Vagrant & SSH Lab

Virtual Machine Management and System Administration

Sprint 2 : System Setup and Lab Configuration

Réalisé par :
Karrouach Ansar

Encadré par :
Hamza bahlaouane

9 février 2026

Table des matières

1. Setup

Objective : Download and start a VM using VMware Desktop and Vagrant.

Steps Completed :

1. Create project directory :

```
mkdir -p ~/dev/labs/vm
cd ~/dev/labs/vm
```

2. Download Vagrantfile :

```
curl -o Vagrantfile https://gist.githubusercontent.com/.../Vagrantfile
```

3. Install VMware Vagrant plugin :

```
vagrant plugin install vagrant-vmware-desktop
```

4. Start the VM :

```
vagrant up --provider vmware_desktop
```

Result : Debian 11 VM created and running with Apache installed by the provisioning script.

2. SSH

Objective : Create a user and enable passwordless SSH access.

Steps Completed :

1. SSH into VM and create user :

```
vagrant ssh
sudo adduser alpha # set password: 123456
id
exit
```

2. Verify SSH details (example alternatives : forwarded or private network) :

```
vagrant ssh-config # shows port and identity info
# or connect directly if using private_network (192.168.33.10)
ssh -p 22 alpha@192.168.33.10
```

3. Generate SSH key pair on host and copy key :

```
ssh-keygen -t rsa -b 4096
scp -P 22 ~/.ssh/id_rsa.pub alpha@192.168.33.10:~/.ssh/authorized_keys
# or:
ssh-copy-id -p 22 alpha@192.168.33.10
```

4. Fix permissions and test :

```
ssh alpha@192.168.33.10 "mkdir -p ~/.ssh && chmod 700 ~/.ssh && chmod 600 ~/.
    ssh/authorized_keys"
ssh-add ~/.ssh/id_rsa
ssh -p 22 alpha@192.168.33.10 # should connect without password
```

Result : User `alpha` exists and passwordless SSH works from the host.

3. SSL

Objective : Inspect TLS issues and demonstrate how to fix an expired certificate.

Steps Completed :

1. Inspect sites with `curl` and `openssl` :

```
curl -v https://expired.badssl.com/
curl -v https://self-signed.badssl.com/
openssl s_client -connect expired.badssl.com:443
```

Explanation : `curl` shows the client-side error ; `openssl` reveals certificate details and chain.

2. Fix an expired certificate (example using Let's Encrypt) :

```
sudo apt-get update
sudo apt-get install -y certbot python3-certbot-apache
sudo certbot certonly --standalone -d example.com
# verify:
openssl x509 -in /etc/letsencrypt/live/example.com/cert.pem -text -noout |
    grep -A2 "Validity"
```

Explanation : Obtain a valid certificate from a trusted CA and verify its validity dates.

Result : Identified expired and self-signed certs ; documented steps to renew using Certbot and verify.

4. Cronjob

Objective : Automatically delete files older than 7 days from ~/temp and log actions.

Steps Completed :

1. Create test directory and files :

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

2. Create cleanup script /clean_temp.sh :

```
#!/bin/bash
LOG=/home/alpha/clean_temp.log
echo "Run: $(date -u)" >> "$LOG"
find /home/alpha/temp -type f -mtime +7 -print -delete >> "$LOG" 2>&1
echo "-----" >> "$LOG"
```

Explanation : Logs timestamp and filenames of deleted files for auditing.

3. Make script executable and add cron entry :

```
chmod +x /home/alpha/clean_temp.sh
# edit crontab (example daily at 10:42):
42 10 * * * /home/alpha/clean_temp.sh >> /home/alpha/clean_temp.log 2>&1
```

4. Test immediately :

```
touch ~/temp/oldfile.txt && touch -d '8 days ago' ~/temp/oldfile.txt
/home/alpha/clean_temp.sh
ls -la ~/temp
tail -n 50 /home/alpha/clean_temp.log
```

4.1 Verification Commands

Run these commands on the VM to verify the cleanup script and cron job. Each command is one-line and shows the expected result in the comment.

1. Make script executable and check permissions :

```
chmod +x ~/clean_temp.sh
ls -l ~/clean_temp.sh # expect: -rwx... /home/alpha/clean_temp.sh
```

2. Create an old test file and run the script :

```
touch ~/temp/oldfile.txt && touch -d '8 days ago' ~/temp/oldfile.txt
/home/alpha/clean_temp.sh
ls -la ~/temp # expect: oldfile.txt is not listed
```

3. Check the log for run entries :

```
tail -n 50 /home/alpha/clean_temp.log
# expect: "Run: <timestamp>" and deleted filename(s)
```

4. Show what would be deleted (dry-run, if needed) :

```
find /home/alpha/temp -type f -mtime +7 -print
# expect: list of files older than 7 days (before deletion)
```

5. Confirm crontab contains the scheduled job :

```
crontab -l
# expect: "42 10 * * * /home/alpha/clean_temp.sh >> /home/alpha/clean_temp.
    log 2>&1"
```

6. Verify cron service and cron execution logs :

```
sudo systemctl status cron
sudo grep CRON /var/log/syslog | tail -n 50
```

7. Check VM time to ensure cron alignment :

```
date '+%Y-%m-%d %H:%M:%S'
```

Result : Cron job scheduled ; manual tests show cleanup works and logging confirms runs.

5. Permissions

Objective : Grant limited sudo access to `ifconfig` for `alpha` and handle file permissions for placing files in `/opt`.

Steps Completed :

1. Grant limited sudo access :

```
sudo visudo
# add: alpha ALL=(ALL) NOPASSWD: /sbin/ifconfig
su - alpha
sudo ifconfig # works without password
sudo apt-get update # fails (permission denied)
```

Explanation : Modifies `/etc/sudoers` to allow `ifconfig` only, limiting admin privileges.

2. Handle file permissions for `/opt` :

```
curl -o /opt/bat.zip https://github.com/sharkdp/bat/archive/refs/tags/v0
    .24.0.zip
# error: Permission denied ( /opt owned by root)
sudo chown alpha /opt
su - alpha
curl -o /opt/bat.zip https://github.com/sharkdp/bat/archive/refs/tags/v0
    .24.0.zip
# or download on host and scp:
# on Mac: curl -o ~/bat.zip <url>
# scp -P 22 ~/bat.zip alpha@192.168.33.10:/opt/bat.zip
python3 -c "import zipfile; zipfile.ZipFile('/opt/bat.zip').extractall('/opt
    /')"
```

Explanation : Changes ownership to allow write access without sudo in the placement command ; uses Python to extract zip without installing unzip.

Result : Limited sudo configured ; file placed in `/opt` and extracted using alternative methods.

6. Webserver

Objective : Set up and configure Apache webserver, resolve port conflicts, and enable external access.

Steps Completed :

1. Apache was pre-installed via Vagrant provisioning, but port 80 was occupied by netcat :

```
ps aux | grep nc
sudo kill 11403 # Kill the netcat process
sudo systemctl restart apache2
```

Explanation : The provisioning script started netcat on port 80, preventing Apache from binding.

2. Verify Apache status and test local access :

```
sudo systemctl status apache2
curl http://localhost # Returns custom HTML page
```

3. Edit Apache config to allow external access (remove denying VirtualHost for 192.168.33.10) :

```
sudo vim /etc/apache2/sites-enabled/000-default.conf
# Comment out or remove:
# <VirtualHost 192.168.33.10:80>
# <Location />
# Require all denied
# </Location>
# </VirtualHost>
sudo systemctl reload apache2
```

Explanation : The config had a VirtualHost denying access to the private network IP ; removing it allows external requests.

4. Test external access from host :

```
# From Mac: Open http://192.168.33.10/ in browser
# Should display the custom "Developer Tools Workspace" page
```

Result : Apache webserver running and accessible externally at `http://192.168.33.10/`, serving the custom HTML page with JS-rendered content.