NGINX Core - Module M12 Labs

**** The Labs Begin on Page 2 ****

PLEASE READ THESE SYSTEM USE INSTRUCTIONS:

The lab systems default to text mode, if you want to take advantage of copy-and-paste functionality, or need to use multiple **Terminal** applications, you'll need to start the GUI Desktop environment using the following instructions.

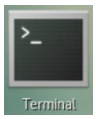
Steps to Prepare for a GUI Lab Environment:

1. Sign on to the system with the user credential **root** and password **training**.

2. Start the GUI desktop with the command:

   ```
   $ startx
   ```

3. When the Graphical Desktop appears, click on the **Terminal** app icon on the left-side of the desktop to open the Terminal.



   Note: You can resize the **Terminal** app to suit your preferences or to show more text output on the screen.

To Copy and Paste to a Lab System in GUI Mode:

1. Select the desired text from this lab guide and use **Cmd+c** (macOS) or **Ctrl-c** (Windows and Linux) to copy the text to the system clipboard.

2. Then click into the **Terminal** app in the lab system, and press Ctrl-Shift-V or select the Terminal application's **Edit** -> **Paste** menu item.

3. You may need to press it twice to insert the copied text.

   Note: When pasting into **Vim**, you MUST be in **Insert** mode first, or the keystrokes will get mis-interpreted. If you get a strange `^M` or other character first, just delete it and re-paste again.

NOTE: The **commands** you should be typing, (or copying & pasting for convenience) are displayed like the below text; (don't type or copy the **$** prompt, just the `green` text):

```
$ command
```

The **output** from commands, or the file contents shown in commands like **vim** are displayed like this:

```
Line of output
Another line of output
```

Labs for Module 12 - Load Balancing

Labs Included

- Lab 12.1 - Create the **`lb_test.conf`** File
- Lab 12.2 - Setup Backend Servers for Load Balancing
- Lab 12.3 - Set protocols and ciphers
- Lab 12.4 - Configure the **`add_header`** directive
- Lab 12.5 - Configure the Diffie-Hellman key exchange
- Lab 12.6 - Force HTTP traffic to HTTPS

PLEASE READ!

You have been allocated **1 hour** for the labs in this section. Please leave the Virtual Lab Machine browser window/tab open after completing the first lab, then continue the section content. You will then be asked to complete the rest of the labs throughout the rest of the section.

If you let too much time lapse and the Virtual Lab Machine times out, simply restart it by clicking on the Launch button for the Lab and start again with the first lab.

Note: We have included configuration file samples in their entirety to make catching up with a particular lab easier. For example, if your lab times out and you were in a later lab, simply copy the configuration file contents into the appropriate file and continue with the next lab.

Lab 12.1: Create the limit_test.conf Configuration File

Learning Objectives

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

- Create a new configuration file and add a service block.
- Test for results

Overview

In this exercise, you will create a new configuration file in preparation for testing restricting access and the view the results of your requests.

Steps

1. Rename the **`default.conf`** configuration file:

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

2. Create a new configuration file named **`lb_test.conf`**:

```
$ vim /etc/nginx/conf.d/lb_test.conf
```

3. Create a server block in the **lb_test.conf** file:

   Note: When in Vim, press **i** to enter insert mode, and unless you are typing the server block manually, refer to the copy-and-paste instructions on the first page of the lab.

   New configuration is shown in **bold** text.

```
server {
  listen 80 default_server;
  root /usr/share/nginx/html;
  server_name localhost;
  access_log /var/log/nginx/lb.access.log combined;
  error_log /var/log/nginx/lb.error.log notice;
    location / {
    }
}
```

   Note: Closely inspect the file contents if you used the copy-and-paste function. You should indent the lines as shown in the example, additionally there may be other formatting issues to fix.

4. When done adding the content to the file, quit the editor by first pressing the **ESC** key and then typing:

```
:wq
```

5. Reload your NGINX configuration with

```
$ nginx -s reload
```

6. Test your NGINX configuration with:

```
$ curl localhost | grep -i "welcome"
```

   Inspect the output for the phrase **Welcome to nginx!**, which should at the bottom of the command's output.

```
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   612  100   612    0     0   111k      0 --:--:-- --:--:-- --:--:--
119k
<title>Welcome to nginx!</title>
<h1>Welcome to nginx!</h1>
```

End of Lab

Lab 12.2: Setup Backend Servers for Load Balancing

Learning Objectives

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

- Create a backend configuration file
- Setup 3 server context with unique ports
- Create sample backend data location files and directories
- Test for results

Overview

In this exercise, you will xxx

Steps

1. Create a new configuration file named **`backends.conf`**:

   ```
   $ vim /etc/nginx/conf.d/backends.conf
   ```

1. Define three server context with root directive pointing to /data/backend 1, backend2 and backend 3, with listen directives on ports 8081, 8082, and 8083, respectively:

   Note: When in Vim, press **`i`** to enter insert mode, and unless you are typing the server block manually, refer to the copy-and-paste instructions on the first page of the lab.

   New configuration is shown in **bold** text.

```
server {
  listen 8081;
  root /data/backend1;
  }

server {
  listen 8082;
  root /data/backend2;
  }

server {
  listen 8083;
  root /data/backend3;
}
```

Note: Closely inspect the file contents if you used the copy-and-paste function. You should indent the lines as shown in the example, additionally there may be other formatting issues to fix.

2. When done adding the content to the file, quit the editor by first pressing the **ESC** key and then typing:

```
:wq
```

3. Reload your NGINX configuration with

```
$ nginx -s reload
```

4. Create 3 backend data directories with:

```
$ mkdir -p /data/backend{1,2,3} ; ls -1 /data
```

The output should reflect the three directories have been created.

5. Create 3 index files in the new directories:

```
$ echo "backend1" | tee -a /data/backend1/index.html

$ echo "backend2" | tee -a /data/backend2/index.html

$ echo "backend3" | tee -a /data/backend3/index.html
```

The output should reflect the values being put into the index.html files.

6. Test the your configuration results with:

```
$ curl http://localhost:808{1,2,3}
```

```
backend1
backend2
backend3
```

The output should be a listing of the content of each of the backend index.html files assigned to each port, **8081**, **8082** and **8083**.

End of Lab

Lab 12.3: Allow Requests from a Specific Server

Learning Objectives

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

- xxx
- Test for results

Overview

In this exercise, you will xxx

Steps

1. Edit the **`lb_test.conf`** file:

   ```
   $ vim /etc/nginx/conf.d/lb_test.conf
   ```

2. Add an **`allow`** directive **ABOVE** the **`deny`** directive to allow a host to access the system.

   Note: When in Vim, press **`i`** to enter insert mode, and unless you are typing the server block manually, refer to the copy-and-paste instructions on the first page of the lab.

   New/updated configuration is shown in **bold** text.

   ```
   upstream myServers {
     server 127.0.0.1:8081;
     server 127.0.0.1:8082;
     server 127.0.0.1:8083;
   }

   server {
     listen 80 default_server;
     root /usr/share/nginx/html;
     server_name localhost;
     access_log /var/log/nginx/lb.access.log combined;
     error_log /var/log/nginx/lb.error.log notice;
       location / {
         proxy_pass http://myServers;
       }
   }
   ```

   Note: Closely inspect the file contents if you used the copy-and-paste function. You should indent the lines as shown in the example, additionally there may be other formatting issues to fix.

3. When done adding the content to the file, quit the editor by first pressing the **ESC** key and then typing:

```
:wq
```

4. Reload your NGINX configuration with

   ```
   $ nginx -s reload
   ```

5. Test the backend server load-balancing by running multiple requests via the **watch** and **curl** commands:

   ```
   $ watch -n 1 curl -s http://localhost
   ```

   ```
   Every 1.0s: curl -s http://localhost            Fri Apr 22 15:58:23 2022

   backend1
   ```

   The **watch** command is running the **curl** command every 1 second, and you'll see the display indicate the backend server instance that is serving the request change each time.

   Note: To exit the **watch** command interface, press the **Ctrl-c** key combination.

End of Lab

Lab 12.4: Server Weight (oh, about 15 lbs?)

Learning Objectives

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

- Configure server weights for your upstream server instances
- Test for results

Overview

In this exercise, you will configure your upstream server instances to use varying server weighting that will affect how often a particular instance is served, and test the results.

Steps

1. Edit the **lb_test.conf** file:

   ```
   $ vim /etc/nginx/conf.d/lb_test.conf
   ```

2. In the upstream myServers block, add weight parameters to the server directives as shown below:

   Note: When in Vim, press **`i`** to enter insert mode, and unless you are typing the server block manually, refer to the copy-and-paste instructions on the first page of the lab.

   New configuration is shown in **bold** text.

```
upstream myServers {
  server 127.0.0.1:8081 weight=1;
  server 127.0.0.1:8082 weight=2;
  server 127.0.0.1:8083 weight=4;
}

server {
  listen 80 default_server;
  root /usr/share/nginx/html;
  server_name localhost;
  access_log /var/log/nginx/lb.access.log combined;
  error_log /var/log/nginx/lb.error.log notice;
 location / {
      proxy_pass http://myServers;
    }
}
```

   Note: Closely inspect the file contents if you used the copy-and-paste function. You should indent the lines as shown in the example, additionally there may be other formatting issues to fix.

3. When done adding the content to the file, quit the editor by first pressing the **ESC** key and then typing:

   `:wq`

6. Test the weighted backend server load-balancing by running multiple requests via the **`watch`** and **`curl`** commands:

   `$ watch -n 1 curl -s http://localhost`

```
Every 1.0s: curl -s http://localhost            Fri Apr 22 15:58:23 2022

backend1
```

   You should see the display indicate the backend server instance that is serving the request change each time, with the new weights causing backend3 to appear more often than backend2 and both of them to appear more often than backend1.

   Note: To exit the **`watch`** command interface, press the **Ctrl-c** key combination.

End of Lab

Lab 12.5: max_fails and fail_timeout

Learning Objectives

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

- Configure fail-related directives for the upstream server instances
- Test for results

Overview

In this exercise, you will configure fail-related directives and parameters for the upstream server instances and then test the configuration results.

Steps

4. Edit the **`lb_test.conf`** file:

   ```
   $ vim /etc/nginx/conf.d/lb_test.conf
   ```

5. In the upstream myServers block, add weight parameters to the server directives as shown below:

   Note: When in Vim, press **`i`** to enter insert mode, and unless you are typing the server block manually, refer to the copy-and-paste instructions on the first page of the lab.

   New configuration is shown in **bold** text.

```
upstream myServers {
  server 127.0.0.1:8081 weight=1 max_fails=5 fail_timeout=90s;
  server 127.0.0.1:8082 weight=2 max_fails=5 fail_timeout=90s;
  server 127.0.0.1:8083 weight=4 max_fails=5 fail_timeout=90s;
}

server {
  listen 80 default_server;
  root /usr/share/nginx/html;
  server_name localhost;
  access_log /var/log/nginx/lb.access.log combined;
  error_log /var/log/nginx/lb.error.log notice;
 location / {
      proxy_pass http://myServers;
```

```
    }
}
```

Note: Closely inspect the file contents if you used the copy-and-paste function. You should indent the lines as shown in the example, additionally there may be other formatting issues to fix.

6. When done adding the content to the file, quit the editor by first pressing the **ESC** key and then typing:

   ```
   :wq
   ```

7. Test the weighted backend server load-balancing by running multiple requests via the `watch` and `curl` commands:

   ```
   $ watch -n 1 curl -s http://localhost
   ```

   ```
   Every 1.0s: curl -s http://localhost            Fri Apr 22 15:58:23 2022

   backend1
   ```

   There will be no apparent difference in the output in this lab than in the last lab, just ensure everything is working.

   Note: To exit the `watch` command interface, press the **Ctrl-c** key combination.

End of Lab

Lab 12.6: Specify a Maximum Number of Connections to Your Backends

Learning Objectives

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

- Configure a maximum number of connections for the upstream server instances
- Test for results

Overview

In this exercise, you will configure the maximum number of connection directive and parameters for the upstream server instances and then test the configuration results.

Steps

7. Edit the **`lb_test.conf`** file:

   ```
   $ vim /etc/nginx/conf.d/lb_test.conf
   ```

8. In the upstream myServers block, add weight parameters to the server directives as shown below:

   Note: When in Vim, press **`i`** to enter insert mode, and unless you are typing the server block manually, refer to the copy-and-paste instructions on the first page of the lab.

   New configuration is shown in **bold** text.

```
upstream myServers {
  server 127.0.0.1:8081 weight=1 max_fails=5 fail_timeout=90s  max_conns=100;
  server 127.0.0.1:8082 weight=2 max_fails=5 fail_timeout=90s  max_conns=100;
  server 127.0.0.1:8083 weight=4 max_fails=5 fail_timeout=90s  max_conns=100;
}

server {
  listen 80 default_server;
  root /usr/share/nginx/html;
  server_name localhost;
  access_log /var/log/nginx/lb.access.log combined;
  error_log /var/log/nginx/lb.error.log notice;
 location / {
      proxy_pass http://myServers;
    }
}
```

   Note: Closely inspect the file contents if you used the copy-and-paste function. You should indent the lines as shown in the example, additionally there may be other formatting issues to fix.

9. When done adding the content to the file, quit the editor by first pressing the **ESC** key and then typing:

   ```
   :wq
   ```

8. Test the weighted backend server load-balancing by running multiple requests via the **`watch`** and **`curl`** commands:

   ```
   $ watch -n 1 curl -s http://localhost
   ```

```
Every 1.0s: curl -s http://localhost            Fri Apr 22 15:58:23 2022

backend1
```

   There will be no apparent difference in the output in this lab than in the last lab, just ensure everything is working.

Note: To exit the **watch** command interface, press the **Ctrl-c** key combination.

End of Lab

Lab 12.7: Setup a queue for your upstream

Learning Objectives

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

- Configure the random load-balancing directive for your upstream server instances
- Test for results

Overview

In this exercise, you will configure the load-balancing directive and parameter for the upstream server instances and then test the configuration results.

Steps

10. Edit the **lb_test.conf** file:

```
$ vim /etc/nginx/conf.d/lb_test.conf
```

11. In the upstream myServers block, add weight parameters to the server directives as shown below:

Note: When in Vim, press **i** to enter insert mode, and unless you are typing the server block manually, refer to the copy-and-paste instructions on the first page of the lab.

New configuration is shown in **bold** text.

```
upstream myServers {
  server 127.0.0.1:8081 weight=1 max_fails=5 fail_timeout=90s  max_conns=100;
  server 127.0.0.1:8082 weight=2 max_fails=5 fail_timeout=90s  max_conns=100;
  server 127.0.0.1:8083 weight=4 max_fails=5 fail_timeout=90s  max_conns=100;
  random two;
}

server {
  listen 80 default_server;
  root /usr/share/nginx/html;
  server_name localhost;
  access_log /var/log/nginx/lb.access.log combined;
  error_log /var/log/nginx/lb.error.log notice;
 location / {
```

```
        proxy_pass http://myServers;
    }
}
```

Note: Closely inspect the file contents if you used the copy-and-paste function. You should indent the lines as shown in the example, additionally there may be other formatting issues to fix.

12. When done adding the content to the file, quit the editor by first pressing the **ESC** key and then typing:

```
:wq
```

9. Test the weighted backend server load-balancing by running multiple requests via the **watch** and **curl** commands:

```
$ watch -n 1 curl -s http://localhost
```

```
Every 1.0s: curl -s http://localhost            Fri Apr 22 15:58:23 2022

backend1
```

You will see that the backend instances are more random now, showing that your configuration is successful.

Note: To exit the **watch** command interface, press the **Ctrl-c** key combination.

End of Lab