PHP and Python with Nginx

FastCGI

CGI Mechanism

The original purpose of a web server was merely to resond to requets from clients by serving te files located on a storage device.

1. Client sends request: `GET /index.html HTTP/1.1` to the web server
2. Web Server process request and sends resopnse `HTTP/1.0 OK or 404 file not found`

Static websites are being progressively abandoned atthe exprense of dynamic ones that contains scripts which are processed by applications such as PHP and Python among others.

1. Client entity (typically web browser) sends request: `GET /index.php HTTP/1.1`
2. Web server pre-processes the reequest (URL rewriting, internal redirects)
3. Web server (Nginx, Apache) forwards request using CGI
4. Backend application (Application, Python, NodeJS) processes request
5. Backed application returns response using CGI
6. Web server post-processes response (gzip compression, character encoding, etc)
7. Web server returns response `HTTP/1.0 200 OK`

When client attmps to visit a dynamic page, web server forwards to the application, processes the script and returns the response to the web server.

To communicate with that ap, CGI protocol was invented in the early 1990s.

Common Gateway Interface

- CGI Protocol v1.1 [RFC3875]
- Allows an HTTP server and a CGI script to share responsibility for resonding a cilent requests
- Server: responsible for managing conection, data transfer, transport, network issues
- CGI script: handles the app issues, data access and document processing
- CGI protocol describes the data exchange between web server and gateway application.

Fast Common Gateway Interface

CGI Protocol is inneficient for servers that are subjects to heavy loads.

- PErsistent processes that handle multiple requests
- server and gateway coomunicates using sockets as TCP or POSIX IPC -> Web Server and application can located in different computers
- It can be implemented on any platform with any programming language.
- Not complex to implement.

Once you have the web server and the backend app running, the only difficulty was remains is to establish the conection between.

1. configure nginx perspective to communicate with the FastCGI application via FastCGI module.

Nginx CGI-derived Module Implementations

- uWSGI module allows Nginx to communicate with applciations through the uwsgi protocol.
- SCGI, variant of CGI like FastCGI. SCGI interfaces and modules can be found in a variety of software projects such as Apache, IISm Java, Cherokee, etc.

FastCGI module	uWSGFI equivalent	SCGI equivalent
`fastcgi_pass`	`uwsgi_pass`	`scgi_pass`
`fastcgi_cache`	uswsgi_cache	scgi_cachje
fastcgi_temp_path	uwsgi_temp_path	scgi_temp_path

FastCGI Main Directives

fastcgi_pass

- location, if
- for TCP: `fastcgi_pass hostname:port`
- for Unix: `fastcgi_pass unix:/path/to/fastcgi.socket;`
- to refer upstream blocks: `fast_pass myblock`

```
# examples
fastcgi_pass localhost:9000;
fastcgi_pass 127.0.0.1:9000;
fastcgi_pass unix:/tmp/fastcgi.socket;
# Using an upstream block
upstream fastcgi {
    server 127.0.0.1:9000;
    server 127.0.0.1:9001;
}
location ~* \.php$ {
    fastcgi_pass fastcgi;
}
```

fastcgi_param

Allow to config the request passed to FastCGI

- http, server, location
- SCRIPT_FILENAME and QUERY_STRING are required
- for POST requests REQUEST_METHOD, CONTENT_TYPE, CONTENT_LENGTH are required.

```
fastcgi_param SCRIPT_FILENAME /home/website.com/www$fastcgi_script_name;
fastcgi_param QUERY_STRING $query_string;
```

```
fastcgi_param REQUEST_METHOD $request_method;
fastcgi_param CONTENT_TYPE $content_type;
fastcgi_param CONTENT_LENGTH $content_length;
```

the fastcgi_params file in Nginx config folder already includes all necessarsy parameter definitios you need.

If the parameter begins with HTTP_ it will override the potentially existing HTTP headers of the client request

fastcgi_bind

- http, server, location
- blinkds the socket to a local ip address, allowing to specify the network interface to use for FastCGI communications
-

fastcgi_pass_header

- specify additional headers that should be passed to the FastCGI server

fastcgi_hide_header

headers that should be hidden from the FastCGI server)Nginx does not forward)

fastcgi_index

FastCGI server does not support automatic directory indexes. IF the requestd URI ends with a /, nginx appends the value fastcgi_index

- fastcgi_index filename;
- fastcgi_index index.php;

Others fastcgi_* directives

- ignore_client_abort
- intercepts_errors
- read_timeout
- connect_timeout
- send_timeout
- split_path_info
- store
- store_access
- temp_path
- max_temp_file_size

- temp_file_write_size
- send_lowat
- pass_request_body
- ignore_headers
- next_upstream
- next_upstream_timeout
- next_upstream_tries
- catch_stderr
- keep_conn
- force_ranges
- limit_rate

Others caching and buffering

- cache
- cache_key
- cache_methos
- cache_min_uses
- cache_path
- cache_use_stale
- cache_valid
- no_cache
- cache_bypass
- cache_revalidate
- bvuffering
- buffers

Config example

```
fast_cacye phpcache;
fastcgi_cache_key "$scheme$host$request_uri"; # request_uri includes the
request arguments such as /page.php?arg=value
fastcgi_cache_min_uses 2; # after 2 hits, a request receives a cached
response
fastcgi_cache_path /tmp/cache levels=1:2 keys_zone=phpcache:10m
inactive=30m max_size=500M
fastcgi_cache_use_stale updating timeout;
fastcgi_cache_valid 404 1m;
fascgi_cache_valid 500 502 504 5m;
```

These directives are valid for pretty much any virtual host configuration. May to save thewse in a separate file `fastcgi_cache` that you can include at the appropiate place

```
server {
    server_name website.com;
    location ~* /.php$ {
        fastcgi_pass 127.0.0.1:9000;
```

```
        fastcgi_param SCRIPT_FILENAME
/home/website.com/www$fastcgi_script_name;
        fastcgi_param PATH_INFO $fastcgi_script_name;
        include fastcgi_params;
        include fastcgi_cache;
    }
}
```

PHP With Nginx

PHP 5+ came with FastCGI process manager. PHP should be confiured with --enable-fpm argument.

Architecture

PHP supports the FastCGI protocol. The PHP binary processes scvripts and is stable to interact with nginx via sockets. The FastCGI Process Manager is kown as PHP-FPM

PHP-FPM

- Script tha tmanages the PHP processes.
- It awaits and receives instructions from Nginx
- Automatically daemonizing PHP
- Execute scripts in chrooted environment
- Improved logging
- IP address restrictions

Nginx Configuration

```
server {
    server_name .website.com;
    listen 80;
    root /home/website/www;
    index index.php;
    location ~* /.php$ { # for requests ending with .php
        # Specify the listening address and port configured in php
        fastcgi_pass 127.0.0.1:9000;
        # the document path to be passesd to PHP-FPM
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        # the script filename to be passed to PHP-FPM
        fastcgi_param PATH_INFO $fastcgi_script_name;
        # include other FastCGI realted configuration settings
        include fastcgi_params;
    }
}
```

```
/usr/local/nginx/sbin/nginx -s reload
or
service nginx reload
```

```
echo "<?php phpinfo(); ?>" > /home/website/www/index.php
```

load http://localhost/ or website URL.