METTRE EN PLACE UN SERVEUR WEB - Grafikart

Le but est qu’en tapant l’ip de la machine on arrive sur un site web

Un **serveur informatique** est un dispositif [informatique](https://fr.wikipedia.org/wiki/Informatique) ([matériel](https://fr.wikipedia.org/wiki/Mat%C3%A9riel_informatique) ou [logiciel](https://fr.wikipedia.org/wiki/Logiciel)) qui offre des [services](https://fr.wikipedia.org/wiki/Service_r%C3%A9seau), à un ou plusieurs [clients](https://fr.wikipedia.org/wiki/Client_(informatique)) (parfois des milliers).

Ce sont généralement de grosses armoires ayant des tiroirs contenant chacun des mini ordinateurs.

Un serveur fonctionne en permanence, répondant automatiquement à des requêtes provenant d'autres dispositifs informatiques (les clients).

Il doit avoir un très bon débit, beaucoup de ram, un bon processeur.

Il a besoin d’un OS : Linux, Windows Server ou Mac OSX server

Les services rendus par un serveur :

Web - Mail - Base de données, etc..

Linux étant très stable et gratuit, la majorité de serveurs sont avec ce système.

Afin d’offrir des services on doit installer des logiciels sur le serveur, si on veut qu’il puisse héberger des pages web on doit installer un serveur http comme Apache ou Nginx.

Si on veut y sauvegarder des données il va falloir installer MySQL, MariaDB, MongoDD, etc..

Pour le mail Postfix

Pour les DNS, Bind9, etc..

Installer un serveur c’est installer de nombreux logiciels.

Quand tu installes un logiciel via apt-get Install, il va chercher les fichiers sources dans /etc/apt/sources.list

Il peut être intéressant d’éditer ce fichier afin de supprimer une zone de recherche par exemple.

Via Virtualbox ou VmWare on installe un Linux en activant le SSH.

On installe apt-get install net-tools afin de pouvoir faire un ifconfig.

On installe vim —> apt-get install vim

On fait un update —> apt-get update

Connexion en SSH

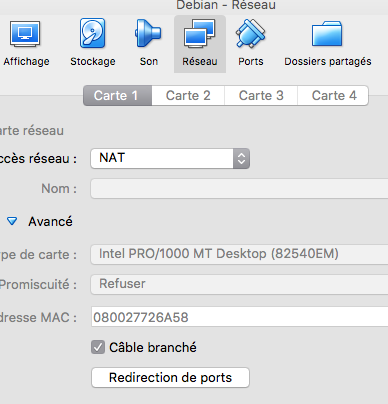
SSH utilise les deux chiffrements : asymétrique et symétrique. Cela fonctionne dans cet ordre.

1. On utilise d'abord le chiffrement asymétrique pour s'échanger discrètement une clé secrète de chiffrement symétrique.
2. Ensuite, on utilise tout le temps la clé de chiffrement symétrique pour chiffrer les échanges.

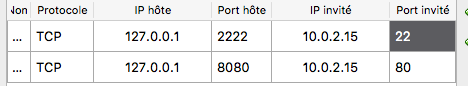
S’il n’a pas été installé lors de l’installation de l’OS, on tape :

apt-get Install openssh-server

Si on utilise Virtualbox il faut faire une redirection de ports dans configuration



Comme on est en local on met comme Ip-hôte l’ip local et en invité l’ip du linux.



Ensuite de ton MAC on se connecte à distance à sa machine via le protocole SSH en tapant ssh login@ip par exemple :

ssh [qeyss@127.0.0.1](mailto:qeyss@127.0.0.1) -p 2222

Le -p 2222 permet de spécifier le port.

1er sécurité : changer le port 22 et en mettre un autre.

Editer le fichier sshd\_config —> Vim /etc/ssh/sshd\_config

Et mettre un autre port, par exemple 5789

PermitRootLogin permet de dire si SSH autorise ou non la connexion au compte root. Par mesure de sécurité, il vous est recommandé de mettre cette option à no.

Puis faire ssh restart

Sous windows il faut utiliser Putty

**Authentification par clé depuis Linux**

Sur la machine client afin de générer une paire de clés publique / privée on tape: ssh-keygen -t ras

Le client génère une paire de clés qu’il va sauvegarder dans des fichiers (un pour la clé publique, un pour la clé privée).

Ensuite taper Entrée à chaque fois sans rien mettre, cela va générer trois fichiers cachés :

Les trois fichiers sont :

* id\_rsa : votre clé privée, qui doit rester secrète. Elle est chiffrée si vous avez rentré une passphrase ;
* id\_rsa.pub : la clé publique que vous pouvez communiquer à qui vous voulez, et que vous devez envoyer au serveur ;
* known\_hosts : c'est la liste des fingerprint que votre PC de client tient à jour. Ça lui permet de se souvenir de l'identité des serveurs et de vous avertir si, un jour, votre serveur est remplacé par un autre (qui pourrait être celui d'un pirate !). Je vous en ai déjà parlé un peu plus tôt.

Ensuite il faut envoyer au serveur la clé publique (id\_rsa.pub) et à l'ajouter à son fichier authorized\_keys (clés autorisées). Le serveur y garde une liste des clés qu'il autorise à se connecter.

On utilise la commande :

ssh-copy-id -i id\_rsa.pub login@ip ex :

ssh-copy-id -i id\_rsa.pub [qeyss@127.0.0.1](mailto:qeyss@127.0.0.1)

Si tu te connectes par un autre port alors tape :

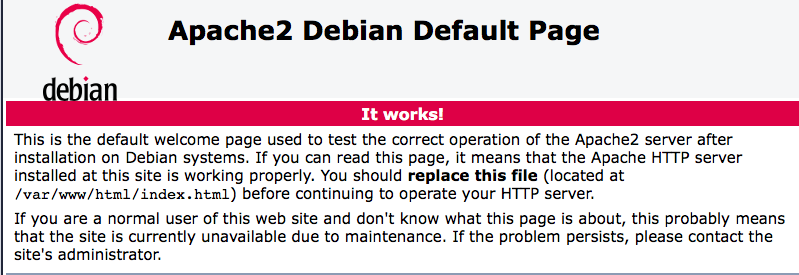
ssh-copy-id -i id\_rsa.pub -p NuméroDuPort login@ip ex :

ssh-copy-id -i id\_rsa.pub -p 2222 [qeyss@127.0.0.1](mailto:qeyss@127.0.0.1)

**Mise en place d’un serveur HTTP APACHE**

apt-get install apache2

Comme on a fait une redirection on verra la page à l’adresse 127.0.0.1:8080



Dans /etc/apache2/ il y a les fichiers de configuration d’apache dont le fichier ports.conf indiquant le port par défaut à savoir le 80.

Il y a aussi les dossiers sites-available et sites-enabled

Sites-available a un fichier 000-defaut.conf qui contient la configuration par défaut.

Le fichier 000-defaut.conf contient des virtualhost, ce qui permet d’héberger plusieurs sites sur le serveur à la même adresse.

Il nous indique que la page web par défaut est située dans /var/www/html/index.html

Via FTP on clique sur l’icône de gestionnaire de site et on configure un nouveau site pour une nouvelle connexion avec pour adresse hôte 127.0.0.1 et le port choisi

Puis on crée un dossier www et on y met nos fichiers pour notre site. Ensuite dans 000-defaut.conf on mettra l’adresse de ce dossier dans DocumentRoot soit /home/qeyss/www

Puis taper service apache2 reload

Dans /var/log/apache2 il y a un fichier error.log, il est important de le consulter pour chaque erreur rencontré, on peut demander à ne voir qu’un certain nombre des dernières lignes en tapant par exemple:

tail -n 30/var/log/apache2/error.log

Donc pour que ça fonctionne il faut aussi ajouter dans /etc/apache2/apache2.conf un répertoire en tapant ceci :

<Directory /home/qeyss/www>

Options FollowSymLinks

AllowOverride none

Require all granted

</Directory>

Puis taper service apache2 reload

Sinon on ajoute pas de directory mais juste un lien symbolique en tapant : ln -s /home/qeyss/www /var/www/[lifeuk.fr](http://lifeuk.fr)

ln -s dossierALier/ nomLienSymbolique

Ensuite on va lui créer son propre VirtualHost dans sites-available qu’on va appeler 001-lifeuk.conf et y mettre ceci:

<Virtualhost \*:80>

ServerAdmin EmailAmettre

ServerName lifeuk

ServerAlias \*.lifeuk.fr

DocumentRoot /var/www/lifeuk.fr

<Directory /var/www/lifeuk.fr>

Options FollowSymLinks

AllowOverride All

</Directory>

</Virtualhost>

Puis on l’active en tapant : a2ensite 001-lifeuk

Si on veut pouvoir y accéder en tapant un nom, il faut éditer le fichier hosts situé dans /etc du Mac et mettre le nom de domaine souhaité face à 127.0.0.1

Lister les liens symboliques :

find . -type l

Supprimer un lien symbolique :

**unlink /mon\_repertoire/nom\_du\_lien**

**Mise en place d’un serveur NGINX**

apt-get install nginx

La config est dans /etc/nginx

Créer dans /home un dossier www où l’on y mettra son site.

Créer dans /home un dossier logs.

Dans /etc/hosts faire pointer son nom de domaine sur l’adresse localhost.

Ensuite il faut aller dans sites-available et créer un fichier.conf du nom du domaine souhaité et y mettre :

server {

listen 80 ;

server-name monsite.fr ;

root /home/www ;

index index.html

error\_log /home/logs/error\_log ;

access\_log /home/logs/access\_log;

}

Faire sudo service nginx start

Nginx ne sait pas traiter les .htaccess

Pour tester qu’une configuration est bonne on tape :

sudo nginx –t

Quand on cherche un paquet à installer sans connaître son nom exact on tape :

apt-cache search + NomDuLogiciel

**Installation de MySQL**

apt-get install mysql-server mysql-client

apt-get install php5-mysql

Il va falloir choisir un mot de passe root

mysql –uroot –p puis mettre le mot de passe root

Puis on va créer une base et un user:

CREATE DATABASE mlv

CREATE USER ‘qeyss’@’localhost’ IDENTIFIED BY ‘grozny77’;

Ensuite on lui attribue les droits sur la base souhaitée:

GRANT ALL PRIVILEGES ON mlv.\* TO ‘qeyss’@’localhost’;

En mettant .\* on lui donne les droits sur toutes les tables de cette base.

Puis on recharge les privilèges en tapant: FLUSH PRIVILEGES;

Puis on se reconnecte avec cet user :

mysql –uqeyss –p

Les fichiers de config sont dans /etc/mysql

Si tu veux une interface visuelle installe phpmyadmin ou adminer

Vidéo regardée jusqu’à 10 mn

Installation d’ATOM

sudo add-apt-repository ppa:webupd8team/atom

sudo apt-get update

sudo apt-get install atom

**Installation d’un serveur avec ANSIBLE**

Permet la configuration et déploiement de serveurs.

$ sudo apt-get update

$ sudo apt-get install software-properties-common

$ sudo apt-add-repository --yes --update ppa:ansible/ansible

$ sudo apt-get install ansible

Créer un fichhier hosts

Dedans y mettre des groupes avec les différents serveurs :

[web]

192.168.39.145

Puis il faut créer des recettes pour permettre l’automatisation et la configuration de ces groupes.

Les recettes sont en yaml et ont pour extension .yml

Syntaxe YAML

---

# A list of tasty fruits

fruits:

- Apple

- Orange

- Strawberry

- Mango

...

Exemple d’un fichier first.yml

On a un serveur de test sur lequel on souhaite installer automatiquement git à l’aide d’une recette.

---

- name: Installation de serveurs WEB

hosts: web

remote\_user: root ( indique le user pour se connecter sur les serveurs)

tasks :

* name: Installation de Git

apt: name=git update\_cache=yes

* name: Installation de Vim

apt: name=vim

...

apt: c’est un module où l’on définit ce que l’on souhaite executer.

Le update\_cache est pour faire un apt-get update, on le note qu’une fois dans la recette.

Ensuite pour exécuter cette recette on tape :

ansible-playbook –i hosts first.yml

Au lieu d’écrire un à un chaque composant à installer, on peut faire une syntaxe avec une boucle, ainsi:

---

- name: Installation de serveurs WEB

hosts: web

remote\_user: root ( indique le user pour se connecter sur les serveurs)

tasks :

* name: Installation de composants

apt: name={{ item }} update\_cache=yes state=latest

* with\_items:

- vim

- git

- zsh

...

Tasks :

- name: Création d’un utilisateur

user: name=qeyss shell=/usr/bin/zsh

- name: Ajout de la clé SSH

Authorized\_key : user=qeyss key=”{{lookup(‘file’, ‘~/ .ssh/id\_rsa.pub’) }}”

Si on veut gérer la création d’un user de manière dynamique on utilise une variable vars:

---

- name: Installation de serveurs WEB

hosts: web

remote\_user: root ( indique le user pour se connecter sur les serveurs)

vars:

user: qeyss

tasks :

* name: Installation de composants

apt: name={{ item }} update\_cache=yes state=latest

* with\_items:

- vim

- git

- zsh

- name: Création d’un utilisateur

user: name={{user}} shell=/usr/bin/zsh

- name: Ajout de la clé SSH

Authorized\_key : user={{user}} key=”{{lookup(‘file’, ‘~/ .ssh/id\_rsa.pub’) }}”

...

On peut ajouter des conditions pour par exemple créer le user que si la variable est définie, avec le mot when, ex :

---

- name: Installation de serveurs WEB

hosts: web

remote\_user: root ( indique le user pour se connecter sur les serveurs)

vars:

user: qeyss

tasks :

* name: Installation de composants

apt: name={{ item }} update\_cache=yes state=latest

* with\_items:

- vim

- git

- zsh

- name: Création d’un utilisateur

when : user is defined

user: name={{user}} shell=/usr/bin/zsh

- name: Ajout de la clé SSH

when : user is defined

Authorized\_key : user={{user}} key=”{{lookup(‘file’, ‘~/ .ssh/id\_rsa.pub’) }}”

...

Au lieu de créer tout un user dans le playbook, on peut créer un rôle de création d’un user et ensuite appeler ce rôle dans le playbook.

Donc il faut créer un dossier roles et dedans un dossier au nom du role que tu veux, par exemple user, et dans user dans un dossier tasks avec un fichier main.yml avec ce qu’il y a à faire exemple :

---

- name: Création d’un utilisateur

when : user is defined

user: name={{user}} shell=/usr/bin/zsh

- name: Ajout de la clé SSH

when : user is defined

Authorized\_key : user={{user}} key=”{{lookup(‘file’, ‘~/ .ssh/id\_rsa.pub’) }}”

…

On va ensuite écrire notre playbook ainsi:

---

- name: Installation de serveurs WEB

hosts: web

remote\_user: root ( indique le user pour se connecter sur les serveurs)

vars:

user: qeyss

tasks :

* name: Installation de composants

apt: name={{ item }} update\_cache=yes state=latest

* with\_items:

- vim

- git

- zsh

roles:

* user

...

Création d’un rôle utils pour vim, git et zsh:

Dans roles 🡪 utils 🡪 tasks 🡪 main.yml où l’on mettra :

---

* name: Installation de composants

apt: name={{ item }} update\_cache=yes state=latest

* with\_items:

- vim

- git

- zsh

…

Playbbok final :

On va ensuite écrire notre playbook ainsi:

---

- name: Installation de serveurs WEB

hosts: web

remote\_user: root ( indique le user pour se connecter sur les serveurs)

vars:

user: qeyss

roles:

* utils
* user

…

Ansible galaxy est un endroit où les gens publient leurs rôles.

Création d’un rôle pour la création de Nginx

Dans roles 🡪 nginx 🡪 tasks 🡪 main.yml

---

* name: Installation de Nginx

apt: name={{ item }} update\_cache=yes state=latest

* with\_items:

- nginx

* name: Start Nginx

service: name=nginx state=started enabled=true

…

Tâche qui va supprimer le fichier par defaut de Nginx

* name: Delete default.conf

file: path=/etc/nginx/sites-enabled/defaut state=absent

Quand tu fais une modification sur le serveur il faut une tâche relançant le serveur

Handlers :

Ce sont des tâches spécifiques à des conditions.

On doit créer un dossier handlers dans nginx avec un main.yml dedans

* name: nginx reload

action: service name=nginx state=reloaded

Ensuite dans le main de nginx sous la têche de suppression du fichier par defaut j’ajoute un notify :

* name: Delete default.conf

file: path=/etc/nginx/sites-enabled/defaut state=absent

notify: nginx reload

Les templates se font en .j2

Tu peux aussi inclure des fichiers de configuration

Par exemple si tu veux installer PHP, tu as dans ton rôle nginx un fichier php.yml où il y a ceci :

Vidéo stoppée à 29.30 car le reste est confus.