
```

```
url.rewrite-once = (
    "^(/static($|/.*))$" => "$1",
    "^(/.*)$" => "/yourapplication.fcgi$1"
)
```

Remember to enable the FastCGI, alias and rewrite modules. This configuration binds the application to `/yourapplication`. If you want the application to work in the URL root you have to work around a lighttpd bug with the **`LighttpdCGIRootFix`** middleware.

Make sure to apply it only if you are mounting the application the URL root. Also, see the Lighty docs for more information on FastCGI and Python (note that explicitly passing a socket to run() is no longer necessary).

Configuring nginx

Installing FastCGI applications on nginx is a bit different because by default no FastCGI parameters are forwarded.

A basic Flask FastCGI configuration for nginx looks like this:

```
location = /yourapplication { rewrite ^ /yourapplication/ last; }
location /yourapplication { try_files $uri @yourapplication; }
location @yourapplication {
    include fastcgi_params;
    fastcgi_split_path_info ^(/yourapplication)(.*)$;
    fastcgi_param PATH_INFO $fastcgi_path_info;
    fastcgi_param SCRIPT_NAME $fastcgi_script_name;
    fastcgi_pass unix:/tmp/yourapplication-fcgi.sock;
}
```

This configuration binds the application to `/yourapplication`. If you want to have it in the URL root it's a bit simpler because you don't have to figure out how to calculate `PATH_INFO` and `SCRIPT_NAME`:

```
location / { try_files $uri @yourapplication; }
location @yourapplication {
    include fastcgi_params;
    fastcgi_param PATH_INFO $fastcgi_script_name;
    fastcgi_param SCRIPT_NAME "";
    fastcgi_pass unix:/tmp/yourapplication-fcgi.sock;
}
```

Running FastCGI Processes

Since nginx and others do not load FastCGI apps, you have to do it by yourself. Supervisor can manage FastCGI processes. You can look around for other FastCGI process managers or write a script to run your *.fcgi* file at boot, e.g. using a SysV `init.d` script. For a temporary solution, you can always run the `.fcgi` script inside GNU screen. See `man screen` for details, and note that this is a manual solution which does not persist across system restart:

```
$ screen
$ /var/www/yourapplication/yourapplication.fcgi
```

Debugging

FastCGI deployments tend to be hard to debug on most web servers. Very often the only thing the server log tells you is something along the lines of "premature end of headers". In order to debug the application the only thing that can really give you ideas why it breaks is switching to the correct user and executing the application by hand.

This example assumes your application is called *application.fcgi* and that your web server user is *www-data*:

```
$ su www-data
$ cd /var/www/yourapplication
$ python application.fcgi
Traceback (most recent call last):
  File "yourapplication.fcgi", line 4, in <module>
ImportError: No module named yourapplication
```

In this case the error seems to be "yourapplication" not being on the python path. Common problems are:

- Relative paths being used. Don't rely on the current working directory.
- The code depending on environment variables that are not set by the web server.
- Different python interpreters being used.