Lesson 3 Labs

Lab Contents

Lab 1: Configure Rate Limits

Lab 2: Logs

- **Exercise 1: Access and Error Log Files**
- **Exercise 2: Logging Rate Limits**

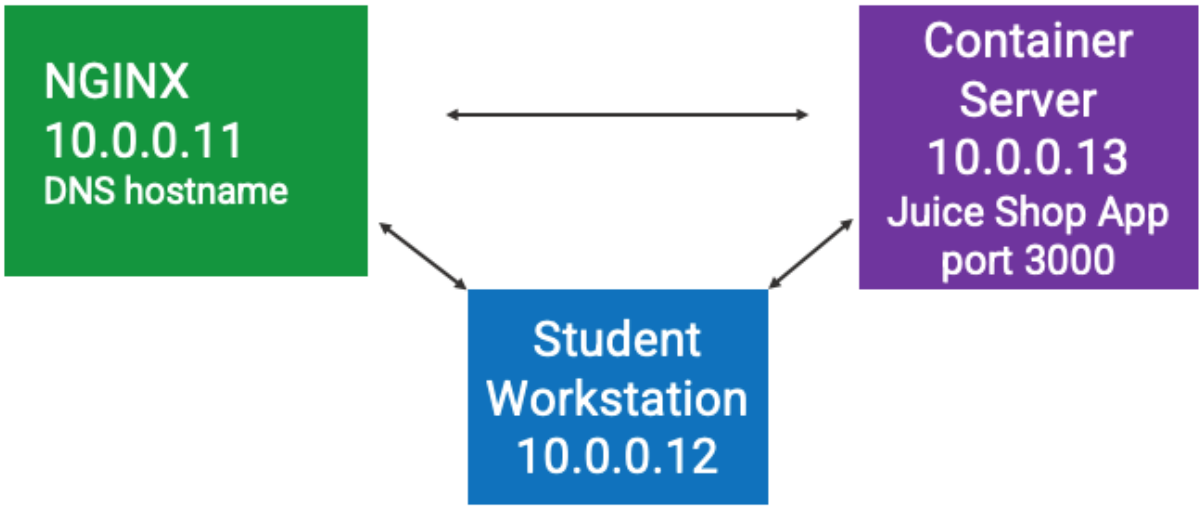
Lab 3: Setup Dynamic IP Deny-list

Lab 4: Dynamic Lazy Certificates

Lab 5: Dynamic In-Memory Certificate

Estimated time to complete labs: **[XX] minutes**

Lab Network Topology



Lab 1: Configure Limit Rates

Estimated time for completion: **15 minutes**

Requirements

The following tasks must be completed before beginning this lab:

- NGINX lab system with access to Juice Shop application running on another Linux lab system

Objectives

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

- Use the NGINX geo and map directives.
- Configure limit request rates in NGINX.

Exercise 1

1. Create a copy of the juice.conf file before configuring limits in NGINX.

   ```
   $ sudo cp /etc/nginx/conf.d/juice.{conf,PRElimits}
   ```

2. Edit the **`juice.conf`** file.

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

3. Add a **`geo`** and **`map`** directive in the **`http`** context (at the top of the file outside of any server context). The **`map`** directive sets the **`$limit_key`** variable to the client's IP address (in byte form via the variable **`$binary_remote_addr`**).

   ```
   geo $limit {
   default 1;
   127.0.0.1 0;
   }
   map $limit $limit_key {
   0 "";
   1 $binary_remote_addr;
   }
   ```

4. In the **http** context, after the **map**, create a **limit_req_zone** using the **$limit_key** variable and set the request rate to 25 requests per second and then create a **limit_conn_zone** using the variable **$server_name** using 10 megabytes for the number of connections.

```
limit_req_zone $limit_key zone=req_zone:10m rate=25r/s;
limit_conn_zone $server_name zone=conn_zone:10m;
```

5. In the **ssl server** context, set **limit_req** with a burst of 15, **nodelay** and use **req_zone** as the **zone** name. Also set the shared memory zone and the maximum allowed number of connections to 15 with the directive **limit_conn**.

```
limit_req zone=req_zone burst=15 nodelay;
limit_conn conn_zone 15;
```

The **juice.conf** file will now look like this with changes highlighted:

```
log_format proxy_log "
Request: $request
    Status: $status
    Client: $remote_addr
    Upstream: $upstream_addr
    Forwarded-For: $proxy_add_x_forwarded_for
    ssl_server_name: $ssl_server_name
";

geo $limit {
      default 1;
      127.0.0.1 0;
}
map $limit $limit_key {
      0 “”;
      1 $binary_remote_addr;
}

limit_req_zone $limit_key zone=req_zone:10m rate=25r/s;
limit_conn_zone $server_name zone=conn_zone:10m;

upstream juice_server {
    server 10.0.0.13:3000;
    zone juiceshop 64k;
}

server {
    listen 80;
    return 301 https://$host$request_uri;
}
server {
    listen 443 ssl default_server;
    status_zone proxy;
    limit_req zone=req_zone burst=15 nodelay;
    limit_conn conn_zone 15;
    server_name www.nginxtraining.com;
    root  /usr/share/nginx/html;
    include /etc/nginx/ssl-configs/ssl-params.conf;
    access_log /var/log/nginx/access.log combined;
    access_log /var/log/nginx/juice.access.log proxy_log;
    error_log /var/log/nginx/juice.error.log info;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For
$proxy_add_x_forwarded_for;
```

6. Save the **`juice.conf`** file and then reload your NGINX configuration to make it active.

```
$ sudo nginx -t && sudo nginx -s reload
```

NOTE
We will test these limits in the next lab exercise.

Expected Results

Your NGINX configuration has limit rates set for maximum requests.

Lab 2: Log Files and Logging Limit Rates

Estimated time for completion: **35 minutes**

Requirements

The following tasks must be completed before beginning this lab:

- NGINX lab system with access to Juice Shop application running on another Linux lab system.
- Have limit request rates set in the **`juice.conf`** configuration file on the NGINX lab system.

Objectives

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

- Test limits written to the NGINX access and error log files.
- Configure the NGINX limit request log level.
- Configure the NGINX limit request status response code.

Exercise 1: Access and Error Log Files

1. In another terminal, view request information by using the **`tail -f`** command on the **access log**.

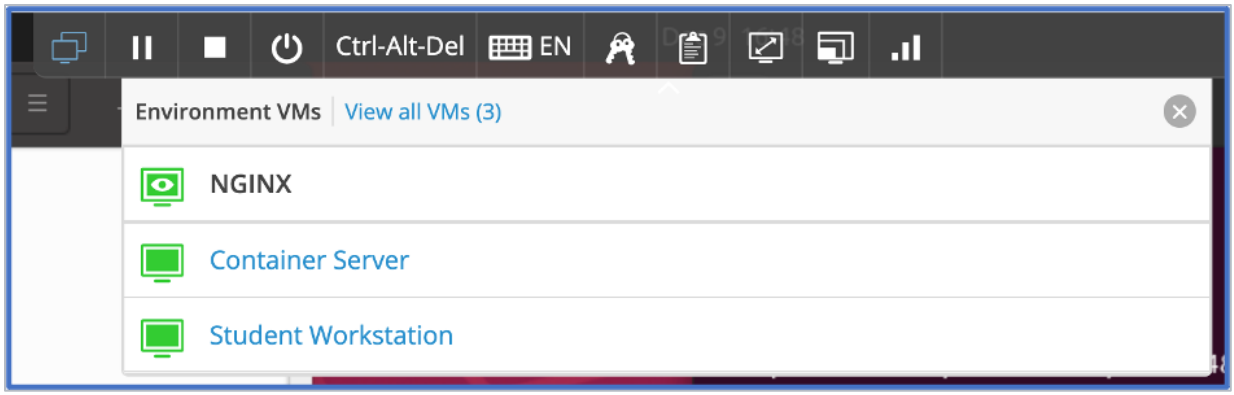
   ```
   $ sudo tail -f /var/log/nginx/juice.access.log
   ```

NOTE
When viewing log files, it is best to have a wide but not too tall terminal.

2. In another terminal, view the error information by using the **`tail -f`** command with the error log file.

   ```
   $ sudo tail -f /var/log/nginx/juice.error.log
   ```

3. At the top of the Skytap environment, click on the two monitors so you can select the **Student Workstation** lab system.



4. Log in to the **Student Workstation** lab system, using **student** for the password.

5. On the Student Workstation lab system, open a terminal and run this **`for`** loop command.

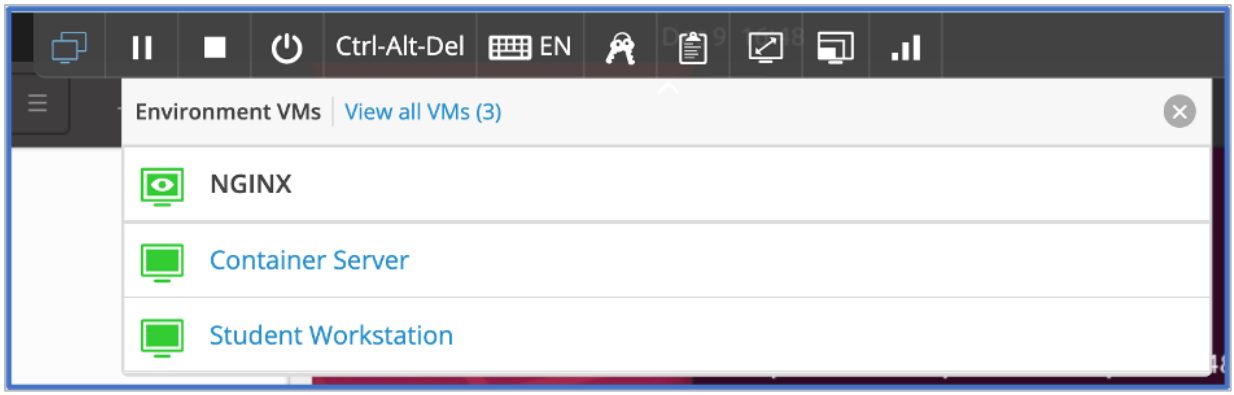
```
$ for i in {0..50}; do (curl -kIs https://www.nginxtraining.com |
head -n1 &) 2> /dev/null; done
```

6. View the output from the for loop on your **Student Workstation** lab system:

```
student@jump:~$ for i in {0..50}; do (curl -kIs
https://www.nginxtraining.com | head -n1 &) 2> /dev/null;
done
HTTP/1.1 200 OK
HTTP/1.1 200 OK
HTTP/1.1 200 OK
HTTP/1.1 200 OK
HTTP/1.1 200 OK
HTTP/1.1 200 OK
HTTP/1.1 200 OK
HTTP/1.1 200 OK
HTTP/1.1 200 OK
HTTP/1.1 200 OK
HTTP/1.1 200 OK
HTTP/1.1 200 OK
HTTP/1.1 200 OK
HTTP/1.1 200 OK
HTTP/1.1 200 OK
HTTP/1.1 503 Service Temporarily Unavailable
HTTP/1.1 200 OK
HTTP/1.1 503 Service Temporarily Unavailable
HTTP/1.1 503 Service Temporarily Unavailable
HTTP/1.1 503 Service Temporarily Unavailable
HTTP/1.1 503 Service Temporarily Unavailable
HTTP/1.1 503 Service Temporarily Unavailable
HTTP/1.1 200 OK
```

NOTE
The for loop sends more requests than what the limits are set to on your NGINX lab system (10.0.0.11) resulting in several ***Service Temporarily Unavailable*** messages.

7. At the top of the Skytap environment, click on the two monitors so you can select the **NGINX** lab system.



8. View the **access log** output from the `tail` command on the **NGINX** lab system.

```
Request: HEAD / HTTP/1.1
    Status: 200
    Client: 10.0.0.12
    Upstream: 10.0.0.13:3000
    Forwarded-For: 10.0.0.12
    ssl_server_name: www.nginxtraining.com


  Request: HEAD / HTTP/1.1
    Status: 503
    Client: 10.0.0.12
    Upstream: -
    Forwarded-For: 10.0.0.12
    ssl_server_name: www.nginxtraining.com


  Request: HEAD / HTTP/1.1
    Status: 503
    Client: 10.0.0.12
    Upstream: -
    Forwarded-For: 10.0.0.12
    ssl_server_name: www.nginxtraining.com


  Request: GET /socket.io/?EIO=4&transport=polling
    Status: 200
    Client: 10.0.0.12
    Upstream: 10.0.0.13:3000
    Forwarded-For: 10.0.0.12
    ssl_server_name: www.nginxtraining.com
```

NOTE

You should get several **503** messages once the limit has been reached. The **Status: 200** indicates the request was successful. **Client: 10.0.0.12** indicates where the request originated, which is your Student Workstation lab system, where the for loop command was run. **Upstream: 10.0.0.13:3000** indicates where NGINX forwarded the request to the Server Container lab system where the Juice Shop application is running on port 3000.

9. Use the following command to search thru the juice.error.log file on your NGINX lab system to view several rate limit warning messages about excess requests.

```
$ sudo cat -vet /var/log/nginx/juice.error.log | grep limiting
```

```
student@nginx:~$ sudo cat -vet /var/log/nginx/juice.error.log
| grep limiting
2023/06/22 08:11:36 [error] 5752#5752: *9861 limiting
requests, excess: 15.850 by zone "req_zone", client:
10.0.0.12, server: www.nginxtraining.com, request: "HEAD /
HTTP/1.1", host: "www.nginxtraining.com"$
2023/06/22 08:11:36 [error] 5754#5754: *9858 limiting
requests, excess: 15.775 by zone "req_zone", client:
10.0.0.12, server: www.nginxtraining.com, request: "HEAD /
HTTP/1.1", host: "www.nginxtraining.com"$
2023/06/22 08:11:36 [error] 5752#5752: *9863 limiting
requests, excess: 15.725 by zone "req_zone", client:
10.0.0.12, server: www.nginxtraining.com, request: "HEAD /
HTTP/1.1", host: "www.nginxtraining.com"$
2023/06/22 08:11:36 [error] 5755#5755: *9865 limiting
requests, excess: 15.475 by zone "req_zone", client:
10.0.0.12, server: www.nginxtraining.com, request: "HEAD /
HTTP/1.1", host: "www.nginxtraining.com"$
```

NOTE
You should get several **503** messages once the limit has been reached. The **Status: 200** indicates the request was successful. **Client: 10.0.0.12** indicates where the request originated, which is your Student Workstation lab system, where the for loop command was run. **Upstream: 10.0.0.13:3000** indicates where NGINX forwarded the request to the Server Container lab system where the Juice Shop application is running on port 3000.

Expected Results

After configuring rate limits in NGINX, you will have several errors and warning messages from sending more requests than allowed.

Exercise 2: Logging Limit Rates

1. On the **NGINX** lab system, edit the **`juice.conf`** file.

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

2. In the **`ssl server`** context, add the **`limit_req_log_level`** and the **`limit_req_status`**. Set the **`log_level`** to warn when the error response code limit is exceeded.

   ```
   limit_req_log_level warn;
   limit_req_status 444;
   ```

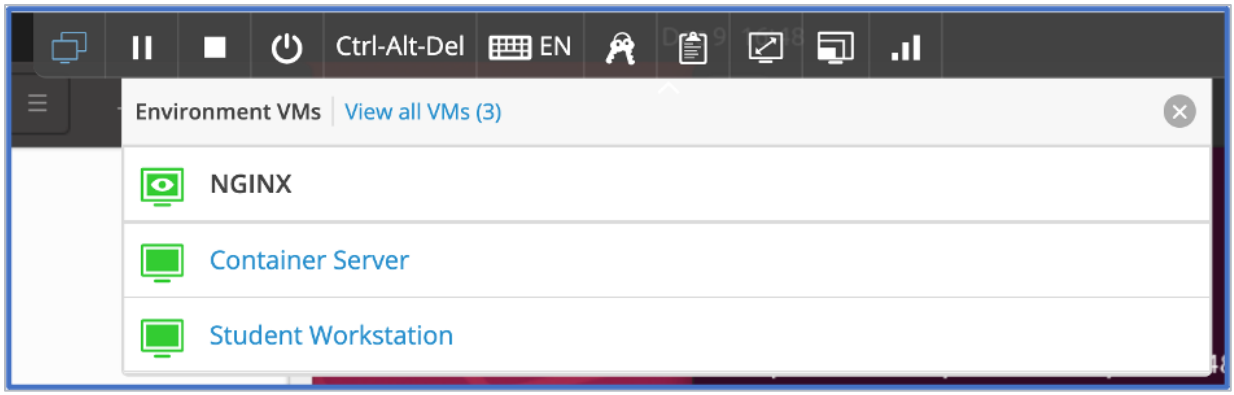
3. Save your configuration file and reload NGINX.

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

4. View the **access log** file.

   ```
   $ sudo tail -f /var/log/nginx/juice.access.log
   ```

5. At the top of the Skytap environment, click on the two monitors so you can select the **Student Workstation** lab system.



6. On the **Student Workstation** lab system, open a terminal and run the same **`for`** loop command from Exercise 1. (*Copy the command below or press the up arrow to recall the command from command history.*)

```
$ for i in {0..50}; do (curl -kIs https://www.nginxtraining.com |
head -n1 &) 2> /dev/null; done
```

7. View the output from the for loop on your Student Workstation lab system. There should only be HTTP/1.1 200 OK messages for the successful requests.

8. Switch back to your **NGINX** lab system to view the output from the NGINX **access log**. Note: You may have to scroll back to see the 444 status messages.

```
Request: HEAD / HTTP/1.1
    Status: 444
    Client: 10.0.0.12
    Upstream: -
    Forwarded-For: 10.0.0.12
    ssl_server_name: www.nginxtraining.com


  Request: HEAD / HTTP/1.1
    Status: 444
    Client: 10.0.0.12
    Upstream: -
    Forwarded-For: 10.0.0.12
    ssl_server_name: www.nginxtraining.com


  Request: HEAD / HTTP/1.1
    Status: 444
    Client: 10.0.0.12
    Upstream: -
    Forwarded-For: 10.0.0.12
    ssl_server_name: www.nginxtraining.com
```

NOTE

The output from the access log has the response code 444 which was set in our configuration file for when the error response code limit is exceeded. The 444 status closes the TCP connection to NGINX.

TROUBLESHOOTING:

If you do not get any **`Status: 444`** messages in the NGINX access log file, make sure to scroll back in the log file or use this command:

`grep 444 /var/log/nginx/juice.access.log`

9. Use **Ctrl+c** to stop the **`tail`** command.

10. For class purposes go back to your **`juice.conf`** file that does not have limits by **renaming** **`juice.conf`** to **`juice.LIMITS`** and then rename the **`juice.PRElimits`** file to **`juice.conf`**.

```
$ sudo mv /etc/nginx/conf.d/juice.{conf,LIMITS}
$ sudo mv /etc/nginx/conf.d/juice.{PRElimits,conf}
```

Expected Results

Configure and test logging limit rates by setting the limit request log level and setting the response status to 444 to terminate TCP connection.

Expected Results

Lab 3: Dynamic IP Deny-List

Estimated time for completion: **25 minutes**

Requirements

The following tasks must be completed before beginning this lab:

- NGINX lab system with access to Juice Shop application running on another Linux lab system.
- Have limit request rates set in the **`juice.conf`** configuration file on the NGINX lab system.
- Configure logging rate limits and set the response status to 444 to terminate TCP connection.

Objectives

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

- Create a Dynamic IP Deny-list
- Use the NGINX Plus **`keyval`** store

Exercise 1: Configure Dynamic IP Deny-list

1. Create a directory that the nginx user has write access to store the key value file.

   ```
   $ sudo mkdir /tmp/nginx
   $ sudo chown nginx:nginx /tmp/nginx
   ```

2. Edit the **`juice.conf`** configuration file.

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

3. In the **`http`** context add a **`keyval_zone`** with the **`state`** parameter and add a **`keyval`** data store, mapping the client's ip address **`$remote_addr`** to the $target variable.

   ```
   keyval_zone zone=one:1m state=/tmp/nginx/one.keyval;
   keyval $remote_addr $target zone=one;
   ```

4. Continuing editing the **juice.conf** file, to create a new **/api** prefix in **server 443** with **write** mode enabled on localhost only.

```
location /api {
    api write=on;
    allow 127.0.0.1;
    deny all;
}
```

5. Continuing editing the **juice.conf** file and after the **api** location add an **if** directive in the **server 443** context with **$target** as the argument and a return **status 403**.

```
if ($target) {
    return 403;
}
```

The **juice.conf** file should now look like this with updates highlighted:

```
log_format proxy_log "
Request: $request
    Status: $status
    Client: $remote_addr
    Upstream: $upstream_addr
    Forwarded-For: $proxy_add_x_forwarded_for
    ssl_server_name: $ssl_server_name
";

keyval_zone zone=one:1m state=/tmp/nginx/one.keyval;
keyval $remote_addr $target zone=one;

upstream juice_server {
    server 10.0.0.13:3000;
    zone juiceshop 64k;
}

server {
    listen 80;
    return 301 https://$host$request_uri;
}
server {
    listen 443 ssl default_server;
    status_zone proxy;
    server_name www.nginxtraining.com;
    root /usr/share/nginx/html;
    include /etc/nginx/ssl-configs/ssl-params.conf;
    access_log /var/log/nginx/access.log combined;
    access_log /var/log/nginx/juice.access.log proxy_log;
    error_log /var/log/nginx/juice.error.log info;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header Host $host;

    location / {
       proxy_pass http://juice_server;
    }

    location /api {
        api write=on;
        allow 127.0.0.1;
        deny all;
    }

    if ($target) {
        return 403;
    }
```

6. Save your configuration file and reload NGINX.

```
$ sudo nginx -t && sudo nginx -s reload
```

7. Send a **`POST`** request to populate the **`keyval`** database with your **Student Workstation** lab systems IP address **10.0.0.12**. By setting a value of **`1`**, the Student Workstation lab system will be on the deny-list. You should get a **`201 Created`** response code which means your **`curl`** command was successful.

```
$ curl -k -v -H "Content-Type: application/json" -d
'{"10.0.0.12":"1"}' 'https://127.0.0.1/api/8/http/keyvals/one'
```

NOTE
You should get a **`201 Created`** response code, which means your **`curl`** command worked.

TROUBLESHOOTING:
If you get a **`409 Conflict`** message showing the **`KeyvalKeyExists`** then you already have a value in memory (test with the next lab step) and you can continue or you could clear out the data and do this command again.

** Output truncated*

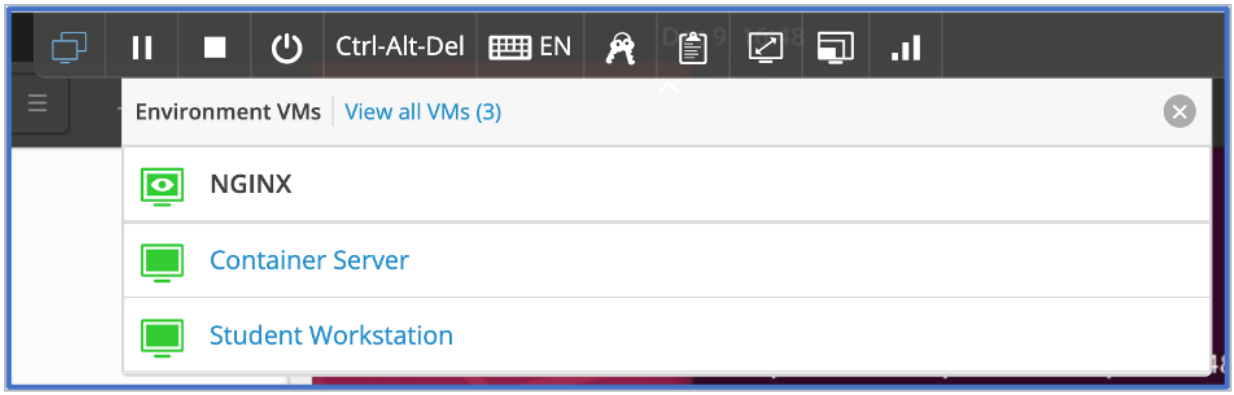
```
student@nginx:~$ curl -k -v -H "Content-Type:
application/json" -d '{"10.0.0.12":"1"}'
'https://127.0.0.1/api/8/http/keyvals/one'
* Trying 127.0.0.1:443...
* TCP_NODELAY set
* Connected to 127.0.0.1 (127.0.0.1) port 443 (#0)
...

...
* upload completely sent off: 17 out of 17 bytes
* TLSv1.3 (IN), TLS handshake, Newsession Ticket (4):
* TLSv1.3 (IN), TLS handshake, Newsession Ticket (4):
* old SSL session ID is stale, removing
* Mark bundle as not supporting multiuse
< HTTP/1.1 201 Created
< Server: nginx/1.23.4
< Date: Thu, 22 Jun 2023 20:22:25 GMT
< Content-Length: 0
< Location: https://127.0.0.1/api/8/http/keyvals/one/
< Connection: keep-alive
< Strict-Transport-Security: max-age=63072000;
includeSubdomains
< X-Frame-Options: DENY
< X-Content-Type-Options: nosniff
<
* Connection #0 to host 127.0.0.1 left intact
student@nginx:~$
```

8. Send a **GET** request to confirm it works. Your output should be: {"10.0.0.12":"1"}

   ```
   $ curl -k https://127.0.0.1/api/8/http/keyvals/one
   ```

9. At the top of the Skytap environment, click on the two monitors so you can select the **Student Workstation** lab system.



10. On the **Student Workstation** lab system, open a browser and enter your **NGINX** lab systems IP address 10.0.0.11. You should get a "**403 Forbidden**" response code since the data part of the key-value pair is set to "1" which means "true" so NGINX will deny your Student Workstation lab system since it is on the deny-list.

    https://www.nginxtraining.com

> TROUBLESHOOTING:
> You may have to click on some browser security questions to get to the **403 Forbidden** response code: clear your browser data cache or use an incognito or private browser window.

11. Switch back to the **NGINX** lab system to use the NGINX api to update the **keyval_zone** data to change the value to a **0** zero.

```
$ curl -k -i -X PATCH -d '{"10.0.0.12":”0"}' -s
'https://127.0.0.1/api/8/http/keyvals/one'
```

    You should get a **204 No Content** response.

```
student@nginx:~$ curl -k -i -X PATCH -d '{"10.0.0.12":"0"}' -
s 'https://127.0.0.1/api/8/http/keyvals/one'
HTTP/1.1 204 No Content
Server: nginx/1.23.4
Date: Thu, 22 Jun 2023 20:40:52 GMT
Connection: keep-alive
Strict-Transport-Security: max-age=63072000;
includeSubdomains
X-Frame-Options: DENY
X-Content-Type-Options: nosniff

student@nginx:~$
```

12. On the NGINX lab system, send a **GET** request to confirm it updated the value from **1** to 0.

    ```
    $ curl -k https://127.0.0.1/api/8/http/keyvals/one
    ```

    Your output should be:
    `{"10.0.0.12":"0"}`

13. Switch back to the **Student Workstation** lab system and in the browser, refresh the webpage https://10.0.0.11 to view if the result is different than previously displayed.

NOTE
Yes. You should now be able to get to the Juice Shop web page without a Forbidden error message because the **0** means **false**, i.e., don't deny-list things from this IP address.

14. Switch back to the **NGINX lab system** and use the NGINX api to update the keyval_zone data to **DELETE** the entry.

    ```
    $ curl -k -i -X DELETE -d '{"10.0.0.12":"0"}' -s
    'https://127.0.0.1/api/8/http/keyvals/one'
    ```

You should get a **204 No Content** response.

```
student@nginx:~$ curl -k -i -X DELETE -d '{"10.0.0.12":"0"}'
-s 'https://127.0.0.1/api/8/http/keyvals/one'
HTTP/1.1 204 No Content
Server: nginx/1.23.4
Date: Thu, 22 Jun 2023 20:45:55 GMT
Connection: keep-alive
Strict-Transport-Security: max-age=63072000;
includeSubdomains
X-Frame-Options: DENY
X-Content-Type-Options: nosniff

student@nginx:~$
```

15. Verify that the value has been deleted.

    ```
    $ curl -k https://127.0.0.1/api/8/http/keyvals/one
    ```

    Your output should be:
    {}

```
student@nginx:~$ curl -k
https://127.0.0.1/api/4/http/keyvals/one
{}student@nginx:~$
student@nginx:~$
```

Expected Results

You have tested using a deny-list in your NGINX configuration.

Lab 4: Use Dynamic Lazy Certificates from disk

Estimated time for completion: **15 minutes**

Requirements

The following tasks must be completed before beginning this lab:

- NGINX lab system with access to Juice Shop application running on another Linux lab system.
- Have limit request rates set in the **`juice.conf`** configuration file on the NGINX lab system.
- Configure logging rate limits and set the response status to 444 to terminate TCP connection.
- NGINX lab system with **`juice.conf`** file including an upstream to the Juice Shop application with certs and HTTPS set up.

Objectives

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

- Use NGINX Dynamic Lazy Certificate file loading from disk.

Steps

1. Before changing to use the NGINX Dynamic Certificate loading method, backup the configuration files: **`ssl-params.conf`**, **`juice.conf`**, and **`api_server.conf`**

   ```
   $ sudo cp /etc/nginx/ssl-configs/ssl-params.{conf,traditional}
   $ sudo cp /etc/nginx/conf.d/juice.{conf,traditional}
   $ sudo cp /etc/nginx/conf.d/api_server.{conf,traditional}
   ```

2. Edit the **`ssl-params.conf`** file to add new ssl certificate directives to use Dynamic Lazy loading. These directives use special variables for dynamically loading the appropriate certificate based on SNI.

   ```
   $ sudo vim /etc/nginx/ssl-configs/ssl-params.conf
   ```

```
ssl_certificate /etc/nginx/ssl-configs/certs/$ssl_server_name.crt;
ssl_certificate_key /etc/nginx/ssl-
configs/certs/$ssl_server_name.key;
```

Comment out the original 2 Traditional certificate directives method of listing each certificate file and pathname on disk. The beginning of the **ssl-params.conf** file is shown.

```
ssl_trusted_certificate /etc/nginx/ssl/ca-cert.crt;

#Traditional cert loading
#ssl_certificate /etc/nginx/ssl/www.nginxtraining.com.crt;
#ssl_certificate_key
/etc/nginx/ssl/www.nginxtraining.com.key;

#Dynamic Lazy Loading
ssl_certificate /etc/nginx/ssl-
configs/certs/$ssl_server_name.crt;
ssl_certificate_key /etc/nginx/ssl-
configs/certs/$ssl_server_name.key;

ssl_dhparam /etc/nginx/dhparam.pem;
ssl_protocols TLSv1.2 TLSv1.3;
ssl_prefer_server_ciphers on;
...
```

NOTE
The **ssl_trusted_certificate** can't be loaded dynamically.

3. Test and reload NGINX to use your updated configuration files.

```
$ sudo nginx -t && sudo nginx -s reload
```

4. Test your configuration with **curl**.

Change into the **/ssl** directory

```
$ cd ~/ssl
```

Test with **`curl`**

```
$ curl -sLIvk --cacert ca-cert.crt https://www.nginxtraining.com
```

Make sure to scroll back to the beginning output to see the connection, port information and response 200 OK.
* *Output truncated*)

```
student@nginx:~/ssl$ curl -sLIvk --cacert ca-cert.crt
https://www.nginxtraining.com
* Trying 10.0.0.11:443...
* TCP_NODELAY set
* Connected to www.nginxtraining.com (10.0.0.11) port 443
(#0)
...
...
* Server certificate:
*  subject: CN=www.nginxtraining.com; OU=Training; O=F5;
L=SanFrancisco; ST=CA; C=US
*  start date: Jun 16 17:39:08 2023 GMT
*  expire date: May 23 17:39:08 2123 GMT
*  issuer: CN=www.nginxtraining.com-ca; OU=Training; O=F5;
L=SanFrancisco; ST=CA; C=US
*  SSL certificate verify ok.
> HEAD / HTTP/1.1
> Host: www.nginxtraining.com
> User-Agent: curl/7.68.0
> Accept: */*
>
* TLSv1.3 (IN), TLS handshake, Newsession Ticket (4):
* TLSv1.3 (IN), TLS handshake, Newsession Ticket (4):
* old SSL session ID is stale, removing
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
HTTP/1.1 200 OK
< Server: nginx/1.23.4
Server: nginx/1.23.4
...

...
<
* Connection #0 to host www.nginxtraining.com left intact
student@nginx:~/ssl$
```

Expected Results

You have configured NGINX to use dynamic lazy certificate loading instead of the traditional certificate loading method.

Lab 5: Use Dynamic In-memory Certificate

Estimated time for completion: **XX minutes**

Requirements

The following tasks must be completed before beginning this lab:

- NGINX lab system with access to Juice Shop application running on another Linux lab system.
- Have limit request rates set in the **`juice.conf`** configuration file on the NGINX lab system.
- Configure logging rate limits and set the response status to 444 to terminate TCP connection.
- NGINX lab system with **`juice.conf`** file including an upstream to the Juice Shop application with certs and HTTPS set up.

Objectives

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

- Configure and test using NGINX Dynamic In-memory certificates.

Steps

1. Before changing to use the NGINX Dynamic In-memory certificate loading method, backup the following configuration files: **`ssl-params.conf`**, **`api_server.conf`**, and **`juice.conf`**.

   ```
   $ sudo cp /etc/nginx/ssl-configs/ssl-params.{conf,lazy}
   $ sudo cp /etc/nginx/conf.d/api_server.{conf,lazy}
   $ sudo cp /etc/nginx/conf.d/juice.{conf,lazy}
   ```

2. Edit the **`juice.conf`** file in the **`http`** context (top of file) add two new **`keyval_zone`** directives; one for the certificate data and one for the private key data and then add two new **`keyval`** directives mapping the key values and zones.

   ```
   keyval_zone zone=ssl_crt:1m;
   keyval_zone zone=ssl_key:1m;
   ```

```
    keyval $ssl_server_name $crt_pem zone=ssl_crt;
    keyval $ssl_server_name $key_pem zone=ssl_key;
```

```
keyval_zone zone=ssl_crt:1m;
keyval_zone zone=ssl_key:1m;
keyval $ssl_server_name $crt_pem zone=ssl_crt;
keyval $ssl_server_name $key_pem zone=ssl_key;

log_format proxy_log "
Request: $request
    Status: $status
    Client: $remote_addr
    Upstream: $upstream_addr
    Forwarded-For: $proxy_add_x_forwarded_for
    ssl_server_name: $ssl_server_name
";

keyval_zone zone=one:1m state=/tmp/nginx/one.keyval;
keyval $remote_addr $target zone=one;

upstream juice_server {
    server 10.0.0.13:3000;
    zone juiceshop 64k;
}
...
```

3. Save the **`juice.conf`** file.

4. Reload NGINX to use your updated configuration files

   ```
   $ sudo nginx -t && sudo nginx -s reload
   ```

5. Configure the certificate and key files into a format that NGINX can store in memory and that your student account has access to. View the **upload.sh** script file.

   ```
   $ cd ~/ssl
   $ cat upload.sh
   ```

```
student@nginx:~/ssl$ cat upload.sh
#! /bin/bash

HOST=$1
echo HOST: $HOST
for EXT in key crt ; do
    CERT=$(sed 's/$/\\n/' $HOST.$EXT | tr -d '\n')
    URL=https://$HOST/api/8/http/keyvals/ssl_$EXT
    echo {\"$HOST\":\""$CERT"\"} | curl -kvH "Content-Type:
application/json" -d @- $URL
done
student@nginx:~/ssl$
```

6. Run the script **upload.sh** passing it the DNS name of your NGINX system. You must run this script while in the **~/ssl** directory so it has access to the cert files.

```
$ ./upload.sh www.nginxtraining.com
```

****
NOTE FROM DAMIAN: cannot update the lab further because this is where I got errors and had to stop.
****

If you look closely at the output there is a **POST** HTTP request which shows where the value is being written.

```
POST /api/8/http/keyvals/ssl_crt HTTP/1.1
```

Also, the **Location** line contains the URL to the written memory file which you can retrieve using **curl -k Location_URL_value**.

Troubleshooting:

Make sure you run the **`upload.sh`** script in the terminal where you have the nginx group as part of your student account (where you ran **`exec su -l student`**) and make sure you are in the directory **`/etc/nginx/ssl-configs/certs`**.

If you need to **delete** the key value store memory entry and run the **`upload.sh`** script again do these commands:

```
$ curl -k -i -X DELETE
https://www.nginxtraining.com/api/8/http/keyvals/ssl_key
$ curl -k -i -X DELETE
https://www.nginxtraining.com/api/8/http/keyvals/ssl_crt
```

OR

```
$ sudo systemctl stop nginx
$ sudo systemctl status nginx
$ sudo systemctl start nginx
```

7. To test that both the certificate and private key files are in memory in the keyval store location.

```
$ curl -k https://www.nginxtraining.com/api/8/http/keyvals | jq
```

8. Now that the key and certificate are in our key-value memory, edit the file **ssl-params.conf** and **comment out** the existing lazy loading definitions for the **ssl_certificate** and **ssl_certificate_key** and then add the new method to use the Dynamic In-memory syntax **data:$variable.**

```
$ sudo vim /etc/nginx/ssl-configs/ssl-params.conf

ssl_certificate data:$crt_pem;
ssl_certificate_key data:$key_pem;
```

NOTE
Your file should look like:

```
ssl_trusted_certificate /etc/nginx/ssl/ca-cert.crt;

#Dynamic InMemory Cert Loading
ssl_certificate data:$crt_pem;
ssl_certificate_key data:$key_pem;

#Dynamic Lazy Loading
#ssl_certificate /etc/nginx/ssl/$ssl_server_name.crt;
#ssl_certificate_key /etc/nginx/ssl/$ssl_server_name.key;

#Traditional cert loading
ssl_certificate /etc/nginx/ssl/www.nginxtraining.com.crt;
ssl_certificate_key /etc/nginx/ssl/www.nginxtraining.com.key;
```

NOTE: Leave the ssl_trusted_certificate directive as it is since it is not loaded dynamically.

9. Edit the api_server.conf file to add proxy ssl name information using the directives **proxy_ssl_name** and **proxy_ssl_server_name**. Add these to the 8080 server block under the location / and after your proxy_pass directive.
   proxy_ssl_name www.nginxtraining.com;
   proxy_ssl_server_name on;

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

   ```
       location / {
           include /etc/nginx/ssl-configs/proxy-ssl-params.conf;
           proxy_pass https://api_server;
           proxy_ssl_name student6.us-west.nginx.learnf5.cloud;
           proxy_ssl_server_name on;
           proxy_ssl_trusted_certificate /etc/letsencrypt/live/student6.us-west.nginx.l
   earnf5.cloud/fullchain.pem;
           proxy_ssl_verify off;
       }
   ```

10. Test and reload NGINX to use your updated configuration files.

    ```
    $ sudo nginx -t && sudo nginx -s reload
    ```

11. Send a GET request to confirm Dynamic Certificate loading is working. The output will be your private key and cert.

```
$ curl -k https://www.nginxtraining.com:8080
$ curl -k https://www.nginxtraining.com:8080
$ curl -k https://www.nginxtraining.com:8080
```

NOTE
You should get alternating upstreams responding such as “this is app1”, then “this is app2!” followed by “this is app1”.

12. Test access to the Juice Shop app.

```
$ curl -k https://www.nginxtraining.com
```

13. For our lab purposes revert back to using the traditional certificates by copying your backup files back to the original files. This will now use the traditional certificate.

```
$ sudo cp /etc/nginx/ssl-configs/ssl-params.{conf,dynamic-in-memory}
$ sudo cp /etc/nginx/ssl-configs/ssl-params.{traditional,conf}
$ sudo cp /etc/nginx/conf.d/juice.{conf,dynamic-in-memory}
$ sudo cp /etc/nginx/conf.d/api_server.{traditional,conf}
$ sudo cp /etc/nginx/conf.d/api_server.{conf,dynamic-in-memory}$ sudo
cp /etc/nginx/conf.d/juice.{traditional,conf}
```

14. Test and reload NGINX to use your updated configuration files.

```
$ sudo nginx -t && sudo nginx -s reload
```

15. Test your configuration with curl. Make sure to scroll back to the beginning output to see 200 OK message.

```
$ cd ~/ssl
$ curl -LIv --cacert ca-cert.crt https://www.nginxtraining.com
```

16. Test using a browser. Your output should be either “this is app1” or “this is app2” and then the Juice Shop app.

```
https://www.nginxtraining.com:8080
https://www.nginxtraining.com
```

Expected Results

You configure and test using the NGINX Dynamic In-memory certificates.


STOP