Using NGINX and NGINX Plus as an Application Gateway with uWSGI and Django

Configure NGINX and NGINX Plus as an application gateway for uWSGI and Django.

This article explains how to use NGINX or NGINX Plus as an application gateway with uWSGI and Django.

Introduction

NGINX is a high-performance, scalable, secure, and reliable web server and a reverse proxy. NGINX enables all the main web acceleration techniques for managing HTTP connections and traffic. For many years, NGINX capabilities such as load balancing, SSL termination, connection and request policing, static content offload, and content caching have helped NGINX users to build reliable and fast websites quickly and efficiently.

NGINX can also act as a secure application gateway, offering a number of specialized built-in interfaces to pass traffic from users to applications. In this regard, not only can NGINX proxy HTTP and HTTPS traffic to an HTTP-enabled application container, it can also directly talk to most of the popular lightweight application servers and web frameworks via optimized app-gateway interfaces implemented in modules like FastCGI, Memcached, scgi, and uwsgi.

Most commonly used application containers have embedded external HTTP interfaces with some routing capabilities, but one important reason to use NGINX as an application gateway is that it provides an all-in-one solution for HTTP connection management, load balancing, content caching, and traffic security. The application backend sits securely behind NGINX for better scalability and performance. It is also very easy to cluster application instances behind NGINX to build highly available applications.

About uWSGI and Django

A few words about "specialized interfaces". As useful as it is, HTTP has never been optimized for modern, lightweight application-deployment scenarios. In recent years, a number of standardized interfaces have evolved for use with various application frameworks and application containers. One of these interfaces is the Web Server Gateway Interface (WSGI), an interface between a web server/proxy and Python-based applications.

One of the most commonly used application servers offering the uwsgi protocol – its own implementation of the WSGI protocol – is the uWSGI application server container.

Other than that, the uWSGI application server supports HTTP, FastCGI, and SCGI – with the uwsgi protocol being recommended as the fastest way to talk to applications.

Configuring NGINX and NGINX Plus for Use with uWSGI and Django

This document provides an example of how to configure NGINX and NGINX Plus for use with a uWSGI server and a Python development environment.

NGINX 0.8.40 and later (and all releases of NGINX Plus) includes native support for passing traffic from users to Python applications via the uwsgi protocol. If you download NGINX Open Source binaries or source from our official repositories, or NGINX Plus from the customer portal, you don't have to do anything to enable support for the uwsgi protocol – NGINX and NGINX Plus support uswgi by default.

Configuring the uWSGI application container itself is outside the scope of this document,; refer to the excellent Quickstart for Python/WSGI applications for more information.

Django is probably the most commonly used Python web framework, so for simplicity's sake the example uses a Django-based setup for the Python app. The Django documentation provides extensive information on how to configure a Django environment.

For illustrative purposes only, this is one way you might invoke your uWSGI server with Django:

```
--chdir=/var/django/projects/myapp \
--module=myapp.wsgi:application \
--env DJANGO_SETTINGS_MODULE=myapp.settings \
--master --pidfile=/usr/local/var/run/uwsgi/project-master.pid \
--socket=127.0.0.1:29000 \
--processes=5 \
--uid=505 --gid=505 \
--harakiri=20 \
--max-requests=5000 \
--vacuum \
--daemonize=/usr/local/var/log/uwsgi/myapp.log
```

With these options in place, here's a sample NGINX configuration for use with a Django project:

```
http {
    # ...
    upstream django {
        server 127.0.0.1:29000;
    }

    server {
        listen 80;
        server_name myapp.example.com;
        root /var/www/myapp/html;

        location / {
            index index.html;
        }

        location /static/  {
            alias /var/django/projects/myapp/static/;
        }

        location /main {
            include /etc/nginx/uwsgi_params;
            uwsgi_pass django;
            uwsgi_param Host $host;
            uwsgi_param X-Real-IP $remote_addr;
            uwsgi_param X-Forwarded-For $proxy_add_x_forwarded_for;
            uwsgi_param X-Forwarded-Proto $http_x_forwarded_proto;
        }
    }
}
```

Notice that the configuration defines an upstream called **django**. The port number on the server in the group, 29000, matches the one the uWSGI server binds to, as specified by the `socket` argument in the sample `uwsgi` command.

Serving of static content is offloaded to NGINX or NGINX Plus, which serves it directly from **/var/django/projects/myapp/static**. Traffic to the application at **/main** is proxied and bridged from HTTP to the uwsgi protocol and passed to the Django app running inside a uWSGI application container.

Conclusion

Lightweight, heterogeneous application environments are becoming an increasingly popular way of building and deploying modern web applications. Newer, standardized application interface protocols like uwsgi and FastCGI enable faster communication between users and applications.

Using NGINX and NGINX Plus in front of an application container has become a common way to free applications from the burden of HTTP traffic management, and to protect the application from unexpected spikes of user traffic, malicious behavior, denial-of-service (DoS) attacks, and more. Unbundling real-world, external HTTP traffic from the actual application allows the developer to fully focus on the application logic, and leave the web acceleration and fundamental HTTP traffic security tasks to NGINX or NGINX Plus.

##Resources

- NGINX support in the uWSGI project documentation
- How to use Django with uWSGI in the Django project documentation

Found a bug? Looking for something new? LET US KNOW

Copyright © F5, Inc. All rights reserved.
Trademarks | Policies | Privacy | California Privacy | Do Not Sell My Personal Information | Cookie Preferences