

NETWORK

LONDON - SWANSEA

Prepared by: David Lopez Rodriguez

id: 2336542



172.16.0.0/30

10.10.0.0/24

10.20.0.0/24



Research, build, and deploy a Virtualised Enterprise network

Task 1.....	1
Introduction.....	1
Hypervisors	1
IP Addressing.....	1
Routing	2
Network Services	2
Conclusion.....	3
Task 2.....	3
2.1 Reasoning Behind the Selected Hypervisor	3
Screenshot 1: Host System Information.....	4
Screenshot 2: VirtualBox Application Interface.....	5
2.2 Virtual Machine Creation and Configuration	5
Screenshot 3: Servers setup	6
Screenshot 4 : Windows Client Settings.....	7
Screenshot 5: Windows Client Network Adapter.....	8
Screenshot 6: Debian client settings	8
2.3 Internal Virtual Networking Setup.....	9
Inter-Site Link (172.16.0.0/30).....	10
Screenshot 7: Windows server Adapter Settings.....	11
Screenshot 8: Debian Server Adapter Settings.....	11
Screenshot 9: ipconfig ip Setup Windows Server	12
Screenshot 10: ip setup interfaces Debian Server.....	13
2.4 IP Addressing Scheme	14
Screenshot 11: ipConfig Windows Server A	15
Screenshot 12: ipConfig Windows Server BLink: ScreenShotLink.....	15
Screenshot 13: ip addr Debian Server	17
2.5 Static Routing Configuration	17
Debian Server-Swansea Static Route.....	18

Screenshot 14: ip route Debian Server	18
Screenshot 14: ip route Windows Server	19
Screenshot 14: ip route Windows Server	20
Successful cross-site tests included	21
Screenshot 15: Debian Server → Windows Server (10.10.0.10)	22
Screenshot 16: Debian Server → Windows Client and Server Link: ScreenShotLink	22
Screenshot 17: Windows Server → Debian Server (10.20.0.10)	23
Screenshot 18: Windows Server → Debian Client and Server	24
Screenshot 19: Debian Client → both Windows Machines	25
Screenshot 20: Windows client ↔ Debian Server and client	26
2.7 Troubleshooting and Observations	28
2.8 Reflection	28
References	29

Task 1

Introduction

This assessment analyses the concepts, components, and configurations for a secure, functional, and scalable enterprise network environment, focusing on four main aspects, hypervisors as test environment and the following protocols: IP addressing, routing, and network services. They are key towards optimising network functionality and interoperability. The NOSS2026 project utilises Windows and Linux servers using virtualisation technologies, addressing schemes, and centrally authenticated hybrid infrastructure to create a working example of an enterprise-scale connectivity between the Swansea and London sites.

Hypervisors

A hypervisor, also known as a virtual machine, is a software layer that allows multiple operating systems to run on a single physical machine by abstracting the hardware resources, such as CPU, memory, and storage, and allocating them to each guest system [1]. There are two types of hypervisors: Type 1 (bare-metal) and Type 2 (hosted). Type 1 hypervisors run directly on the hardware and offer better performance, stability, and isolation, while Type 2 hypervisors run on top of an existing operating system and are more commonly used for labs and small environments [2].

I have chosen Type 2 hypervisor (Oracle VirtualBox) for my project because it is more flexible and easy to configure for non-enterprise hardware. The host system will be my laptop with Ubuntu 24.04 OS,

which has enough power and memory (Intel i7 CPU and 32GB RAM) to support multiple VMs. According to Vasanthraj and Ravi Kumar [1], Type 1 hypervisors have better scalability and latency than Type 2, but Type 2 also has reliable performance by using soft resources allocation. Cisco [2] also states that the same hardening practices, such as separating management interfaces and implementing least-privilege access, should be applied to virtualized systems. As PowerCert Animated Videos [3] show, Type 2 virtualization is an efficient and realistic way to create an enterprise network environment on limited resources.

IP Addressing

An IP address uniquely identifies a machine on a network for the purposes of sending and receiving data [4]. Engineers did not anticipate the expansion of the Internet in the early years, and when IPv4 was developed, the 32-bit space (≈ 4.3 billion addresses) quickly became inadequate. To overcome this, IPv6 was designed using a 128-bit address space with nearly unlimited addresses and security/auto-configuration features built-in [4]. IPv4 is most widely used and deployed in enterprise networks in the field, while IPv6 provides scalability between geographic locations like Swansea and London.

Subnetting (dividing larger address blocks into more manageable, smaller subnets to increase routing efficiency and reduce the size of broadcast domains) and VLSM (allocating variable subnet sizes based on need to conserve addresses) are both very useful design practices [4]. As a general rule, according to Red Hat [8], static addresses are reserved by administrators for critical services such as DNS, DHCP, and authentication servers, and dynamic addresses are leased to clients via the DHCP DORA (Discovery, Offer, Request, Acknowledgment) cycle. PowerCert Animated Videos [5] provide an overview of private IP ranges (e.g. 192.168.x.x) reserved for internal use and routers using NAT (Network Address Translation) to allow sharing of a single public IP for external connectivity, which provides efficiency as well as security. For this project, we design a custom VLSM plan to assign each server LAN, client LAN, and router-to-router link its own unique subnet.

Routing

Routing is the process of moving data between different networks. A router (physical or virtual) looks at destination IP addresses and forwards packets based on the best path as per the current routing table [4]. There are primarily two methods: static (manual) and dynamic (auto-configured). Static routing involves manually defining routes, providing predictable and secure behaviour with zero CPU overhead but less flexibility in adapting to topology changes [5]. It's best suited to smaller/fixed environments for reliability and deterministic control without overcomplexity.

Dynamic protocols such as RIP, OSPF, and EIGRP automatically exchange route information, calculate the best paths, and dynamically adapt to network changes. Link-state protocols such as OSPF provide faster convergence and better scalability, while distance-vector protocols like RIP are simpler but slower [5]. CiscoNet Training Solutions [6] further notes that static routes have the lowest administrative distance (1), making routers always prefer them over dynamically learned routes, offering deterministic control.

Static Routing is used in the NOSS2026 topology to connect Swansea and London, in order to provide a reliable and secure way to route between subnets. The Debian server is set up to be a DHCP relay, so it can assign addresses across routed networks, while static routes ensure DHCP, DNS and client systems can communicate. For future scalability scenarios (Coursework 2), OSPF could be used for automatic route discovery and performance optimisation across multiple sites.

Network Services

Network services and daemons such as DHCP, Active Directory Domain Services (AD DS), Certificate Authority (CA), and VPNs can be deployed to support secure enterprise network communication, identity, and access management. DHCP facilitates automatic assignment of IP addresses, including optional gateways and DNS, preventing address conflicts and reducing manual network administration effort [9]. IP addresses can be leased and automatically reclaimed by the server for reuse if the lease expires unused, balancing administrative overhead with optimizing usage for defined address scopes. Special reservations for network nodes such as routers or domain controllers that require static and predictable addresses (per this project's design for both the Swansea and London subnets) can also be managed by DHCP [9].

Per Red Hat [8], Linux servers such as Debian can join a Windows AD DS with no trust using the System Security Services Daemon (SSSD) component, with centralized login and authorization to access resources using Kerberos authentication, and Lightweight Directory Access Protocol (LDAP) lookups. Thus, network-wide identity management and integration of user and service accounts across Linux and Windows can be readily achieved, and changes can be synchronized dynamically for DNS and DHCP servers. The Canadian Centre for Cyber Security [7] states that AD DS is the "keys-to-the-kingdom" component of the enterprise network and recommends stringent least-privilege access and multi-factor authentication, with CA-hardened Kerberos sign-in and audit logging of user logins and access control lists to resources.

In this project, AD DS is used to establish a domain (SWANSEA.ACCA) for centralized, consistent, and authenticated login to network resources, DHCP and DNS services are established and integrated on the Debian server, and a VPN component is used to create an encrypted tunnel between the Swansea and London sites to provide resilient, authenticated, and encrypted network communication.

Conclusion

In conclusion, the NOSS2026 network design effectively incorporates virtualization, planned IP addressing, static routing, and critical enterprise services to establish a secure, interconnected network environment. By leveraging Oracle VirtualBox, custom subnetting, and AD DS integration, the system enables seamless communication and identity management across different sites. This implementation not only meets the current coursework objectives but also sets a scalable foundation

that can be adapted for future enhancements, including dynamic routing and expanded service deployment.

Task 2

2.1 Reasoning Behind the Selected Hypervisor

Oracle VirtualBox was chosen as the hypervisor for this assessment for its high compatibility with Ubuntu 24.04, the host machine, vast community documentation, and zero-cost, open source licensing. Running on a native Linux host, VirtualBox is a stable platform, with excellent capabilities for a learning/lab environment, where both configurability and flexibility were key. Internal networks, snapshots, Windows Server 2022 and Debian 13 both were needed to build a multi OS network topology, like the one in NOSS2026, and VirtualBox had out-of-the-box support for all of these requirements.

The host hardware a laptop with Intel Raptor Lake-P i9 (2P+8E), 32 GB RAM, NVMe Samsung 980 had no issues running all 4 VMs simultaneously. In testing, each VM ran without problem with only 2 GB RAM assigned, so a Type 2 hypervisor was more than adequate for the scale of this coursework. As this work did not need any high-availability, clustering, or enterprise-grade performance benefits, a Type 1 hypervisor would have added no discernible benefit.

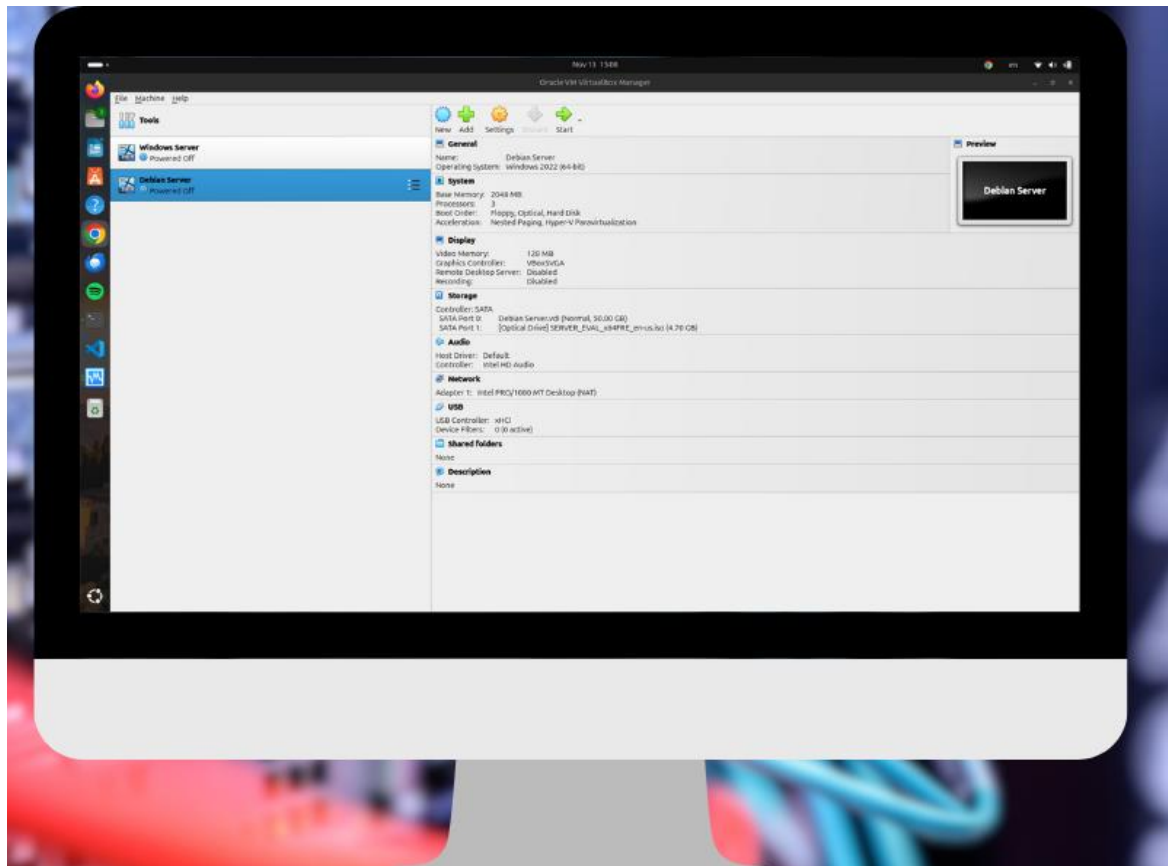
The one challenge that VirtualBox had with the deployment of this environment was some intermittent instability with some virtual network adapters. Attaching ISOs or changing the mode of an adapter would cause some DHCP/connectivity problems, but a reset or reconfiguration of the adapter fixed those issues. VirtualBox was flexible, approachable, and entirely capable in this assessment.

Screenshot 1: Host System Information

```
W -short
WG: you should run this program as super-user.
ath      Device      Class      Description
=====
          system     Computer
          bus        Motherboard
          memory     32GB System memory
          processor  13th Gen Intel(R) Core(TM) i7-1355
0         /dev/fb0     display    Raptor Lake-P i7135e Graphics
0/4       generic     Raptor Lake Dynamic Platform and T
0/6       bridge      Raptor Lake PCIe 4.0 Graphics Port
0/6/0     /dev/nvme0        storage    Samsung SSD 980 500GB
0/6/0/0   hwnon3            disk       NVMe disk
0/6/0/2   /dev/ng0n1        disk       NVMe disk
0/6/0/1   /dev/nvme0n1      disk       NVMe disk
0/8       generic     QNA Scoring Accelerator module
0/8       generic     Raptor Lake Crashlog and Telemetry
0/14      bus         Alder Lake PCH USB 3.2 xHCI Host C
0/14.2    memory     RAM memory
0/14.3    wlp0s20f3        network    Raptor Lake PCH CNVi Wi-Fi
0/15      bus         Alder Lake PCH Serial IO I2C Contr
0/15.3    bus         Alder Lake PCH Serial IO I2C Contr
0/16      communication Alder Lake PCH HECI Controller
0/17      scsi1        Alder Lake-P SATA AHCI Controller
0/17/0-0.0 /dev/cdrom      disk       SORW-08U1MT
0/1c      bridge      Intel Corporation
0/1c/0    bus         SD/MMC Card Reader Controller
0/1f      bridge      Raptor Lake LPC/eSPI Controller
0/1f/0    system     PnP device PNP0c02
0/1f/1    system     PnP device PNP0c02
0/1f/2    input      PnP device PNP0303
0/1f/3    input      PnP device PNP0f13
0/1f/4    system     PnP device PNP0c02
0/1f/5    system     PnP device PNP0c02
0/1f/6    system     PnP device PNP0c02
0/1f.3    card0       multimedia Raptor Lake-P/U/H cAVS
0/1f.3/0 input27      input      HDA Intel PCH HDMI/DP,pcm=3
0/1f.3/1 input28      input      HDA Intel PCH HDMI/DP,pcm=7
0/1f.3/2 input29      input      HDA Intel PCH HDMI/DP,pcm=8
0/1f.3/3 input30      input      HDA Intel PCH HDMI/DP,pcm=9
0/1f.4    bus         Alder Lake PCH-P SMBus Host Contro
0/1f.5    bus         Alder Lake-P PCH SPI Controller
0/1f.6    enp0s31f6    network    Ethernet Connection (16) I219-V
```

Link: [ScreenShotLink](#)

Screenshot 2: VirtualBox Application Interface



Link: [ScreenShotLink](#)

2.2 Virtual Machine Creation and Configuration

The four VMs provisioned for this project were: Windows Server, Windows Client, Debian Server, and Debian Client. All four machines were built from official ISO images with a common configuration pattern to ensure stable and predictable behaviour during testing. The order of machine creation was Windows Server, Windows Client, Debian Server, and Debian Client. This order was intentional: building and configuring domain services in that order made it easier to add clients and integrate DHCP service with the domain.

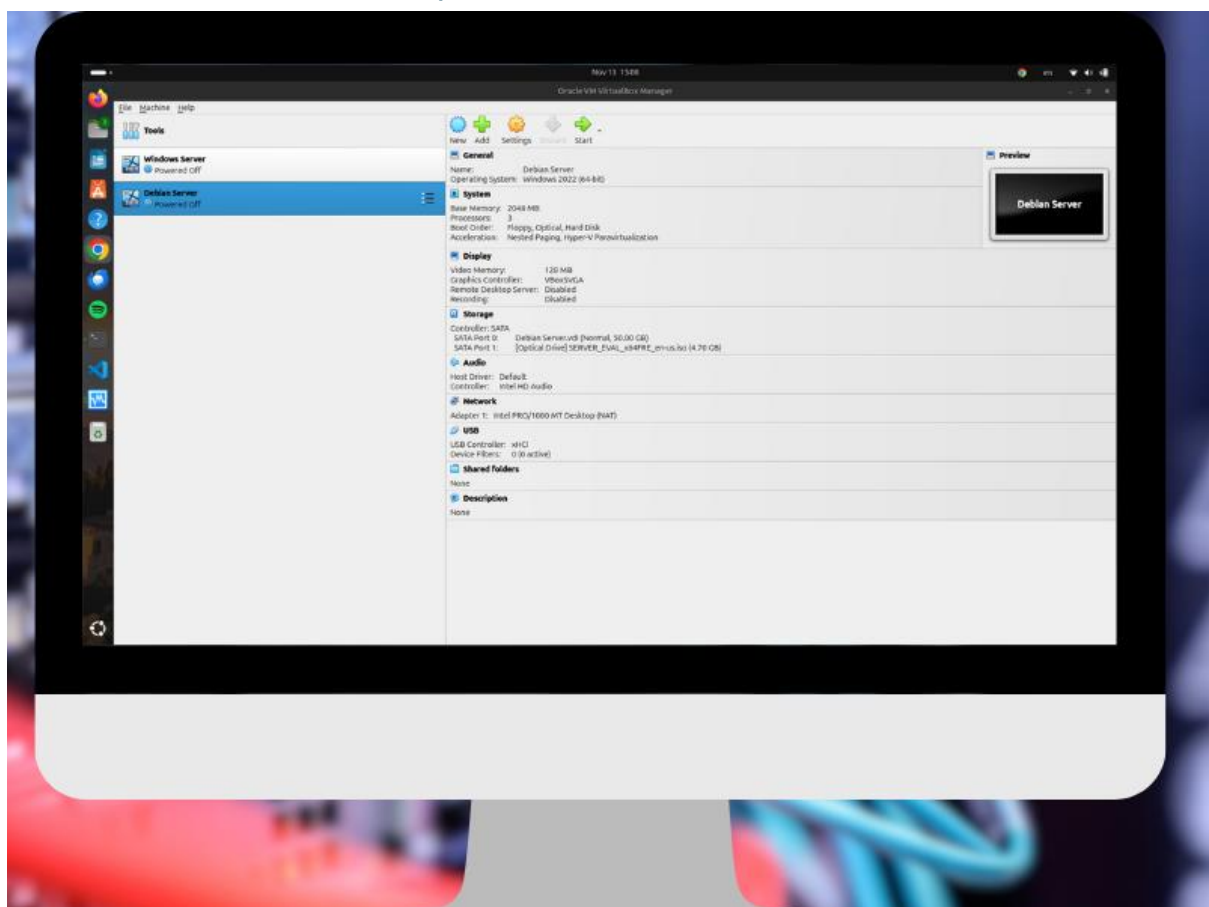
All virtual machines are intentionally lightweight, given my host hardware could comfortably run four instances simultaneously. Peak assessment workload running the 4 of them had each provisioned with 2 GB RAM, 1 vCPU, and a modest virtual disk. This was more than enough to support assessment scope, given the core of this work was routing, addressing, and network services configuration, rather than a workload-intensive configuration.

The remaining two machines Windows Server and Debian Server were provisioned using the same base specification (2 GB RAM, 1 vCPU, BIOS firmware, single storage controller), differing only by

network topology function. Windows Server was bridged to both LondonLAN and the inter-site link, while Debian Server was bridged to SwanseaLAN and the inter-site link, to permit routing between the two sites.

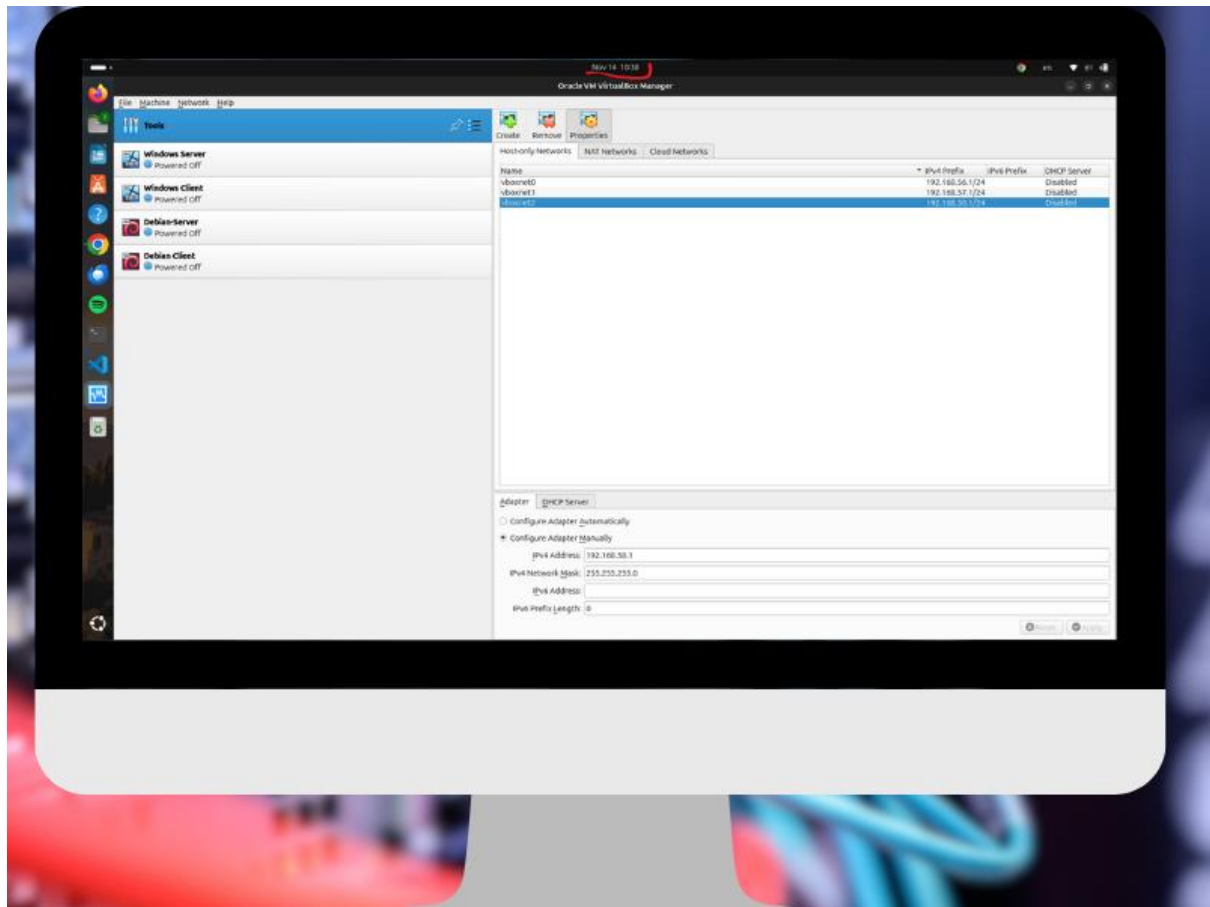
This lightweight configuration was intentional: the nature of this assessment did not warrant a more computationally intensive service, and by limiting the resource footprint per instance, I was able to run them in parallel without contention. The primary limitation was imposed by VirtualBox network adapters and ISO mounting, in that it occasionally became necessary to reassign adapters or reattach virtual storage devices. This instability was unusual and did not detract from overall setup processes, which were stable and predictable, thanks to Oracle's extensive documentation and Linux compatibility.

Screenshot 3: Servers setup



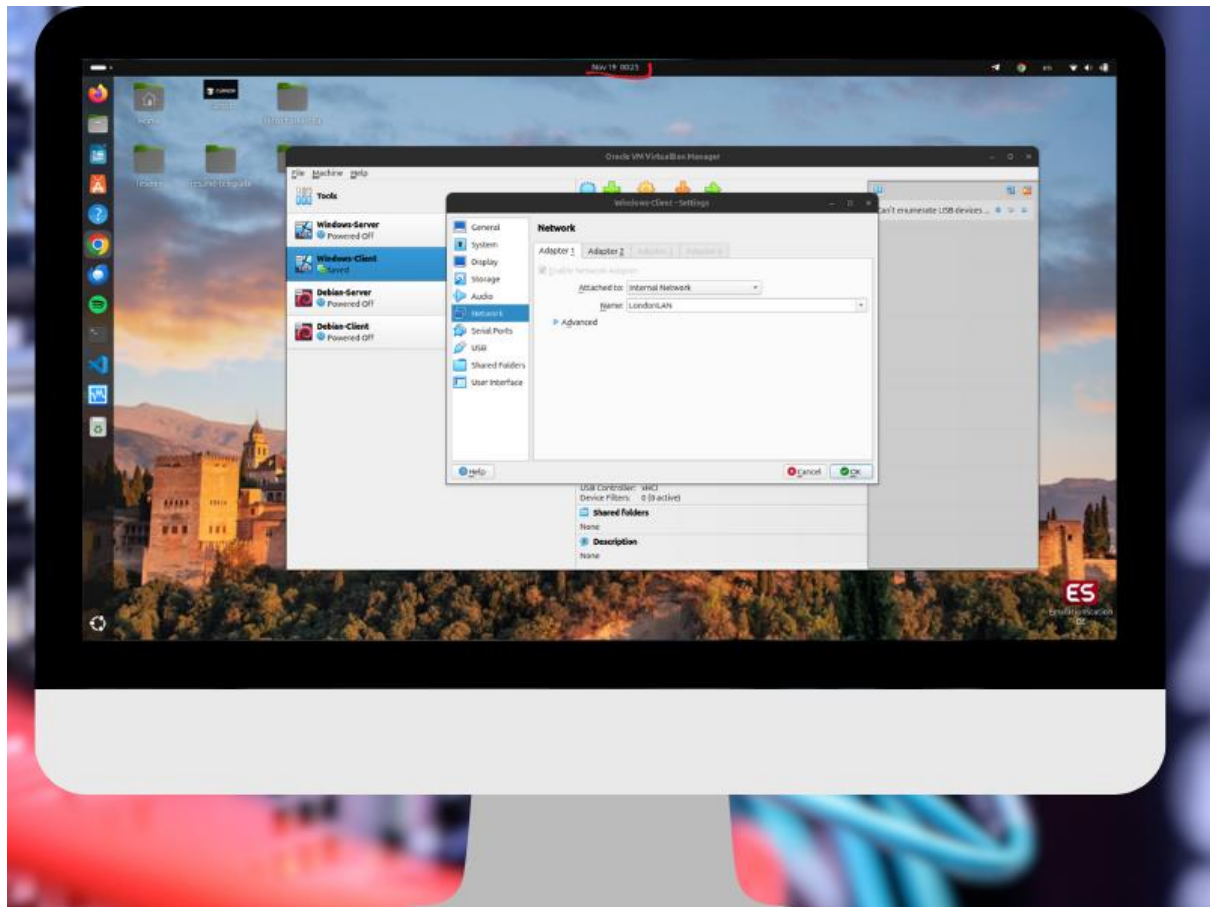
Link: [ScreenShotLink](#)

Screenshot 4 : Windows Client Settings



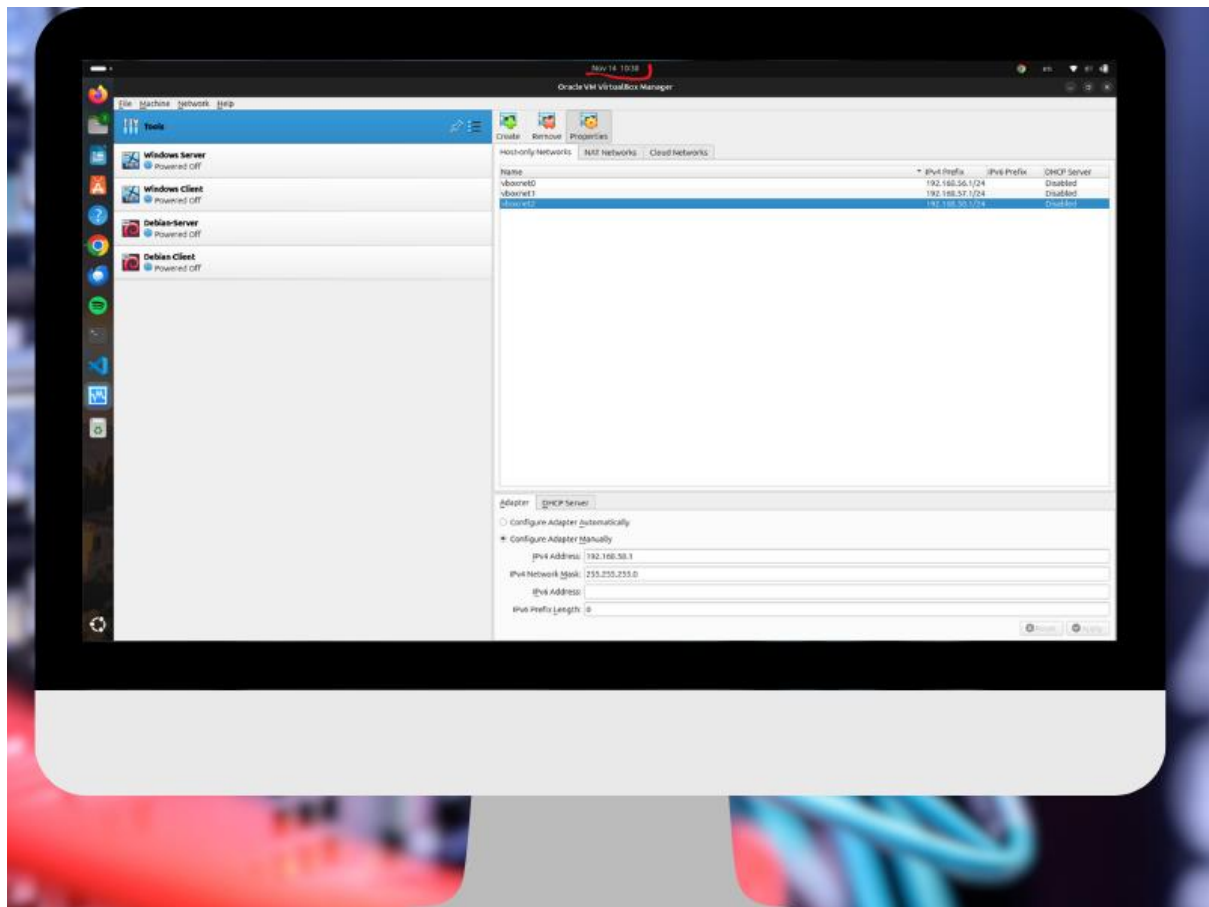
Link: [ScreenShotLink](#)

Screenshot 5: Windows Client Network Adapter



Link: [ScreenShotLink](#)

Screenshot 6: Debian client settings



Link: [ScreenShotLink](#)

2.3 Internal Virtual Networking Setup

There were three completely isolated VirtualBox internal networks created: LondonLAN, SwanseaLAN, and an internal route-only link between the two servers called Interlink. VirtualBox's Internal Network mode was used so that each internal network acted like a completely isolated host-level network, but allowed communication between guests on the same virtual switch. These two internal networks, LondonLAN and SwanseaLAN, were configured with their own subnets and DHCP scopes, so London and Swansea would be completely separate LANs from each other. This closely models a two-site enterprise network, in which London and Swansea are two separate branches of a corporation.

LondonLAN (10.10.0.0/24)

The Windows Server and Windows Client were both members of the LondonLAN internal network. This virtual LAN represents the production network for the London site. The Windows Server was configured with a static IP address (10.10.0.10), and the client received a dynamic lease from the Windows DHCP server. This approximates a typical Windows domain network, in which the domain controller is providing DHCP and DNS services for the local subnet.

SwanseaLAN (10.20.0.0/24)

The Debian Server and Debian Client were both members of the SwanseaLAN network. The Debian server had a static IP of 10.20.0.10, and the client received a dynamic IP address from the Debian DHCP scope. By keeping these two virtual LANs completely separate, the lab environment accurately modeled a multi-site enterprise network with multiple broadcast domains.

Inter-Site Link (172.16.0.0/30)

To allow the two sites to talk to ping other, each host was given a second network adapter attached to a separate internal network to represent the point-to-point link:

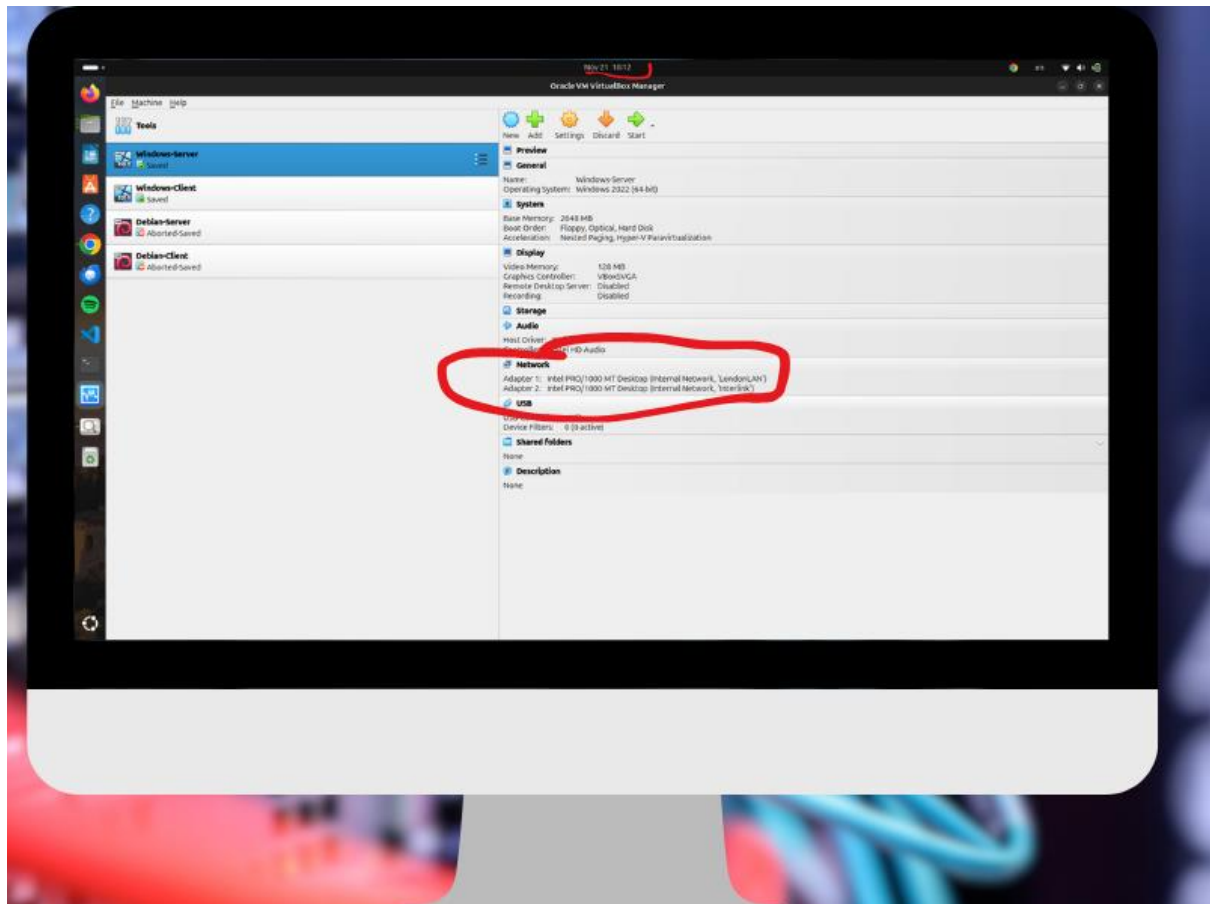
-Windows Server: 172.16.0.1/30

-Debian Server: 172.16.0.2/30

The /30 subnet was small enough to only contain the two hosts, providing exactly 2 usable IP addresses. This small subnet size is desirable for a point-to-point router-to-router or server-to-server link. Static routing was later configured on both machines to allow LondonLAN and SwanseaLAN to use the link. The complete setup now resembled two WAN-connected branch networks talking to each other over a controlled routing path.

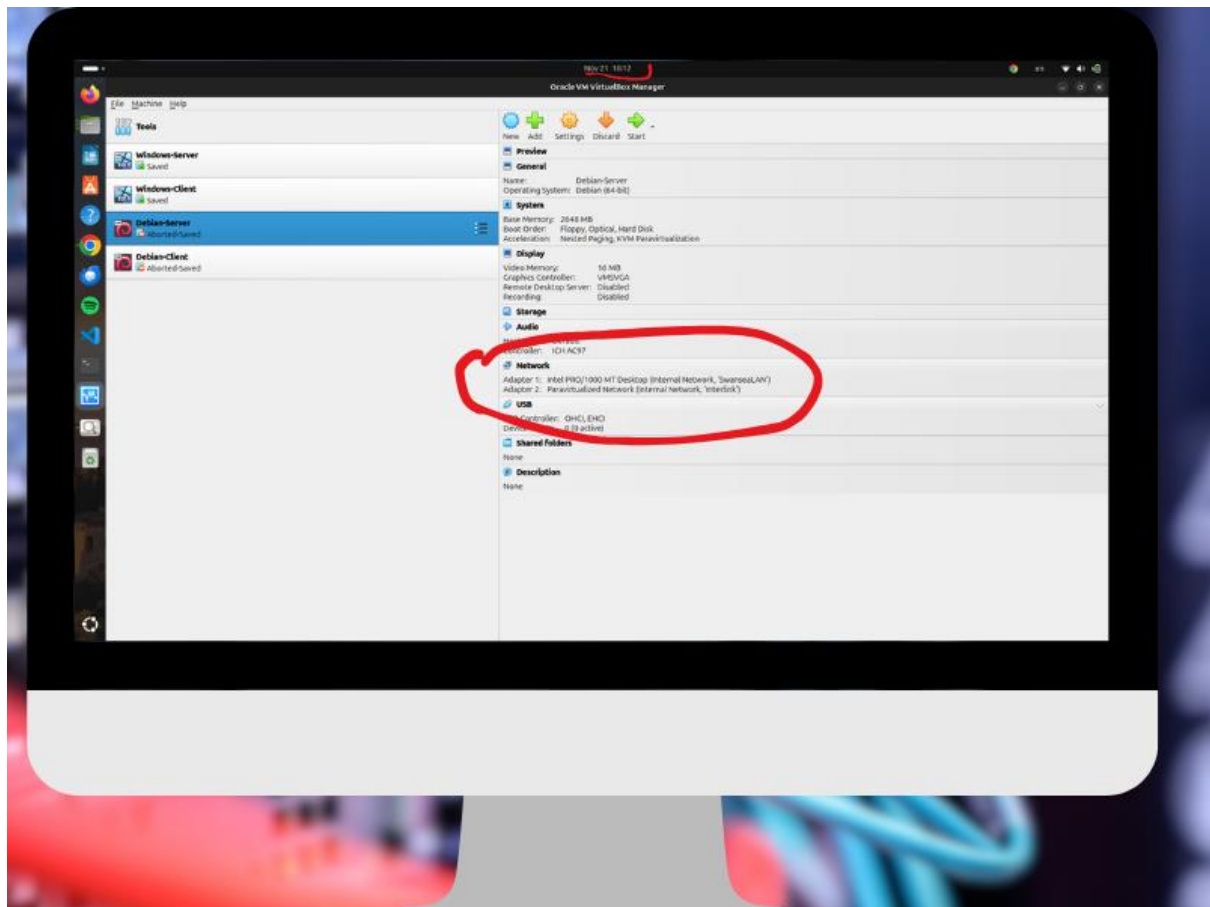
Since all the networks were isolated at the VirtualBox level, the host's networking was not affected by the internal VMs and vice versa. This provides deterministic configuration and ensures that there are no two DHCP servers talking to each other between LANs. The result is a clean and reproducible multi-site virtual topology suitable for testing Active Directory, DHCP relay behaviour, and routing between subnets.

Screenshot 7: Windows server Adapter Settings



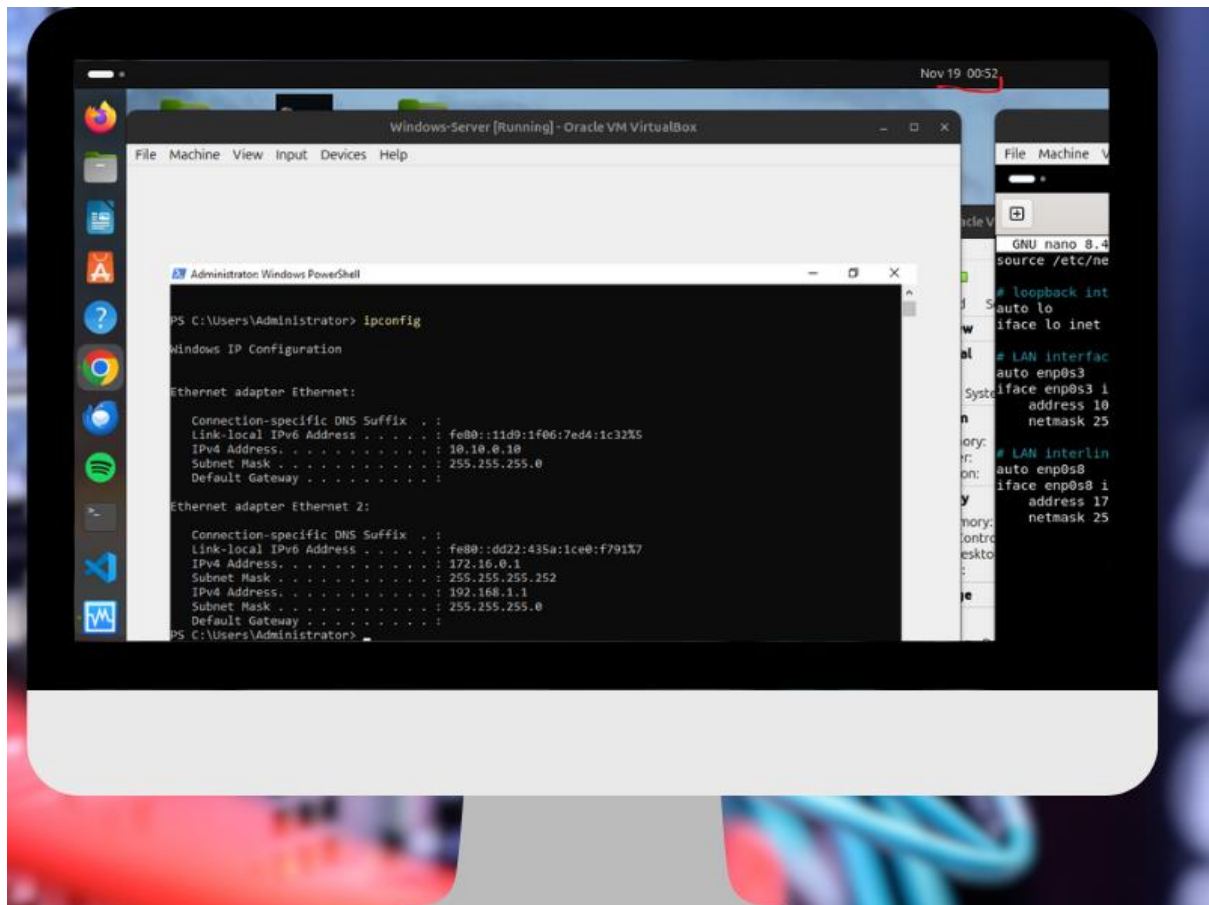
Link: [ScreenShotLink](#)

Screenshot 8: Debian Server Adapter Settings



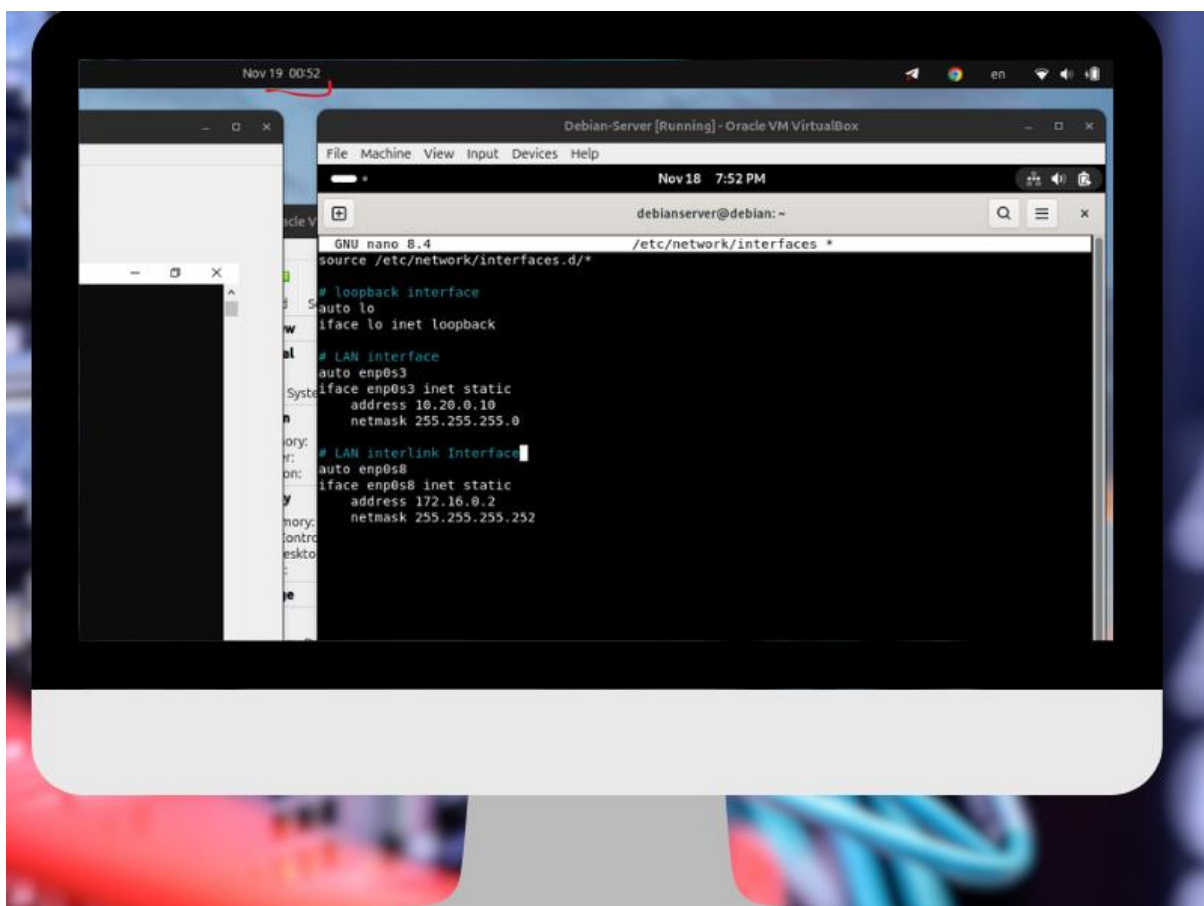
Link: [ScreenShotLink](#)

Screenshot 9: ipconfig ip Setup Windows Server



Link: [ScreenShotLink](#)

Screenshot 10: ip setup interfaces Debian Server



Link: [ScreenShotLink](#)

2.4 IP Addressing Scheme

A structured IP addressing scheme was used to clearly separate the London and Swansea sites. VLSM principles were applied to allocate subnets efficiently and assign fixed static IPs to server interfaces.

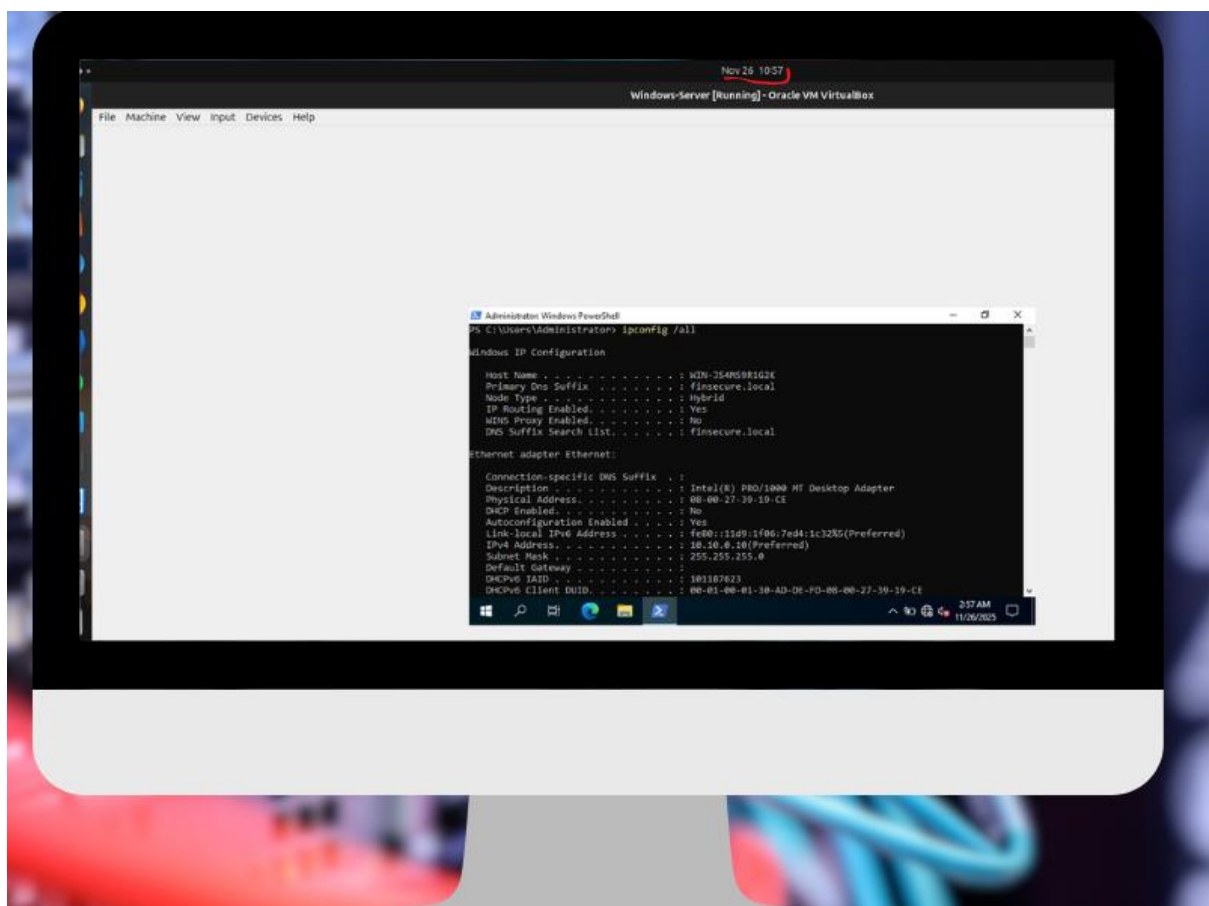
Network	Subnet	Purpose	Key Addresses
LondonLAN	10.10.0.0/24	London site	Windows Server: 10.10.0.10
SwanseaLAN	10.20.0.0/24	Swansea site	Debian Server: 10.20.0.10

Interlink	172.16.0.0/30	Point-to-point link	WS: 172.16.0.1 / Debian: 172.16.0.2
-----------	---------------	---------------------	-------------------------------------

The use of separate /24 networks for each site allows clear broadcast domain separation, simplified troubleshooting, and easier demonstration of routing and inter-site communication.

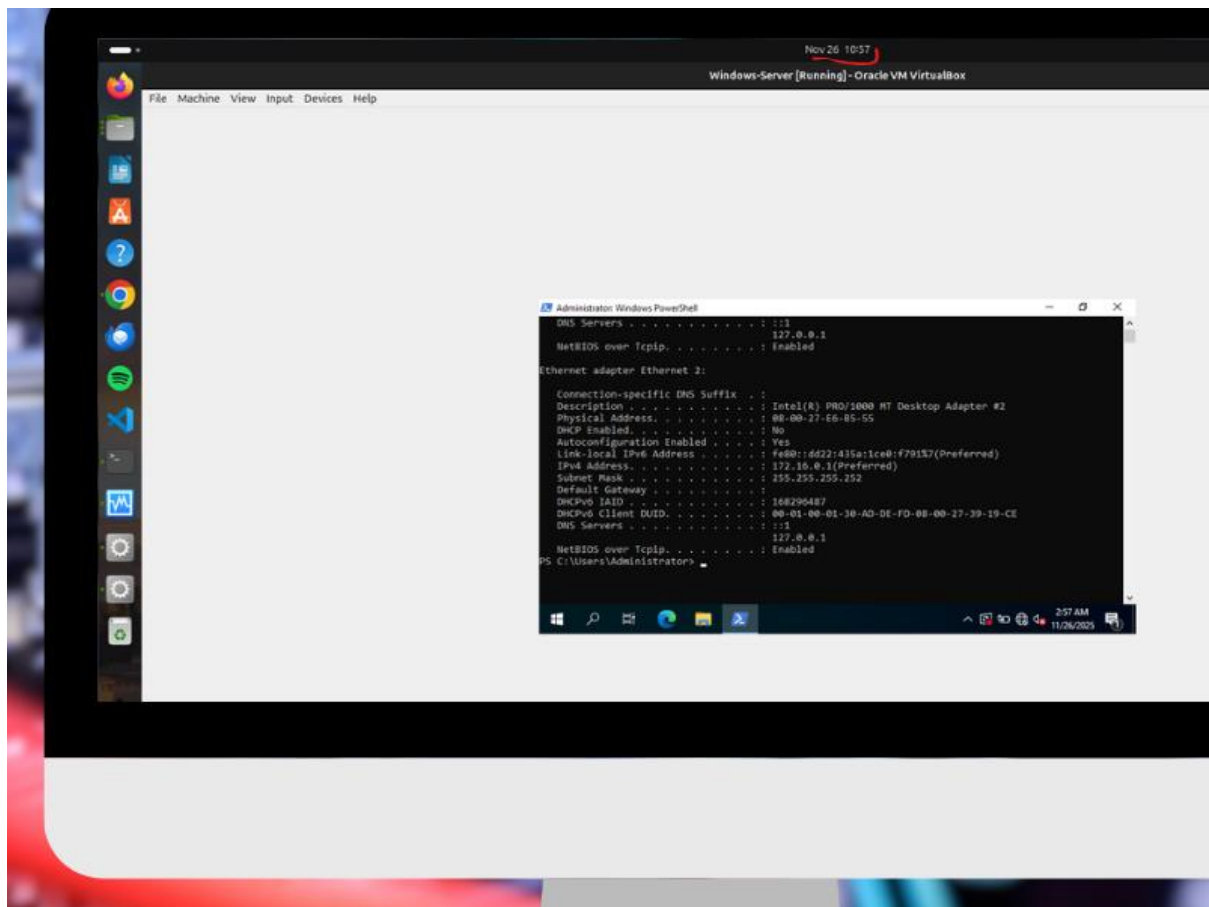
The Windows Server's IP configuration screenshot confirmed both 10.10.0.10 and 172.16.0.1 were correctly applied. The Debian Server's screenshot confirmed the dual-interface configuration with 10.20.0.10 and 172.16.0.2.

Screenshot 11: ipConfig Windows Server A



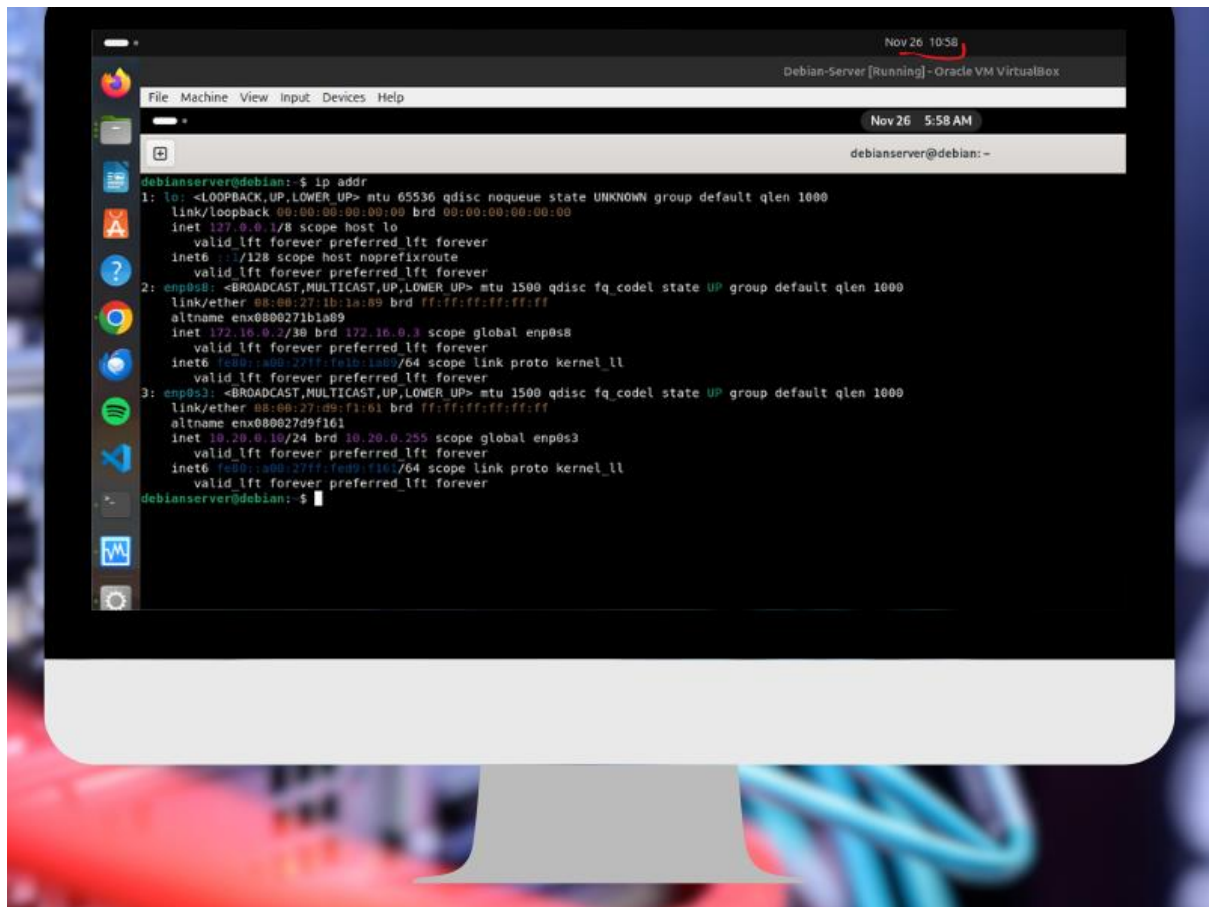
Link: [ScreenShotLink](#)

Screenshot 12: ipConfig Windows Server B



Link: [ScreenShotLink](#)

Screenshot 13: ip addr Debian Server



Link: [ScreenShotLink](#)

2.5 Static Routing Configuration

Static routing was implemented to facilitate communication between LondonLAN and SwanseaLAN. Traffic from one site had to be sent to the other, via the inter-site link, rather than being isolated on each LAN.

Windows Server-London Static Route

The following persistent static route was added to enable the London Server to reach the 10.20.0.0/24 network via the inter-site gateway:

route add 10.20.0.0 mask 255.255.255.0 172.16.0.2 -p - Destination: SwanseaLAN, Gateway: Debian Server's interlink address.

Debian Server-Swansea Static Route

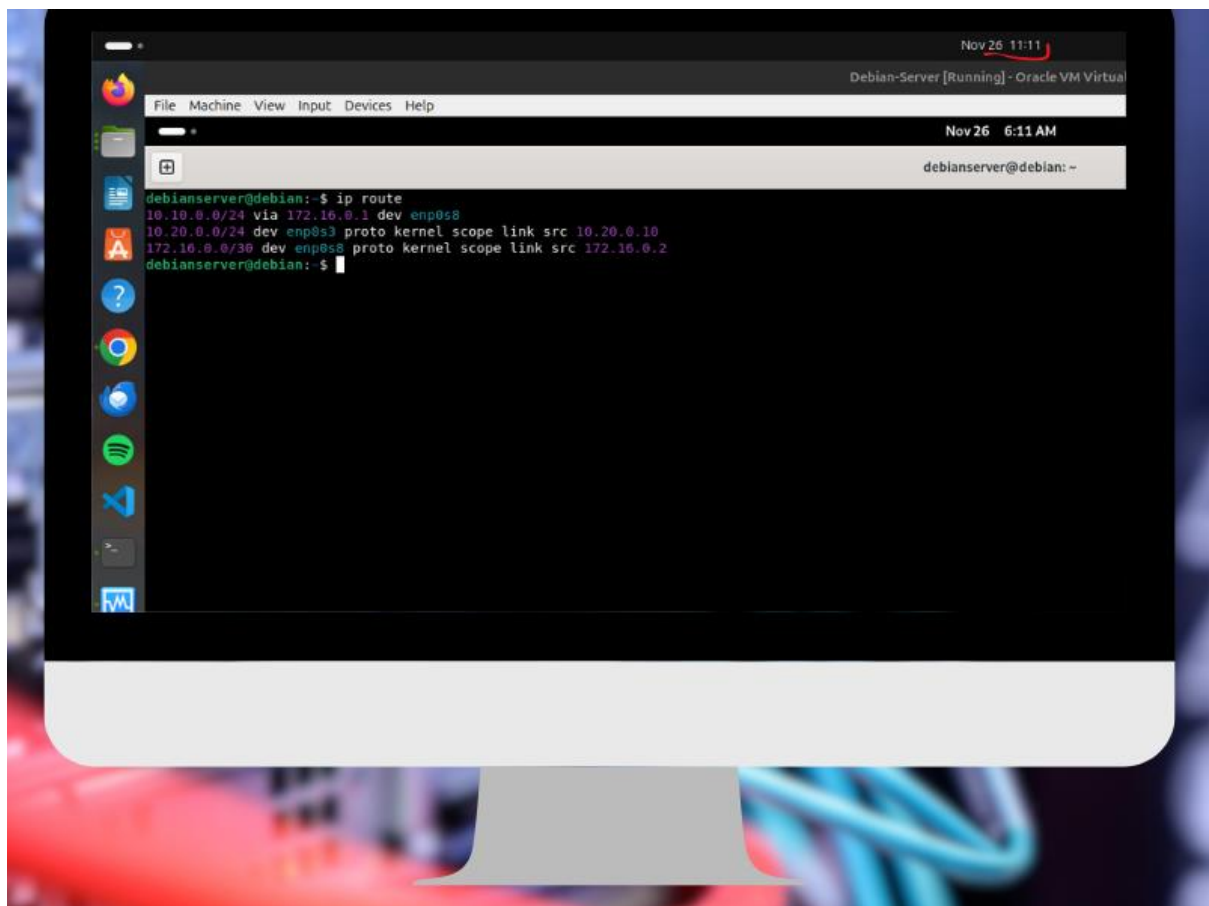
On Debian, routing to the London site was configured using:

ip route add 10.10.0.0/24 via 172.16.0.1 Destination: LondonLAN, Gateway: Windows Server's interlink address.

This established the correct forwarding path toward the London Server.

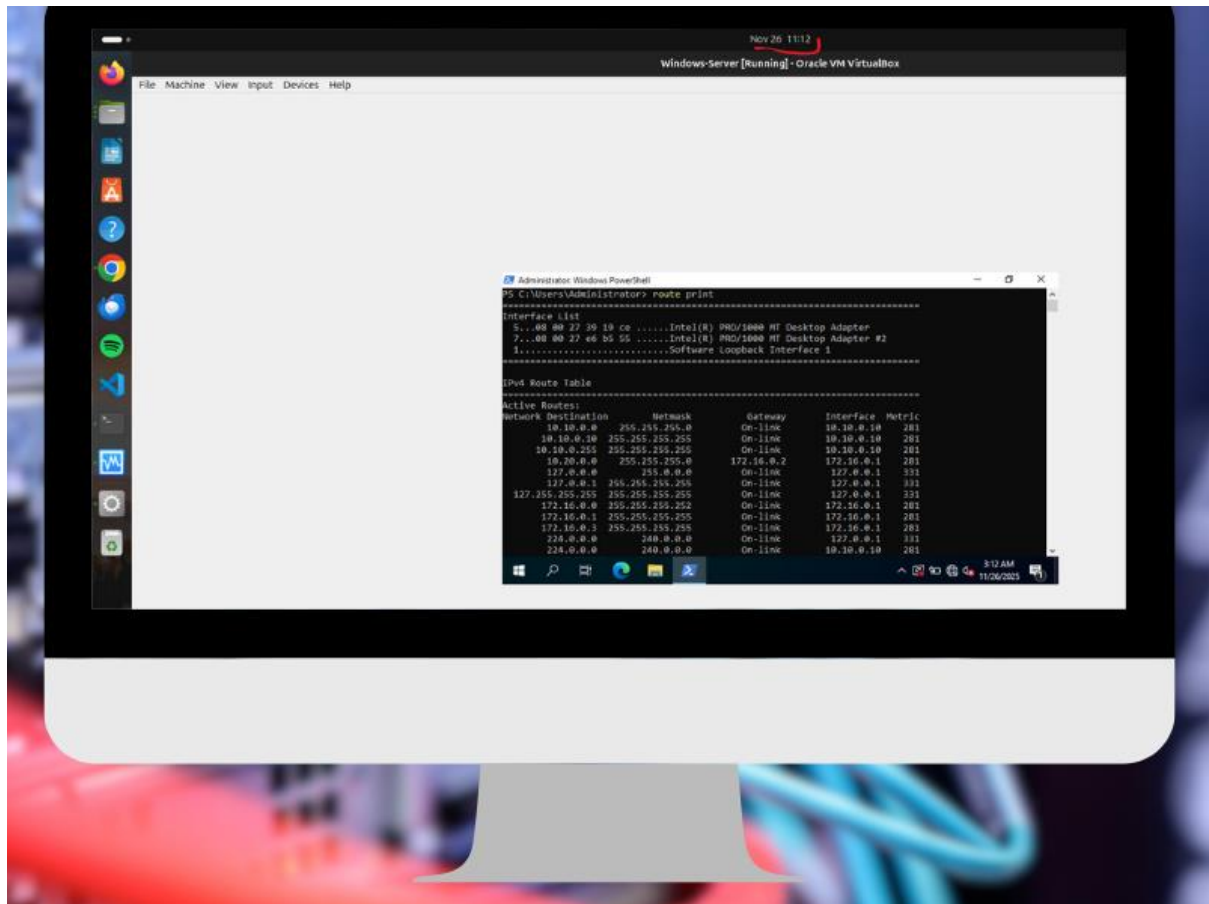
Static routing was appropriate because the network topology is simple, fixed, and does not require dynamic routing protocols such as OSPF or RIP.

Screenshot 14: ip route Debian Server



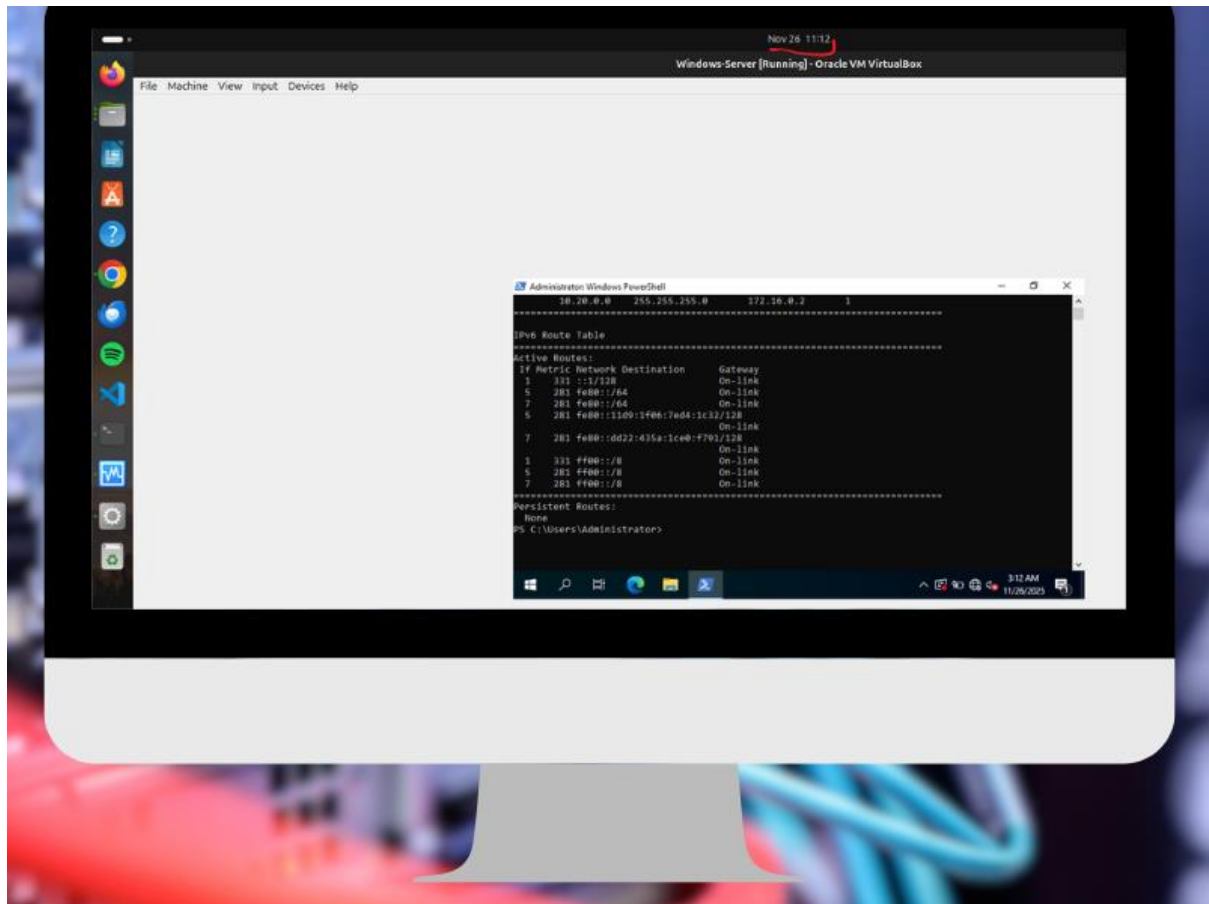
Link: [ScreenShotLink](#)

Screenshot 14: ip route Windows Server



Link: [ScreenShotLink](#)

Screenshot 14: ip route Windows Server



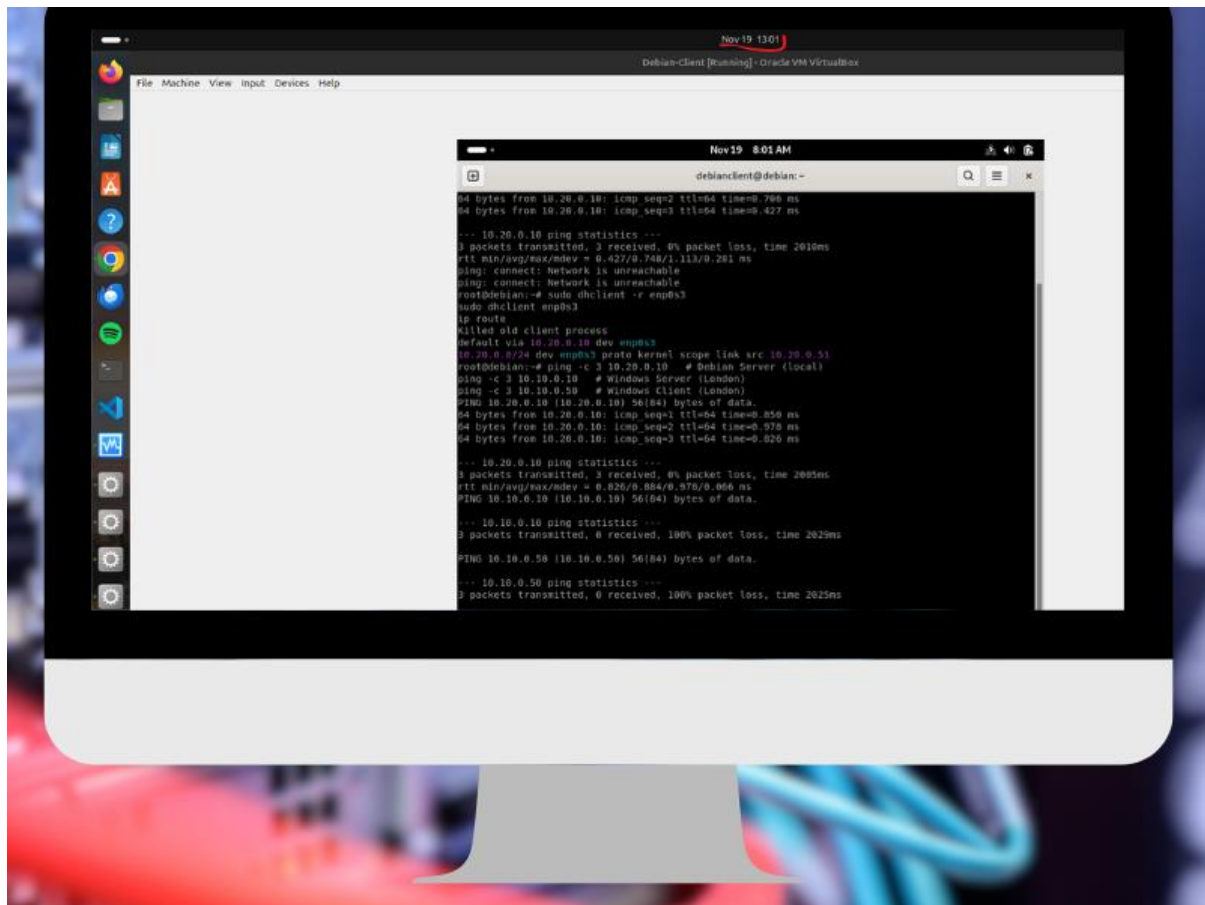
Link: [ScreenShotLink](#)

2.6 Connectivity Tests and Verification

Verification of the full network configuration was performed using ping tests across all machines. Some initial ICMP tests were failing due to local firewall restrictions on both Windows Server and Debian hosts. Firewall rules were temporarily changed to let ICMP echo requests and routing traffic. This resolved the connectivity issues and allowed for successful cross site testing. These tests validate that IP addressing, interface configuration, and static routing were implemented correctly.

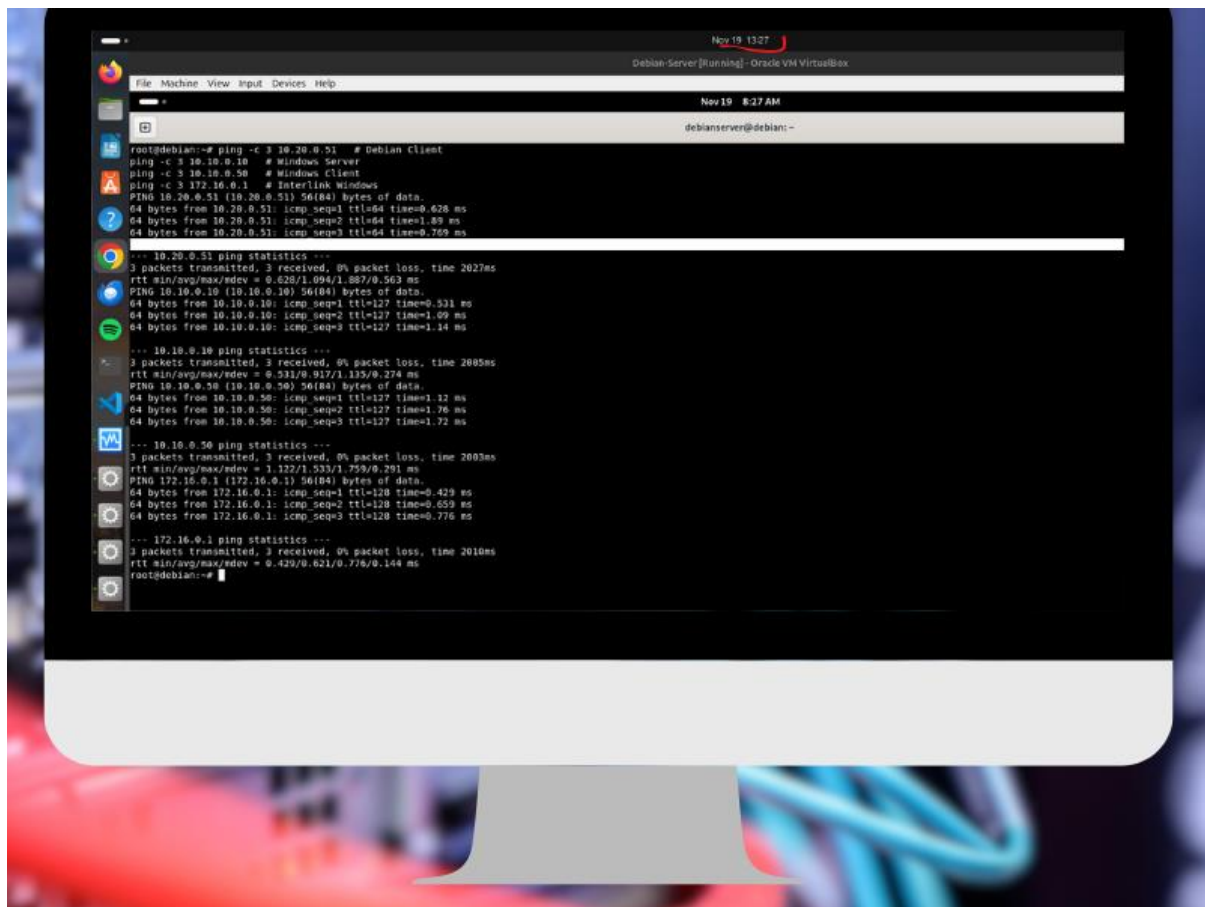
Successful cross-site tests included

Screenshot 15: Debian Server → Windows Server (10.10.0.10)



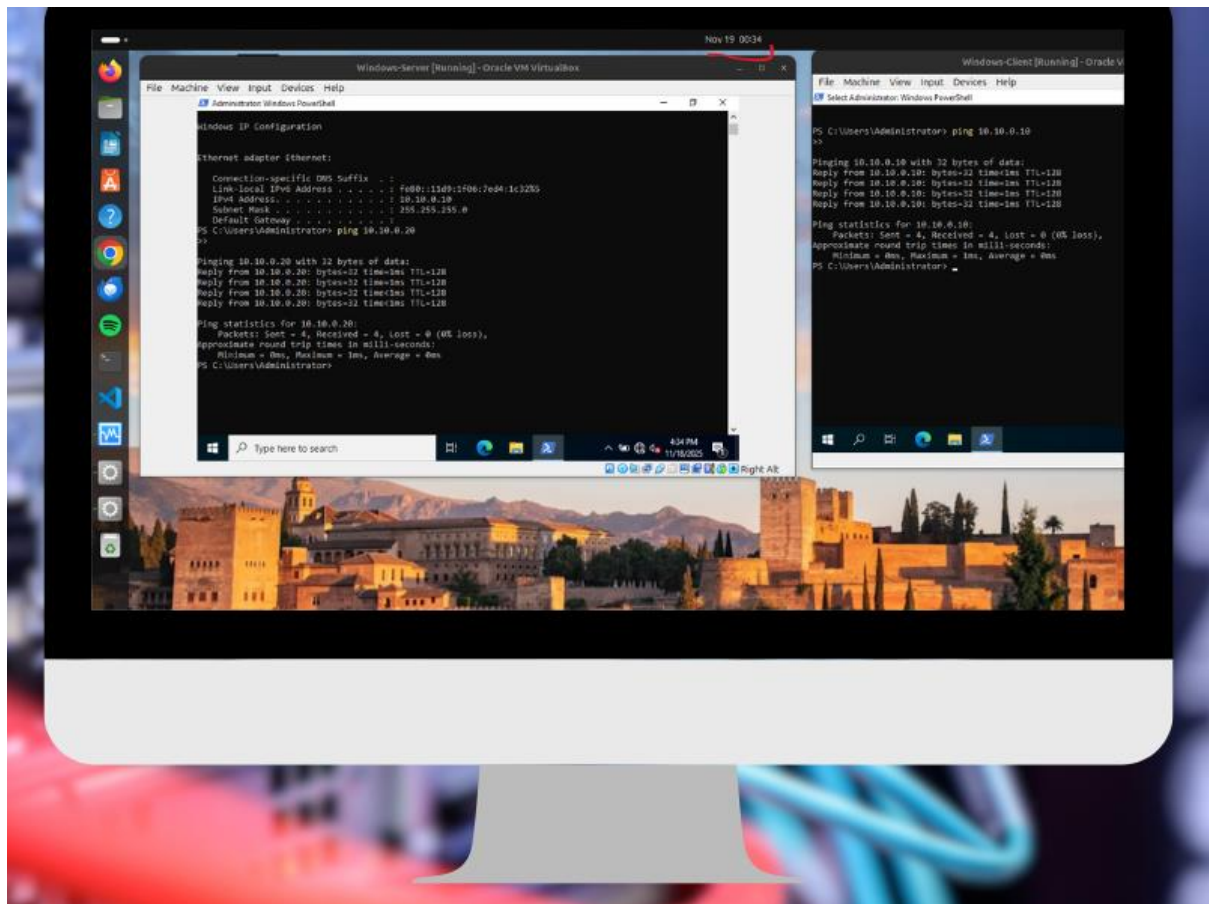
Link: [ScreenShotLink](#)

Screenshot 16: Debian Server → Windows Client and Server



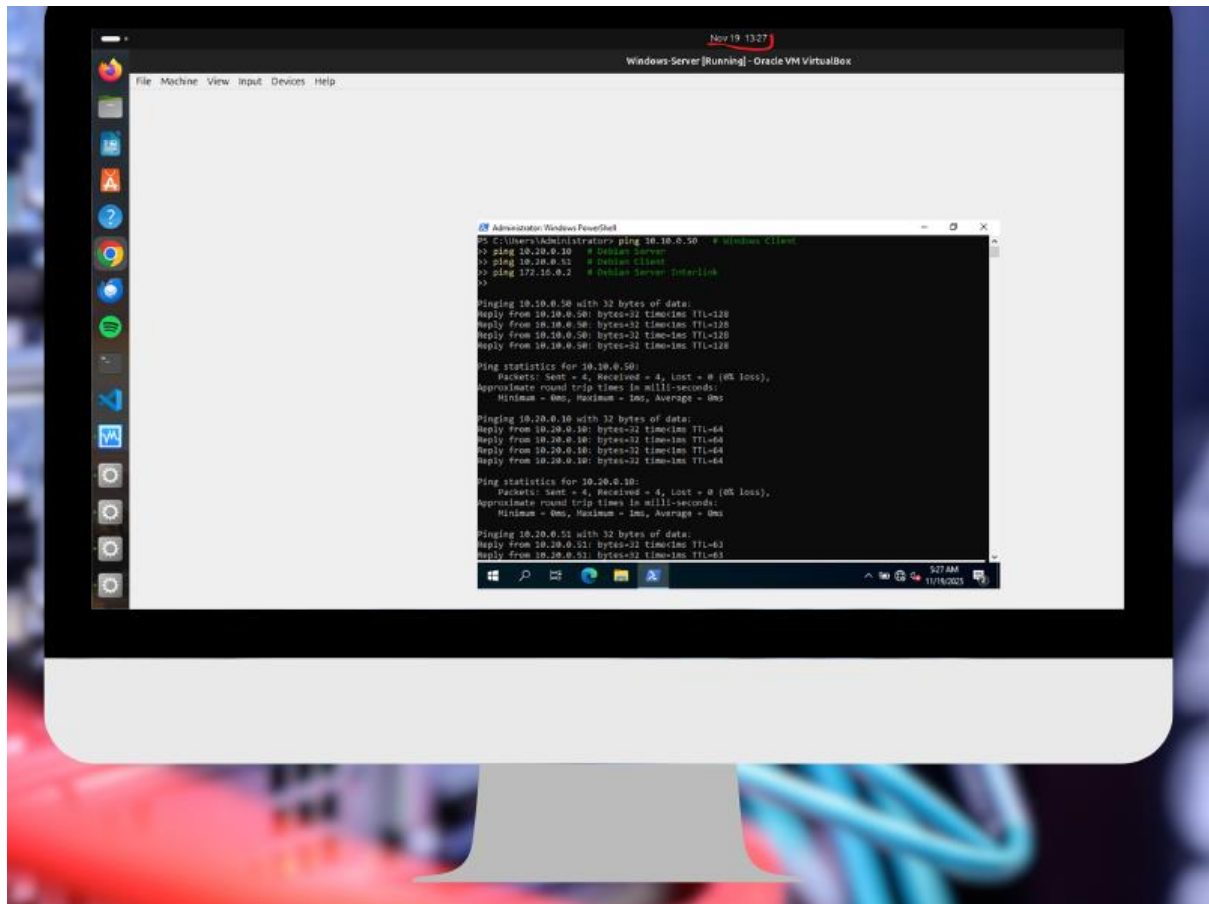
Link: [ScreenShotLink](#)

Screenshot 17: Windows Server → Debian Server (10.20.0.10)



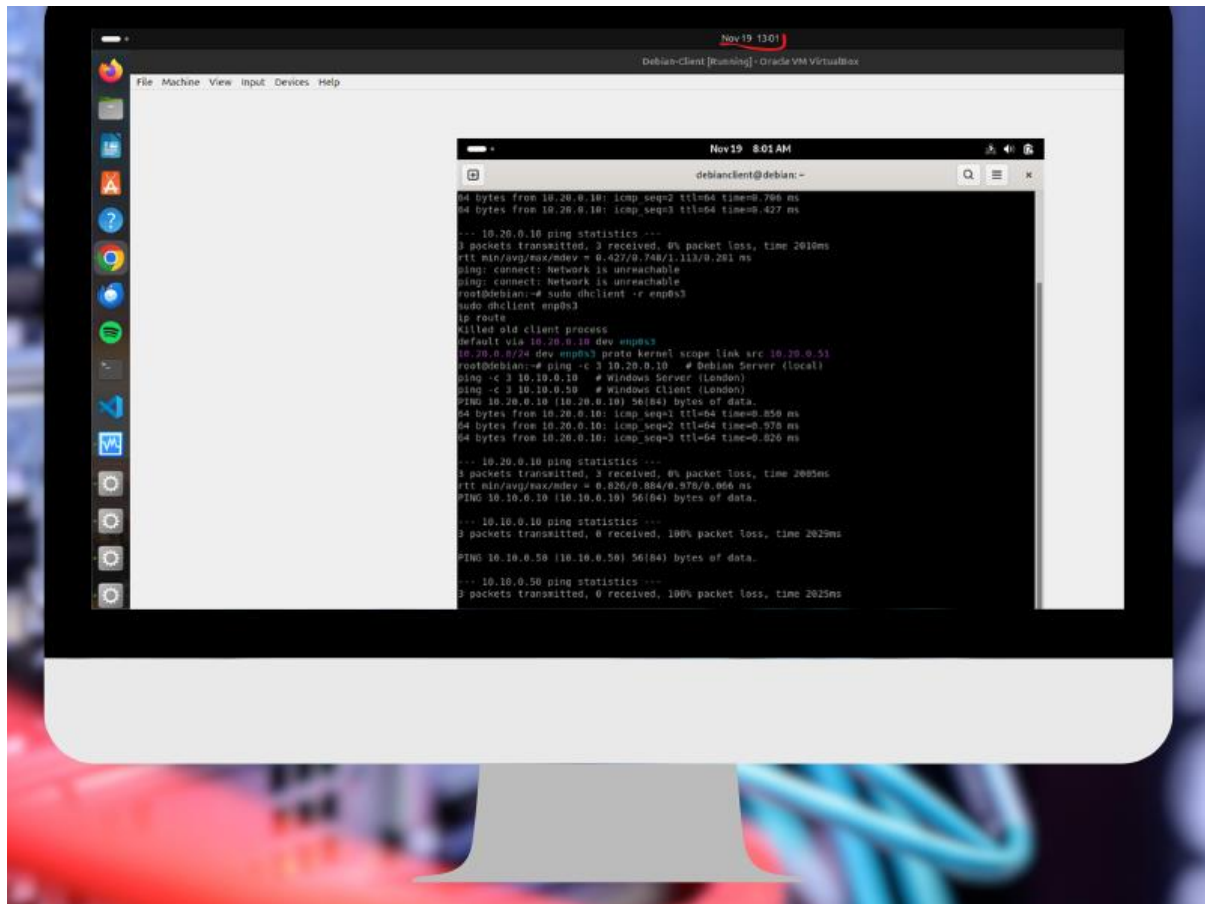
Link: [ScreenShotLink](#)

Screenshot 18: Windows Server → Debian Client and Server



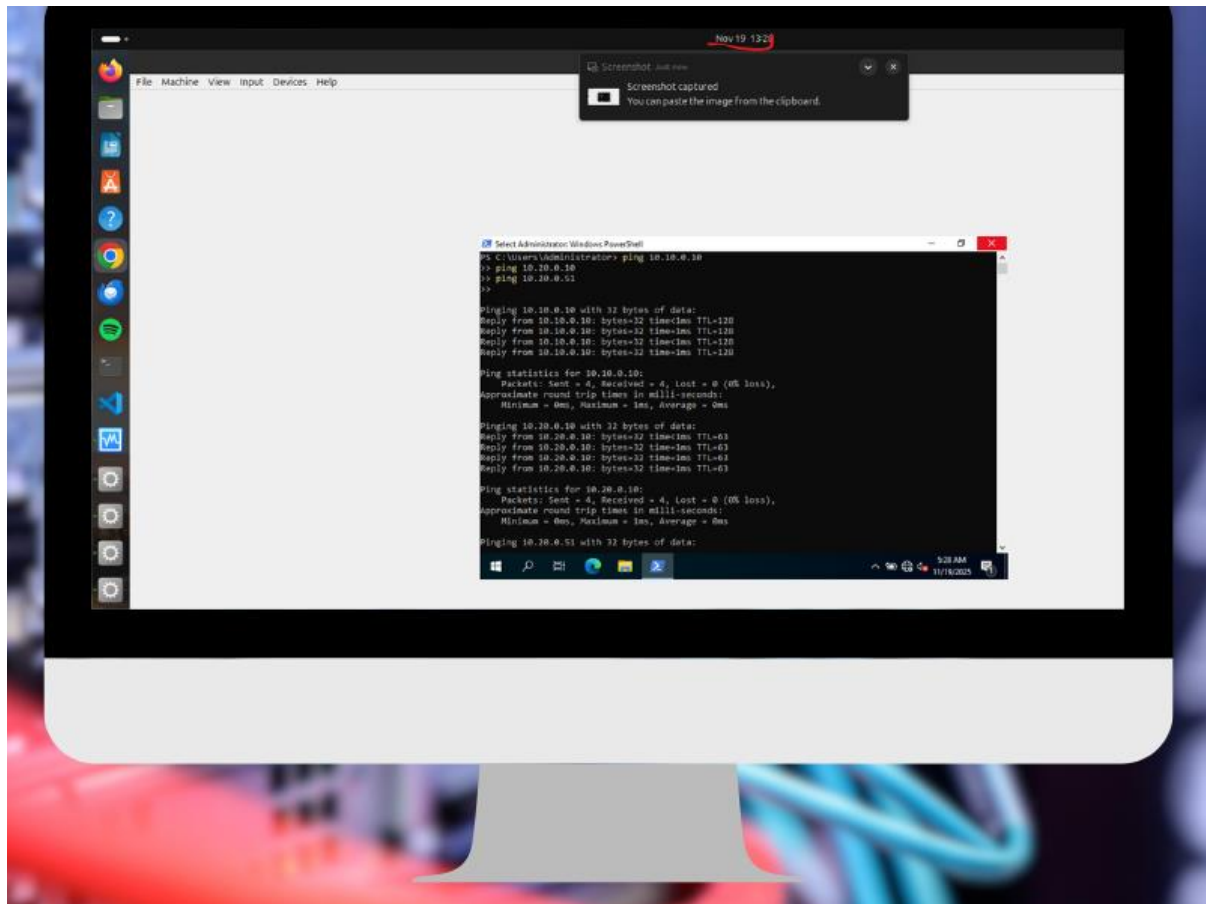
Link: [ScreenShotLink](#)

Screenshot 19: Debian Client → both Windows Machines



Link: [ScreenShotLink](#)

Screenshot 20: Windows client ↔ Debian Server and client



Link: [ScreenShotLink](#)

All ping tests succeeded once routing was configured, demonstrating that:

- Both sites' LANs were reachable.
- The inter-site link was functioning correctly.
- Traffic flowed bidirectionally across networks.
- Clients could reach systems on the opposite site.

2.7 Troubleshooting and Observations

During configuration and testing several problems with the VirtualBox setup were encountered. VirtualBox virtual network adapters set to internal did not create a link between the Debian and Windows guests, as evidenced by the ping command returning repeated "Destination Host Unreachable" error messages.

Troubleshooting involved using `ip -br link` to confirm which interfaces were in the DOWN or NO-CARRIER state. This pointed to incorrectly assigned VirtualBox network types, and a disabled cable on the link adapters. Relinking the adapter to the correct Internal Network (LondonLAN, SwanseaLAN and Interlink) and setting the Cable Connected parameter to yes cleared the problem.

Repeated interface undetection issues between mis-configured adapters, notably Intel PRO/1000 and VirtIO emulations on Debian, was due to driver mismatches. Setting both endpoints to the same type of adapter cleared the problem, allowing link level communications.

Conflicts with NAT interfaces that had been left on the host for a previous, but abandoned test were intermittent. VirtualBox adapters were removed for the NAT networks and this left the 3 internal networks completely isolated from each other, and the host machine. The configuration now gives full, two-way connectivity across all 3 links and static routing is working as expected.

2.8 Reflection

I have enjoyed building this virtual enterprise network, and configuring routers and static routes to connect internal LANs within and between subnets.. This project has reinforced how different network layers work together and how static routes are used in real enterprise systems to achieve both security and efficiency in internal connectivity and resource sharing.

It has been a challenge to implement this in VirtualBox: although the conceptual plan was clear, some problems arose in the multiple internal LAN configuration and it was necessary to take things step-by-step and isolate the issues (disconnected adapters, driver mismatch, routing failure, etc.). I had to divide the problem into more straightforward tasks and validate each network layer separately: IP configuration, link state, route propagation. I spent most time troubleshooting these connectivity problems, but the solution and diagnostic learning was also valuable for developing my problem-solving and system analysis skills.

In terms of outcomes beyond the practical, this project has enhanced my understanding of both server-client communication and the inter-dependencies of software systems on network architecture. For a future software developer, this knowledge helps to bridge application design with the server and network infrastructure. If I were to do it again, I would expand this setup by adding a dynamic routing protocol such as OSPF and a site-to-site VPN for automated route discovery and encrypted secure communication between sites.

References

- [1] V. Vasanthraj, "Security and Virtualization in the Data Center," *International Journal of Computer Applications*, vol. 975, pp. 1–5, 2010.
- [2] D. G. Raghunath *et al.*, "Security and Virtualization in the Data Center," *IEEE Computer Society*, 2010.
- [3] CiscoNet Training Solutions, "Dynamic Routing Protocols: OSPF, EIGRP, RIPv2, IS-IS, BGP," *Cisco Community*, Dec. 2021. [Online]. Available: <https://community.cisco.com/t5/networking-knowledge-base/dynamic-routing-protocols-ospf-eigrp-ripv2-is-is-bgp/ta-p/4511577>. **Accessed: 03-Nov-2025.**
- [4] M. Raza *et al.*, "An Overview of IP Addressing," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 4, no. 3, pp. 1–5, 2014.
- [5] PowerCert Animated Videos, "Difference between Public and Private IP Addresses," *YouTube*, 2021. [Online]. Available: <https://www.youtube.com/watch?v=po8ZFG0Xc4Q>. **Accessed: 05-Nov-2025.**
- [6] PowerCert Animated Videos, "DHCP Explained – Dynamic Host Configuration Protocol," *YouTube*, 2023. [Online]. Available: <https://www.youtube.com/watch?v=e6-TaH5bkjo>. **Accessed: 07-Nov-2025.**
- [7] Canadian Centre for Cyber Security, "Practitioner Guidance for Securing Microsoft Active Directory Services in Your Organization – ITSP.60.100," Ottawa, ON, Canada, Dec. 2023. [Online]. Available: <https://www.cyber.gc.ca/en/guidance/practitioner-guidance-securing-microsoft-active-directory-services-your-organization-itsp60100>. **Accessed: 09-Nov-2025.**
- [8] Red Hat, "Integrating RHEL Systems Directly with Windows Active Directory," *Red Hat Enterprise Linux 8 Documentation*, 2024. [Online]. Available: https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/8/html/integrating_rhel_systems_directly_with_windows_active_directory. **Accessed: 11-Nov-2025.**
- [9] Cisco Networking Academy, "Routing Basics and Protocols," *Cisco Press*, 2021. [Online]. Available: <https://www.cisco.com>. **Accessed: 13-Nov-2025.**
- [10] VMware, "Understanding Hypervisors: Type 1 and Type 2 Virtualization," *VMware Docs*, 2022. [Online]. Available: <https://www.vmware.com>. **Accessed: 14-Nov-2025.**

[11] Microsoft, “Windows Server Virtualization and Hyper-V Overview,” *Microsoft Learn*, 2023. [Online]. Available: <https://learn.microsoft.com/en-us/windows-server/virtualization/hyper-v>. **Accessed: 15-Nov-2025.**