NGINX Playground

NGINX Playground

1. **Add location-based config and return a 404. Any content that ends with "css, jpg, jpeg, js, json, png, mp4, pdf" should return a 404 with curl.**

* To achieve this, you must set up a location block in your Nginx web server configuration that matches the specified file extensions and returns a 404-status code for them. To perform this, we must verify whether Nginx server has been installed or not if not install NGINX and start the server. After that open NGINX configuration file to update the location-based config to return the specified error.

Upon opening configuration file, we have server block where we must update the following code.

**location ~\* \.(css|jpg|jpeg|js|json|png|mp4|pdf)$**

**{**

**return 404;**

**}**

* After updating the configuration file test, the code for any error, If the configuration is correct, you should see a message saying, "syntax is ok" and "test is successful."
* To verify the functionality of the function that we added, use curl to test if the specified files return a 404. This should return a 404 Not Found.

Internally the process of working will be in this way:

* A client (e.g., a web browser) makes an HTTP request to the server for a resource ending in .css, .jpg, .jpeg,.js, .json, .png, .mp4, or .pdf.
* NGINX checks the request against the defined location blocks and finds a match with the block using the regular expression for these extensions.
* NGINX immediately returns a 404 Not Found response as specified in the location block.
* The response is sent back to the client, and the event is logged in NGINX’s access log.

**2. Log below fields in access log.**

Time

Nginx Version Remote Address Request ID Status

Body Bytes Sent User Agent

Proxy Protocol Address Server Name

Upstream address Request Time

Upstream Connect Time Upstream Header Time Upstream Response Time Request URI

Upstream Status SSL Session reused X-Forwarded-For

To do this we have update the config file with the following code. As explained in the above assessment open NGINX configuration file and search for HTTP block. Inside the http {} block in the nginx.conf file, define a custom log format that includes all the required fields.

http {

log\_format custom\_log\_format '$time\_local '

'nginx/$nginx\_version '

'$remote\_addr '

'$request\_id '

'$status '

'$body\_bytes\_sent '

'$http\_user\_agent '

'$proxy\_protocol\_addr '

'$server\_name '

'$upstream\_addr '

'$request\_time '

'$upstream\_connect\_time '

'$upstream\_header\_time '

'$upstream\_response\_time '

'$request\_uri '

'$upstream\_status '

'$ssl\_session\_reused '

'$http\_x\_forwarded\_for';

access\_log /usr/local/var/log/nginx/access.log custom\_log\_format;

# Other configurations can go here...

}

* **Test the configuration** to ensure there are no syntax errors. If the configuration is correct, you should see a message saying "syntax is ok" and "test is successful."
* Now reload the NGINX server, after reloading NGINX and making some requests to our server, check the access logs to see the custom log entries. Use the following command to verify the logs

**sudo tail -f /usr/local/var/log/nginx/access.log**

We should see log entries with the format you defined, including all the specified fields.

* The default NGINX access log format, known as the "combined" log format, typically logs a standard set of information for each request. The default log format logs the entire HTTP request line but our custom format logs just the URI part of the request. This might be more focused and easier to analyze for certain applications.
* The custom log format provides more detailed and granular information, especially related to upstream server interactions, SSL/TLS usage, and unique request identifiers.

1. **Add HTTP security headers at NGINX only if header value is not set in response from the upstream. Default is set to none and only set in case nothing is returned from the upstream.**

|  |  |
| --- | --- |
| HTTP Headers | Value |
| Strict-Transport-Security | max-age=31536000; includeSubDomains |
| X-Content-Type-Options | nosniff |
| X-XSS-Protection | 1; mode=block |
| X-Frame-Options | DENY |
| Content-Security-Policy | frame-ancestors 'none' |
| Access-Control-Allow-Credentials | TRUE |

* To add HTTP security headers in NGINX only if they are not set by the upstream server, we can use the add\_header directive with the always parameter. The always parameter allows us to add headers conditionally, ensuring they are only set when not already present in the response from the upstream server.
* Open terminal, navigate to the NGINX configuration directory and open the nginx.conf file.
* In our server {} block, add the following add\_header directives to set the security headers only if they are not already present in the response from the upstream server

Add security headers conditionally

add\_header Strict-Transport-Security "max-age=31536000; includeSubDomains" always; .   
add\_header X-Content-Type-Options "nosniff" always; add\_header X-XSS-Protection "1; mode=block" always; add\_header X-Frame-Options "DENY" always; add\_header Content-Security-Policy "frame-ancestors 'none'" always;   
add\_header Access-Control-Allow-Credentials "TRUE" always;

* add\_header ... always;: This directive adds the header only if it is not already set by the upstream server. If the upstream server has already set the header, NGINX will not override it.
* Test the configuration to ensure there are no syntax errors, after reloading NGINX, you can use curl to inspect the headers returned by your server. The response should include the security headers only if they are not set by the upstream server.