DATADOG

NGINX 502 Bad Gateway: Gunicorn

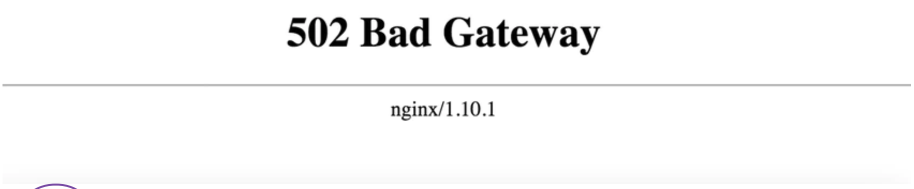
Evan Mouzakitis @vagelim

tip / nginx / 502 bad gateway

Published: December 8, 2016

This post is part of a series on troubleshooting NGINX 502 Bad Gateway errors. If you're not using Gunicorn, check out our other article on troubleshooting NGINX 502s with PHP-FPM as a backend.

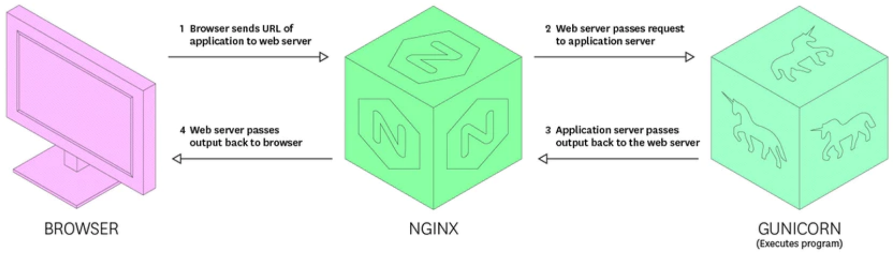
502 Bad Gateway errors can occur for a number of reasons. Determining the exact cause of 502s varies depending on the webserver in use, as well as the application server interpreting the request.



502 Bad Gateway

nginx/1.10.1

underlying issue is either excessive latency or exceedingly short timeout windows.


1 Browser sends URL of application to web server
2 Web server passes request to application server
3 Application server passes output back to the web server
4 Web server passes output back to browser
BROWSER
NGINX
GUNICORN
(Executes program)

In others, 502s are caused by a misconfigured application server—the webserver understood the request and passed it along to the appropriate handler, but a disruption of some sort occurred between the two.

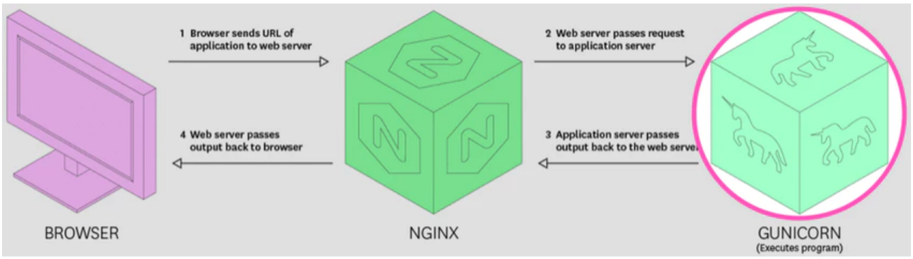
Diagnosing the cause of 502 bad gateway errors mostly depends on the application server you're using, and we've laid out solutions for a couple of common issues related to the Gunicorn application server below.

Common issues

Gunicorn can cause 502s under the following conditions:

- Gunicorn is not running
- Gunicorn won't start
- NGINX is misconfigured
- Gunicorn is timing out

Gunicorn isn't running


1 Browser sends URL of application to web server
2 Web server passes request to application server
3 Application server passes output back to the web server
4 Web server passes output back to browser
BROWSER
NGINX
GUNICORN
(Executes program)

Like PHP-FPM, the easiest way to tell if Gunicorn is running is with `ps`. If you see something similar to the output below, move on to the next step.

```
www-data@nginx0:/var/log/nginx$ ps aux | grep
gunicorn www-data 13805 0.0 1.8 52292 18460 pts/0 S
20:32 0:00 /home/www-data/test_app/bin/python
```

```
data/test_app/bin/gunicorn --error-logfile
/var/log/gunicorn/errors.log -b 0.0.0.0:8080 wsgi
```

If not, try starting your Gunicorn daemon.

Gunicorn won’t start

If Gunicorn won’t start, a typo in the configuration file (if you use one), port conflict, inaccessible log directory, or any combination thereof, could be the cause in which case remedying the situtation is straightforward.

To check your Gunicorn configuration, run:

`gunicorn --check-config [APP_MODULE]`

***Note**: Command line options override configuration files*

However, if Gunicorn *is* running but still causing 502s, the root cause of the error may be in your application.

NGINX is misconfigured

If Gunicorn is properly configured, it may be that NGINX is not looking for it in the right place. Open your NGINX configuration file (`/etc/nginx/nginx.conf`) and look for a block starting with the "upstream" keyword, like so:

```
upstream app_servers { server 127.0.0.1:8080; }
```

The above block tells NGINX to redirect requests for "app_servers" to, in this case, `127.0.0.1:8080`. If Gunicorn is not bound to `127.0.0.1` or is not listening on `8080`, either change Gunicorn's configuration to match NGINX's, or change NGINX's configuration to match Gunicorn's.

Last but not least, verify that your site configuration is redirecting your application to the appropriate (in our example, the Python application is served from `/my-app`):

```
location /my-app { proxy_pass http://app_servers;
proxy_redirect off; proxy_set_header Host $host;
proxy_set_header X-Real-IP $remote_addr;
proxy_set_header X-Forwarded-For
$proxy_add_x_forwarded_for; proxy_set_header X-
Forwarded-Host $server_name; }
```

Notice how the `proxy_pass` directive must match the name given to the upstream block in your `nginx.conf` configuration file.

Gunicorn is timing out

If your application takes a long time to respond (> 30s by default), Gunicorn may return a 502 to NGINX. You can verify that the problem is with the application by checking your Gunicorn logs (defaults to STDOUT if no logfile set).

because Gunicorn thinks the worker hung:

```
[2016-09-21 20:33:04 +0000] [13805] [CRITICAL] WORKER
TIMEOUT (pid:13810)
```

Depending on what the application is supposed to do, a long execution time (> 30s) may be expected. In that case, increasing Gunicorn's max execution time as outlined below would be the best solution.

However, if long execution periods are abnormal in the context of your application and the data set it is processing, increasing timeout windows may not be the answer. Instead, you may benefit from profiling and optimizing the application in question.

If workers timeout during the window of expected execution time, you can increase Gunicorn's timeout in your configuration file, or with the `--timeout [SECONDS]` argument at the command line.

You should note that solving the issue in this way could potentially cause another issue: NGINX may return a 504 Gateway Timeout error because NGINX's timeout threshold, follow these steps to increase the timeout window for NGINX:

1. Open your NGINX config (`/etc/nginx/nginx.conf`)
2. Add `fastcgi_read_timeout XXX;` within the `http` block, where XXX is the timeout window in seconds (see below for an example)
3. Save and close the file
4. Reload your NGINX config (see below for system-specific commands)

```
http { ... fastcgi_buffers 8 16k;
fastcgi_buffer_size 32k; fastcgi_connect_timeout
300; fastcgi_send_timeout 300; fastcgi_read_timeout
300; }
```

init system	command
SysV	`service nginx reload`

init.d	`/etc/init.d/nginx reload`
Upstart	`initctl reload nginx`

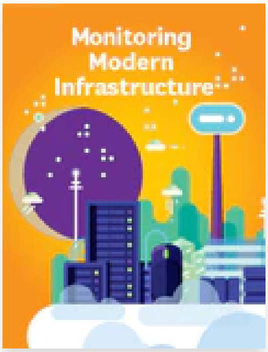
200 OK

Whether the problem was in your application, Gunicorn, or NGINX doesn't matter to your end-users. So while you've solved the immediate issue, monitoring the performance and health of your infrastructure and applications is an ongoing concern.

With Datadog, you can collect, visualize, alert, and act on performance and health metrics from across your entire infrastructure. And with 15 months of retention at full granularity, you can compare past performance with recent trends, forecast traffic and load, and fire alerts when things deviate from expectations.

If you're already a Datadog customer, you can start monitoring NGINX, Gunicorn and the rest of your

Want to write articles like this one? Our team is hiring!



Related Posts

DATADOG

Monitoring Apache web server performance | Datadog

Top ELB health and performance metrics | Datadog

Start monitoring your metrics in minutes

FIND OUT HOW

DATADOG