Répertorier et définir les outils nécessaires

Afin d'organiser notre **environnement de développement** plusieurs outils vont nous être nécessaires :

- Le **serveur Apache** : il va nous permettre de travailler **localement** sur nos machines. C'est lui qui va fournir le **service web**, autrement dit qui va recevoir les requête HTTP et délivrer les pages internet. On va lui adjoindre un module PHP afin de pouvoir interpréter le code PHP.

- L'**éditeur de texte Atom** : il va nous permettre d'écrire notre code.

- Le **framework CodeIgniter** : il va nous fournir un ensemble de fonctions qui vont nous faciliter la production du code.

- Le **logiciel de gestion de version Git**

Installation d'un serveur Apache sous linux

L'installation d'Apache se fait très simplement à partir du Terminal, en ligne de commande, grâce au gestionnaire de paquets ***apt*** :

```
apt install apache2
```

Une fois Apache installé, vous pouvez accédez au site par défaut via l'adresse IP de votre serveur.

Etant donné que vous utilisez un serveur local, son adresse est :
127.0.0.1 ou l**ocalhost.**

Vérifiez via votre navigateur, si apache est installé.

Installation d'un serveur Apache sous linux

Commandes utiles

Désactiver Démarrage auto : `systemctl disable apache2`

Réactiver : `systemctl enable apache2`

Démarrage du serveur : `/etc/init.d/apache2 start` **ou** `service apache2 start`

Redémarrage du serveur : `/etc/init.d/apache2 restart` **ou** `service apache2 restart`

Arrêt du serveur : `/etc/init.d/apache2 stop` **ou** `service apache2 stop`

Statut du serveur : `systemctl status apache2`

Installation d'un serveur MySQL sous linux

L'installation de MySQL se fait très simplement à partir du Terminal, en ligne de commande, grâce au gestionnaire de paquets ***apt*** :

```
apt install mysql-server
```

ou

```
apt install mariadb-server
```

Une fois l'installation, on lance le script de configuration pour la sécurité du serveur avec la commande :

```
mysql_secure_installation
```

Installation d'un serveur MySQL sous linux

Pour finir, nous allons créer un utilisateur admin(istratif) auquel l'on donnera l'ensemble des permissions, afin de pouvoir se connecter avec un compte différent de root :

```
mariadb
```

```
GRANT ALL ON *.* TO 'admin'@'localhost' IDENTIFIED BY 'password' WITH GRANT OPTION;
```

```
FLUSH PRIVILEGES;
```

Installation d'un serveur MySQL sous linux

Commandes utiles

Désactiver Démarrage auto : `systemctl disable mariadb`

Réactiver : `systemctl enable mariadb`

Démarrage du serveur : `/etc/init.d/mariadb start` **ou** `service mariadb start`

Redémarrage du serveur : `/etc/init.d/mariadb restart` **ou** `service mariadb restart`

Arrêt du serveur : `/etc/init.d/mariadb stop` **ou** `service mariadb stop`

Statut du serveur : `systemctl status mariadb`

Installation de PHP sous linux

L'installation de PHP se fait très simplement à partir du Terminal, en ligne de commande, grâce au gestionnaire de paquets ***apt*** :

```
apt install php
```

Une fois l'installation terminée, il ne reste plus qu'à installer les modules nécessaires à nos serveurs apache et mySQL :

```
apt install libapache2-mod-php
```

```
apt install php-mysql
```

LAMP

LAMP est un acronyme pour **L**inux, **A**pache, **M**ySQL, **P**HP. C'est une pile logicielle du système d'exploitation Linux, comprenant un serveur HTTP, un système de gestion de bases de données et un langage de programmation interprété, et qui permet de mettre en place un serveur web.

Toutes les étapes que nous avons effectuées précédemment se résument en fait à la mise en place de cette pile logicielle.

L'installation de LAMP peut se faire en réalité en une ligne, via la commande :

```
apt install apache2 php libapache2-mod-php mariadb-server php-mysql
```

Virtual Host

Dans le répertoire ***/etc/apache2/sites-available*** on crée le fichier qui va contenir notre virtualhost

```
sudo nano /etc/apache2/sites-available/codeigniter.conf
```

```
<VirtualHost *:80>
	ServerName tuto.local # Adresse principale
	ServerAlias www.tuto.local # Aliases du domaine, si l'adresse ou les adresses sont utilisée, on pointe au même endroit, facultatif
	DocumentRoot /home/wwwapp/tuto_ci/public # Répertoire où pointe le domaine
	ErrorLog /var/log/apache2/codeigniter-error_log
	CustomLog /var/log/apache2/codeigniter-access_log combined
	<Directory/home/wwwapp/tuto_ci/public>
		Require all granted
		AllowOverride All
		Options +Indexes
	</Directory>
</VirtualHost>
```

Virtual Host

Pour activer le virtual host on utilise la commande ***a2ensite*** et on redémarre le serveur apache

```
a2ensite codeigniter
systemctl restart apache2
```

Pour désactiver le virtual host on utilise la commande ***a2dissite*** et on redémarre le serveur apache

```
a2dissite codeigniter
systemctl restart apache2
```

Note : on peut aussi directement creer le fichier de configuration du virtual host dans le répertoire ***/etc/apache2/sites-enable***, puis redemarrer le serveur.

Virtual Host

Pour finir on doit enregistrer notre nouveau nom de domaine dans le fichier ***hosts***

```
sudo nano /etc/hosts
```

En y ajoutant la ligne suivante:

```
127.0.0.1         tuto.local
```

CodeIgniter

Afin que CodeIgniter puisse fonctionner, le module ***rewrite*** d'apache2 doit être activé les extensions php ***mbstring, curl, intl et xml*** doivent être installées:

```
apt install php-mbstring php-curl php-intl php-xml
```

```
a2enmod rewrite
```