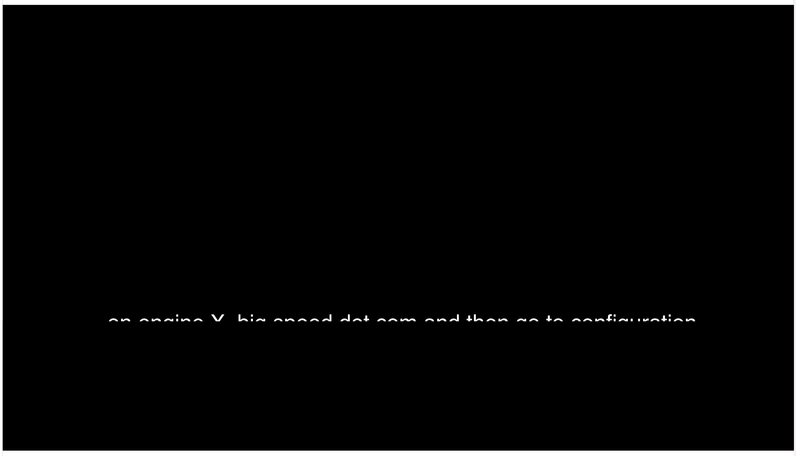
Lecture: Use Case: PageSpeed by Google



Google is the technology company with the best reputation when it comes to speed and performance. In this use case, we'll go over how to install the `ngx_pagespeed` module that Google created to make it easy to use the server side best practices that they've discovered over the years.

Note: the commands in this video are run as the `root` user.

Documentation For This Video

- Pagespeed by Google
- `ngx_pagespeed` documentation

Compiling `ngx_pagespeed` Dynamic Module

Similar to how we needed to build a separate module for ModSecurity, we'll need to compile the `ngx_pagespeed` module. Unlike ModSecurity, there is an install script accessible to us that we can configure quite easily:

```
[root] $ cd /opt
[root] $ bash <(curl -f -L -sS https://ngxpagespeed.com/install) -b . --dynamic-module --ngx-pagespeed-version latest-stable
...
ngx_pagespeed is ready to be built against nginx.
When running ./configure:
  Give ./configure the following arguments:
    --add-dynamic-module=./incubator-pagespeed-ngx-latest-stable

If this is for integration with an already-built nginx, make sure
to include any other arguments you originally passed to
./configure.  You can see these with 'nginx -V'.
```

With the module itself prepared, we need to cd into the NGINX source that we use to build our dynamic modules so that we can run `configure` and `make modules`:

```
[root] $ cd nginx-1.12.2
[root] $ ./configure --with-compat --add-dynamic-module=../incubator-pagespeed-ngx-latest-stable
[root] $ make modules
[root] $ cp objs/ngx_pagespeed.so /etc/nginx/modules/
```

Now we're ready to utilize the PageSpeed module.

Bandwidth and Render Speed Without PageSpeed

To test out the rendering of the content, we should measure the speed and page size of our example application, which in this case is going to be our WordPress install for `blog.example.com`.

Here's what I got from the Firefox developer tools "Network" tab after disabling NGINX caching and refreshing the page a few times (not the most scientific of observations):

> 11 requests 252.78 KB / 604.81 KB transferred Finish: 533 ms

This gives us a baseline to compare against after PageSpeed is enabled.

Using the PageSpeed Module

To enable PageSpeed, we need to include the new module in our `nginx.conf` file and then utilize it in our `server` block:

/etc/nginx/nginx.conf (partial)

```
# Load ModSecurity dynamic module
load_module /etc/nginx/modules/ngx_http_modsecurity_module.so;
Load PageSpeed dynamic module
load_module /etc/nginx/modules/ngx_pagespeed.so;
```

We're going to utilize the base configuration to enable the module as recommended by the configuration docs:

/etc/nginx/conf.d/blog.example.com (partial)

```
fastcgi_cache_path /var/cache/nginx/blog levels=1:2
                   keys_zone=blog:10m max_size=1g inactive=60m;
server {
    listen 80;
    server_name blog.example.com;
    root /var/www/blog.example.com;
    index index.php;
    modsecurity on;
    modsecurity_rules_file /etc/nginx/modsecurity/modsecurity_includes.conf;
Before - 11 requests 437.29 KB / 268.15 KB transferred Finish: 584 ms

    pagespeed on;

Needs to exist and be writable by nginx.  Use tmpfs for best performance.

    pagespeed FileCachePath /var/cache/nginx/ngx_pagespeed_cache;

Ensure requests for pagespeed optimized resources go to the pagespeed handler

and no extraneous headers get set.

    location ~ ".pagespeed.([a-z].)?[a-z]{2}.[^.]{10}.[^.]+" {
      add_header "" "";
    }
    location ~ "^/pagespeed_static/" { }
    location ~ "^/ngx_pagespeed_beacon$" { }
```

```
    access_log /var/log/nginx/access.log vhost;

    fastcgi_cache_key $scheme$request_method$host$request_uri;

    # Configuration continued ...</code></pre>
```

Before reloading our configuration, we need to make sure that we've created the directory for the FileCachePath:

```
[root] $ mkdir -p /var/cache/nginx/ngx_pagespeed_cache
[root] $ chown nginx:nginx /var/cache/nginx/ngx_pagespeed_cache
[root] $ systemctl reload nginx
```

Here are the results that I saw after enabling PageSpeed:

> 11 requests 417.38 KB / 253.10 KB transferred Finish: 518 ms

Compared to what we were seeing before, there's not a *huge* difference, and that's because WordPress if fairly optimized. That being said even on a fairly optimized site we saw the total size of the page go down as well as the time to render. I encourage you to look into all of the filters that you can specifically tell PageSpeed to use in the documentation.