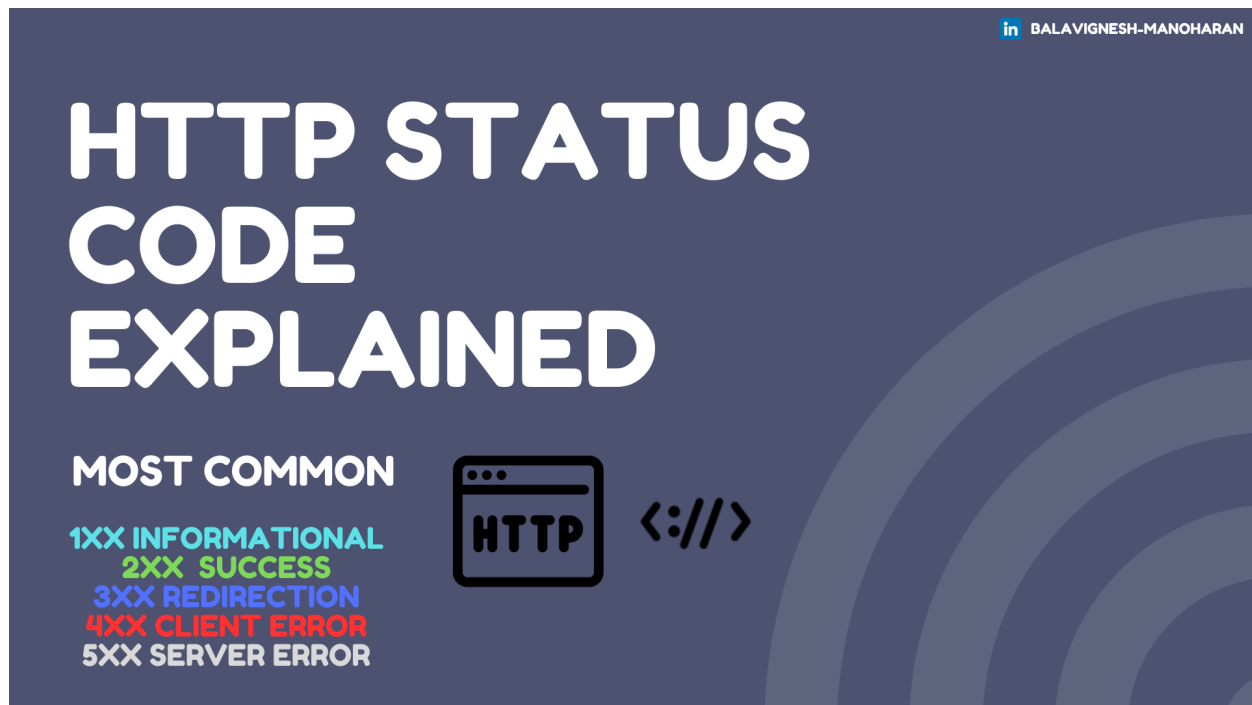


HTTP Status Code Explained - 13 HTTP Status Codes



Introduction:

HTTP Status code are an essential part of communication. They provide the information about the outcome of the request made to a server. They have a very significance in troubleshooting as well as understanding the state of your application.

In this blog we will look into the most common HTTP Status code that one generally encounters and what does that exactly mean.

Broadly they are split into 5 different categories: The difference in classes are indicated through the first digit of the error code. Example: 404 or 4xx.

- **1xx - Informational**
- **2xx - Successful**

- **3xx - Redirection**
- **4xx - Client Error**
- **5xx - Server Error**

1xx - Informational

- 100 Continue

The initial part of the request has been received by the server and that the client should proceed with the request.

- 101 Switching protocols

The server is changing protocols as requested by the client.

2xx - Success Code

- 200 OK

The request was successful.

- 201 Created

The request was successful, and a new resource or multiple new resources has been created. [POST request]

- 204 No Content

The server processed the request, but there is no content to send back. [DELETE request]

3xx - Redirection Code

- 301 Moved Permanently

The requested resource has been moved permanently to a new URL.

- 302 Found

The requested resource has been changed temporarily. Further change can be made in the future, so the same URL should be used by the client in future.

4xx - Client Error Code

- 400 Bad Request

The server could not understand the request because of invalid syntax.

- 401 Unauthorized

Although it specifies unauthorized, it should be unauthenticated. The request cannot be completed as the server requires user authentication.

- 403 Forbidden

The request has been rejected as the client does not have right to access the content. Basically insufficient permission.

- 404 Not Found

The server cannot find the requested resource. As the resource itself does not exists. This is the most common error encountered by the users.

5xx - Server Error Code

- 500 Internal Server Error

There is some issue with the server. The code is not working, the new build has some issues, problem with the server configuration files.

- 502 Bad Gateway

This error occurs when a server acting as a gateway or proxy receives an invalid response from an upstream server. It typically means that while the proxy server successfully forwarded the request, it didn't receive a valid response back. This often happens in microservice architectures where one service relies on responses from another.

Hands-on Demo for some of the status code from each categories:

Pre-requisites:

- AWS Account
- Python3 Installed

Steps:

1. Login to your AWS Account
2. Launch an free-tier t2.micro EC2 instance.
3. Update the server using the below code.

```
sudo apt update
```

4. Clone the application using the below code

```
git clone https://github.com/iam-veeramalla/http-status-codes.gi
```

5. Install python3-venv using the below command

```
sudo apt install python3.10-venv
```

6. Create virtual env using the below code.

```
python3 -m venv myenv

# Activate the virtual environment
# On Windows
myenv\Scripts\activate

# On macOS/Linux
source myenv/bin/activate
```

7. Go inside the http-status-codes folder and then go inside the 502 folder for this example.

```
cd http-status-codes/502
```

8. Install dependencies

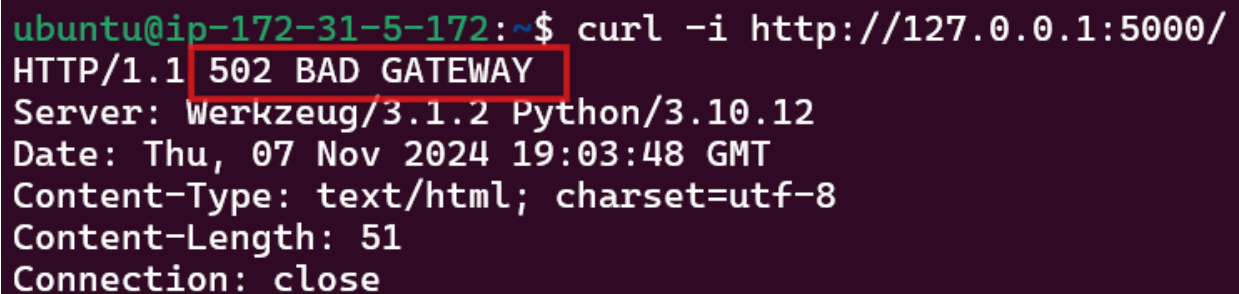
```
pip3 install -r requirements.txt
```

9. Run the application

```
python3 bad_gateway.py
```

10. Open a new terminal and Verify using CURL

```
curl -i http://127.0.0.1:5000/
```

A terminal window with a dark purple background. The prompt is 'ubuntu@ip-172-31-5-172:~\$'. The command 'curl -i http://127.0.0.1:5000/' has been executed. The output is: 'HTTP/1.1 502 BAD GATEWAY', 'Server: Werkzeug/3.1.2 Python/3.10.12', 'Date: Thu, 07 Nov 2024 19:03:48 GMT', 'Content-Type: text/html; charset=utf-8', 'Content-Length: 51', and 'Connection: close'. The status code '502 BAD GATEWAY' is highlighted with a red rectangular box.

```
ubuntu@ip-172-31-5-172:~$ curl -i http://127.0.0.1:5000/  
HTTP/1.1 502 BAD GATEWAY  
Server: Werkzeug/3.1.2 Python/3.10.12  
Date: Thu, 07 Nov 2024 19:03:48 GMT  
Content-Type: text/html; charset=utf-8  
Content-Length: 51  
Connection: close
```

Solution: Increase timeout or set a ideal timeout

For the next example of 404 you need to switch to a 404 folder.

1. Go back to 404 folder

```
cd 404
```

2. Follow the same steps and simulate the examples.

3. Run the python command

```
python3 not_found.py
```

4. Open a new terminal and Verify using CURL

```
curl -i http://127.0.0.1:5000/
```

```
ubuntu@ip-172-31-5-172:~$ curl -i http://127.0.0.1:5000/
HTTP/1.1 404 NOT FOUND
Server: Werkzeug/3.1.2 Python/3.10.12
Date: Thu, 07 Nov 2024 19:07:13 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 73
Connection: close

Not Found: The requested resource does not exist on the upstream service
```

Solution: Check the URL and also make sure the requested resource exists.

For the next example of 302 you need to switch to a 302 folder.

1. Go back to 302 folder.

```
cd 302
```

2. Rename the file.

```
sudo mv 'redirect.py ' redirect.py
```

3. Run the python command

```
python3 redirect.py
```

4. Open a new terminal and Verify using CURL

```
curl -i http://127.0.0.1:5000/
```

```
ubuntu@ip-172-31-5-172:~/http-status-codes/302$ curl -i http://127.0.0.1:5000/
HTTP/1.1 302 FOUND
Server: Werkzeug/3.1.2 Python/3.10.12
Date: Thu, 07 Nov 2024 19:19:36 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 195
Location: /get
Connection: close

<!doctype html>
<html lang=en>
<title>Redirecting...</title>
<h1>Redirecting...</h1>
<p>You should be redirected automatically to the target URL: <a href="/get">/get</a>. If not, click the link.
```

Solution: In case of an unexpected redirection, check the application or server configuration.

For the next example of 200 you need to switch to a 200 folder.

1. Go back to 200 folder.

```
cd 200
```

2. Run the python command

```
python3 success.py
```

3. Open a new terminal and Verify using CURL

```
curl -i http://127.0.0.1:5000/
```

```
ubuntu@ip-172-31-5-172:~/http-status-codes/302$ curl -i http://127.0.0.1:5000/
HTTP/1.1 200 OK
Server: Werkzeug/3.1.2 Python/3.10.12
Date: Thu, 07 Nov 2024 19:24:02 GMT
Content-Type: application/json
Content-Length: 299
Connection: close

{
  "args": {},
  "headers": {
    "Accept": "*/*",
    "Accept-Encoding": "gzip, deflate",
    "Host": "httpbin.org",
    "User-Agent": "python-requests/2.31.0",
    "X-Amzn-Trace-Id": "Root=1-672d13cd-322923d727f3ec3d0c6aae8f"
  },
  "origin": "13.232.58.10",
  "url": "https://httpbin.org/get"
}
```

The request is successful!

Checking HTTP Status in Chrome Browser

- Open the URL in which you want to check in your browser.

- Open the developer tab (F12) and go to network section.
- Refresh the page.
- Scroll from the list of requests and look for type document.
- In "Status" you can find the HTTP response code.

Finally we have executed some of the HTTP Code hands-on. These are some of the most popular code which you might encounter frequently.

Tip: Instead of showing "404 Not Found" you can display something like "The page you're looking for doesn't exists"

Conclusion:

Understanding the HTTP codes are crucial for both frontend and backend developers. Whether you're debugging an issue or optimizing performance, these status code are your first line of sight into your behavior of your services.

Thank you!

If you found this post useful, give it a like 👍

Repost 🔄

Follow @Bala Vignesh for more such posts 🚀