

# Compte rendu HTTP

1. En utilisant la commande nc (netcat), émettez une requête HTTP sur le port 80 de la machine [www.example.com](http://www.example.com). Pour cette première requête, vous utiliserez le protocole 1.0 et la méthode HEAD.

```
C:\Windows\System32>ncat www.example.com 80

HEAD / HTTP/1.0
Host: www.example.com

HTTP/1.0 200 OK
Content-Type: text/html
ETag: "84238dfc8092e5d9c0dac8ef93371a07:1736799080.121134"
Last-Modified: Mon, 13 Jan 2025 20:11:20 GMT
Cache-Control: max-age=86000
Date: Thu, 11 Sep 2025 07:52:20 GMT
Connection: close
```

2. Exécutez une nouvelle fois la commande précédente, mais cette fois en utilisant la méthode HTTP GET. Qu'obtenez-vous en plus par rapport à la requête précédente ?

```
PS C:\Windows\System32>ncat www.example.com 80

GET / HTTP/1.0
Host: www.example.com

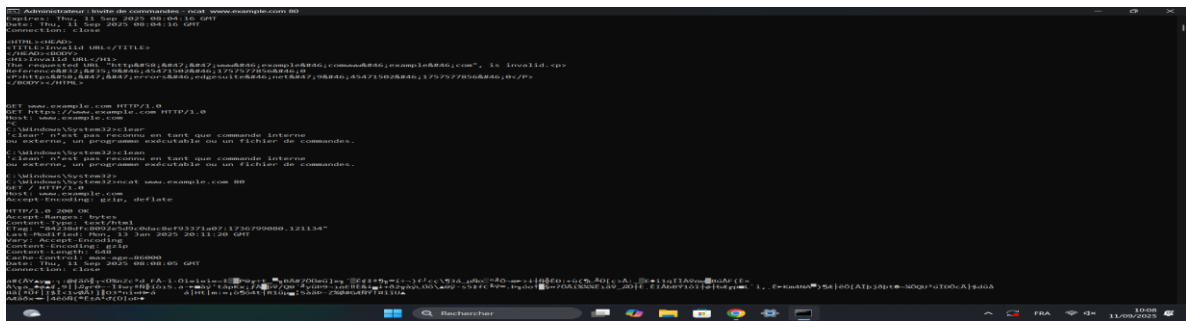
HTTP/1.0 200 OK
Content-Type: text/html
ETag: "84238dfc8092e5d9c0dac8ef93371a07:1736799080.121134"
Last-Modified: Mon, 13 Jan 2025 20:11:20 GMT
Cache-Control: max-age=86000
Date: Thu, 11 Sep 2025 07:52:20 GMT
Content-Length: 1256
Connection: close
X-NC: 5

<!doctype html>
<html>
<head>
<title>Example Domain</title>

<meta charset="utf-8" />
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1" />
<style type="text/css">
body {
background-color: #f0f0f2;
margin: 0;
padding: 0;
font-family: -apple-system, system-ui, BlinkMacSystemFont, "Segoe UI", "Open Sans", "Helvetica Neue", Helvetica, Arial, sans-serif;
}
div {
width: 600px;
margin: 5em auto;
padding: 2em;
background-color: #f0f0f2;
border-radius: 0.5em;
box-shadow: 2px 3px 3px rgba(0,0,0,0.02);
}
a:link, a:visited {
color: #8080ff;
text-decoration: none;
}
@media (max-width: 700px) {
div {
margin: 0 auto;
width: auto;
}
}
</style>
</head>
```

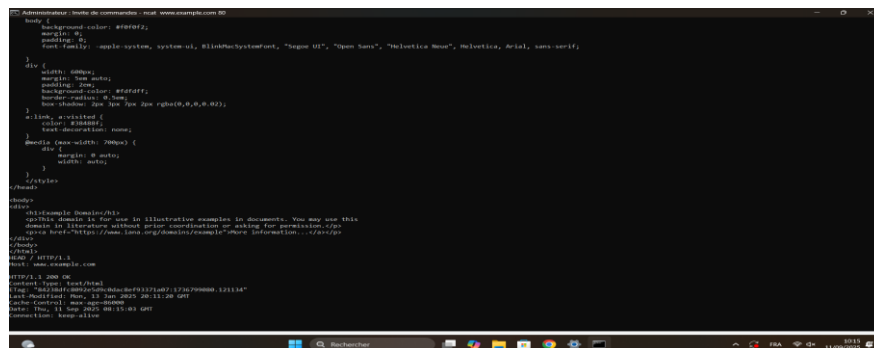
J'obtiens le corps de la page HTML en plus de l'entête.

3. Exécutez la commande précédente, mais cette fois en ajoutant l'entête Accept-Encoding: gzip, deflate. Que remarquez-vous ?



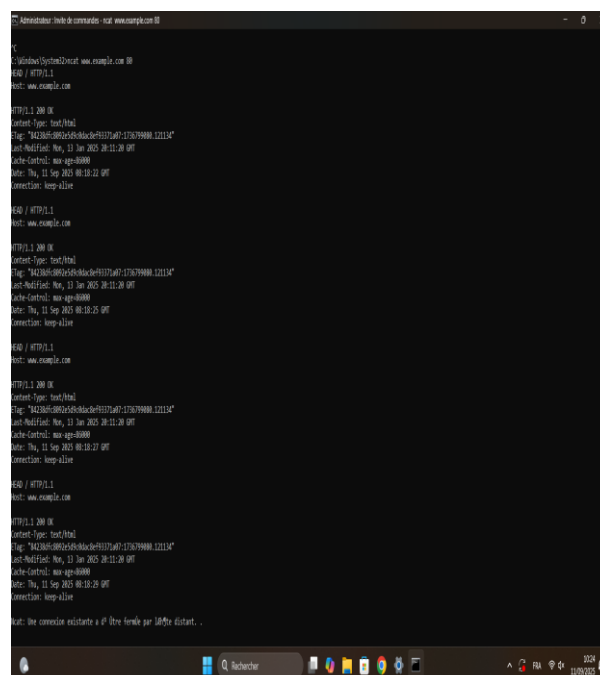
Le corps de la réponse retournée est incompréhensible car il est mal encodé/décodé.

4. Exécutez une nouvelle fois la commande de la première question, mais cette fois avec la version 1.1 du protocole HTTP. Que remarquez-vous par rapport à la première question ? Quel mécanisme vu est mis en œuvre ?



Ici la connexion est dite “keep-alive” et plus “close” c’est le mécanisme de connexion persistante.

5. Émettez plusieurs requêtes HTTP au sein d'une même connexion TCP.



On peut effectivement réaliser plusieurs requêtes sur la même connexion.

6. Émettez une requête HTTP avec la méthode POST sur le port 80 de la machine [www.example.com](http://www.example.com) pour transmettre les données username=bob. Qu'obtenez-vous comme réponse ? Que pouvez-vous en déduire ?

(Bascule sous linux pour php-cli)

```
sharizahr@sharizahr-GE65-Raider-9SE:~$ nc www.example.com 80
POST / HTTP/1.1
Host: www.example.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 13

username=bob

HTTP/1.1 403 Forbidden
Mime-Version: 1.0
Content-Type: text/html
Content-Length: 369
Cache-Control: max-age=86000
Date: Thu, 11 Sep 2025 12:15:16 GMT
Connection: close

<HTML><HEAD>
<TITLE>Access Denied</TITLE>
</HEAD><BODY>
<H1>Access Denied</H1>

You don't have permission to access "http://www.example.com/" on this server.<P>
Reference: https://errors.edgesuite.net/18/48471502/1757592907/42b330df</P>
</BODY>
</HTML>
```

7. Installez la commande php-cli sur votre machine (sudo apt install php-cli, pour une distribution Debian ou Ubuntu). Démarrez localement sur le port 8080 un serveur Web capable d'exécuter du code PHP, via la commande php -S localhost:8080. Émettez la même requête qu'à la question précédente, mais cette fois pour ce serveur Web. Quelle réponse obtenez-vous ?

```
[Thu Sep 11 11:15:00 2025] 127.0.0.1:54746 Accepted
[Thu Sep 11 11:15:10 2025] 127.0.0.1:54746 [404]: POST / - No such file or directory
[Thu Sep 11 11:15:10 2025] 127.0.0.1:54746 Closing
```

```
sharizahr@sharizahr-GE65-Raider-9SE:~$ nc localhost 8080
POST / HTTP/1.1
Host: localhost
Content-Type: application/x-www-form-urlencoded
Content-Length: 13

username=bob

HTTP/1.1 404 Not Found
Host: localhost
Date: Thu, 11 Sep 2025 09:15:10 GMT
Connection: close
Content-Type: text/html; charset=UTF-8
Content-Length: 533

<!doctype html><html><head><title>404 Not Found</title><style>
body { background-color: #f0f0f0; color: #333333; margin: 0; padding: 0; }
h1 { font-size: 1.5em; font-weight: normal; background-color: #9999cc; min-height: 2em; line-height: 2em; border-bottom: 1px inset black; margin: 0; }
h1, p { padding-left: 10px; }
code, url { background-color: #eeeeee; font-family: monospace; padding: 0 2px; }
</style>
</head><body><h1>Not Found</h1><p>The requested resource <code class="url"></code> was not found on this server.</p></body></html>
```

8. Créez dans le répertoire où vous avez exécuté la commande php -S localhost:8080, un fichier index.php contenant le code suivant. Exécutez une

nouvelle fois la commande précédente. Quelle réponse obtenez-vous cette fois ?

```
<!doctype html>
<html lang="fr">
<head>
  <meta charset="utf-8">
  <title>HTTP POST</title>
</head>
<body>

<?php
  echo "HTTP Method = " . $_SERVER['REQUEST_METHOD'] . "\n";
  echo "Username = " . $_REQUEST['username'] . "\n";
?>

</body>
</html>
```

```
[Thu Sep 11 11:12:48 2025] 127.0.0.1:54948 Accepted
[Thu Sep 11 11:12:51 2025] 127.0.0.1:54948 [200]: POST /
[Thu Sep 11 11:12:51 2025] 127.0.0.1:54948 Closing
```

```
sharizahr@sharizahr-GE65-Raider-9SE:~$ nc localhost 8080
POST / HTTP/1.1
Host: localhost
Content-Type: application/x-www-form-urlencoded
Content-Length: 13

username=bob
HTTP/1.1 200 OK
Host: localhost
Date: Thu, 11 Sep 2025 09:12:51 GMT
Connection: close
X-Powered-By: PHP/8.3.6
Content-type: text/html; charset=UTF-8

<!doctype html>
<html lang="fr">
<head>
  <meta charset="utf-8">
  <title>HTTP POST</title>
</head>
<body>

HTTP Method = POST
Username = bob

</body>
</html>
```

9. En utilisant la commande curl, émettez une requête HTTP avec la méthode GET sur le port 80 de la machine [www.example.com](http://www.example.com).

```
sharizahr@sharizahr-GE65-Raider-9SE:~$ curl -X GET http://www.example.com:80/
<!doctype html>
<html>
<head>
  <title>Example Domain</title>

  <meta charset="utf-8" />
  <meta http-equiv="Content-type" content="text/html; charset=utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <style type="text/css">
    body {
      background-color: #f0f0f2;
      margin: 0;
      padding: 0;
      font-family: -apple-system, system-ui, BlinkMacSystemFont, "Segoe UI", "Open Sans", "Helvetica Neue", Helvetica, Arial, sans-serif;
    }
    div {
      width: 600px;
      margin: 5em auto;
      padding: 2em;
      background-color: #fdfdff;
      border-radius: 0.5em;
      box-shadow: 2px 3px 7px 2px rgba(0,0,0,0.02);
    }
    a:link, a:visited {
      color: #38488f;
      text-decoration: none;
    }
    @media (max-width: 700px) {
      div {
        margin: 0 auto;
        width: auto;
      }
    }
  </style>
</head>
<body>
```

10. En utilisant la commande curl, émettez une requête HTTP avec la méthode POST pour transmettre à votre serveur Web local fonctionnant sur le port 8080 les données username=bob.

```
[Thu Sep 11 14:18:11 2025] 127.0.0.1:59164 Accepted
[Thu Sep 11 14:18:11 2025] 127.0.0.1:59164 [200]: POST /
[Thu Sep 11 14:18:11 2025] 127.0.0.1:59164 Closing
```

```
sharizahr@sharizahr-GE65-Raider-9SE:~$ curl -X POST http://localhost:8080/ -d "username=bob"
<!doctype html>
<html lang="fr">
<head>
  <meta charset="utf-8">
  <title>HTTP POST</title>
</head>
<body>

HTTP Method = POST
Username = bob

</body>
</html>
```

11. En utilisant la commande curl et l'option --http2 de celle-ci émettez une requête HTTP GET au serveur [www.nghttp2.org](http://www.nghttp2.org) sur le port 80. Vous ajouterez l'option -v pour avoir une trace de la requête et des entêtes de la réponse. Qu'observez-vous ?

```

sharizahr@sharizahr-GE65-Raider-9SE:~$ curl -v --http2 http://www.nghttp2.org:80/
* Host www.nghttp2.org:80 was resolved.
* IPv6: 2400:8902::f03c:91ff:fe69:a454
* IPv4: 139.162.123.134
*   Trying [2400:8902::f03c:91ff:fe69:a454]:80...
*   Trying 139.162.123.134:80...
* Connected to www.nghttp2.org (2400:8902::f03c:91ff:fe69:a454) port 80
> GET / HTTP/1.1
> Host: www.nghttp2.org
> User-Agent: curl/8.5.0
> Accept: */*
> Connection: Upgrade, HTTP2-Settings
> Upgrade: h2c
> HTTP2-Settings: AAMAAABkAAQAAoAAAAIAAAAA
>
< HTTP/1.1 101 Switching Protocols
< Connection: Upgrade
< Upgrade: h2c
* Received 101, Switching to HTTP/2
* Copied HTTP/2 data in stream buffer to connection buffer after upgrade: len=39
< HTTP/2 200
< date: Thu, 11 Sep 2025 12:33:25 GMT
< content-type: text/html
< last-modified: Tue, 02 Sep 2025 12:20:44 GMT
< etag: "68b6e11c-18b4"
< accept-ranges: bytes
< content-length: 6324
< x-backend-header-rtt: 0.001432
< server: nghttpx
< alt-svc: h3=":443"; ma=3600
< via: 2 nghttpx
< x-frame-options: SAMEORIGIN
< x-xss-protection: 1; mode=block
< x-content-type-options: nosniff
<
<!DOCTYPE html>
<!--[if IEMobile 7 ]><html class="no-js iem7"><![endif]-->
<!--[if lt IE 9]><html class="no-js lte-ie8"><![endif]-->

```

J'observe que la requête s'est passée en deux temps. D'abord la partie HTTP/1.1 avec la demande d'upgrade puis l'upgrade pour HTTP/2.

12. En utilisant la commande curl et l'option `--http2` de celle-ci émettez une requête HTTP GET au serveur [www.nghttp2.org](http://www.nghttp2.org), mais cette fois sur le port 443. Vous ajouterez l'option `-v` pour avoir une trace de la requête et des entêtes de la réponse. Qu'observez-vous ?

```

sharizahr@sharizahr-GE65-Raider-9SE:~$ curl -v --http2 http://www.nghttp2.org:443/
* Host www.nghttp2.org:443 was resolved.
* IPv6: 2400:8902::f03c:91ff:fe69:a454
* IPv4: 139.162.123.134
*   Trying [2400:8902::f03c:91ff:fe69:a454]:443...
*   Trying 139.162.123.134:443...
* Connected to www.nghttp2.org (2400:8902::f03c:91ff:fe69:a454) port 443
> GET / HTTP/1.1
> Host: www.nghttp2.org:443
> User-Agent: curl/8.5.0
> Accept: */*
> Connection: Upgrade, HTTP2-Settings
> Upgrade: h2c
> HTTP2-Settings: AAMAAABkAAQAAoAAAAIAAAAA
>
* Empty reply from server
* Closing connection
curl: (52) Empty reply from server

```

Le server ne donne pas suite à la requête et coupe la connexion car il attend une communication TLS sur ce port.