

# DOCUMENTATION TECHNIQUE PROJET BTC25.1 CYBERSECURITE

Rémi KORZENIOWSKI, Léon VACHETTE et Théo Menant--Ferry

ÉCOLE IPSSI PARIS BTC 25.1 - 2022

# Tables des matières

Projet 1 : Recherche de flags .....	2
1. Introduction : .....	2
2. Premier flag : .....	2
3. Deuxième flag : .....	4
4. Troisième flag : .....	5
5. Quatrième flag : .....	7
6. Cinquième flag : .....	7
7. Dernier flag : .....	8
8. Récapitulatif : .....	9
Projet 2 : Serveur Web Ubuntu .....	11
1. Introduction : .....	11
2. Installation de VirtualBox : .....	11
3. VM MASTER : .....	11
3.1. Création d'une VM MASTER et Installation de Ubuntu SERVER : .....	11
3.2. Installation Apache sur VM MASTER : .....	13
4. Le Load Balancing : .....	13
5. Création de VM NODE1 et NODE2, Installation Ubuntu SERVER et Installation Apache : .....	14
5.1 Rôle des serveurs : .....	14
5.2 Configuration paramètres réseaux des NODES : .....	14
5.3 Configuration des paramètres réseaux du MASTER : .....	16
5.4 Configuration des paramètres LB du MASTER : .....	16
5.5 Hébergement d'un Site sur les NODES : .....	17
5.6 Ajout du protocole HTTPS et SSL : .....	17

# Projet 1 : Recherche de flags

## 1. Introduction :

Le but de ce premier projet est de réussir à trouver six flags à l'aide de nos compétences se trouvant sur une machine dont l'adresse est <http://home.knl.im:1212>

Chaque flag que nous trouverons aura une structure semblable à celle-ci :  
btc{5446zfz6d2z1fz6fe684fd162f1}.

## 2. Premier flag :

Rendons-nous sur l'adresse de la machine à l'aide de notre navigateur WEB préféré.  
Nous atterrissons sur une page composée de deux champs, un login et un password.



Rappelons que généralement, il existe trois types de flags pouvant se trouver sur des serveurs :

- Les flags « SERVEUR » dans le code source de la page par exemple.
- Les flags utilisateurs dans le répertoire /home du serveur.
- Ainsi que les flags Root dans les répertoires systèmes (/home) ainsi qu'à la racine (/).

Essayons dans un premier temps d'observer le code source de cette page.

Avec l'aide de l'inspecteur de notre navigateur, nous pouvons observer le code source de cette page.

```
<!-- YnRjezYxMGMzOGY2NjkxZDM5ZTY2MTNlYjAxMmQ3MmE5ZmRhQ== -->
<!DOCTYPE html>
<html lang="en">
  <head>...</head>
  <body> == $0
    <div class="login-page">...</div>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/2.1.3/jquery.min.js" type="262354d38827fef26dd2baee-text/javascript"></script>
    <script src="./script.js" type="262354d38827fef26dd2baee-text/javascript"></script>
```

On observe en haut du code source à la première ligne une ligne verte comportant un premier hash :  
<!-- YnRjezYxMGMzOGY2NjkxZDM5ZTY2MTNlYjAxMmQ3MmE5ZmRhQ== -->.

Rappelons qu'un Hash est le produit d'une fonction mathématique à sens unique, plus simplement un algorithme. Globalement un hash c'est une suite de caractère générée aléatoirement par un algorithme défini comme le MD5.

Revenons à notre hash, on va devoir le « traduire » afin de pouvoir relever notre premier flag. Pour ce faire nous allons utiliser un outil magique appelé CyberChef.

Rendons-nous sur CyberChef à l'adresse suivante : <http://gchq.github.io/CyberChef> et mettons le hash dans la fenêtre Input. On observe deux égales à la fin du hash, ce qui est généralement une caractéristique de l'encodage Base64.

Mettons dans la fenêtre Recipe, FromBase64, pour indiquer à CyberChef que nous voulons traduire le hash depuis de l'encodage Base64. Nous obtenons comme résultat dans la fenêtre Output, btc{610c38f6691d39e6613eb012d72a9fda}.

The screenshot displays the CyberChef web application interface. At the top, there is a header with 'Download CyberChef', 'Last build: 18 days ago', and links for 'Options' and 'About / Support'. The main interface is divided into three panels: 'Operations', 'Recipe', and 'Input'. The 'Operations' panel on the left lists various tools like 'To Base64', 'From Base64', 'To Hex', etc. The 'Recipe' panel in the center shows a 'From Base64' recipe selected, with options for 'Alphabet' (A-Za-z0-9+/=) and 'non-alphabet chars' (checked). The 'Input' panel on the right contains the hash: `<!-- YnRjezYxMGZ0Y2NjZDM5ZTY2MTNlYjAxMmQ3MmESZmRhQ== -->`. Below the input, the 'Output' panel shows the result: `btc{610c38f6691d39e6613eb012d72a9fda}`. At the bottom, there is a 'BAKE!' button and an 'Auto Bake' checkbox.

Nous avons trouvé notre premier flag !

### 3. Deuxième flag :

Pour trouver notre deuxième flag, retournons sur la page [home.knl.im:1212](http://home.knl.im:1212)

En ouvrant de nouveau l'inspecteur, on observe une longue liste de caractères en vert.

```
.. <!--
4e 47 55 67 4e 44 51 67 4e 47 51 67 4e 6a 63 67 4e 47 55 67 4e 6d 51 67 4e 54 6b 67 4e 6a 63 67 4e 47
55 67 4e 6d 51 67 4e 54 55 67 4e 6a 63 67 4e 47 55 67 4e 6d 45 67 4e 6a 4d 67 4e 6a 63 67 4e 47 55 67
4e 32 45 67 4e 44 6b 67 4e 6a 63 67 4e 47 55 67 4e 6d 45 67 4e 44 55 67 4e 6a 63 67 4e 47 55 67 4e 32
45 67 4e 54 45 67 4e 6a 63 67 4e 47 55 67 4e 32 45 67 4e 54 55 67 4e 6a 63 67 4e 47 55 67 4e 6d 51 67
4e 47 51 67 4e 6a 63 67 4e 47 55 67 4e 6d 45 67 4e 44 55 67 4e 6a 63 67 4e 47 55 67 4e 32 45 67 4e 54
45 67 4e 6a 63 67 4e 47 55 67 4e 6d 45 67 4e 6d 49 67 4e 6a 63 67 4e 47 55 67 4e 6d 51 67 4e 54 6b 67
4e 6a 63 67 4e 47 55 67 4e 6d 51 67 4e 54 55 67 4e 6a 63 67 4e 47 55 67 4e 32 45 67 4e 47 51 67 4e 6a
63 67 4e 47 51 67 4e 6d 51 67 4e 47 51 67 4e 6a 63 67 4e 47 51 67 4e 6d 45 67 4e 44 45 67 4e 6a 63 67
4e 47 55 67 4e 6d 45 67 4e 6a 63 67 4e 6a 63 67 4e 47 55 67 4e 6d 45 67 4e 54 55 67 4e 6a 63 67 4e 47
55 67 4e 32 45 67 4e 44 6b 67 4e 6a 63 67 4e 47 55 67 4e 6d 45 67 4e 54 55 67 4e 6a 63 67 4e 47 51 67
4e 6d 45 67 4e 44 45 67 4e 6a 63 67 4e 47 55 67 4e 6d 45 67 4e 6d 49 67 4e 6a 63 67 4e 47 55 67 4e 32
45 67 4e 47 51 67 4e 6a 63 67 4e 47 51 67 4e 6d 45 67 4e 44 45 67 4e 6a 63 67 4e 47 55 67 4e 32 45 67
4e 6d 49 67 4e 6a 63 67 4e 47 55 67 4e 6d 51 67 4e 54 6b 67 4e 6a 63 67 4e 47 55 67 4e 32 45 67 4e 54
55 67 4e 6a 63 67 4e 47 55 67 4e 32 45 67 4e 44 6b 67 4e 6a 63 67 4e 47 51 67 4e 6d 45 67 4e 44 45 67
4e 6a 63 67 4e 47 55 67 4e 6d 51 67 4e 47 51 67 4e 6a 63 67 4e 47 55 67 4e 6d 51 67 4e 54 6b 67 4e 6a
63 67 4e 47 55 67 4e 6d 45 67 4e 6a 4d 67 4e 6a 63 67 4e 47 55 67 4e 6d 45 67 4e 6d 49 67 4e 6a 63 67
4e 47 55 67 4e 6d 51 67 4e 54 55 67 4e 6a 63 67 4e 47 51 67 4e 6d 45 67 4e 44 45 67 4e 6a 63 67 4e 47
55 67 4e 32 45 67 4e 54 45 67 4e 6a 63 67 4e 47 55 67 4e 6d 51 67 4e 54 6b 67 4e 6a 63 67 4e 47 51 67
4e 6d 45 67 4e 44 45 67 4e 6a 63 67 4e 47 55 67 4e 32 45 67 4e 54 45 67 4e 6a 63 67 4e 47 55 67 4e 6d
45 67 4e 6a 63 67 4e 6a 63 67 4e 47 55 67 4e 6d 45 67 4e 54 55 67 4e 6a 63 67 4e 47 51 67 4e 6d 45 67
4e 44 45 67 4e 6a 63 67 4e 47 55 67 4e 6d 45 67 4e 6a 63 67 4e 6a 63 67 4e 47 55 67 4e 6d 45 67 4e 44
55 67 4e 6a 63 67 4e 47 55 67 4e 6d 45 67 4e 47 51 67 4e 6a 63 67 4e 47 55 67 4e 6d 51 67 4e 44 6b 67
4e 6a 63 67 4e 47 55 67 4e 6d 45 67 4e 54 55 67 4e 6a 63 67 4e 47 55 67 4e 32 45 67 4e 44 6b 67 4e 6a
63 67 4e 47 55 67 4e 32 45 67 4e 47 51 67 4e 6a 63 67 4e 47 55 67 4e 32 45 67 4e 44 45 67 4e 6a 63 67
4e 47 55 67 4e 6d 45 67 4e 44 55 67 4e 6a 63 67 4e 47 55 67 4e 6d 45 67 4e 47 51 67 4e 6a 63 67 4e 47
55 67 4e 6d 45 67 4e 54 55 67 4e 6a 63 67 4e 47 51 67 4e 6d 45 67 4e 44 45 67 4e 6a 63 67 4e 47 55 67
4e 32 45 67 4e 44 45 67 4e 6a 63 67 4e 47 55 67 4e 6d 45 67 4e 44 55 67 4e 6a 63 67 4e 47 55 67 4e 6d
51 67 4e 54 55 67 4e 6a 63 67 4e 47 55 67 4e 6d 45 67 4e 54 55 67 4e 6a 63 67 4e 47 55 67 4e 6d 51 67
4e 47 51 67 4e 6a 63 67 4e 47 51 67 4d 7a 49 67 4e 44 55 67 4e 6a 63 67 4e 47 51 67 4e 44 63 67 4e 44
55 67 4e 6a 63 67 4e 47 55 67 4e 6d 45 67 4e 6a 63 67 4e 6a 63 67 4e 47 55 67 4e 6d 45 67 4e 44 55 67
4e 6a 63 67 4e 47 55 67 4e 6d 45 67 4e 47 51 67 4e 6a 63 67 4e 47 55 67 4e 6d 51 67 4e 44 6b 67 4e 6a
63 67 4e 47 55 67 4e 6d 45 67 4e 54 55 67 4e 6a 63 67 4e 47 55 67 4e 32 45 67 4e 44 6b 67 4e 6a 63 67
4e 47 51 67 4d 7a 49 67 4e 44 55 67 4e 6a 63 67 4e 47 51 67 4e 32 45 67 4e 54 6b 67 4e 6a 63 67 4e 47
51 67 4e 32 45 67 4e 54 6b 67 4e 6a 63 67 4e 47 51 67 4e 32 45 67 4e 54 45 67 4e 6a 63 67 4e 47 55 67
4e 54 51 67 4e 54 45 67 4e 6a 63 67 4e 47 55 67 4e 6d 45 67 4e 54 45 67 4e 6a 63 67 4e 47 55 67 4e 6d
```

Nous copions ce hash et le transposons dans CyberChef.

On observe de nombreux chiffres accompagnés de lettres, on remarque que cela ressemble fortement à de l'hexadécimal et y appliquons un « FromHex ». Nous obtenons dans Output une suite de caractères. En appliquant un « FromBase64 » dans Recipe, nous obtenons encore de l'hexadécimal. Nous réappliquons « FromHex » puis « FromBase64 » ainsi que pour finir « FromHex ». Nous obtenons finalement ceci :

Congratulations, here is your login to the hackerspace panel:

hacker:664Tdad4xJXW7Gm5m.

Nous avons ici un nom d'utilisateur suivi d'un mot de passe.

Essayons de le taper sur le site. Cela fonctionne, un texte apparait à l'écran indiquant :

```
Welcome to the hacker interface
btc{bbb9376c5d4df9d566790da9afd05be7}
```

Nous avons trouvé notre deuxième flag !

#### 4. Troisième flag :

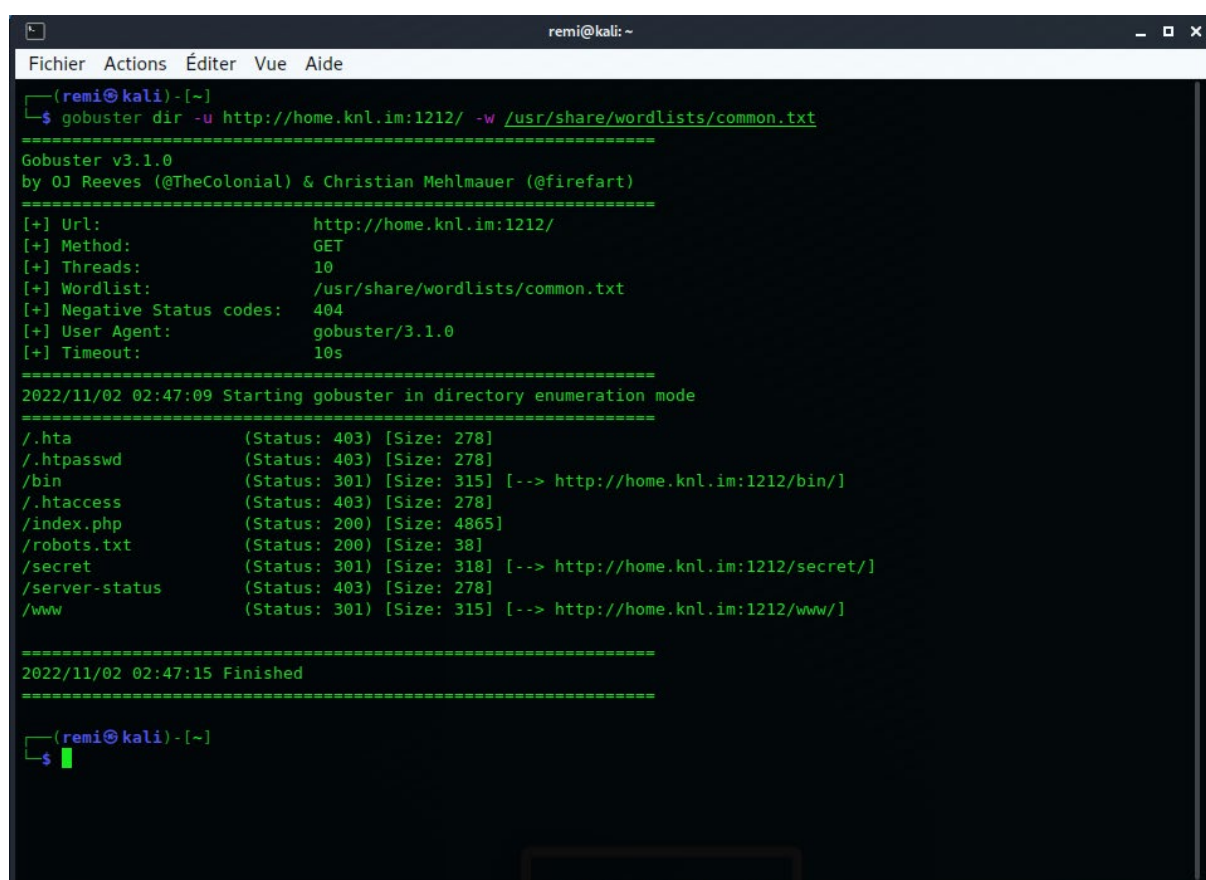
A présent nous passons à la partie qui nous mettra légèrement plus en difficulté.

Nous avons inspecté la page principale [home.knl.im:1212](http://home.knl.im:1212) ainsi que [home.knl.im:1212/login.php](http://home.knl.im:1212/login.php) mais nous n'avons rien de trouver de pertinent avec le code source des pages ainsi que les outils d'inspection du navigateur. Suite à cela, nous décidons de bruteforcer la page [home.knl.im:1212](http://home.knl.im:1212) pour voir si nous pouvons découvrir des fichiers inaccessibles de base.

Pour cela, nous lançons via notre terminal Linux un gobuster sur la page à l'aide de la commande :  
« gobuster dir -u <http://home.knl.im:1212/> -w /usr/share/wordlists/common.txt »

Ici l'argument dir indique que l'on recherche des dossiers sur l'URL <http://home.knl.im:1212/> à l'aide de la wordlist common.txt située dans le chemin /usr/share/wordlists.

Une wordlist comme son nom l'indique est une liste comportant des mots.



```
remi@kali: ~  
Fichier Actions Éditer Vue Aide  
remi@kali) ~  
$ gobuster dir -u http://home.knl.im:1212/ -w /usr/share/wordlists/common.txt  
=====
```

Gobuster v3.1.0  
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)

```
=====
```

[+] Url: http://home.knl.im:1212/  
[+] Method: GET  
[+] Threads: 10  
[+] Wordlist: /usr/share/wordlists/common.txt  
[+] Negative Status codes: 404  
[+] User Agent: gobuster/3.1.0  
[+] Timeout: 10s

```
=====
```

2022/11/02 02:47:09 Starting gobuster in directory enumeration mode

```
=====
```

File	Status	Size	Notes
/.hta	403	278	
/.htpasswd	403	278	
/bin	301	315	http://home.knl.im:1212/bin/
/.htaccess	403	278	
/index.php	200	4865	
/robots.txt	200	38	
/secret	301	318	http://home.knl.im:1212/secret/
/server-status	403	278	
/www	301	315	http://home.knl.im:1212/www/

```
=====
```

2022/11/02 02:47:15 Finished

```
=====
```

remi@kali) ~  
\$

A cette étape-là du projet, nous avons perdu pas mal de temps car nous nous étions trompé de wordlists. En effet, nous avons utilisé la wordlist rockyou.txt qui était prévue initialement pour des mots de passe. Or nous recherchons des noms de fichiers communs dans un serveur et non pas des mots de passe.

Pour en revenir au Gobuster, nous avons trouvé plusieurs fichiers sur le serveur dont notamment des pages nous intéressant sur lesquelles nous allons « naviguer » :

bin, index.php, robots.txt, secret, server-status et www.

Nous décidons de nous rendre sur la page secret car forcément il s'agit de celle qui nous attire le plus. Nous tombons sur une page blanche avec un POP-UP nous indiquant de trouver le flag caché. Nous ouvrons un inspecteur et ne trouvons rien dans celui-ci. Nous passons à la console et nous y trouvons un nouveau hash.

```
YnRje2NkMDBiZDIzZDMYn2Y2NGNjMDEyZmM2MGRkNWE2NWNhfQ==
```

Nous le transposons dans CyberChef, nous remarquons les deux égales, signes typiques d'un encodage en Base64. Nous appliquons « FromBase64 » dans Recipe et obtenons dans Output le troisième Flag : btc{cd00bd23d327f64cc012fc60dd5a65ca}

The screenshot displays the CyberChef interface. On the left, the 'Recipe' panel shows a 'From Base64' recipe selected. The 'Alphabet' dropdown is set to 'A-Za-z0-9+/', and the 'non-alphabet chars' checkbox is checked. On the right, the 'Input' panel contains the Base64 string 'YnRje2NkMDBiZDIzZDMYn2Y2NGNjMDEyZmM2MGRkNWE2NWNhfQ=='. The 'Output' panel at the bottom shows the decoded result: 'btc{cd00bd23d327f64cc012fc60dd5a65ca}'. Metadata for the input shows a length of 52 and 1 line, while the output shows a length of 37 and 1 line, with a processing time of 2ms.

## 5. Quatrième flag :

Après avoir passé la page secret au peigne fin, nous décidons de passer à une autre page, robots.txt. Communément, le fichier Robots.txt est un fichier texte permettant d'indiquer aux robots des moteurs de recherche les accès qui leur sont autorisés. Nous nous rendons sur cette page et nous avons directement notre quatrième flag : `btc{4b85f92a522361010f8850e692fc4aa0}`

```
btc{4b85f92a522361010f8850e692fc4aa0}
```

## 6. Cinquième flag :

Nous inspectons robots.txt mais n'y relevons rien de pertinent. Nous naviguons vers la page [home.knl.im:1212/index.php](http://home.knl.im:1212/index.php) et nous remarquons qu'il s'agit de la page principale du site, la première sur laquelle nous étions arrivés lorsque nous avons taper l'adresse [home.knl.im:1212](http://home.knl.im:1212) dans la barre d'URL. Nous y retrouvons bien nos deux premiers flags mais rien de plus.

Nous passons à une autre page, [home.knl.im:1212/server-status](http://home.knl.im:1212/server-status)

Nous n'avons malheureusement pas accès à la page car nous n'avons pas les permissions nécessaires et nous ne pouvons rien faire de plus avec cette page.

Passons à la page [home.knl.im:1212/www](http://home.knl.im:1212/www)

Nous arrivons sur une page internet comportant un titre ainsi que trois liens vers trois pages.

# My awesome website

[Page 1](#) [Page 2](#) [Page 3](#)

Nous allons sur la première page et l'inspectons mais nous ne trouvons rien d'intéressant.

Nous passons à la deuxième page. A la lecture du texte sur cette page, nous n'avons rien d'intéressant mais en ouvrant l'inspecteur nous remarquons un cinquième flag directement !  
`btc{88dabf2f31578458d1060cd1c7db3089}`

```
viverra suspendisse. Potenti nullam ac tortor vitae pul  
quis risus sed vulputate odio. Tincidunt ornare massa e  
vitae suscipit tellus mauris a. "  
<!-- btc{88dabf2f31578458d1060cd1c7db3089} -->  
" Scelerisque felis imperdiet proin fermentum leo vel d  
Fermentum iaculis eu non diam phasellus vestibulum lore  
tincidunt eget nullam non nisi. Varius morbi enim nunc
```



## 7. Dernier flag :

Recherchons le dernier flag. Nous regardons et inspectons le reste de la page 2 dans www ainsi que la page 3 mais nous n'y trouvons rien. Il ne nous reste qu'une seule page où naviguer :

[home.knl.im:1212/bin](http://home.knl.im:1212/bin)

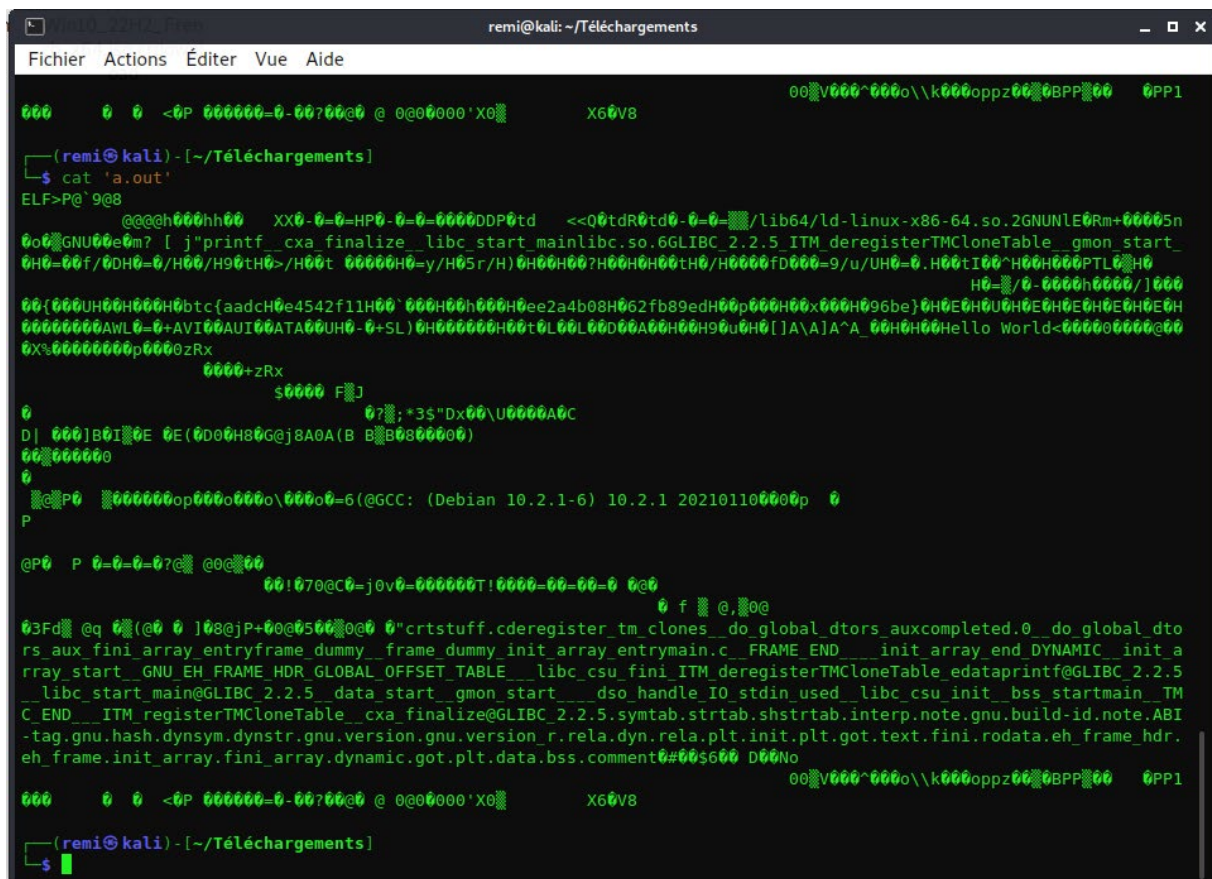
Nous arrivons sur un Index, c'est-à-dire une interface avec une table regroupant les fichiers d'un répertoire d'un serveur. En inspectant cette page nous ne remarquons rien de pertinent.

## Index of /bin

Name	Last modified	Size	Description
<a href="#">Parent Directory</a>	-		
<a href="#">a.out</a>	2022-10-23 10:09 16K		

Apache/2.4.54 (Debian) Server at home.knl.im Port 1212

En cliquant sur « Parent Directory » nous retombons sur la page principale de connexion du site. Nous téléchargeons donc le fichier a.out dans le répertoire Téléchargements et l'ouvrons.



```
remi@kali: ~/Téléchargements
Fichier Actions Éditer Vue Aide

000  0 0 <P 000000=0-00?00@ @ 0@0000'X0  X60V8  00V000~000o\\k000oppz00BPP00 0PP1

(remi@kali) - [~/Téléchargements]
$ cat 'a.out'
ELF>P@`908
@@@h00hh00 XX0-0=0=HP0-0=0=000DDP0td <<Q0tdR0td0-0=0=00/lib64/ld-linux-x86-64.so.2GNULE0Rm+00005n
000GNU00e0m? [ j"printf_cxa_finalize_libc_start_mainlibc.so.6GLIBC 2.2.5 ITM_deregisterTMCloneTable_gmon_start_
H0=00f/0DH0=0/H00/H90tH0>/H00t 0000H0=y/H05r/H)0H00H00?H00H0H00tH0/H0000fD000=9/u/UH0=0.H00tI00^H00H000PTL0H0
H0=0/0-0000h0000/]000
00{000UH00H000H0bt{aadcH0e4542f11H00`000H00h000H0ee2a4b08H062fb89edH00p000H00x000H096be}0H0E0H0U0H0E0H0E0H0E0H0E0H
0000000AWL0=0+AVI00AUI00ATA00UH0-0+SL)0H000000H00t0L00L00D00A0H00H90u0H0[]A\A]A^A_00H0H00Hello World<000000000@00
0X%0000000p0000zRx
0000+zRx
$0000 F0J
0?;*3$"Dx00\U0000A0C
D| 000JB0I00E 0E(0D00H80G@j8A0A(B B0080000)
00000000
0
0cP0 000000op000o000o\000o0=6(@GCC: (Debian 10.2.1-6) 10.2.1 202101100000p 0
P
@P0 P 0=0=0?@ @0@00
00!070@C0=j0v0=000000T!0000=00=00 0@0
0 f @,00@
03Fd @q 00@0 0 J08@jP+00@050000@00 0"crtstuff.cderegister_tm_clones__do_global_dtors_auxcompleted.0__do_global_dto
rs_aux_fini_array_entryframe_dummy__frame_dummy_init_array_entrymain.c__FRAME_END____init_array_end_DYNAMIC__init_a
rray_start__GNU_EH_FRAME_HDR_GLOBAL_OFFSET_TABLE__libc_csu_fini ITM_deregisterTMCloneTable_edataprntf@GLIBC 2.2.5
__libc_start_main@GLIBC 2.2.5__data_start__gmon_start__dso_handle_IO_stdin_used__libc_csu_init__bss_startmain__TM
C_END__ITM_registerTMCloneTable__cxa_finalize@GLIBC 2.2.5.symtab.strtab.shstrtab.interp.note.gnu.build-id.note.ABI
-tag.gnu.hash.dynsym.dynstr.gnu.version.gnu.version_r.rela.dyn.rela.plt.init.plt.got.text.fini.rodata.eh_frame_hdr.
eh_frame.init_array.fini_array.dynamic.got.plt.data.bss.comment0#00$000 D00No
00V000~000o\\k000oppz00BPP00 0PP1
000  0 0 <P 000000=0-00?00@ @ 0@0000'X0  X60V8

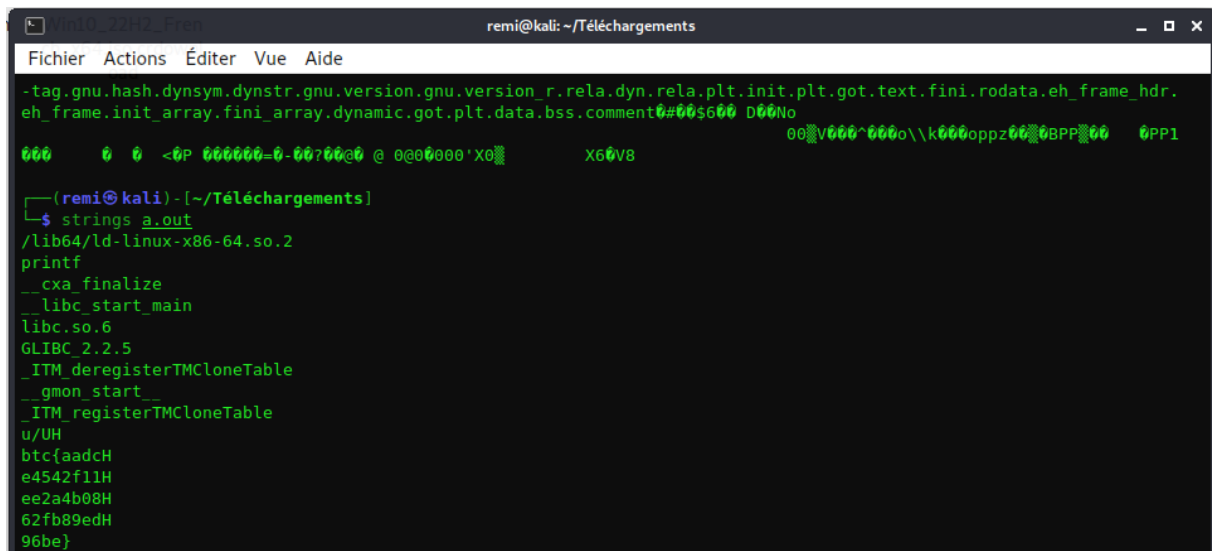
(remi@kali) - [~/Téléchargements]
$
```

On ouvre un terminal et on se place dans le répertoire de téléchargements ici « /home/remi/Téléchargements » à l'aide de la commande « cd /home/remi/Téléchargements ».

On affiche le fichier avec la commande cat : « cat a.out ».

On a ici un fichier comportant des symboles quelconques ainsi que du texte. Il s'agit d'un fichier binaire. Nous rappelons que la commande « strings » permet d'afficher uniquement les chaînes

de caractères « lisibles par l'homme » d'un fichier. Nous l'appliquons à l'aide de la commande « strings a.out ».



```
remi@kali: ~/Téléchargements
Fichier Actions Éditer Vue Aide
- tag.gnu.hash.dynsym.dynstr.gnu.version.gnu.version_r.rela.dyn.rela.plt.init.plt.got.text.fini.rodata.eh_frame_hdr.
eh_frame.init_array.fini_array.dynamic.got.plt.data.bss.comment0#00$600 D00No
000 0 0 <0P 000000=0-00?00@0 @ 000000'X0  X60V8 00V000~000o\\k000oppz0000BPP00 0PP1

(remi@kali) - [~/Téléchargements]
$ strings a.out
/lib64/ld-linux-x86-64.so.2
printf
__cxa_finalize
__libc_start_main
libc.so.6
GLIBC_2.2.5
_ITM_deregisterTMCloneTable
__gmon_start__
_ITM_registerTMCloneTable
u/UH
btc{aadCHe
e4542f11H
ee2a4b08H
62fb89edH
96be}
```

Nous y voyons un petit peu plus clair. Nous remarquons directement le dernier flag sur les cinq dernières lignes de notre Capture : [btc{aadCHe4542f11Hee2a4b08H62fb89edH96be}](#)

Nous venons de trouver nos six flags !

## 8. Récapitulatif :

### Six flags trouvés :

- [btc{bbb9376c5d4df9d566790da9afd05be7}](#)
- [btc{610c38f6691d39e6613eb012d72a9fda}](#)
- [btc{cd00bd23d327f64cc012fc60dd5a65ca}](#)
- [btc{4b85f92a522361010f8850e692fc4aa0}](#)
- [btc{88dabf2f31578458d1060cd1c7db3089}](#)
- [btc{aadCHe4542f11Hee2a4b08H62fb89edH96be}](#)

### Outils utilisés :

- Kali Linux avec un terminal, gobuster.
- Un navigateur WEB, ici Mozilla Firefox avec un inspecteur.
- Le site internet CyberChef : <http://gchq.github.io/CyberChef>

### Commandes utilisées :

- gobuster dir -u <URL> -w <Wordlists PATH> : permet de bruteforcer une URL avec une wordlist.
- cd <PATH> : permet de se rendre dans un chemin dans le terminal.
- cat <file> : permet d'afficher le contenu « textuel » d'un fichier.
- strings <file> : permet d'afficher les chaînes de caractères d'un fichier.



# Projet 2 : Serveur Web Ubuntu

## 1. Introduction :

L'objectif de ce deuxième projet est de mettre en place un serveur web sous linux (ici Ubuntu) utilisant les protocoles HTTPS et TLS. Nous devons héberger une page d'accueil qui sera affichée sur un navigateur client.

Voyons les différents outils nécessaires :

- Un PC
- VirtualBox d'installé
- Un .iso de Ubuntu SERVER
- Apache2
- OpenSSL
- Un éditeur de texte

## 2. Installation de VirtualBox :

VirtualBox est un logiciel gratuit permettant faire de la virtualisation et ainsi créer des machines virtuelles. On peut ainsi avoir d'autres ordinateurs (virtuels) avec d'autres OS (Operating System) et garder son OS principal. VirtualBox est disponible sous Windows, Mac OS X et Linux.

Pour le télécharger il nous suffit d'aller sur ce lien : <https://www.virtualbox.org/wiki/Downloads> et de sélectionner la version correspondant à son OS. Ensuite, nous lançons l'installation et démarrons le logiciel.

## 3. VM MASTER :

### 3.1. Création d'une VM MASTER et Installation de Ubuntu SERVER :

Nous souhaitons faire du Load Balancing (nous expliquerons ce concept plus bas), ainsi il nous faut au minimum trois machines virtuelles : une machine maître (Serveur MASTER) et deux machines esclaves, dites nœuds (Serveurs NODE1 et NODE2).

Nous créons dans un premier temps notre VM Master.

Nous téléchargeons un .iso de Ubuntu SERVER : <https://releases.ubuntu.com/22.04.1/ubuntu-22.04.1-live-server-amd64.iso>

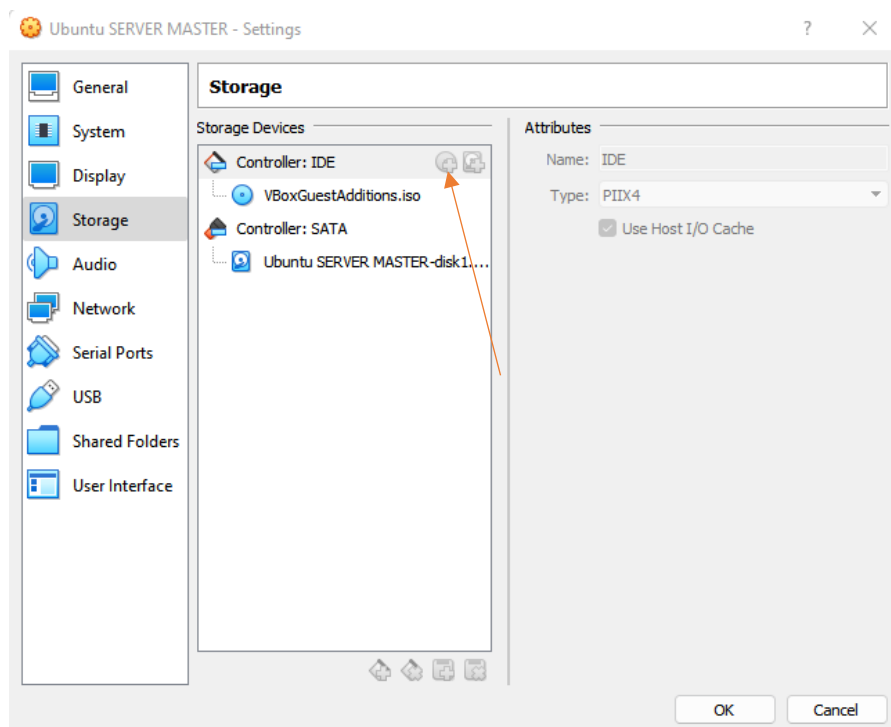


Dans VirtualBox, appuyer sur « NEW », effectuer la configuration de la VM en insérant l'iso. Mettre à minima 1024mb de RAM.

Dans « Settings » puis « System » puis « Processor », mettre à minima 2 CPUs ou l'installation risque de ne pas aboutir (kernel Panic !). La VM ne voulant pas se lancer sans 2 CPUs minimum.

Lancer la VM avec le bouton « Start ». Insérer l'iso d'Ubuntu SERVER téléchargé précédemment dans la fenêtre apparente. Si aucune fenêtre n'apparaît au démarrage, quitter la VM, aller dans

« Settings » puis « Storage », aller dans l'onglet Storage Devices et appuyer sur le petit disque et ajouter l'iso.



Une fois la VM démarrée, nous suivons les instructions d'installation.

Nous veillerons bien à effectuer le cryptage du disque pour des raisons de sécurité.

Une fois cela fait, une fenêtre apparaît demandant des informations :

- Notre nom
- Nom de la machine : nom de la machine sur le réseau et dans le terminal.
- Nom d'utilisateur : username, nom d'utilisateur que l'on utilisera dans le terminal.
- Mot de passe : mot de passe accompagnant l'username dans le terminal.
- Confirmation du mot de passe

```
Installation du système [ Help ]

subiquity/Userdata/apply_autoinstall_config
subiquity/Package/apply_autoinstall_config
subiquity/Debian/apply_autoinstall_config
subiquity/Kernel/apply_autoinstall_config
subiquity/2dev/apply_autoinstall_config
subiquity/Late/apply_autoinstall_config
configuring apt
curtin command in-target
installing system
curtin command install
preparing for installation
configuring storage
  running 'curtin block-meta simple'
  curtin command block-meta
  removing previous storage devices
  configuring disk: disk-sda
  configuring partition: partition-0
  configuring partition: partition-1
  configuring format: format-0
  configuring partition: partition-2
  configuring lvm_volgroup: lvm_volgroup-0
  configuring lvm_partition: lvm_partition-0
  configuring format: format-1
  configuring mount: mount-1
  configuring mount: mount-0
writing install sources to disk
running 'curtin extract'
curtin command extract
  acquiring and extracting image from cp:///tmp/tmp9h_o11kj/mount \

[ View full log ]
```

Nous remplissons ces informations puis nous poursuivons l'installation.

Une fois l'installation terminée, appuyer sur « Restart Now ».

### 3.2. Installation Apache sur VM MASTER :

Une fois que la VM est redémarrée, il faut se connecter dans le terminal à l'aide de son nom d'utilisateur et de son mot de passe. Une fois ceci fait, nous allons procéder à l'installation d'Apache.

Mais à quoi sert Apache dans un premier temps ?


Apache est un service qui permet de faire un serveur HTTP notamment sous Linux.

On peut procéder à une installation simple d'Apache via les paquets APT ou nous pouvons également l'installer avec LAMP.

Nous installons Apache sur notre serveur Maître.

Dans notre terminal,

```
testmachine@testmachine:~$
```



Nous tapons "sudo apt install apache2", nous validons l'installation en appuyant sur la touche O quand on y est invité. Ensuite nous pouvons ensuite éteindre notre VM avec la commande « shutdown 0 ». Nous reviendrons sur notre VM MASTER plus tard, occupons-nous à présent des VM NODES.

### 4. Le Load Balancing :

Pour rappel, nous voulons mettre en place un système de Load Balancing.

C'est-à-dire un système capable de répartir différents utilisateurs sur différents serveurs en fonction de plusieurs facteurs. Cela permet d'optimiser en général la latence ainsi que le débit mais également de pouvoir accueillir un plus grand nombre d'utilisateur tout en gardant une expérience de navigation agréable.

Notre Load Balancing sera composé d'un serveur maître accompagné de deux serveurs esclaves.

Il va s'agir du Load Balancing le plus simple puisqu'il s'agit d'un Load Balancing by Requests. C'est-à-dire qu'à chaque requête envoyée par un navigateur vers le serveur maître, ce dernier va répartir le client sur un serveur différent (ici alterne un serveur sur deux).

## 5. Création de VM NODE1 et NODE2, Installation Ubuntu SERVER et Installation Apache :

Comme dit précédemment, nous allons faire un LB. Ceci nécessite un serveur maître déjà installé ainsi que deux serveurs esclaves. Configurons le premier serveur esclave, que nous appellerons node1.

Création VM NODE 1 et VM NODE 2 ainsi qu'Installation Ubuntu SERVER : voir 3.1, mêmes étapes.  
Installation Apache : voir 3.2, mêmes étapes.

### 5.1 Rôle des serveurs :

Le rôle du serveur maître va être de recueillir le client et de le diriger vers le serveur esclave le plus adéquat. Le serveur maître joue ici un rôle de Reverse Proxy. De fait, il faut que nos deux nœuds (serveurs esclaves) communiquent avec le master (serveur maître). Voyons comment faire cela.

### 5.2 Configuration paramètres réseaux des NODES :

Nous allons démarrer nos deux nodes, nous connecter et vérifier qu'Apache est bien démarré avec la commande « `sudo systemctl status apache2` » en vérifiant bien qu'il y ait écrit active.

Nous avons eu quelques soucis avec la commande ci-dessus car nous avons mis comme nom d'unité « Apache » et non pas « Apache2 ».

Nous voulons faire des serveurs locaux pour le moment. Pour cela, nous allons dans les paramètres de la VM puis dans « network » et nous mettons l'adapter 1 de « NAT » à « Host-only Adapter ». On retourne dans la VM et l'on tape la commande « `ip a` ». Nous prenons l'IP de la deuxième carte réseaux commençant par 192.168. Nous tapons cette IP dans notre navigateur en local et nous devrions avoir une page comme ci-après.

```
Last login: Wed Nov  2 04:56:53 UTC 2022 on tty1
testmachine@testmachine:~$ sudo systemctl status apache2
[sudo] password for testmachine:
• apache2.service - The Apache HTTP Server
   Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2022-11-02 05:06:53 UTC; 19min ago
     Docs: https://httpd.apache.org/docs/2.4/
   Process: 615 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
  Main PID: 695 (apache2)
    Tasks: 55 (limit: 1030)
   Memory: 7.6M
      CPU: 175ms
   CGroup: /system.slice/apache2.service
           └─695 /usr/sbin/apache2 -k start
             └─699 /usr/sbin/apache2 -k start
               └─701 /usr/sbin/apache2 -k start
```





# Ubuntu

## Apache2 Default Page

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

### Configuration Overview

Ubuntu's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Ubuntu tools. The configuration system is **fully documented in `/usr/share/doc/apache2/README.Debian.gz`**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the `apache2-doc` package was installed on this server.

The configuration layout for an Apache2 web server installation on Ubuntu systems is as follows:

```
/etc/apache2/
|-- apache2.conf
|   |-- ports.conf
|-- mods-enabled
|   |-- *.load
|   |-- *.conf
|-- conf-enabled
|   |-- *.conf
|-- sites-enabled
|   |-- *.conf
```

- `apache2.conf` is the main configuration file. It puts the pieces together by including all remaining configuration files when starting up the web server.
- `ports.conf` is always included from the main configuration file. It is used to determine the listening ports for incoming connections, and this file can be customized anytime.
- Configuration files in the `mods-enabled/`, `conf-enabled/` and `sites-enabled/` directories contain particular configuration snippets which manage modules, global configuration fragments, or virtual host configurations, respectively.
- They are activated by symlinking available configuration files from their respective `*-available/` counterparts. These should be managed by using our helpers `a2enmod`, `a2dismod`, `a2ensite`, `a2dissite`, and `a2enconf`, `a2disconf`. See their respective man pages for detailed information.
- The binary is called `apache2` and is managed using `systemd`, so to start/stop the service use `systemctl start apache2` and `systemctl stop apache2`, and use `systemctl status apache2` and `journalctl -u apache2` to check status. `system` and `apache2ctl` can also be used for service management if desired. **Calling `/usr/bin/apache2` directly will not work** with the default configuration.

### Document Roots

By default, Ubuntu does not allow access through the web browser to *any* file outside of those located in `/var/www`, **public\_html** directories (when enabled) and `/usr/share` (for web applications). If your site is using a web document root located elsewhere (such as in `/srv`) you may need to whitelist your document root directory in `/etc/apache2/apache2.conf`.

The default Ubuntu document root is `/var/www/html`. You can make your own virtual hosts under `/var/www`.

### Reporting Problems

Please use the `ubuntu-bug` tool to report bugs in the Apache2 package with Ubuntu. However, check **existing bug reports** before reporting a new bug.

Please report bugs specific to modules (such as PHP and others) to their respective packages, not to the web server itself.

Notre serveur WEB du node1 est bien en ligne. Répétons l'étape 5.2 pour le deuxième node.



### 5.3 Configuration des paramètres réseaux du MASTER :

Nous démarrons notre master, nous connectons et nous vérifions bien qu'Apache est bien lancé avec la commande « `sudo systemctl status apache2` ». Nous allons dans les paramètres de la VM puis dans « Network » et dans l'Adapter 1 nous mettons « NAT » sur « Host-Only Adapter ».

Cela devrait aisé nous l'avons précédemment fait.

### 5.4 Configuration des paramètres LB du MASTER :

Nous allons à présent configurer le load balancing sur notre master.

Pour se faire nous avons besoin d'activer un proxy à l'aide de la commande « `sudo a2enmod proxy` ».

Un proxy est pour rappel un intermédiaire entre de machine qui sert de relais.

Ensuite nous ajoutons le protocole http au proxy avec la commande :

« `sudo a2enmod proxy_http` »

Puis nous ajoutons le load balancer au proxy :

« `sudo a2enmod proxy_balancer` »

Nous précisons le type de load balancing, ici by requests :

« `sudo a2enmod lbmethod_byrequests` »

Enfin nous redémarrons Apache avec la commande « `sudo systemctl restart apache2` ».

Ensuite nous allons dans le répertoire `/etc/apache2/sites-enabled` avec la commande « `cd <PATH>` » et nous éditons `000-default.conf` comme ci-dessous qui est un fichier de configuration à l'aide de la commande « `sudo nano 000-default.conf` ».

```
GNU nano 6.2                                000-default.conf *
<VirtualHost *:80>
    ServerName TEST.COM
    ServerAlias test.com
    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/html

    # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
    # error, crit, alert, emerg.
    # It is also possible to configure the loglevel for particular
    # modules, e.g.
    #LogLevel info ssl:warn

    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined

    <Proxy balancer://monproxy>
        BalancerMember http://ipNODE1
        BalancerMember http://ipNODE2
    </Proxy>
    ProxyPreserveHost On
    ProxyPass / balancer://monproxy
    ProxyPassReverse / balancer://monproxy
    # For most configuration files from conf-available/, which are
    # enabled or disabled at a global level, it is possible to
    # include a line for only one particular virtual host. For example the
    # following line enables the CGI configuration for this host only
    # after it has been globally disabled with "a2disconf".
    #Include conf-available/serve-cgi-bin.conf
</VirtualHost>

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
```

## 5.5 Hébergement d'un Site sur les NODES :

Notre Load Balancing est prêt !

A présent hébergeons notre site sur nos nodes.

Les étapes seront identiques sur les nodes 1 et 2.

Il vous suffit de mettre vos fichiers HTML dans le dossier /var/www/html et ils seront disponibles !

Si vous souhaitez remplacer la page par défaut, supprimer la page « index.html » avec la commande « `sudo rm -r index.html` » et renommer la page souhaitée en index.html avec la commande « `sudo mv <nom.html> index.html` ».

## 5.6 Ajout du protocole HTTPS et SSL :

Nous devons ajouter les protocoles HTTPS et SSL uniquement sur le serveur MASTER car c'est celui-ci qui sera connecté directement avec le client et non pas les NODES qui seront donc connectés uniquement au serveur MASTER.

Pour ce faire, nous allons utiliser le terminal du serveur MASTER.

Commençons par activer le mode de proxy ssl via la commande « `sudo a2enmod ssl` »

Une fois cela fait, nous devons redémarrer le service apache2 avec la commande « `sudo systemctl restart apache2` ».

Ensuite, il va falloir créer un certificat auto-signé.

Nous allons utiliser openssl pour générer un certificat auto-signé.

Dans un terminal, nous tapons la commande suivante :

« `sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/ssl/private/server.test-celebrance.com.key -out /etc/ssl/certs/server.test-celebrance.com.crt` » .

-newkey rsa:4096 : Créer une clé RSA en 4096 bit pour l'utiliser avec le certificat. RSA 2048 est l'option par défaut pour la plupart des versions d'OpenSSL. Pour être sûr du degré de chiffrement, il faut le spécifier durant la configuration.

-x509 : Créer un certificat auto-signé.

-sha256 : Générer une requête de certificat en chiffrement en 256-bit SHA (Secure Hash Algorithm).

-days : Détermine la durée de validité du certificat en jours.

-nodes : Créer un certificat ne nécessitant pas de passphrase.

On doit par la suite répondre à plusieurs questions. Il faut suivre ce qu'il y a à l'écran.

Ensuite nous devons éditer le fichier de configuration du master dans le chemin /etc/apache2/sites-available et nous devons ajouter ces trois lignes :

SSLEngine on

SSLCertificateKeyFile /etc/ssl/private/server.test-celebrance.com.key

SSLCertificateFile /etc/ssl/certs/server.test-celebrance.com.crt

Nous sauvegardons le fichier et nous lançons la commande « `sudo a2ensite test-celebrance.com-ssl.conf` » Pour lancer une version SSL de notre site. On redémarre ensuite notre apache2 avec la commande « `sudo systemctl restart apache2` ».

Pour accéder à votre server MASTER, il suffit de taper son nom de domaine ou son adresse IP.

Notre site :

<https://test-celebrance.com>

192.168.56.104