Lab 7: Logging

Estimated time for completion: **15 minutes**

Requirements

The following tasks must be completed before beginning this lab:

- Getting Started with NGINX, (the Getting Started Guide in LearnF5)
- Log into Hosted Environment, your lab initialization instructions are located in the LearnF5 course

Scenario

In this exercise, add an error log and an access log configuration in the server block.

Objectives

At the end of this lab you will be able to:

- Set up an error and access log
- Create and test a custom log

Lab Contents

Exercise 1: Add logging directives to your configuration file and test them.

Exercise 2: Customize the access log with additional variables.

IMPORTANT
You can copy and paste the commands and text from the examples to your terminal or editor, (just make sure you don't copy and paste the $ prompt!)

Exercise 1: Add logging directives to your configuration file and test them.

1. Rename your **`default.conf`** file to **`default.bak`**:

```
$ cd /etc/nginx/conf.d
$ sudo mv default.{conf,bak}
```

2. Rename your **`mywebserver.bak`** file to **`mywebserver.conf`**.

```
$ sudo mv mywebserver.{bak,conf}
```

3. Add specific filenames for the NGINX error and log files.

   a. Edit your **`mywebserver.conf`** file

```
$ sudo vim mywebserver.conf
```

   b. In the **server** context add an **error log directive** with a level of **`info`** and add an **access log directive** with a type/name of **`combined`**.

   Your file should read as follows (update it to include the **`two highlighted lines`**):

```
server {
    listen 80;
    root /home/ubuntu/public_html;

    error_log /var/log/nginx/server1.error.log info;
    access_log /var/log/nginx/server1.access.log combined;

    location /application1 {
        index app1.html;
    }

    location /application2 {
        index app2.html;
    }

    location /images {
        root /data;
    }
}
```

IMPORTANT

The **error_log** and **access_log** directives can be placed anywhere in the **server** context, but an organized approach would be after the **listen** directive and before any **location** blocks.

4. Save the file and reload NGINX. (**esc** and **:wq**).

```
$ sudo nginx -s reload
```

5. Look at the access log using the following command:

```
$ tail -f /var/log/nginx/mywebserver.access.log
```

6. Open a browser and test that NGINX is using your custom access log file:

```
$ http://localhost/application1
```

IMPORTANT

If you do not have an entry for **application1** in the output from the **tail** command, then refresh your browser.

```
NGINX$ tail -f /var/log/nginxserver1.access.log
67.164.62.131 - - [10/Oct/2022:15:12:35 +0000] "GET
/application1/HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Macintosh; Intel
Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/94.0.4606.61 Safari/537.36"
```

7. To exit the **tail** command, press **ctrl-c**.

Exercise 2: Customize the access log with additional variables.

1. Edit your **`mywebserver.conf`** file:

   ```
   $ sudo vim mywebserver.conf
   ```

2. Add the following **`log_format`** command in the **http context** (i.e. at the top of the file above the **server** context).

```
# This is the http context

log_format test_log '"Request: $request\n Status: $status\n
Request_URI: $request_uri\n Host: $host\n Client_IP: $remote_addr\n
Proxy_IP(s): $proxy_add_x_forwarded_for\n Proxy_Hostname:
$proxy_host\n Real_IP: $http_x_real_ip"';

server {
    error_log /var/log/nginx/server1.error.log info;
    access_log /var/log/nginx/server1.access.log test_log;

    listen 80;
    root /home/ubuntu/public_html;
    . . .
```

3. In the **access_log** directive change **combined** to match your custom log format name **test_log**:

```
# This is the http context

log_format test_log '"Request: $request\n Status: $status\n
Request_URI: $request_uri\n Host: $host\n Client_IP: $remote_addr\n
Proxy_IP(s): $proxy_add_x_forwarded_for\n Proxy_Hostname:
$proxy_host\n Real_IP: $http_x_real_ip"';

server {
    listen 80;
    root /home/ubuntu/public_html;

    error_log /var/log/nginx/server1.error.log info;
    access_log /var/log/nginx/server1.access.log test_log;

    location /appllication1 {
    index app1.html;
    }

    location /application2 {
    index app2.html;
    }

    location /images {
    root /data;
    }
}
```

4. Save the file and reload NGINX. (**esc** and **:wq**).

```
$ sudo nginx -s reload
```

5. Look at the access log using the following command:

```
$ tail -f /var/log/nginx/mywebserver.access.log
```

You should see that the logfile is now providing one variable per line with a descriptive indicator before each.

6. Send another request to your lab system:

   http://localhost/application1

   The log file now provides one variable per line with a descriptive indicator before each.

```
NGINX$ tail -f /var/log/nginxserver1.access.log
67.164.62.131 - - [10/Oct/2022:15:12:35 +0000] "GET /application1/
HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X
10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/94.0.4606.61
Safari/537.36"
52.67.69.51 - - [10/Oct/2022:15:12:35 +0000] "POST / HTTP/1.1/" 405
157 "-" "SAMSUNG-SGH-E250/1.0 Profile/MIDP-2.0 Configuration/CLDC-
1.1 UP.Browser/6.2.3.3.c.1.101 (GUI) MMP/2.0"
52.67.69.51 - - [10/Oct/2022:15:12:35 +0000] "GET /.env HTTP/1.1/"
404 153 "-" "SAMSUNG-SGH-E250/1.0 Profile/MIDP-2.0
Configuration/CLDC-1.1 UP.Browser/6.2.3.3.c.1.101 (GUI) MMP/2.0"

"Request: GET /application1/ HTTP/1.1
Status: 304
Request_URI: /application1/
Host: ec2-54-176-255-210.us-west1.compute.amazonaws.com
Client_IP: 67.164.62.131
Proxy_IP(s): 67.164.62.131
Proxy_Hostname: -
Real_IP: -"
```

7. Enter **Control c** to exit the tail command.

Expected Results

In these exercises, you were able to add an error log and an access log configuration in the server block & change the format of your access log by adding a log format directive and referencing its custom name/type.