

MANUEL TECHNIQUE ÉTUDIANT Atelier

Hébergement Web



MISSION 1 – Préparation

Objectif

Comprendre ce que vous allez mettre en place avant de manipuler un serveur.

Étape 1 – Définir ce qu'est un hébergement web

Rédige en 2 à 3 phrases ce que tu comprends de ces trois points :

- à quoi sert un hébergement web
- pourquoi un site ne peut pas exister sans serveur
- quel est le rôle d'un serveur web

Étape 2 – Comprendre les types d'hébergement

Explique en quelques lignes :

- Mutualisé
- VPS
- Dédié
- Cloud

Pour chaque type, indique :

- ce que c'est
- dans quel cas on l'utilise
- les avantages et les limites

Étape 3 – Faire un choix

Réponds à la question suivante : « Pour héberger une application web simple, quel type d'hébergement est le plus adapté ? Pourquoi ? »

MISSION 2 – Installation du serveur web

Tu vas maintenant installer un serveur web complet. Travaille lentement, étape par étape. Lis toujours deux fois les commandes avant d'appuyer sur Entrée.

Étape 2.1 – Se connecter au serveur

Dans ton terminal, tape :

```
ssh utilisateur@ADRESSE_IP
```

Remplace "utilisateur" par le nom fourni et "ADRESSE_IP" par l'adresse indiquée par ton formateur.

Si un message du type "Are you sure you want to continue connecting?" apparaît tape :

```
yes
```

Puis entre ton mot de passe (même si rien ne s'affiche, c'est normal).

Si tu arrives à une ligne qui se termine par "\$", tu es bien connecté.

Étape 2.2 – Mettre à jour le système

Tape :

```
sudo apt update  
sudo apt upgrade -y
```

Le système va télécharger et installer des mises à jour. Tu n'as rien à faire, attends la fin. Quand le terminal te redonne la main, l'étape est réussie.

Étape 2.3 – Installer Apache

Tape :

```
sudo apt install apache2 -y
```

L'installation défile pendant quelques secondes. Quand c'est terminé, tape :

```
sudo systemctl status apache2
```

Tu dois voir une ligne contenant :

```
active (running)
```

Si c'est le cas, Apache fonctionne. Pour quitter l'écran, appuie sur la touche :

```
q
```

Étape 2.4 – Installer le firewall UFW et autoriser les services nécessaires

Tape :

```
sudo apt install ufw -y  
sudo ufw allow 'OpenSSH'  
sudo ufw allow 'Apache'  
sudo ufw enable  
sudo ufw status
```

Après la commande `sudo ufw enable`, tape "y" si une confirmation est demandée.

Dans `sudo ufw status`, tu dois voir les lignes suivantes :

```
Apache ALLOW  
OpenSSH ALLOW
```

Étape 2.5 – Tester Apache dans le navigateur

Pour connaître l'IP du serveur, tape :

```
hostname -I
```

Tu verras une adresse du type :

```
192.168.1.75
```

Dans ton navigateur (Chrome, Firefox...), tape cette adresse dans la barre :

<http://192.168.1.75>

Tu devrais voir apparaître la page :

Apache2 Ubuntu Default Page

Si la page s'affiche, ton serveur web fonctionne.

Si elle ne s'affiche pas :

- vérifie que tu as bien autorisé Apache dans UFW
- vérifie que tu n'as pas fait d'erreur dans l'adresse IP
- vérifie que ton PC est sur le même réseau que le serveur

Étape 2.6 – Installer PHP

Tape :

```
sudo apt install php libapache2-mod-php php-mysql php-xml php-mbstring php-curl php-zip php-gd -y
```

Quand l'installation est terminée, vérifie la version :

```
php -v
```

Tu devrais voir une ligne du type :

```
PHP 8.1...
```

Étape 2.7 – Installer MySQL

Tape :

```
sudo apt install mysql-server -y
```

Puis sécurise l'installation :

```
sudo mysql_secure_installation
```

Le système pose plusieurs questions. Voici les réponses recommandées :

- Set root password : yes
- Remove anonymous users : yes
- Disallow root login remotely : yes
- Remove test database : yes
- Reload privilege tables : yes

Quand c'est terminé, tu reviens au prompt du terminal.

Étape 2.8 – Créer une base de données et un utilisateur

Tape :

```
sudo mysql
```

Tu entres dans le terminal MySQL (tu verras "mysql>"). Copie-colle les commandes suivantes :

```
CREATE DATABASE appweb DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;  
CREATE USER 'appuser'@'localhost' IDENTIFIED BY 'MotDePasseFort123!';  
GRANT ALL PRIVILEGES ON appweb.* TO 'appuser'@'localhost';  
FLUSH PRIVILEGES;  
EXIT;
```

Si tu reviens au prompt normal (avec le signe \$), l'opération est réussie.

MISSION 3 – Création d'un site web avec VirtualHost

Le but est de créer un dossier pour votre site, d'y mettre une page de test, puis de configurer Apache pour que ce site réponde à une adresse précise.

Étape 3.1 – Créer le dossier du site

Tape la commande suivante :

```
sudo mkdir -p /var/www/appweb
```

Ensuite, attribue le dossier à l'utilisateur du serveur web :

```
sudo chown -R www-data:www-data /var/www/appweb  
sudo chmod -R 755 /var/www/appweb
```

Tu ne devrais voir aucun message d'erreur. Si la commande ne renvoie rien, c'est normal : cela signifie qu'elle a fonctionné.

Étape 3.2 – Créer une page de test PHP

Ouvre un fichier dans l'éditeur nano :

```
sudo nano /var/www/appweb/index.php
```

Le fichier est vide. Tape ceci :

```
<?php  
phpinfo();
```

Pour enregistrer :

1. Appuie sur Ctrl + O
2. Appuie sur Entrée
3. Appuie sur Ctrl + X pour quitter

Tu dois revenir au terminal normal.

Étape 3.3 – Créer le fichier de configuration du site (VirtualHost)

Tape :

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

Le fichier est vide. Colle ce contenu :

```
<VirtualHost *:80>
    ServerName appweb.local
    DocumentRoot /var/www/appweb

    <Directory /var/www/appweb>
        Options Indexes FollowSymLinks
        AllowOverride All
        Require all granted
    </Directory>

    ErrorLog ${APACHE_LOG_DIR}/appweb_error.log
    CustomLog ${APACHE_LOG_DIR}/appweb_access.log combined
</VirtualHost>
```

Enregistre (Ctrl+O, Entrée) puis quitte (Ctrl+X).

Étape 3.4 – Activer le site

Tape :

```
sudo a2ensite appweb.conf
```

Tu devrais voir un message indiquant que le site est activé. Ensuite, désactive le site par défaut d'Apache :

```
sudo a2dissite 000-default.conf
```

Apache propose de recharger la configuration. Avant ça, teste la configuration :

```
sudo apache2ctl configtest
```

Tu dois absolument obtenir :

```
Syntax OK
```

Si tu vois une erreur, c'est que ton fichier .conf contient une faute (espace, balise mal fermée, erreur de chemin). Retourne l'éditer :

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

Quand tout est bon, recharge Apache :

```
sudo systemctl reload apache2
```

Aucune sortie = c'est bon.

Étape 3.5 – Tester le site dans le navigateur

Option A — Si tu utilises un vrai domaine

Tape dans le navigateur :

```
http://appweb.local
```

Option B — Si tu dois simuler le domaine

Sur ton ordinateur, édite ton fichier hosts :

Windows

Ouvre :

```
C:\Windows\System32\drivers\etc\hosts
```

Ajoute une ligne à la fin :

```
ADRESSE_IP_DU_SERVEUR appweb.local
```

Ensuite, dans ton navigateur :

```
http://appweb.local
```

Tu dois voir apparaître la page phpinfo().

Si la page ne s'affiche pas :

- vérifie l'adresse IP
- vérifie ton fichier hosts
- vérifie que tu as bien rechargé Apache
- vérifie que ton VirtualHost est activé

MISSION 4 – Sécurisation du serveur

Étape 4.1 – Configuration SSH avec clé

Tu vas empêcher la connexion au serveur par simple mot de passe. Pour ça, tu dois d'abord générer une clé SSH sur ton ordinateur, pas sur le serveur.

Dans ton terminal local :

```
ssh-keygen
```

Appuie sur Entrée à chaque question. Tu obtiens quelque chose comme :

```
Your identification has been saved in /home/user/.ssh/id_rsa
Your public key has been saved in /home/user/.ssh/id_rsa.pub
```

Maintenant, envoie ta clé publique sur le serveur :

```
ssh-copy-id utilisateur@ADRESSE_IP
```

Si tout est correct, le message dira que la clé a été copiée. Tu peux tester ensuite :

```
ssh utilisateur@ADRESSE_IP
```

Normalement, il ne te demande plus ton mot de passe.

Étape 4.2 – Désactiver l'accès SSH par mot de passe

Sur le serveur, ouvre la configuration sshd :

```
sudo nano /etc/ssh/sshd_config
```

Cherche les lignes suivantes (utilise les flèches du clavier) :

```
PasswordAuthentication yes
PermitRootLogin yes
```

Modifie-les ainsi :

```
PasswordAuthentication no
PermitRootLogin no
```

Enregistre (Ctrl+O, Entrée)

Quitte (Ctrl+X)

Recharge SSH :

```
sudo systemctl reload ssh
```

Aucune sortie = c'est correct.

Ne ferme pas ta session SSH tant que tu n'as pas vérifié que la connexion par clé fonctionne avec une deuxième session.

Étape 4.3 – HTTPS (si vous avez un domaine réel)

Si un domaine a été fourni, installe Certbot (outil de certificat automatique) :

```
sudo apt install certbot python3-certbot-apache -y
```

Ensuite :

```
sudo certbot --apache -d votredomaine.fr -d www.votredomaine.fr
```

Suis les instructions :

- accepte les conditions
- mets un email valable
- choisis l'option de redirection automatique HTTP → HTTPS

Test dans le navigateur :

```
https://votredomaine.fr
```

Tu dois voir un cadenas dans la barre d'adresse.

Si tu n'as pas de domaine, passe à la mission suivante.

MISSION 5 – Installation d'une application web (WordPress)

Dans cette mission, tu vas installer une application web complète sur ton serveur. WordPress sert ici d'exemple, car il utilise Apache, PHP et MySQL, ce qui permet de valider toutes les compétences d'hébergement.

AVEC WORDPRESS

Étape 5.1 – Télécharger WordPress

Tu vas travailler dans le dossier temporaire /tmp. Tape :

```
cd /tmp
curl -O https://wordpress.org/latest.tar.gz
tar -xzf latest.tar.gz
```

Le fichier se télécharge puis se décompresse automatiquement. À la fin, tu dois avoir un dossier /tmp/wordpress.

Tu peux vérifier :

```
ls /tmp
```

Tu dois voir apparaître "wordpress".

Étape 5.2 – Déployer WordPress dans ton site

Tu vas effacer la page phpinfo créée précédemment et mettre WordPress à la place. Attention : cette commande supprime tout le contenu du site. Tape :

```
sudo rm -rf /var/www/appweb/*
```

Puis copie WordPress :

```
sudo cp -R /tmp/wordpress/* /var/www/appweb/
```

Ensuite, applique les permissions nécessaires :

```
sudo chown -R www-data:www-data /var/www/appweb
sudo chmod -R 755 /var/www/appweb
```

Aucune sortie = c'est correct.

Étape 5.3 – Configurer WordPress

Tu vas créer le fichier de configuration principal. Tape :

```
cd /var/www/appweb
sudo cp wp-config-sample.php wp-config.php
sudo nano wp-config.php
```

Cherche les lignes suivantes et remplace-les par :

```
define( 'DB_NAME', 'appweb' );
define( 'DB_USER', 'appuser' );
define( 'DB_PASSWORD', 'MotDePasseFort123!' );
define( 'DB_HOST', 'localhost' );
```

Enregistre : Ctrl+O, Entrée

Quitte : Ctrl+X

Étape 5.4 – Lancer l'installation

Ouvre dans ton navigateur :

```
http://appweb.local
```

ou si tu as un vrai domaine :

```
https://votredomaine.fr
```

L'assistant WordPress apparaît. Suis les étapes :

- choisir une langue
- entrer un titre de site
- créer un identifiant administrateur
- un mot de passe
- un email

Quand tu arrives sur le tableau de bord, l'installation est réussie.

Étape 5.5 – Vérifications essentielles

Tu dois pouvoir :

- te connecter à l'admin
- créer un article
- créer une page
- voir la page d'accueil du site

Ces tests valident que la base de données, PHP et Apache fonctionnent correctement ensemble.

Avec votre projet Qit sur le serveur

Dans cette mission, vous allez mettre en ligne votre propre projet récupéré depuis votre dépôt Git. Vous suivrez les étapes dans l'ordre, comme un vrai administrateur système.

Étape 5.1 – Récupérer votre projet depuis Qit

Dans votre serveur, allez dans le dossier /tmp :

```
cd /tmp
```

Clonez votre dépôt :

```
git clone URL_DE_VOTRE_DEPOT
```

Remplacez URL_DE_VOTRE_DEPOT par l'adresse HTTPS de votre projet. Par exemple :

```
git clone https://github.com/nom/projet.git
```

Si tout est correct, Git affiche :

```
Cloning into 'projet'...
```

Puis une liste de fichiers téléchargés. Si vous voyez un message d'erreur, vérifiez l'URL.

Étape 5.2 – Copier votre projet dans le dossier du site

Effacez le contenu actuel du site :

```
sudo rm -rf /var/www/appweb/*
```

Ensuite, copiez vos fichiers :

```
sudo cp -R /tmp/NOM_DU_DOSSIER/* /var/www/appweb/
```

Listez le dossier pour vérifier :

```
ls /var/www/appweb
```

Vous devez voir vos fichiers (index, css, js, images, etc.).

Étape 5.3 – Appliquer les permissions

Tapez :

```
sudo chown -R www-data:www-data /var/www/appweb
sudo chmod -R 755 /var/www/appweb
```

Si aucune erreur n'apparaît, c'est bon.

Étape 5.4 – Vérifier le type de projet et adopter le déploiement

Cas 1 : Projet HTML / CSS / JS

Rien d'autre à faire. Vos fichiers doivent s'afficher directement dans le navigateur.

Cas 2 : Projet PHP

Vérifiez que vous avez un fichier index.php ou un fichier principal. Tapez :

```
ls /var/www/appweb
```

Si vous voyez un index.php :

- Apache va l'exécuter automatiquement
- Si vous voyez une erreur "500", allez lire les logs (Mission 7)

Cas 3 : Projet Node.js

Le site ne peut pas tourner directement avec Apache. Vous devez lancer votre application manuellement :

```
cd /var/www/appweb
npm install
npm start
```

Le terminal doit afficher quelque chose comme :

```
Server running on port 3000
```

Ensuite, pour exposer votre app dans un VirtualHost Apache, il faudra utiliser un proxy (option avancée). Tu me dis si tu veux que je l'intègre, Alex.

Étape 5.5 – Tester dans le navigateur

Tapez dans votre navigateur :

```
http://appweb.local
```

ou si vous utilisez l'IP :

```
http://ADRESSE_IP
```

Selon votre projet :

- un site HTML doit s'afficher immédiatement
- un site PHP doit s'afficher index.php existe
- une erreur 500 indique un problème PHP (voir les logs)
- une page blanche signifie souvent un chemin incorrect ou un fichier manquant

Étape 5.6 – Debugger en cas d'erreur

Si le site ne s'affiche pas :

1. Vérifier les fichiers

```
ls /var/www/appweb
```

Le fichier principal (index.html ou index.php) doit être présent.

2. Vérifier les logs Apache

```
sudo tail -f /var/log/apache2/error.log
```

Lisez les dernières lignes pour identifier l'erreur. Arrêtez l'affichage avec Ctrl + C.

3. Vérifier les permissions

```
ls -l /var/www/appweb
```

Propriétaire attendu : www-data www-data

Si nécessaire :

```
sudo chown -R www-data:www-data /var/www/appweb
```

Étape 5.7 – Résultat attendu

À la fin de la mission, vous devez avoir :

- votre projet accessible depuis un navigateur
- aucun message d'erreur bloquant
- des permissions correctes
- une compréhension du cycle : Git → serveur → configuration → affichage

MISSION 6 – Architecture Haute Disponibilité

Tu vas maintenant comprendre comment on structure un hébergement professionnel pour garantir la disponibilité. Il ne s'agit pas d'installer quelque chose, mais de concevoir et représenter une architecture complète.

Étape 6.1 – Faire un schéma

Sur papier ou sur un outil numérique (Excalidraw, Whimsical, etc.), dessine les éléments suivants :

1. Des utilisateurs (navigateurs)
2. Un point d'entrée appelé "load balancer"
3. Deux serveurs web identiques
4. Une base de données répliquée (master → replica)
5. Un stockage partagé ou synchronisé pour les fichiers du site (optionnel)

Le schéma doit montrer comment les requêtes circulent.

Étape 6.2 – Rédiger deux explications

Texte 1 : Explique comment cette architecture continue de fonctionner même si un serveur tombe en panne.

Texte 2 : Explique comment elle peut absorber plus de trafic si le nombre d'utilisateurs augmente.

10 lignes au total suffisent.

MISSION 7 – Logs et Permissions Linux

Cette mission est obligatoire pour valider les compétences systèmes.

Étape 7.1 – Lire les logs d'Apache

Tape :

```
sudo tail -f /var/log/apache2/error.log
```

Tu vas voir les erreurs en direct. Pour arrêter l'affichage, appuie sur :

```
Ctrl + C
```

Étape 7.2 – Lire les logs de MySQL

Tape :

```
sudo tail -f /var/log/mysql/error.log
```

Même fonctionnement que pour Apache. Arrête : Ctrl+C.

Étape 7.3 – Voir les logs systèmes d'Apache

Tape :

```
journalctl -u apache2 --since "15 minutes ago"
```

Tu vois toutes les actions enregistrées par le service Apache.

Étape 7.4 – Comprendre les permissions Linux

Tape :

```
ls -l /var/www/appweb
```

Observe les colonnes. Le propriétaire doit être :

```
www-data www-data
```

Les droits doivent ressembler à :

```
drwxr-xr-x
```

Rappels importants :

- chmod modifie les droits
- chown modifie le propriétaire et le groupe

Tu peux tester :

```
sudo chown www-data:www-data /var/www/appweb  
sudo chmod 755 /var/www/appweb
```

MISSION 8 – Hardening Basique du Serveur

Le but ici est de réduire les informations affichées par le serveur et de renforcer le firewall.

Étape 8.1 – Modifier la configuration de sécurité d'Apache

Ouvre le fichier :

```
sudo nano /etc/apache2/conf-available/security.conf
```

Cherche ces lignes :

```
ServerSignature On  
ServerTokens Full
```

Remplace par :

```
ServerSignature Off  
ServerTokens Prod
```

Enregistre (Ctrl+O) puis quitte (Ctrl+X).

Recharge Apache :

```
sudo systemctl reload apache2
```

Étape 8.2 – Renforcer le firewall

Tape :

```
sudo ufw default deny incoming  
sudo ufw default allow outgoing  
sudo ufw status
```

Vérifie que les règles affichées correspondent bien aux services autorisés.