2020

NGINX Caching E-learning Lab Guide 4

STUDENT LAB GUIDE

NGINX, INC. | 795 Folsom Street, San Francisco, CA. 94107

Table of Contents

Getting Started

This lab guide provides step-by-step instructions for lab exercises. Each lab corresponds to a module covered in class and provides you with hands-on experience working with the NGINX Plus as a caching server.

Course Pre-requisites

This course is intended to provide training on NGINX Plus caching configurations to IT professionals who have completed the NGINX Core course. It is assumed students have familiarity with:

- IT operations
- Web servers
- Linux
- Text editor: Vim, Vi, Emacs, etc.
- Networking topologies

Log into Hosted Environment

- Open your email and find the lab systems assigned to you. Your lab systems have NGINX Plus pre-installed.

- There is a login for the machine with a username ***student*** and the password ***student***.

Lab 4: Fine Tuning the Cache

Learning Objectives

By the end of the lab you will be able to:

- Configure and test a cache purge
- Find your cache files

Exercise 1: Finding your Cached Files

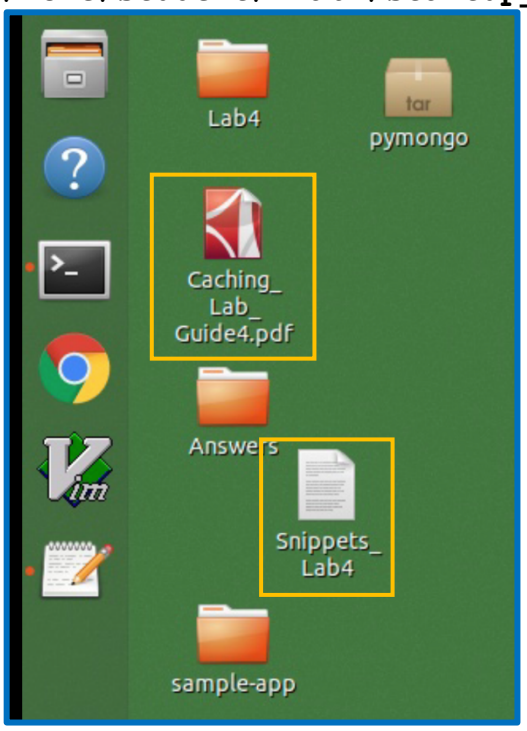
Overview

In this exercise we will explore how to locate our cached files

Steps

1. Open a terminal window on your lab system, NGINX_Cache1 and then open the ***Snippets_Lab4*** file so it will be easy to copy and paste commands.

   Note: Only if you **do not** have these files **Snippets_Lab4** file and **Caching_Lab_Guide4.pdf** on your desktop, then run:
   `/home/student/.Lab4/startup_Lab4`



2. Run the following curl command twice to ensure that we get a cached copy of the response stored.

   $ **curl -I localhost**

```
NGINX1$ curl -I localhost
HTTP/1.1 200 OK
Server: nginx/1.19.5
Date: Tue, 27 Apr 2021 23:01:49 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 2358
Connection: keep-alive
X-Cache-Status: MISS

NGINX1$ curl -I localhost
HTTP/1.1 200 OK
Server: nginx/1.19.5
Date: Tue, 27 Apr 2021 23:01:56 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 2358
Connection: keep-alive
X-Cache-Status: HIT
```

3. Change directory into **/data/nginx/cache**, which is our configured folder for the cache.

   $ **cd /data/nginx/cache**

4. List the files. You should see something similar to the following.

   $ **ls**

```
NGINX1$ cd /data/nginx/cache
NGINX1$ ls
5
NGINX1$ _
```

5. Now let's figure out where our cached entry for our "localhost" request resides. First, we need to get the MD5 hash. We know that we are using the `$scheme$host$request_uri` as our hash key. The scheme of the request is "http". The host is "localhost" and the request URI is "/".

   Run the echo command to look at the value.

   $ **echo -n httplocalhost/ | md5sum**
   `f7fbc9561a3975ce1ecff55583e50465  -`

In our example, the hash value is f7fbc9561a3975ce1ecff55583e50465.

Note: If you do not get the same hash value as our example here, make sure you have passed the md5sum command the exact string “httplocalhost/” including the slash at the end.

6. Our cache is configured to use a 2 level folder structure. The first level folder is the last character of the hash value. In this case “5”.

   Look inside the “5” folder.

   ```
   $ sudo ls 5
   46
   ```

   Note: Make sure to use the last number in your hash not in the example here.

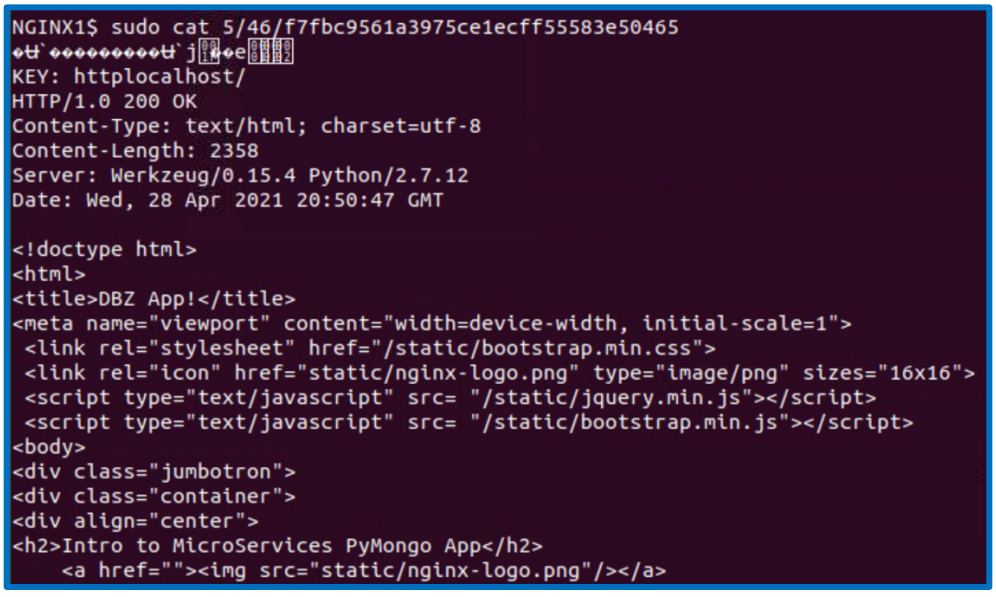
7. Notice how 5 is the last character in the hash and the next level folder of “46” is the previous 2 characters before that. Now look inside the “46” folder.

   ```
   $ sudo ls 5/46
   f7fbc9561a3975ce1ecff55583e50465
   ```

   Note: If you do not have a file under the 46 directory, then run the curl command twice: `curl -I localhost`

8. Open the contents of the file you found in step 6. You should see the contents of the response.

   ```
   $ sudo cat 5/46/f7fbc9561a3975ce1ecff55583e50465
   ```



Exercise 2: Cache Purging

Overview

In this exercise, you set up a map to configure and test the proxy purge.

Steps

1. Open the default.conf configuration file and create the following map in the http context, adding this map after your other map directive:

```
map $request_method $purge_method {
     default  0;
     PURGE    1;
}
```

2. Add the `proxy_cache_purge` directive to the `location /` block, adding as the first directive in the location / block:

```
location / {
     proxy_cache_purge $purge_method;
     ...
}
```

Note: The changed parts of your file should be similar to this:

```
map $request_method $purge_method {
        default  0;
        PURGE    1;
}

server {
    listen 80 default_server;
    root /usr/share/nginx/html;
    index index.html;
    status_zone proxy;

    access_log /var/log/nginx/py-mongo.access.log json_log;

    add_header X-Cache-Status $cache_status;

#    proxy_cache_min_uses 5;

    proxy_cache_key $scheme$host$request_uri;
    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_ignore_headers Cache-Control Expires;
    proxy_cache_revalidate on;
    proxy_cache_use_stale error;

    location / {
        proxy_cache_purge $purge_method;
        proxy_cache py-mongo;
        proxy_cache_valid 200 30s;
        proxy_pass http://py-mongo;
    }
```

3. Save and reload NGINX.

   `$ sudo nginx -s reload`

4. Send a purge command using curl:

   `$ curl -I -X PURGE http://localhost/`

   The successful PURGE results in an HTTP 204 code:

```
NGINX1$ curl -I -X PURGE http://localhost/
HTTP/1.1 204 No Content
Server: nginx/1.19.5
Date: Wed, 28 Apr 2021 20:59:29 GMT
Connection: keep-alive
```