System Setup & Lab Configuration

Stage PFE – Entreprise **VOID**

Stagiaire : Hibat Allah Rguiti

Encadrant : Hamza Bahlaouane

06 février 2026

1. Chapter 1 : Setup

1.1. Vagrant and VMware

Vagrant est un outil qui permet de créer et gérer facilement des machines virtuelles (VM) pour le développement. Il permet de :

— Créer une VM en une seule commande.
— La configurer automatiquement.
— Partager cette configuration avec toute l'équipe.

1.2. Vagrantfile

C'est le fichier de configuration principal pour Vagrant. Il décrit :

— Le système d'exploitation à installer (Ubuntu, Debian...)
— La RAM et le CPU de la VM
— Le réseau
— Les scripts à exécuter lors du provisioning

1.3. Commandes utilisées

```
mkdir -p ~/dev/labs/vm
cd ~/dev/labs/vm
curl -o Vagrantfile https://gist.githubusercontent.com/Bahlaouane-Hamza/.../
    Vagrantfile
vagrant up --provider vmware_desktop
```

1.4. Remarque

La VM aura une IP interne, par exemple : `192.168.33.10`. Le Mac pourra accéder à la VM via cette adresse.

Pour provoquer une erreur volontaire avec Apache plus tard :

```
sudo nohup nc -l -p 80 &
```

2. Chapter 2 : SSH

2.1. Connexion SSH

SSH (Secure Shell) est un protocole qui permet de se connecter à distance à un serveur de manière sécurisée.

2.2. Méthodes de connexion

1. Utilisation du mot de passe :

```
ssh alpha@127.0.0.1
```

2. Utilisation de la clé générée par Vagrant :

```
ssh -i /path/to/private_key alpha@127.0.0.1 -p 2222
```

3. Approche **Passwordless** :

```
ssh-keygen -t rsa -b 4096 -f ~/.ssh/my_key
scp -P 2222 ~/.ssh/my_key.pub alpha@127.0.0.1:~/.ssh/authorized_keys
ssh alpha@127.0.0.1 -i ~/.ssh/my_key
```

2.3. Commande id

La commande `id` permet de connaître :

— UID (User ID)

— GID (Group ID)

— Groupes dont l'utilisateur fait partie

3. Chapter 3 : SSL/TLS Certificates and Errors

SSL/TLS certificates are used to secure communication between clients and servers over HTTPS. They ensure authenticity, confidentiality, and integrity of the data.

3.1. Using curl to Access URLs

We can use `curl` in verbose mode (`-v`) to observe SSL/TLS errors :

```
curl -v https://expired.badssl.com/
curl -v https://self-signed.badssl.com/
```

Observations :

— **expired.badssl.com** :

SSL certificate problem : certificate has expired

Explanation : The certificate was issued by a trusted CA but its validity period has ended.

— **self-signed.badssl.com** :

SSL certificate problem : self signed certificate

Explanation : The certificate is not signed by a trusted CA. Clients do not trust it by default.

3.2. Inspecting Certificates with openssl

```
openssl s_client -connect expired.badssl.com:443
openssl s_client -connect self-signed.badssl.com:443
```

Check for :

— `subject` – Owner of the certificate

— `issuer` – Who issued the certificate

— `Not Before / Not After` – Validity period

— Warnings about self-signing or expiration

Example observations :

— **Expired certificate :**

notBefore=Mar 10 00 :00 :00 2015 GMT
notAfter=Mar 10 23 :59 :59 2016 GMT

— **Self-signed certificate :**

issuer= /CN=self-signed.badssl.com

3.3. Fixing SSL/TLS Issues

1. **Expired certificate :** Renew the certificate with a trusted Certificate Authority (CA), e.g., using Let's Encrypt.
2. **Self-signed certificate :**
   — Replace it with a certificate signed by a trusted CA.
   — For testing purposes only, you can add the self-signed certificate to your client's trusted store.

3.4. Summary Table

URL	Error Type	Explanation	Fix
expired.badssl.com	Expired certificate	Validity period ended	Renew certificate with CA
self-signed.badssl.com	Self-signed	Not trusted by client	Use CA-signed certificate

4. Chapter 4 : Cronjob

Un **cronjob** est une tâche planifiée sur un système Linux/Unix qui s'exécute automatiquement à intervalles réguliers (minutes, heures, jours, etc.). Il est géré par le démon `cron` et permet d'automatiser des tâches répétitives comme les sauvegardes, le nettoyage de fichiers ou l'envoi de rapports.

4.1. Objectif du Cronjob

L'objectif de ce cronjob est de supprimer tous les fichiers âgés de plus de 7 jours dans le répertoire `/temp`.

4.2. Création des fichiers et du répertoire de test

```
mkdir ~/temp
touch ~/temp/file{1..10}.txt
```

4.3. Script clean_temp.sh

```
#!/bin/bash
LOGFILE=~/temp/clean_temp.log
echo "$(date): Script started" >> $LOGFILE
find ~/temp -type f -mtime +7 -delete 2>> $LOGFILE
echo "$(date): Script finished" >> $LOGFILE
```

Explications :

— `-mtime +7` : sélectionne les fichiers modifiés il y a plus de 7 jours.

— `-delete` : supprime les fichiers trouvés.

— Les messages et erreurs sont enregistrés dans le fichier `clean_temp.log`.

4.4. Planification avec cron

Pour exécuter le script tous les jours à minuit :

```
0 0 * * * /home/username/clean_temp.sh
```

4.5. Problème rencontré

Si les fichiers viennent juste d'être créés, le cronjob ne les supprime pas car `-mtime +7` ne correspond à aucun fichier. Exemple de test :

```
./clean_temp.sh
ls ~/temp
```

Résultat possible : les fichiers restent présents.

4.6. Solutions pour tester immédiatement

1. **Supprimer tous les fichiers sans condition** :

   ```
   find ~/temp -type f -delete
   ```

2. **Simuler des fichiers anciens** :

   ```
   touch -d "8 days ago" ~/temp/file*.txt
   ./clean_temp.sh
   ls ~/temp
   ```

   Cela rend les fichiers "âgés de 8 jours" et le script les supprime normalement.

4.7. Vérification et logging

Pour vérifier si le cronjob s'exécute correctement, consulter le fichier de log :

```
cat ~/temp/clean_temp.log
```

Ou vérifier les logs système du cron :

```
tail -f /var/log/syslog | grep cron
journalctl -u cron -f
```

Remarque : Si le script n'est pas exécutable (`chmod +x clean_temp.sh`), le cronjob échouera et enregistrera une erreur dans le log.

5. Chapter 5 : Permissions

5.1. Sudo pour utilisateur spécifique

Pour donner accès à des commandes spécifiques sans accorder tous les privilèges :

```
sudo visudo
# Ajouter une ligne :
alpha ALL=(ALL) NOPASSWD: /sbin/ifconfig
```

5.2. Permissions de dossier

Si vous exécutez :

```
curl -o /opt/bat.zip https://github.com/sharkdp/bat/archive/refs/tags/v0
    .24.0.zip
```

Vous obtenez une erreur `Permission denied` car `/opt` appartient à root.

5.2.1 Solution

Changer le propriétaire :

```
sudo chown alpha /opt/
```

Extraire le fichier zip (tar utilisé ici car unzip n'est pas installé) :

```
tar -xf /opt/bat.zip -C /opt/
ls -lh /opt/bat.zip
```

6. Chapter 6 : Webserver

6.1. Installation d'Apache

Apache est un serveur web qui permet de servir des pages HTML sur le réseau.

```
vagrant ssh
sudo su -
sudo apt-get install -y apache2
sudo systemctl start apache2
```

6.2. Dépannage

Si Apache ne démarre pas, utilisez :

```
systemctl status apache2.service
journalctl -xe
lsof -i :80
sudo ss -tulpn | grep 80
sudo kill <PID>
```

6.3. Vérification

Accédez au serveur web via le navigateur pour vérifier que la page HTML s'affiche correctement.

Solution de Forbidden : Modifier le fichier sudo nano /etc/apache2/sites-enabled/000-default.conf

7. Sources

— Vagrant Tutorial on YouTube