HTTP

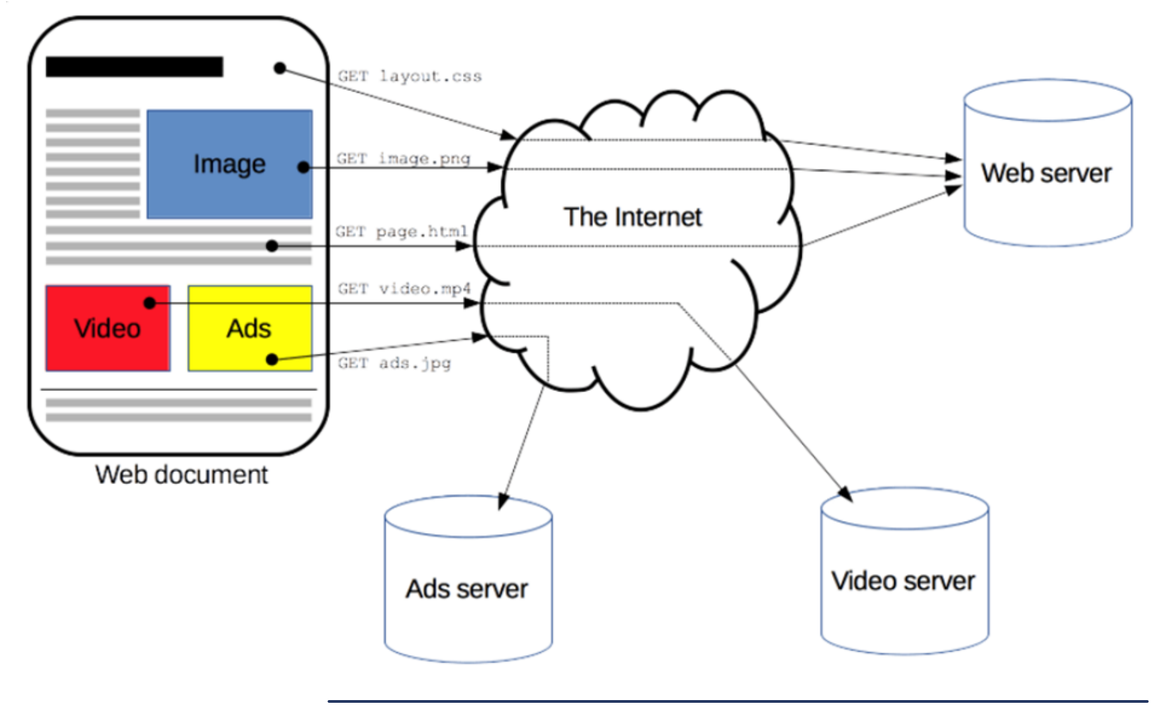
HyperText Transfer Protocol

What is HTTP?

An application-layer in the TCP/IP suite for transmitting hypermedia documents, such as HTML

Designed for communication between web browsers and web servers

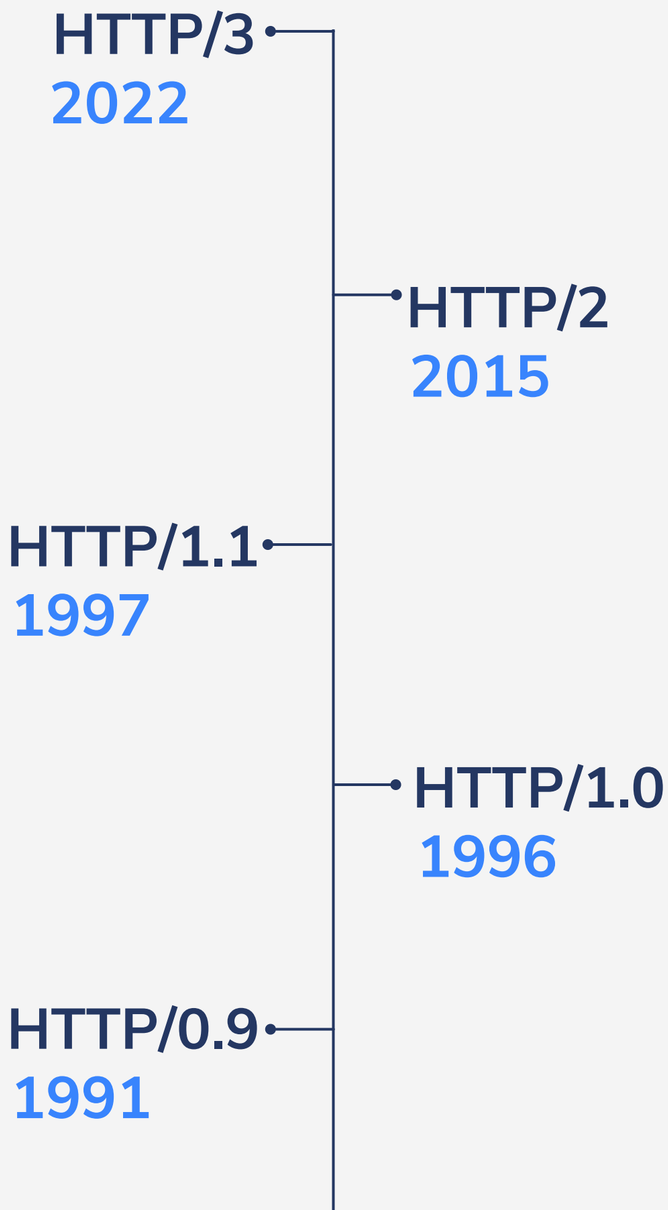
It was developed by Tim Berners-Lee and his team between 1989-1991.



Summary of HTTP versions

There exist 5 different versions of HTTP.

The 0.9 and 1.0 are now **obsolete**



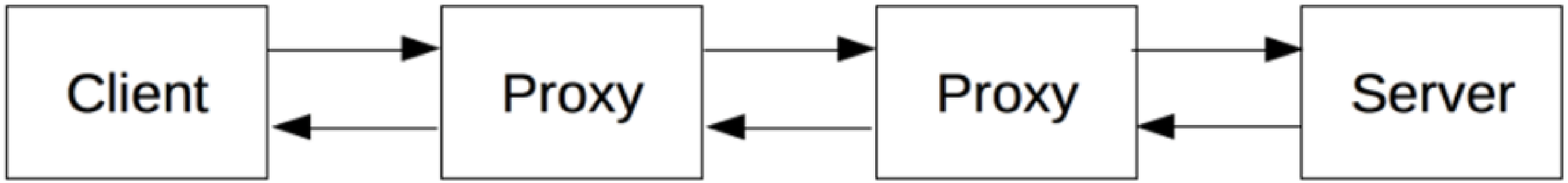
How does it work?

It follows a classical client-server model

A client opens a connection to make a request, then waits until it receives a response.
Usually, the client is a web browser.

Proxies

Between the client and the server there are numerous entities, collectively called **proxies**, which perform different operations and act as gateways



HTTP FLOW

When the client wants to communicate with the server, the following steps happen:

```
GET / HTTP/1.1
Host: developer.mozilla.org
Accept-Language: fr
```

```
HTTP/1.1 200 OK
Date: Sat, 09 Oct 2010 14:28:02 GMT
Server: Apache
Last-Modified: Tue, 01 Dec 2009 20:18:22 GMT
ETag: "51142bc1-7449-479b075b2891b"
Accept-Ranges: bytes
Content-Length: 29769
Content-Type: text/html

<!DOCTYPE html>… (here come the 29769 bytes of the requested web page)
```

01 **Open a TCP connection**

It is used to send requests and receive an answer. The client can open a new connection, reuse an existing or open different TCP connections to the servers.

02 **Send an HTTP message**

HTTP/1.1 messages are human-readable.

03 **Read the response sent by the server**

04 **Close or reuse the connection for further requests**

HTTP MESSAGES REQUESTS

HTTP requests have the following elements:

An HTTP method

They are many, the more used are:

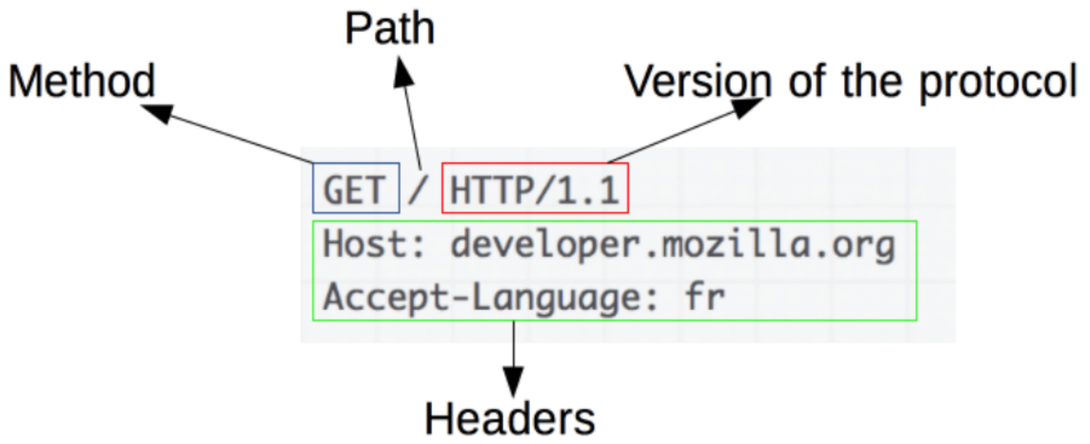
- GET
  - used to read a representation of a resource
  - in case of response - status code 200
  - if error - 404 (not found) or 400 (bad request)
- POST
  - often used to create new resources
  - if succesful - status code 201
- PATCH
  - used to modify resources
  - the patch request should contain the instructions of how a resource should be changed
- DELETE
  - used to delete a resource identified by filters or ID
  - if succesful deletion - status code 204 (No Content)

A path

Version of the HTTP protocol

Headers

That convey additional information for the servers



HTTP MESSAGES RESPONSES

HTTP responses have the following elements:

Version of the protocol they follow

Status code

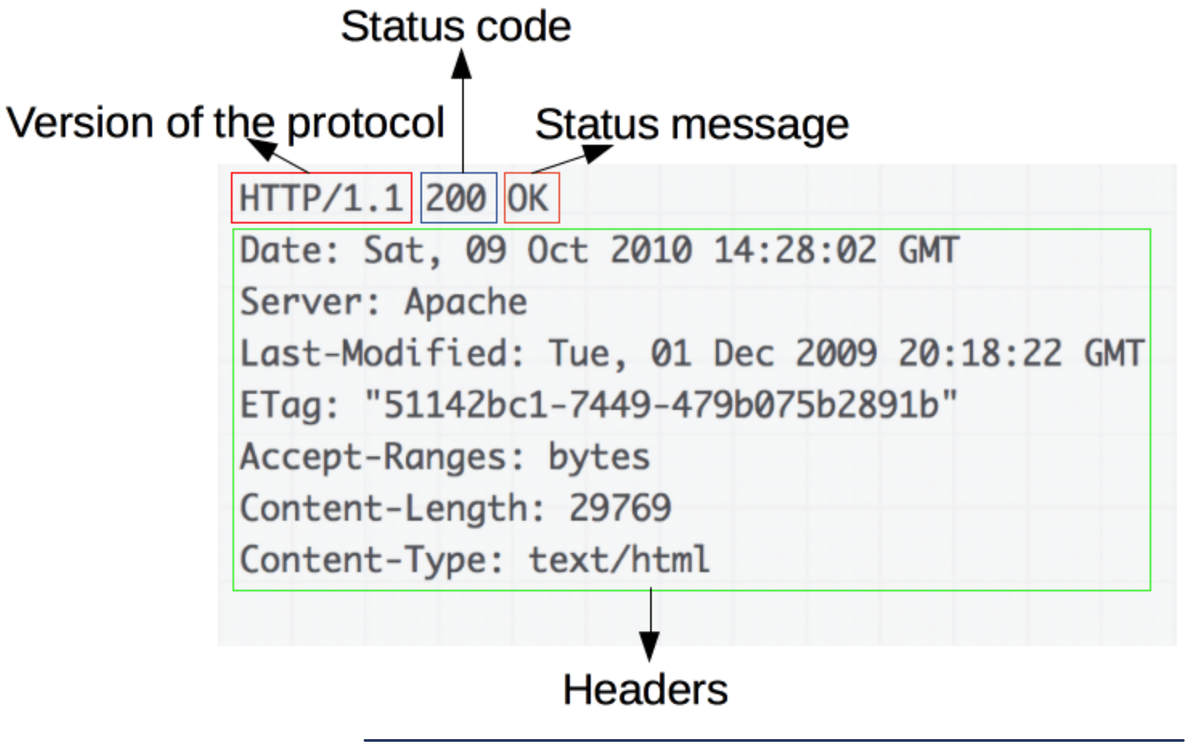
Indicating if the request was successful or not

Status message

Short description of the status code

Headers

Like the ones for the requests



Basics aspects of HTTP/1.1

01

Simple

HTTP/1.1 is designed to be simple. It can be read and understood by humans, making easier testing for developers and reducing complexity for newcomers.

02

Extensible

HTTP headers make the protocol easy to extend and experiment with.

03

Stateless

There is no link between two requests being successively carried out on the same connection.

What can be controlled by HTTP/1.1?

Caching

HTTP/1.1 can control how documents are cached

Sessions

HTTP/1.1 cookies allow you to link requests with the state of the server, and it creates sessions.

Authentification

HTTP/1.1 can provide basic authentification by using WWW-Authenticate and similar headers or by setting specific session using HTTP/1.1 cookies.
Authentification protects pages so that only specific users can access them.

Proxying and tunneling

Clients and servers usually are located on intranets and hide their true IP address. To cross this network barrier, HTTP/1.1 requests go through proxies.

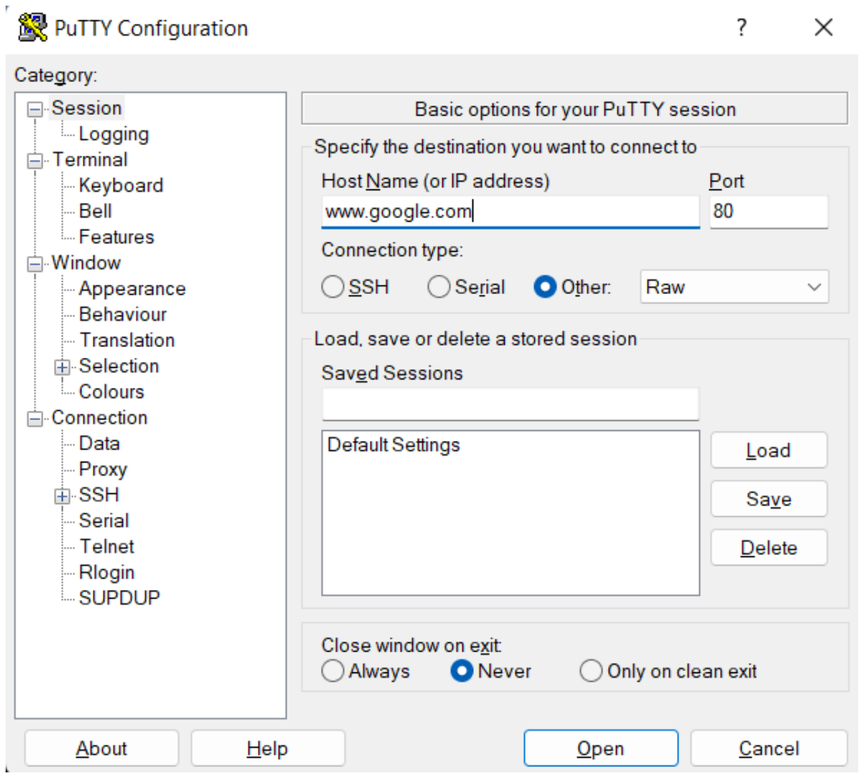
Relaxing the origin constraint

To prevent snooping and other privacy invasions, Web browsers enforce strict separation between Web sites. HTTP headers can relax this strict separation on the server side, allowing a document to become a patchwork of information sourced from different domains

Exercise 1.1

Try a GET

1. With putty, I connected myself to www.google.com
   - I used some specific settings that you can see on the image below



2. Then with the command prompt, I made a GET request to www.google.com

```
GET / HTTP/1.1
Host:www.google.com
```

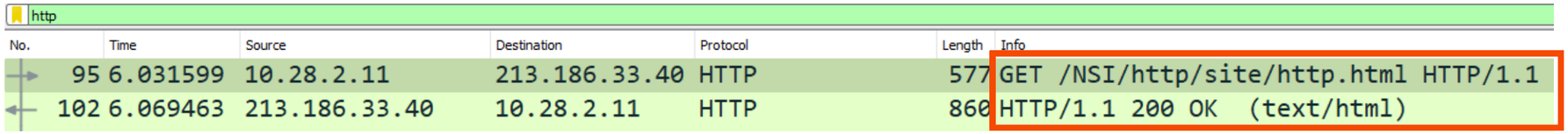
3. We can see that google answered.
   - Status code - 200 OK

```
HTTP/1.1 200 OK
Date: Fri, 07 Oct 2022 08:46:22 GMT
Expires: -1
Cache-Control: private, max-age=0
Content-Type: text/html; charset=ISO-8859-1
Server: gws
X-XSS-Protection: 0
X-Frame-Options: SAMEORIGIN
Set-Cookie: AEC=AakniGP8d4JuSw4fATBAIrnOjWwpezA0_4lkGGdccA6gUiHrDAxiuZ-XDlQ; exp
ires=Wed, 05-Apr-2023 08:46:22 GMT; path=/; domain=.google.com; Secure; HttpOnly
; SameSite=lax
Accept-Ranges: none
Vary: Accept-Encoding
Transfer-Encoding: chunked
```

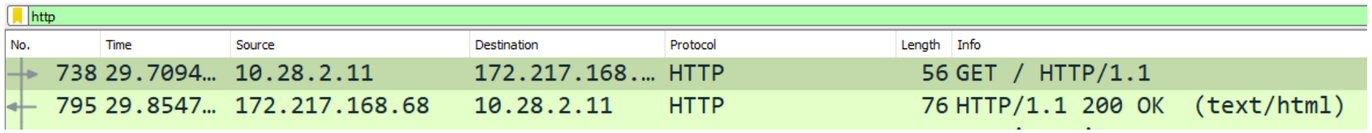
Exercise 1.2

Replay a GET

1. I opened Wireshark, to try to find the methods I used previously during my web browsing.
    - We can see in the picture below, circled in red, that the GET method was used



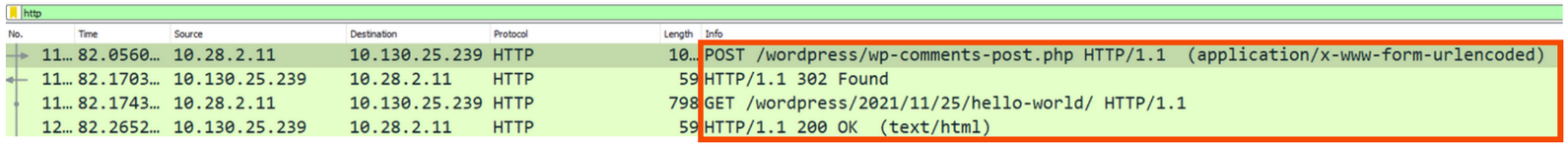
2. I then replaied the GET method with putty:



Exercise 1.3

Replay a POST

1. With WireShark, I captured the communication between me and the server when I posted a comment on a website:



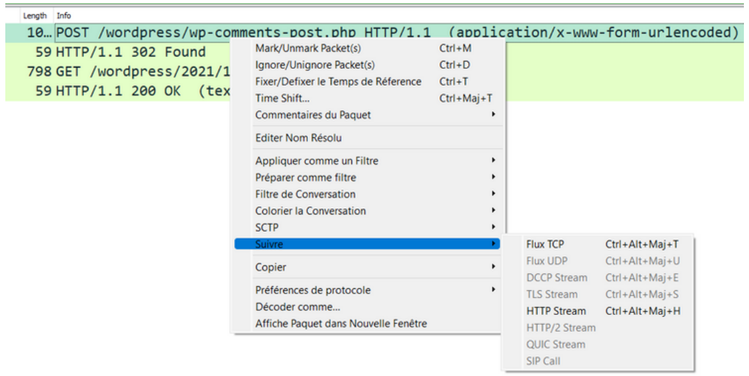
We can see that the POST method was used to post the comment, and the GET method to charge the website.

Exercise 1.3

Replay a POST changing the comment content

To replay a post, changing the comment content, I did the following steps:

1. Right click on the POST info (on Wireshark) - "Suivre" - "Flux TCP"



Exercise 1.3

Replay a POST changing the comment content

2. I copied the red section, and pasted it onto notes.

3. Then I changed the comment (you can see it on the image below, red underligned)

*Sans titre - Bloc-notes

Fichier Modifier Affichage

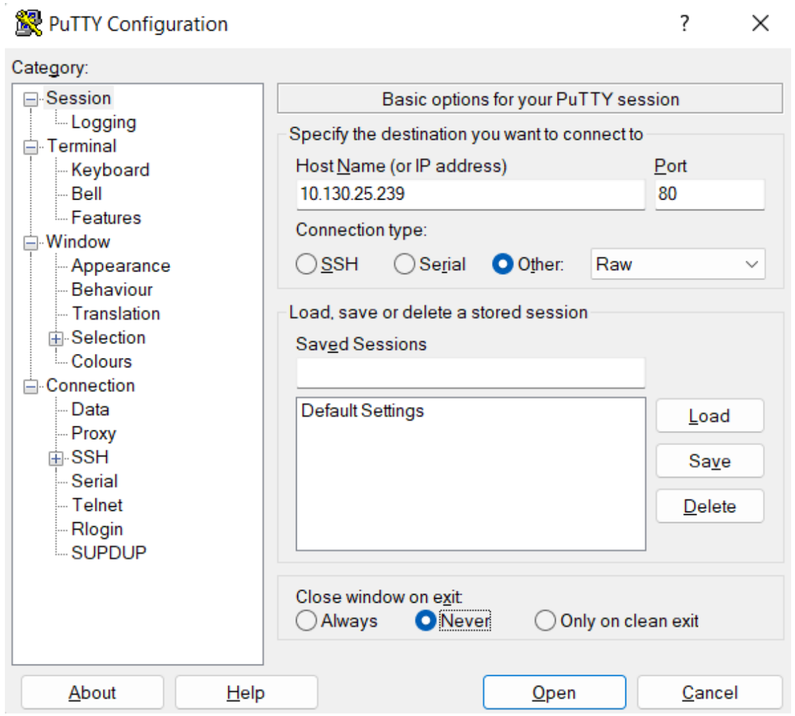
```
POST /wordpress/wp-comments-post.php HTTP/1.1
Host: 10.130.25.239
Connection: keep-alive
Content-Length: 157
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://10.130.25.239
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/105.0.0.0 Safari/537.36 Edg/105.0.1343.53
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Referer: http://10.130.25.239/wordpress/2021/11/25/hello-world/
Accept-Encoding: gzip, deflate
Accept-Language: fr,fr-FR;q=0.9,en;q=0.8,en-GB;q=0.7,en-US;q=0.6,pt;q=0.5
Cookie: comment_author_0190db5a3df525585739bd739a2cc472=Userx; comment_author_email_0190db5a3df525585739bd739a2cc472=netflix84sab%40gmail.com

comment=hello+ice+tea&author=Userx&email=netflix84sab%40gmail.com&url=&wp-comment-cookies-consent=yes&submit=Post+Comment&comment_post_ID=1&comment_parent=0
```

Exercise 1.3

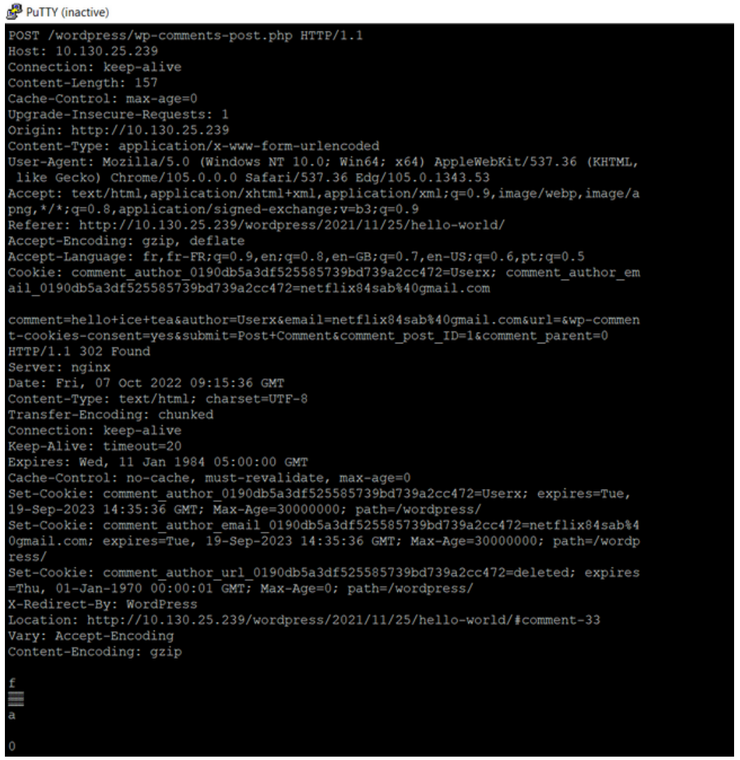
Replay a POST changing the comment content

4. Then on putty, I connected myself to the website with the IP address.



5. I pasted on the command prompt the POST request.

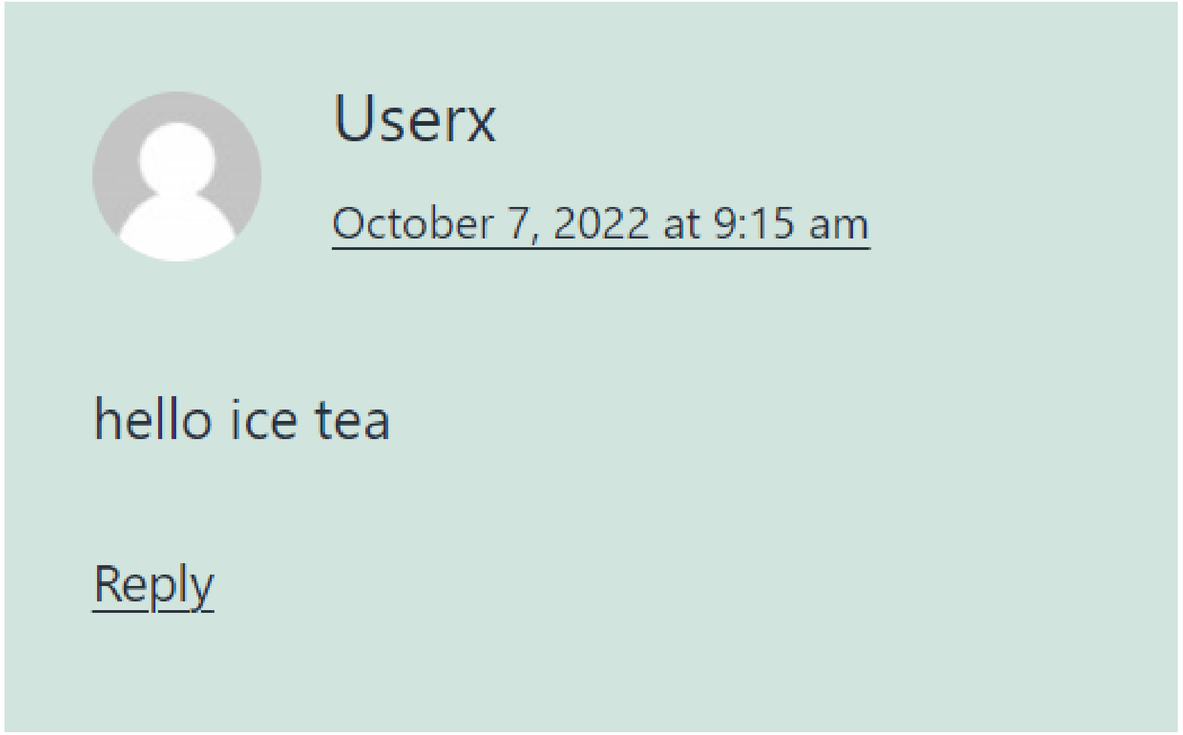
- We can see that the request was successful. (status code = 302 Found)



Exercise 1.3

Replay a POST changing the comment content

6. Then I checked on the website if my new comment appeared, and it did.



Bibliography of pictures

- Slide 1 - https://upload.wikimedia.org/wikipedia/commons/8/83/Internet1.svg
- Slide 2 - https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview/fetching_a_page.png
- Slide 5, picture 1 and 2
  - Taken on https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview#http_flow
- Slide 6 - https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview/http_request.png
- Slide 7 - https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview/http_response.png

Bibliography of informations

- https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview
- https://doc.oroinc.com/api/http-methods
- https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol