Lab 10: Mapping Variables

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, you will ...

Objectives

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

- Use the map directive with a regular expression to determine a location
- Use the map directive with an `if` statement to set up conditional logging

Lab Contents

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: Using the map Directive

Use a map to determine location

1. Edit the **server1.conf** file.

```
$ sudo vim /etc/nginx/conf.d/server1.conf
```

a. Create the following map in the http context:

```
map $uri $is_redirect {
 default 0;
   /test1 1;
   /test2 2;
   /test3 3;
}
```

IMPORTANT
Place it at the top of the file, above the **log_format** configuration and below the comment line **# This is the http context.**

```
# This is the http context

map $uri $is_redirect {
default 0;
/test1 1;
/test2 2;
   /test3 3;
}

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

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

b. In the server context, add a regular expression location for **`/test`**, as follows:

```
locatino ~* /test(/d+)$ {return 200 "variable = $is_redirect /n";
}
```

IMPORTANT
The **d+** means **`/test`** can be followed by any digit such as **`/test1`**, **`/test2`** etc. Also since the regular expression is set to case insensitive (**`~*`**) you can use **`/TEST1`**, **`/TEST2`** etc.

```
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;

# rewrite ^/shop/greatproducts/(\d+)$/shop/product/product$1.html;
  rewrite ^/media/pics/(.+\.(gif|jpe?g|png))$ /pictures/$1;

  location ~* /test(test/d+)$ {
      return 200 "variable = $is_redirect \n";
  }

  location ~ ^/pictures/(.+\.(gif|jpe?g|png))$ {
      alias /data/images/$1;
  }

    location /shop {
           rewrite ^/shop/greatproducts/(\d+)$ /shop/product$1.html
break;
           rewrite ^/shop/.+/(\d+)$ /shop/services/service$1.html
break;
    }
```

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

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

3. Test the map as follows:

   ```
   $ curl http://localhost/test1
   ```

```
NGINX$ curl http://localhost/test1
variable = 1
```

4. Try testing other values:

   ```
   $ curl http://localhost/test1
   ```

   ```
   $ curl http://localhost/test2
   $ curl http://localhost/test3
   $ curl http://localhost/TEST3
   $ curl http://localhost/test4
   $ curl http://localhost/test4241234
   ```

   Why did you get the results you got?

```
NGINX$ curl http://localhost/test1
variable = 1
NGINX$ curl http://localhost/test2
variable = 2
NGINX$ curl http://localhost/test3
variable = 3
NGINX$ curl http://localhost/TEST3
variable = 3
NGINX$ curl http://localhost/test4
variable = 0
NGINX$ curl http://localhost/test4241234
variable = 0
```

Exercise 2: Map Directive with Conditional Logging

Use the map directive with an `if` statement to set up conditional logging.

Scenario

You add a map with the `$loggable` variable to log access when the returned status code starts with a number other than 2 or 3.

1. Edit the `server1.conf` file.

   ```
   $ sudo vim /etc/nginx/conf.d/server1.conf
   ```

   a. In the **http** context (at the top of the file), create a map for excluding response codes that begin with a 2 or a 3, such as 200 and 301:

```
map $status $loggable {
   ~^[23]   0;
   default 1;
}
```

```
# This is the http context

map $status $loggable {
      ~^[23]   0;
      default 1;
}

map $uri $is_redirect {
default 0;
 /test1 1;
 /test2 2;
 /test3 3;
}
```

   b. In the `access_log` directive, add the `if` parameter with the new custom variable `$loggable` as follows:

```
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
if=loggable;
```

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

   `$ sudo nginx -s reload`

3. Test using valid an invalid requests.

   a. Make a valid request for:

   `$ curl http://localhost`

   b. Make an invalid request (for example a path/file that does not exist) such as:

   `$ curl http://localhost/nowhere`

```
NGINX$ curl http://localhost/nowhere
<html>
<head><title>404 Not Found</title></head>
<body>
<center><h1>404 Not Found</h1></center>
</body>
</html>
```

   c. View the access log

```
$ sudo cat /var/log/nginx/server1.accesslog | grep -C 5
404
```

IMPORTANT
The `-C` option for grep shows context around any found search terms, so you will see each of the **404** errors in the log. Below is the last (most recent) 404 error, you will have more than one 404 error from previous lab steps.

```
Client_IP: 45.61.184.37
Proxy_IP(s): 45.61.184.37
Proxy_Hostname: -
Real_IP: -"
"Request: GET /nowhere HTTP/1.1
Status: 404
Request_URI: /nowhere
Host: localhost
Clien_IP: 127.0.0.1
Proxy_IP(s): 127.0.0.1
Proxy_Hostname: -
```

Expected Results

In these exercises, you were able to determine what NGINX processes are running on Linux and view the primary NGINX configuration file.