

TUNISIAN REPUBLIC

**MINISTRY OF
HIGHER EDUCATION
AND SCIENTIFIC RESEARCH**



UNIVERSITY OF SFAX

**FACULTY OF SCIENCES
OF SFAX**

**DEPARTMENT OF
COMPUTER SCIENCE AND
COMMUNICATIONS**

Multi-Site Distributed Virtualized Infrastructure Project

First Year of Engineering Cycle

Realized by:

Nour El Houda Ben Slimène

Louay Ben Rjab

Supervised by:

Mr. Oussama Besbes

Academic Year 2025–2026

Contents

1	Introduction	3
1.1	Project Context	3
1.2	Educational Objectives	3
2	Infrastructure Architecture	4
2.1	Global Network Topology	4
2.2	Data Center Analysis	4
2.2.1	Data Center A (Management and Core Services)	4
2.2.2	Data Center B (Production and Edge Services)	4
3	Simulation of the WAN via VPN : Implementation with Tailscale	5
3.1	Technical Rationale	5
3.2	Technical Implementation Steps	5
4	Network Configuration and Inter-Site Routing	7
4.1	Static IP Addressing Plan	7
4.2	Network Interface Configuration (Netplan)	7
4.3	Inter-Network Routing Setup	8
5	Deployment of Services	9
5.1	Initialization of the Docker Swarm Cluster	9
5.2	Deployment of the Application Stack	10
5.3	NGINX Reverse Proxy Configuration and Verification	11
6	Testing	12
6.1	Network Connectivity Routing Validation	12
6.1.1	Local LAN Connectivity (Internal)	12
6.1.2	VPN Tunnel Establishment (WAN Simulation)	12
6.1.3	Inter-Site ICMP Echo Requests (End-to-End Ping)	13
6.1.4	Routing Path Analysis (Traceroute)	13
6.2	DNS Resolution Test	13
6.2.1	DNS Server Reachability	13
6.2.2	Infrastructure Nodes Resolution	14
6.2.3	Application Services Resolution	14
6.3	Web Services Access via NGINX	14
7	Conclusion	16

List of Figures

1	Global network diagram showing two Data Centers connected via WAN .	4
2	Netplan configuration file for a gateway	7
3	Netplan configuration file for an internal machine	8
4	Screenshot of the <code>docker-compose.yml</code> file captured from the VM, showing MySQL, phpMyAdmin, and Apache services with persistent storage for MySQL.	10
5	NGINX reverse proxy configuration file.	11
6	Output of <code>sudo systemctl status nginx</code> confirming that the service is active and running.	11
7	Local LAN Connectivity Test: Ping from NGINX (VM6) to Worker (VM5). .	12
8	Tailscale VPN Status: Active connections and offline machines.	12
9	End-to-End ICMP Ping Test: From Worker to Gateway A and to Manager.	13
10	Traceroute from Gateway B to Manager (10.10.1.20).	13
11	DNS server reachability test from VM5 in LAN B. Positive response indicates that DNS traffic can traverse the Gateways and VPN.	13
12	Resolution of infrastructure node hostnames using DNS. The server correctly returns the corresponding private IP addresses.	14
13	Resolution of application service hostnames using DNS (CNAME/A records). Both 'app.lan' and 'pma.lan' point to the NGINX server IP (10.0.1.1), which then routes the traffic to the appropriate containers.	14
14	Accessing the Apache Web service via the URL <code>http://app.lan</code>	15
15	phpMyAdmin login interface via the URL <code>http://pma.lan</code>	15
16	Validation of access to application services via the Reverse Proxy.	15

1 Introduction

1.1 Project Context

This project simulates a professional infrastructure in which two independent Data Centers operate collaboratively while maintaining their autonomy. Each data center is represented by a distinct physical workstation equipped with its own internal local area network (LAN).

To realistically reproduce the operational conditions of a Wide Area Network (WAN), inter-site communication is established through gateway machines acting as routers. Although the initial specification suggested the use of a shared WiFi network, this implementation relies on a **VPN tunnel** to interconnect the two physical sites, ensuring secure, reliable, and controlled inter-network routing.

1.2 Educational Objectives

This project aims to develop essential competencies in system and network engineering, including:

- **Virtualization and Network Architecture:** Configuration of virtual machines and implementation of inter-network routing.
- **Infrastructure Services:** Installation and configuration of an internal DNS server for centralized name resolution.
- **Container Orchestration:** Deployment of a multi-node Docker Swarm cluster.
- **Reverse Proxy Configuration:** Deployment of NGINX as a single entry point for all services.
- **Application Deployment:** Management of MySQL with data persistence, php-MyAdmin, and Apache services.

2 Infrastructure Architecture

2.1 Global Network Topology

The architecture simulates two isolated sites interconnected through a WAN. Figure 1 illustrates the overall topology.

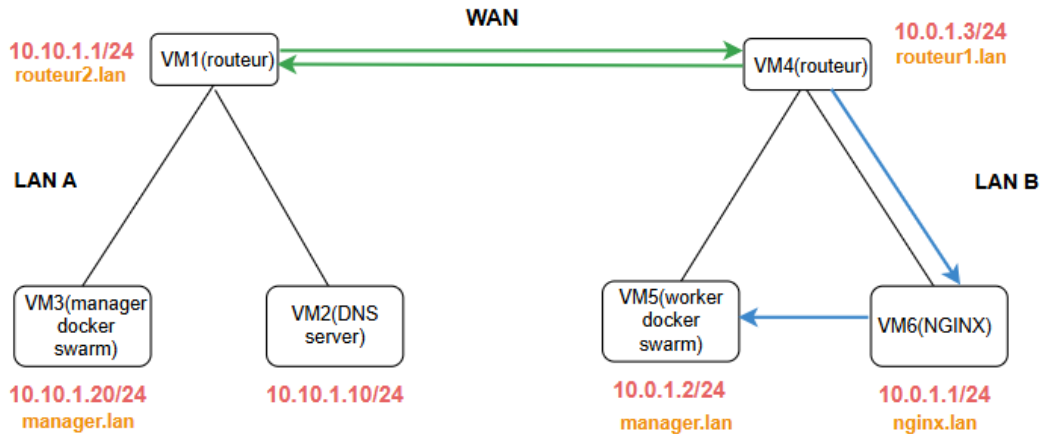


Figure 1: Global network diagram showing two Data Centers connected via WAN

2.2 Data Center Analysis

2.2.1 Data Center A (Management and Core Services)

Data Center A hosts the core infrastructure management services:

- **Gateway A (VM1):** Ensures routing between LAN A and the WAN.
- **DNS Server (VM2):** Provides internal name resolution (pma.lan, app.lan).
- **Docker Manager (VM3):** Acts as the primary node of the Docker Swarm cluster.

2.2.2 Data Center B (Production and Edge Services)

Data Center B focuses on service delivery and user access:

- **Gateway B (VM4):** Connects LAN B to the shared WAN.
- **NGINX Proxy (VM6):** Acts as the reverse proxy and unique entry point.
- **Docker Worker (VM5):** Serves as a secondary node in the distributed cluster.

3 Simulation of the WAN via VPN : Implementation with Tailscale

3.1 Technical Rationale

The decision to use **Tailscale** to simulate the WAN infrastructure was driven by three key technical factors:

- **Simplified Connectivity (NAT Traversal):** It bypasses complex firewall restrictions and unstable local network environments, ensuring a persistent link between Gateway A and Gateway B.
- **High-Level Security:** All inter-site traffic is encrypted using the **WireGuard®** protocol, providing a secure tunnel for sensitive data such as MySQL synchronization.
- **Network Overlay Architecture:** By creating a virtual interface (`tailscale0`), it abstracts the underlying physical network, allowing for clean and professional routing management between the isolated LANs.

3.2 Technical Implementation Steps

This section describes the systematic procedure followed to configure the virtual WAN on both gateway nodes (VM1 and VM4).

A. Kernel-Level Routing Activation (IP Forwarding)

Before installing the VPN solution, packet forwarding must be enabled on both Gateways to allow traffic redirection between the local area network (LAN) and the VPN tunnel.

1. Edit the configuration file: `sudo nano /etc/sysctl.conf`
2. Activate the parameter: `net.ipv4.ip_forward=1`
3. Apply changes: `sudo sysctl -p`

B. Tailscale Installation and Initialization

The Tailscale client was installed exclusively on both gateway machines, which act as the sole egress points to the WAN infrastructure.

1. Install via the official script:
`curl -fsSL https://tailscale.com/install.sh | sh`
2. Authenticate and join the private Tailnet:
`sudo tailscale up`

C. Subnet Routing Configuration

This critical step enables internal machines (DNS, Manager, Worker) without Tailscale installation to communicate across the WAN.

1. On Gateway A, advertise LAN A subnet:
`tailscale up -advertise-routes=10.10.1.0/24`
2. On Gateway B, advertise LAN B subnet:
`tailscale up -advertise-routes=10.0.1.0/24`

D. Administrative Console Validation

In the Tailscale Admin Console, subnet routes were manually approved for each gateway node, officially authorizing the VPN tunnel to transport traffic destined for both Data Center IP ranges.

4 Network Configuration and Inter-Site Routing

4.1 Static IP Addressing Plan

Table 1: Static IP Addressing Plan

Network	Machine	Interface Role	IP Address
LAN A	VM1	Gateway (LAN)	10.10.1.1/24
LAN A	VM2	DNS Server	10.10.1.10/24
LAN A	VM3	Docker Swarm Manager	10.10.1.20/24
LAN B	VM4	Gateway (LAN)	10.0.1.3/24
LAN B	VM5	Docker Swarm Worker	10.0.1.2/24
LAN B	VM6	NGINX Server	10.0.1.1/24
WAN (Tailscale)	VM1 – VM4	VPN Tunnel	Tailscale Assigned IPs

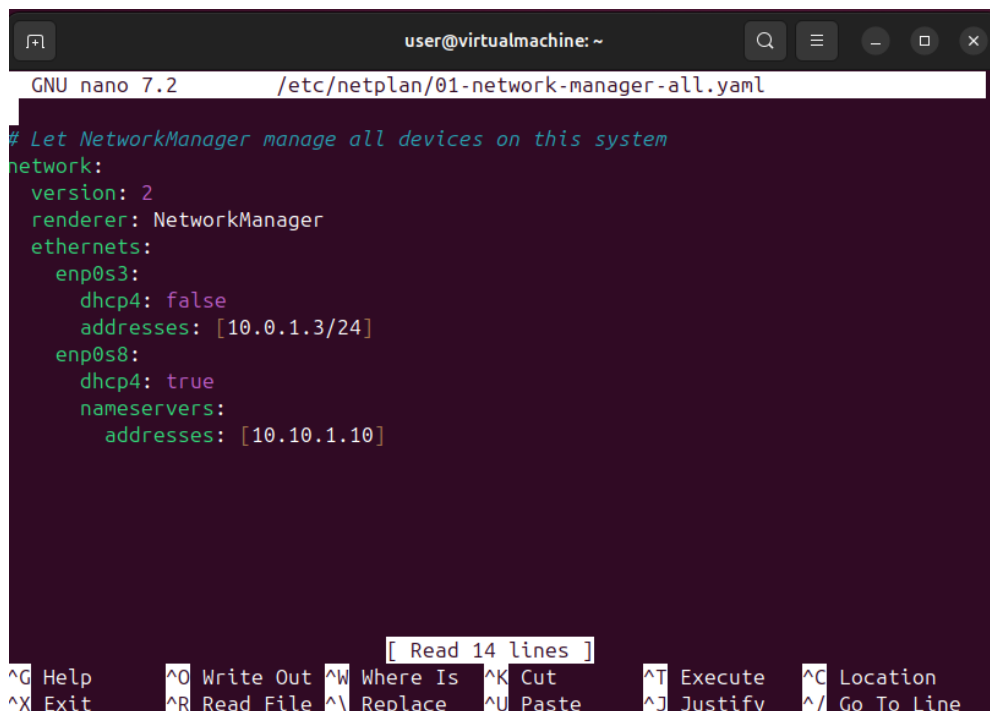
4.2 Network Interface Configuration (Netplan)

Network interfaces are configured using **Netplan** with static IP addressing.

Gateway Configuration (Dual-Homing) Gateways **VM1** and **VM4** use two network interfaces (LAN and WAN via Tailscale) to enable routing.

The configuration file is edited using:

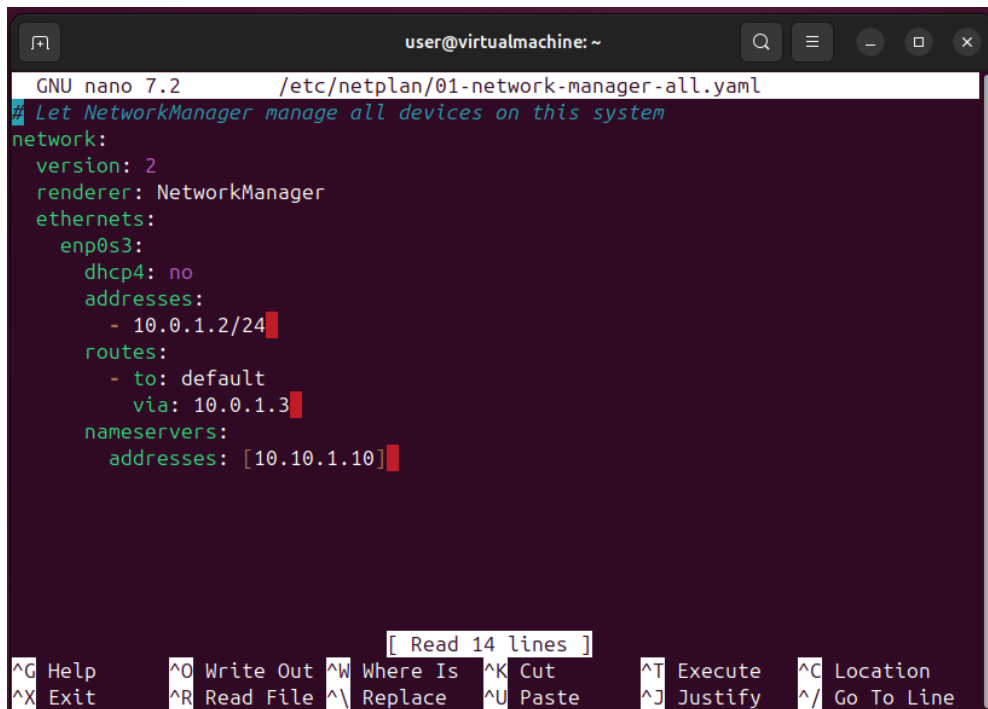
```
sudo nano /etc/netplan/01-netcfg.yaml
```



```
user@virtualmachine: ~  
GNU nano 7.2 /etc/netplan/01-network-manager-all.yaml  
# Let NetworkManager manage all devices on this system  
network:  
  version: 2  
  renderer: NetworkManager  
  ethernets:  
    enp0s3:  
      dhcp4: false  
      addresses: [10.0.1.3/24]  
    enp0s8:  
      dhcp4: true  
      nameservers:  
        addresses: [10.10.1.10]  
[ Read 14 lines ]  
^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location  
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^_ Go To Line
```

Figure 2: Netplan configuration file for a gateway

Internal Machines Configuration Internal machines use a single LAN interface with static IP addressing and a default gateway.



The screenshot shows a terminal window with the title bar "user@virtualmachine: ~". The terminal is running the GNU nano 7.2 editor, editing the file /etc/netplan/01-network-manager-all.yaml. The configuration file content is as follows:

```
Let NetworkManager manage all devices on this system
network:
  version: 2
  renderer: NetworkManager
  ethernets:
    enp0s3:
      dhcp4: no
      addresses:
        - 10.0.1.2/24
      routes:
        - to: default
          via: 10.0.1.3
      nameservers:
        addresses: [10.10.1.10]
```

At the bottom of the terminal, there is a status bar showing "[Read 14 lines]" and a list of keyboard shortcuts: ^G Help, ^O Write Out, ^W Where Is, ^K Cut, ^T Execute, ^C Location, ^X Exit, ^R Read File, ^\ Replace, ^U Paste, ^J Justify, ^_ Go To Line.

Figure 3: Netplan configuration file for an internal machine

4.3 Inter-Network Routing Setup

Default Gateway Definition Internal machines redirect traffic outside their local subnet to the gateway using a default route.

```
sudo ip route add default via 10.0.1.3
```

5 Deployment of Services

5.1 Initialization of the Docker Swarm Cluster

The cluster must run on two distributed nodes. The Manager is located in LAN A and the Worker in LAN B.

On the Manager (VM3 - LAN A): Initialize the cluster by specifying the machine's IP in LAN A.

```
docker swarm init --advertise-addr 10.10.1.20
```

This command generates a connection token.

On the Worker (VM5 - LAN B): Join the cluster using the Manager's IP. Thanks to the VPN routing, the Worker can reach the 10.10.1.20/24 network.

```
docker swarm join --token <RECEIVED_TOKEN> 10.10.1.20:2377
```

5.2 Deployment of the Application Stack

The services MySQL, phpMyAdmin, and Apache are deployed via Docker Swarm. For MySQL, data persistence is mandatory.

```
GNU nano 7.2                                docker-compose.yml
version: "3.9"

services:
  mysql:
    image: mysql:8.0
    container_name: mysql
    environment:
      MYSQL_ROOT_PASSWORD: rootpassword
      MYSQL_DATABASE: testdb
      MYSQL_USER: user
      MYSQL_PASSWORD: userpassword
    volumes:
      - mysql_data:/var/lib/mysql
    networks:
      - app_net
    deploy:
      replicas: 1
      restart_policy:
        condition: on-failure

  apache:
    image: httpd:latest
    container_name: apache
    ports:
      - "8080:80"
    networks:
      - app_net
    deploy:
      replicas: 1
      restart_policy:
        condition: on-failure

  phpmyadmin:
    image: phpmyadmin/phpmyadmin:latest
    container_name: pma
    environment:
      PMA_HOST: mysql
      PMA_USER: root
      PMA_PASSWORD: rootpassword
    ports:
      - "8181:80"
    networks:
      - app_net
    depends_on:
      - mysql
    deploy:
      replicas: 1
      restart_policy:
        condition: on-failure

networks:
  app_net:
    driver: overlay

volumes:
  mysql_data:
    driver: local
```

Figure 4: Screenshot of the docker-compose.yml file captured from the VM, showing MySQL, phpMyAdmin, and Apache services with persistent storage for MySQL.

5.3 NGINX Reverse Proxy Configuration and Verification

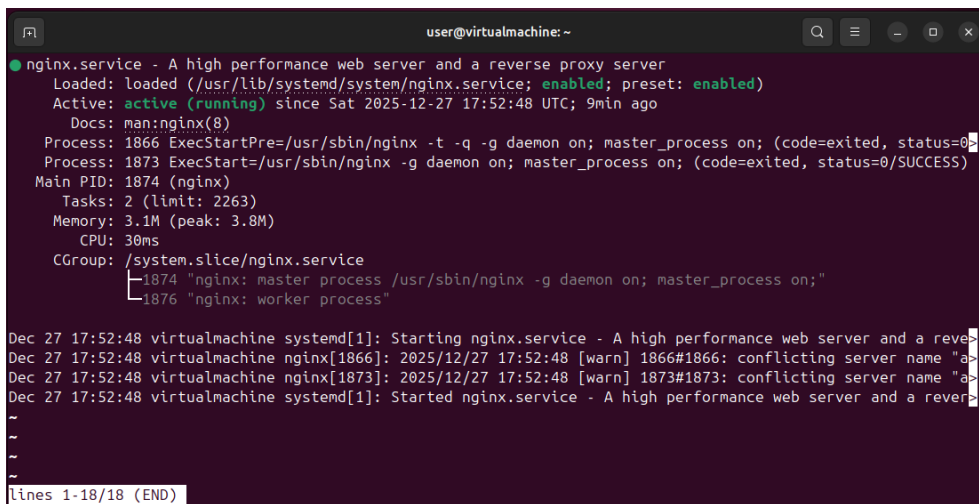
The NGINX server (VM6 in LAN B) serves as the single external entry point for all containerized services. It performs reverse proxying based on hostnames to route traffic to the appropriate Docker containers.



```
GNU nano 7.2 /etc/nginx/sites-available/reverse-proxy
server {
    listen 80;
    server_name pma.lan;
    location / {
        proxy_pass http://10.0.1.2:8080;
    }
}

server {
    listen 80;
    server_name app.lan;
    location / {
        proxy_pass http://10.0.1.2:8081;
    }
}
```

Figure 5: NGINX reverse proxy configuration file.



```
● nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; preset: enabled)
   Active: active (running) since Sat 2025-12-27 17:52:48 UTC; 9min ago
     Docs: man:nginx(8)
  Process: 1866 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on; (code=exited, status=0)
  Process: 1873 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
 Main PID: 1874 (nginx)
    Tasks: 2 (limit: 2263)
   Memory: 3.1M (peak: 3.8M)
      CPU: 30ms
   CGroup: /system.slice/nginx.service
           └─1874 "nginx: master process /usr/sbin/nginx -g daemon on; master_process on;"
             └─1876 "nginx: worker process"

Dec 27 17:52:48 virtualmachine systemd[1]: Starting nginx.service - A high performance web server and a reverse proxy server:
Dec 27 17:52:48 virtualmachine nginx[1866]: 2025/12/27 17:52:48 [warn] 1866#1866: conflicting server name "app.lan"
Dec 27 17:52:48 virtualmachine nginx[1873]: 2025/12/27 17:52:48 [warn] 1873#1873: conflicting server name "app.lan"
Dec 27 17:52:48 virtualmachine systemd[1]: Started nginx.service - A high performance web server and a reverse proxy server:
~
~
~
lines 1-18/18 (END)
```

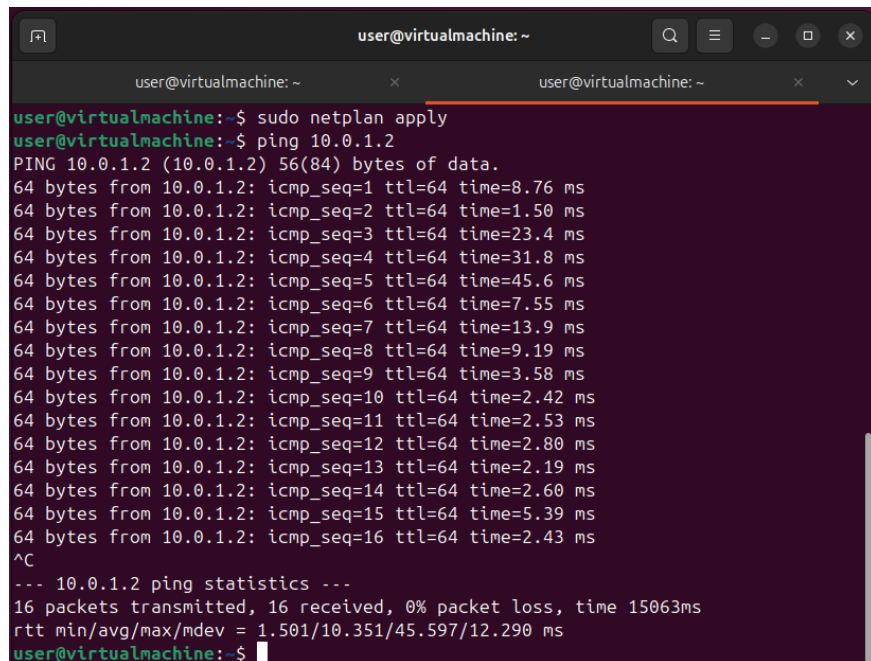
Figure 6: Output of `sudo systemctl status nginx` confirming that the service is active and running.

All external access is thus securely centralized through this reverse proxy.

6 Testing

6.1 Network Connectivity Routing Validation

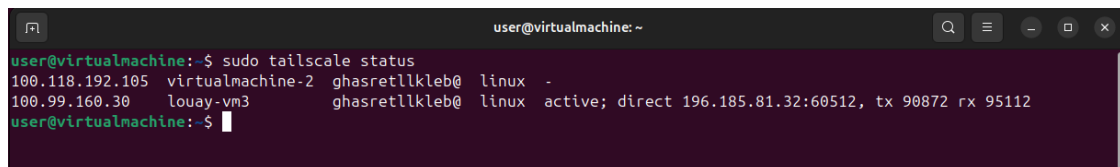
6.1.1 Local LAN Connectivity (Internal)

A terminal window titled 'user@virtualmachine: ~' showing a series of commands and their outputs. The user runs 'sudo netplan apply' and then 'ping 10.0.1.2'. The output shows 16 successful ping requests to 10.0.1.2 with varying response times. Finally, the user presses '^C' to stop the ping, and the terminal displays the ping statistics: 16 packets transmitted, 16 received, 0% packet loss, and a total time of 15063ms.

```
user@virtualmachine:~$ sudo netplan apply
user@virtualmachine:~$ ping 10.0.1.2
PING 10.0.1.2 (10.0.1.2) 56(84) bytes of data.
 64 bytes from 10.0.1.2: icmp_seq=1 ttl=64 time=8.76 ms
 64 bytes from 10.0.1.2: icmp_seq=2 ttl=64 time=1.50 ms
 64 bytes from 10.0.1.2: icmp_seq=3 ttl=64 time=23.4 ms
 64 bytes from 10.0.1.2: icmp_seq=4 ttl=64 time=31.8 ms
 64 bytes from 10.0.1.2: icmp_seq=5 ttl=64 time=45.6 ms
 64 bytes from 10.0.1.2: icmp_seq=6 ttl=64 time=7.55 ms
 64 bytes from 10.0.1.2: icmp_seq=7 ttl=64 time=13.9 ms
 64 bytes from 10.0.1.2: icmp_seq=8 ttl=64 time=9.19 ms
 64 bytes from 10.0.1.2: icmp_seq=9 ttl=64 time=3.58 ms
 64 bytes from 10.0.1.2: icmp_seq=10 ttl=64 time=2.42 ms
 64 bytes from 10.0.1.2: icmp_seq=11 ttl=64 time=2.53 ms
 64 bytes from 10.0.1.2: icmp_seq=12 ttl=64 time=2.80 ms
 64 bytes from 10.0.1.2: icmp_seq=13 ttl=64 time=2.19 ms
 64 bytes from 10.0.1.2: icmp_seq=14 ttl=64 time=2.60 ms
 64 bytes from 10.0.1.2: icmp_seq=15 ttl=64 time=5.39 ms
 64 bytes from 10.0.1.2: icmp_seq=16 ttl=64 time=2.43 ms
^C
--- 10.0.1.2 ping statistics ---
16 packets transmitted, 16 received, 0% packet loss, time 15063ms
rtt min/avg/max/mdev = 1.501/10.351/45.597/12.290 ms
user@virtualmachine:~$
```

Figure 7: Local LAN Connectivity Test: Ping from NGINX (VM6) to Worker (VM5).

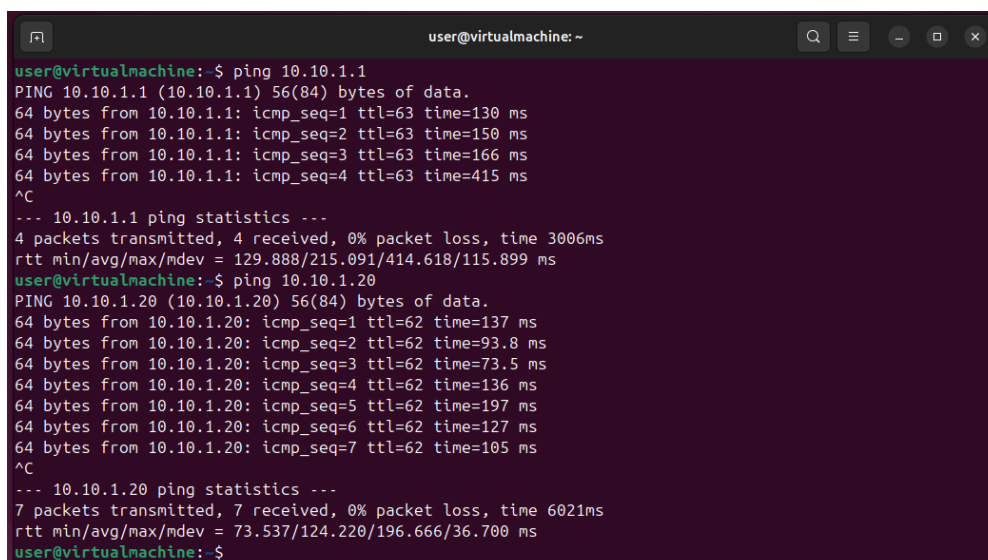
6.1.2 VPN Tunnel Establishment (WAN Simulation)

A terminal window titled 'user@virtualmachine: ~' showing the output of the 'sudo tailscale status' command. The output lists three machines: 'virtualmachine-2' (100.118.192.105) which is offline, 'louay-vm3' (100.99.160.30) which is active, and another instance of 'virtualmachine-2' (100.118.192.105) which is also offline. The active machine shows direct IP 196.185.81.32 and packet statistics.

```
user@virtualmachine:~$ sudo tailscale status
100.118.192.105 virtualmachine-2 ghasretllkleb@ linux -
100.99.160.30 louay-vm3 ghasretllkleb@ linux active; direct 196.185.81.32:60512, tx 90872 rx 95112
user@virtualmachine:~$
```

Figure 8: Tailscale VPN Status: Active connections and offline machines.

6.1.3 Inter-Site ICMP Echo Requests (End-to-End Ping)



```
user@virtualmachine: ~  
user@virtualmachine:~$ ping 10.10.1.1  
PING 10.10.1.1 (10.10.1.1) 56(84) bytes of data.  
64 bytes from 10.10.1.1: icmp_seq=1 ttl=63 time=130 ms  
64 bytes from 10.10.1.1: icmp_seq=2 ttl=63 time=150 ms  
64 bytes from 10.10.1.1: icmp_seq=3 ttl=63 time=166 ms  
64 bytes from 10.10.1.1: icmp_seq=4 ttl=63 time=415 ms  
^C  
--- 10.10.1.1 ping statistics ---  
4 packets transmitted, 4 received, 0% packet loss, time 3006ms  
rtt min/avg/max/mdev = 129.888/215.091/414.618/115.899 ms  
user@virtualmachine:~$ ping 10.10.1.20  
PING 10.10.1.20 (10.10.1.20) 56(84) bytes of data.  
64 bytes from 10.10.1.20: icmp_seq=1 ttl=62 time=137 ms  
64 bytes from 10.10.1.20: icmp_seq=2 ttl=62 time=93.8 ms  
64 bytes from 10.10.1.20: icmp_seq=3 ttl=62 time=73.5 ms  
64 bytes from 10.10.1.20: icmp_seq=4 ttl=62 time=136 ms  
64 bytes from 10.10.1.20: icmp_seq=5 ttl=62 time=197 ms  
64 bytes from 10.10.1.20: icmp_seq=6 ttl=62 time=127 ms  
64 bytes from 10.10.1.20: icmp_seq=7 ttl=62 time=105 ms  
^C  
--- 10.10.1.20 ping statistics ---  
7 packets transmitted, 7 received, 0% packet loss, time 6021ms  
rtt min/avg/max/mdev = 73.537/124.220/196.666/36.700 ms  
user@virtualmachine:~$
```

Figure 9: End-to-End ICMP Ping Test: From Worker to Gateway A and to Manager.

6.1.4 Routing Path Analysis (Traceroute)

```
user@virtualmachine:~$ traceroute 10.10.1.20  
traceroute to 10.10.1.20 (10.10.1.20), 64 hops max  
 1  100.99.160.30 48.867ms 65.089ms 46.554ms  
 2  10.10.1.20 56.204ms 54.970ms 56.619ms  
user@virtualmachine:~$
```

Figure 10: Traceroute from Gateway B to Manager (10.10.1.20).

6.2 DNS Resolution Test

6.2.1 DNS Server Reachability

```
user@virtualmachine:~$ ping 10.10.1.10  
PING 10.10.1.10 (10.10.1.10) 56(84) bytes of data.  
64 bytes from 10.10.1.10: icmp_seq=1 ttl=62 time=172 ms  
64 bytes from 10.10.1.10: icmp_seq=2 ttl=62 time=256 ms  
64 bytes from 10.10.1.10: icmp_seq=3 ttl=62 time=144 ms  
64 bytes from 10.10.1.10: icmp_seq=4 ttl=62 time=104 ms  
64 bytes from 10.10.1.10: icmp_seq=5 ttl=62 time=122 ms  
64 bytes from 10.10.1.10: icmp_seq=6 ttl=62 time=213 ms  
64 bytes from 10.10.1.10: icmp_seq=7 ttl=62 time=93.5 ms  
^C  
--- 10.10.1.10 ping statistics ---  
8 packets transmitted, 7 received, 12.5% packet loss, time 7027ms  
rtt min/avg/max/mdev = 93.532/157.886/256.109/55.204 ms  
user@virtualmachine:~$
```

Figure 11: DNS server reachability test from VM5 in LAN B. Positive response indicates that DNS traffic can traverse the Gateways and VPN.

6.2.2 Infrastructure Nodes Resolution

```
user@virtualmachine:~$ nslookup manager.lan
Server:      10.10.1.10
Address:     10.10.1.10#53

Name:   manager.lan
Address: 10.10.1.20

user@virtualmachine:~$ nslookup worker.lan
Server:      10.10.1.10
Address:     10.10.1.10#53

Name:   worker.lan
Address: 10.0.1.2

user@virtualmachine:~$
```

Figure 12: Resolution of infrastructure node hostnames using DNS. The server correctly returns the corresponding private IP addresses.

6.2.3 Application Services Resolution

```
user@virtualmachine:~$ nslookup app.lan
Server:      10.10.1.10
Address:     10.10.1.10#53

Name:   app.lan
Address: 10.0.1.1

user@virtualmachine:~$
```

Figure 13: Resolution of application service hostnames using DNS (CNAME/A records). Both 'app.lan' and 'pma.lan' point to the NGINX server IP (10.0.1.1), which then routes the traffic to the appropriate containers.

6.3 Web Services Access via NGINX

This stage constitutes the final validation of the infrastructure. Successfully accessing these interfaces via a web browser confirms the integrity of the entire technical chain: DNS resolution, inter-site routing via the VPN tunnel, NGINX reverse proxying, and Docker Swarm orchestration.

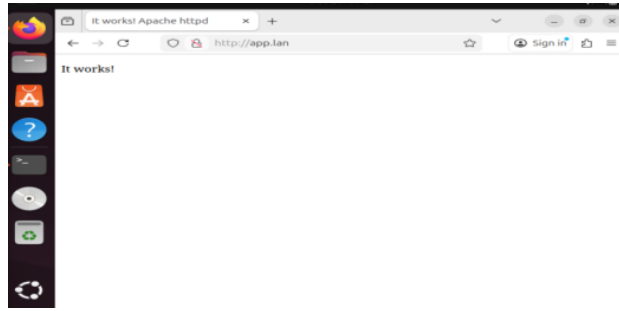


Figure 14: Accessing the Apache Web service via the URL `http://app.lan`

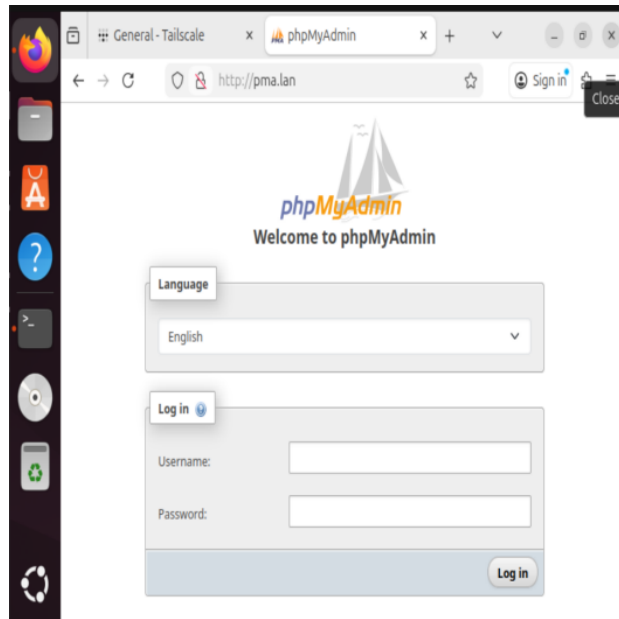


Figure 15: phpMyAdmin login interface via the URL `http://pma.lan`

Figure 16: Validation of access to application services via the Reverse Proxy.

7 Conclusion

This project provided a comprehensive hands-on experience in designing and deploying a **Multi-Site Distributed Virtualized Infrastructure**. By simulating two isolated Data Centers, we successfully navigated the complexities of network segmentation, secure inter-site routing, and modern service orchestration.

Throughout the implementation, several critical milestones were reached:

- **Network Security & Connectivity:** The transition from a simple shared WiFi to a **Tailscale VPN tunnel** (WireGuard) ensured a professional-grade WAN simulation, providing encrypted and stable communication between Gateways.
- **Infrastructure Automation:** The deployment of a **Docker Swarm** cluster across two physical workstations demonstrated the power of container orchestration in a distributed environment.
- **Service Accessibility:** The combination of a centralized **DNS server** and an **NGINX Reverse Proxy** allowed for a seamless user experience, accessing services through professional URLs (`app.lan`, `pma.lan`) rather than technical IP addresses.
- **Data Integrity:** The successful implementation of Docker volumes guaranteed **MySQL data persistence**, a crucial requirement for any production-ready database environment.

In conclusion, the infrastructure is fully operational, meeting all the requirements of the initial specifications. This experience serves as a solid foundation for our future engineering roles in cloud computing and system administration.