

Documentation technique

Création de 2 serveurs Linux pouvant se synchroniser automatiquement toutes les 5 minutes et accessibles seulement depuis un serveur reverse proxy avec NGINX.

Table des matières

1. Définitions
2. Prérequis
3. Etape 1 : Paramétrer une machine virtuelle
4. Etape 2 : Installation du serveur
5. Etape 3 : Installer le service Docker
6. Etape 4 : Créer les conteneurs et les démarrer
7. Etape 5 : Configuration du lien SSH entre les 2 serveurs web et synchronisation automatique des 2 serveurs
8. Etape 6 : Installer et paramétrer le reverse proxy
9. Etape 7 : Vérification du reverse proxy

1. Définitions

Linux : Linux est un système d'exploitation de type Unix. Il a été conçu pour équiper les ordinateurs personnels d'un système d'exploitation gratuit ou à très faible coût, comparable aux versions Unix classiques, généralement plus coûteuses. (source : <https://www.lemagit.fr/definition/Linux>)

Ubuntu : est un système d'exploitation GNU/Linux basé sur la distribution Debian. Il est libre, gratuit, et simple d'utilisation. (source : <https://www.ubuntu-fr.org/>)

Serveur Proxy : Un serveur proxy joue le rôle de passerelle entre Internet et l'utilisateur. C'est un serveur intermédiaire qui sépare les utilisateurs, des sites Web sur lesquels ils naviguent. Les serveurs proxy assurent différents niveaux de fonctionnalité, de sécurité et de confidentialité, selon votre type d'utilisation, vos besoins ou la politique de votre entreprise. (source : <https://www.varonis.com/fr/blog/serveur-proxy>)

Reverse Proxy : Un proxy inverse ou serveur mandataire inverse est un type de serveur, habituellement placé en frontal de serveurs web. Contrairement au serveur proxy qui permet à un utilisateur d'accéder au réseau Internet, le proxy inverse permet à un utilisateur d'Internet d'accéder à des serveurs internes. (source : https://fr.wikipedia.org/wiki/Proxy_inverse)

Conteneur : Un conteneur est un environnement d'exécution léger et une alternative aux méthodes de virtualisation traditionnelles basées sur les machines virtuelles. Un conteneur permet de rendre un logiciel modulaire, portable et standardisé afin qu'il puisse être facilement déployé sur n'importe quel environnement informatique. Les conteneurs sont conçus pour contenir le code d'une application ainsi que toutes ses dépendances, de manière à ce que tous les éléments nécessaires pour exécuter l'application se trouvent à un seul endroit. (source : https://www.splunk.com/fr_fr/data-insider/what-is-a-container.html)

Docker : Docker est un système d'exploitation pour conteneurs. Docker permet d'automatiser le déploiement des applications au sein d'un environnement de conteneurs. Grâce à ces divers outils, les utilisateurs profitent d'un accès complet aux applications et sont en mesure d'accélérer le déploiement, de contrôler les versions et de les attribuer. (source : <https://datascientest.com/docker-guide-complet>)

Load-balancing : Le Load Balancing consiste à répartir efficacement le trafic réseau entrant sur un groupe de serveurs, également connu sous le nom de parc de serveurs ou de pool de serveurs. (source : <https://actualiteinformatique.fr/cloud/definition-load-balancing>)

NGINX : NGINX est un serveur web open-source qui, depuis son succès initial en tant que serveur web, est maintenant aussi utilisé comme reverse proxy, cache HTTP, et load balancer. (source : <https://kinsta.com/fr/base-de-connaissances/qu-est-ce-que-nginx/>)

2. Prérequis

Pour créer nos serveurs Linux qui peuvent se synchroniser, nous avons besoin d'un logiciel de virtualisation, type VirtualBox ou VmWare Workstation.

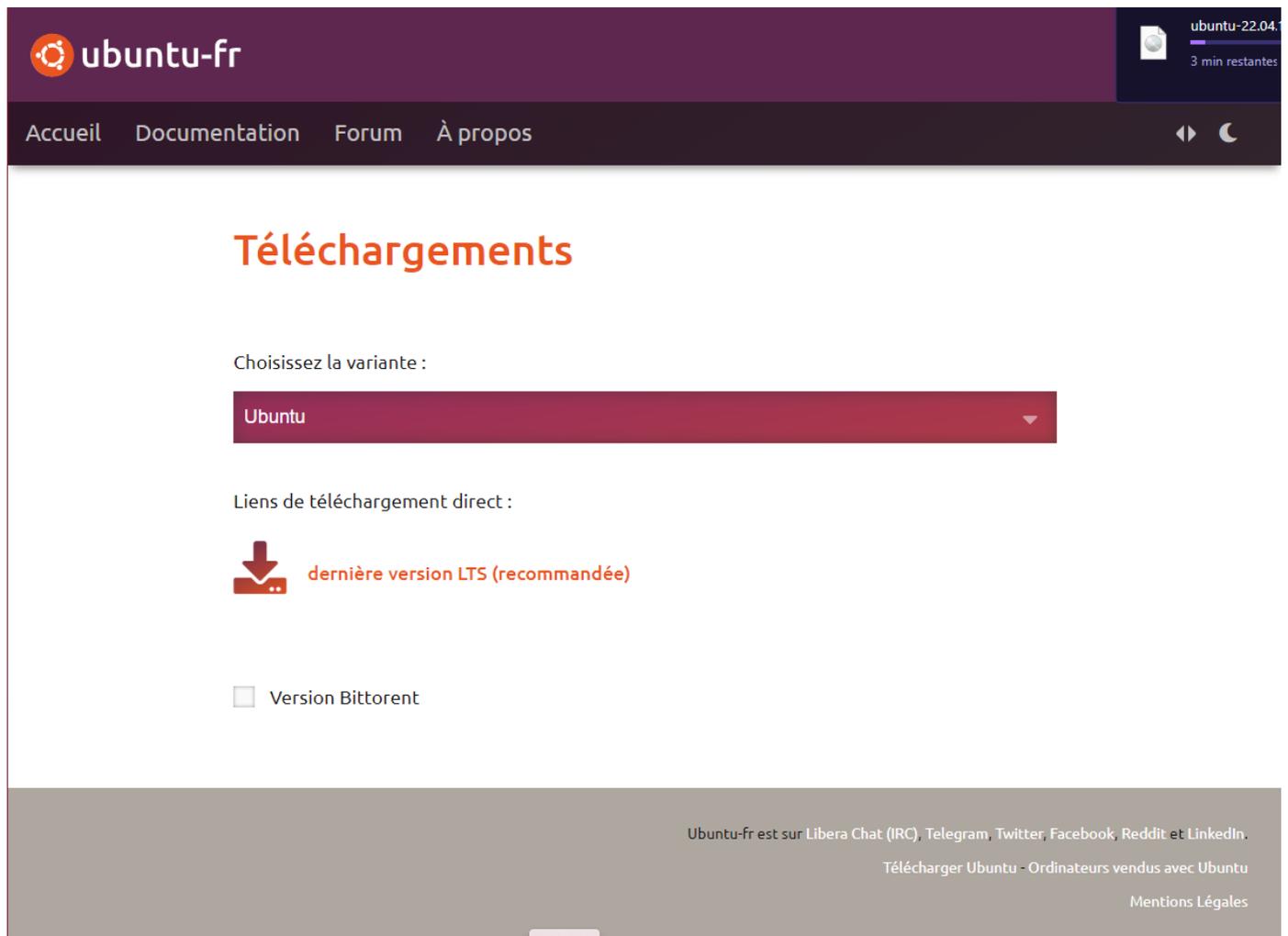
Pour fonctionner, prévoir d'allouer à minima pour chaque machine virtuelle :

- 2 Gb de mémoire
- 1 Processeur
- 20 Gb pour le disque dur

Votre PC doit disposer d'autre moins 2 Go de RAM et d'une carte réseau.

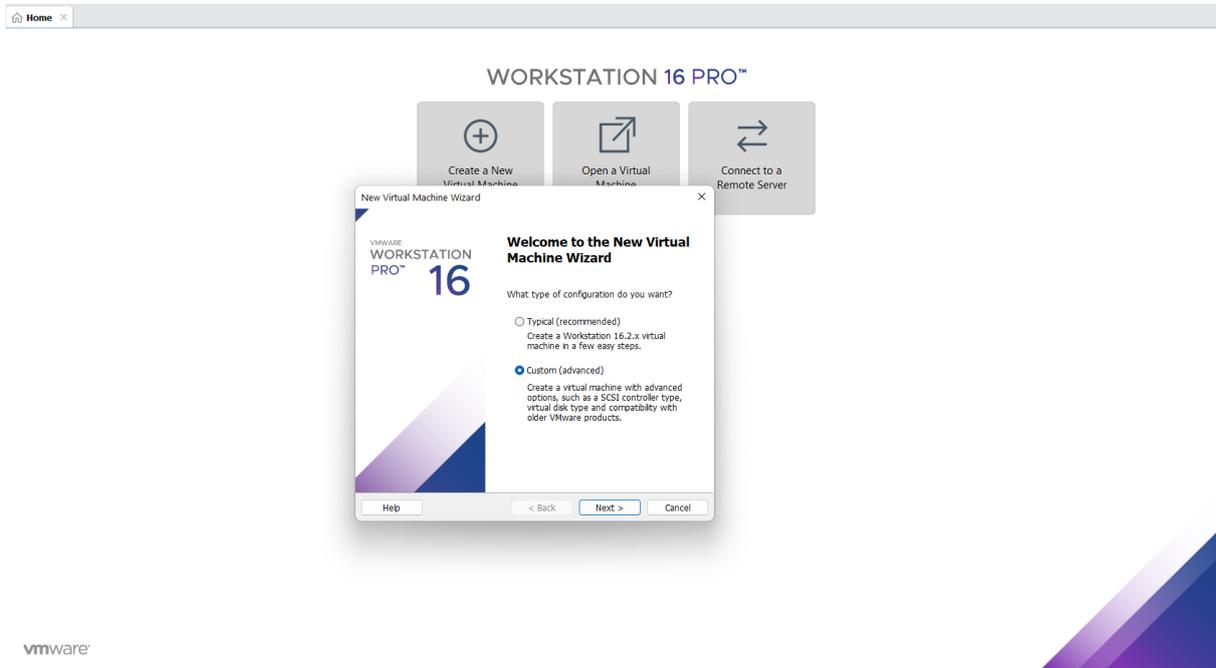
ETAPE 1 – PARAMETRER UNE MACHINE VIRTUELLE

Nous allons tout d'abord télécharger un système d'exploitation Linux, ici Ubuntu, depuis :

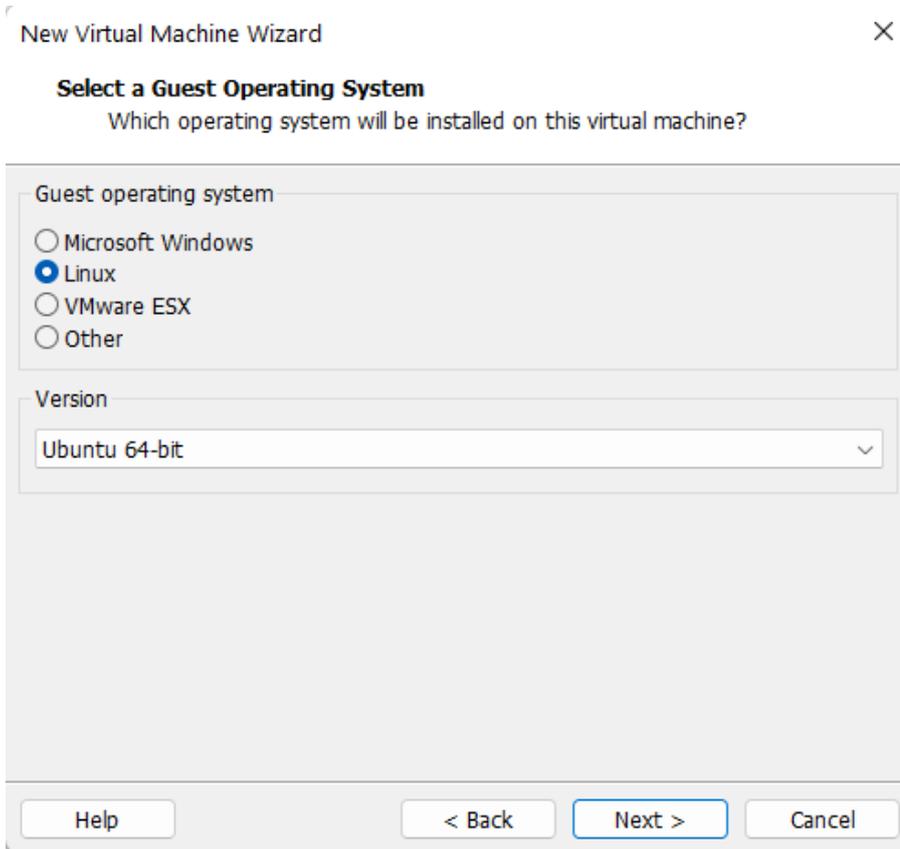


The screenshot shows the 'ubuntu-fr' website's download page. At the top left is the 'ubuntu-fr' logo. A navigation bar contains 'Accueil', 'Documentation', 'Forum', and 'À propos'. In the top right corner, there is a system tray showing 'ubuntu-22.04' and '3 min restantes'. The main heading is 'Téléchargements'. Below it, a dropdown menu is set to 'Ubuntu'. Under 'Liens de téléchargement direct', there is a download icon and the text 'dernière version LTS (recommandée)'. There is also an unchecked checkbox for 'Version Bittorent'. The footer contains social media links for Libera Chat (IRC), Telegram, Twitter, Facebook, Reddit, and LinkedIn, along with links for 'Télécharger Ubuntu - Ordinateurs vendus avec Ubuntu' and 'Mentions Légales'.

Nous allons ensuite créer une machine Virtuelle (ici depuis le logiciel VMware Workstation)

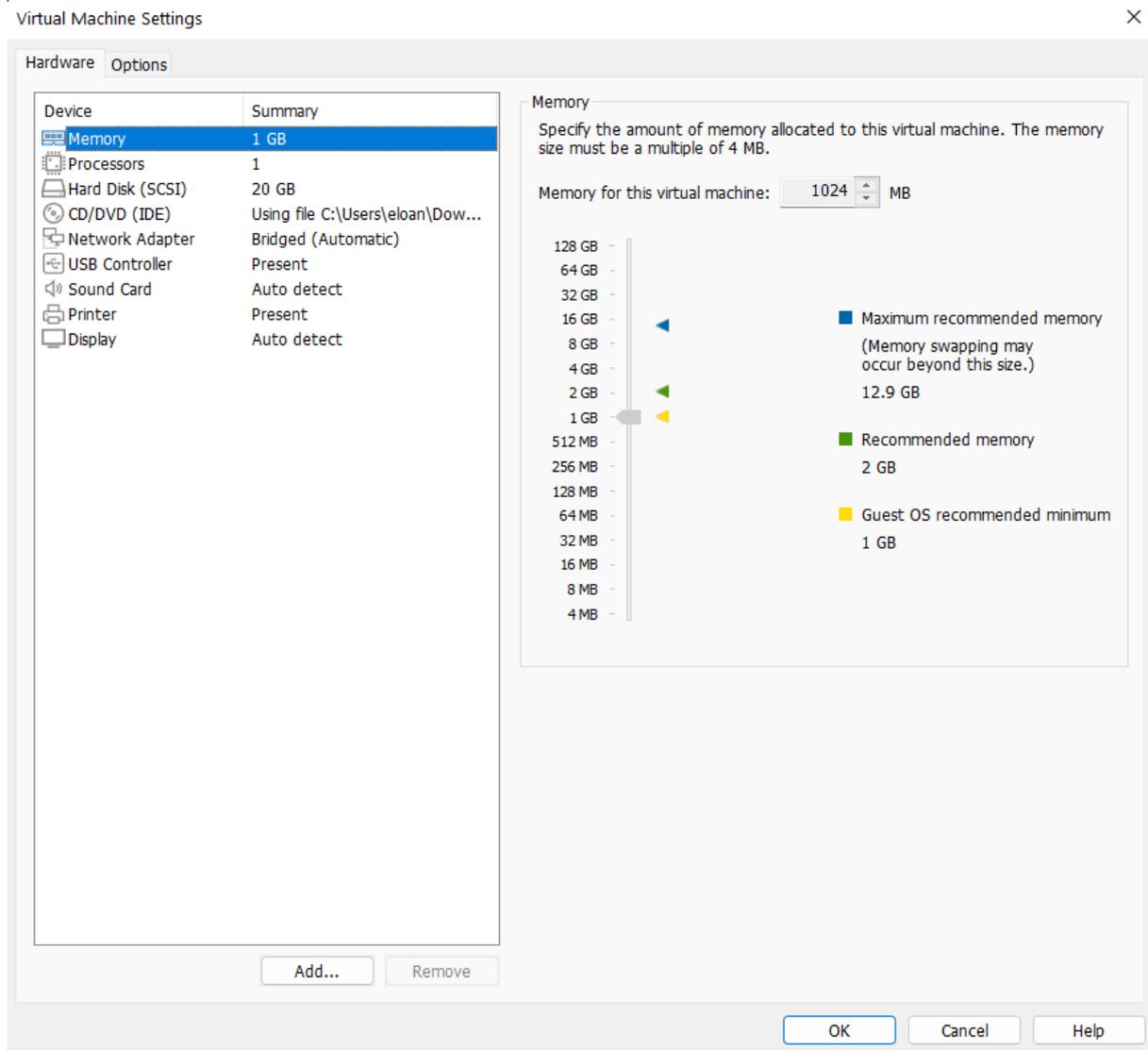


Durant la création de la machine virtuelle, on va choisir la version Ubuntu 64-Bit



Les choix suivants seront ceux par défaut.

Une fois notre machine virtuelle créée, nous allons procéder aux paramétrages.



Mémoire → Mettre environ 1GB de mémoire

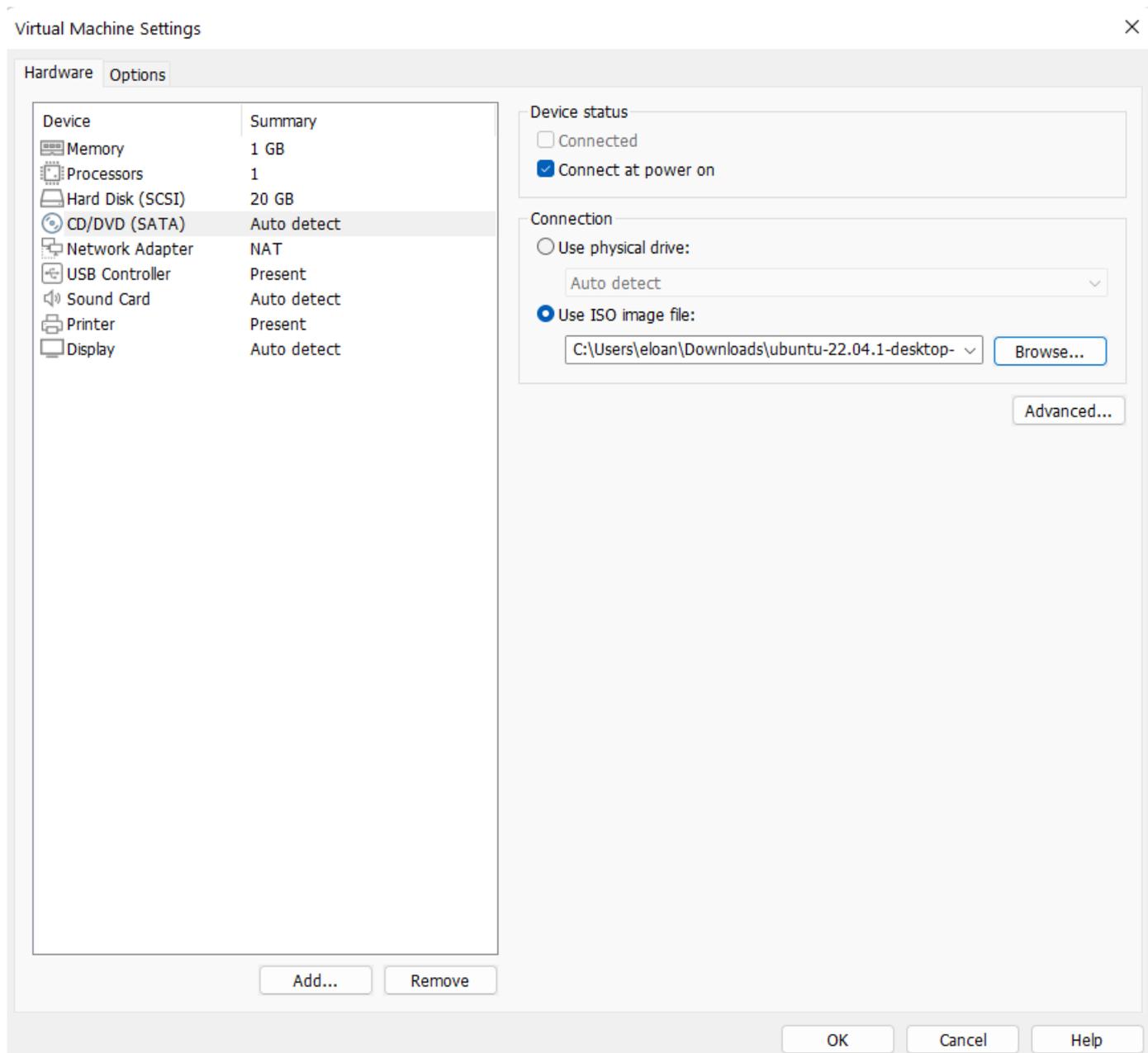
Processeur → Un seul processeur suffit

Disque dur → Mettre 20 GB

Network → Bridged (Automatique)

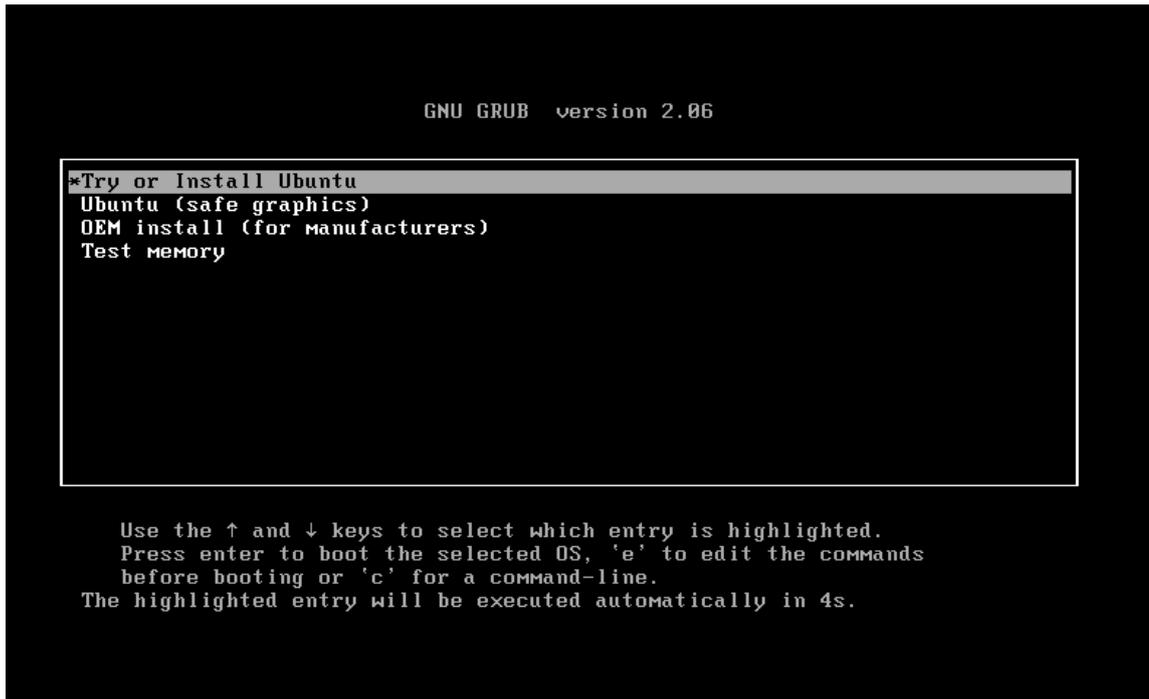
Les autres paramètres peuvent être laissés par défaut.

Pour la partie CD/DVD → Nous allons pouvoir insérer notre image précédemment téléchargée dans « Use ISO image file » comme ci-dessous :

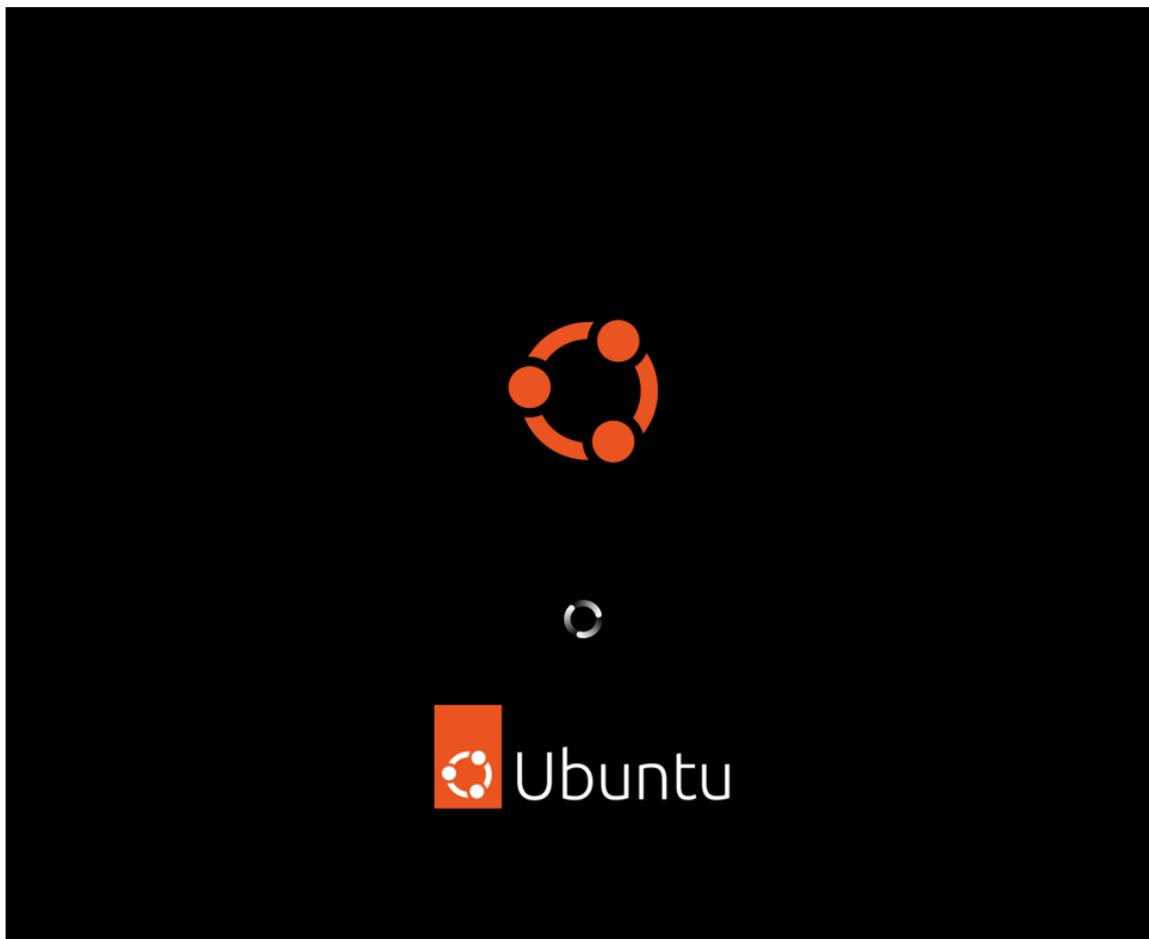


ETAPE 2 : INSTALLER UBUNTU

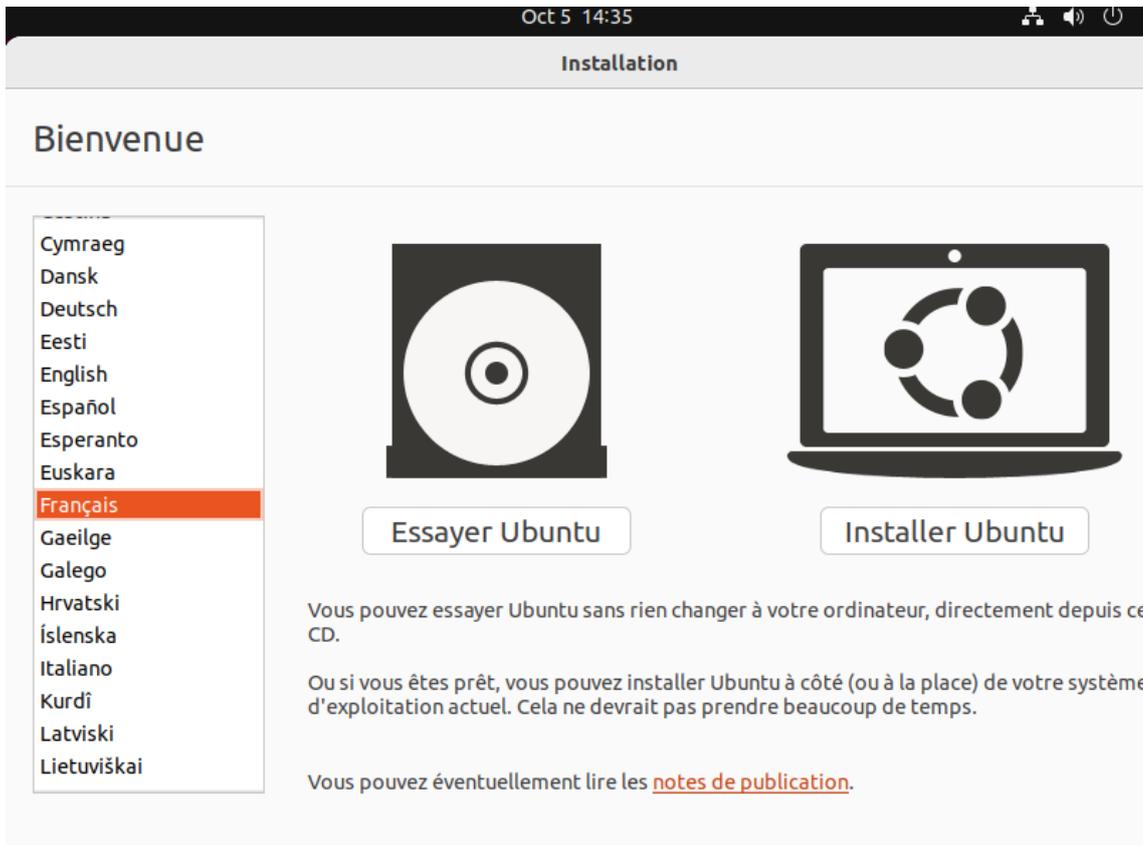
On lance la machine virtuelle et on sélectionne « Try or Install Ubuntu »



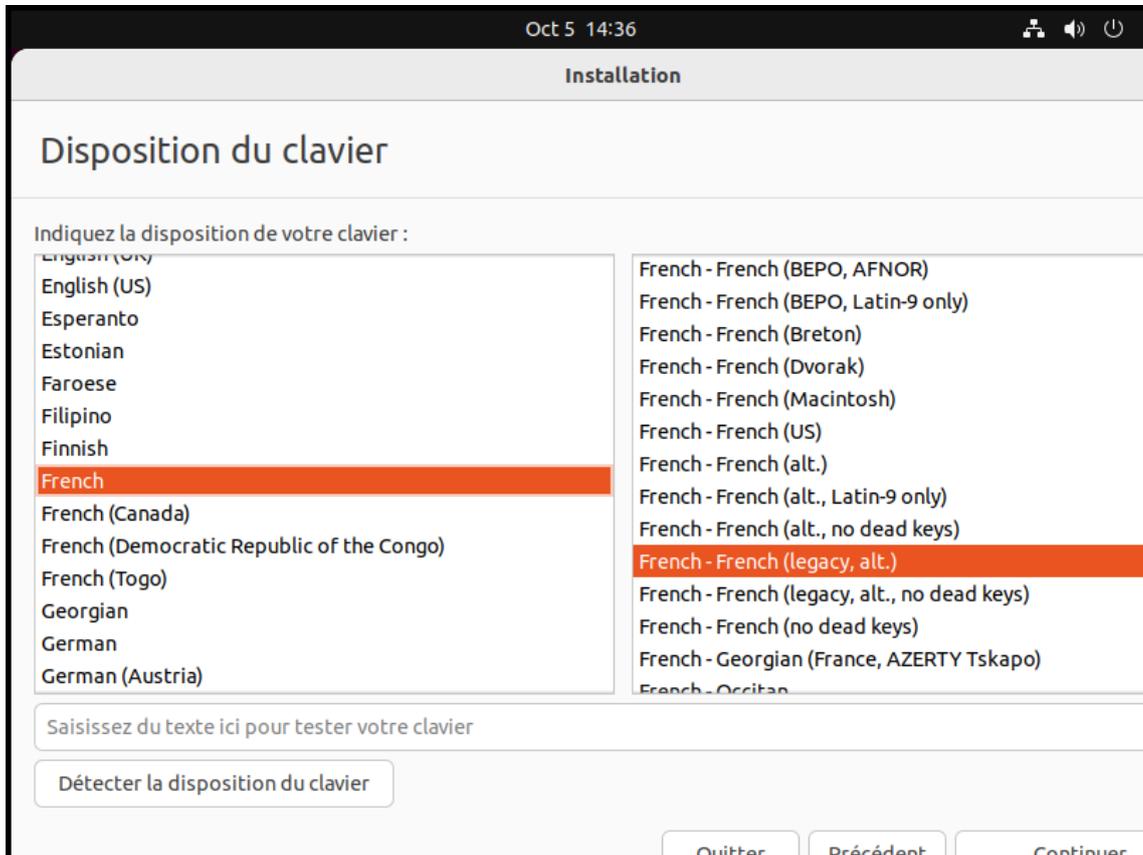
Attendre la fin de chargement de l'écran suivant (cela peut prendre plusieurs minutes) :



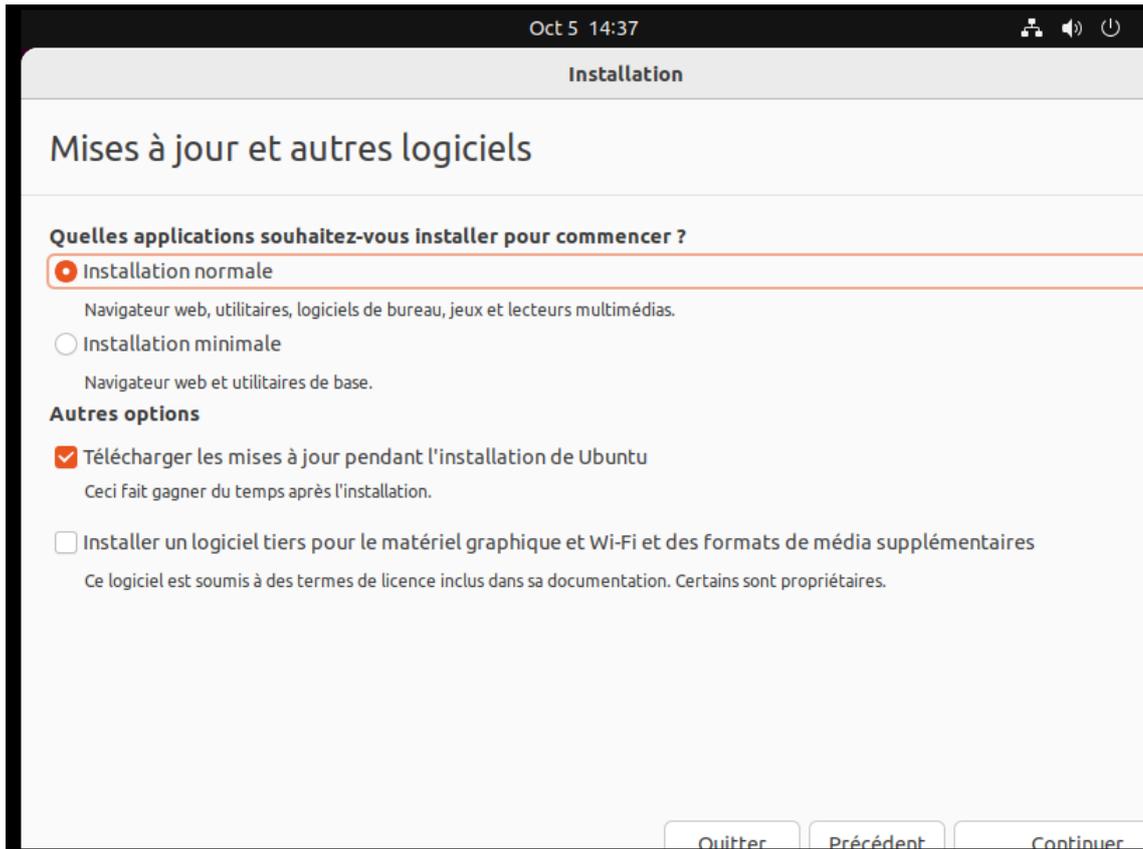
Au lancement, sélectionner « français » et « Installer Ubuntu » :



Sélectionner la disposition du clavier souhaitée :

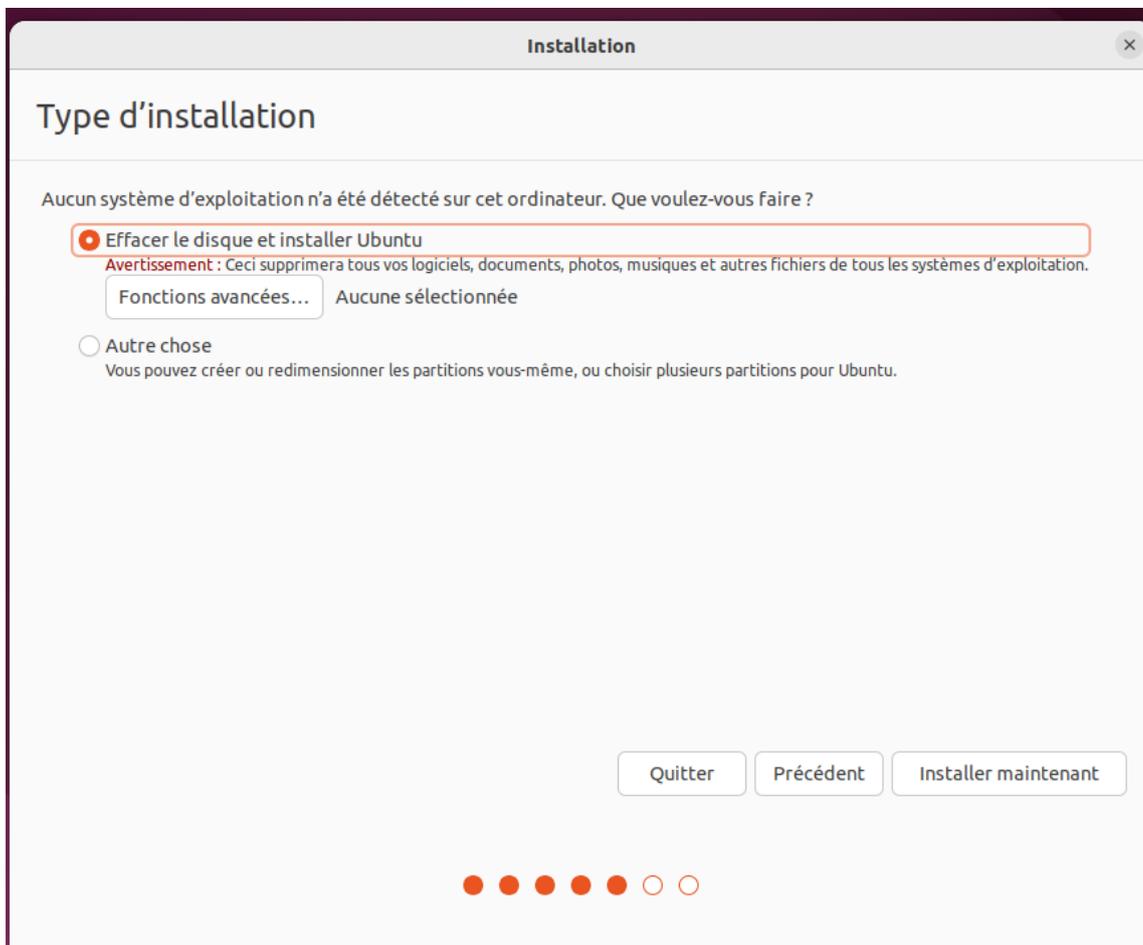


Sélectionner « Installation normale » + « Mises à jour pendant l'installation » :



Attendre la fin de l'installation (cela peut prendre plusieurs minutes)

Choisir « effacer les disque et installer Ubuntu » (uniquement si vous êtes sur une machine virtuelle) :



Choisir votre localisation, puis vos identifiants et mots de passe :

Installation

Qui êtes-vous ?

Votre nom : ✓

Le nom de votre ordinateur : ✓
Le nom qu'il utilise pour communiquer avec d'autres ordinateurs.

Choisir un nom d'utilisateur : ✓

Choisir un mot de passe :

Confirmez votre mot de passe :

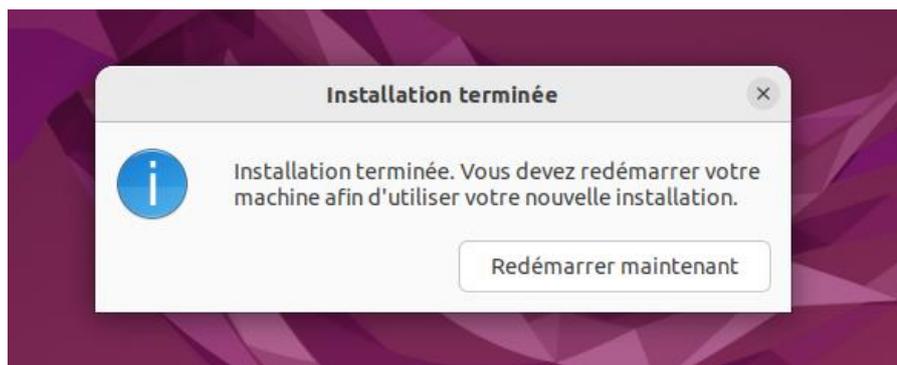
Ouvrir la session automatiquement
 Demander mon mot de passe pour ouvrir une session
 Utiliser Active Directory
Vous saisissez le domaine et d'autres détails à l'étape suivante.

● ● ● ● ● ● ●

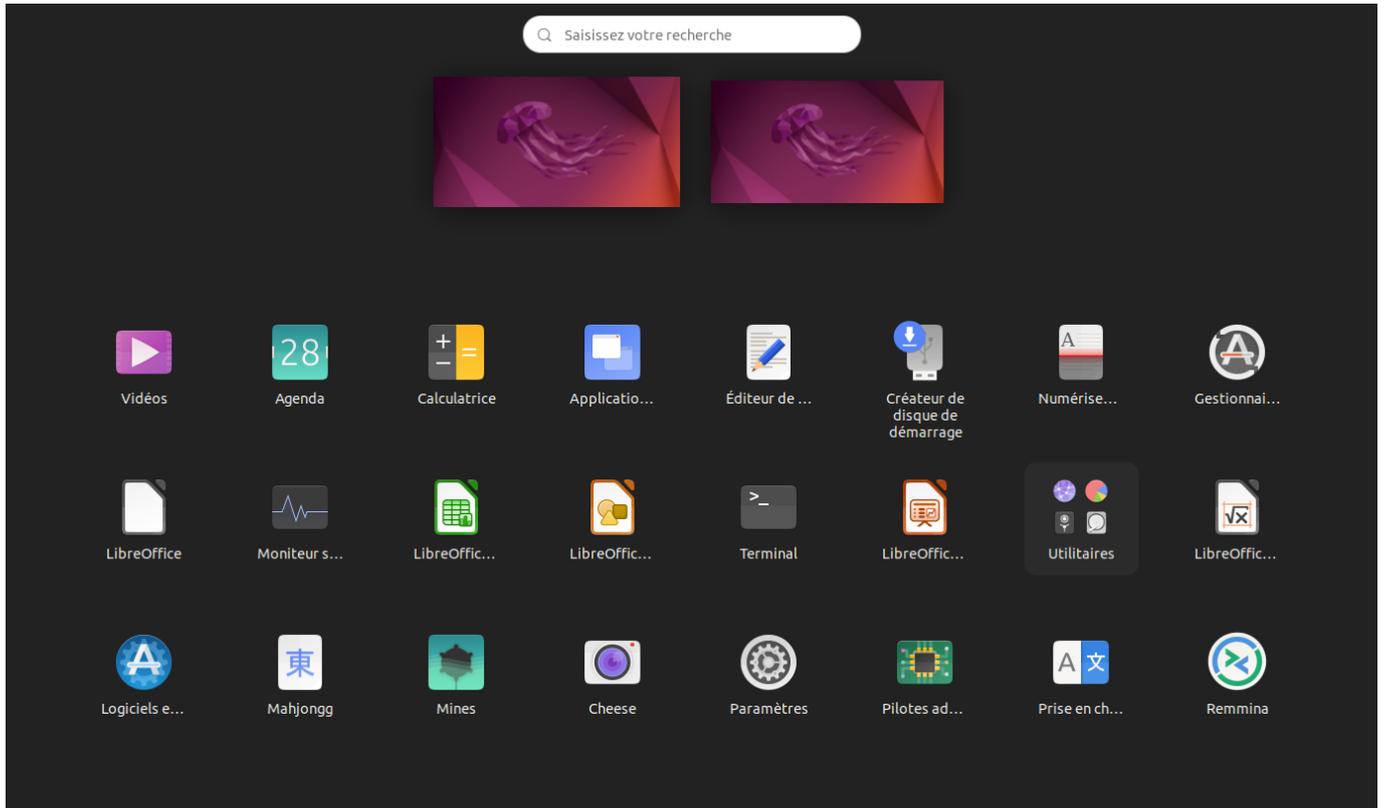
Ex de mdp : Ipssi2022

Attendre la fin de l'installation du système (cela peut prendre plusieurs minutes)

Redémarrer une fois l'installation terminée



Ouvrir la session puis accéder au Terminal :



ETAPE 3 : INSTALLER LE SERVICE DOCKER

Une fois dans le terminal, taper la suite de commandes permettant de passer en root de manière permanente :

```
user@user-virtual-machine:~$ sudo -i
```

Ensuite on met à jour tout le système à l'aide des commandes ci-dessous. Apt update va chercher les mises à jour disponibles pour le système et les applications tandis que Apt upgrade va les appliquer. De cette façon on est sûr que la machine possède sa configuration optimale.

```
root@user-virtual-machine:~# apt update
```

```
root@user-virtual-machine:~# apt upgrade -y
```

Indiquer « Y » (Yes) à la question « Do you want to continue ? »

Une fois la procédure complétée, tapez la commande ci-dessous afin d'installer tous les paquets nécessaires pour permettre à Apt d'utiliser des paquets sur HTTPS.

```
root@user-virtual-machine:~# apt install apt-transport-https ca-certificates curl software-properties-common
```

Vous pouvez ensuite effectuer cette commande afin d'ajouter la clé GPG officielle de Docker à votre système.

```
root@user-virtual-machine:~# curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

Tapez ensuite cette commande afin d'ajouter le référentiel Docker aux sources Apt.

```
root@user-virtual-machine:~# add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu focal stable"
```

Vous pouvez entrer cette commande pour vous assurer que vous téléchargez bien depuis le dépôt de Docker et non depuis celui d'Ubuntu.

```
root@user-virtual-machine:~# apt-cache policy docker-ce
```

Finalement il vous suffira d'entrer cette commande pour bien installer Docker. Appuyer sur "Y" à la question "Do you want to continue ?"

```
root@user-virtual-machine:~# apt install docker-ce
```

ETAPE 4 : INSTALLATION DE NGINX + CREER LES CONTENEURS ET LES DEMARRER

On va ensuite chercher une image docker debian avec nginx sur https://hub.docker.com/_/nginx

```
root@user-virtual-machine:~# docker pull nginx
```

Afin de vérifier si l'image à bien été importé, tapez la commande ci-dessous :

```
root@user-virtual-machine:~# docker images
```

Maintenant, créez les conteneurs ayant nginx de base d'installé grâce aux commandes suivantes :

```
root@user-virtual-machine:~# docker run --name rproxy -d -p 80:80 nginx
aad474ffb70ac005255d9a9df00d7b52169a29135df59ee0883e9fa61b3d83
root@user-virtual-machine:~# docker run --name webs1 -d nginx
7fb09bdff6190c7c6ab8570121fa07c530f473288feda2115259377d9d8eb8c0
root@user-virtual-machine:~# docker run --name webs2 -d nginx
7b58a3ea597c3276ed82da05564e171e16f45738c9e6030cb8c6b1ac4a7c462b
```

Normalement ces conteneurs se lanceront tous seuls de base, mais afin d'être sûr de ce bon lancement, vous pouvez utiliser cette commande afin d'obtenir la liste des conteneurs lancés.

```
root@user-virtual-machine:~# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS
NAMES
7b58a3ea597c   nginx    "/docker-entrypoint..." About a minute ago Up About a minute 80/tcp
webs2
7fb09bdff619   nginx    "/docker-entrypoint..." About a minute ago Up About a minute 80/tcp
webs1
aad474ffb70a   nginx    "/docker-entrypoint..." About a minute ago Up About a minute 0.0.0.0:80->80/tcp, :::80->80
/tcp
rproxy
```

Le dernier conteneur correspond au reverse proxy, les 2 premiers à nos serveurs web. On peut désormais entrer dans les conteneurs à l'aide la commande suivante, qu'on peut évidemment adapter selon le conteneur. N'oubliez pas d'utiliser le apt update et apt upgrade pour mettre à jour les conteneurs convenablement :

```
root@user-virtual-machine:~# docker exec -it webs1 /bin/bash
```

```
root@user-virtual-machine:~# docker exec -it webs2 /bin/bash
```

```
root@user-virtual-machine:~# docker exec -it rproxy /bin/bash
```

```
root@7fb09bdff619:/# apt update
```

```
root@7fb09bdff619:/# apt upgrade -y
```

Pour prendre de l'avance et finaliser la création des conteneurs de base, dans les conteneurs entrez cette commande afin d'installer vim qui nous permettra de modifier aisément les fichiers programmes.

```
root@7fb09bdff619:/# apt install vim
```

Si besoin de sortir d'un conteneur, il suffit de taper "exit".

ETAPE 5 : CREATION ET CONFIGURATION DU LIEN SSH ENTRE LES 2 SERVEURS WEB ET SYNCHRONISATION AUTOMATIQUE DES 2 SERVEURS (5 MINUTES)

On va ensuite installer les applications clients et serveurs OpenSSH. OpenSSH client étant déjà installé par défaut, on va installer OpenSSH serveur depuis le serveur 2 (webs2)

On va ensuite installer OpenSSH pour serveur dans le serveur 2 (webs2) grâce à la commande suivante :

```
root@user-virtual-machine:~# docker exec -it webs2 /bin/bash
```

```
root@7b58a3ea597c:/# apt install openssh-server -y
```

On va ensuite retourner dans le serveur 1 (webs1) pour générer la clé privé et publique pour que les 2 serveurs puissent communiquer en ssh.

```
root@user-virtual-machine:~# docker exec -it webs1 /bin/bash
```

```
root@7fb09bdff619:~# apt install openssh-client
```

```
root@7b58a3ea597c:/# ssh-keygen -t rsa
```

Ensuite, appuyer 3 fois sur « entrée » pour skipper tous les mots de passe.

```
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Created directory '/root/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa
Your public key has been saved in /root/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:PqD0WZtP75Lrs0hrsBgfLJHoGM7osz65ZXIEhx8m3v8 root@7fb09bdff619
The key's randomart image is:
+---[RSA 3072]---+
|
|  =
| + X .
| *o * . . S
| o++.=o = o
| ...0.=o * ..
| =* o o. *o.
| .+= o+E. .++o
+---[SHA256]-----+
```

Modifier le fichier sshd_config (avec Vim) sur le serveur 2 grâce à la commande :

```
root@user-virtual-machine:~# docker exec -it webs2 /bin/bash
root@7b58a3ea597c:/# vim /etc/ssh/sshd_config
```

Toujours dans le serveur 2, on va autoriser le serveur 1 à se connecter avec l'utilisateur root (administrateur) du serveur 2 en modifiant la ligne comme ci-dessous :

```
#Port 22
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::

#HostKey /etc/ssh/ssh_host_rsa_key
#HostKey /etc/ssh/ssh_host_ecdsa_key
#HostKey /etc/ssh/ssh_host_ed25519_key

# Ciphers and keying
#RekeyLimit default none

# Logging
#SyslogFacility AUTH
#LogLevel INFO

# Authentication:

#LoginGraceTime 2m
PermitRootLogin yes
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10

#PubkeyAuthentication yes
"/etc/ssh/sshd config" 123L, 3274B
```

Pour enregistrer ces changements, appuyer sur SHIFT + ZZ.

Relancer le service sshd grâce à la commande :

```
root@7b58a3ea597c:~# service ssh restart
Restarting OpenBSD Secure Shell server: sshd.
```

Nous avons besoin du mot de passe de l'utilisateur root du serveur 2 pour l'étape suivante.

Aller sur le serveur 2. Pour trouver le mot de passe du serveur 2, on peut taper « passwd » comme ci-dessous :

```
root@7b58a3ea597c:~# passwd
New password:
Retype new password:
passwd: password updated successfully
```

Sortir du webs2 avec la commande « exit » puis entrer la commande

```
root@user-virtual-machine:~# docker inspect bridge
```

Cette commande vous permet d'avoir l'ip de chacune de vos containers maintenant noter celle du webs2.

```
[
  {
    "Name": "bridge",
    "Id": "a934ddd3c66e2cac1a69d8e69fd21c3996371ecd8ddd517fdf78bfe8712c3220",
    "Created": "2022-10-18T23:18:21.605418898+02:00",
    "Scope": "local",
    "Driver": "bridge",
    "EnableIPv6": false,
    "IPAM": {
      "Driver": "default",
      "Options": null,
      "Config": [
        {
          "Subnet": "172.17.0.0/16"
        }
      ]
    },
    "Internal": false,
    "Attachable": false,
    "Ingress": false,
    "ConfigFrom": {
      "Network": ""
    },
    "ConfigOnly": false,
    "Containers": {
      "7b58a3ea597c3276ed82da05564e171e16f45738c9e6030cb8c6b1ac4a7c462b": {
        "Name": "webs2",
        "EndpointID": "ab0c5929eadfeb1a8c0474949456657b9a6586cbb483e3c2bb3e7c7640a93389",
        "MacAddress": "02:42:ac:11:00:04",
        "IPv4Address": "172.17.0.4/16",
        "IPv6Address": ""
      },
      "7fb09bdff6190c7c6ab8570121fa07c530f473288feda2115259377d9d8eb8c0": {
        "Name": "webs1",
        "EndpointID": "f24670378e114a074e44c731d4661068f9c4511afd61c6a82fa248213fb2120d",
        "MacAddress": "02:42:ac:11:00:03",
        "IPv4Address": "172.17.0.3/16",
        "IPv6Address": ""
      },
      "aad474ffbbb70ac005255d9a9df00d7b52169a29135df59ee0883e9fa61b3d83": {
        "Name": "rproxy",
        "EndpointID": "8e000d5147d7df54548838778791454a2d922a0b3e8c57b42b0de5b24a0940dd",
        "MacAddress": "02:42:ac:11:00:02",
        "IPv4Address": "172.17.0.2/16",
        "IPv6Address": ""
      }
    },
    "Options": {
      "com.docker.network.bridge.default_bridge": "true",
      "com.docker.network.bridge.enable_icc": "true",
      "com.docker.network.bridge.enable_ip_masquerade": "true",
      "com.docker.network.bridge.host_binding_ipv4": "0.0.0.0",
      "com.docker.network.bridge.name": "docker0",
      "com.docker.network.driver.mtu": "1500"
    },
    "Labels": {}
  }
]
```

Retourner sur le serveur 1. Aller dans le répertoire /root/.ssh grâce à la commande suivante :

```
root@7fb09bdff619:~# cd /root/.ssh
```

Puis saisir la commande :

```
root@7fb09bdff619:~/.ssh# ssh-copy-id -i id_rsa.pub root@172.17.0.4
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "id_rsa.pub"
The authenticity of host '172.17.0.4 (172.17.0.4)' can't be established.
ECDSA key fingerprint is SHA256:bTsrjs5EKJl2Kq3tMmXB3iQlNir4BgbwFRSy3uldUbM.
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the fingerprint: yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
root@172.17.0.4's password:
Number of key(s) added: 1
Now try logging into the machine, with: "ssh 'root@172.17.0.4'"
and check to make sure that only the key(s) you wanted were added.
```

Note : L'adresse ip doit correspondre à l'adresse de votre serveur 2.

Cela permet alors d'envoyer les clés autorisées au serveur 2.

Vérifier sur le serveur 2 que les clés ont bien été envoyée à l'aide de la commande suivante :

```
root@7b58a3ea597c:~# cd /root/.ssh
root@7b58a3ea597c:~/ .ssh# ls
authorized_keys
```

Les clefs ont bien été envoyées.

On va ensuite installer *crontab* afin de pouvoir inclure le timer. Retourner dans le serveur 1 (webs1) et entrer la commande suivante :

```
root@7fb09bdff619:~# apt install cron -y
```

Dans le serveur 1, afin d'inclure le timer de 5 minutes, entrer la commande suivante qui va permettre d'indiquer la durée voulue :

```
root@webs1:~# crontab -e
```

Entrer ensuite la commande suivante dans le fichier crontab :

```

# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow  command
*/5 * * * * scp /usr/share/nginx/html root@172.17.0.4:/var/www/html/index.html
~
~
~
~

```

Note : L'adresse ip est celle du serveur 2.

Note : Les étoiles « * » vont respectivement représenter les durées. Le « / » va permettre d'indiquer que le timer doit se faire toutes les 5 minutes.

Une fois le timer configuré, écrire la commande que vous voulez automatiser à la suite des « * ».

Afin de vérifier que le script du timer est opérationnel, entrer la commande suivante, toujours dans le serveur 1 :

```
root@7fb09bdf619:~# cat /var/log/syslog
```

On va ensuite aller dans le reverse proxy et on va venir modifier le fichier « default » avec VIM :

ETAPE 6: INSTALLER ET PARAMETRER LA FONCTION DU REVERSE PROXY

Pour commencer le paramétrage on entre dans le conteneur rproxy au moyen de la commande :

```
root@user-virtual-machine:~# docker exec -it rproxy /bin/bash
```

On se déplace ensuite jusqu'à ce fichier pour l'éditer au moyen de vim. Dedans on modifiera les paramètres selon la capture globale ci-dessous :

```
root@aad474ffbbb7:~# vim /etc/nginx/conf.d/default.conf
```

```
server {
    listen      80;
    listen     [::]:80;
    server_name localhost;

    #access_log /var/log/nginx/host.access.log main;

    location / {
        proxy_pass http://172.17.0.3/;
    }

    #error_page 404              /404.html;

    # redirect server error pages to the static page /50x.html
    #
    error_page   500 502 503 504  /50x.html;
    location = /50x.html {
        root   /usr/share/nginx/html;
    }

    # proxy the PHP scripts to Apache listening on 127.0.0.1:80
    #
    #location ~ /\.php$ {
    #    proxy_pass    http://127.0.0.1;
    #}

    # pass the PHP scripts to FastCGI server listening on 127.0.0.1:9000
    #
    #location ~ /\.php$ {
    #    root           html;
    #    fastcgi_pass   127.0.0.1:9000;
    #    fastcgi_index  index.php;
    #    fastcgi_param  SCRIPT_FILENAME  /scripts$fastcgi_script_name;
    #    include        fastcgi_params;
    #}

    # deny access to .htaccess files, if Apache's document root
    # concurs with nginx's one
    #
    #location ~ /\.ht {
    #    deny  all;
    #}
}
server {
    listen      8080;
    listen     [::]:8080;
    server_name localhost;

    location / {
        proxy_pass http://172.17.0.4/;
    }
}
```

On finit en redémarrant le service nginx afin que les modifications soient appliquées. Les lignes ci-dessous confirmeront son état de fonctionnement sain.

```
root@a982f27ac651:/# systemctl restart nginx
```

```
root@aad474ffbbb7:~# nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
root@aad474ffbbb7:~# nginx -s reload
2022/10/18 23:36:57 [notice] 484#484: signal process started
```

DERNIERE ETAPE: VERIFICATION DU REVERSE PROXY

On commence par modifier les 2 index.html dans les serveurs webs1 et webs2

On se connecte au serveur 1 avec la commande ci-dessous :

```
root@user-virtual-machine:~# docker exec -it webs1 /bin/bash
```

On modifie ensuite le fichier index.html en insérant cette partie de code dans le fichier html du serveur web 1.

```
root@7fb09bdff619:~# vim /usr/share/nginx/html/index.html
```

```
<!DOCTYPE html>
<html>
<head>
<title>Docker 1</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Page 1!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
~
~
~
```

Ensuite on va sur le serveur 2 et on recommence la procédure. On fera attention à différencier les pages pour de futurs tests de fonctionnement.

```
root@user-virtual-machine:~# docker exec -it webs2 /bin/bash
```

On répète l'étape de déplacement jusqu'au fichier pour le modifier.

```
root@7b58a3ea597c:~# vim /usr/share/nginx/html/index.html
```

```

<!DOCTYPE html>
<html>
<head>
<title>Docker 2</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Page 2</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<b>g</b>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
~
~
~

```

On teste le reverse proxy avec ces 2 commandes.

```
root@user-virtual-machine:~# curl 172.17.0.2
```

```
root@user-virtual-machine:~# curl 172.17.0.2:8080
```