

# Lab 12: Monitoring and Managing Systems

Estimated time for completion: **20 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 these exercises, you will be able to configure NGINX Plus to view performance metrics on the dashboard and use the NGINX Plus API to configure back end servers.

## Objectives

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

- Configure NGINX Plus to view performance metrics on the dashboard
- Use the NGINX Plus API to configure back end servers

## Lab Contents

1. **Exercise 1: Define and test a dashboard page**
2. **Exercise 2: Specify and test a server zone for each server**
3. **Exercise 3: Enable the dynamic API and execute API commands**
4. **Exercise 4: Enable and test back end persistent state changes**



### 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: Define and test a dashboard page

### Defining a Dashboard

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

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

2. Edit the `myservers.conf` file:

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

- a. Copy and paste the below **highlighted** text into your new file:

```

upstream myservers {

# hash $scheme$host$request_uri;

zone http_backend 64k; server 127.0.0.1:8081;
    server 127.0.0.1:8082;
    server 127.0.0.1:8083;
}

server {
    listen 8080;
    root /usr/share/nginx/html;

    location /api {
api;
    }

    location /dashboard.html {
    }
}

server {
    listen 80;
    root /usr/share/nginx/html;

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

    location / {
        proxy_pass http://myservers;
    }
}

```

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

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

4. Test the Dashboard in a browser:

`http://localhost:8080/dashboard.html`

The screenshot shows the NGINX Plus dashboard. At the top, there's a header with the NGINX Plus logo and tabs for 'HTTP Upstreams' (active), 'Shared Zones', and a settings gear. The main content area is divided into several sections:

- nginx-plus-r24-p1 (1.19.10)**: Server details including Address (172.31.29.148), PID (1103), and Uptime (1m).
- Connections**: A table showing connection statistics. Accepted: 85.
- Requests**: A table showing request statistics. Total: 574.
- HTTP Upstreams**: A section with a green checkmark icon, showing Total (1) and Problems (0).
- Servers**: A section showing All: 3 / Up: 3 and Failed: 0.

5. Click the **HTTP Upstreams** tab to view the server information:

The screenshot shows the 'HTTP Upstreams' tab in the NGINX Plus dashboard. It features a 'Show upstreams list' button and a 'Failed only' toggle. Below this is a section for 'myServers' with a 'Zone: 40%' indicator and a 'Show all' button. The main part of the page is a table with the following columns: Server, Requests, Responses, Conns, Traffic, Server checks, Health monitors, and Response time. The table contains three rows of data for servers 127.0.0.1:8081, 127.0.0.1:8082, and 127.0.0.1:8083.

Server	Requests	Responses	Conns	Traffic	Server checks	Health monitors	Response time														
Name	DT	W	Total	Req/s	...	4xx	5xx	A	L	Sent/s	Rcvd/s	Sent	Rcvd	Fails	Unavail	Checks	Fails	Unhealthy	Last	Headers	Response
127.0.0.1:8081	0ms	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	-	-
127.0.0.1:8082	0ms	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	-	-
127.0.0.1:8083	0ms	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	-	-

6. Click the **Shared Zones** tab to view the name of your shared memory zone and memory usage:

The screenshot shows the 'Shared Zones' tab in the NGINX Plus dashboard. It features a table with the following columns: Zone, Total memory pages, Used memory pages, and Memory usage. The table contains one row of data for the 'http\_backend' zone.

Zone	Total memory pages	Used memory pages	Memory usage
http_backend	15	6	40 %

## Exercise 2: Specify and test a server zone for each server

### Learning Objectives:

- Set up and test a dashboard page for individual or grouped server metrics

### Scenario

In this exercise, you configure a shared memory zone for each server to allow view access of your server metrics.

1. Create a new file named `backends.conf` in the `conf.d` directory.

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



#### NOTE

In this file you'll define backend servers that will be referenced by your configuration, and assign `status_zone` directives to group them into geographic zones.

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

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

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

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

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

- Test the Dashboard page again, looking for the total number of HTTP Zones, which should be 2:

`http://localhost:8080/dashboard.html`

The screenshot displays the NGINX Plus dashboard interface. At the top, there's a header with the NGINX Plus logo and navigation tabs for 'HTTP Zones', 'HTTP Upstreams', and 'Shared Zones'. The main content area is divided into several sections:

- nginx-plus-r24-p1 (1.19.10)**: A box containing system information: Address **172.31.29.148**, PID **1103**, and Uptime **39.17s**.
- Connections**: A table showing connection statistics. The 'Accepted' column is expanded to show 'Accepted: 92'. The table has columns for Current, Accepted/s, Active, Idle, and Dropped. Values are: Current: 7, Accepted/s: 0, Active: 1, Idle: 6, Dropped: 0.
- Requests**: A table showing request statistics. The 'Total' column is expanded to show 'Total: 2252'. The table has columns for Current and Req/s. Values are: Current: 1, Req/s: 12.
- HTTP Zones**: A box with a green checkmark icon. It shows 'Total: 2' and 'Problems: 0' (indicated by a green '0' in a box). Below this is a 'Traffic' section with 'In: 0' and 'Out: 0'.
- HTTP Upstreams**: A box with a green checkmark icon. It shows 'Total: 1' and 'Problems: 0' (indicated by a green '0' in a box). Below this is a 'Servers' section with 'All: 3 / Up: 3' and 'Failed: 0'.

## Exercise 3: Enable the dynamic API and execute API commands

### Learning Objectives:

- Enable access to the API in order to write configuration changes
- Execute API commands to configure backend servers

### Scenario

In this exercise, you will enable the `api` location with `write=on` access and use `curl` commands to remove a server from the backend server pool. You will also add the server back in with a different weight, as well as viewing these changes in your dashboard.

1. Enable the `api` location to allow write access. In the `myservers.conf` file, add the `write=on` parameter to the `api` directive. This enables you to send commands to your configuration, via the NGINX Plus API (**PUT**, **POST**, **PATCH** and **DELETE**).

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



#### NOTE

Make the addition of the `write=on` parameter to your `/api` location as indicated by the highlighted text below.

```
location /api {  
    api write=on;  
}
```

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

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

3. Click the **HTTP Upstreams** tab of your NGINX Plus Dashboard page. You should have 3 servers. Arrange your windows so you can watch this page as you send the following API command using `curl`:



## NOTE

The below command is all on one line, you may have to use the **Left** and **Right** arrows and the **Del** and **Backspace** keys to properly format it for execution.

```
$ curl -X DELETE  
"http://localhost:8080/api/3/http/upstreams/myservers/servers/2" -H "accept:application/json" | jq
```

This deletes your third server (since it starts numbering at 0) and pipes the output/result to your terminal in jQuery so it is easy to read. The dashboard changes to show only the first two servers.

NGINX Plus

HTTP Zones HTTP Upstreams

HTTP Upstreams Show upstreams list

myServers Zone: 47 %

Server			Requests		Responses			Conns		Traffic				Server checks		Health monitors		
Name	DT	W	Total	Req/s	...	4xx	5xx	A	L	Sent/s	Rcvd/s	Sent	Rcvd	Fails	Unavail	Checks	Fails	Unhealthy
127.0.0.1:8081	0ms	1	0	0		0	0	0	∞	0	0	0	0	0	0	0	0	0
127.0.0.1:8082	0ms	1	0	0		0	0	0	∞	0	0	0	0	0	0	0	0	0

4. Reload NGINX in the terminal

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

Continue to watch your dashboard page. Why does the third server re-appear?



## IMPORTANT

You deleted it via the API, but when you reloaded NGINX, what configuration was read back in?

5. Use the **Up** arrow to go back in your command history and re-execute same command you used in **Step 3** to remove the server. Either that or use the command below to accomplish the task.



#### NOTE



Because of the limitations of Copy and Paste between PDFs and the Command Line, the below command will be broken up into several lines, you may have to use the **Left** and **Right** arrows and the **Del** and **Backspace** keys to properly format it for execution.

```
$ curl -X DELETE
"http://localhost:8080/api/3/http/upstreams/myservers/serv
ers/2" -H "accept: application/json" | jq
```

6. Now add the server back with a weight of 5:

#### NOTE



Because of the limitations of Copy and Paste between PDFs and the Command Line, the below command will be broken up into several lines, you may have to use the **Left** and **Right** arrows and the **Del** and **Backspace** keys to properly format it for execution.




```
$ curl -X POST
"http://localhost:8080/api/3/http/upstreams/myServers/serv
ers/" -H "accept: application/json" -H "Content-Type:
application/json" -d "{ \"server\":
\"127.0.0.1:8083\", \"weight\": \"5\"}" | jq
```

You'll see that the server has been added back with a new weight of 5:

# Upstreams

[Show upstreams list](#)**myServers**

Zone: 40 %

Server				Requests		Responses		
	Name	DT .....	W .....	Total .....	Req/s	...	4xx	5xx
	<b>127.0.0.1:8081</b>	0ms	1	0	0		0	0
	<b>127.0.0.1:8082</b>	0ms	1	0	0	▶	0	0
	<b>127.0.0.1:8083</b>	0ms	5	0	0		0	0

## Exercise 4: Enable and test back end persistent state changes

### Learning Objectives:

- Use the **state** directive to allow the API changes to be permanent

### Scenario

In this exercise, you create a directory for the state file and provide write permissions to NGINX on that file. You remove the servers from the upstream block in your configuration file and add the **state** directive. Finally, you use the API to re-create the upstream servers and test to determine that the changes persist with this configuration.

1. Check if the state directory exists and if not create a directory for the **state** file:

```
$ ls -ld /var/lib/nginx/state
```

```
$ ls -ld /var/lib/nginx/state
drwxr-xr-x 2 nginx root 4096 Dec 7 2022 /var/lib/nginx/state
```

2. Change the ownership on the **state** directory to the user **nginx**, so that NGINX has write permissions to it:

```
$ sudo chown nginx:nginx /var/lib/nginx/state
$ ls -ld /var/lib/nginx/state
```

```
NGINX$ ls -ld /var/lib/nginx/state
drwxr-xr-x 2 nginx nginx 4096 Dec 7 2022 /var/lib/nginx/state
```

3. Open the **myservers.conf** file and add the **state** directive as shown. Comment out the **server** directives in the **upstream** context:

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

```

upstream myservers {
# hash $scheme$host$request_uri;
zone http backend 64k;
state /var/lib/nginx/state/http_backend.state;
# server 127.0.0.1:8081;
# server 127.0.0.1:8082;
# server 127.0.0.1:8083;
}

```



#### NOTE

Be sure you have commented out all three of your servers, as well as adding the **state** directive line.

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

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

5. View the dashboard in your browser. You should see that the servers have been removed.

NGINX Plus

HTTP Zones HTTP Upstreams Shared Zones

HTTP Upstreams Show upstreams list

Failed only ☐

myServers Zone: 27% Show all

Server		Requests		Responses		Conns		Traffic		Server checks		Health monitors				Response time					
Name	DT	W	Total	Req/s	...	4xx	5xx	A	L	Sent/s	Rcvd/s	Sent	Rcvd	Fails	Unavail	Checks	Fails	Unhealthy	Last	Headers	Response
No servers with 'all' state found in this upstream group.																					

6. Use the NGINX Plus API to create 3 upstream servers.
  - a. Add one server using port 8081. Because of the limitations of Copy and Paste between PDFs and the Command Line, the below command will be broken up into several lines, you may have to use the **Left** and **Right** arrows and the **Del** and **Backspace** keys to properly format it for execution.

```

$ curl -X POST
"http://localhost:8080/api/3/http/upstreams/myservers/serv
ers/" -H "accept: application/json" -H "Content-Type:

```

```
application/json" -d "{ \"server\":  
\"127.0.0.1:8081\", \"weight\": \"4\"}" | jq
```

b. Add another server using port 8082. Because of the limitations of Copy and Paste between PDFs and the Command Line, the below command will be broken up into several lines, you may have to use the **Left** and **Right** arrows and the **Del** and **Backspace** keys to properly format it for execution.

#### IMPORTANT



You can use the **Up** arrow key to bring back the previous command, (where you added the 8081 server) and change JUST the 8081 to 8082 and press **Enter** to add this server. Or copy and paste the command below.

```
$ curl -X POST  
"http://localhost:8080/api/3/http/upstreams/myservers/serv  
ers/" -H "accept: application/json" -H "Content-Type:  
application/json" -d "{ \"server\":  
\"127.0.0.1:8082\", \"weight\": \"4\"}" | jq
```

c. Add another server using port 8083. Because of the limitations of Copy and Paste between PDFs and the Command Line, the below command will be broken up into several lines, you may have to use the **Left** and **Right** arrows and the **Del** and **Backspace** keys to properly format it for execution.

#### IMPORTANT




You can use the **Up** arrow key to bring back the previous command, (where you added the 8082 server) and change JUST the 8082 to 8083 and press **Enter** to add this server. Or copy and paste the command below.

```
$ curl -X POST  
"http://localhost:8080/api/3/http/upstreams/myservers/serv  
ers/" -H "accept: application/json" -H "Content-Type:  
application/json" -d "{ \"server\":  
\"127.0.0.1:8083\", \"weight\": \"4\"}" | jq
```


OR

To use the dashboard to add servers, click the edit (pencil) button (refresh the browser to see this).

 **NGINX Plus**

## HTTP Upstreams

Show upstreams list

**myServers**

Edit selected

Add server

Zone: 40 %

Server			Requests		Responses			Conns		Traffic	
<input type="checkbox"/>	Name	DT ms	W	Total	Req/s	...	4xx	5xx	A %	L %	Sent/s
<input type="checkbox"/>	127.0.0.1:8081	0ms	4	0	0		0	0	0	∞	0

Click **Add server**. Enter information to add the server:

Add server to "myServers" ✕

**Server address**

**Server route**

☐ Add as backup server

**weight**

**max\_conns**

**max\_fails**

**fail\_timeout**

**slow\_start**

**service**

**Set state**

☒ Up ☐ Down ☐ Drain

Add

Cancel

7. Be sure to add back all 3 servers whether using the API with the `curl` command or using the GUI.
8. Reload NGINX and see that the changes persist this time.

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

## Expected Results

In these exercises, you were able to configure NGINX Plus to view performance metrics on the dashboard and use the NGINX Plus API to add, remove and reconfigure back end servers.

