Sprint 2 – Server Setup and DevOps Basics

Hatim Salhi

February 9, 2026

Contents

1 Setup

- Installed VMware Fusion and Vagrant with the VMware plugin.
- Created a project directory: `/dev/labs/vm`.
- Downloaded and configured the Vagrantfile for Debian 11.
- Started the VM using: `vagrant up -provider vmware_desktop`.

1.1 Screenshots

```
[void@MacBook-Pro-de-void vm % vagrant status                                    ]
Current machine states:

default                   running (vmware_desktop)

The VM is running. To stop this VM, you can run `vagrant halt` to
shut it down, or you can run `vagrant suspend` to simply suspend
the virtual machine. In either case, to restart it again, run
`vagrant up`.
[void@MacBook-Pro-de-void vm % vagrant ssh-config                                ]
Host default
  HostName 127.0.0.1
  User vagrant
  Port 2222
  UserKnownHostsFile /dev/null
  StrictHostKeyChecking no
  PasswordAuthentication no
  IdentityFile /Users/void/dev/labs/vm/.vagrant/machines/default/vmware_desktop/
private_key
  IdentitiesOnly yes
  LogLevel FATAL
  PubkeyAcceptedKeyTypes +ssh-rsa
  HostKeyAlgorithms +ssh-rsa
```

Figure 1: VMware + Vagrant setup

2 SSH

- Verified SSH access to the VM.
- Created a new user `alpha`.
- Configured passwordless SSH using key pairs.
- Verified key-based login: `ssh alpha@192.168.209.128`.

2.1 Screenshots

```
void@MacBook-Pro-de-void vm % ssh-keygen -t rsa -b 4096 -C "alpha@127.0.0.1"
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/void/.ssh/id_rsa):
Created directory '/Users/void/.ssh'.
Enter passphrase for "/Users/void/.ssh/id_rsa" (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /Users/void/.ssh/id_rsa
Your public key has been saved in /Users/void/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:nyHvAefwRlWnTF9h8FLXqK67by5rdMy+QBPIz+akXYk alpha@127.0.0.1
The key's randomart image is:
+---[RSA 4096]----+
|             .+=*|
|       . .   +=++|
|        o . .oo..|
|         o +...  |
|        S E+o    |
|         ^.==    |
|        ..%+     |
|         o+oo    |
|         .=O+.   |
+----[SHA256]-----+
```

Figure 2: SSH login and key configuration

3 SSL

- Tested SSL/TLS scenarios using `curl -v` to badssl.com.
- Observed expired and self-signed certificate errors.
- Used `openssl s_client -connect <host>:443` to inspect certificates.
- Learned how to fix SSL issues (renew certificates, use trusted CA).

3.1 Screenshots

```
void@MacBook-Pro-de-void ~ % ccurl -v https://expired.badssl.com/
zsh: command not found: ccurl
void@MacBook-Pro-de-void ~ % curl -v https://expired.badssl.com/
* Host expired.badssl.com:443 was resolved.
* IPv6: (none)
* IPv4: 104.154.89.105
*   Trying 104.154.89.105:443...
* Connected to expired.badssl.com (104.154.89.105) port 443
* ALPN: curl offers h2,http/1.1
* (304) (OUT), TLS handshake, Client hello (1):
*  CAfile: /etc/ssl/cert.pem
*  CApath: none
* (304) (IN), TLS handshake, Server hello (2):
* TLSv1.2 (IN), TLS handshake, Certificate (11):
* TLSv1.2 (OUT), TLS alert, certificate expired (557):
* SSL certificate problem: certificate has expired
* Closing connection
* TLSv1.2 (IN), TLS handshake, Certificate (11):
* TLSv1.2 (OUT), TLS alert, certificate expired (557):
curl: (60) SSL certificate problem: certificate has expired
More details here: https://curl.se/docs/sslcerts.html

curl failed to verify the legitimacy of the server and therefore could not
establish a secure connection to it. To learn more about this situation and
how to fix it, please visit the web page mentioned above.
```

Figure 3: SSL certificate inspection

```
[void@MacBook-Pro-de-void ~ % curl -v https://self-signed.badssl.com/
* Host self-signed.badssl.com:443 was resolved.
* IPv6: (none)
* IPv4: 104.154.89.105
*   Trying 104.154.89.105:443...
* Connected to self-signed.badssl.com (104.154.89.105) port 443
* ALPN: curl offers h2,http/1.1
* (304) (OUT), TLS handshake, Client hello (1):
*  CAfile: /etc/ssl/cert.pem
*  CApath: none
* (304) (IN), TLS handshake, Server hello (2):
* TLSv1.2 (IN), TLS handshake, Certificate (11):
* TLSv1.2 (OUT), TLS alert, unknown CA (560):
* SSL certificate problem: self signed certificate
* Closing connection
* TLSv1.2 (IN), TLS handshake, Certificate (11):
* TLSv1.2 (OUT), TLS alert, unknown CA (560):
curl: (60) SSL certificate problem: self signed certificate
More details here: https://curl.se/docs/sslcerts.html

curl failed to verify the legitimacy of the server and therefore could not
establish a secure connection to it. To learn more about this situation and
how to fix it, please visit the web page mentioned above.
```

Figure 4: SSL certificate inspection

```
depth=0 C = US, ST = California, L = San Francisco, O = BadSSL Fallback. Unknown subdom
ain or no SNI., CN = badssl-fallback-unknown-subdomain-or-no-sni
verify error:num=20:unable to get local issuer certificate
verify return:1
depth=0 C = US, ST = California, L = San Francisco, O = BadSSL Fallback. Unknown subdom
ain or no SNI., CN = badssl-fallback-unknown-subdomain-or-no-sni
verify error:num=21:unable to verify the first certificate
verify return:1
write W BLOCK
---
Certificate chain
 0 s:/C=US/ST=California/L=San Francisco/O=BadSSL Fallback. Unknown subdomain or no SNI
./CN=badssl-fallback-unknown-subdomain-or-no-sni
   i:/C=US/ST=California/L=San Francisco/O=BadSSL/CN=BadSSL Intermediate Certificate Au
thority
---
Server certificate
-----BEGIN CERTIFICATE-----
MIIE8DCCAtigAwIBAgIJAM28Wkrsl2exMA0GCSqGSIb3DQEBCwUAMH8xCzAJBgNV
BAYTAlVTMRMwEQYDVQQIDApDYWxpZm9ybmlhMRYwFAYDVQQHDA1TYW4gRnJhbmNp
c2NvMQ8wDQYDVQQKDAZCYWRTU0wxMjAwBgNVBAMMKUJhZFNTTCBJbnRlcm1lZGlh
dGUgQ2VydGlmaWNhdGUgQXV0aG9yaXR5MB4XDTE2MDgwODIxMTcwNVoXDTE4MDgw
ODIxMTcwNVowgagxCzAJBgNVBAYTAlVTMRMwEQYDVQQIDApDYWxpZm9ybmlhMRYw
FAYDVQQHDA1TYW4gRnJhbmNpc2NvMTYwNAYDVQQKDC1CYWRTU0wgRmFsbGJhY2su
```

Figure 5: SSL certificate inspection

```
-----END CERTIFICATE-----
subject=/C=US/ST=California/L=San Francisco/O=BadSSL Fallback. Unknown subdomain or no
SNI./CN=badssl-fallback-unknown-subdomain-or-no-sni
issuer=/C=US/ST=California/L=San Francisco/O=BadSSL/CN=BadSSL Intermediate Certificate
Authority
---
No client certificate CA names sent
Server Temp Key: ECDH, P-256, 256 bits
---
SSL handshake has read 1938 bytes and written 413 bytes
---
New, TLSv1/SSLv3, Cipher is ECDHE-RSA-AES128-GCM-SHA256
Server public key is 2048 bit
Secure Renegotiation IS supported
Compression: NONE
Expansion: NONE
No ALPN negotiated
SSL-Session:
    Protocol  : TLSv1.2
    Cipher    : ECDHE-RSA-AES128-GCM-SHA256
    Session-ID: 8F773FB0AFD9485E144666FF34F809BFD57665A35E99497EFC41BE2B7B9F936B
    Session-ID-ctx:
    Master-Key: 21ABBB8280A06DB1671B8D85A066450919A4D1AF4EBDD35F81301D3F11379618ED56455
3716E96825B75BCDF6F5F2A07
    TLS session ticket lifetime hint: 300 (seconds)
    TLS session ticket:
    0000 - 49 c6 14 71 f7 0e f3 97-04 35 53 c3 da 22 db 2b   I..q.....5S..".+
    0010 - a8 77 6d 5c d0 61 44 40-4b 4b 93 e1 a9 3e 6e a8   .wm\.aD@KK...>n.
    0020 - 43 f4 07 75 8b c0 40 0c-b4 46 72 2b 56 14 39 9d   C..u..@..Fr+V.9.
    0030 - 51 15 a2 4f b6 d1 ab 54-7f 04 ae bd 42 df b7 e4   Q..O...T....B...
    0040 - dd 2e f0 ca 50 47 f9 63-ae 88 50 9c 1b 06 5e 2c   ....PG.c..P...^,
    0050 - 65 9d 68 0f a0 55 54 82-97 c6 71 a1 8a 58 05 fe   e.h..UT...q..X..
    0060 - 95 69 cc 6a 3f 78 c8 66-98 04 95 be b7 bd 81 a4   .i.j?x.f........
```

4 Cronjob

- Scheduled a cronjob to delete files older than 7 days in `/temp`.
- Created a script: `clean_temp.sh` with `find /temp -type f -mtime +7 -delete`.
- Tested script execution and verified cron scheduling.
- Added logging for errors.

4.1 Screenshots

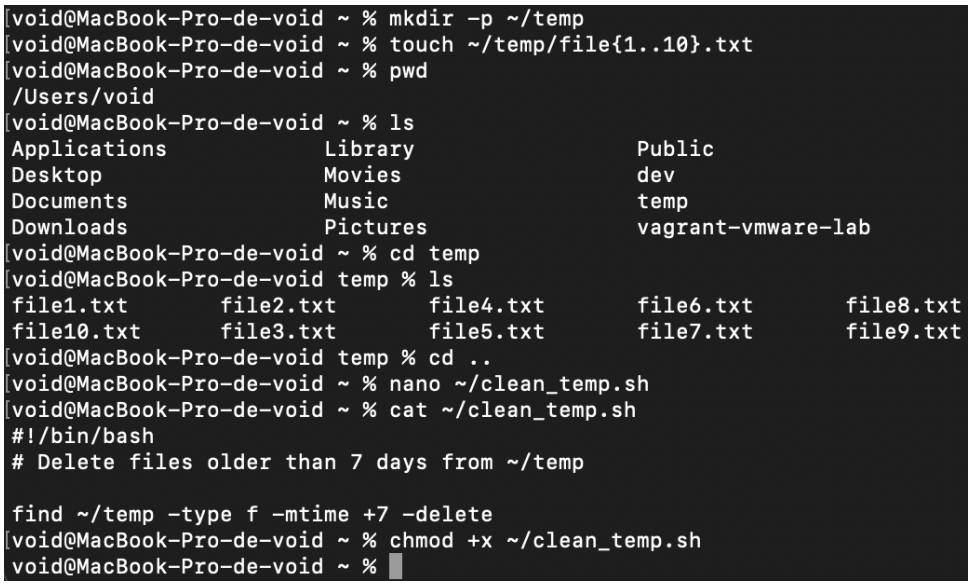

Figure 7: Cronjob and script execution

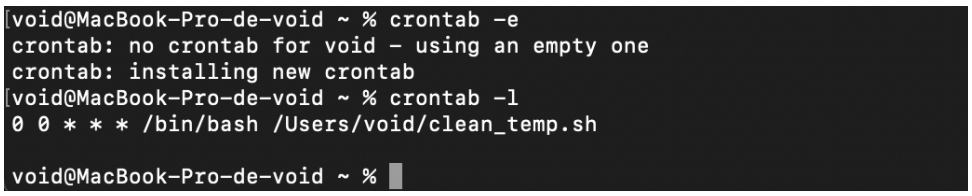

Figure 8: Cronjob and script execution

5 Permissions

- Created user-specific sudo permissions for `alpha`.
- Ensured limited access to commands (e.g., `ifconfig`) without full root privileges.
- Handled file ownership and directory permissions in `/opt`.
- Downloaded and extracted files without sudo errors.

5.1 Screenshots

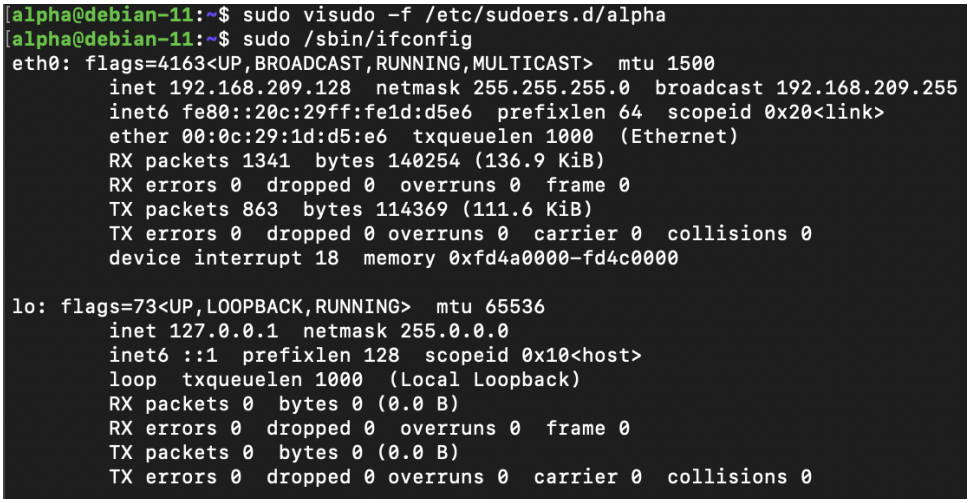

Figure 9: User permissions and folder setup

6 Webserver

- Installed Apache web server using `sudo apt-get install -y apache2`.
- Started the service and resolved conflicts using `lsof` or `ss -tulpn`.
- Verified web access via browser.
- Observed default HTML page listing apps for the environment.

7 Conclusion

- The server environment is now fully set up for development and testing.
- SSH and SSL practices ensure secure access and encrypted communication.
- Cronjobs and permissions are properly configured for automation and user-specific tasks.
- Apache webserver is operational and ready for hosting applications.