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:

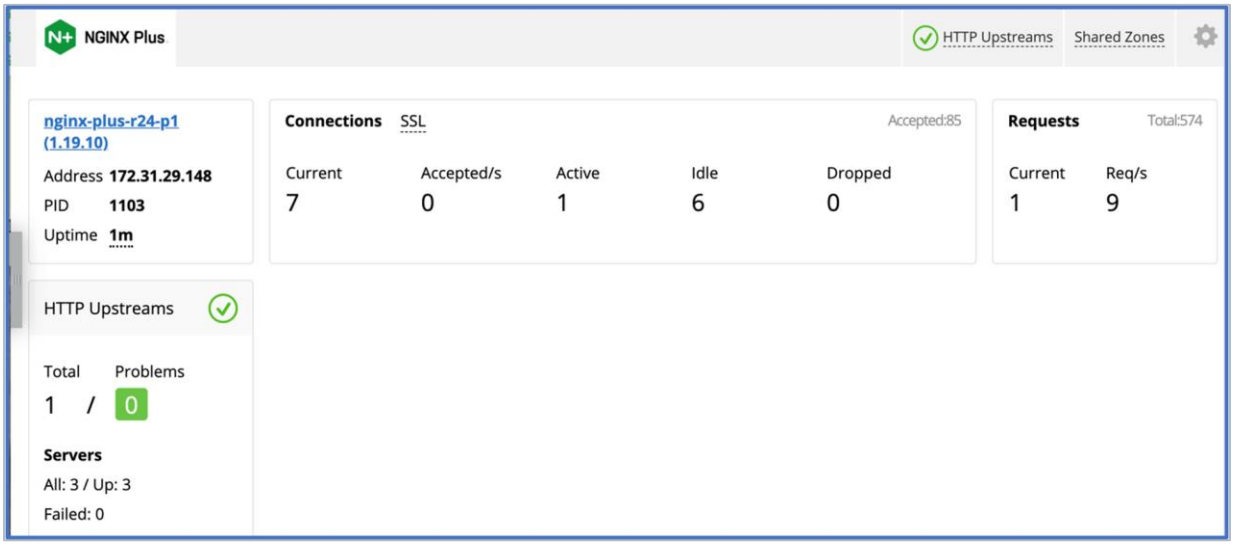
```
upstream myservers {

# 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**).

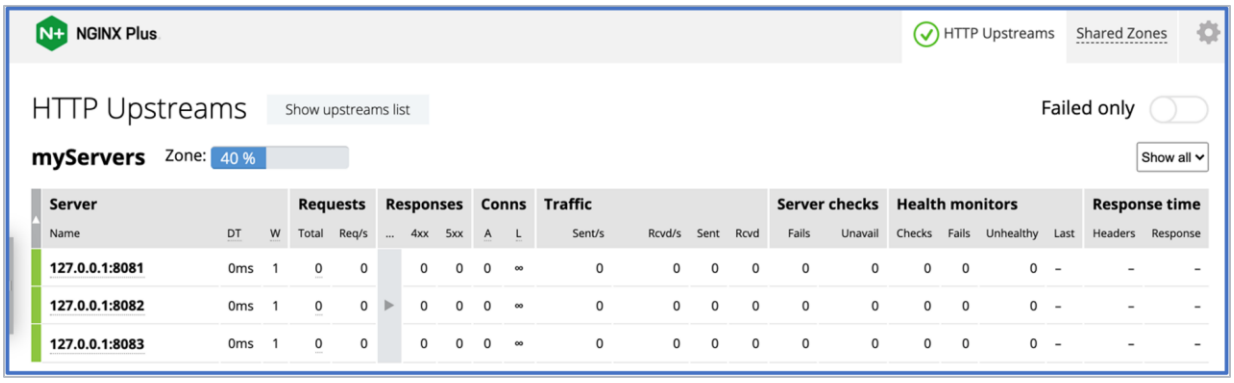
```
$ sudo nginx -s reload
```

4. Test the Dashboard in a browser:

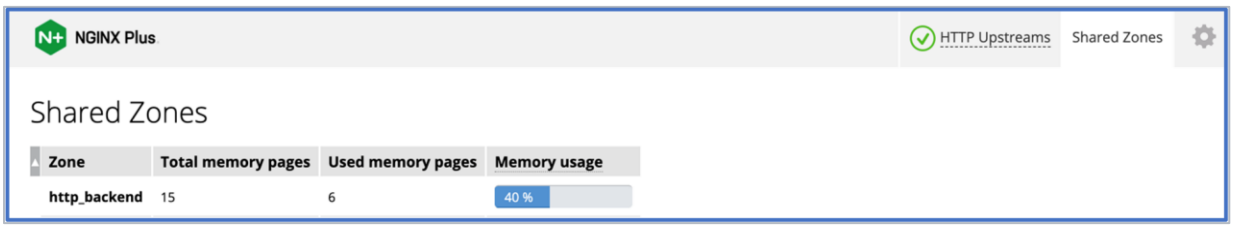
   `http://localhost:8080/dashboard.html`



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



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



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/config.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.

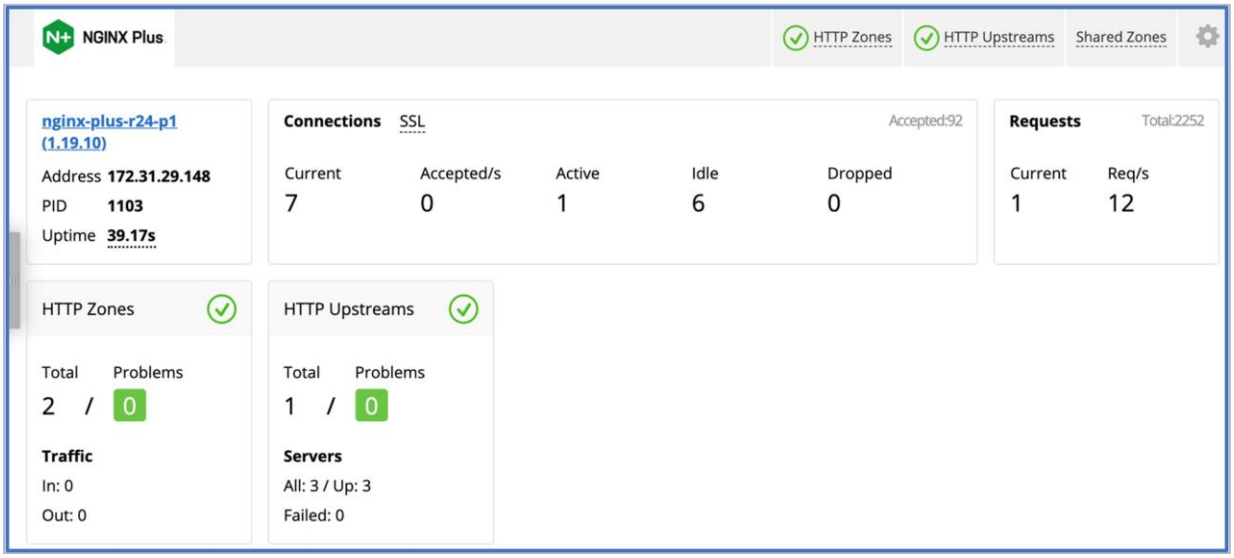
```
server {listen 8081;
    root /data/backend1;
    status_zone USA;
}

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
```

3. Test the Dashboard page again:

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



Exercise 3: Enable the dynamic API and execute API commands

Learning Objectives:

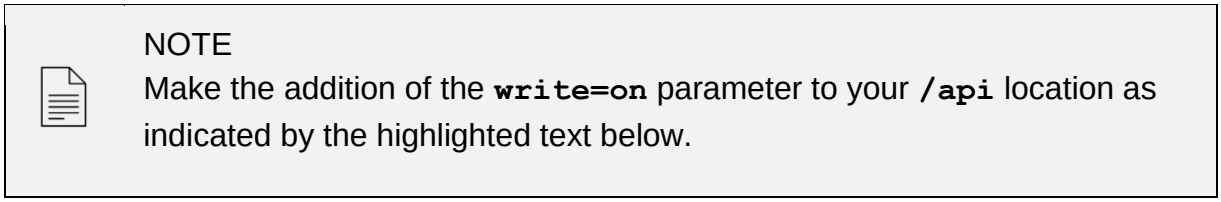
- Enable access to the API in order to write configuration changes
- Execute API commands to configure backend servers

Scenario

In this exercise, you enable the **`api`** location with write access and use **`curl`** commands to remove a server from the backend server pool. You add the server back in with a different weight, and you view 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
   ```




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

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

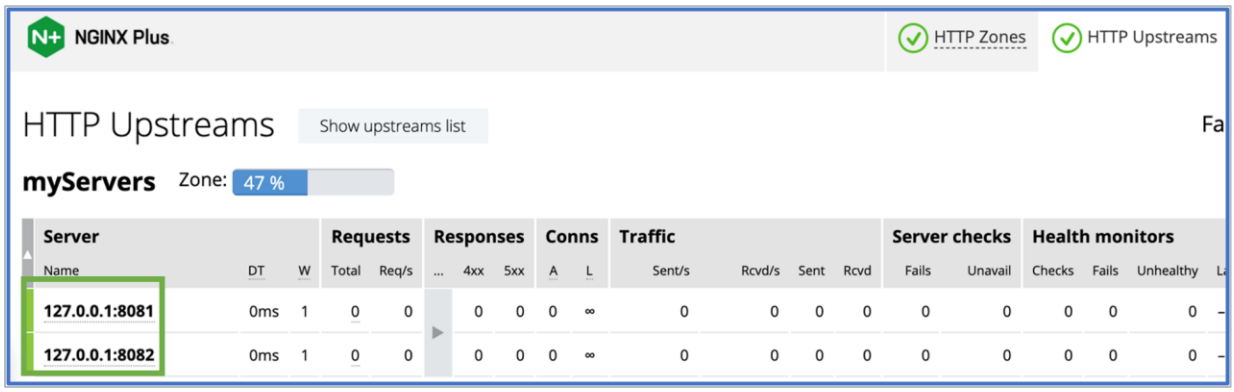
   ```
   $ sudo nginx -s reload
   ```

3. Click the HTTP Upstreams tab of your NGINX dashboard page. You have 3 servers. 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.



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

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.

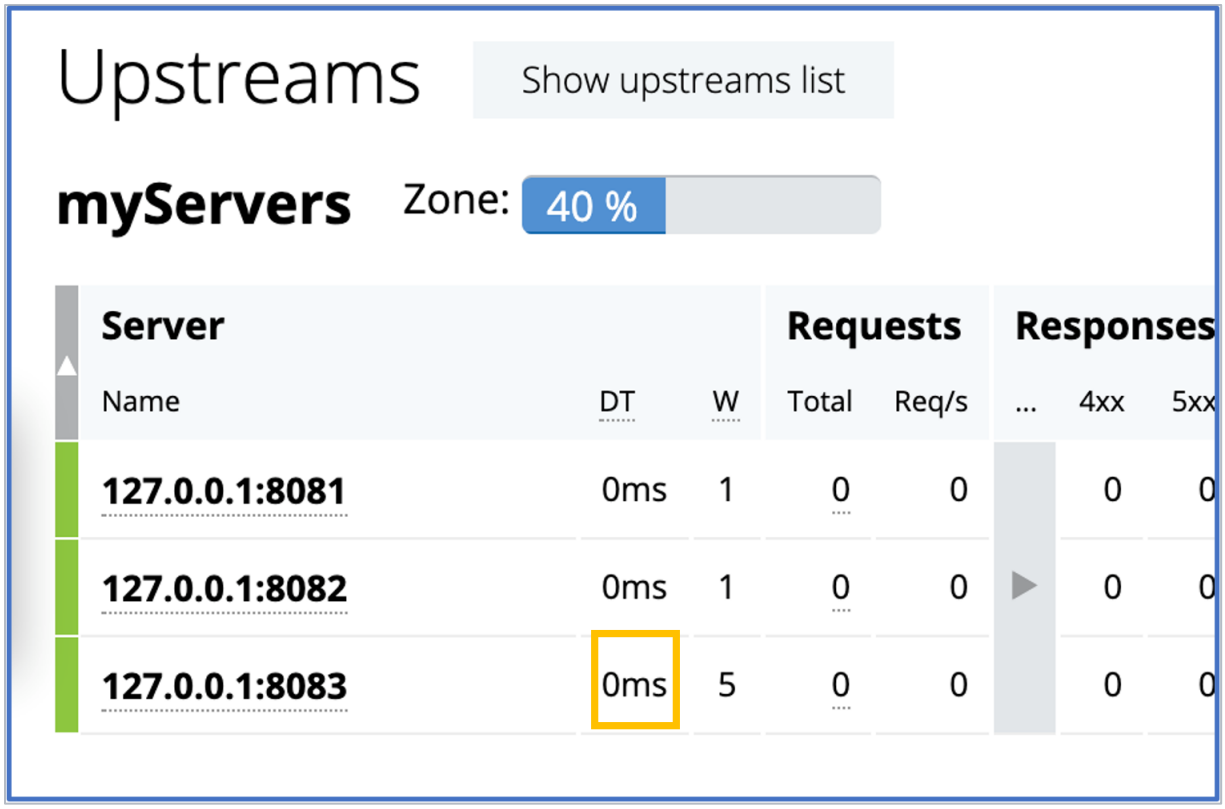
```
$ curl -X DELETE
"http://localhost:8080/api/3/http/upstreams/myservers/serv
ers/2" -H "accept: application/json" | jq
```

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

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 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 jas 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
127.0.0.1:8081
0ms
1
0
0
0
0
127.0.0.1:8082
0ms
1
0
0
0
0
127.0.0.1:8083
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 existis 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.

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

   $ sudo nginx -s reload

5. View the dashboard in your browser. The servers have been removed.

6. Use the API to create 3 upstream servers.

   a. Add one server using port **8081**. 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 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**. 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.

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.

```
$ 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
```

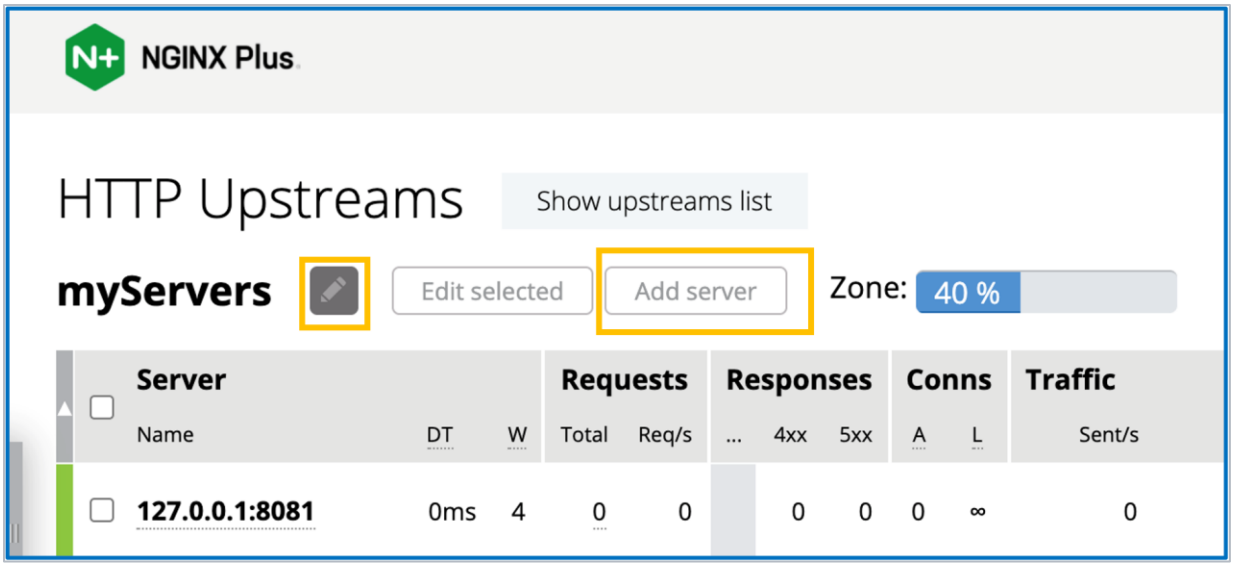
c. Add another server using port **8083**. The below command is all on one line, you may have to use the **Left** and **Right** arrows and the **Del** and **Backspace** keys to properly format it for execution.

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.

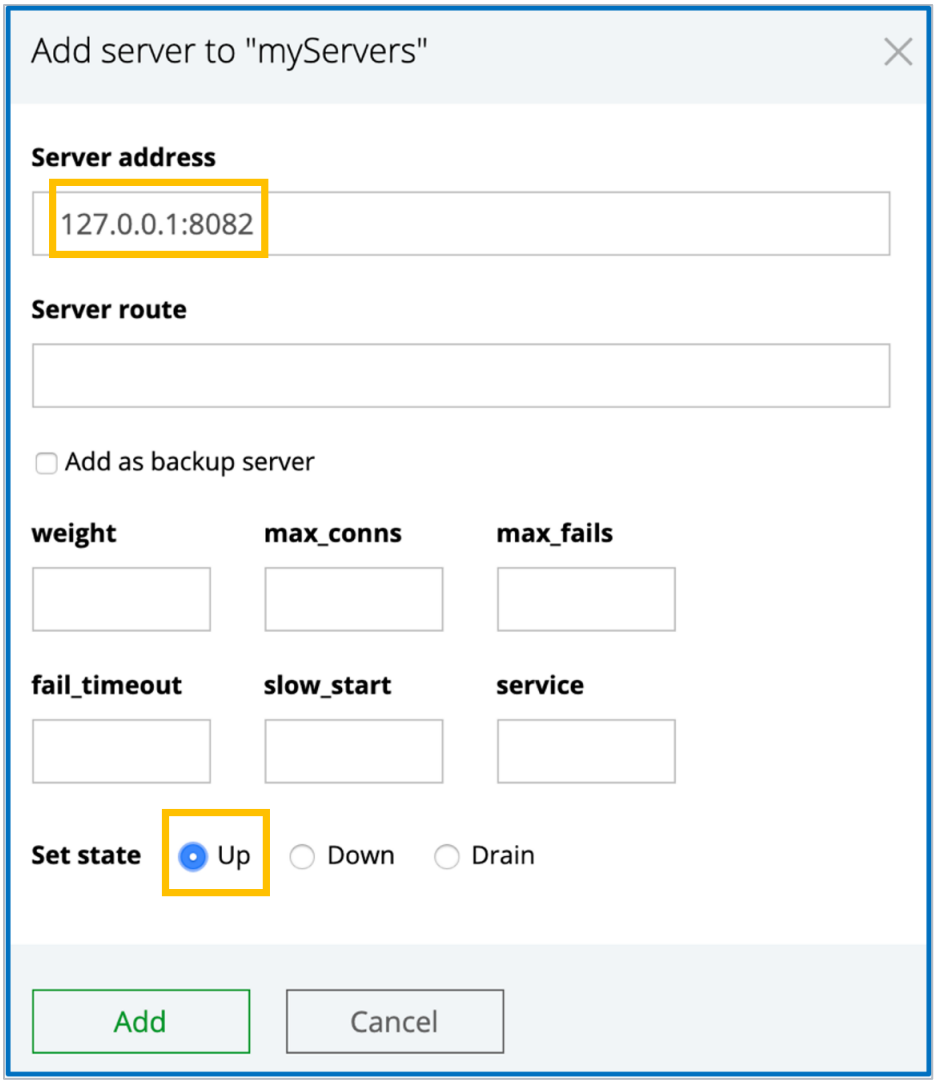
```
$ curl -X POST
"http://localhost:8080/api/3/http/upstreams/myservers/serv
ers/" -H "accept: application/json" -H "Content-Type:
application/json" -d "{ \"server\":
\"127.0.0.1:8083\",\"weight\": \"4\"}" | jq
```

OR

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



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



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 configure back end servers.