Mansoura University

Faculty of Engineering

Electronics and Communications Department

Network Automation and Monitoring using DevOps tools

SUBMITTED BY

Members

- **MAHMOUD REDA EL-GHREEB**
- **MAHMOUD SHABAAN MOHAMED**
- **ABD EL-RAHMAN REDA MUSTAFA**

SUPERVISED BY

Assoc.Prof.Dr.

- **EMAN MAHMOUD**
- **Dr. Fatma Alzahra AL-Agery**
- **Dr. Asmaa Helmy**

Abstract

In today's fast-paced digital landscape, organizations require network infrastructures that are not only scalable and reliable but also automated and secure. Manual approaches to network management have proven insufficient in the face of increasing complexity, demand for uptime, and rapid threat evolution.

This project, titled Network Automation and Monitoring Using DevOps Tools, presents an integrated approach to enhancing network operations by adopting modern DevOps practices. The project addresses key challenges through the following pillars:

- **Automation:**

  Using Ansible, we implemented automated configuration management across routers, switches, and end devices. This reduced manual intervention, minimized errors, and ensured consistent network states.

- **Monitoring and Observability:**

  The system leverages Prometheus for real-time metric collection and Grafana for intuitive visualization. Additional tools such as Loki, Promtail, Node Exporter, and SNMP Exporter were integrated to extend observability and collect logs and system data from multiple sources.

- **Security and Alerting:**

  Snort was deployed as a Network Intrusion Detection System (NIDS) to identify and classify suspicious traffic. Alertmanager was integrated to generate and route alerts based on predefined security conditions and network anomalies.

To simulate a realistic enterprise network, we built the infrastructure using GNS3 and VMware Workstation, including multiple departments connected through VLAN segmentation. The entire environment was designed to reflect real-world enterprise conditions, enabling us to evaluate the system's reliability, scalability, and responsiveness.

Overall, this project demonstrates how combining traditional networking principles with DevOps methodologies can transform static infrastructures into intelligent, self-managed, and adaptive networks. The outcome includes enhanced efficiency, improved incident handling, and a resilient foundation for fu

Acknowledgment

At the outset, we would like to extend our profound gratitude to **Dr. Eman Mahmoud**, whose guidance, encouragement, and invaluable feedback have been instrumental in the success of this project. Your unwavering support inspired us to push beyond our limits and achieve more than we imagined possible.

We also extend our sincere thanks to all the professors and mentors at **Mansoura University, Faculty of Engineering, Department of Electronics and Communications Engineering**, whose teachings laid the foundation for our technical and professional growth throughout our academic journey.

Our heartfelt appreciation goes to our families and friends, who stood beside us with endless encouragement, patience, and unwavering belief in our abilities. Their support fueled our determination during both the highs and lows of this project.

A special acknowledgment goes to each member of our team. Through collaboration, dedication, and resilience, we transformed ideas into reality. Each contribution, no matter how small, played a vital role in making this project a success.

This project is not merely the result of individual efforts, but the outcome of the collective spirit of everyone who believed in us, guided us, and supported us.

To all of you — thank you.
This achievement is as much yours as it is

CONTENTS

..... 1

Contents 4

list of tables 8

list of figures 10

CHAPTER 1 : Introduction 15

1.1 Preface 16

1.2 Problem Statement 16

1.3 Objectives 19

1.4 Methodology 20

1.5 Benefits of the Hierarchical Network Model 21

1.6 Traditional Networking vs Automated Networking 22

1.7 Expected Results 24

1.8 Prerequisites 24

1.9 Thesis Outline 26

1.10 Team Members & Responsibilities 26

CHAPTER 2 : Tools and Technologies (Protocols) 29

2.1 Network technologies/mechanisms 30

2.1.1 Virtual Local Area Network 30

2.1.2 ACCESS CONTROL LIST ACL 32

2.1.3 Routing Protocols 33

2.1.4 IPsec Protocol 35

2.1.5 DHCP Protocol 37

2.1.6 DNS Protocol 38

2.1.7 NAT (Network Address Translation) 40
2.1.8 TFTP SERVER 42
2.1.9 Availability & HSRP(Hot Standby Router Protocol) 43
2.1.10 Ether-Channel 45
2.1.11 SSH Protocol 48
2.2 Best practices for using the hierarchical network model 49
2.3 Examples of using the hierarchical network model 50
2.4 Network Topology HQ and branches 54
2.5 Hierarchical network model 56
2.6 Network problems and solutions 58
2.7 Network security 62
2.7.1 Router Security Configuration Commands 63
2.7.2 Snort 68
2.7.3 Types of Attacks Simulated in the Project 68
2.8 Simulation Tools 71
2.8.1 GNS3 (Graphical Network Simulator 3) 71
2.8.2 VM ware workstation 73
2.8.3 Required resources (GNS3 Image settings) 74
2.8.1 IOS image installation 83
2.9 Operating Systems Used 85
2.9.1 Ubuntu 85
2.9.2 Kali Linux 86
2.10 Automation and Monitoring Tools 87
2.10.1 Ansible 87

2.10.2 Prometheus 88
2.10.3 Grafana 89
2.10.4 Loki 90
2.10.5 Promtail 91
2.10.6 Node Exporter 91
2.10.7 SNMP Exporter 92
2.10.8 Alert Manager 93
CHAPTER 3 : Simulation Environment 94
3.1 Introduction 95
3.2 NETWORK TOPOLOGY DESIGN 95
3.2.1 Departmental Segmentation 96
3.2.2 VLAN Design and Implementation 97
3.3 Interconnection between Routers and Switches using MLS 98
3.4 Network Configuration 100
3.4.1 IP Addressing Scheme 100
3.4.2 Addressing Table 100
3.5 Switch & routres Configuration 106
3.6 Connectivity Testing 130
CHAPTER 4 : Monitoring System Implementation 133
4.1 kali linux 134
4.2 Ansible 141
4.3 Prometheus 146
4.4 Snort 151
4.5 LOKI 152

4.6 Promtail 154
4.7 Grafana Dashboard Setup 155
4.8 Alerts Manager 156
4.9 Testing and Results 161
4.9.1 Test connectivity between devices 161
4.9.2 ping from kali to all devices 172
4.10 Monitoring 178
4.10.1 PROMETHEUS 178
4.10.2 Grafana Dashboard 181
4.10.3 node exporter dashboard 182
4.10.4 snmp exporter dashboard 184
4.10.5 loki dashboard 187
CHAPTER 5 : Conclusion 189
5.1 Project Outcomes 190
5.2 Limitations 190
5.3 Challenges and Lessons Learned 191
5.4 SUMMARY 193
5.5 Future Work and Recommendations 194
REFERENCES 195

LIST OF TABLES

Table 2-1 VLAN Configuration 31

Table 2-2 ACL Configuration 32

Table 2-3 Open Shortest Path First (OSPF) 33

Table 2-4 Routing Information Protocol (RIP) 35

Table 2-5 Enable DNS Server 38

Table 2-6 . Enable Domian Lookup 39

Table 2-7 Public Name Server Configuration 39

Table 2-8 . Host / IP Address Mappings 39

Table 2-9 Interface IP Configuration 40

Table 2-10 Network Address Translation (NAT) 41

Table 2-11 Trivial File Transfer Protocol (TFTP) 42

Table 2-12 EtherChannel 46

Table 2-13 Layer 3 EtherChannel 46

Table 214- Table SSH Protocol 49

Table 2-15 HIERARCHICAL NETWORK MODEL 57

Table 2-16 NETWORK PROBLEMS AND SOLUTIONS 59

Table `hostname` 217- 63

Table `security passwords` 218- 64

Table 2-19 Router Security Configuration 64

Table Secret password 220- 66

Table 2-21 PORT SECURITY 67

Table 3-1:Addressing Table 100

Table 3-2 Multi-Layer Switch 1 106

Table 3-3 Multi-Layer Switch 2 110

Table 3-4 Router1 114

Table 3-5 Access switch-1 116

list of tables

Table 3-6 Accessswitch -2 117
Table 3-7 Access switch-3 118
Table 3-8 Access switch-4 119
Table 3-9 Access switch-5 119
Table 3-10 Access switch-6 120
Table 3-11 Router 2 121
Table 3-12 Router 3 122
Table 3-13 Multi-Layer Switch 3 (iou10) 125
Table 3-14 Multi-Layer Switch 4 (iou11) 127
Table 3-15 Access switch -8 129
Table 3-16 Access switch-9 129
Table 3-17 Security Hardening 131
Table 4-1 ANSIBLE INSTALLATION 143
Table 42- Installing Prometheus 146
Table 43- Node Exporter 149
Table 44- Snort 151
Table 45- Loki 152
Table 46- Promtail 154
Table 47- GRAFANA DASHBOARD 156
Table 48- ALERTS MANAGER 156

LIST OF FIGURES

Figure 1-1 : project image.....21

Figure 2-1 TOOLS AND TECHNOLOGIES.....29

Figure 2-2 Virtual Local Area Network in MLS.....30

Figure 2-3 Virtual Local Area Network in router.....30

Figure 2-4 Hot Standby Router Protocol.....44

Figure 2-5 Enterprise Data Center Infrastructure.....47

Figure 2-6 Data Center's Aggregation Layer.....47

Figure 2-7 EtherChannels.....48

Figure 2-8 Cisco Hierarchical Network Model Layers.....51

Figure 2-9 Enterprise Campus Module Components.....53

Figure 2-10 Enterprise Edge Components.....54

Figure 2-11 Network Topology.....55

Figure 2-12 Hierarichal topology.....56

Figure 2-13 port scan.....69

Figure 2-14 Dos attack.....70

Figure 2-15 DNS.....70

Figure 2-16 Ping Sweep (ICMP).....71

Figure 2-17 router.....72

Figure GNS3 GUI layout 218-.....73

Figure 2-19 VM ware workstation.....73

Figure 2-20 Hierarchical Design.....75

Router Console Access 221-.....77

Figure 2 22-CLI access.....78

Figure 2-23 : access switch gns3.....78

Figure 2-24 Distribution switch gns3.....79

list of figures

Figure 2-25 core- switch gns3 81

Figure 226- : GNS3 IOS image installation.....83

Figure 227- : GNS3 IOS image setup.....84

Figure 2-28 Ubuntu 85

Figure 2-29 Kali linux 86

Figure 2-30 Ansible 87

Figure 2-31 Prometheus.....88

Figure 2-32 Grafana.....89

Figure 2-33 Loki.....90

Figure 2-34 SNMP Exporter.....92

Figure 2-35 Alert Manager 93

Figure 3-1simulation topology 96

Figure 4-1 Kali Linux Add 134

Figure 4-2 Kali Linux Setting.....135

Figure 4-3 Kali Linux VM.....135

Figure 4-4 Kali Linux Adapter NAT 137

Figure 4-5 Kali Linux Adapter 138

Figure 4-6 Kali Linux preferences Vnet5 138

Figure 4-7 Kali Linux GNS3 139

Figure 4-8 Kali Linux install 139

Figure 4-9 Kali Linux 140

Figure 4-10 Kali Linux configration.....140

Figure 4-11 Ansible host 145

Figure 4-12 Ansible outcome 145

Figure 4-13 Prometheus Configuration 147

Figure 4-14 Prometheus Configuration 148

Figure 4-15 Node Exporter.....150

list of figures

. Figure 4-16 snmp exporter 151
Figure 4-17 Loki 153
Figure 4-18 Promtail 155
Figure 4-19 GRAFANA DASHBOARD SETUP 156
Figure 4-20 ALERTS MANAGER 158
Figure 4-21 Node and Snort Alerta 159
Figure 4-22 ALERTS MANAGER 160
Figure 4-23 Test DHCP in pc1 161
Figure 4-24 Test DHCP in pc2 161
Figure 4-25 Ping PC1 VLAN 10 VRRP 161
Figure 4-26 Ping PC1 (VLAN 10) to VLAN 20 162
Figure 4-27 Show ip route command in R3 163
Figure 4-28 Show ip route command Dist 1 163
Figure 4-29 Show ip route command core 1 (R1) 164
Figure 4-30 Show ip route command dist 2 165
Figure 4-31 Test connectivity from Pc1 to Dist 4 166
Figure 4-32 Test connectivity from Pc1 to Dist 3 166
Figure 4-33Test connectivity from Pc1 to vrrp in another campus 167
Figure 4-34 connectivity from pc1 to R3 in another campus 167
Figure 4-35 Test connectivity from pc1to R1 168
Figure 4-36 Test connectivity from pc1 to vrrp 169
Figure 4-37 Test connectivity from pc2 to vrrp in another campus 170
Figure 4-38Test connectivity from dist 1 to R1 170
Figure 4-39 Test connectivity from Dist 1 to R3 171
Figure 4-40 ping from kali 172
Figure 4-41 PROMETHEUS target 178
Figure 4-42 Alerts 179

list of figures

Figure 4-43 Grafana Dashboard 181

Figure 4-44 node exporter 182

Figure 4-45 snmp exporter.....184

Figure 4-46 loki dashboard.....187

Abbreviation

Abbreviation	Full Form
GNS3	Graphical Network Simulator 3
VMware	Virtual Machine Software
VM	Virtual Machine
VLAN	Virtual Local Area Network
VRRP	Virtual Router Redundancy Protocol
RIP	Routing Information Protocol
IPSec	Internet Protocol Security
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
SSH	Secure Shell
HTTP	Hyper Text Transfer Protocol
Ansible	Automation Configuration Tool
Snort	Intrusion Detection System
Prometheus	Monitoring and Alerting Toolkit
Grafana	Open-Source Analytics & Monitoring Platform
YAML	YAML Ain’t Markup Language (used in Ansible)
VPN	Virtual Private Network
NIC	Network Interface Card
LAN	Local Area Network
WAN	Wide Area Network
IOT	Internet of Things

CHAPTER 1 : INTRODUCTION

CHAPTER 1 : Introduction

1.1 PREFACE

As technology continues to evolve, network infrastructures are required to be more robust, intelligent, and secure. To meet these expectations, manual network management is no longer sufficient. Instead, the integration of DevOps principles into networking—known as NetDevOps—is emerging as a powerful approach that combines automation, monitoring, and security into a single unified system.

Our graduation project, titled "Network Automation using DevOps Tools," was designed to demonstrate how modern tools can be applied to traditional networks to achieve a smarter, more reliable, and self-maintaining infrastructure. By leveraging simulation tools like GNS3, virtualization platforms such as VMware, and DevOps tools like Ansible, Prometheus, and Snort, we built a fully automated and monitored enterprise network environment capable of detecting intrusions and sending real-time alerts.

1.2 PROBLEM STATEMENT

Traditional network environments suffer from several significant limitations:

- **Manual Configuration**
  - Device setup and management rely heavily on manual input.
  - This approach is time-consuming, repetitive, and highly prone to human error.
- **Limited Visibility**
  - Lack of real-time monitoring tools causes delays in detecting performance issues.
  - Administrators often become aware of problems only after users report them.
- **Delayed Threat Detection**
  - Most traditional networks lack proactive security mechanisms.
  - Threats and intrusions may go unnoticed until damage has occurred.
- **Scalability Challenges**

- Managing large numbers of devices manually is inefficient and unsustainable.
- Expanding the network increases the risk of misconfiguration and downtime.

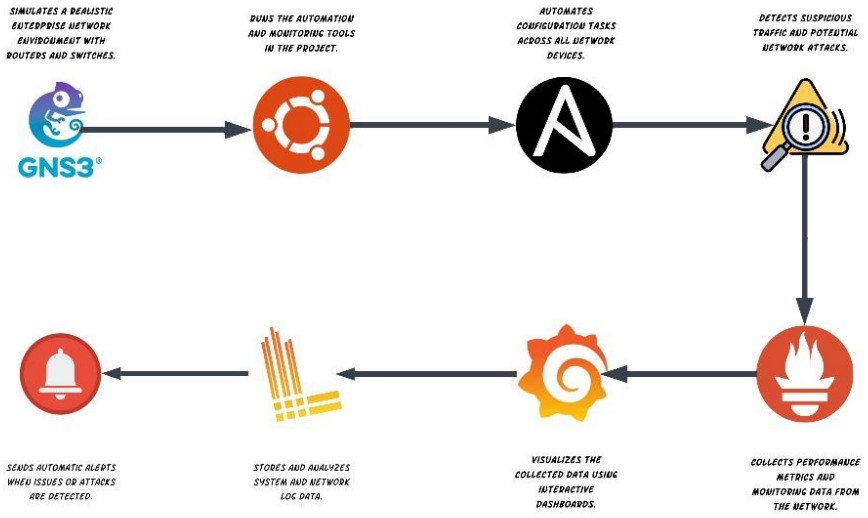
Our Solution

To overcome these challenges, our project proposes a comprehensive solution that integrates modern DevOps tools and practices:

- **Automation with Ansible**
  - Reduced manual configuration time by automating device setup.
  - Improved consistency and minimized errors in network deployment.
- **Monitoring with Prometheus and Grafana**
  - Enabled real-time performance tracking and alerting.
  - Provided administrators with a clear and interactive view of network health.
- **Security with Snort and Loki**
  - Implemented real-time intrusion detection and log analysis.
  - Enhanced security response and visibility across the network.
- **Simulation with GNS3**
  - Allowed us to create a realistic and complex multi-device environment.
  - Tested the solution in scenarios that closely mimic real-world networks.

Despite the complexity of building and integrating these tools, especially in a simulated environment, we succeeded in creating a scalable, secure, and automated network infrastructure. This would not have been possible without leveraging open-source DevOps technologies, which dramatically reduced the time, effort, and risk associated with traditional methods.

Proposed Solution



To tackle the challenges of traditional network management, our project followed a systematic and structured roadmap that aligns with DevOps methodologies. The roadmap ensures automation, monitoring, and security within a simulated yet realistic network environment. The key phases were as follows:

1. Network Environment Design and Simulation (GNS3)

Designed a scalable and modular virtual network using GNS3 to replicate real-world topologies.

Included multiple routers, switches, and Linux-based nodes to serve as monitoring, automation, and attack-detection endpoints.

2. Infrastructure Automation (Ansible)

Developed and executed Ansible playbooks to automate device configuration, reducing manual workload.

Ensured consistent deployments and minimized human errors across the infrastructure.

3. Monitoring System Implementation (Prometheus & Grafana)

Integrated Prometheus to collect time-series data from devices and services.

Built interactive Grafana dashboards to provide real-time insights, alerting, and network health visualization.

4. Security Framework Deployment (Snort & Loki)

Employed Snort for real-time network intrusion detection and attack classification.

Integrated Loki to aggregate and analyze logs for improved security visibility and threat correlation.

5. System Validation and Testing

Conducted various operational and attack scenarios to assess the system's performance, responsiveness, and reliability.

Verified the effectiveness of automation, monitoring, and security mechanisms under simulated production-like conditions.

Despite the complexity of integrating multiple open-source tools, our DevOps-driven approach enabled the creation of a robust, secure, and automated network management framework with reduced time, effort, and operational risks.

1.3 OBJECTIVES

The main objective of this project is to design and implement a **scalable**, **secure**, and **automated** network infrastructure using modern **DevOps tools**. The specific objectives are:

- Design an enterprise-level network topology using **GNS3** to simulate real-world environments.
- Configure and interconnect **routers**, **switches**, and **virtual machines** to reflect multi-department setups.

- Implement **Ansible** to automate network configuration and repetitive administrative tasks.
- Install and configure **Prometheus** and **Grafana** to monitor network performance metrics in real-time.
- Deploy **Snort** to detect, analyze, and respond to suspicious or malicious network traffic.
- Integrate **Alert Manager** to send automated alerts to administrators upon detecting issues or threats.
- Test the system using **simulated cyberattacks** and validate its ability to respond and generate proper alerts.

1.4 METHODOLOGY

The project was developed through a structured methodology divided into key phases:

- **Environment Setup**
  - Install and configure **GNS3**, **VMware Workstation**, and all necessary virtual machines.
- **Network Design**
  - Create a detailed network topology simulating multiple departments such as **Finance**, **IT**, and **Students**.
  - Incorporate routers, switches, and monitoring nodes to represent a realistic enterprise infrastructure.
- **Device Configuration**
  - Assign IP addresses to all network devices.
  - Configure routing protocols and ensure **end-to-end connectivity** across the network.
- **Automation Deployment**
  - Use **Ansible playbooks** to push configurations automatically and manage devices efficiently.
- **Monitoring Integration**
  - Deploy **Prometheus** for data collection and **Grafana** for visualizing metrics on dynamic dashboards.
- **Security Implementation**
  - Install and configure **Snort** as an Intrusion Detection System (IDS).
  - Set up **Alert Manager** for real-time notification of security threats and system issues.

- **Testing and Validation**
  - Perform **simulated attacks** using **Kali Linux**.
  - Analyze the network's response and validate detection and alerting mechanisms.

Network Topology Diagram

The following figure illustrates the network topology designed and implemented for the project:

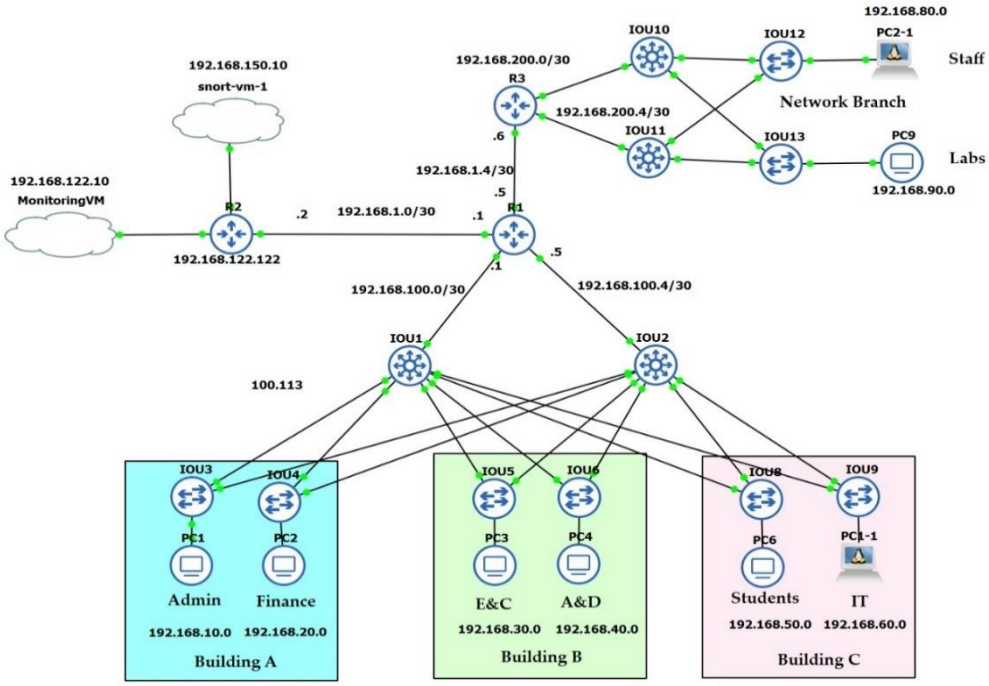


Figure 1-1 : project image

1.5 BENEFITS OF THE HIERARCHICAL NETWORK MODEL

The hierarchical network model is a design framework that organizes a network into distinct layers, each with specific functions. Typically, these layers are the Access Layer, Distribution Layer, and Core Layer. Implementing this model offers several key advantages:

Scalability: By segmenting the network into layers, it becomes easier to expand without disrupting existing services.

Improved Performance: Each layer can be optimized for its specific role, reducing bottlenecks and improving overall network efficiency.

Simplified Troubleshooting: A structured layout makes it easier to isolate and diagnose network problems.

Enhanced Security: The model enables the implementation of security policies at multiple points within the network, limiting the spread of attacks.

Flexibility and Resilience: Modular design allows for easier upgrades and enhances network availability by isolating failures.

This approach aligns perfectly with modern network automation principles, making it an ideal foundation for integrating DevOps tools like Ansible, Prometheus, and Snort.

1.6 TRADITIONAL NETWORKING VS AUTOMATED NETWORKING

In traditional networking environments, network devices like routers and switches are configured manually, often using command-line interfaces. Tools like Packet Tracer simulate such networks, where each device is set up individually, and every configuration change requires human intervention.
This approach, while educational and necessary for learning core networking concepts, becomes highly inefficient and error-prone in real-world large-scale networks.

On the other hand, automated networking — as implemented in our project — relies on DevOps tools such as **Ansible**, **Prometheus**, and **Grafana** to manage, monitor, and secure the network. Instead of manually configuring each device, we use automation scripts (playbooks) to push configurations to multiple devices at once, ensuring consistency, speed, and reducing human errors.

Moreover, monitoring systems like Prometheus and Grafana provide real-time visibility into network performance without the need for manual checking, and security systems like Snort automatically detect intrusions, improving network resilience and reducing downtime.

Thus, the transition from traditional networking to automated, DevOps-enabled networking represents a significant leap towards scalability, efficiency, and security.

Aspect	Traditional Networking (Packet Tracer)	Automated Networking (Using DevOps Tools)
Configuration	Manual, device-by-device using CLI	Automated using Ansible Playbooks across multiple devices
Scalability	Difficult to scale, time-consuming	Easy to scale, fast deployment of changes
Error Rate	High probability of human errors	Minimal errors due to standardized scripts
Monitoring	Manual checks or basic simulations	Real-time monitoring with Prometheus and Grafana dashboards
Security	Requires manual configuration and monitoring	Automated intrusion detection using Snort and alert management
Resource Efficiency	High human effort required	Saves time and reduces administrative overhead
Flexibility	Changes take time and often require downtime	Rapid updates and minimal downtime with automation
Data Visualization	Limited, no built-in monitoring	Rich dashboards and alerts (Grafana, Alert Manager)

1.7 EXPECTED RESULTS

Following the successful implementation of the project, the following results were achieved:

- **A fully operational enterprise-grade network** was simulated using **GNS3** and **VMware**, accurately replicating a real-world multi-departmental organizational structure, including Finance, IT, and Students.
- **Network device configuration and management** were automated using **Ansible**, significantly reducing manual work, ensuring consistency across devices, and eliminating human errors in repetitive tasks.
- **Real-time performance monitoring** was achieved through the integration of **Prometheus** and **Grafana**, providing live dashboards that display metrics such as CPU usage, memory load, and device availability—enhancing visibility and proactive response.
- **Intrusion detection capabilities** were implemented using **Snort**, which successfully detected a variety of simulated network attacks (e.g., port scanning, brute-force attempts) and generated immediate alerts.
- An **automated alerting mechanism** was established using **Alert Manager**, which sent real-time notifications upon detecting system failures, suspicious activities, or performance anomalies—enabling fast incident response.
- The final network design demonstrated **superior scalability, maintainability, and security** compared to traditional manual setups, proving the value of adopting DevOps practices in network operations.
- **All phases of the project were documented comprehensively**, including detailed descriptions of the topology, configuration steps, automation scripts, test scenarios, and evaluation outcomes.
- The project resulted in a **reproducible model** that serves as a solid foundation for future enhancements, such as the integration of **machine learning-based threat detection** or **cloud-based network extensions**.

1.8 PREREQUISITES

To successfully design, implement, and operate the network automation and monitoring system, the following prerequisites must be fulfilled:

- **Technical Knowledge:**

- Basic to intermediate knowledge of networking concepts (IP addressing, routing, switching).
- Familiarity with network simulation environments like GNS3.
- Understanding of Linux system administration and basic command-line usage.
- Fundamental knowledge of automation concepts and tools such as Ansible.

- **Software Requirements:**
  - GNS3 installed and configured for network simulation.
  - VMware Workstation for running virtual machines (e.g., Kali Linux, Ubuntu servers).
  - Ansible installed for automation tasks.
  - Prometheus and Grafana installed for monitoring and visualization.
  - Snort installed for intrusion detection.
  - Alert Manager configured for handling notifications.
  - Node Exporter and SNMP Exporter installed to collect system and network device metrics.
- **Hardware Requirements:**
  - A computer with sufficient processing power, RAM (at least 16 GB recommended), and disk space to run multiple virtual machines and simulations.
  - Stable internet connection for downloading packages and updates.
- **Other Requirements:**
  - Clear project plan and division of responsibilities among team members.
  - Backup and recovery plan for virtual machines and network configurations.
  - Time allocation for testing, troubleshooting, and documentation.

1.9 THESIS OUTLINE

This thesis is organized into several chapters, each covering a critical aspect of the project:

Chapter One: Introduction

Provides background information, defines the problem, states the objectives, describes the methodology, and highlights the benefits of a hierarchical network model.

Chapter Two: Literature Review

Discusses previous research and studies related to network automation, monitoring, and security using DevOps tools.

Chapter Three: System Design and Implementation

Details the network topology, device configurations, automation scripts, monitoring setups, and security deployments.

Chapter Four: Testing and Results

Describes the testing methodology, attack simulations using Kali Linux, system responses, and evaluation of performance.

Chapter Five: Conclusion and Future Work

Summarizes the project's outcomes, lessons learned, and potential directions for further development.

1.10 TEAM MEMBERS & RESPONSIBILITIES

The project was carried out by a team of three members, each responsible for a specific domain within the project lifecycle:

- ***Mahmoud Shaaban Mohamed – Network Engineer***

  Led the initial phases of the project, including defining the problem scope, designing the overall network topology, and configuring routing and switching protocols. He was also responsible for implementing VLAN segmentation and ensuring interdepartmental communication.

- ***Abdelrahman Reda Mostafa – Automation & Security Engineer***

  Handled the automation layer of the project using Ansible Playbooks for device configuration. He also managed the security aspect by deploying Snort for intrusion detection, simulating cyberattack scenarios, and configuring Alertmanager to generate real-time alerts.

- ***Mahmoud Reda Elghreeb Mohamed – Team Leader & Monitoring Engineer***

  Oversaw the full project execution, coordinated team responsibilities, and managed documentation. He was responsible for implementing system monitoring using Prometheus and Grafana, integrating log analysis tools like Loki and Promtail, and proposing future improvements and scalability plans.

CHAPTER 2 : TOOLS AND TECHNOLOGIES (PROTOCOLS)

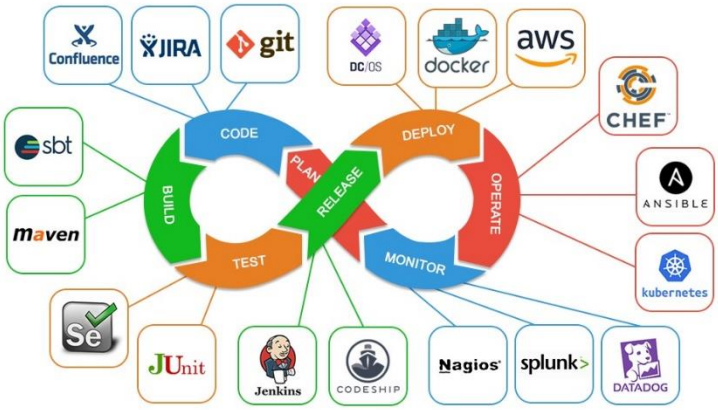


Figure 2-1 TOOLS AND TECHNOLOGIES

2.1 NETWORK TECHNOLOGIES/MECHANISMS

2.1.1 Virtual Local Area Network

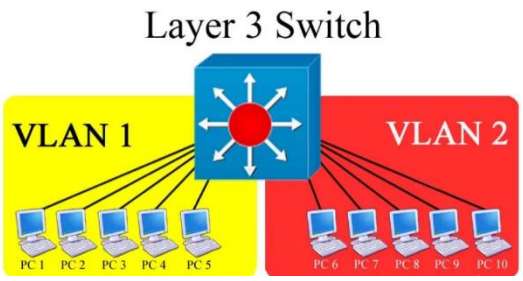


Figure 2-2 Virtual Local Area Network in MLS

Inter-VLAN routing is a method used to route traffic between different VLANs (Virtual Local Area Networks) on a network. This is essential because devices on different VLANs cannot communicate directly due to broadcast domain limitations. Here's an overview of how to set up inter-VLAN routing, particularly in a GNS3 environment:

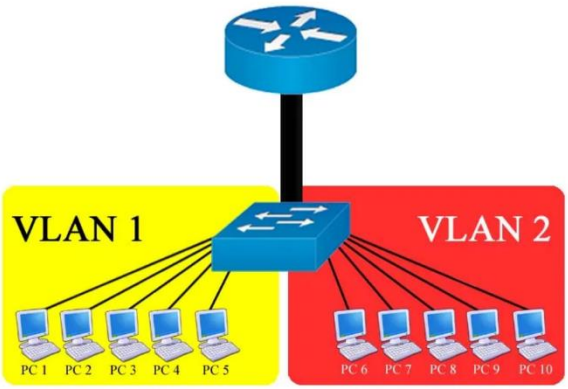


Figure 2-3 Virtual Local Area Network in router

Key Concepts

- **VLAN**: A VLAN is a logical subgroup within a larger network, allowing for segmentation and organization of devices.
- **Router-on-a-Stick**: This is a common method for inter-VLAN routing using a single router interface. The router uses subinterfaces to handle traffic for different VLANs.
- **Layer 3 Switch**: If you have a Layer 3 switch, it can handle inter-VLAN routing internally without needing a separate router.

Table 2-1 VLAN Configuration

```
enable
configure terminal
interface g0/0.10
encapsulation dot1Q 10
ip address 192.168.10.1 255.255.255.0
exit

interface g0/0.20
encapsulation dot1Q 20
ip address 192.168.20.1 255.255.255.0
exit
```

2.1.2 ACCESS CONTROL LIST ACL

Access Control Lists (ACLs) are a fundamental part of network security, allowing you to control the traffic entering or leaving a network device based on specified criteria. They are commonly used on routers and switches to filter traffic and enforce policies. Here's an overview of ACLs and how to configure them, especially in a GNS3 environment.

Types of ACLs:

- **Standard ACLs**: Filter traffic based on source IP addresses only. These are numbered from 1 to 99 and 1300 to 1999.
- **Extended ACLs**: Filter traffic based on source and destination IP addresses, protocol types, and port numbers. These are numbered from 100 to 199 and 2000 to 2699.

ACL Configuration: ACLs can be applied to interfaces in either inbound or outbound directions:

- **Inbound**: Traffic is filtered before it enters the interface.
- **Outbound**: Traffic is filtered after it exits the interface.

Table 2-2 ACL Configuration

```
R1 (config) #ip access-list extended TVTC_ACL

R1 (config-ext-nacl) # deny tcp any any eq telnet

R1 (config-ext-nacl) # deny tcp any any eq www

R1 (config-ext-nacl) # permit ip any any

R1 (config-ext-nacl) # deny ip any any

R1 (config-ext-nacl) # interface Serial0/1/0

R1 (config-if) # ip access-group TVTC_ACL in

R1 (config-if) #exit
```

2.1.3 Routing Protocols

OSPF (Open Shortest Path First)

OSPF is a widely used link-state routing protocol that operates within an Autonomous System (AS). It efficiently finds the best path for data packets by considering various factors, such as link cost and network topology. OSPF is especially suitable for large enterprise networks due to its scalability and fast convergence. Here's an overview of OSPF, its configuration, and how to set it up in GNS3.

Areas: OSPF uses a hierarchical design with areas to optimize routing. The backbone area (Area 0) is the core of the OSPF network.

Table 2-3 Open Shortest Path First (OSPF)

```
router ospf 1

network 192.168.1.0 0.0.0.255 area 0

network 192.168.2.0 0.0.0.255 area 0

exit
```

RIP (Routing Information Protocol)

RIP is a distance-vector routing protocol used to exchange routing information within an Autonomous System (AS). It is one of the oldest routing protocols and is known for its simplicity and ease of configuration, making it suitable for small to medium-sized networks.

RIP determines the best path based on hop count, where the route with the fewest number of hops is considered the best. The maximum hop count is 15, and any route beyond that is considered unreachable.

Versions:

RIP v1: Does not support classless routing and does not send subnet mask information.

RIP v2: Supports classless routing, sends subnet mask information, and includes authentication features.

Advantages:

Easy to configure.

Suitable for small networks.

Uses UDP on port 520.

Limitations:

Slower convergence compared to protocols like OSPF or EIGRP.

Not ideal for large networks due to the 15-hop limit.

Configuration in GNS3:

Open your project in GNS3 and add a router (e.g., Cisco 7200).

Access the router CLI.

Enter the following commands:

bash

Copy

Edit

Table 2-4 Routing Information Protocol (RIP)

```
Router> enable
Router# configure terminal
Router(config)# router rip
Router(config-router)# version 2
Router(config-router)# network 192.168.1.0
Router(config-router)# network 10.0.0.0
Router(config-router)# no auto-summary
Router(config-router)# exit
Router(config)# exit
Router# write memory
```

Notes:

Use version 2 to enable RIP v2.

no auto-summary is important to prevent automatic summarization across discontiguous networks.

Ensure all relevant networks are included using the network command.

2.1.4 IPsec Protocol

IPsec (Internet Protocol Security) is a set of protocols used to secure communications over the internet or private networks by protecting data at the network layer (Layer 3) of the OSI model. The goal of IPsec is to provide confidentiality, integrity, and authentication for data transmitted across a network, thus enhancing security for data transmission between devices.

IPsec consists of several key components, such as:

Data Encryption:

Data is encrypted before being sent to ensure that it cannot be read by unauthorized parties. Encryption algorithms like AES or 3DES are used to achieve this goal.

Authentication:

The identity of the communicating parties is verified using techniques like HMAC (Hash-based Message Authentication Code). This process ensures that the data has not been tampered with during transmission.

Integrity:

Ensures that the data being sent has not been altered or corrupted during transmission between the parties.

Negotiation and Exchange:

Security parameters like the encryption algorithms used to secure the communication are negotiated between the parties using the IKE (Internet Key Exchange) protocol.

IPsec Modes:

Transport Mode:

In this mode, only the data (payload) is encrypted, not the source and destination addresses in the packet. This mode is typically used in private networks, such as between two devices in a VPN.

Tunnel Mode:

In this mode, the entire packet (both the payload and the headers) is encrypted and encapsulated in a new packet. This mode is mainly used for creating VPNs between networks.

Applications of IPsec:

IPsec is primarily used to create Virtual Private Networks (VPNs), allowing devices to securely connect to a private network over the internet.

It is also used to secure communications between routers, servers, and other devices within a network.

In summary, IPsec is a crucial tool for ensuring the security of data transmission across networks, used in various security applications such as VPNs, verifying identities, and ensuring data confidentiality.

2.1.5 DHCP Protocol

Dynamic Host Configuration Protocol (DHCP) is a network management protocol used to automate the process of configuring devices on IP networks. It dynamically allocates IP addresses and other network configuration parameters to devices (clients) so they can communicate on a network.

Types of DHCP Allocations

1. **Dynamic Allocation**:
   - **Description**: DHCP dynamically assigns IP addresses from a predefined pool to clients for a specific lease duration. When the lease expires, the IP address can be reallocated to another client.
   - **Use Case**: Ideal for networks with many devices that frequently connect and disconnect (e.g., guest networks).
2. **Automatic Allocation**:
   - **Description**: DHCP permanently assigns an IP address to a client. This address will always be leased to the same client based on its MAC address.
   - **Use Case**: Suitable for devices that require a fixed IP address, such as printers or servers.
3. **Manual Allocation (Static DHCP)**:

- **Description**: The DHCP server assigns IP addresses to clients based on their MAC addresses. The server retains the mapping of MAC addresses to IP addresses.
- **Use Case**: Useful when you want to ensure that specific devices always receive the same IP address.

2.1.6 DNS Protocol

DNS (Domain Name System) is a crucial part of the internet infrastructure that translates human-friendly domain names (like www.example.com) into IP addresses (like 192.0.2.1). This allows users to access websites and other resources using easy-to-remember names rather than numerical addresses.

Setting Up a DNS Server on a Cisco Router

1. Enable DNS Server
2. Enable Domian Lookup
3. Public Name Server Configuration
4. Host / IP Address Mappings
5. Router Interface IP Configuration
6. Hosts IP Configuration
7. Configuration Verification

Table 2-5 Enable DNS Server

RouterA # **configure terminal** RouterA (config)# **ip dns server**

Table 2-6 . Enable Domian Lookup

RouterA (config)# **ip domain lookup**

Table 2-7 Public Name Server Configuration

RouterA (config)# **ip name-server 192.168.30.89**
RouterA (config)# **ip name-server 192.168.30.90**

Table 2-8 . Host / IP Address Mappings

RouterA (config)# **ip host PC1 192.168.30.11**

RouterA (config)# **ip host PC2 192.168.30.12**

RouterA (config)# **ip host PC3 192.168.30.13**

RouterA (config)# **ip host PC4 192.168.30.14**

RouterA (config)# **ip host PC5 192.168.30.15**

RouterA (config)# **ip host PC6 192.168.30.16**

RouterA (config)# **ip host PC7 192.168.30.17**

RouterA (config)# **ip host PC8 192.168.30.18**

RouterA (config)# **ip host PC9 192.168.30.19**

RouterA (config)# **ip host PC10 192.168.30.20**

RouterA (config)# **ip host printer 192.168.30.21**

.

"**show hosts**"

Table 2-9 Interface IP Configuration

```
1. Router# show hosts
   Default domain is not set
   Name/address lookup uses domain service
   Name servers are 8.8.8.8
2. Codes: UN – unknown, EX – expired, OK – OK, ?? – revalidate
   temp – temporary, perm – permanent
   NA – Not Applicable None – Not defined
3. Host Port Flags Age Type Address(es)
   PC1 None (perm, OK) 0 IP 192.168.30.11
   PC2 None (perm, OK) 0 IP 192.168.30.12
```

2.1.7 NAT (Network Address Translation)

Network Address Translation (NAT) is a technique used to modify the IP address information in IP packet headers while in transit across a traffic routing device. It enables multiple devices on a local network to share a single public IP address for accessing the internet, thus conserving IP addresses and enhancing security.

Key Concepts of NAT

- **Types of NAT**:
  - **Static NAT**: Maps a single private IP address to a single public IP address. It is often used for servers that need to be accessed from outside the network.
  - **Dynamic NAT**: Maps a private IP address to a public IP address from a pool of available addresses. It allows for multiple devices to use a limited number of public IPs.

- **Port Address Translation (PAT)**: Also known as NAT overload, it allows multiple devices to share a single public IP address by using different ports.

- **How NAT Works**: When a device on a private network sends a packet to the internet, NAT translates the private IP address to the public IP address. When a response comes back, NAT translates the public IP address back to the private IP address, allowing the response to reach the correct device.

Table 2-10 Network Address Translation (NAT)

```
R1 (config) # ip nat pool mypool 17.0.0.1 17.0.0.50 netmask 255.255.0.0
R1 (config) # ip nat inside source list 1 pool mypool
R1 (config) # interface g0/0/0
R1 (config-if) # ip nat inside
R1 (config-if) # interface s0/1/0
R1 (config-if) # ip nat outside
R1 (config-if) # crypto isakmp policy 1
R5 (config) # ip nat pool mypool 17.0.0.1 17.0.0.50 netmask 255.255.0.0
R5 (config) # ip nat inside source list 1 pool mypool
R5 (config) # interface g0/0/0
R5 (config-if) # ip nat inside
R5 (config-if) # interface s0/1/0
R5 (config-if) # ip nat outside
R5 (config-if) # crypto isakmp policy 1
```

2.1.8 TFTP SERVER

TFTP (Trivial File Transfer Protocol) is a simple, lightweight protocol used for transferring files across a network. It is often used for transferring configuration files, firmware images, and backups, especially in network environments where simplicity and minimal overhead are desired.

Step-by-Step Setup

1. Install TFTP Server Software

- **Download TFTP Server**: Download a TFTP server application suitable for your operating system.
- **Install and Configure**: Follow the installation instructions and set the directory where files will be stored (this is usually the root directory for file transfers).

2. Configure the TFTP Server

1. **Set the TFTP Root Directory**: This is where the files will be stored or accessed.
2. **Allow Access**: Make sure the firewall settings allow TFTP traffic (typically UDP port 69).

Table 2-11 Trivial File Transfer Protocol (TFTP)

```
Sw1 (config) # copy running-config tftp:

Address or name of remote host []? 192.168.2.60

Destination filename [Switch-confg]? Backup_cfg_for_Sw1
```

2.1.9 Availability & HSRP(Hot Standby Router Protocol)

Network availability and network management are two of the most critical technology areas in network design. High availability assures redundancy for network services.

Some level of high availability solutions is designed for any network module proposed. The Access Layer can have multiple ways of connecting access devices and it can have multiple connections to multiple devices in the Distribution Layer, and the Distribution Layer can have some redundancy when connecting to the Core Layer.

Redundancy can be accomplished in multiple ways, including the following:

- ✓ Proxy ARP on routers
- ✓ Explicit configuration
- ✓ ICMP Router Discovery Protocol (IRDP)
- ✓ Routing Information Protocol (RIP)
- ✓ Hot Standby Router Protocol (HSRP)
- ✓ Virtual Router Redundancy Protocol (VRRP)
- ✓ Gateway Load Balancing Protocol (GLBP)

The preferred solution is a technology that does not place any burden on the hosts and that is completely transparent to them. The hosts need only to configure a single default gateway because the entire redundancy configuration is made on the routers.

The protocols that can be used to accomplish this are generically called First Hop Redundancy Protocols (FHRPs), and they include the following:

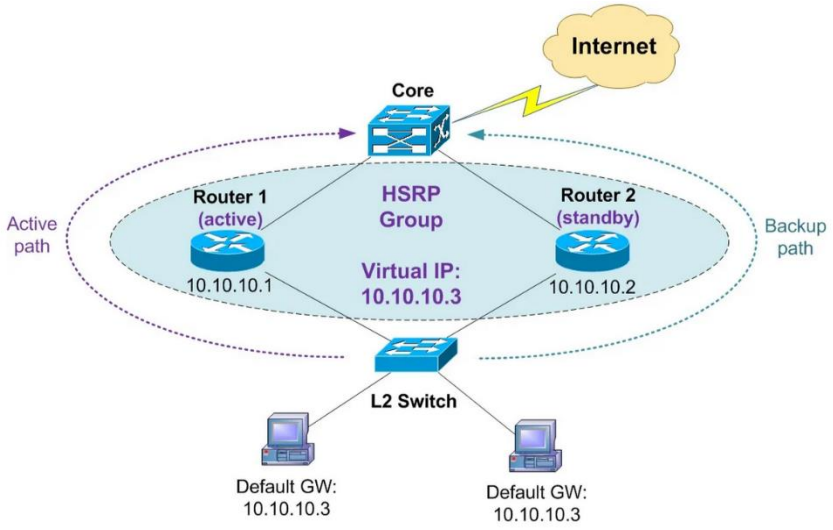
- ✓ HSRP
- ✓ VRRP
- ✓ GLBP

HSRP is a Cisco proprietary protocol that inspired IEEE to create the open standard protocol VRRP. The functionality of the two protocols is almost identical. GLBP, another Cisco invention, is the most recent and the most sophisticated FHRP.

HSRP

Figure below illustrates a network that has two gateway routers connected to one Layer 2 switch and that aggregates the network hosts:

Figure 2-4 Hot Standby Router Protocol



Router 1 and Router 2 are configured in the HSRP group, presenting the clients with a virtual default gateway address of 10.10.10.3 as a virtual address.

Router 1 is the active device, forwarding traffic. Router 2 is the standby HSRP device. The two routers exchange HSRP messages to each other.

These devices are transparently providing access for the clients, as they are transparently serving up the virtual default gateway address.

2.1.10 Ether-Channel

EtherChannel is a technology used to bundle multiple physical Ethernet links into a single logical link, effectively increasing bandwidth and providing redundancy. This is particularly useful in environments where high availability and increased throughput are required, such as in enterprise networks and data centers.

Layer 2 EtherChannel

EtherChannel can be configured at both Layer 2 (L2) and Layer 3 (L3) in networking, each serving different purposes based on the network design and requirements. Here's a breakdown of both configurations:

Layer 2 EtherChannel

is primarily used to aggregate multiple Ethernet links between switches. It allows for increased bandwidth and redundancy at the data link layer.

Table 2-12 EtherChannel

```
interface range fa0/1 – 2

channel-group 1 mode desirable

interface Port-channel1

switchport mode trunk
```

Layer 3 EtherChannel

is used for routing between switches and can also be configured between switches and routers. It allows for the routing of IP traffic at Layer 3, providing IP addressing to the EtherChannel interface.

Table 2-13 Layer 3 EtherChannel

```
interface range fa0/1 – 2

channel-group 1 mode desirable

interface Port-channel1

ip address 192.168.1.1 255.255.255.0

no shutdown
```

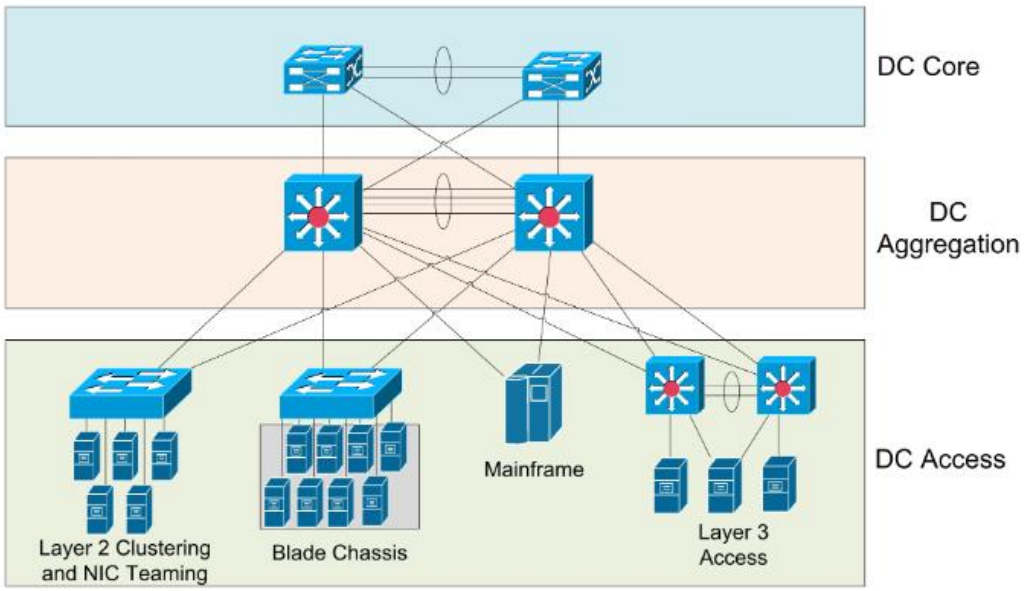


Figure 2-5 Enterprise Data Center Infrastructure

The role of the data center's Aggregation Layer is to aggregate Layer 2 and/or Layer 3 links from the Access Layer and to connect upstream links to the data center's Core Layer. Layer 3 connectivity is usually implemented between the Aggregation and Core Layers.

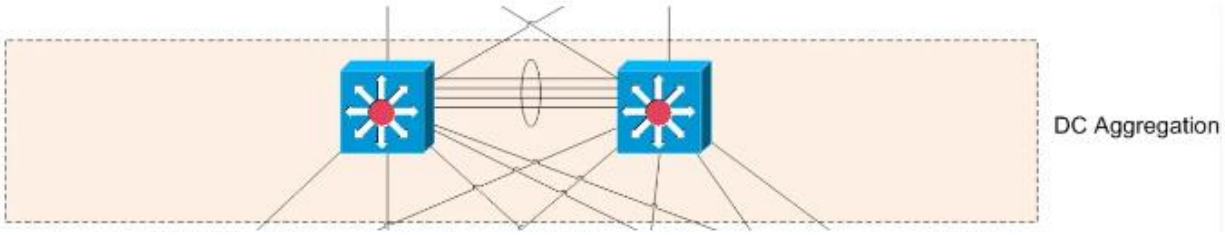


Figure 2-6 Data Center's Aggregation Layer

Media redundancy when connecting to multiple paths in case one of the links failsuse the Spanning Tree Protocol (STP) at Layer 2 in order to avoid loops that can bring the network down. EtherChannel aggregation technique can be used between switches. The bundled links can appear as one single link between specific devices,

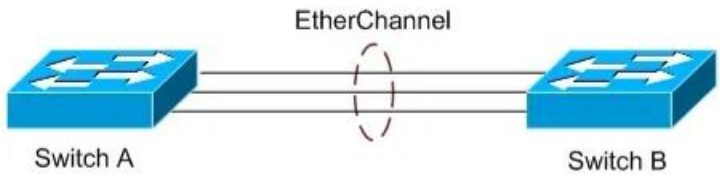


Figure 2-7 EtherChannels

EtherChannels are important when using STP because when all of the links look like one link. Links in an EtherChannel can be load balanced across dynamically.

Cisco switches support the following implementation options for EtherChannels:

- ✓ Port Aggregation Protocol (PagP) – a Cisco proprietary protocol
- ✓ Link Aggregation Control Protocol (LACP) – an open standard protocol

2.1.11 SSH Protocol

Secure Shell enables a user to access a remote device and manage it remotely. However, with SSH, all data transmitted over a network (including usernames and passwords) is encrypted and secure from eavesdropping.

SSH is a client-server protocol, with an SSH client and an SSH server. The client machine (such as a PC) establishes a connection to an SSH server running on a remote device (such as a router). Once the connection has been established, a network admin can execute commands on the remote device.

Table SSH Protocol 14-2 Table

```
R1(config) # username TVTC secret cisco123
R1(config) # enable secret cisco123
R1(config) # line vty 0 4
R1(config-line) # transport input ssh
R1(config-line) # login local
R1(config-line) # exit
R1(config) # Ip ssh authentication-retries 3
R1(config) # ip ssh time-out 60
```

2.2 BEST PRACTICES FOR USING THE HIERARCHICAL NETWORK MODEL

In order to make the most of a hierarchical network model, some best practices and guidelines should be followed. To start, define your network objectives and requirements in relation to the functions and features of each layer. Additionally, select the right devices and protocols for each layer based on the network size, traffic patterns, and budget. Furthermore, ensure that there is consistent and standardized naming conventions, addressing schemes, and documentation for each layer and device. You can also optimize the network performance and security by using VLANs, routing protocols, access lists, and encryption. Lastly, it is important to monitor and troubleshoot the network regularly with tools such as ping, traceroute, or packet capture.

The main advantages of the Cisco hierarchical network model are as follows:

- Ease to understand and implement
- Flexibility
- Cost savings
- Modularity
- Easily modified
- Easy network growth
- Facilitates summarization
- Fault isolation

2.3 EXAMPLES OF USING THE HIERARCHICAL NETWORK MODEL

To illustrate how to use the hierarchical network model for your projects, here are some examples of network scenarios and solutions based on the model. A small business network can be set up with a single switch as the access layer, a router as the distribution layer, and a firewall as the core layer. This switch can create VLANs to separate departments and devices, the router can provide routing and DHCP services, and the firewall can protect from external threats while providing NAT and VPN services. For a medium-sized campus network with 500 users and devices across multiple buildings and floors, multiple switches and wireless access points

should be used as the access layer, multiple multilayer switches as the distribution layer, and a pair of routers as the core layer. The access layer devices can provide physical and logical connectivity and security, the distribution layer devices can aggregate and filter traffic while implementing VLANs and QoS policies, and core layer devices can provide high-speed data transmission with redundancy. For larger enterprise networks with 5000 users across multiple locations, multiple switches and wireless access points should be used for access control, multiple routers and firewalls for segmentation and security, while backbone routers and switches should be used for interconnection among different sites.

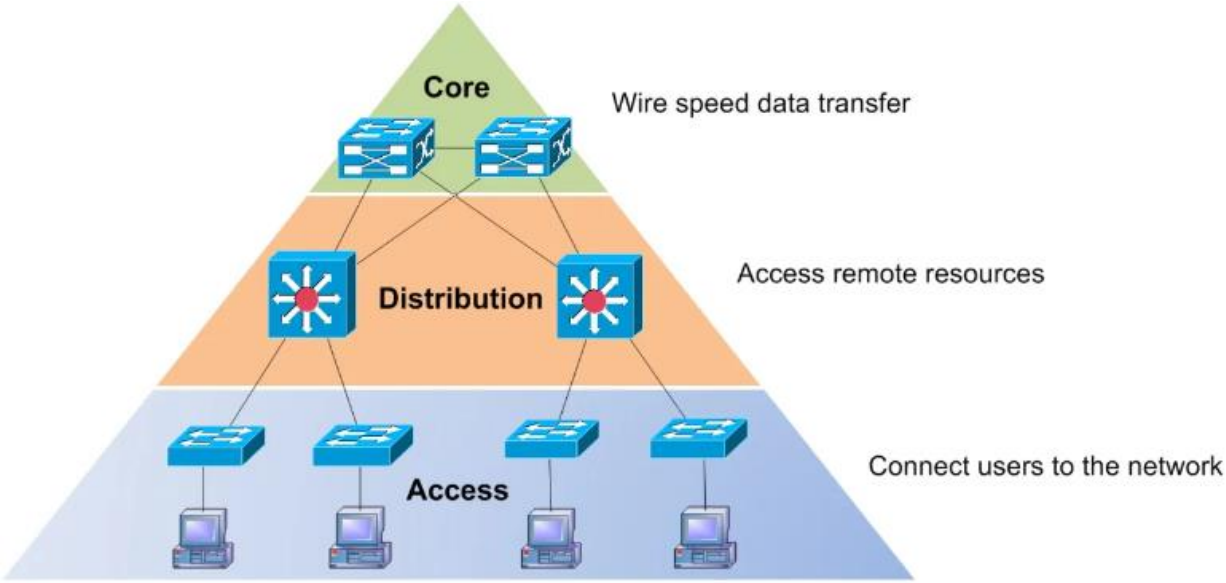


Figure 2-8 Cisco Hierarchical Network Model Layers

The Core Layer

The Core Layer of the network refers to the following aspects:

- High speed
- Reliability and availability
- Redundancy
- Fault tolerance
- Load balancing
- Manageability and scalability
- No filters, packet handling, or other overhead
- Limited, consistent diameter
- Quality of Service

The Distribution Layer

The Distribution Layer has some of the same features as the Core Layer does, such as QoS techniques, redundancy, and load balancing features. Except for these aspects, the Distribution Layer performs the following unique and completely different set of functions:

- Access control to core devices
- Redundancy to access devices

- Routing protocol boundaries
- Redistribution
- Filtering
- Route summarization
- Policy routing
- Security
- Separate multicast and broadcast domains
- Routing between VLANs
- Media translation and boundaries (e.g., FastEthernet to GigabitEthernet)
- Redistribution

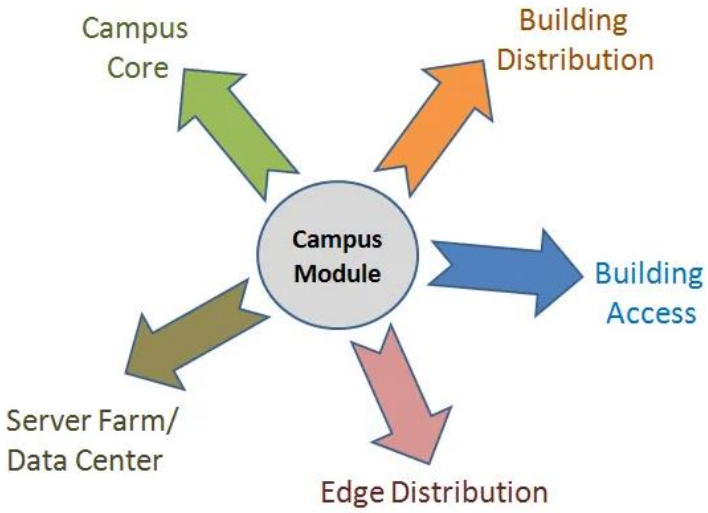


Figure 2-9 Enterprise Campus Module Components

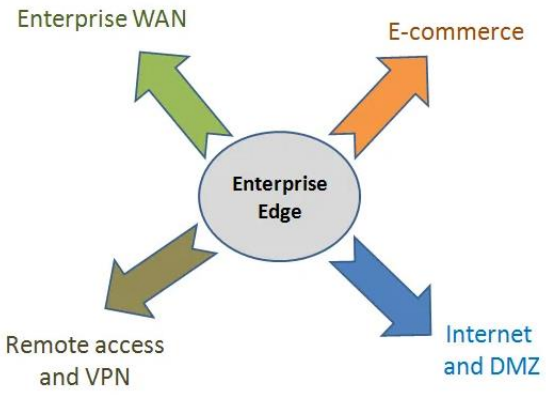


Figure 2-10 Enterprise Edge Components

2.4 **NETWORK TOPOLOGY HQ AND BRANCHES.**

The designed secure campus network topology consists of two main parts or buildings. Each part contains different devices as switches, computers, phones, printers, access point, wireless router, and servers. All of those devices are connected with a switch that connects them directly with a router. The routers in the campus are connected with each other dynamically as it's shown in Figure 1.

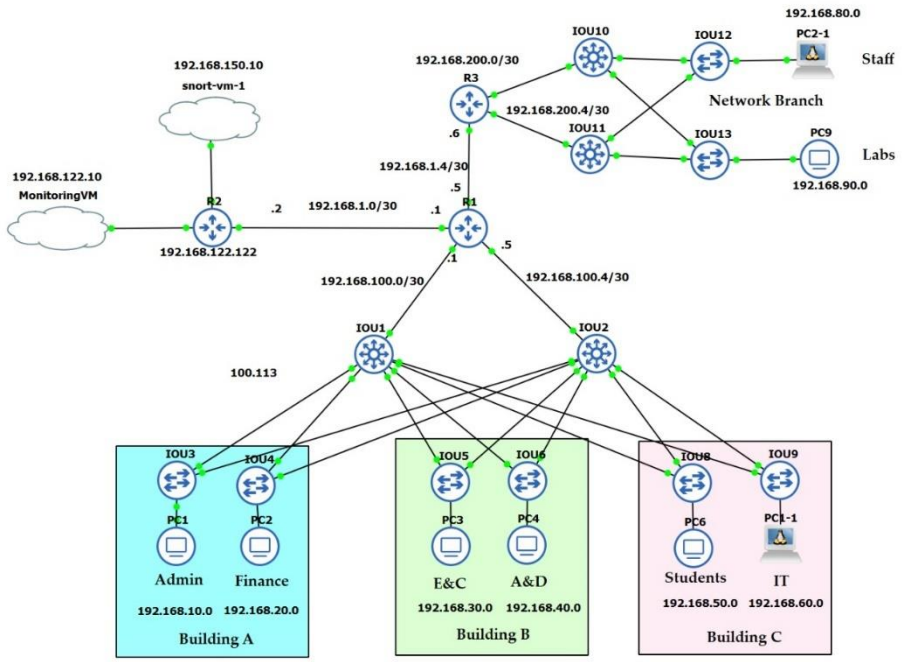


Figure 2-11 Network Topology

In large-scale networking with branch chains or branch offices, branch networks need to access the Internet, HQ, DCs, or other branches. In addition, branch networks may require private lines or VPNs for connecting to the HQ network. In this scenario where interconnection between branches and between branches and the HQ/DC is required.

On a traditional campus network, branch networks connect to the HQ network through private lines, incurring high costs and low deployment efficiency. The coverage scope and network performance of the rapidly developing Internet improve

continuously. This makes more and more enterprise branches access the HQ network by using the low-cost Internet. To ensure network security and reliability, IPsec VPN needs to be deployed to enhance the security and efficiency of network transmission and increase reliability and flexibility of branch interconnection.

IPsec VPN for interconnection between multiple campuses or branches. Each campus or branch is called a site in the multi-campus/branch interconnection networking scenario. The internal networking solution of a site is the same as that of a single campus network.

2.5 HIERARCHICAL NETWORK MODEL

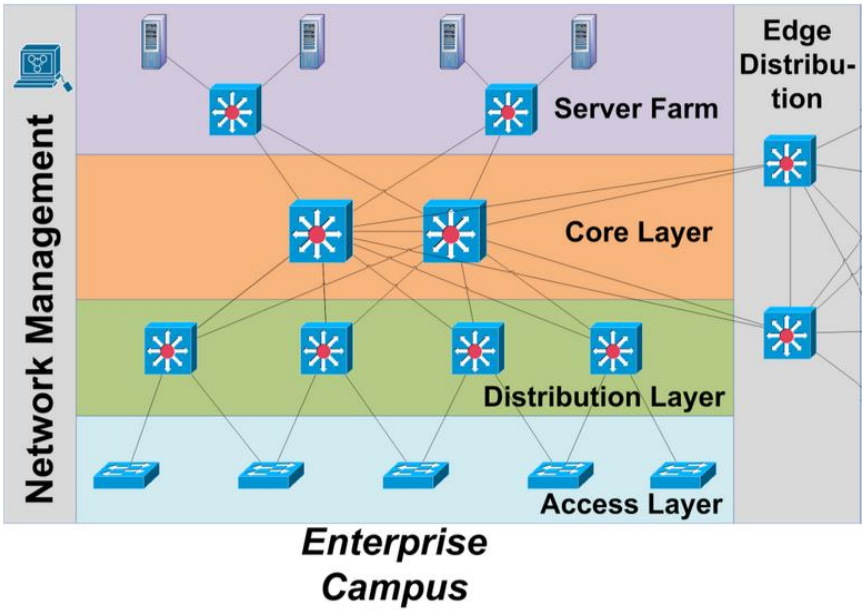


Figure 2-12 Hierarichal topology

The importance of different features per network components, when considering the switching infrastructure, is summarized below:

Table 2-15 HIERARCHICAL NETWORK MODEL

Components	Scalability	Availability	Performance	Cost
Access Layer	HIGH	LOW	LOW	LOW
Distribution Layer	HIGH	MEDIUM	MEDIUM	MEDIUM
Backbone	LOW	HIGH	HIGH	HIGH
Server Farm	MEDIUM	HIGH	HIGH	HIGH
Edge Distribution	LOW	HIGH	HIGH	HIGH

Core layer
Distribution layer
Access layer

Core Layer

The core layer is the backbone of the network that connects different distribution layers and provides high-speed, reliable, and redundant data transmission. The core layer devices, such as routers and switches, are designed to handle large amounts of traffic and route packets efficiently. The core layer should avoid complex configurations, such as access lists, filtering, or encryption that can slow down the data flow. The core layer should also implement fault tolerance and load balancing mechanisms to ensure network availability and resilience.

Distribution Layer

The distribution layer is the intermediary between the core and access layers that aggregates and filters the traffic from the lower layer devices. The distribution layer devices, such as multilayer switches and routers, are responsible for implementing network policies, such as security, quality of service, routing, and VLANs. The distribution layer should also isolate and limit the scope of broadcast domains and failures to improve network performance and reliability. The distribution layer should also provide redundancy and scalability options for the network growth and expansion.

Access Layer

The access layer is the edge of the network that connects the end users and devices, such as PCs, printers, servers, and cameras, to the network. The access layer devices, such as switches and wireless access points, are responsible for providing physical and logical connectivity, as well as enforcing access control and security measures. The access layer should also support different types of media, such as Ethernet, Wi-Fi, or fiber, and different speeds and standards, such as 10/100/1000 Mbps or PoE. The access layer should also facilitate network management and monitoring by using features such as SNMP, DHCP, or Spanning Tree Protocol.

2.6 **NETWORK PROBLEMS AND SOLUTIONS**

Table 2-16 NETWORK PROBLEMS AND SOLUTIONS

Security problems	proposed solution
Attackers	port security
	Add servers DMZ Zone in ASA
	add username & password in all devices
	limit login trials in all devices
	user IPSEC VPN tunnel between HQ and branch sites
	NAT and policy on ASA
network failure	use L2 & l3 ether channel
	use OSPF for dynamic routing & enable availability
	DHCP enabled on Dist switch
	use multiple departments in HQ each has dist switch
	load sharing using HSRP on Dist switches

Summary

For many years, Cisco recommended a three-layer hierarchical network design model, which included the Access Layer, the Distribution Layer, and the Core

Layer. However, to provide for enhanced scalability and flexibility, Cisco later introduced the Cisco enterprise architecture model, which categorizes enterprise networks into six modules. The three layers can be found in each of these modules. Specifically, each module can contain its own network infrastructure, services, and applications. This section summarizes the design considerations surrounding the modules that comprise the Cisco enterprise architecture model.

Traditionally, Cisco prescribed a three-layer model for network designers, as follows:

- **Access Layer:** Typically, wiring closet switches connected to end-user stations.
- **Distribution Layer:** An aggregation point for wiring closet switches, where routing and packet manipulation occur.
- **Core Layer:** The network backbone, where high-speed traffic transport is the main priority.

When designing the enterprise campus module, four primary areas must be addressed, as follows:

- **Access Layer block:** Connects end-user devices to the network.
- **Distribution Layer block:** Aggregates building access switches and performs Layer 3 switching (i.e., routing) functions.

- **Core Layer block:** Provides high-speed, redundant connectivity between buildings.
- **Server farm/data center block:** Consolidates application servers, e-mail servers, domain name servers, file servers, and network management applications.

The enterprise edge module connects the enterprise campus module with the WAN and Internet blocks. The four building blocks are as follows:

- Servers including the following:
  - Web servers
  - Application servers
  - Database servers
  - Security servers
  - (DNS) servers
  - Public web servers
  - Security servers

Enterprise locations are supported via the following modules:

- Enterprise branch
- Enterprise data center
- Enterprise teleworker

Layered on top of an enterprise's network infrastructure are infrastructure services, which enable business applications. Examples of these infrastructure services include the following:

- Network management
- Network security
- Quality of Service
- Network availability, including:
    - Workstation-to-router redundancy
    - Server redundancy
    - Route redundancy
    - Media redundancy

2.7 NETWORK SECURITY

There are a lot of techniques to protect the network from interior and exterior attackers. Attacking could be physical by sabotaging, ruining, or stealing the devices; or it can be by hacking the system and accessing without authorization. Thus, in order to protect the network from those types of attacks, we need a strong security system. Network security is a set of policies and procedures that monitor the entire network continuously to secure and prevent it from unauthorized access. Consequently, in SCN we used a high level of security in the main devices like routers and switches. We secured all the ports so those devices would not accept any access without authentication.

In the routers' case, we secured VTY lines and console lines by adding passwords to require authentication from the user; as it is clear in the following router configuration. In the switch's case, we also secured all ports; also, we disabled some protocols that show the information of the devices to others such as cisco discovery protocol (CDP).

- We start the security process by putting a password for the line console in order to prevent remote access by others.
- We put a password for line VTY 0 4 to restrict the telnet and SSH unauthorized access.
- We used the message-digest algorithm MD5 to encrypt passwords.
- Finally, we put some restricted features in creating new passwords such as minimum length, and the number of attempts. For example, we verified restriction by entering new user with weak password and the router rejected.

2.7.1 Router Security Configuration Commands

Router security is crucial for protecting network devices from unauthorized access and attacks. Configuring security features on routers involves a combination of access control, authentication, and monitoring practices. Below are key security configuration commands for Cisco routers.

17-2 Table `hostname`

```
Router(config)# hostname R1
Router(config)# hostname R2
Router(config)# hostname R3
```

18-2 Table security passwords

```
R1(config)# security passwords min-length 8

R2(config)# security passwords min-length 8

R3(config)# security passwords min-length 8
```

Table 2-19 Router Security Configuration

```
R1(config)# service password-encryption
R2(config)# service password-encryption
R3(config)# service password-encryption
## configure enable password
R1(config)# enable secret cisco123
R2(config)# enable secret cisco123
R3(config)# enable secret cisco123
## Restrict router login attempts to 3 times for wait time 70 sec to retry
R1(config)# login block-for 70 attempts 3 within 30
R2(config)# login block-for 70 attempts 3 within 30
R3(config)# login block-for 70 attempts 3 within 30
## Create username & password for router login
R1(config)# username HQ password cisco123
R2(config)# username HQ password cisco123
R3(config)# username HQ password cisco123
```

```
## Create Banner message

R1(config)# banner motd $WORNING. ACCESS IS ALLOWED FOR AUTHRIZD PEOPLE$
R2(config)# banner motd $WORNING. ACCESS IS ALLOWED FOR AUTHRIZD PEOPLE$
R3(config)# banner motd $WORNING. ACCESS IS ALLOWED FOR AUTHRIZD PEOPLE$

##Console Configuration


R1(config)# line console 0

R1(config-line) # PASSWORD cisco123

R1(config-line) # EXEC-timeout 3

R1(config-line) # LOGGING SYnchronous

R1(config-line) # login

R1(config-line) # end

##VTY Configuration

R1(config)# line VTY 0 4

R1(config-line) # PASSWORD cisco123

R1(config-line) # EXEC-timeout 3

R1(config-line) # LOGGING SYnchronous

R1(config-line) # login LOCAL

R1(config-line) # end

Apply on all Other Routers

switch

Sw1(config)# enable secret cisco123

Sw2(config)# enable secret cisco123

Sw3(config)# enable secret cisco123

Sw4(config)# enable secret cisco123

Sw5(config)# enable secret cisco123

SW6(config)# enable secret cisco123
```

Secret password

To configure a secret password on a Cisco router, follow these steps. The enable secret password provides an added layer of security by encrypting the password, making it difficult to view in plain text.

20-2 Table Secret password

```
R1(config)# enable secret cisco123

R2(config)# enable secret cisco123

R3(config)# enable secret cisco123

switch

E1(config)# enable secret cisco123

E2(config)# enable secret cisco123

E3(config)# enable secret cisco123

E4(config)# enable secret cisco123

E5(config)# enable secret cisco123

E7(config)# enable secret cisco123
```

PORT SECURITY

Persistent (Sticky) MAC is a Layer 2 port security feature that prevents unauthorized devices from connecting to your network. When this feature is enabled, the switch will observe the incoming source MAC addresses on a configured port and dynamically learn/save this address to memory.

To enable sticky learning, enter the switchport port-security mac-address sticky command. When you enter this command, the interface converts all the dynamic secure MAC addresses, including those that were dynamically learned before sticky learning was enabled, to sticky secure MAC addresses.

Dynamic secure MAC addresses – are dynamically learned by the switch and stored in its MAC address table. They are removed from the configuration when the switch restarts. Sticky secure MAC addresses – like Dynamic secure MAC addresses, MACs are learned dynamically but are saved in the running configuration.

Switch

Table 2-21 PORT SECURITY

```
interface FastEthernet0/6
 switchport access vlan 101
 switchport mode access
 switchport port-security
 switchport port-security mac-address sticky
 switchport port-security mac-address sticky 00D0.BAA2.A86
interface FastEthernet0/7
```

```
switchport access vlan 101
switchport mode access
switchport port-security
switchport port-security mac-address sticky
switchport port-security mac-address sticky 0009.7C30.6262
```

2.7.2 Snort

Snort is a highly respected open-source intrusion detection and prevention system (IDS/IPS). It can perform real-time traffic analysis and packet logging on IP networks, detecting a wide variety of attacks and probes.
In our project:

- Snort was installed on one of the Kali Linux machines.
- It was configured with custom rules to detect network threats such as port scanning, brute-force login attempts, and denial-of-service (DoS) attacks.
- Alerts generated by Snort were passed to the Alert Manager for notification.

We tested Snort by simulating attacks from another machine, and the system successfully detected and logged suspicious activity.

2.7.3 Types of Attacks Simulated in the Project

In order to evaluate the effectiveness of our intrusion detection and alerting mechanisms, several common cyberattack techniques were simulated within the network environment. These attacks were executed using Kali Linux tools and observed through Snort and Alertmanager to assess the system's detection capabilities.

1. Port Scanning Attacks

Examples: Nmap Scan, TCP SYN Scan, UDP Scan

Objective: These reconnaissance techniques aim to identify open ports and active services on target devices. Port scanning is often the first step in a broader attack strategy, enabling attackers to map the network and find vulnerable entry points.

```
┌──(kali㉿kali)-[~/Desktop/ansible]
└─$ nmap -sS -T4 -p 1-1000 192.168.150.10

Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-06-26 11:17 EDT
```

2. Brute Force Attacks

```
┌──(kali㉿kali)-[~]
└─$ tail -f /var/log/snort/alert_fast.txt
06/26-18:00:01.101010 [**] [1:1000001:1] "Port Scan Detected" [**] [Priority: 0]
{TCP} 192.168.122.10:55821 → 192.168.150.10:22
06/26-18:00:01.101010 [**] [1:1000001:1] "Port Scan Detected" [**] [Priority: 0]
{TCP} 192.168.122.10:55821 → 192.168.150.10:22
06/26-18:00:01.101010 [**] [1:1000001:1] "Port Scan Detected" [**] [Priority: 0]
{TCP} 192.168.122.10:55821 → 192.168.150.10:22
06/26-18:00:01.101010 [**] [1:1000001:1] "Port Scan Detected" [**] [Priority: 0]
{TCP} 192.168.122.10:55821 → 192.168.150.10:22
06/26-18:00:01.101010 [**] [1:1000001:1] "Port Scan Detected" [**] [Priority: 0]
{TCP} 192.168.122.10:55821 → 192.168.150.10:22
06/26-18:00:01.101010 [**] [1:1000001:1] "Port Scan Detected" [**] [Priority: 0]
{TCP} 192.168.122.10:55821 → 192.168.150.10:22
06/26-18:00:01.101010 [**] [1:1000001:1] "Port Scan Detected" [**] [Priority: 0]
{TCP} 192.168.122.10:55821 → 192.168.150.10:22
06/26-18:00:01.101010 [**] [1:1000001:1] "Port Scan Detected" [**] [Priority: 0]
{TCP} 192.168.122.10:55821 → 192.168.150.10:22
06/26-18:00:01.101010 [**] [1:1000001:1] "Port Scan Detected" [**] [Priority: 0]
{TCP} 192.168.122.10:55821 → 192.168.150.10:22
```

Figure 2-13 port scan

Examples: SSH password guessing, Telnet login attempts

Objective: Brute force attacks attempt to gain unauthorized access to network devices by systematically trying a large number of username and password combinations. These attacks are commonly directed at remote management protocols like SSH and Telnet.

3. Denial of Service (DoS) Attacks

Examples: Ping of Death, TCP SYN Flood, ICMP Flood

Objective: The goal of DoS attacks is to overwhelm network devices or services with excessive traffic, rendering them unresponsive or unavailable to legitimate users.

```
┌──(kali㉿kali)-[~]
└─$ tail -f /var/log/snort/alert_fast.txt
06/26-18:02:44.303030 [**] [1:1000003:1] "Possible DoS attack" [**] [Priority: 1] {TCP} 192.168.122.10:45555 → 192.168.150.10:80
06/26-18:02:44.303030 [**] [1:1000003:1] "Possible DoS attack" [**] [Priority: 1] {TCP} 192.168.122.10:45555 → 192.168.150.10:80
06/26-18:02:44.303030 [**] [1:1000003:1] "Possible DoS attack" [**] [Priority: 1] {TCP} 192.168.122.10:45555 → 192.168.150.10:80
06/26-18:02:44.303030 [**] [1:1000003:1] "Possible DoS attack" [**] [Priority: 1] {TCP} 192.168.122.10:45555 → 192.168.150.10:80
06/26-18:02:44.303030 [**] [1:1000003:1] "Possible DoS attack" [**] [Priority: 1] {TCP} 192.168.122.10:45555 → 192.168.150.10:80
06/26-18:02:44.303030 [**] [1:1000003:1] "Possible DoS attack" [**] [Priority: 1] {TCP} 192.168.122.10:45555 → 192.168.150.10:80
06/26-18:02:44.303030 [**] [1:1000003:1] "Possible DoS attack" [**] [Priority: 1] {TCP} 192.168.122.10:45555 → 192.168.150.10:80
06/26-18:02:44.303030 [**] [1:1000003:1] "Possible DoS attack" [**] [Priority: 1] {TCP} 192.168.122.10:45555 → 192.168.150.10:80
06/26-18:02:44.303030 [**] [1:1000003:1] "Possible DoS attack" [**] [Priority: 1] {TCP} 192.168.122.10:45555 → 192.168.150.10:80
```

Figure 2-14 Dos attack

.

4. DNS Spoofing / DNS Poisoning

Objective: This attack involves corrupting the DNS cache to redirect users to fraudulent or malicious websites. It is commonly used for phishing and man-in-the-middle attacks.

```
┌──(kali㉿kali)-[~/Desktop/ansible]
└─$ sudo dnsspoof -i eth1

dnsspoof: listening on eth1 [udp dst port 53 and not src 192.168.122.10]
^C
```

.

```
┌──(kali㉿kali)-[~]
└─$ tail -f /var/log/snort/alert_fast.txt
06/26-18:03:55.404040 [**] [1:1000004:1] "DNS Spoofing Attempt" [**] [Priority: 2] {UDP} 192.168.122.10:53 → 192.168.150.10:51432
06/26-18:03:55.404040 [**] [1:1000004:1] "DNS Spoofing Attempt" [**] [Priority: 2] {UDP} 192.168.122.10:53 → 192.168.150.10:51432
06/26-18:03:55.404040 [**] [1:1000004:1] "DNS Spoofing Attempt" [**] [Priority: 2] {UDP} 192.168.122.10:53 → 192.168.150.10:51432
06/26-18:03:55.404040 [**] [1:1000004:1] "DNS Spoofing Attempt" [**] [Priority: 2] {UDP} 192.168.122.10:53 → 192.168.150.10:51432
06/26-18:03:55.404040 [**] [1:1000004:1] "DNS Spoofing Attempt" [**] [Priority: 2] {UDP} 192.168.122.10:53 → 192.168.150.10:51432
06/26-18:03:55.404040 [**] [1:1000004:1] "DNS Spoofing Attempt" [**] [Priority: 2] {UDP} 192.168.122.10:53 → 192.168.150.10:51432
06/26-18:03:55.404040 [**] [1:1000004:1] "DNS Spoofing Attempt" [**] [Priority: 2] {UDP} 192.168.122.10:53 → 192.168.150.10:51432
06/26-18:03:55.404040 [**] [1:1000004:1] "DNS Spoofing Attempt" [**] [Priority: 2] {UDP} 192.168.122.10:53 → 192.168.150.10:51432
06/26-18:03:55.404040 [**] [1:1000004:1] "DNS Spoofing Attempt" [**] [Priority: 2] {UDP} 192.168.122.10:53 → 192.168.150.10:51432
```

Figure 2-15 DNS

5. ARP Spoofing / ARP Poisoning

Objective: By sending fake ARP messages, an attacker can associate their MAC address with the IP address of another host, effectively intercepting, modifying, or blocking network traffic. This technique is often used to launch Man-in-the-Middle (MITM) attacks.

6. Malicious Payloads / Malware Traffic Detection

Objective: The network was also monitored for traffic containing potentially harmful scripts, executables, or payloads that may indicate malware infection or lateral movement within the system.

7. Ping Sweep (ICMP Network Discovery)

Objective: This scanning technique uses ICMP Echo Requests to identify live hosts within a subnet. Although less intrusive than full scans, ping sweeps are often a precursor to more targeted attacks. .

```
┌──(kali㉿kali)-[~/Desktop/ansible]
└─$ nmap -sn 192.168.150.0/24
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-06-26 11:31 EDT
```

.

```
┌──(kali㉿kali)-[~]
└─$ tail -f /var/log/snort/alert_fast.txt
06/26-18:06:33.707070 [**] [1:1000007:1] "ICMP Ping Sweep Detected" [**] [Priority: 2] {ICMP} 192.168.122.10 → 192.168.150.10
06/26-18:06:33.707070 [**] [1:1000007:1] "ICMP Ping Sweep Detected" [**] [Priority: 2] {ICMP} 192.168.122.10 → 192.168.150.10
06/26-18:06:33.707070 [**] [1:1000007:1] "ICMP Ping Sweep Detected" [**] [Priority: 2] {ICMP} 192.168.122.10 → 192.168.150.10
06/26-18:06:33.707070 [**] [1:1000007:1] "ICMP Ping Sweep Detected" [**] [Priority: 2] {ICMP} 192.168.122.10 → 192.168.150.10
06/26-18:06:33.707070 [**] [1:1000007:1] "ICMP Ping Sweep Detected" [**] [Priority: 2] {ICMP} 192.168.122.10 → 192.168.150.10
06/26-18:06:33.707070 [**] [1:1000007:1] "ICMP Ping Sweep Detected" [**] [Priority: 2] {ICMP} 192.168.122.10 → 192.168.150.10
06/26-18:06:33.707070 [**] [1:1000007:1] "ICMP Ping Sweep Detected" [**] [Priority: 2] {ICMP} 192.168.122.10 → 192.168.150.10
06/26-18:06:33.707070 [**] [1:1000007:1] "ICMP Ping Sweep Detected" [**] [Priority: 2] {ICMP} 192.168.122.10 → 192.168.150.10
06/26-18:06:33.707070 [**] [1:1000007:1] "ICMP Ping Sweep Detected" [**] [Priority: 2] {ICMP} 192.168.122.10 → 192.168.150.10
```

Figure 2-16 Ping Sweep (ICMP)

2.8 SIMULATION TOOLS

2.8.1 GNS3 (Graphical Network Simulator 3)

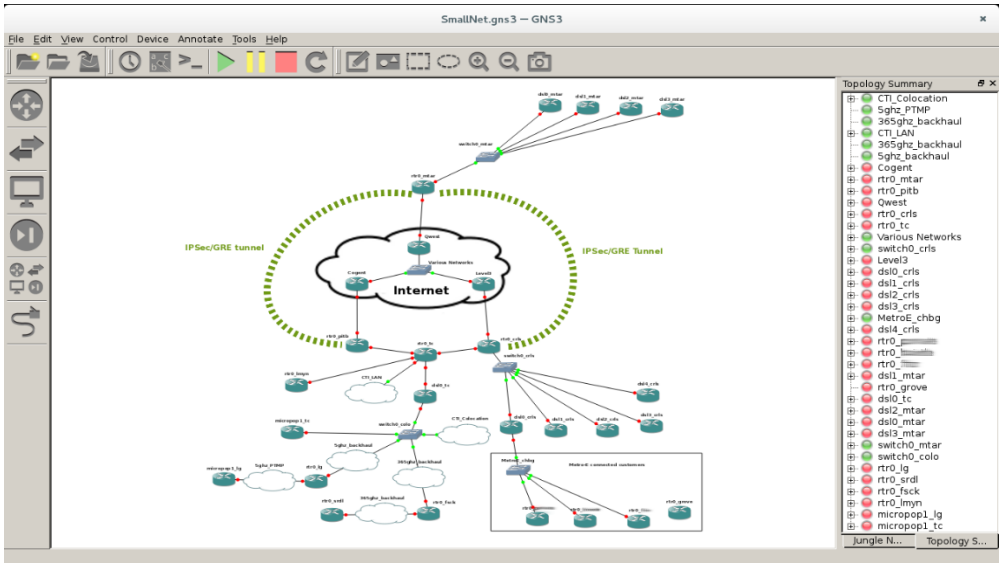
GNS3 is a powerful open-source network emulation platform that allows users to design, test, and configure virtual networks without the need for physical hardware. It provides real-time simulation and supports integration with real operating systems and devices.
In our project, GNS3 was used as the primary platform to simulate the network topology, including routers, switches, and connections between different network segments such as Admin, Finance, IT, and Students. GNS3 allowed us to visualize the architecture, test configurations, and troubleshoot any issues efficiently before implementing the security and automation systems

GNS3 key features

- **Device Emulation**: You can run real Cisco IOS images and other operating systems, allowing for realistic simulations.
- **User-Friendly Interface**: The graphical interface makes it easy to drag and drop devices and connect them.
- **Integration with Virtual Machines**: You can integrate GNS3 with software like VirtualBox or VMware to run other operating systems or applications alongside your network devices.
- **Support for Multiple Vendors**: While it started primarily with Cisco devices, GNS3 supports a range of vendors and devices, making it versatile.
- **Community and Resources**: There's a strong community around GNS3, providing tutorials, documentation, and forums for support.

Figure 2-17 router

GUI Layout



18-2 Figure GNS3 GUI layout

2.8.2 VM ware workstation

Figure 2-19 VM ware workstation

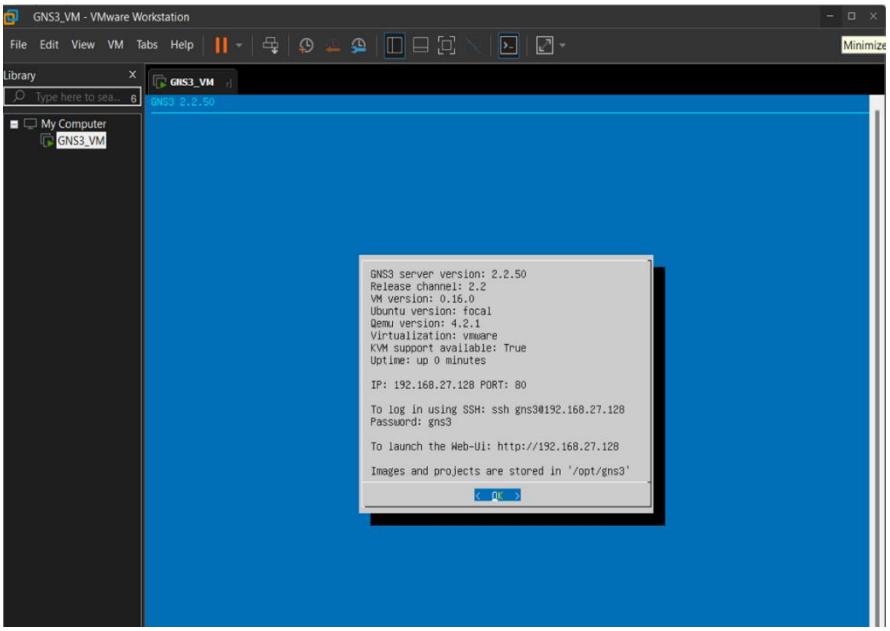
VMware Workstation enabled us to run multiple virtual machines (VMs) on a single physical host system. This was particularly important because it allowed us to simulate different servers and client systems in our network.
Two virtual machines running Kali Linux were created and connected to the GNS3 topology:

Monitoring Server: Hosted Prometheus and Grafana to monitor network performance.

Automation/Security Server: Used to run Ansible for automation and Snort for intrusion detection.

Using VMware gave us the flexibility to run full-fledged Linux environments where we could install necessary software, access real file systems, and communicate directly with network nodes via virtual interfaces.

2.8.3 Required resources (GNS3 Image settings)



Each layer in OSI layers of the hierarchical network model will involve Access, Distribution, and Core Layers. The Core Layer will use mostly transport and network services, but there is some intersection between the Core and Distribution Layers in this regard. The Distribution Layer also grips transport and network services, but with more prominence on the Network Layer (Layer 3).

In order to design a secure campus network (SCN), we used different devices wired and wireless. Also, we used different types of communication media to connect the devices. After connecting the devices, we implemented many important configurations as VLANs, dynamic host configuration protocol (DHCP), and OSPF. Moreover, we applied security and management techniques in the main devices of the network; to make the campus network safer and to protect it from interior and exterior attackers. So, the sanctity and the privacy of the user will be granted.

We used different types of devices in our work to show different connectivity cases. Most of the devices are connected using cables like PCs. However, some of them connected by wireless such as cameras. The descriptions of the devices are:

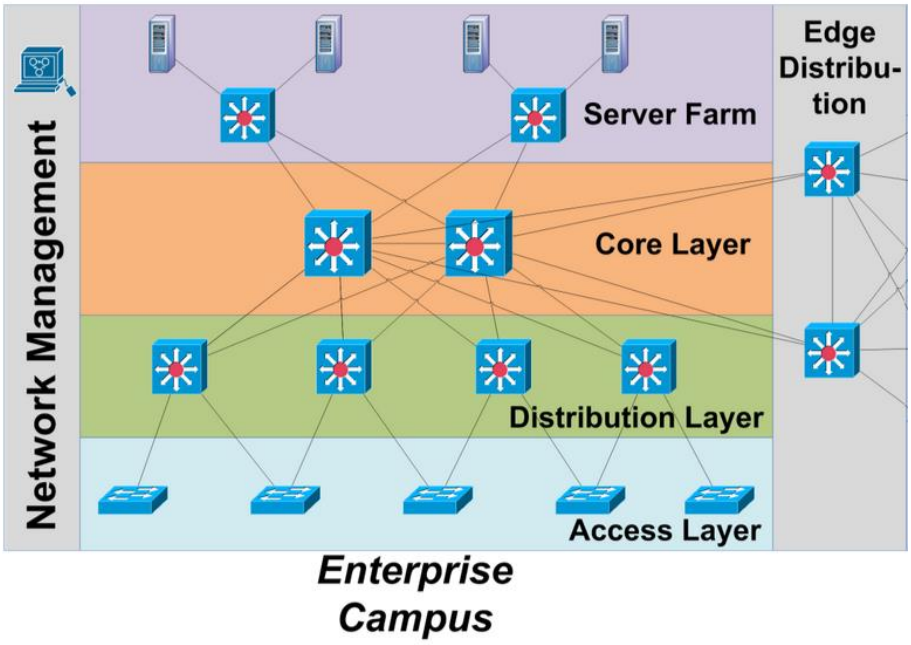


Figure 2-20 Hierarchical Design

- Routers
- Core Switches
- Distribution Switches
- Access Switches
- PCs
- Ethernet cables as shown in the topology

Router R2 Act for branch site 2.

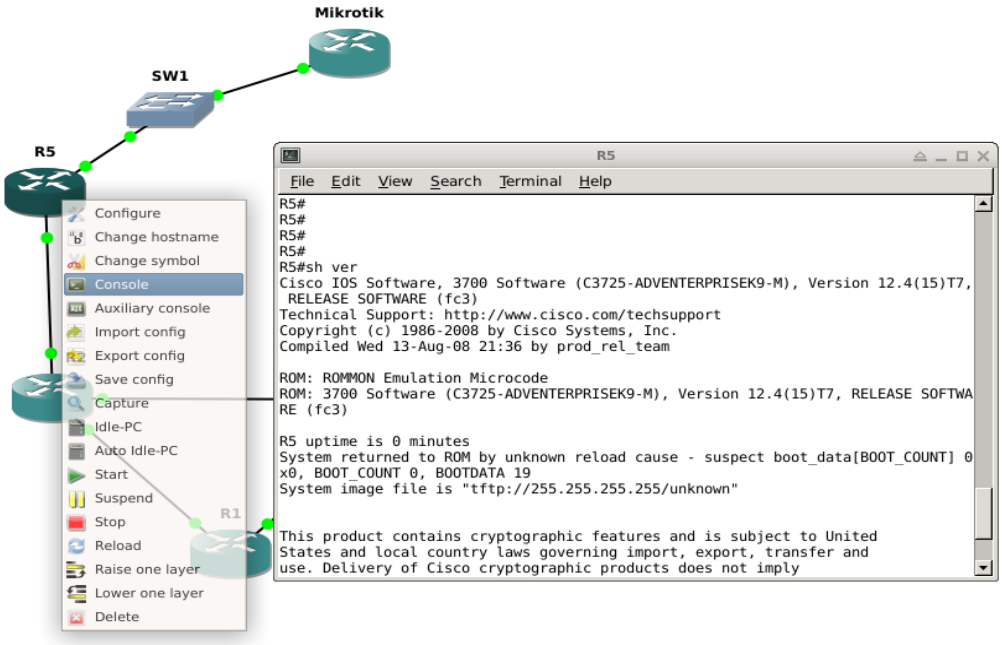
Router R3 Act for branch site 1.

Router ISP Act for HQ.

Multilayer Switches for each branch site.

Access Switches for each branch site.

Router Console Access



21-2 Router Console Access

Accessing the console of a router in GNS3 is essential for configuration and management.

Start the Router:

Right-click on the router icon in the workspace and select "Start". You can also start it by clicking the green play button at the top of the GNS3 window.

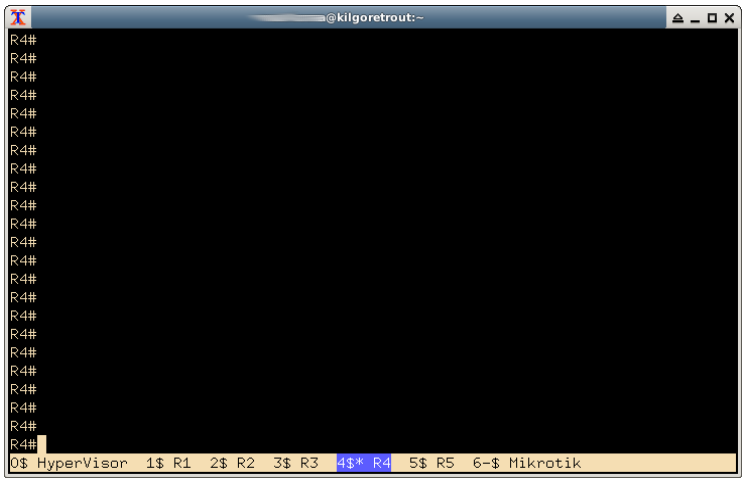
Access the Console:

Right-click on the running router and select "Console". This will open a console window for the router, allowing you to interact with it.

Alternatively, you can select the router and click on the "Console" button in the toolbar.

3. **Using the Console**:

In the console window, you'll be prompted with a command-line interface (CLI) where you can enter commands to configure the router.



CLI access 22-2 Figure

Access switch

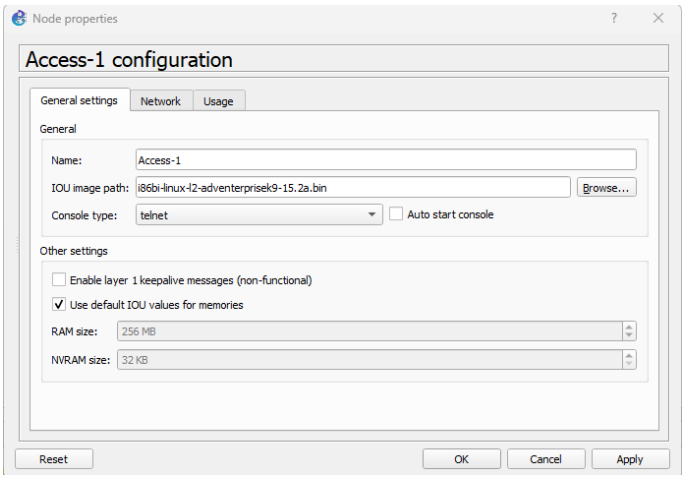


Figure 2-23 : access switch gns3

In a campus design, the Access Layer is characterized by virtual LANs (VLANs) to the PCs and servers.

The Access Layer should provide security features, such as port security, and VLAN Access Control Lists (VACLs). These security features.

Other functions of the Access Layer are implementing authentication (using a TACACS+ or RADIUS server), broadcast control, and defining QoS trust boundaries (i.e., which devices are trusted in the QoS settings, or how far from the actual desktop stations classification and marking takes place). In addition, the Access Layer implements rate limiting Denial of Services (DoS) attacks on the network.

Distribution switch

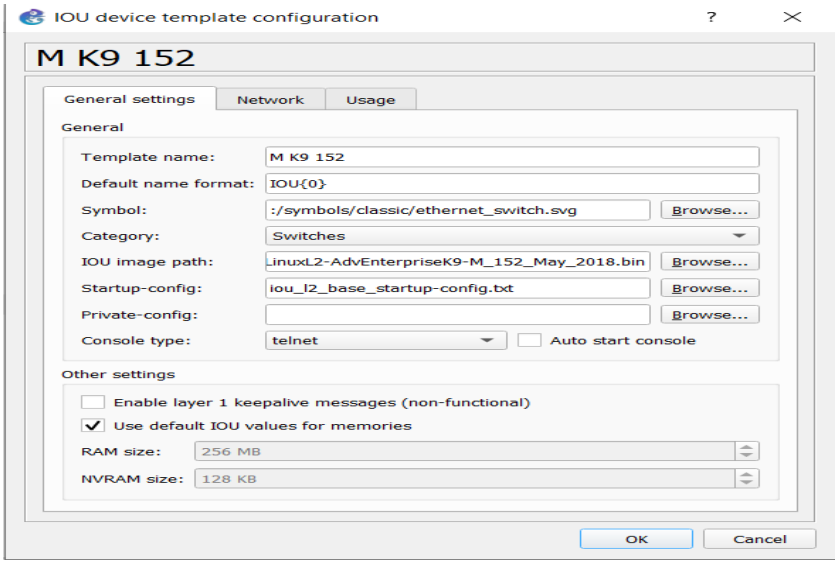


Figure 2-24 Distribution switch gns3

Unlike the Core Layer, the Distribution Layer usually includes features where policies control who can access what resources from outside of the network. This is also where many security features are implemented due to the growing number of attacks from the Internet.

The Distribution Layer has some of the same features as the Core Layer does, such as QoS techniques, redundancy, and load balancing features. Except for these aspects, the Distribution Layer performs the following unique and completely different set of functions:

- Access control to core devices
- Redundancy to access devices
- Routing protocol boundaries
- Redistribution
- Filtering
- Route summarization
- Policy routing
- Security
- Separate multicast and broadcast domains
- Routing between VLANs
- Media translation and boundaries (e.g., FastEthernet to GigabitEthernet)
- Redistribution

The value of this separation of functions is that it is easier to understand, troubleshoot, and manage the network when different jobs are occurring in these hierarchical network layers.

Core switch

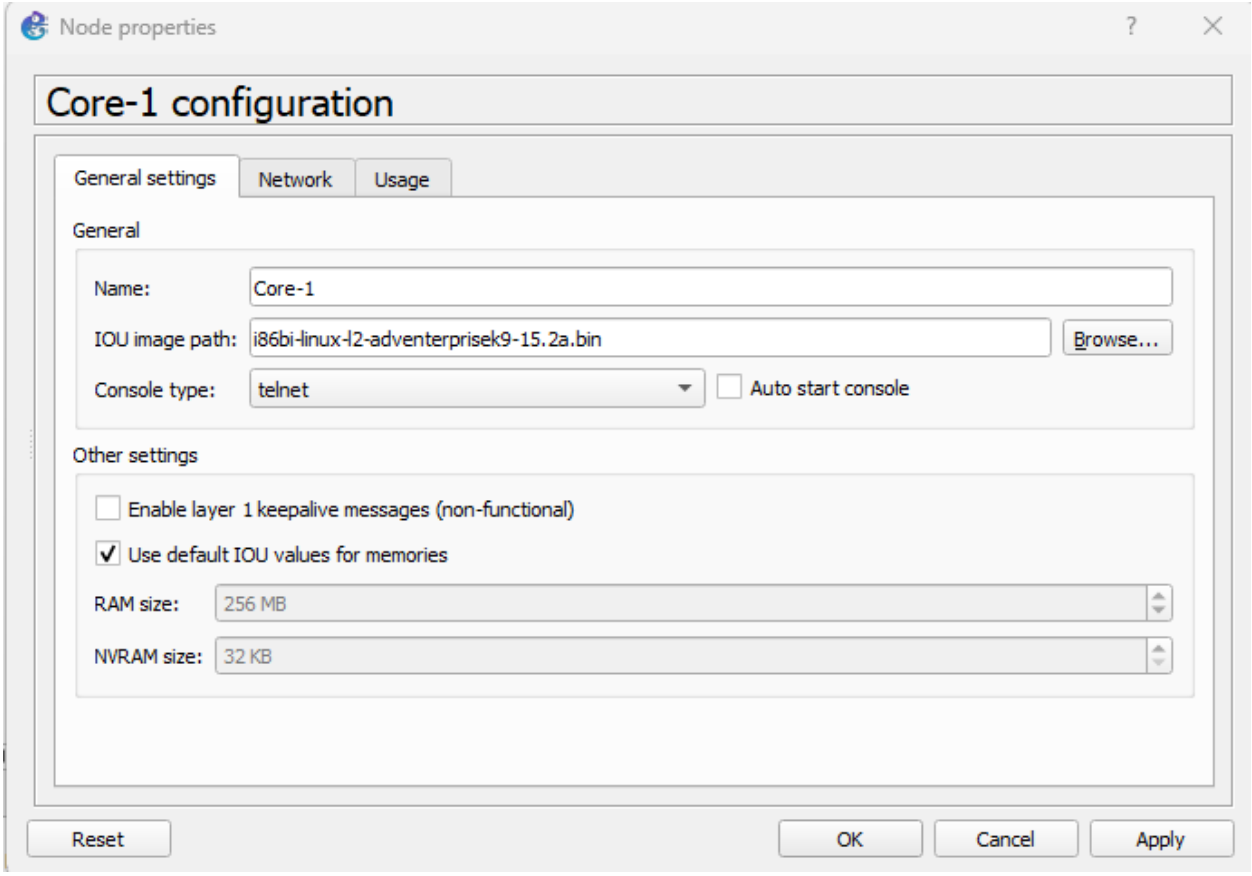


Figure 2-25 core- switch gns3

The Core Layer of the network refers to the following aspects:

- High speed
- Reliability and availability
- Redundancy
- Fault tolerance
- Load balancing
- Manageability and scalability
- No filters, packet handling, or other overhead

- Limited, consistent diameter
- Quality of Service

Speed refers to the ports operating at true wire speed on the core devices. Reliability is another important issue, especially when defining how often the equipment functions at normal parameters. Redundancy influence the recovery time without impacting users.

A very powerful feature used in the Core Layer is called Quality of Service (QoS); however, the QoS parameters implemented in the Core Layer are usually different from the QoS parameters used at the network edge. QoS is a complex and sophisticated topic that will be covered in detail later in this manual. The major QoS techniques found in the Core Layer are as follows:

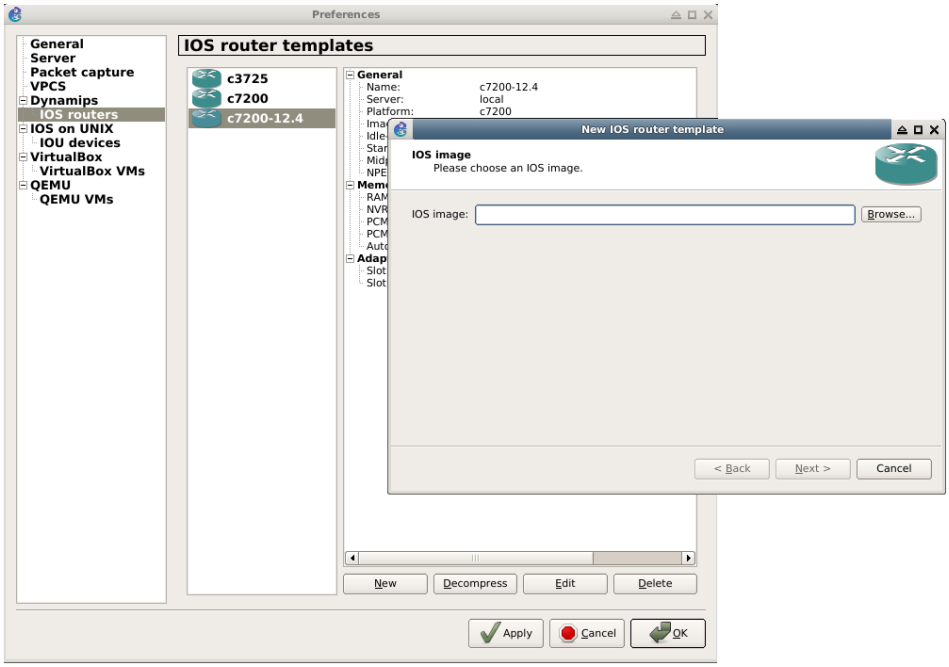
- ✓ **Best effort:** This means over-provisioning of the network, adding, for example, more bandwidth than is currently necessary. This is the most efficient QoS method, but it is also the most expensive.
- ✓ **Integrated Services approach (IntServ):** This is based on the resource reservation concept on the network devices. IntServ uses protocols such as Resource Reservation Protocol (RSVP).
- ✓ **Differentiated Services approach (DiffServ):** This is the most modern and popular QoS approach. DiffServ classifies and marks traffic, and treats it a certain way on a hop-by-hop basis. The DiffServ approach is comprised of the following disciplines:

- Classification
- Marking

- Congestion management
- Congestion avoidance
- Traffic shaping and policing
- Traffic compression
- Link Fragmentation and Interleaving (LFI)

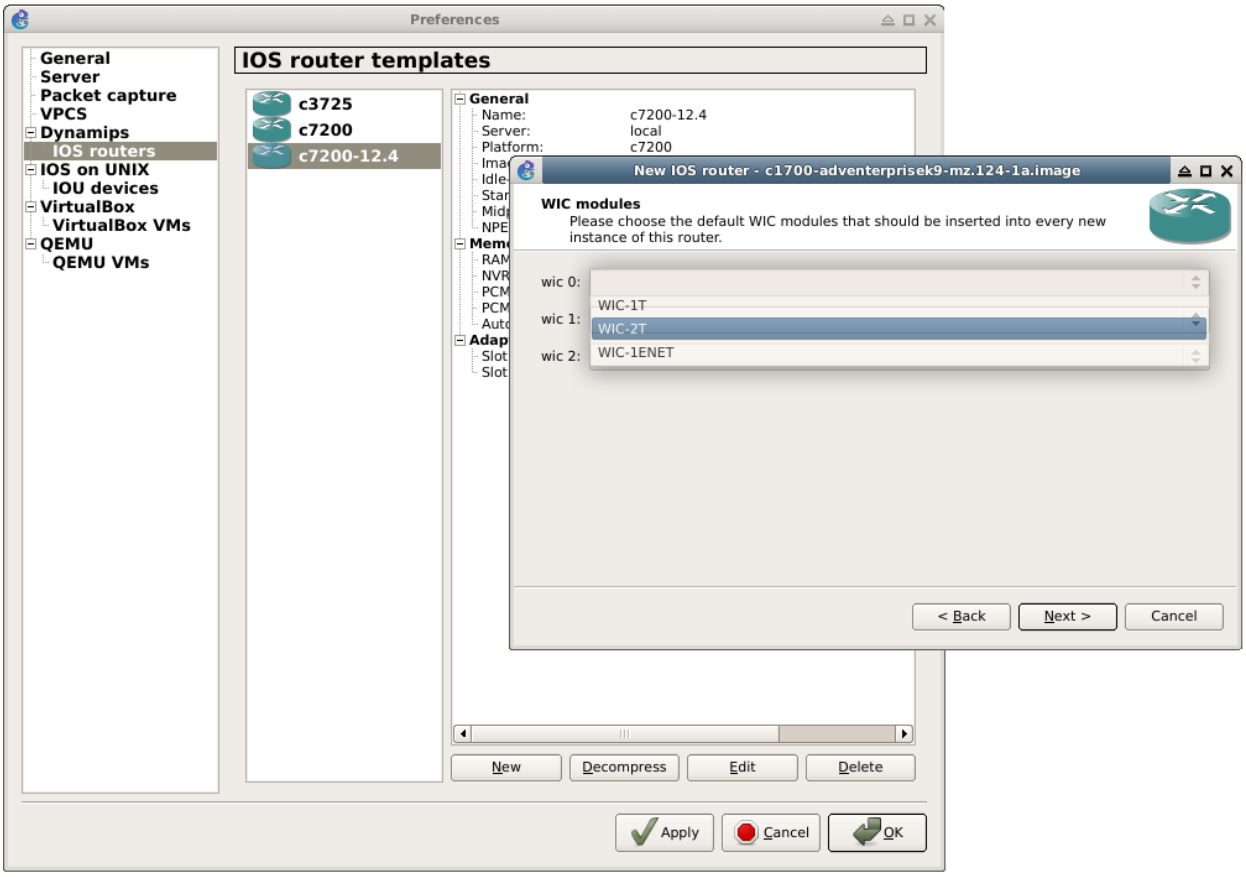
Usually, the classification and marking is carried out at the Access Layer. When using DiffServ, each layer is assigned specific QoS mechanisms that integrate and work with QoS mechanisms at other layers.

2.8.1 IOS image installation



GNS3 IOS image installation :26-2 Figure

IOS Image Setup



GNS3 IOS image setup :27-2 Figure

Cisco Router C7200 & Cisco Switch

These devices formed the backbone of our network infrastructure in the simulation. The **Cisco 7200 router** was selected due to its compatibility with GNS3 and its advanced routing capabilities. It was configured with IP addresses and routing protocols to allow communication between various departments.
The **Cisco switches** were used to create logical segmentation within the network using VLANs. Each department was assigned its own VLAN, enhancing both performance and security. We manually configured each router and switch using Cisco IOS command-line interface (CLI), giving us hands-on experience in real-world configuration practices.

2.9 OPERATING SYSTEMS USED

2.9.1 Ubuntu

Figure 2-28 Ubuntu

- A Linux-based operating system known for its stability and security.
- Used as the base system to host Ansible and run automation scripts.
- Provides an open-source and customizable environment for scripting and networking tasks.

2.9.2 Kali Linux

Figure 2-29 Kali linux

Kali Linux is a security-focused operating system widely used for penetration testing, vulnerability assessment, and ethical hacking. In our project, it served multiple purposes:

- One Kali VM was dedicated to monitoring, running Prometheus and Grafana.
- The other was used for automation and intrusion detection, running Ansible and Snort. Kali's flexibility and pre-installed tools made it a perfect choice for both development and testing of security mechanisms in our network.

2.10 AUTOMATION AND MONITORING TOOLS

2.10.1 Ansible

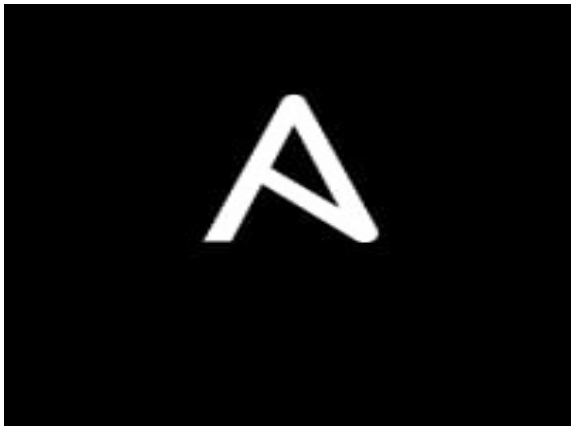


Figure 2-30 Ansible

Ansible is an agentless automation tool that uses YAML-based playbooks to automate system configurations, application deployments, and network device management. It operates over SSH and requires no installation on target machines. In this project, Ansible was used to:

- Push network configuration to routers and switches.
- Standardize settings across devices.
- Reduce time and human error in repetitive tasks.

We developed custom Ansible playbooks that included configuration templates and commands for network devices. By executing them from the Kali Linux Ansible controller, we were able to efficiently configure multiple routers and switches with a single command.

2.10.2 Prometheus

Figure 2-31 Prometheus

Prometheus is an open-source monitoring and alerting toolkit originally developed by SoundCloud. It collects metrics from configured targets at regular intervals, evaluates rule expressions, and can trigger alerts if certain conditions are met.
In our implementation:

- Prometheus was installed on the monitoring server.
- It was configured to collect SNMP data and device statistics from routers and switches.
- It stored the data in a time-series format which was later visualized using Grafana.

Prometheus helped us monitor various parameters such as CPU usage, memory consumption, interface traffic, and device availability.

2.10.3 Grafana

Figure 2-32 Grafana

Grafana is a popular visualization and analytics platform that works seamlessly with Prometheus. It allows users to create interactive and customizable dashboards that visualize real-time data.
We used Grafana to:

- Create dashboards displaying critical metrics.
- Visualize network traffic and device health.
- Track anomalies or abnormal behavior over time.

Dashboards were tailored for our network, showing graphs of bandwidth usage, interface status, system uptime, and alert logs.

2.10.4 Loki

Figure 2-33 Loki

Loki is a log aggregation system developed by Grafana Labs, designed to work like Prometheus but for logs. It stores and queries logs using the same label system as Prometheus.

Key Features:

Efficient and lightweight log aggregation.

Seamless integration with Grafana.

Does not require indexing logs for fast performance.

2.10.5 Promtail

Promtail is an agent that ships the contents of local logs to a Loki instance. It is usually deployed to every machine that has applications needed to be monitored.

Key Features:

Collects log data from files.

Labels logs for searchability in Loki.

Lightweight and easy to configure.

2.10.6 Node Exporter

Node Exporter is a Prometheus exporter for hardware and OS metrics exposed by *nix systems. It provides essential insights into system-level resource usage.

Key Features:

Reports CPU, memory, disk, and network metrics.

Designed for Linux systems.

Seamless integration with Prometheus.

2.10.7 SNMP Exporter

Figure 2-34 SNMP Exporter

The SNMP Exporter enables Prometheus to collect metrics from network devices using the SNMP protocol. It's essential for environments that include traditional network hardware.

Key Features:

Supports multiple SNMP-enabled devices.

Uses Prometheus-compatible formats.

Customizable using SNMP MIBs.

2.10.8 Alert Manager

Figure 2-35 Alert Manager

Alert Manager works in tandem with Prometheus to manage alerts. It handles grouping, deduplication, and routing of alerts to appropriate notification channels. In our setup:

- Alert Manager was integrated with Prometheus and Snort.
- It was configured to send alerts via email or a messaging system when certain thresholds or rule conditions were triggered.
- Alerts were categorized based on severity, allowing us to prioritize incidents.

This helped us maintain proactive security monitoring and respond quickly to potential threats.

CHAPTER 3 : SIMULATION ENVIRONMENT

3.1 INTRODUCTION

Simulating a network topology using GNS3 is a powerful tool that allows you to create and customize a complex network environment with great flexibility. Unlike traditional network simulators like Cisco Packet Tracer, which focus on simpler networks, GNS3 offers a more advanced and accurate simulation environment, enabling you to simulate real network devices such as routers and switches using real device images.

With GNS3, you can build and test complex network simulations, observe how data interacts across different network components, including routing, firewalls, VPNs, and monitor packet flow through the network. It also offers an interactive learning environment that allows you to customize every aspect of the network and route packets across different layers of the OSI model.

Additionally, GNS3 supports the integration of tools like Wireshark for packet analysis, enabling real-time data flow monitoring, making it an ideal tool for advanced learning and professional networking. GNS3 also provides multi-user collaboration, facilitating teamwork and sharing simulations between different teams working on network projects.

3.2 NETWORK TOPOLOGY DESIGN

This part outlines the network topology designed and implemented using GNS3 to represent the traditional network structure before applying DevOps tools. It highlights the layout of departments, interconnections between network devices, and the logical segmentation implemented using VLANs.

Overview of the Network Layout

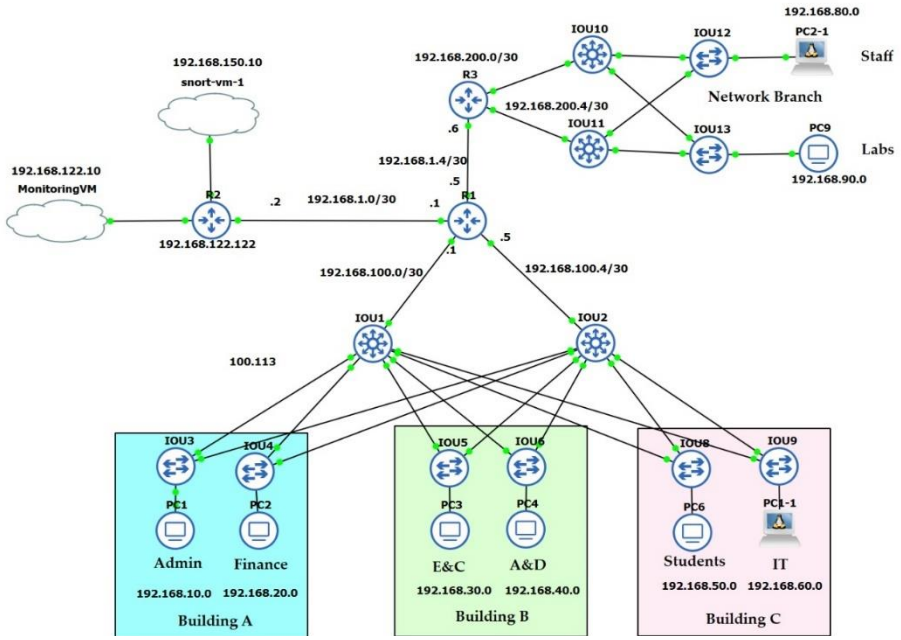


Figure 3-1simulation topology

Our network was designed to simulate a medium-sized educational or enterprise environment. The network consists of multiple segments representing various departments such as Administration, Finance, IT, and Students. Each department was assigned its own VLAN to logically separate traffic and enhance network performance and security.

3.2.1 Departmental Segmentation

The network topology includes:

Admin Department

Finance Department

IT Department

Students Network

STAFF Network

E&C Department

A&D Department

LABs Department

Each of these departments is connected via Layer 2 switches to a central Layer 3 router that handles inter-VLAN routing. This segmentation ensures data isolation and allows for better traffic management.

Topology Visualization

The initial network layout was first drafted using Packet Tracer to visualize the structure in a traditional network design approach. It included:

One main router (Cisco 7200)

multilayer switch (MLS)

Multiple Layer 2 switches

End devices representing departmental PCs

Connections between all components

Later, this topology was migrated to GNS3 for advanced configuration and realistic simulation using Cisco IOS.

3.2.2 VLAN Design and Implementation

Each switch was configured with multiple VLANs corresponding to the departments. VLANs were implemented to:

Reduce broadcast domains

Isolate traffic between departments

Improve security and network performance

Example VLAN Configuration:

```
Switch(config)# vlan 10
Switch(config-vlan)# name Admin
Switch(config)# vlan 20
Switch(config-vlan)# name Finance
Switch(config)# vlan 30
Switch(config-vlan)# name E&C
Switch(config)# vlan 40
Switch(config-vlan)# name A&D
Switch(config)# vlan 50
Switch(config-vlan)# name Students
Switch(config)# vlan 60
Switch(config-vlan)# name it
Switch(config)# vlan 70
Switch(config-vlan)# name STAFF
Switch(config)# vlan 80
Switch(config-vlan)# name LAB
```

3.3 INTERCONNECTION BETWEEN ROUTERS AND SWITCHES USING MLS

In this setup, Multilayer Switching (MLS) was used for inter-VLAN communication. The Cisco 7200 router was integrated with MLS-capable switches to provide both switching and routing functionalities at Layer 3, reducing the need for traditional router sub-interfaces for inter-VLAN routing.

Each switch was configured with VLAN interfaces and enabled to perform Layer 3 switching to forward traffic between VLANs directly, eliminating the need for trunking through a router. This setup enables more efficient communication between VLANs by offloading routing tasks to the switch, thereby improving overall network performance.

Example Configuration for Multilayer Switch:

```
Switch(config)# interface Vlan10

Switch(config-if)# ip address 192.168.10.1 255.255.255.0

Switch(config-if)# no shutdown
```

With MLS, the switch performs both Layer 2 switching and Layer 3 routing on the same device, optimizing traffic flow between VLANs while reducing latency and enhancing performance.

Final Design Implementation in GNS3

Once verified in Packet Tracer, the design was recreated in GNS3. The final design in GNS3 included:

The actual devices used: Cisco Router C7200, Cisco Switches, and Kali Linux VMs.

Configurations applied for IP addressing, VLANs, trunking, and routing.

Connectivity checks to ensure all departments could communicate as expected.

Diagram and Visual Representation

The following diagram (taken from the Packet Tracer design) visually represents the network layout, showing how each department is connected and how the core routing and switching infrastructure is built.

3.4 NETWORK CONFIGURATION

details the manual configuration process of the network within GNS3. It includes IP addressing schemes, VLAN assignment, inter-VLAN routing, and connectivity verification through practical CLI commands.

3.4.1 IP Addressing Scheme

Each VLAN was assigned a specific subnet for clarity and routing purposes:

- VLAN 10 (Admin): 192.168.10.0/24
- VLAN 20 (Finance): 192.168.20.0/24
- VLAN 30 (E&C): 192.168.30.0/24
- VLAN 40 (A&D): 192.168.40.0/24
- VLAN 50 (Students): 192.168.50.0/24
- VLAN 60 (IT): 192.168.60.0/24
- VLAN 70 (STAFF): 192.168.70.0/24
- VLAN 80 (LAB): 192.168.80.0/24

This logical segmentation enhances network management and security.

3.4.2 Addressing Table

Table 3-1:Addressing Table

Dist-sw1		
Int vlan n	192.168.n.2	255.255.255.0
VRRP	192.168.n.1	255.255.255.0

Ethernet 0/0	192.168.100.2	255.255.255.252
Ethernet 0/1	trunk encapsulation dot1q	
Ethernet 0/2	trunk encapsulation dot1q	
Ethernet 0/3	trunk encapsulation dot1q	
Ethernet 1/0	trunk encapsulation dot1q	
Ethernet 1/1	trunk encapsulation dot1q	
Ethernet 1/2	trunk encapsulation dot1q	
Ethernet 2/0	trunk encapsulation dot1q	
Access-switch-0		
Int vlan 10	192.168.10.5	255.255.255.0
Ethernet 0/0	trunk encapsulation dot1q	
Ethernet 0/1	Access vlan 10	
Access-switch-2		
Int vlan 1	192.168.20.5	255.255.255.0
Ethernet 0/0	trunk encapsulation dot1q	
Ethernet 0/1	Access vlan 20	
Access-switch-3		

Int vlan 1	192.168.30.5	255.255.255.0
Ethernet 0/0	trunk encapsulation dot1q	
Ethernet 0/1	Access vlan 30	
Access-switch-4		
Int vlan 1	192.168.40.5	255.255.255.0
Ethernet 0/0	trunk encapsulation dot1q	
Ethernet 0/1	Access vlan 40	
Access-switch-5		
Int vlan 1	192.168.50.5	255.255.255.0
Ethernet 0/0	trunk encapsulation dot1q	
Ethernet 0/1	Access vlan 50	
Access-switch-6		
Int vlan 1	192.168.60.5	255.255.255.0
Ethernet 0/0	trunk encapsulation dot1q	
Ethernet 0/1	Access vlan 60	
Access-switch-7		

Int vlan 1	192.168.70.5	255.255.255.0
Ethernet 0/0	trunk encapsulation dot1q	
Ethernet 0/1	Access vlan 70	
Access-switch-8		
Int vlan 1	192.168.80.5	255.255.255.0
Ethernet 0/0	trunk encapsulation dot1q	
Ethernet 0/1	Access vlan 80	
R1		
F0/0	192.168.100.1	255.255.255.252
F1/0	192.168.100.6	255.255.255.252
F1/1	192.168.1.1	255.255.255.252
F2/0	192.168.1.5	255.255.255.252
Tunnel (R1<>R3)	10.10.10.1	255.255.255.0
R3		
F0/0	192.168.1.6	255.255.255.252
F1/0	192.168.200.1	255.255.255.252

F1/1	192.168.200.5	255.255.255.252
Tunnel (R1<>R3)	10.10.10.2	255.255.255.0
R2		
F0/0	192.168.1.2	255.255.255.252
F0/1	192.168.122.0	255.255.255.0
F0/1	192.168.150.1	255.255.255.0
Dist-sw2		
Int vlan n	192.168.n.3	255.255.255.0
VRRP	192.168.n.1	255.255.255.0
Ethernet 0/0	192.168.100.6	255.255.255.252
Ethernet 0/1	trunk encapsulation dot1q	
Ethernet 0/2	trunk encapsulation dot1q	
Ethernet 0/3	trunk encapsulation dot1q	
Ethernet 1/0	trunk encapsulation dot1q	
Ethernet 1/1	trunk encapsulation dot1q	
Ethernet 1/2	trunk encapsulation dot1q	

Ethernet 2/0	trunk encapsulation dot1q	
Dist-sw3(iou 10)		
Int vlan n	192.168.n.2	255.255.255.0
VRRP	192.168.n.1	255.255.255.0
Ethernet 0/0	192.168.200.2	255.255.255.252
Ethernet 0/1	trunk encapsulation dot1q	
Ethernet 0/2	trunk encapsulation dot1q	
Dist-sw4(iou 11)		
Int vlan n	192.168.n.3	255.255.255.0
VRRP	192.168.n.1	255.255.255.0
Ethernet 0/0	192.168.200.6	255.255.255.252
Ethernet 0/1	trunk encapsulation dot1q	
Ethernet 0/2	trunk encapsulation dot1q	

3.5 SWITCH & ROUTRES CONFIGURATION

Each switch port was assigned to the appropriate VLAN according to the department it belongs to. Trunk links were configured between switches and the router to support multiple VLANs.

MLS 1

Table 3-2 Multi-Layer Switch 1

```
en
conf t
hostname MLS1
vlan 10
vlan 20
vlan 30
vlan 40
vlan 50
vlan 60
ip domain-name example.com
username admin privilege 15 password mma
crypto key generate rsa mod 2048
ip ssh version 2
line vty 0 4
login local
transport input ssh
ip dhcp pool admin
```

```
network 192.168.10.0 255.255.255.0
default-router 192.168.10.1
dns-server 192.168.10.1
ip dhcp pool student
network 192.168.20.0 255.255.255.0
default-router 192.168.20.1
dns-server 192.168.20.1
ip dhcp pool it
network 192.168.30.0 255.255.255.0
default-router 192.168.30.1
dns-server 192.168.30.1
ip dhcp pool HR
network 192.168.40.0 255.255.255.0
default-router 192.168.40.1
dns-server 192.168.40.1
ip dhcp pool ENGINEER
network 192.168.50.0 255.255.255.0
default-router 192.168.50.1
dns-server 192.168.50.1
ip dhcp pool FINANCE
network 192.168.60.0 255.255.255.0
default-router 192.168.60.1
dns-server 192.168.60.1
interface  range e0/1-3
```

```
switchport trunk encapsulation dot1q
 switchport mode trunk
 no shutdown
interface range e1/0-2
switchport trunk encapsulation dot1q
 switchport mode trunk
 no shutdown
interface  e0/0
no switch
ip add 192.168.100.2 255.255.255.252
ROUTER RIP
VER 2
network 192.168.10.0
network 192.168.100.0
network 192.168.100.4
network 192.168.20.0
network 192.168.30.0
network 192.168.40.0
network 192.168.50.0
network 192.168.60.0
ip route 0.0.0.0 0.0.0.0 192.168.100.1
interface  vlan 10
no shu
ip add 192.168.10.2 255.255.255.0
```

```
interface  vlan 20
no shu
ip add 192.168.20.2 255.255.255.0
interface  vlan 30
no shu
ip add 192.168.30.2 255.255.255.0
interface  vlan 40
no shu
ip add 192.168.40.2 255.255.255.0
interface  vlan 50
no shu
ip add 192.168.50.2 255.255.255.0
interface  vlan 60
no shu
ip add 192.168.60.2 255.255.255.0
int vlan 10
vrrp 10 ip 192.168.10.1
int vlan 20
vrrp 10 ip 192.168.20.1
int vlan 30
vrrp 10 ip 192.168.30.1
int vlan 40
vrrp 10 ip 192.168.40.1
int vlan 50
```

```
vrrp 10 ip 192.168.50.1
int vlan 60
vrrp 10 ip 192.168.60.1
do wr
```

MLS 2

Table 3-3 Multi-Layer Switch 2

```
en
conf t
hostname MLS2
vlan 10
vlan 20
vlan 30
vlan 40
vlan 50
vlan 60
ip domain-name example.com
username admin privilege 15 password mma
crypto key generate rsa mod 2048
ip ssh version 2
line vty 0 4
login local
transport input ssh
ip dhcp pool admin
network 192.168.10.0 255.255.255.0
```

```
default-router 192.168.10.1
dns-server 192.168.10.1
ip dhcp pool student
network 192.168.20.0 255.255.255.0
default-router 192.168.20.1
dns-server 192.168.20.1
ip dhcp pool it
network 192.168.30.0 255.255.255.0
default-router 192.168.30.1
dns-server 192.168.30.1
ip dhcp pool HR
network 192.168.40.0 255.255.255.0
default-router 192.168.40.1
dns-server 192.168.40.1
ip dhcp pool ENGINEER
network 192.168.50.0 255.255.255.0
default-router 192.168.50.1
dns-server 192.168.50.1
ip dhcp pool FINANCE
network 192.168.60.0 255.255.255.0
default-router 192.168.60.1
dns-server 192.168.60.1
interface  range e0/1-3
switchport trunk encapsulation dot1q
```

```
 switchport mode trunk
 no shutdown
interface  e1/0-2
switchport trunk encapsulation dot1q
 switchport mode trunk
 no shutdown
interface  e0/0
no switch
ip add 192.168.100.6 255.255.255.252
ROUTER RIP
VER 2
network 192.168.100.0
network 192.168.100.4
network 192.168.10.0
network 192.168.20.0
network 192.168.30.0
network 192.168.40.0
network 192.168.50.0
network 192.168.60.0
ip route 0.0.0.0 0.0.0.0 192.168.100.5
interface  vlan 10
no shu
ip add 192.168.10.3 255.255.255.0
interface  vlan 20
```

```
no shu
ip add 192.168.20.3 255.255.255.0
interface  vlan 30
no shu
ip add 192.168.30.3 255.255.255.0
interface  vlan 40
no shu
ip add 192.168.40.3 255.255.255.0
interface  vlan 50
no shu
ip add 192.168.50.3 255.255.255.0
interface  vlan 60
no shu
ip add 192.168.60.3 255.255.255.0
int vlan 10
vrrp 10 ip 192.168.10.1
int vlan 20
vrrp 10 ip 192.168.20.1

int vlan 30
vrrp 10 ip 192.168.30.1
int vlan 40
vrrp 10 ip 192.168.40.1
int vlan 50
```

```
vrrp 10 ip 192.168.50.1
int vlan 60
vrrp 10 ip 192.168.60.1

do wr
```

R1

Table 3-4 Router1

```
en
conf t
hostname R1core-1
interface FastEthernet0/0
ip address 192.168.100.1 255.255.255.252
no shutdown
interface FastEthernet1/0
ip address 192.168.100.5 255.255.255.252
no shutdown
interface FastEthernet1/1
ip address 192.168.1.1 255.255.255.252
no shutdown
interface FastEthernet2/0
ip address 192.168.1.5 255.255.255.252
no shutdown
ip domain-name example.com
username admin privilege 15 password mma
```

```
crypto key generate rsa mod 2048
ip ssh version 2
line vty 0 4
login local
transport input ssh
ROUTER RIP
VER 2
network 192.168.1.4
network 192.168.1.0
network 192.168.100.0
network 192.168.100.4
do wr
interface Tunnel1
ip address 10.10.10.1 255.255.255.0
tunnel source f2/0
tunnel destination 192.168.1.6
tunnel mode ipsec ipv4
tunnel protection ipsec profile TUNSEC
no shutdown
exit
ip route 10.10.10.0 255.255.255.0 Tunnel1
crypto isakmp policy 10
authentication pre-share
encryption aes 256
```

```
hash sha256
group 2
lifetime 86400
exit
crypto isakmp key VPN_SECRET_KEY address 192.168.1.6
crypto ipsec transform-set VPN_TRANSFORM_SET esp-aes 256 esp-sha256-hmac
crypto  ipsec profile TUNSEC
set transform-set VPN_TRANSFORM_SET
exit
crypto map VPN_MAP 10 ipsec-isakmp
set peer 192.168.1.6
set transform-set VPN_TRANSFORM_SET
 match address 100
interface FastEthernet2/0
crypto map VPN_MAP
do wr
```

Access switch-**1**

Table 3-5 Access switch-1

```
en
conf t
vlan 10
int  e0/2
switchport mode access
switchport access  vlan 10
```

```
no sh
int range  e0/0-1
switchport trunk encapsulation dot1q
 switchport mode trunk
 no shutdown
no sh
int vlan 1
ip add 192.168.100.113 255.255.255.0
no sh
do wr
```

Access switch**-2**

Table 3-6 Accessswitch -2

```
en
conf t
vlan 20
int  e0/2
switchport mode access
switchport access  vlan 20
no sh
int range  e0/0-1
switchport trunk encapsulation dot1q
 switchport mode trunk
 no shutdown
no sh
```

```
int vlan 1
ip add 192.168.100.114 255.255.255.0
no sh
do wr
```

Access switch-**3**

Table 3-7 Access switch-3

```
en
conf t
vlan 30
int  e0/2
switchport mode access
switchport access  vlan 30
no sh
int range  e0/0-1
switchport trunk encapsulation dot1q
 switchport mode trunk
 no shutdown
no sh
int vlan 1
ip add 192.168.100.115 255.255.255.0
no sh
do wr
```

Access switch-**4**

Table 3-8 Access switch-4

```
en
conf t
vlan 40
int  e0/2
switchport mode access
switchport access  vlan 40
no sh
int range  e0/0-1
switchport trunk encapsulation dot1q
 switchport mode trunk
 no shutdown
no sh
int vlan 1
ip add 192.168.100.116 255.255.255.0
no sh
do wr
```

Access switch-**5**

Table 3-9 Access switch-5

```
en
conf t
vlan 10
```

```
int  e0/2
switchport mode access
switchport access  vlan 50
no sh
int range  e0/0-1
switchport trunk encapsulation dot1q
 switchport mode trunk
 no shutdown
no sh
int vlan 1
ip add 192.168.100.118 255.255.255.0
no sh
do wr
```

Access switch**-6**

Table 3-10 Access switch-6

```
en
conf t
vlan 10
int  e0/2
switchport mode access
switchport access  vlan 60
no sh
int range  e0/0-1
switchport trunk encapsulation dot1q
```

```
 switchport mode trunk
 no shutdown
no sh
int vlan 1
ip add 192.168.100.119 255.255.255.0
no sh
do wr
```

R2

Table 3-11 Router 2

```
conf t
hostname R2core-2
interface FastEthernet0/0
ip address 192.168.1.2 255.255.255.252
no shutdown
interface FastEthernet1/0
ip address 192.168.122.1 255.255.255.0
no shutdown
ip dhcp pool admin
network 192.168.122.0 255.255.255.0
default-router 192.168.122.1
dns-server 192.168.122.1
ip domain-name example.com
username admin privilege 15 password mma
crypto key generate rsa mod 2048
```

```
ip ssh version 2
line vty 0 4
login local
transport input ssh
ROUTER RIP
VER 2
network 192.168.122.0
network 192.168.1.0
network 192.168.150.0
do wr
```

R3

Table 3-12 Router 3

```
R3core-3
en
conf t
hostname R3core-3
interface FastEthernet0/0
ip address 192.168.1.6 255.255.255.252
no shutdown
interface FastEthernet1/0
ip address 192.168.200.1 255.255.255.252
no shutdown
interface FastEthernet1/1
ip address 192.168.200.5 255.255.255.252
```

```
no shutdown
ip domain-name example.com
username admin privilege 15 password mma
crypto key generate rsa mod 2048
ip ssh version 2
line vty 0 4
login local
transport input ssh
ROUTER RIP
VER 2
network 192.168.1.4
network 192.168.200.0
network 192.168.200.4
do wr
configure terminal
interface FastEthernet0/0
 ip address 192.168.1.6 255.255.255.252
 no shutdown
interface Tunnel1
 ip address 10.10.10.2 255.255.255.0
 tunnel source FastEthernet0/0
 tunnel destination 192.168.1.5
 tunnel mode ipsec ipv4
 tunnel protection ipsec profile TUNSEC
```

```
no sh
 exit
ip route 10.10.10.0 255.255.255.0 Tunnel1
crypto isakmp policy 10
 authentication pre-share
 encryption aes 256
 hash sha256
 group 2
 lifetime 86400
 exit
crypto isakmp key VPN_SECRET_KEY address 192.168.1.5
crypto ipsec transform-set VPN_TRANSFORM_SET esp-aes 256 esp-sha256-hmac
 exit
crypto ipsec profile TUNSEC
 set transform-set VPN_TRANSFORM_SET
 exit
crypto map VPN_MAP 10 ipsec-isakmp
 set peer 192.168.1.5
 set transform-set VPN_TRANSFORM_SET
 match address 100
 exit
interface FastEthernet0/0
crypto map VPN_MAP
do wr
```

MLS 3 (iou10)

Table 3-13 Multi-Layer Switch 3 (iou10)

```
conf t
hostname MLS3
vlan 80
vlan 90
ip domain-name example.com
username admin privilege 15 password mma
crypto key generate rsa mod 2048
ip ssh version 2
line vty 0 4
login local
transport input ssh
ip dhcp pool staf
network 192.168.80.0 255.255.255.0
default-router 192.168.80.1
dns-server 192.168.80.1
ip dhcp pool student-lab
network 192.168.90.0 255.255.255.0
default-router 192.168.90.1
dns-server 192.168.90.1
interface  range e0/1-2
switchport trunk encapsulation dot1q
```

```
 switchport mode trunk
 no shutdown
interface  e0/0
no switch
ip add 192.168.200.2 255.255.255.252
ROUTER RIP
VER 2
network 192.168.200.0
network 192.168.200.4
network 192.168.80.0
network 192.168.90.0
ip route 0.0.0.0 0.0.0.0 192.168.200.1
interface  vlan 80
no shu
ip add 192.168.80.2 255.255.255.0
interface  vlan 90
no shu
ip add 192.168.90.2 255.255.255.0
int vlan 80
vrrp 10 ip 192.168.80.1
int vlan 90
vrrp 10 ip 192.168.90.1
```

MLS 4 (iou11)

Table 3-14 Multi-Layer Switch 4 (iou11)

```
en
conf t
hostname MLS4
vlan 80
vlan 90
ip domain-name example.com
username admin privilege 15 password mma
crypto key generate rsa mod 2048
ip ssh version 2
line vty 0 4
login local
transport input ssh
ip dhcp pool staf
network 192.168.80.0 255.255.255.0
default-router 192.168.80.1
dns-server 192.168.80.1
ip dhcp pool student-lab
network 192.168.90.0 255.255.255.0
default-router 192.168.90.1
dns-server 192.168.90.1
interface  range e0/1-2
switchport trunk encapsulation dot1q
```

```
 switchport mode trunk
 no shutdown
interface  e0/0
no switch
ip add 192.168.200.6 255.255.255.252
ROUTER RIP
VER 2
network 192.168.200.0
network 192.168.200.4
network 192.168.80.0
network 192.168.90.0
ip route 0.0.0.0 0.0.0.0 192.168.200.5
interface  vlan 80
no shu
ip add 192.168.80.3 255.255.255.0
interface  vlan 90
no shu
ip add 192.168.90.3 255.255.255.0
int vlan 80
vrrp 10 ip 192.168.80.1
int vlan 90
vrrp 10 ip 192.168.90.1
do wr
```

Access switch-**8**

Table 3-15 Access switch -8

```
en
conf t
vlan 80
int  e0/2
switchport mode access
switchport access  vlan 80
no sh
int range  e0/0-1
switchport trunk encapsulation dot1q
 switchport mode trunk
 no shutdown
no sh
int vlan 1
ip add 192.168.200.12 255.255.255.0
no sh
do wr
```

Access switch-**9**

Table 3-16 Access switch-9

```
conf t
vlan 90
int  e0/2
```

```
switchport mode access
switchport access  vlan 90
no sh
int range  e0/0-1
switchport trunk encapsulation dot1q
 switchport mode trunk
 no shutdown
no sh
int vlan 1
ip add 192.168.200.13 255.255.255.0
no sh
do wr
```

3.6 CONNECTIVITY TESTING

To validate successful configuration, we used basic network tools:

- **Ping Test**: Ensure connectivity between VLANs
- **Traceroute**: Confirm proper routing path
- **show commands**: Verify port assignments, interface statuses, and routing tables

Troubleshooting Techniques

During testing, some common issues were encountered:

- **Missing trunking configuration**: Resolved by ensuring all trunk links were active
- **Incorrect IP/subnet assignments**: Fixed by double-checking addressing tables
- **Disabled interfaces**: Solved by enabling interfaces with no shutdown

Security Hardening

We implemented basic Layer 2 security configurations:

Table 3-17 Security Hardening

```
Switch(config)# interface range fa0/6 - 23
Switch(config-if-range)# shutdown
Switch(config)# service password-encryption
Switch(config)# enable secret securepass
```

Design Consistency Check

The final implementation was reviewed against the original topology design to ensure:

- Logical alignment between departments and VLANs
- Proper device interconnections
- Scalable and modular structure

CHAPTER 4 : MONITORING SYSTEM IMPLEMENTATION

4.1 KALI LINUX

Kali Linux is a Debian-based Linux distribution specifically geared towards penetration testing, ethical hacking, and security auditing.

🛠 What Kali Linux Does:

Provides a pre-installed collection of over 600 penetration-testing tools.

Supports activities like vulnerability analysis, wireless attacks, reverse engineering, and digital forensics.

⭐ Key Features:

Massive toolset: Comes bundled with top security tools like Nmap, Wireshark, Metasploit, Burp Suite, John the Ripper, and Aircrack-ng.

Customizable: Allows users to build their own Kali ISOs and choose specific toolsets.

Support for ARM devices: Can be run on Raspberry Pi and similar devices.

Live Boot capability: Can be booted directly from USB/CD without installing.

Updated Regularly: Frequent updates to keep tools and system security up-to-date.

🔥 Why We Use Kali Linux in Our Project:

To simulate real-world cyberattacks such as DoS/DDoS for IDS testing (using Snort).

To perform vulnerability assessments of the network setup.

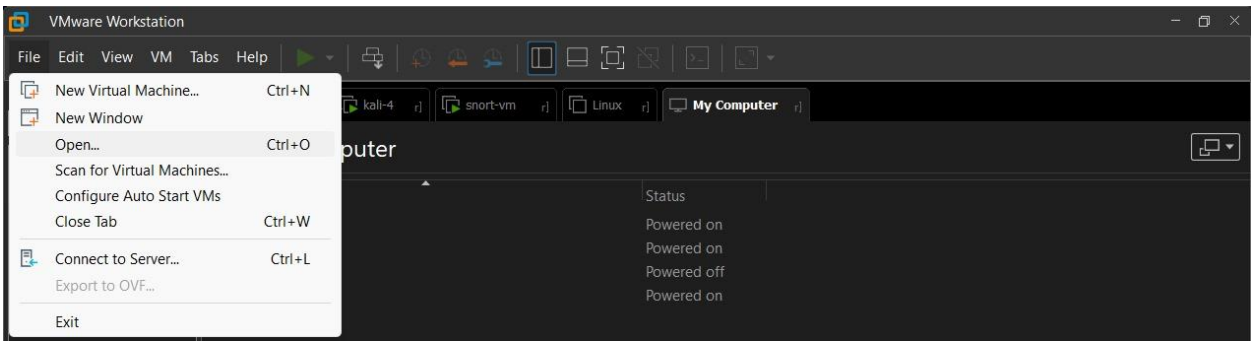


Figure 4-1 Kali Linux Add

To generate traffic for testing monitoring and alerting systems.

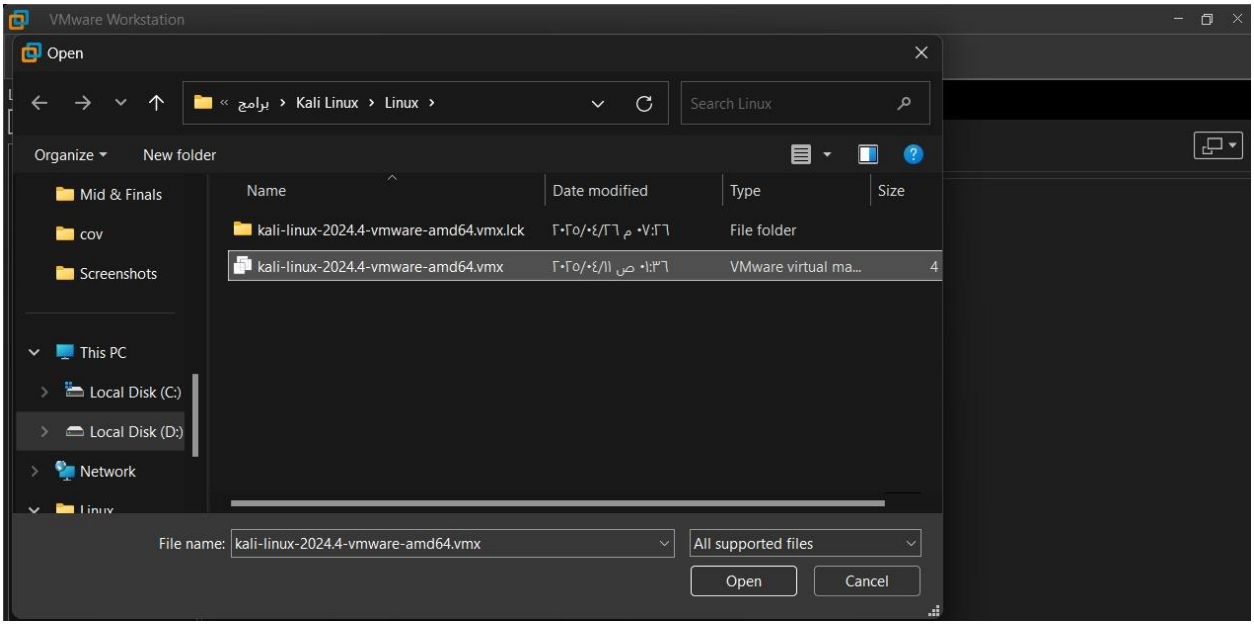


Figure 4-3 Kali Linux VM

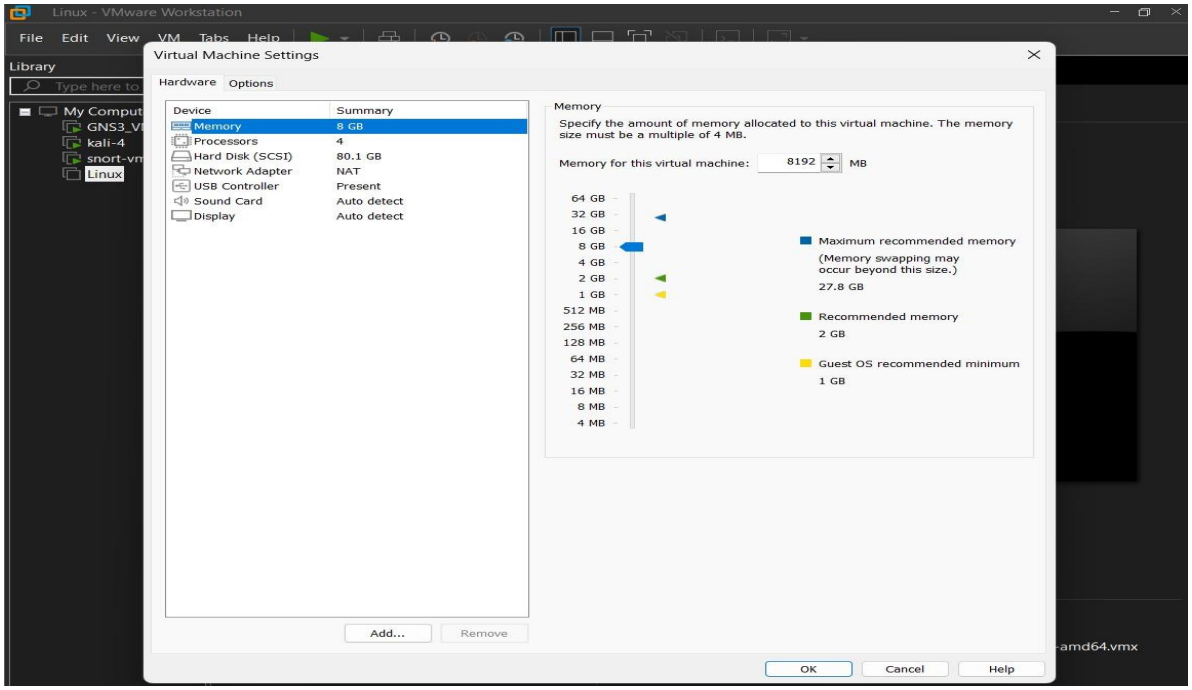
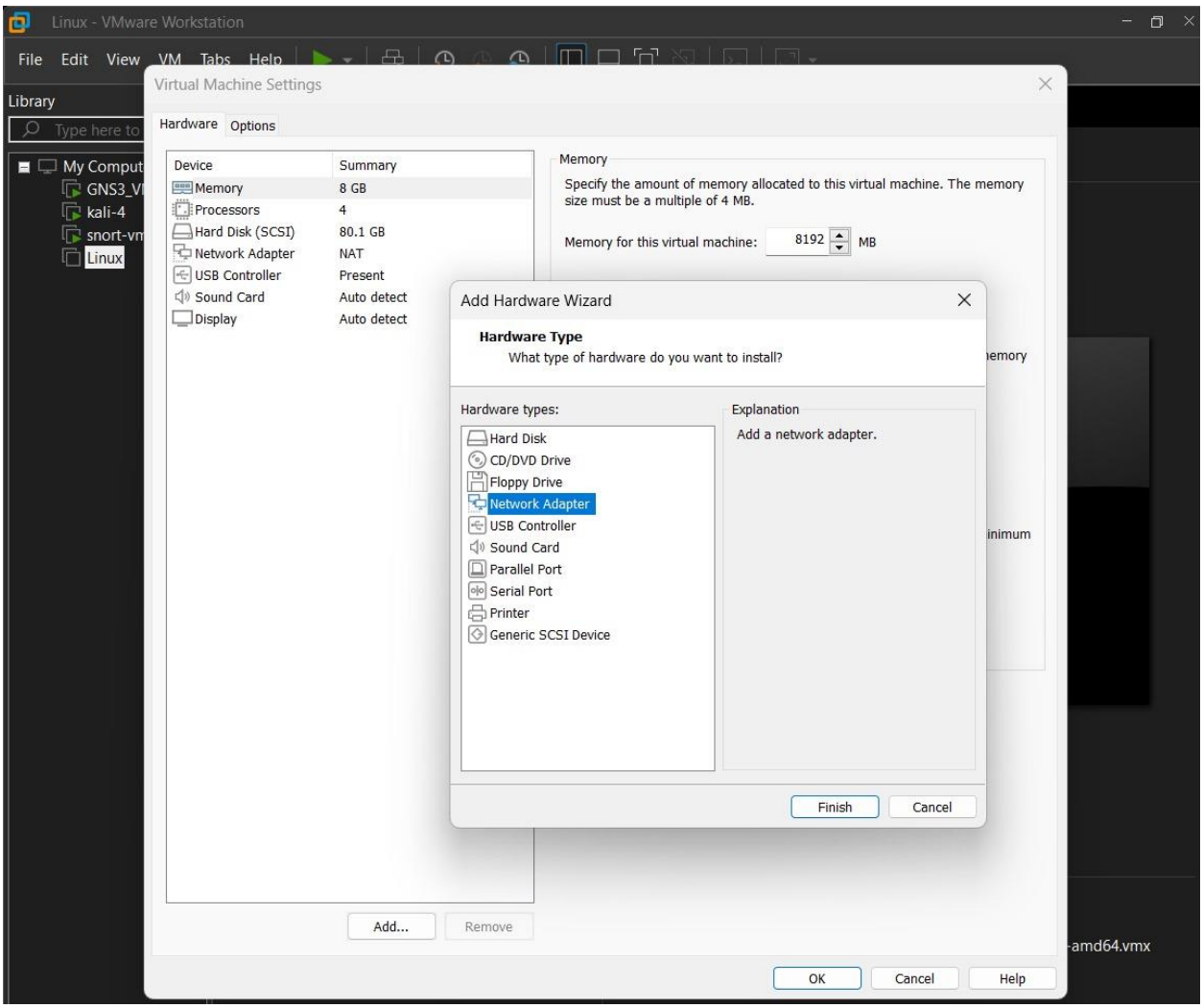


Figure 4-2 Kali Linux Setting


Linux - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to
My Comput
GNS3_V
kali-4
snort-vm
Linux
Virtual Machine Settings
Hardware Options
Device Summary
Memory 8 GB
Processors 4
Hard Disk (SCSI) 80.1 GB
Network Adapter NAT
USB Controller Present
Sound Card Auto detect
Display Auto detect
Memory
Specify the amount of memory allocated to this virtual machine. The memory size must be a multiple of 4 MB.
Memory for this virtual machine: 8192 MB
Add Hardware Wizard
Hardware Type
What type of hardware do you want to install?
Hardware types:
Hard Disk
CD/DVD Drive
Floppy Drive
Network Adapter
USB Controller
Sound Card
Parallel Port
Serial Port
Printer
Generic SCSI Device
Explanation
Add a network adapter.
Finish
Cancel
Add...
Remove
OK
Cancel
Help
-amd64.vmx

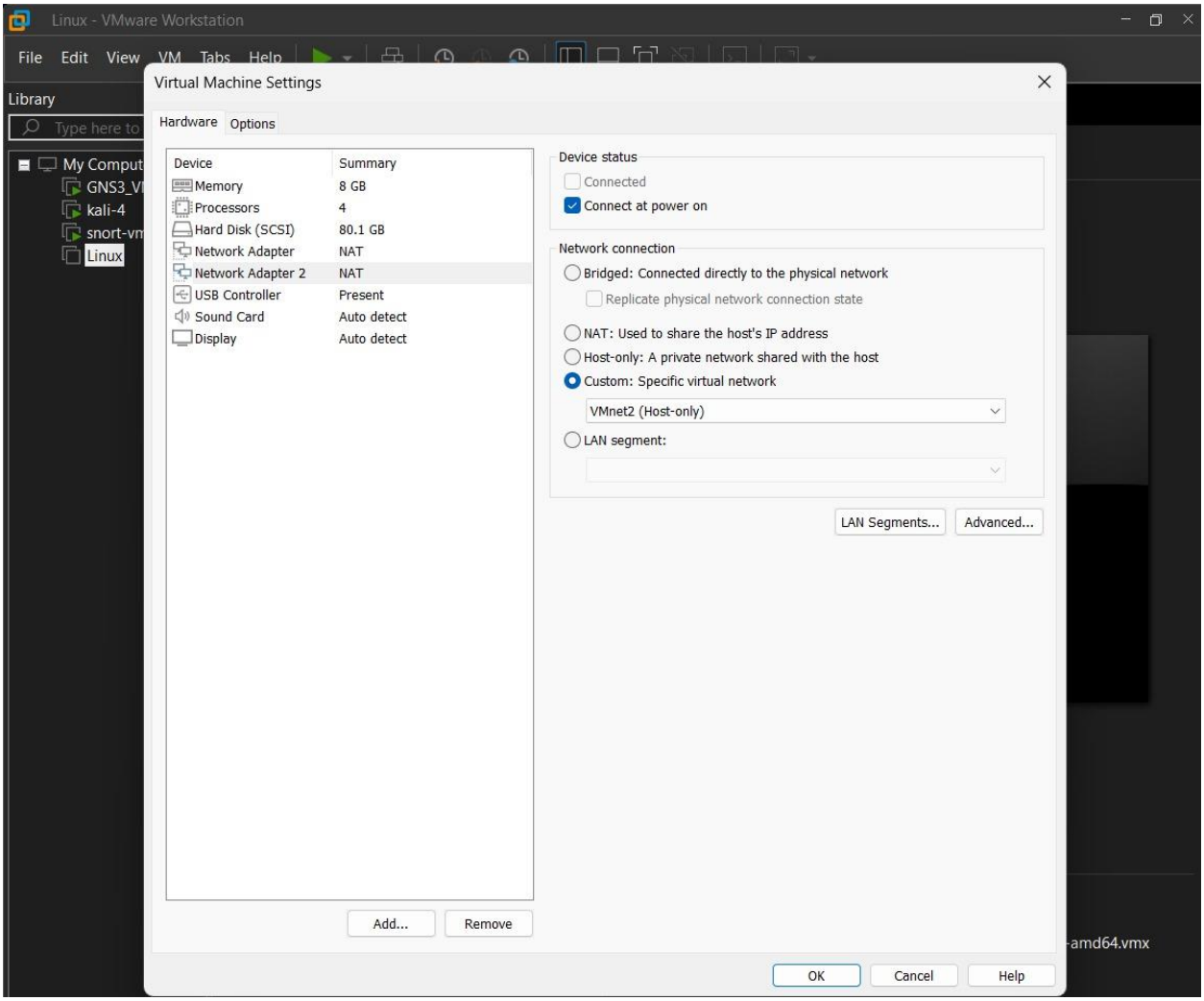


Figure 4-4 Kali Linux Adapter NAT

Note: We must first add a network adapter to the GNS3 VM itself, set it to Custom, and configure it as Host-Only.

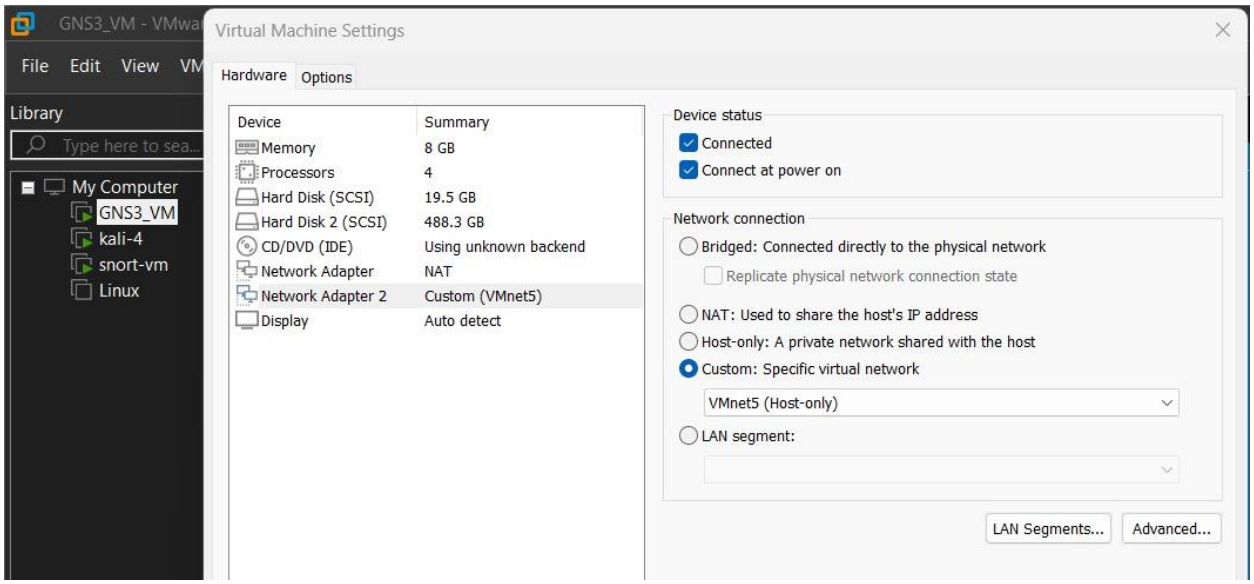


Figure 4-5 Kali Linux Adapter

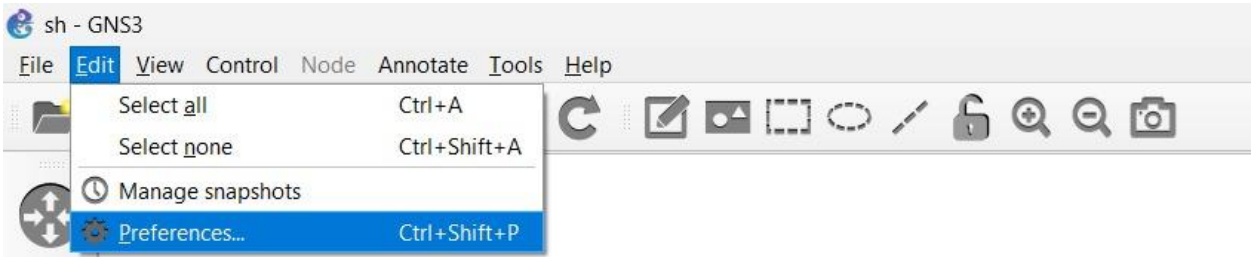


Figure 4-6 Kali Linux preferences Vnet5

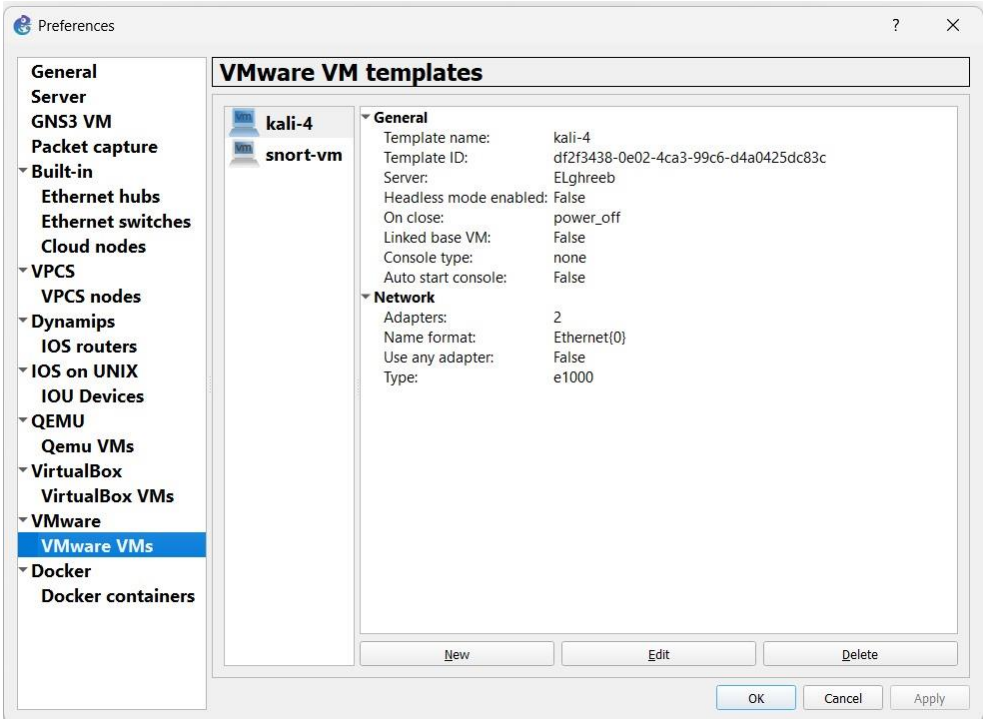


Figure 4-7 Kali Linux GNS3

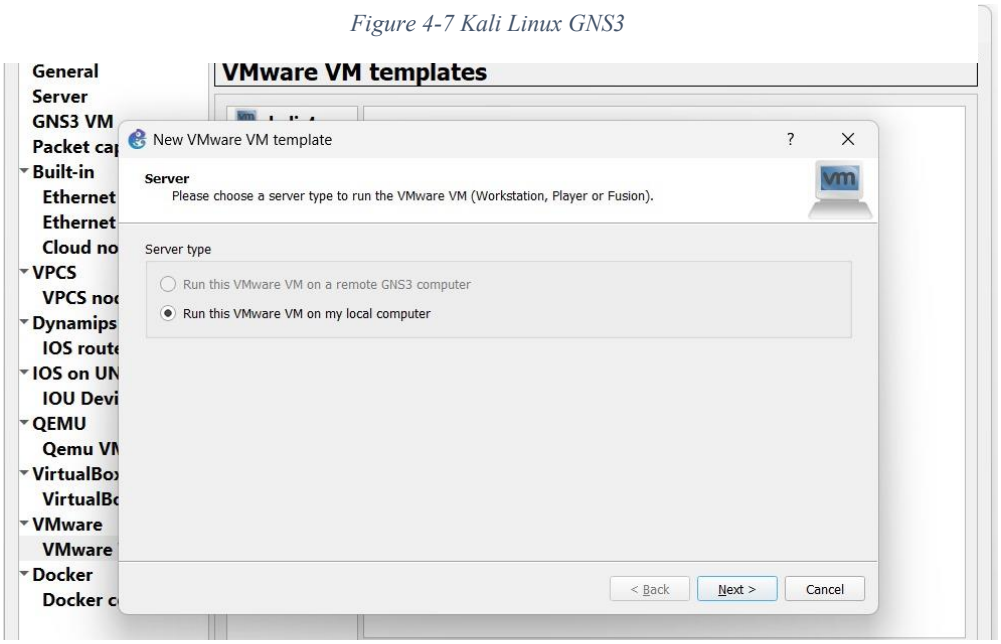


Figure 4-8 Kali Linux install

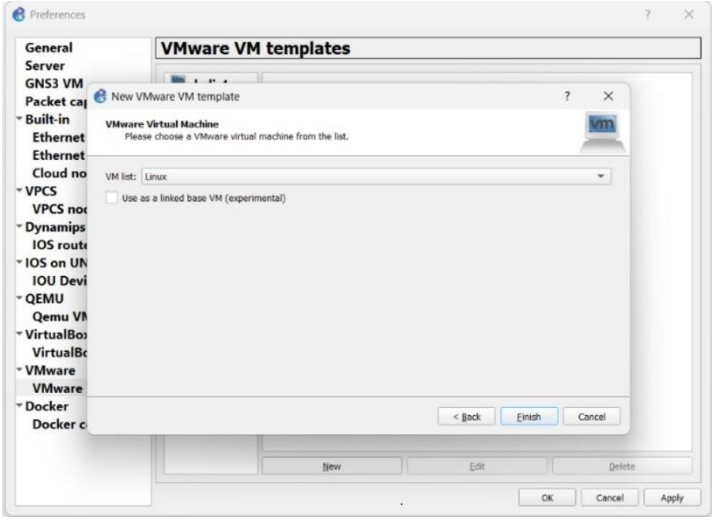


Figure 4-9 Kali Linux

Note: By default, there is only Adapter 1. We will add a second adapter. When connecting any device in the network to the GNS3 VM, make sure to connect it to the interface we configured as Custom when setting up the new adapter

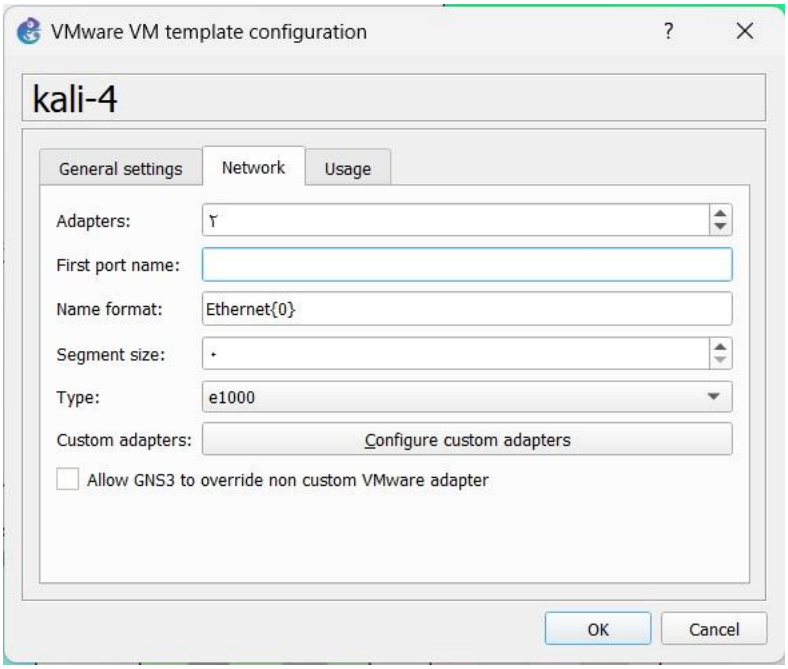


Figure 4-10 Kali Linux configration

4.2 ANSIBLE

Ansible is an open-source automation tool used for configuration management, application deployment, and IT orchestration. It enables users to automate repetitive tasks across multiple servers without requiring manual intervention.

Key Points:

- Agentless: Unlike other automation tools, Ansible does not require agents to be installed on target machines; it uses SSH for communication.
- Declarative Language: Uses YAML-based Playbooks to define automation tasks.
- Idempotent Execution: Ensures tasks are only executed if necessary, avoiding redundant changes.
- Inventory: Manages a list of hosts and groups to target during execution.
- Extensive Integration: Works with cloud providers (AWS, Azure), network devices, containers, and more.

11.1.2 Ansible VS Python In Network Automation

Both Ansible and Python are widely used for network automation, but they serve different purposes and have distinct advantage.

Ansible VS Python In Network Automation

Both Ansible and Python are widely used for network automation, but they serve different purposes and have distinct advantages.

Feature	Ansible	Python
Ease of Use	Uses simple **YAML-based Playbooks**, requiring minimal coding knowledge.	Requires scripting skills, using libraries like **Netmiko, Paramiko, NAPALM.**
Agent Requirement	**Agentless**, communicates over SSH.	No agent required, but Python must be installed on both ends.
Scalability	Designed for managing **multiple network devices** efficiently.	Better suited for **custom automation** on a smaller scale.
Idempotency	Ensures that tasks **only execute changes if needed.**	Requires additional logic to avoid redundant execution.
Modules & Libraries	Built-in **network automation modules** (e.g., `ios_config`, `junos_command`).	Needs external libraries like **Netmiko, NAPALM, Paramiko** for network automation.
Parallel Execution	Can execute tasks **on multiple devices at once.**	Requires **manual threading or multiprocessing** for parallel execution.
Flexibility	Best for structured automation with **predefined tasks.**	More flexible for **custom logic, data processing, and complex automation.**
Use Cases	Configuring routers, switches, firewalls, and cloud infrastructure with **minimal effort.**	Advanced automation, custom parsing, data extraction, and API interactions.

Why Did We Use Ansible?

We chose Ansible for network automation because it provides a simple, scalable, and agentless way to manage network devices efficiently. Since our project is focused on network automation using DevOps tools, Ansible is the ideal starting point.

Key Reasons for Using Ansible:

1.Integration with DevOps Tools:

We plan to incorporate other DevOps tools like Jenkins, Docker, and Kubernetes in the next phase of our project.

Ansible easily integrates with these tools for CI/CD, container orchestration, and infrastructure automation.

2.Scalability & Efficiency:

Ansible allows us to automate multiple network devices at once, making it easier to scale our project.

We can define configurations in YAML-based Playbooks and apply them consistently across devices.

3Agentless & Secure:

Unlike other automation tools, Ansible does not require agents on target devices, reducing complexity.

It uses SSH for Linux-based devices and API or CLI commands for network devices, ensuring secure execution.

4. Future-Proofing the Project:

Since Ansible is widely used in modern DevOps environments, learning and implementing it now prepares us for advanced automation tasks.

It lays the foundation for Infrastructure as Code (IaC), which we may extend with Terraform or Kubernetes in the future.

5. Simplifies Network Automation:

With built-in network modules (e.g., ios_config, junos_command), Ansible makes it easy to configure routers, switches, and firewalls.

It ensures idempotency, meaning configurations are only applied if needed, avoiding redundant changes.

ANSIBLE INSTALLATION

Table 4-1 ANSIBLE INSTALLATION

```
sudo apt update
sudo apt install ansible
```

Automation using Ansibl

Simple syntax (YAML)

Agentless operation

Wide support for network modules

Writing Playbooks

A playbook was created to automate VLAN configuration:

```yaml
all:
  vars:
    ansible_connection: network_cli
    ansible_user: admin
    ansible_password: mma


  children:
    routers:
      hosts:
        R1:
          ansible_host: 192.168.100.1
          ansible_network_os: ios
        R2:
          ansible_host: 192.168.122.122
          ansible_network_os: ios
        R3:
          ansible_host: 192.168.1.6
          ansible_network_os: ios

    switches:
      hosts:
        IOU1:
          ansible_host: 192.168.100.2
          ansible_network_os: ios
        IOU2:
          ansible_host: 192.168.100.6
          ansible_network_os: ios
        IOU3:
          ansible_host: 192.168.10.5
          ansible_network_os: ios
        IOU4:
          ansible_host: 192.168.20.5
          ansible_network_os: ios
        IOU5:
          ansible_host: 192.168.30.5
          ansible_network_os: ios
        IOU6:
          ansible_host: 192.168.40.5
          ansible_network_os: ios
```

```yaml
        IOU8:
          ansible_host: 192.168.50.5
          ansible_network_os: ios
        IOU9:
          ansible_host: 192.168.60.5
          ansible_network_os: ios
        IOU10:
          ansible_host: 192.168.200.2
          ansible_network_os: ios
        IOU11:
          ansible_host: 192.168.200.6
          ansible_network_os: ios
        IOU12:
          ansible_host: 192.168.80.5
          ansible_network_os: ios
        IOU13:
          ansible_host: 192.168.90.5
          ansible_network_os: ios

    pcs:
      hosts:
        PC1:
          ansible_host: 192.168.60.2
        PC2:
          ansible_host: 192.168.80.2

    vms:
      hosts:
        MonitoringVM:
          ansible_host: 192.168.122.10
        SnortVM:
          ansible_host: 192.168.150.10
```

.

.

```yaml
- name: Enable SNMP on devices
  hosts: all
  gather_facts: no
  tasks:
    - name: Set SNMP community string
      ios_config:
        lines:
          - snmp-server community public RO

    - name: Set SNMP contact
      ios_config:
        lines:
          - snmp-server contact Admin

    - name: Set SNMP location
      ios_config:
        lines:
          - snmp-server location GNS3-Lab
```

Figure 4-11 Ansible host

```
┌──(kali㉿kali)-[~/Desktop/ansible]
└─$ ansible-playbook -i hosts.yaml 6-SNMP.yml

PLAY [Enable SNMP on devices] ********************************
*****************
```

Running the Playbooks

```
PLAY RECAP *********************************************************************
*****
IOU1                       : ok=2    changed=2    unreachable=0    failed=0
    skipped=0    rescued=0    ignored=0
IOU10                      : ok=2    changed=1    unreachable=0    failed=0
    skipped=0    rescued=0    ignored=0
IOU11                      : ok=2    changed=1    unreachable=0    failed=0
    skipped=0    rescued=0    ignored=0
IOU12                      : ok=2    changed=1    unreachable=0    failed=0
    skipped=0    rescued=0    ignored=0
IOU13                      : ok=2    changed=1    unreachable=0    failed=0
    skipped=0    rescued=0    ignored=0
IOU2                       : ok=2    changed=1    unreachable=0    failed=0
    skipped=0    rescued=0    ignored=0
IOU3                       : ok=2    changed=1    unreachable=0    failed=0
    skipped=0    rescued=0    ignored=0
IOU4                       : ok=2    changed=1    unreachable=0    failed=0
    skipped=0    rescued=0    ignored=0
IOU5                       : ok=2    changed=1    unreachable=0    failed=0
    skipped=0    rescued=0    ignored=0
IOU6                       : ok=2    changed=1    unreachable=0    failed=0
    skipped=0    rescued=0    ignored=0
IOU8                       : ok=2    changed=1    unreachable=0    failed=0
    skipped=0    rescued=0    ignored=0
IOU9                       : ok=2    changed=1    unreachable=0    failed=0
    skipped=0    rescued=0    ignored=0
R1                         : ok=2    changed=1    unreachable=0    failed=0
    skipped=0    rescued=0    ignored=0
R2                         : ok=2    changed=1    unreachable=0    failed=0
    skipped=0    rescued=0    ignored=0
R3                         : ok=2    changed=1    unreachable=0    failed=0
    skipped=0    rescued=0    ignored=0
```

Figure 4-12 Ansible outcome

4.3 PROMETHEUS

Installing Prometheus

Installed on Kali Linux virtual machine.

Used apt package manager to install both tools:

Installing Prometheus 2-4 Table

cd /tmp
Move to temporary directory -
wget
https://github.com/prometheus/prometheus/releases/download/v2.52.0/prometheus-2.52.0.linux-amd64.tar.gz
Download Prometheus -
tar xvf prometheus-2.52.0.linux-amd64.tar.gz
Extract the archive -
sudo mv prometheus-2.52.0.linux-amd64 /opt/prometheus
Move Prometheus to a system directory -
/sudo cp /opt/prometheus/prometheus /usr/local/bin
Copy binary -
/sudo cp /opt/prometheus/promtool /usr/local/bin
Copy promtool binary -
sudo mkdir -p /etc/prometheus /var/lib/prometheus
Create config and data directories -
/sudo cp -r /opt/prometheus/consoles /opt/prometheus/console_libraries /etc/prometheus
Copy templates -

```
sudo cp /opt/prometheus/prometheus.yml /etc/prometheus/
```

- Copy default config

Prometheus Configuration

Prometheus was configured to scrape metrics from devices.

```
┌──(kali㉿kali)-[~/Desktop]
└─$ cd /etc/prometheus

┌──(kali㉿kali)-[/etc/prometheus]
└─$ ls
alert_rules.yml  prometheus.yml

┌──(kali㉿kali)-[/etc/prometheus]
└─$ sudo vi prometheus.yml
```

Figure 4-13 Prometheus Configuration

Targets included Linux machines and routers using node exporters and SNMP.

.

```
# my global config
global:
  scrape_interval: 15s
  evaluation_interval: 15s

# Alertmanager configuration
alerting:
  alertmanagers:
    - static_configs:
        - targets:
            - 'localhost:9093'

# Load rules once and periodically evaluate them according to the global 'evaluation_interval'.
rule_files:
  - "alert_rules.yml"
```

.

```
scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries scraped from this config.
  - job_name: "prometheus"
    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.
    static_configs:
      - targets: ["localhost:9090"]
  - job_name: 'snmp'
    static_configs:
      - targets:
        - '192.168.122.122'
        - '192.168.100.1'
        - '192.168.1.6'
        - '192.168.100.2'
        - '192.168.100.6'
        - '192.168.200.2'
        - '192.168.200.6'
        - '192.168.10.5'
        - '192.168.20.5'
        - '192.168.30.5'
        - '192.168.40.5'
        - '192.168.50.5'
        - '192.168.60.5'
        - '192.168.80.5'
        - '192.168.90.5'
    metrics_path: /snmp
    params:
      module: [if_mib]
    relabel_configs:
      - source_labels: [__address__]
        target_label: __param_target
      - source_labels: [__param_target]
        target_label: instance
      - target_label: __address__
        replacement: 'localhost:9116'

  - job_name: "node"
    static_configs:
      - targets:
        - '192.168.150.10:9100'
        - '192.168.122.10:9100'

  - job_name: 'loki'
    static_configs:
      - targets: ['192.168.150.10:3100']
```

Figure 4-14 Prometheus Configuration

```
┌──(kali㉿kali)-[/etc/prometheus]
└─$ sudo systemctl restart prometheus
```

.

Node Exporter

Node Exporter is a lightweight, open-source tool developed by the Prometheus project.

Its primary role is to collect hardware and operating system-level metrics from Linux or Unix servers and expose them to Prometheus.

🛠 What Node Exporter Does:

Collects system metrics: CPU, memory, disk I/O, network stats, etc.

Makes metrics available over HTTP /metrics endpoint.

⭐ Key Features:

Lightweight and resource-efficient

Easy to deploy

Highly reliable for server health monitoring

Node Exporter 3-4 Table

cd /tmp- Move to temp directory
wget
https://github.com/prometheus/node_exporter/releases/download/v1.8.1/node_exporter-1.8.1.linux-amd64.tar
gz- Download Node Exporter.
tar -xvf node_exporter-1.8.1.linux-amd64.tar.gz- Extract it
sudo mv node_exporter-1.8.1.linux-amd64/node_exporter /usr/local/bin/- Move binary to system path

```
┌──(kali㉿kali)-[/etc/prometheus]
└─$ sudo systemctl start node_exporter

┌──(kali㉿kali)-[/etc/prometheus]
└─$ sudo systemctl enable node_exporter

┌──(kali㉿kali)-[/etc/prometheus]
└─$ sudo systemctl status node_exporter
● node_exporter.service - Node Exporter
     Loaded: loaded (/etc/systemd/system/node_exporter.service; enabled; preset: disabled)
     Active: active (running) since Sat 2025-04-26 11:43:21 EDT; 26min ago
 Invocation: 56d6425c632b4063a8e6f36681627792
   Main PID: 1033 (node_exporter)
      Tasks: 5 (limit: 2207)
     Memory: 18.4M (peak: 24.4M, swap: 104K, swap peak: 184K)
        CPU: 1.799s
     CGroup: /system.slice/node_exporter.service
             └─1033 /usr/local/bin/node_exporter

Apr 26 11:43:21 kali node_exporter[1033]: time=2025-04-26T15:43:21.650Z level=INFO source=node_exporter.go:141 msg=
Apr 26 11:43:21 kali node_exporter[1033]: time=2025-04-26T15:43:21.650Z level=INFO source=node_exporter.go:141 msg=
Apr 26 11:43:21 kali node_exporter[1033]: time=2025-04-26T15:43:21.650Z level=INFO source=node_exporter.go:141 msg=
Apr 26 11:43:21 kali node_exporter[1033]: time=2025-04-26T15:43:21.650Z level=INFO source=node_exporter.go:141 msg=
Apr 26 11:43:21 kali node_exporter[1033]: time=2025-04-26T15:43:21.650Z level=INFO source=node_exporter.go:141 msg=
Apr 26 11:43:21 kali node_exporter[1033]: time=2025-04-26T15:43:21.650Z level=INFO source=node_exporter.go:141 msg=
Apr 26 11:43:21 kali node_exporter[1033]: time=2025-04-26T15:43:21.650Z level=INFO source=node_exporter.go:141 msg=
Apr 26 11:43:21 kali node_exporter[1033]: time=2025-04-26T15:43:21.650Z level=INFO source=node_exporter.go:141 msg=
Apr 26 11:43:21 kali node_exporter[1033]: time=2025-04-26T15:43:21.656Z level=INFO source=tls_config.go:347 msg="Li
Apr 26 11:43:21 kali node_exporter[1033]: time=2025-04-26T15:43:21.656Z level=INFO source=tls_config.go:350 msg="TL
```

Figure 4-15 Node Exporter

snmp exporter

SNMP Exporter is used to collect metrics from devices that expose information via SNMP (Simple Network Management Protocol).

🛠 What SNMP Exporter Does:

Converts SNMP data into Prometheus-readable metrics.

Allows monitoring of network devices (routers, switches, printers, etc.).

⭐ Key Features:

Supports various MIBs (Management Information Bases)

Essential for integrating legacy network hardware into modern monitoring systems

Highly customizable via configuration files

```
┌──(kali㉿kali)-[~/Desktop/snmp]
└─$ ls
snmp_exporter-0.28.0.linux-386  snmp_exporter-0.28.0.linux-386.tar.gz

┌──(kali㉿kali)-[~/Desktop/snmp]
└─$ cd snmp_exporter-0.28.0.linux-386

┌──(kali㉿kali)-[~/Desktop/snmp/snmp_exporter-0.28.0.linux-386]
└─$ ls
LICENSE  NOTICE  snmp_exporter  snmp.yml

┌──(kali㉿kali)-[~/Desktop/snmp/snmp_exporter-0.28.0.linux-386]
└─$ ./snmp_exporter
time=2025-04-26T16:45:50.738Z level=INFO source=main.go:211 msg="Starting snmp_exporter" version="(version=0.28.0, branch=HEAD, revision=9f9547f664b61449281ff888fe690c68e
f94c9f2)" concurrency=0x9d920c4 debug_snmp=0x9d920c8
time=2025-04-26T16:45:50.738Z level=INFO source=main.go:212 msg="operational information" build_context="(go=go1.23.6, platform=linux/386, user=root@cff6a587972e, date=20
250206-23:59:56, tags=unknown)"
time=2025-04-26T16:45:50.854Z level=INFO source=tls_config.go:347 msg="Listening on" address=[::]:9116
time=2025-04-26T16:45:50.854Z level=INFO source=tls_config.go:350 msg="TLS is disabled." http2=false address=[::]:9116
```

. Figure 4-16 snmp exporter

4.4 SNORT

Snort is a free and open-source network intrusion detection system (IDS) and intrusion prevention system (IPS).

🛠 What Snort Does:

Monitors network traffic in real-time.

Performs packet logging and analysis.

Detects and prevents a wide variety of attacks (e.g., DoS, buffer overflows, port scans).

⭐ Key Features:

Real-time traffic analysis and packet logging

Highly customizable detection rules

Capable of running in IDS mode (detection only) or IPS mode (prevention)

Snort 4-4 Table

cd /tmp - Move to temp directory wget https://www.snort.org/downloads/snort/snort3_3.1.65.0-1_amd64.deb - Download Snort 3 .deb package

```
sudo dpkg -i snort3_3.1.65.0-1_amd64.deb

- Install the package

sudo apt --fix-broken install

- Fix any dependency issues

snort -V

- Verify Snort versio
```

.

4.5 LOKI

Loki is a horizontally scalable, highly available, multi-tenant log aggregation system inspired by Prometheus.

🛠 What Loki Does:

Stores and indexes logs efficiently without heavy indexing.

Works alongside Prometheus to link metrics and logs together.

⭐ Key Features:

Highly cost-efficient compared to traditional logging systems (e.g., ELK Stack).

Native integration with Grafana for easy log visualization.

Designed for scalable, distributed environments.

Loki 5-4 Table

```
cd /tmp

- Move to temp directory

wget https://github.com/grafana/loki/releases/download/v2.9.4/loki-linux-amd64.zip

- Download Loki binary
```

unzip loki-linux-amd64.zip

- Extract the zip

chmod +x loki-linux-amd64

- Make binary executable

sudo mv loki-linux-amd64 /usr/local/bin/loki

- Move it to system path

wget https://raw.githubusercontent.com/grafana/loki/v2.9.4/cmd/loki/loki-local-config.yaml

- Download config file

sudo mv loki-local-config.yaml /etc/loki-config.yaml

- Move config file

.

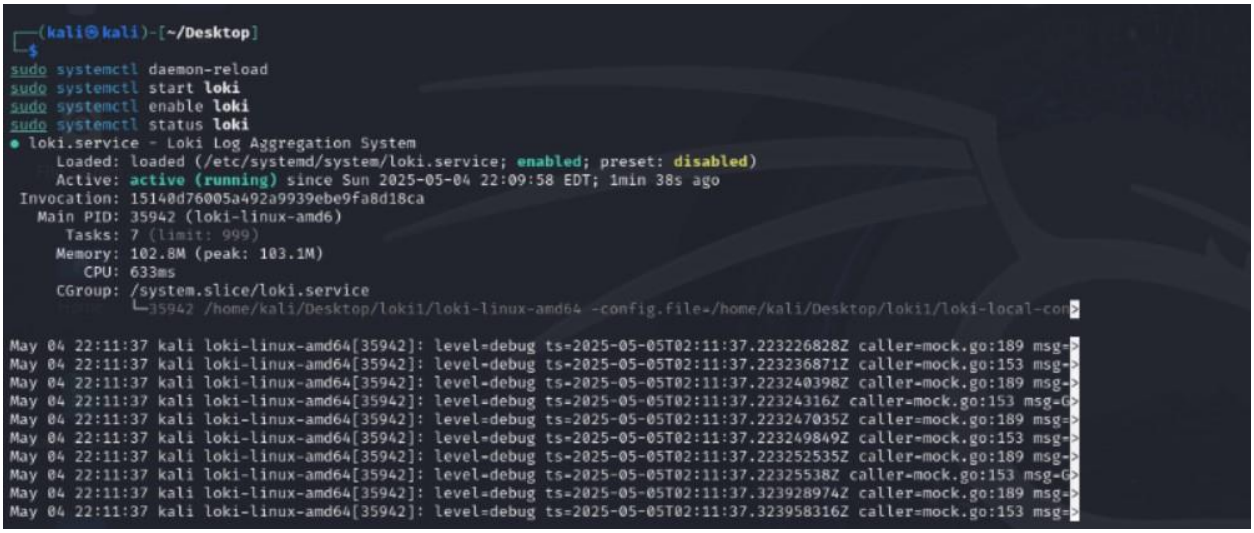


Figure 4-17 Loki

. .

4.6 PROMTAIL

Promtail is an agent that ships the contents of local logs to Loki.

🛠 What Promtail Does:

Reads log files from servers.

Labels and transforms log entries.

Forwards them to a Loki server for storage and querying.

⭐ Key Features:

Integrates tightly with Loki and Kubernetes

Lightweight and simple

Supports systemd journal logs and plain log files

Promtail6-4 Table

```
cd /tmp

- Move to temp directory

wget https://github.com/grafana/loki/releases/download/v2.9.4/promtail-linux-amd64.zip

- Download Promtail

unzip promtail-linux-amd64.zip

- Extract zip

chmod +x promtail-linux-amd64

- Make it executable

sudo mv promtail-linux-amd64 /usr/local/bin/promtail

- Move to system path

wget https://raw.githubusercontent.com/grafana/loki/v2.9.4/clients/cmd/promtail/promtail-local-config.yaml
```

- Download config

sudo mv promtail-local-config.yaml /etc/promtail-config.yaml

- Move config file

.

```
┌──(kali㉿kali)-[~/Desktop/promtail]
└─$ sudo systemctl daemon-reload
sudo systemctl start promtail
sudo systemctl enable promtail
sudo systemctl status promtail

Created symlink '/etc/systemd/system/multi-user.target.wants/promtail.service' → '/etc/systemd/system/promt
ail.service'.
● promtail.service - Promtail Log Collector for Loki
     Loaded: loaded (/etc/systemd/system/promtail.service; enabled; preset: disabled)
     Active: active (running) since Sun 2025-05-04 22:21:39 EDT; 430ms ago
 Invocation: b8ea9c237a4942c9939a7debcad1a8f2
   Main PID: 42588 (promtail-linux-)
      Tasks: 6 (limit: 999)
     Memory: 17.6M (peak: 17.9M)
        CPU: 47ms
     CGroup: /system.slice/promtail.service
             └─42588 /home/kali/Desktop/promtail/promtail-linux-amd64 -config.file=/home/kali/Desktop/prom>

May 04 22:21:39 kali systemd[1]: Started promtail.service - Promtail Log Collector for Loki.
May 04 22:21:39 kali promtail-linux-amd64[42588]: level=info ts=2025-05-05T02:21:39.203122342Z caller=promt>
May 04 22:21:39 kali promtail-linux-amd64[42588]: level=info ts=2025-05-05T02:21:39.204105857Z caller=serv>
May 04 22:21:39 kali promtail-linux-amd64[42588]: level=info ts=2025-05-05T02:21:39.204198902Z caller=main>
May 04 22:21:39 kali promtail-linux-amd64[42588]: level=warn ts=2025-05-05T02:21:39.205122285Z caller=prom>
```

Figure 4-18 Promtail

.

4.7 GRAFANA DASHBOARD SETUP

Grafana connected to Prometheus as a data source.

Dashboards were created to visualize:

CPU usage

Memory consumption

Interface traffic

Custom alerts were added to trigger based on specific thresholds.

GRAFANA DASHBOARD 7-4 Table

cd /tmp - Move to temp directory wget https://dl.grafana.com/oss/release/grafana_11.0.0_amd64.deb - Download Grafana .deb package sudo dpkg -i grafana_11.0.0_amd64.deb - Install the .deb package sudo apt-get install -f - Fix dependencies if needed

Figure 4-19 GRAFANA DASHBOARD SETUP

4.8 ALERTS MANAGER

This Part explains how alerts were automated using Alert Manager.

ALERTS MANAGER 8-4 Table

cd /tmp - Move to temp directory wget

https://github.com/prometheus/alertmanager/releases/download/v0.27.0/alertmanager-0.27.0.linux-amd64.tar

.gz

- Download Alertmanager

tar -xvf alertmanager-0.27.0.linux-amd64.tar.gz

- Extract it

sudo mv alertmanager-0.27.0.linux-amd64 /opt/alertmanager

- Move files to system path

sudo cp /opt/alertmanager/alertmanager /usr/local/bin/

- Copy Alertmanager binary

sudo cp /opt/alertmanager/amtool /usr/local/bin/

- Copy amtool binary

sudo cp /opt/alertmanager/alertmanager.yml /etc/alertmanager/

- Copy config file

.

```
┌──(kali㉿kali)-[~/Desktop/alertt/alertmanager-0.28.1.linux-amd64]
└─$ sudo systemctl daemon-reload
sudo systemctl start alertmanager
sudo systemctl enable alertmanager

┌──(kali㉿kali)-[~/Desktop/alertt/alertmanager-0.28.1.linux-amd64]
└─$ sudo systemctl status alertmanager.service
● alertmanager.service - Alert Manager
     Loaded: loaded (/etc/systemd/system/alertmanager.service; enabled; preset: disabled)
     Active: active (running) since Sun 2025-06-22 02:40:40 EDT; 6s ago
 Invocation: 12b2bca4a1154a98993282085f45af90
   Main PID: 15324 (alertmanager)
      Tasks: 10 (limit: 2207)
     Memory: 15.3M (peak: 15.7M)
        CPU: 212ms
     CGroup: /system.slice/alertmanager.service
             └─15324 /usr/local/bin/alertmanager --config.file=/etc/alertmanager/alertmanager.yml --storage.path=/v>

Jun 22 02:40:40 kali systemd[1]: Started alertmanager.service - Alert Manager.
Jun 22 02:40:40 kali alertmanager[15324]: time=2025-06-22T06:40:40.454Z level=INFO source=main.go:191 msg="Starting>
Jun 22 02:40:40 kali alertmanager[15324]: time=2025-06-22T06:40:40.454Z level=INFO source=main.go:192 msg="Build co>
Jun 22 02:40:40 kali alertmanager[15324]: time=2025-06-22T06:40:40.465Z level=INFO source=cluster.go:185 msg="setti>
Jun 22 02:40:40 kali alertmanager[15324]: time=2025-06-22T06:40:40.469Z level=INFO source=cluster.go:674 msg="Waiti>
Jun 22 02:40:40 kali alertmanager[15324]: time=2025-06-22T06:40:40.559Z level=INFO source=coordinator.go:112 msg="L>
Jun 22 02:40:40 kali alertmanager[15324]: time=2025-06-22T06:40:40.560Z level=INFO source=coordinator.go:125 msg="C>
Jun 22 02:40:40 kali alertmanager[15324]: time=2025-06-22T06:40:40.569Z level=INFO source=tls_config.go:347 msg="Li>
Jun 22 02:40:40 kali alertmanager[15324]: time=2025-06-22T06:40:40.569Z level=INFO source=tls_config.go:350 msg="TL>
Jun 22 02:40:42 kali alertmanager[15324]: time=2025-06-22T06:40:42.470Z level=INFO source=cluster.go:699 msg="gossi>
lines 1-21/21 (END)
```

Figure 4-20 ALERTS MANAGER

.

```yaml
groups:
  - name: node_alerts
    rules:

      - alert: InstanceDown
        expr: up == 0
        for: 1m
        labels:
          severity: critical
        annotations:
          summary: "Instance {{ $labels.instance }} down"
          description: "{{ $labels.instance }} has been down for more than 1 minute."

      - alert: HighCPUUsage
        expr: rate(node_cpu_seconds_total{mode="user"}[1m]) > 0.7
        for: 2m
        labels:
          severity: warning
        annotations:
          summary: "High CPU Usage on {{ $labels.instance }}"
          description: "CPU usage is above 70% for more than 2 minutes."

      # New Snort alerts
      - alert: SSHAlert
        expr: snort_alerts{msg="SSH alert"} == 1
        for: 1m
        labels:
          severity: critical
        annotations:
          summary: "SSH attack detected"
          description: "An SSH alert was triggered by Snort for SSH attempts."

      - alert: TelnetAlert
        expr: snort_alerts{msg="Telnet alert"} == 1
        for: 1m
        labels:
          severity: critical
        annotations:
          summary: "Telnet attack detected"
          description: "A Telnet alert was triggered by Snort for Telnet attempts."
```

Figure 4-21 Node and Snort Alerta

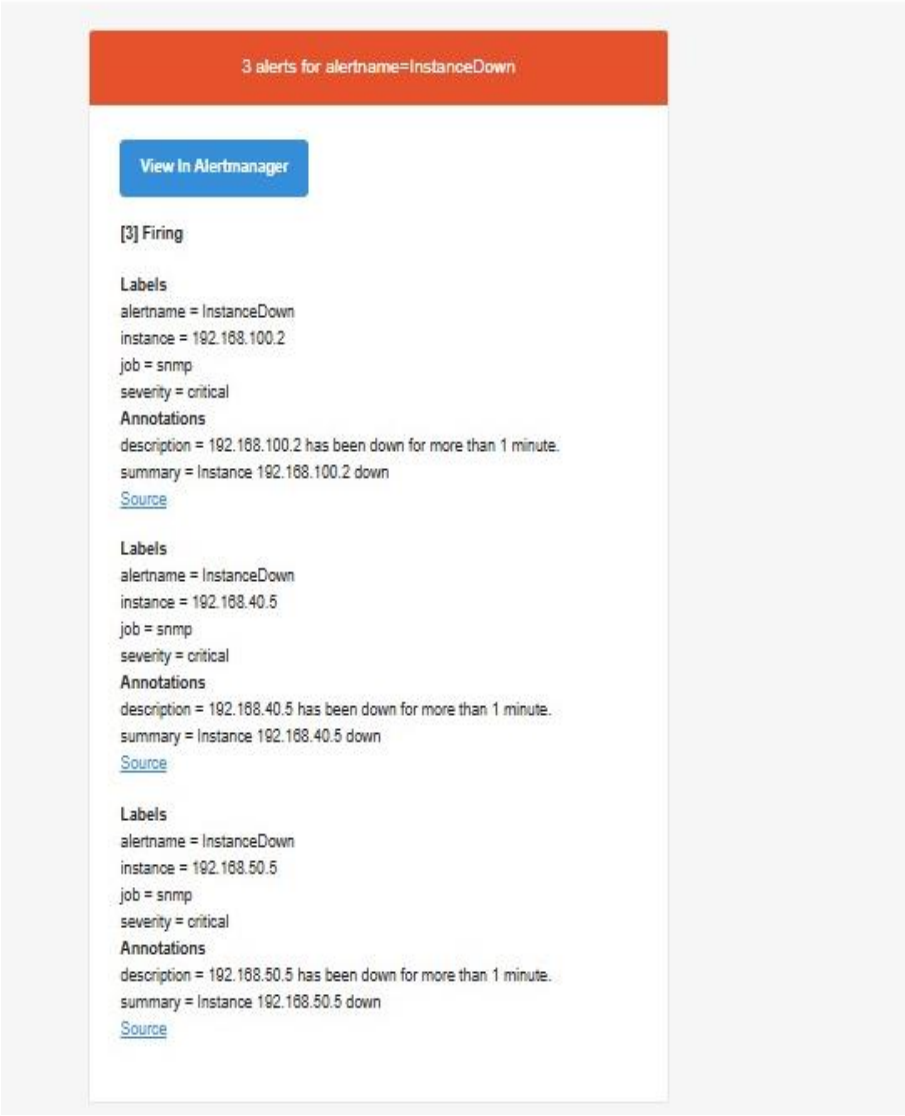


Figure 4-22 ALERTS MANAGER

4.9 TESTING AND RESULTS

This part documents the tests conducted to ensure functionality.

4.9.1 Test connectivity between devices

Test DHCP in pc 1 AND pc2

```
PC1> dhcp
DDORA IP 192.168.10.4/24 GW 192.168.10.1

PC1>
```

Figure 4-23 Test DHCP in pc1

```
Executing the startup file

PC2> dhcp
DDORA IP 192.168.20.4/24 GW 192.168.20.1
```

Figure 4-24 Test DHCP in pc2

Ping PC1 VLAN 10 VRRP (192.168.10.1) & Dist-1 (192.168.10.2) in dist 1&2

```
PC1>
PC1> ping 192.168.10.1
84 bytes from 192.168.10.1 icmp_seq=1 ttl=255 time=1.408 ms
84 bytes from 192.168.10.1 icmp_seq=2 ttl=255 time=1.329 ms
84 bytes from 192.168.10.1 icmp_seq=3 ttl=255 time=1.454 ms
84 bytes from 192.168.10.1 icmp_seq=4 ttl=255 time=1.539 ms
84 bytes from 192.168.10.1 icmp_seq=5 ttl=255 time=1.569 ms

PC1> ping 192.168.10.2
84 bytes from 192.168.10.2 icmp_seq=1 ttl=255 time=1.333 ms
84 bytes from 192.168.10.2 icmp_seq=2 ttl=255 time=1.724 ms
84 bytes from 192.168.10.2 icmp_seq=3 ttl=255 time=1.608 ms
84 bytes from 192.168.10.2 icmp_seq=4 ttl=255 time=1.946 ms
```

Figure 4-25 Ping PC1 VLAN 10 VRRP

.

.

```
PC1> ping 192.168.100.6
84 bytes from 192.168.100.6 icmp_seq=1 ttl=255 time=1.405 ms
84 bytes from 192.168.100.6 icmp_seq=2 ttl=255 time=1.242 ms
84 bytes from 192.168.100.6 icmp_seq=3 ttl=255 time=2.964 ms
84 bytes from 192.168.100.6 icmp_seq=4 ttl=255 time=1.585 ms
84 bytes from 192.168.100.6 icmp_seq=5 ttl=255 time=1.550 ms

PC1>
```

Ping PC1 (VLAN 10) to VLAN 20

```
PC1> dhcp
DDORA IP 192.168.10.4/24 GW 192.168.10.1

PC1> ping 192.168.20.4
192.168.20.4 icmp_seq=1 timeout
192.168.20.4 icmp_seq=2 timeout
84 bytes from 192.168.20.4 icmp_seq=3 ttl=63 time=2.814 ms
84 bytes from 192.168.20.4 icmp_seq=4 ttl=63 time=2.713 ms
84 bytes from 192.168.20.4 icmp_seq=5 ttl=63 time=2.750 ms

PC1> ping 192.168.20.4
84 bytes from 192.168.20.4 icmp_seq=1 ttl=63 time=2.720 ms
84 bytes from 192.168.20.4 icmp_seq=2 ttl=63 time=2.404 ms
84 bytes from 192.168.20.4 icmp_seq=3 ttl=63 time=2.211 ms
84 bytes from 192.168.20.4 icmp_seq=4 ttl=63 time=2.529 ms
84 bytes from 192.168.20.4 icmp_seq=5 ttl=63 time=2.683 ms

PC1>
```

Figure 4-26 Ping PC1 (VLAN 10) to VLAN 20

Show ip route command in R3

```
Gateway of last resort is not set

      192.168.1.0/24 is variably subnetted, 3 subnets, 2 masks
R        192.168.1.0/30 [120/1] via 192.168.1.5, 00:00:02, FastEthernet0/0
C        192.168.1.4/30 is directly connected, FastEthernet0/0
L        192.168.1.6/32 is directly connected, FastEthernet0/0
R     192.168.10.0/24 [120/2] via 192.168.1.5, 00:00:02, FastEthernet0/0
R     192.168.20.0/24 [120/2] via 192.168.1.5, 00:00:02, FastEthernet0/0
R     192.168.30.0/24 [120/2] via 192.168.1.5, 00:00:02, FastEthernet0/0
R     192.168.40.0/24 [120/2] via 192.168.1.5, 00:00:02, FastEthernet0/0
R     192.168.50.0/24 [120/2] via 192.168.1.5, 00:00:02, FastEthernet0/0
R     192.168.60.0/24 [120/2] via 192.168.1.5, 00:00:02, FastEthernet0/0
R     192.168.80.0/24 [120/1] via 192.168.200.6, 00:00:19, FastEthernet1/1
                      [120/1] via 192.168.200.2, 00:00:18, FastEthernet1/0
R     192.168.90.0/24 [120/1] via 192.168.200.6, 00:00:19, FastEthernet1/1
                      [120/1] via 192.168.200.2, 00:00:18, FastEthernet1/0
R     192.168.100.0/24 [120/1] via 192.168.1.5, 00:00:02, FastEthernet0/0
R     192.168.122.0/24 [120/2] via 192.168.1.5, 00:00:02, FastEthernet0/0
      192.168.200.0/24 is variably subnetted, 4 subnets, 2 masks
C        192.168.200.0/30 is directly connected, FastEthernet1/0
L        192.168.200.1/32 is directly connected, FastEthernet1/0
C        192.168.200.4/30 is directly connected, FastEthernet1/1
L        192.168.200.5/32 is directly connected, FastEthernet1/1
```

Figure 4-27 Show ip route command in R3

Show ip route command Dist 1

```
IOU1   IOU2   PC1   PC2   IOU3

Gateway of last resort is 192.168.100.1 to network 0.0.0.0

S*    0.0.0.0/0 [1/0] via 192.168.100.1
R     192.168.1.0/24 [120/1] via 192.168.100.1, 00:00:40, Ethernet0/0
      192.168.10.0/24 is variably subnetted, 2 subnets, 2 masks
C        192.168.10.0/24 is directly connected, Vlan10
L        192.168.10.2/32 is directly connected, Vlan10
      192.168.20.0/24 is variably subnetted, 2 subnets, 2 masks
C        192.168.20.0/24 is directly connected, Vlan20
L        192.168.20.2/32 is directly connected, Vlan20
      192.168.30.0/24 is variably subnetted, 2 subnets, 2 masks
C        192.168.30.0/24 is directly connected, Vlan30
L        192.168.30.2/32 is directly connected, Vlan30
      192.168.40.0/24 is variably subnetted, 2 subnets, 2 masks
C        192.168.40.0/24 is directly connected, Vlan40
L        192.168.40.2/32 is directly connected, Vlan40
      192.168.50.0/24 is variably subnetted, 2 subnets, 2 masks
C        192.168.50.0/24 is directly connected, Vlan50
L        192.168.50.2/32 is directly connected, Vlan50
      192.168.60.0/24 is variably subnetted, 2 subnets, 2 masks
C        192.168.60.0/24 is directly connected, Vlan60
L        192.168.60.2/32 is directly connected, Vlan60
R     192.168.80.0/24 [120/3] via 192.168.100.1, 00:00:40, Ethernet0/0
R     192.168.90.0/24 [120/3] via 192.168.100.1, 00:00:40, Ethernet0/0
      192.168.100.0/24 is variably subnetted, 4 subnets, 3 masks
R        192.168.100.0/24 [120/1] via 192.168.60.3, 00:00:25, Vlan60
                          [120/1] via 192.168.50.3, 00:00:22, Vlan50
                          [120/1] via 192.168.40.3, 00:00:01, Vlan40
                          [120/1] via 192.168.30.3, 00:00:23, Vlan30
C        192.168.100.0/30 is directly connected, Ethernet0/0
L        192.168.100.2/32 is directly connected, Ethernet0/0
R        192.168.100.4/30 [120/1] via 192.168.100.1, 00:00:40, Ethernet0/0
R     192.168.122.0/24 [120/2] via 192.168.100.1, 00:00:40, Ethernet0/0
R     192.168.200.0/24 [120/2] via 192.168.100.1, 00:00:40, Ethernet0/0
MLS1#
MLS1#
```

Figure 4-28 Show ip route command Dist 1

Show ip route command core 1 (R1)

```
Gateway of last resort is not set

      192.168.1.0/24 is variably subnetted, 4 subnets, 2 masks
C        192.168.1.0/30 is directly connected, FastEthernet1/1
L        192.168.1.1/32 is directly connected, FastEthernet1/1
C        192.168.1.4/30 is directly connected, FastEthernet2/0
L        192.168.1.5/32 is directly connected, FastEthernet2/0
R     192.168.10.0/24 [120/1] via 192.168.100.6, 00:00:09, FastEthernet1/0
                      [120/1] via 192.168.100.2, 00:00:06, FastEthernet0/0
R     192.168.20.0/24 [120/1] via 192.168.100.6, 00:00:09, FastEthernet1/0
                      [120/1] via 192.168.100.2, 00:00:06, FastEthernet0/0
R     192.168.30.0/24 [120/1] via 192.168.100.6, 00:00:09, FastEthernet1/0
                      [120/1] via 192.168.100.2, 00:00:06, FastEthernet0/0
R     192.168.40.0/24 [120/1] via 192.168.100.6, 00:00:09, FastEthernet1/0
                      [120/1] via 192.168.100.2, 00:00:06, FastEthernet0/0
R     192.168.50.0/24 [120/1] via 192.168.100.6, 00:00:09, FastEthernet1/0
                      [120/1] via 192.168.100.2, 00:00:06, FastEthernet0/0
R     192.168.60.0/24 [120/1] via 192.168.100.6, 00:00:09, FastEthernet1/0
                      [120/1] via 192.168.100.2, 00:00:06, FastEthernet0/0
R     192.168.80.0/24 [120/2] via 192.168.1.6, 00:00:01, FastEthernet2/0
R     192.168.90.0/24 [120/2] via 192.168.1.6, 00:00:01, FastEthernet2/0
      192.168.100.0/24 is variably subnetted, 5 subnets, 3 masks
R        192.168.100.0/24 [120/2] via 192.168.100.6, 00:00:09, FastEthernet1/0
                          [120/2] via 192.168.100.2, 00:00:06, FastEthernet0/0
C        192.168.100.0/30 is directly connected, FastEthernet0/0
L        192.168.100.1/32 is directly connected, FastEthernet0/0
C        192.168.100.4/30 is directly connected, FastEthernet1/0
L        192.168.100.5/32 is directly connected, FastEthernet1/0
R     192.168.122.0/24 [120/1] via 192.168.1.2, 00:00:26, FastEthernet1/1
R     192.168.200.0/24 [120/1] via 192.168.1.6, 00:00:01, FastEthernet2/0
R1core-1#
R1core-1#
R1core-1#
R1core-1#
R1core-1#
R1core-1#
```

Figure 4-29 Show ip route command core 1 (R1)

Show ip route command dist 2

```
IOU1   IOU2   PC1   PC2   IOU3

Gateway of last resort is 192.168.100.5 to network 0.0.0.0

S*    0.0.0.0/0 [1/0] via 192.168.100.5
R     192.168.1.0/24 [120/1] via 192.168.100.5, 00:00:04, Ethernet0/0
      192.168.10.0/24 is variably subnetted, 2 subnets, 2 masks
C        192.168.10.0/24 is directly connected, Vlan10
L        192.168.10.3/32 is directly connected, Vlan10
      192.168.20.0/24 is variably subnetted, 2 subnets, 2 masks
C        192.168.20.0/24 is directly connected, Vlan20
L        192.168.20.3/32 is directly connected, Vlan20
      192.168.30.0/24 is variably subnetted, 2 subnets, 2 masks
C        192.168.30.0/24 is directly connected, Vlan30
L        192.168.30.3/32 is directly connected, Vlan30
      192.168.40.0/24 is variably subnetted, 2 subnets, 2 masks
C        192.168.40.0/24 is directly connected, Vlan40
L        192.168.40.3/32 is directly connected, Vlan40
      192.168.50.0/24 is variably subnetted, 2 subnets, 2 masks
C        192.168.50.0/24 is directly connected, Vlan50
L        192.168.50.3/32 is directly connected, Vlan50
      192.168.60.0/24 is variably subnetted, 2 subnets, 2 masks
C        192.168.60.0/24 is directly connected, Vlan60
L        192.168.60.3/32 is directly connected, Vlan60
R     192.168.80.0/24 [120/3] via 192.168.100.5, 00:00:04, Ethernet0/0
R     192.168.90.0/24 [120/3] via 192.168.100.5, 00:00:04, Ethernet0/0
      192.168.100.0/24 is variably subnetted, 4 subnets, 3 masks
R        192.168.100.0/24 [120/1] via 192.168.50.2, 00:00:00, Vlan50
                          [120/1] via 192.168.30.2, 00:00:11, Vlan30
                          [120/1] via 192.168.20.2, 00:00:05, Vlan20
                          [120/1] via 192.168.10.2, 00:00:28, Vlan10
R        192.168.100.0/30 [120/1] via 192.168.100.5, 00:00:04, Ethernet0/0
C        192.168.100.4/30 is directly connected, Ethernet0/0
L        192.168.100.6/32 is directly connected, Ethernet0/0
R     192.168.122.0/24 [120/2] via 192.168.100.5, 00:00:04, Ethernet0/0
R     192.168.200.0/24 [120/2] via 192.168.100.5, 00:00:04, Ethernet0/0
MLS2#
```

Figure 4-30 Show ip route command dist 2

Test connectivity from Pc1 to Dist 3 & DIst 4 in another campus

```
PC1>
PC1>
PC1>
PC1>
PC1>
PC1> ping 192.168.200.2
84 bytes from 192.168.200.2 icmp_seq=1 ttl=252 time=62.657 ms
84 bytes from 192.168.200.2 icmp_seq=2 ttl=252 time=63.136 ms
84 bytes from 192.168.200.2 icmp_seq=3 ttl=252 time=61.673 ms
84 bytes from 192.168.200.2 icmp_seq=4 ttl=252 time=62.649 ms
84 bytes from 192.168.200.2 icmp_seq=5 ttl=252 time=63.288 ms
```

Figure 4-32 Test connectivity from Pc1 to Dist 3

```
PC1> ping 192.168.200.6
84 bytes from 192.168.200.6 icmp_seq=1 ttl=252 time=62.482 ms
84 bytes from 192.168.200.6 icmp_seq=2 ttl=252 time=62.265 ms
84 bytes from 192.168.200.6 icmp_seq=3 ttl=252 time=62.475 ms
84 bytes from 192.168.200.6 icmp_seq=4 ttl=252 time=62.817 ms
84 bytes from 192.168.200.6 icmp_seq=5 ttl=252 time=62.555 ms
```

Figure 4-31 Test connectivity from Pc1 to Dist 4

Test connectivity from Pc1 to vrrp of (vlan 80 & vlan 90) in another campus

```
PC1> ping 192.168.80.1
192.168.80.1 icmp_seq=1 timeout
84 bytes from 192.168.80.1 icmp_seq=2 ttl=252 time=62.909 ms
84 bytes from 192.168.80.1 icmp_seq=3 ttl=252 time=63.092 ms
84 bytes from 192.168.80.1 icmp_seq=4 ttl=252 time=62.790 ms
84 bytes from 192.168.80.1 icmp_seq=5 ttl=252 time=62.546 ms

PC1> ping 192.168.90.1
84 bytes from 192.168.90.1 icmp_seq=1 ttl=252 time=62.594 ms
84 bytes from 192.168.90.1 icmp_seq=2 ttl=252 time=63.379 ms
84 bytes from 192.168.90.1 icmp_seq=3 ttl=252 time=62.584 ms
84 bytes from 192.168.90.1 icmp_seq=4 ttl=252 time=62.363 ms
84 bytes from 192.168.90.1 icmp_seq=5 ttl=252 time=63.894 ms

PC1>
```

Test connectivity from Pc1 to (vlan 80 & vlan 90) in another campus

```
PC1> ping 192.168.80.4
84 bytes from 192.168.80.4 icmp_seq=1 ttl=60 time=62.388 ms
84 bytes from 192.168.80.4 icmp_seq=2 ttl=60 time=62.620 ms
84 bytes from 192.168.80.4 icmp_seq=3 ttl=60 time=63.187 ms
84 bytes from 192.168.80.4 icmp_seq=4 ttl=60 time=62.436 ms
84 bytes from 192.168.80.4 icmp_seq=5 ttl=60 time=62.646 ms
```

Figure 4-33Test connectivity from Pc1 to vrrp in another campus

.

```
PC1> ping 192.168.90.4
84 bytes from 192.168.90.4 icmp_seq=1 ttl=60 time=62.605 ms
84 bytes from 192.168.90.4 icmp_seq=2 ttl=60 time=62.442 ms
84 bytes from 192.168.90.4 icmp_seq=3 ttl=60 time=62.686 ms
84 bytes from 192.168.90.4 icmp_seq=4 ttl=60 time=62.710 ms
84 bytes from 192.168.90.4 icmp_seq=5 ttl=60 time=62.538 ms
```

.

Test connectivity from pc1 to R3 (3 network) in another campus

```
PC1>
PC1> ping 192.168.1.6
84 bytes from 192.168.1.6 icmp_seq=1 ttl=253 time=46.921 ms
84 bytes from 192.168.1.6 icmp_seq=2 ttl=253 time=47.338 ms
84 bytes from 192.168.1.6 icmp_seq=3 ttl=253 time=47.433 ms
84 bytes from 192.168.1.6 icmp_seq=4 ttl=253 time=47.476 ms
84 bytes from 192.168.1.6 icmp_seq=5 ttl=253 time=47.113 ms

PC1> ping 192.168.200.1
84 bytes from 192.168.200.1 icmp_seq=1 ttl=253 time=47.003 ms
84 bytes from 192.168.200.1 icmp_seq=2 ttl=253 time=47.353 ms
84 bytes from 192.168.200.1 icmp_seq=3 ttl=253 time=47.325 ms
84 bytes from 192.168.200.1 icmp_seq=4 ttl=253 time=47.162 ms
84 bytes from 192.168.200.1 icmp_seq=5 ttl=253 time=47.188 ms

PC1> ping 192.168.200.5
84 bytes from 192.168.200.5 icmp_seq=1 ttl=253 time=47.315 ms
84 bytes from 192.168.200.5 icmp_seq=2 ttl=253 time=47.338 ms
84 bytes from 192.168.200.5 icmp_seq=3 ttl=253 time=47.214 ms
84 bytes from 192.168.200.5 icmp_seq=4 ttl=253 time=47.624 ms
84 bytes from 192.168.200.5 icmp_seq=5 ttl=253 time=47.616 ms

PC1>
```

Figure 4-34 connectivity from pc1 to R3 in another campus

Test connectivity from pc1to R1 (4 network)

```
IOU1   IOU2   PC1   PC2   IOU3

PC1> ping 192.168.100.6
84 bytes from 192.168.100.6 icmp_seq=1 ttl=255 time=1.405 ms
84 bytes from 192.168.100.6 icmp_seq=2 ttl=255 time=1.242 ms
84 bytes from 192.168.100.6 icmp_seq=3 ttl=255 time=2.964 ms
84 bytes from 192.168.100.6 icmp_seq=4 ttl=255 time=1.585 ms
84 bytes from 192.168.100.6 icmp_seq=5 ttl=255 time=1.550 ms

PC1> ping 192.168.100.1
84 bytes from 192.168.100.1 icmp_seq=1 ttl=254 time=16.691 ms
84 bytes from 192.168.100.1 icmp_seq=2 ttl=254 time=17.295 ms
84 bytes from 192.168.100.1 icmp_seq=3 ttl=254 time=16.258 ms
84 bytes from 192.168.100.1 icmp_seq=4 ttl=254 time=17.486 ms
84 bytes from 192.168.100.1 icmp_seq=5 ttl=254 time=16.297 ms

PC1> ping 192.168.100.2
84 bytes from 192.168.100.2 icmp_seq=1 ttl=255 time=16.043 ms
84 bytes from 192.168.100.2 icmp_seq=2 ttl=255 time=16.480 ms
84 bytes from 192.168.100.2 icmp_seq=3 ttl=255 time=16.007 ms
84 bytes from 192.168.100.2 icmp_seq=4 ttl=255 time=16.451 ms
84 bytes from 192.168.100.2 icmp_seq=5 ttl=255 time=18.191 ms

PC1> ping 192.168.1.1
84 bytes from 192.168.1.1 icmp_seq=1 ttl=254 time=17.312 ms
84 bytes from 192.168.1.1 icmp_seq=2 ttl=254 time=16.112 ms
84 bytes from 192.168.1.1 icmp_seq=3 ttl=254 time=16.985 ms
84 bytes from 192.168.1.1 icmp_seq=4 ttl=254 time=16.487 ms
84 bytes from 192.168.1.1 icmp_seq=5 ttl=254 time=16.488 ms

PC1> ping 192.168.1.5
84 bytes from 192.168.1.5 icmp_seq=1 ttl=254 time=16.143 ms
84 bytes from 192.168.1.5 icmp_seq=2 ttl=254 time=16.747 ms
84 bytes from 192.168.1.5 icmp_seq=3 ttl=254 time=16.557 ms
84 bytes from 192.168.1.5 icmp_seq=4 ttl=254 time=16.400 ms
84 bytes from 192.168.1.5 icmp_seq=5 ttl=254 time=17.660 ms

PC1>
```

Figure 4-35 Test connectivity from pc1to R1

Test connectivity from pc2 to vrrp and ip in dist1& dist 2

```
IOU1   IOU2   PC1   PC2   IOU3

PC1>
PC1>
PC1>
PC1>
PC1>
PC1>
PC1>
PC1>
PC1>
PC1>
PC1>
PC1> dhcp
DORA IP 192.168.10.4/24 GW 192.168.10.1

PC1> ping 192.168.10.1
84 bytes from 192.168.10.1 icmp_seq=1 ttl=255 time=1.442 ms
84 bytes from 192.168.10.1 icmp_seq=2 ttl=255 time=1.354 ms
84 bytes from 192.168.10.1 icmp_seq=3 ttl=255 time=1.388 ms
84 bytes from 192.168.10.1 icmp_seq=4 ttl=255 time=1.383 ms
84 bytes from 192.168.10.1 icmp_seq=5 ttl=255 time=1.589 ms

PC1> ping 192.168.10.2
84 bytes from 192.168.10.2 icmp_seq=1 ttl=255 time=1.959 ms
84 bytes from 192.168.10.2 icmp_seq=2 ttl=255 time=1.233 ms
84 bytes from 192.168.10.2 icmp_seq=3 ttl=255 time=1.327 ms
84 bytes from 192.168.10.2 icmp_seq=4 ttl=255 time=1.632 ms
84 bytes from 192.168.10.2 icmp_seq=5 ttl=255 time=1.406 ms

PC1> ping 192.168.10.3
84 bytes from 192.168.10.3 icmp_seq=1 ttl=255 time=1.298 ms
84 bytes from 192.168.10.3 icmp_seq=2 ttl=255 time=1.281 ms
84 bytes from 192.168.10.3 icmp_seq=3 ttl=255 time=1.195 ms
84 bytes from 192.168.10.3 icmp_seq=4 ttl=255 time=1.289 ms
84 bytes from 192.168.10.3 icmp_seq=5 ttl=255 time=1.333 ms

PC1>

solarwinds | Solar-PuTTY free tool   © 2019-2024 SolarWinds Worldwide, LLC. All rights reserved.
```

Figure 4-36 Test connectivity from pc1 to vrrp

Test connectivity from pc1 to vrrp and ip vlan 80 in dist 3&4 in another campus

```
PC1> ping 192.168.80.1
84 bytes from 192.168.80.1 icmp_seq=1 ttl=252 time=62.554 ms
84 bytes from 192.168.80.1 icmp_seq=2 ttl=252 time=62.481 ms
84 bytes from 192.168.80.1 icmp_seq=3 ttl=252 time=63.039 ms
84 bytes from 192.168.80.1 icmp_seq=4 ttl=252 time=62.067 ms
84 bytes from 192.168.80.1 icmp_seq=5 ttl=252 time=62.391 ms

PC1> ping 192.168.80.2
84 bytes from 192.168.80.2 icmp_seq=1 ttl=252 time=62.800 ms
84 bytes from 192.168.80.2 icmp_seq=2 ttl=252 time=62.947 ms
84 bytes from 192.168.80.2 icmp_seq=3 ttl=252 time=62.732 ms
84 bytes from 192.168.80.2 icmp_seq=4 ttl=252 time=62.765 ms
84 bytes from 192.168.80.2 icmp_seq=5 ttl=252 time=62.402 ms

PC1> ping 192.168.80.3
84 bytes from 192.168.80.3 icmp_seq=1 ttl=252 time=63.151 ms
84 bytes from 192.168.80.3 icmp_seq=2 ttl=252 time=62.199 ms
84 bytes from 192.168.80.3 icmp_seq=3 ttl=252 time=62.248 ms
84 bytes from 192.168.80.3 icmp_seq=4 ttl=252 time=62.761 ms
84 bytes from 192.168.80.3 icmp_seq=5 ttl=252 time=62.662 ms
```

Figure 4-37 Test connectivity from pc2 to vrrp in another campus

Test connectivity from dist 1 to R1 (3 network)

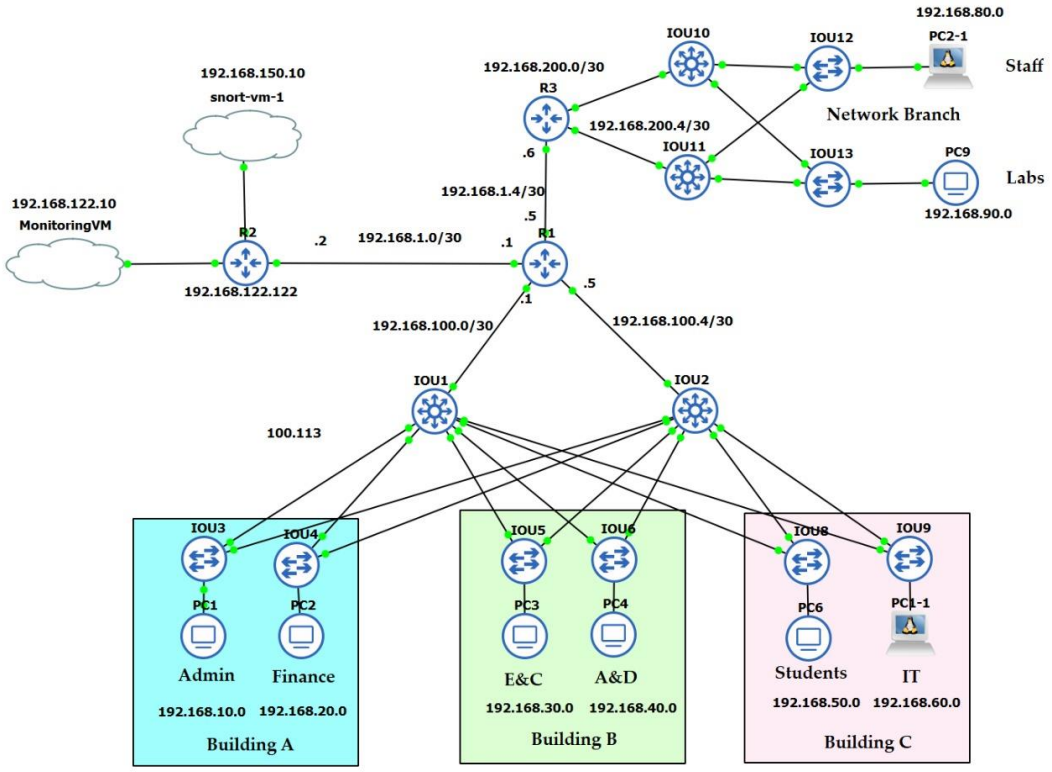
```
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.100.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 7/13/16 ms
MLS1#ping 192.168.1.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 11/14/16 ms
MLS1#ping 192.168.1.5
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.1.5, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 15/15/16 ms
MLS1#
```

Figure 4-38Test connectivity from dist 1 to R1

Test connectivity from Dist 1 to R3 (3network) in another campus

```
Sending 5, 100-byte ICMP Echos to 192.168.1.6, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 35/43/47 ms
MLS1#ping 192.168.1.6
*May  6 06:58:38.689: %CDP-4-DUPLEX_MISMATCH: duplex mismatch discovered on Ethernet1/2 (not full duplex), with IOU9 Et
rnet0/0 (full duplex).
MLS1#ping 192.168.200.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.200.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 39/44/46 ms
MLS1#
*May  6 06:58:45.085: %CDP-4-DUPLEX_MISMATCH: duplex mismatch discovered on Ethernet0/3 (not full duplex), with IOU5 Et
rnet0/0 (full duplex).
*May  6 06:58:45.863: %CDP-4-DUPLEX_MISMATCH: duplex mismatch discovered on Ethernet0/2 (not full duplex), with IOU4 Et
rnet0/0 (full duplex).
MLS1#ping 192.168.200.5
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.200.5, timeout is 2 seconds:
!!!!!
```

Figure 4-39 Test connectivity from Dist 1 to R3



4.9.2 ping from kali to all devices

```
┌──(kali㉿kali)-[~]
└─$ ping iou5
PING iou5 (192.168.30.5) 56(84) bytes of data.
64 bytes from iou5 (192.168.30.5): icmp_seq=2 ttl=252 time=52.2 ms
64 bytes from iou5 (192.168.30.5): icmp_seq=3 ttl=252 time=50.2 ms
64 bytes from iou5 (192.168.30.5): icmp_seq=4 ttl=252 time=54.6 ms
^C
--- iou5 ping statistics ---
4 packets transmitted, 3 received, 25% packet loss, time 3034ms
rtt min/avg/max/mdev = 50.202/52.312/54.583/1.792 ms

┌──(kali㉿kali)-[~]
└─$ ping iou6
PING iou6 (192.168.40.5) 56(84) bytes of data.
64 bytes from iou6 (192.168.40.5): icmp_seq=2 ttl=252 time=60.0 ms
64 bytes from iou6 (192.168.40.5): icmp_seq=3 ttl=252 time=58.0 ms
64 bytes from iou6 (192.168.40.5): icmp_seq=4 ttl=252 time=61.1 ms
^C
--- iou6 ping statistics ---
4 packets transmitted, 3 received, 25% packet loss, time 3023ms
rtt min/avg/max/mdev = 57.974/59.691/61.128/1.302 ms

┌──(kali㉿kali)-[~]
└─$ ping iou7
^C

┌──(kali㉿kali)-[~]
└─$ ping iou8
PING iou8 (192.168.50.5) 56(84) bytes of data.
64 bytes from iou8 (192.168.50.5): icmp_seq=2 ttl=252 time=59.6 ms
64 bytes from iou8 (192.168.50.5): icmp_seq=3 ttl=252 time=57.5 ms
64 bytes from iou8 (192.168.50.5): icmp_seq=4 ttl=252 time=58.1 ms
64 bytes from iou8 (192.168.50.5): icmp_seq=5 ttl=252 time=59.0 ms
^C
--- iou8 ping statistics ---
5 packets transmitted, 4 received, 20% packet loss, time 4012ms
rtt min/avg/max/mdev = 57.504/58.548/59.605/0.816 ms
```

Figure 4-40 ping from kali

2

```
┌──(kali㉿kali)-[~]
└─$ ping iou9
PING iou9 (192.168.60.5) 56(84) bytes of data.
64 bytes from iou9 (192.168.60.5): icmp_seq=2 ttl=252 time=60.9 ms
64 bytes from iou9 (192.168.60.5): icmp_seq=3 ttl=252 time=52.2 ms
64 bytes from iou9 (192.168.60.5): icmp_seq=4 ttl=252 time=52.3 ms
^C
--- iou9 ping statistics ---
4 packets transmitted, 3 received, 25% packet loss, time 3015ms
rtt min/avg/max/mdev = 52.201/55.111/60.880/4.079 ms

┌──(kali㉿kali)-[~]
└─$ ping iou10
PING iou10 (192.168.200.2) 56(84) bytes of data.
64 bytes from iou10 (192.168.200.2): icmp_seq=1 ttl=252 time=80.8 ms
64 bytes from iou10 (192.168.200.2): icmp_seq=2 ttl=252 time=82.6 ms
64 bytes from iou10 (192.168.200.2): icmp_seq=3 ttl=252 time=92.2 ms
64 bytes from iou10 (192.168.200.2): icmp_seq=4 ttl=252 time=92.6 ms
^C
--- iou10 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 80.827/87.082/92.648/5.387 ms

┌──(kali㉿kali)-[~]
└─$ ping iou11
PING iou11 (192.168.200.6) 56(84) bytes of data.
64 bytes from iou11 (192.168.200.6): icmp_seq=1 ttl=252 time=86.2 ms
64 bytes from iou11 (192.168.200.6): icmp_seq=2 ttl=252 time=93.5 ms
64 bytes from iou11 (192.168.200.6): icmp_seq=3 ttl=252 time=90.5 ms
^C
--- iou11 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 86.172/90.034/93.468/2.993 ms
```

3

```
┌──(kali㉿kali)-[~]
└─$ ping pc2
PING pc2 (192.168.80.2) 56(84) bytes of data.
64 bytes from pc2 (192.168.80.2): icmp_seq=1 ttl=60 time=83.1 ms
64 bytes from pc2 (192.168.80.2): icmp_seq=2 ttl=60 time=93.0 ms
64 bytes from pc2 (192.168.80.2): icmp_seq=3 ttl=60 time=90.1 ms
^C
--- pc2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 83.092/88.718/92.958/4.145 ms

┌──(kali㉿kali)-[~]
└─$ ping snort-vm
PING snort-vm (192.168.150.10) 56(84) bytes of data.
64 bytes from snort-vm (192.168.150.10): icmp_seq=1 ttl=63 time=29.2 ms
64 bytes from snort-vm (192.168.150.10): icmp_seq=2 ttl=63 time=24.0 ms
64 bytes from snort-vm (192.168.150.10): icmp_seq=3 ttl=63 time=19.3 ms
^C
--- snort-vm ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 19.339/24.149/29.153/4.008 ms

┌──(kali㉿kali)-[~]
└─$ ping prometheus-vm
PING prometheus-vm (192.168.122.10) 56(84) bytes of data.
64 bytes from prometheus-vm (192.168.122.10): icmp_seq=1 ttl=64 time=0.022 ms
64 bytes from prometheus-vm (192.168.122.10): icmp_seq=2 ttl=64 time=0.029 ms
64 bytes from prometheus-vm (192.168.122.10): icmp_seq=3 ttl=64 time=0.032 ms
64 bytes from prometheus-vm (192.168.122.10): icmp_seq=4 ttl=64 time=0.032 ms
^C
--- prometheus-vm ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3062ms
rtt min/avg/max/mdev = 0.022/0.028/0.032/0.004 ms
```

4

```
┌──(kali㉿kali)-[~]
└─$ ping iou12
PING iou12 (192.168.80.5) 56(84) bytes of data.
64 bytes from iou12 (192.168.80.5): icmp_seq=2 ttl=251 time=90.6 ms
64 bytes from iou12 (192.168.80.5): icmp_seq=3 ttl=251 time=86.6 ms
64 bytes from iou12 (192.168.80.5): icmp_seq=4 ttl=251 time=88.9 ms
^C
--- iou12 ping statistics ---
4 packets transmitted, 3 received, 25% packet loss, time 3024ms
rtt min/avg/max/mdev = 86.628/88.692/90.588/1.621 ms

┌──(kali㉿kali)-[~]
└─$ ping iou13
PING iou13 (192.168.90.5) 56(84) bytes of data.
64 bytes from iou13 (192.168.90.5): icmp_seq=2 ttl=251 time=84.7 ms
64 bytes from iou13 (192.168.90.5): icmp_seq=3 ttl=251 time=78.8 ms
64 bytes from iou13 (192.168.90.5): icmp_seq=4 ttl=251 time=80.1 ms
^C
--- iou13 ping statistics ---
4 packets transmitted, 3 received, 25% packet loss, time 3014ms
rtt min/avg/max/mdev = 78.826/81.195/84.657/2.502 ms

┌──(kali㉿kali)-[~]
└─$ ping pc1
PING pc1 (192.168.60.2) 56(84) bytes of data.
64 bytes from pc1 (192.168.60.2): icmp_seq=1 ttl=61 time=63.4 ms
64 bytes from pc1 (192.168.60.2): icmp_seq=2 ttl=61 time=60.5 ms
64 bytes from pc1 (192.168.60.2): icmp_seq=3 ttl=61 time=56.4 ms
^C
--- pc1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 56.366/60.073/63.385/2.879 ms
```

5

```
┌──(kali㉿kali)-[~]
└─$ ping iou1
PING iou1 (192.168.100.2) 56(84) bytes of data.
64 bytes from iou1 (192.168.100.2): icmp_seq=1 ttl=253 time=63.3 ms
64 bytes from iou1 (192.168.100.2): icmp_seq=2 ttl=253 time=53.3 ms
64 bytes from iou1 (192.168.100.2): icmp_seq=3 ttl=253 time=59.5 ms
^C
--- iou1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 53.326/58.707/63.315/4.114 ms

┌──(kali㉿kali)-[~]
└─$ ping iou2
PING iou2 (192.168.100.6) 56(84) bytes of data.
64 bytes from iou2 (192.168.100.6): icmp_seq=1 ttl=253 time=51.3 ms
64 bytes from iou2 (192.168.100.6): icmp_seq=2 ttl=253 time=57.4 ms
64 bytes from iou2 (192.168.100.6): icmp_seq=3 ttl=253 time=53.8 ms
^C
--- iou2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 51.349/54.191/57.377/2.472 ms

┌──(kali㉿kali)-[~]
└─$ ping iou3
PING iou3 (192.168.10.5) 56(84) bytes of data.
64 bytes from iou3 (192.168.10.5): icmp_seq=2 ttl=252 time=51.8 ms
64 bytes from iou3 (192.168.10.5): icmp_seq=3 ttl=252 time=63.6 ms
64 bytes from iou3 (192.168.10.5): icmp_seq=4 ttl=252 time=50.6 ms
^C
--- iou3 ping statistics ---
4 packets transmitted, 3 received, 25% packet loss, time 3006ms
rtt min/avg/max/mdev = 50.606/55.332/63.634/5.889 ms

┌──(kali㉿kali)-[~]
└─$ ping iou4
PING iou4 (192.168.20.5) 56(84) bytes of data.
64 bytes from iou4 (192.168.20.5): icmp_seq=2 ttl=252 time=52.1 ms
64 bytes from iou4 (192.168.20.5): icmp_seq=3 ttl=252 time=52.6 ms
64 bytes from iou4 (192.168.20.5): icmp_seq=4 ttl=252 time=52.2 ms
^C
--- iou4 ping statistics ---
4 packets transmitted, 3 received, 25% packet loss, time 3034ms
rtt min/avg/max/mdev = 52.144/52.315/52.616/0.213 ms
```

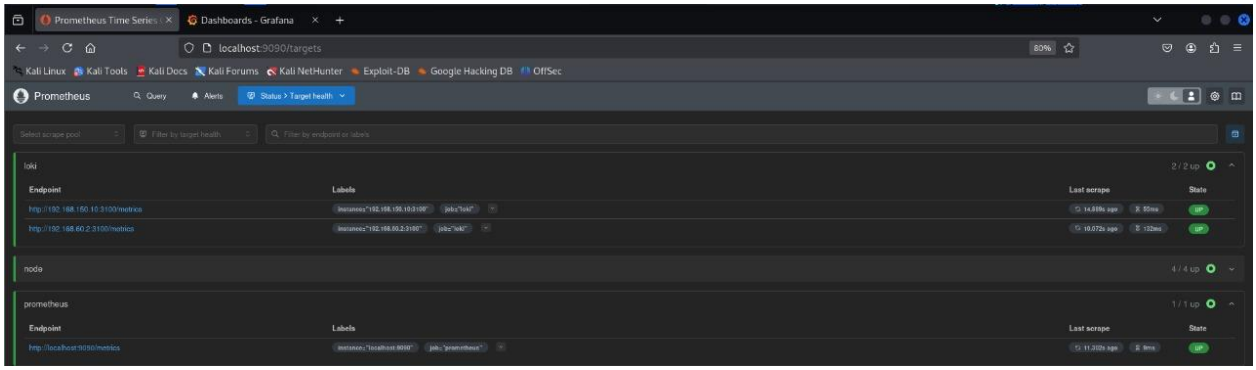
6

```
┌──(kali㉿kali)-[~]
└─$ ping r1
PING r1 (192.168.100.1) 56(84) bytes of data.
64 bytes from r1 (192.168.100.1): icmp_seq=1 ttl=254 time=51.7 ms
64 bytes from r1 (192.168.100.1): icmp_seq=2 ttl=254 time=46.9 ms
^C
--- r1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 46.911/49.299/51.688/2.388 ms

┌──(kali㉿kali)-[~]
└─$ ping r2
PING r2 (192.168.122.122) 56(84) bytes of data.
64 bytes from r2 (192.168.122.122): icmp_seq=1 ttl=255 time=8.14 ms
64 bytes from r2 (192.168.122.122): icmp_seq=2 ttl=255 time=3.65 ms
64 bytes from r2 (192.168.122.122): icmp_seq=3 ttl=255 time=3.88 ms
^C
--- r2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 3.652/5.223/8.142/2.066 ms

┌──(kali㉿kali)-[~]
└─$ ping r3
PING r3 (192.168.1.6) 56(84) bytes of data.
64 bytes from r3 (192.168.1.6): icmp_seq=1 ttl=253 time=79.5 ms
64 bytes from r3 (192.168.1.6): icmp_seq=2 ttl=253 time=72.9 ms
64 bytes from r3 (192.168.1.6): icmp_seq=3 ttl=253 time=66.2 ms
^C
--- r3 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 66.176/72.848/79.493/5.436 ms

┌──(kali㉿kali)-[~]
└─$ ping iou1
PING iou1 (192.168.100.2) 56(84) bytes of data.
64 bytes from iou1 (192.168.100.2): icmp_seq=1 ttl=253 time=63.3 ms
64 bytes from iou1 (192.168.100.2): icmp_seq=2 ttl=253 time=53.3 ms
64 bytes from iou1 (192.168.100.2): icmp_seq=3 ttl=253 time=59.5 ms
^C
```

4.10 MONITORING

4.10.1 PROMETHEUS

Target



.

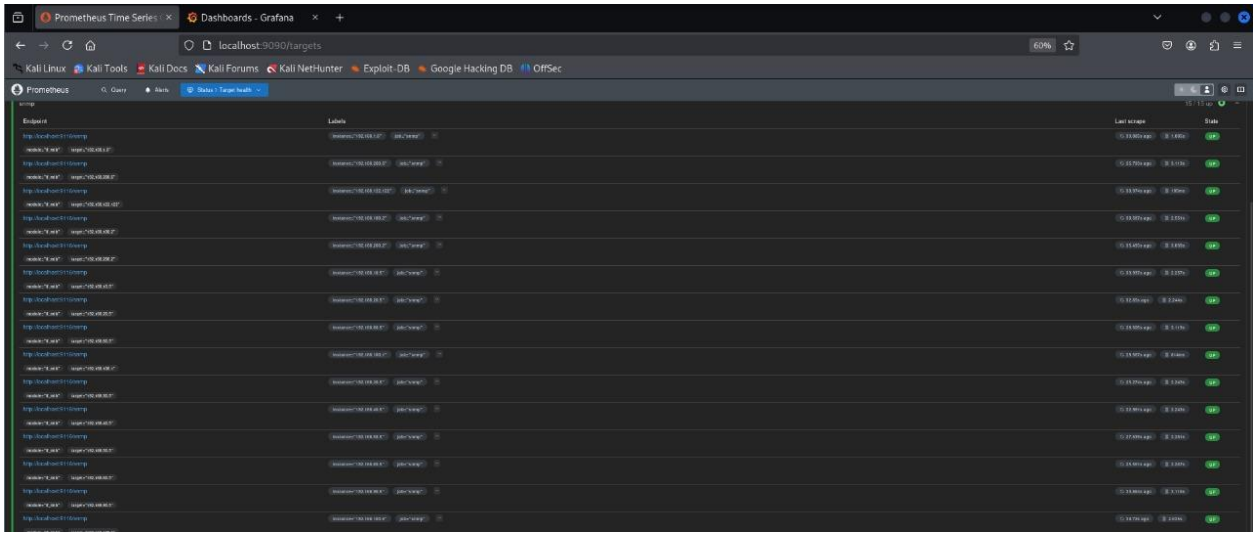


Figure 4-41 PROMETHEUS target

This view shows the active targets being monitored by Prometheus. Each target represents a network node or service from which metrics are being collected in real-time.

ALERTS

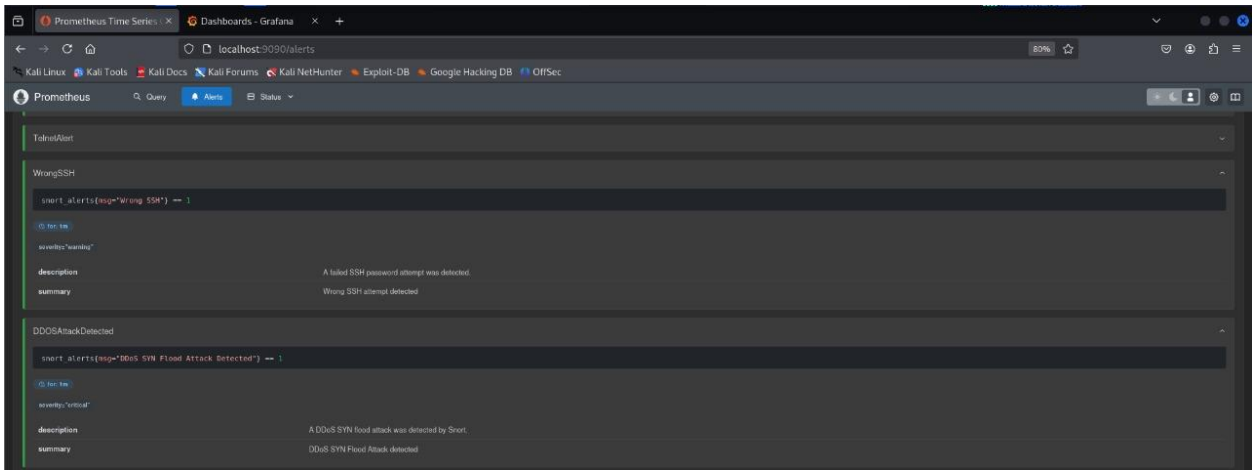
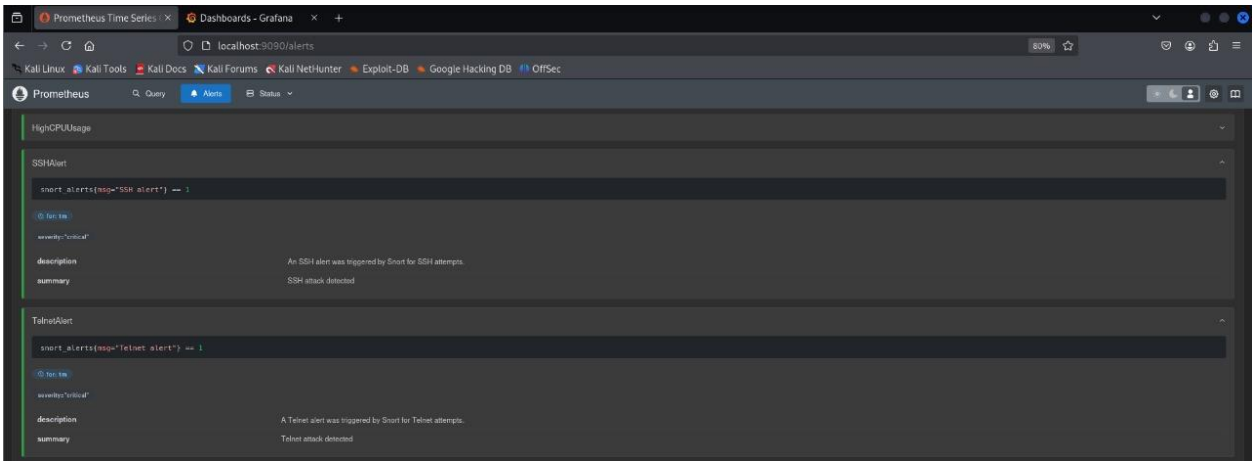
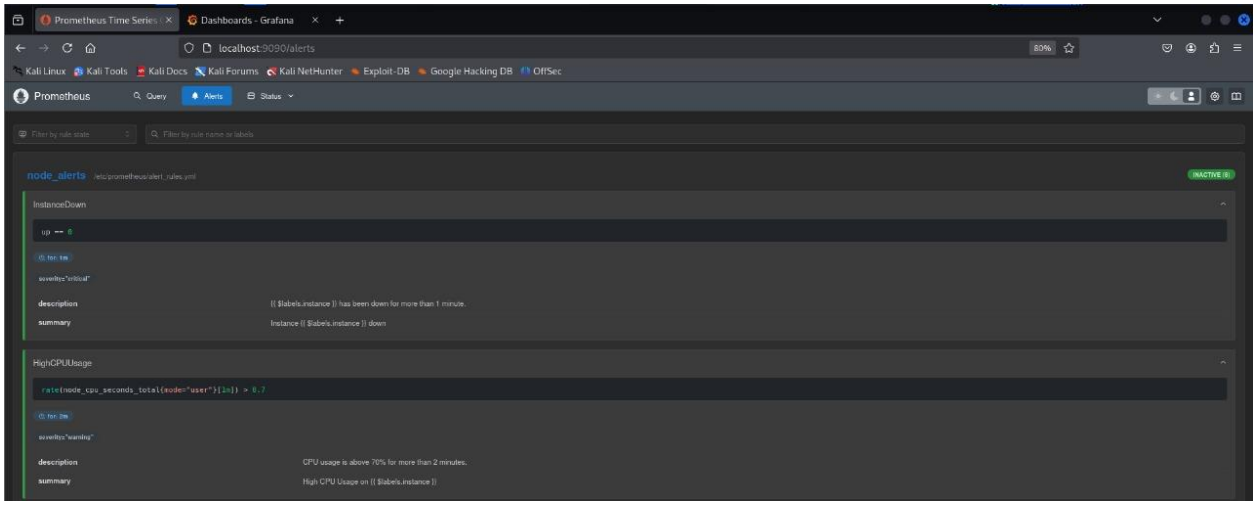


Figure 4-42 Alerts

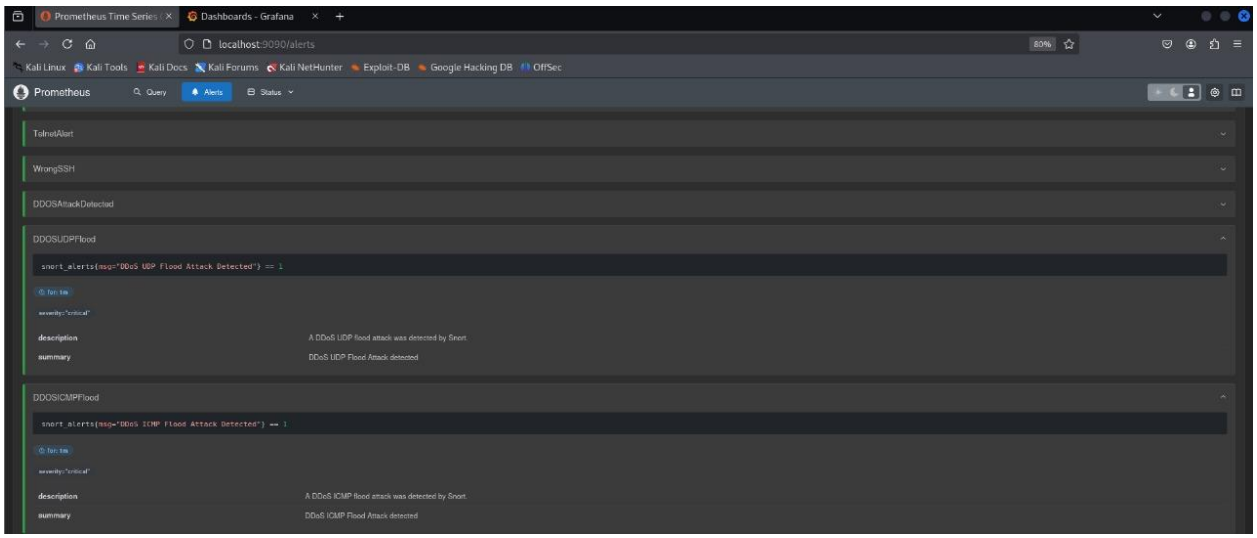
.



.



.



This figure displays the list of triggered alerts based on predefined rules. These alerts notify administrators about critical issues such as high CPU usage, downed nodes, or suspicious network activity.

4.10.2 Grafana Dashboard

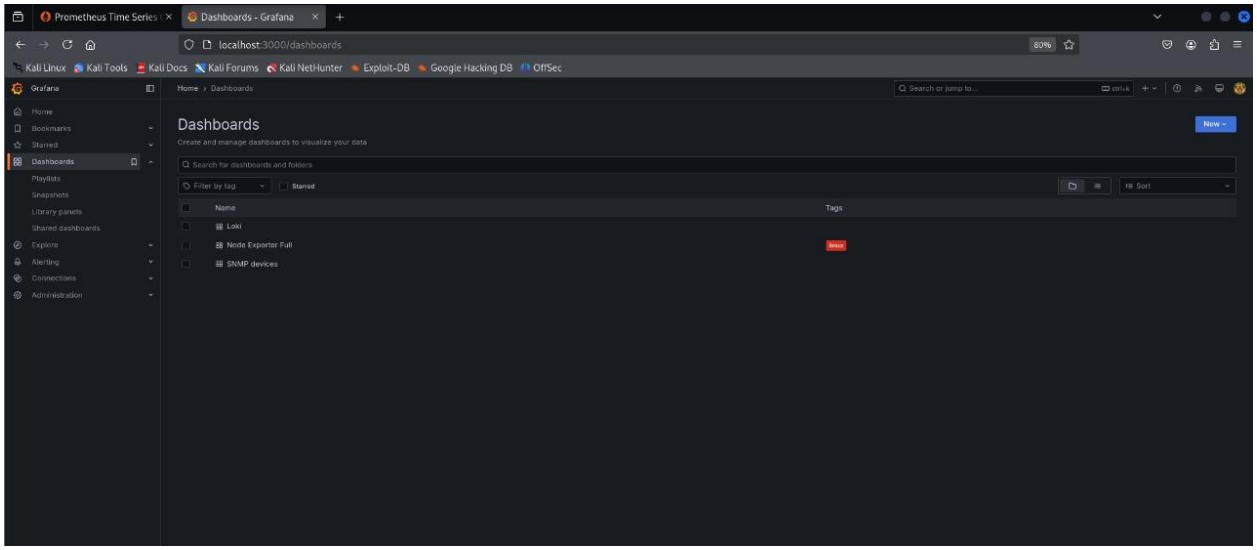
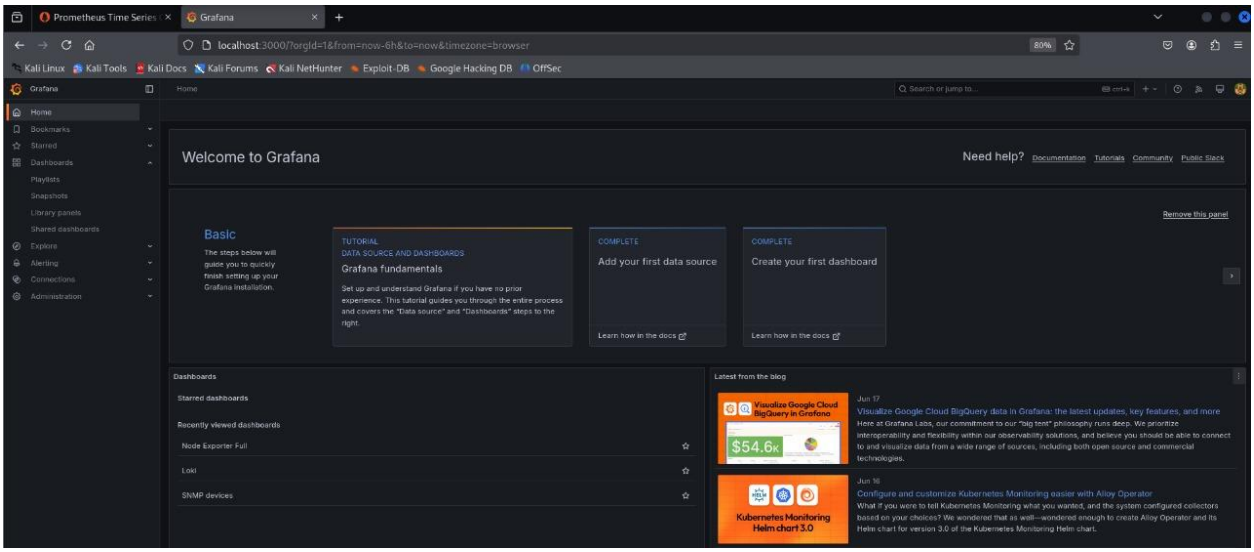


Figure 4-43 Grafana Dashboard



.

This dashboard provides real-time visualization of network performance metrics collected via Prometheus, such as CPU usage, memory consumption, and traffic load.

4.10.3 node exporter dashboard

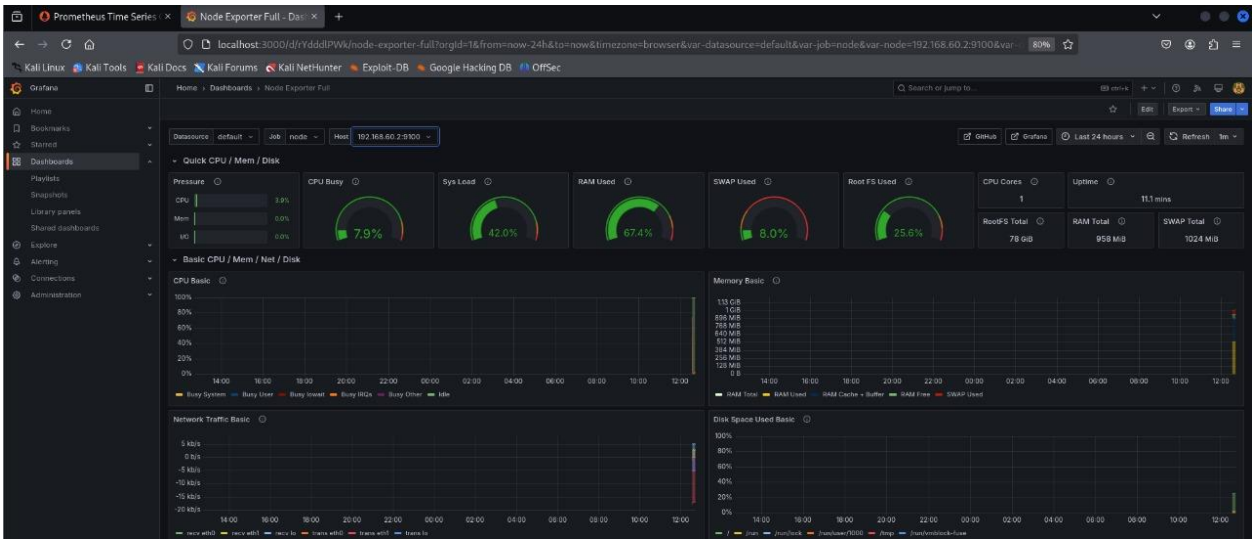
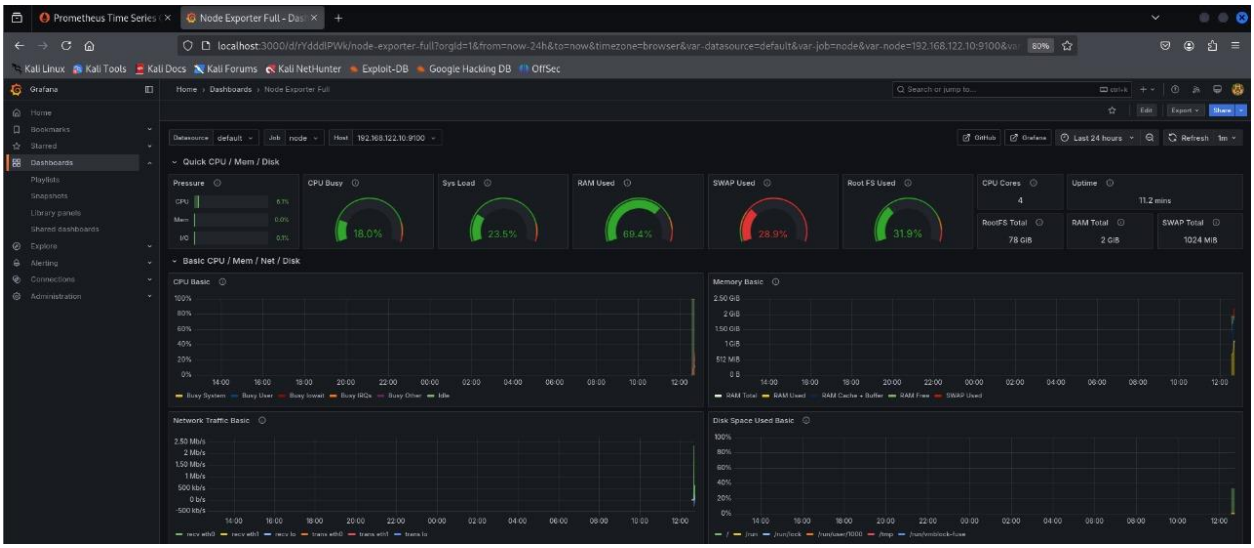
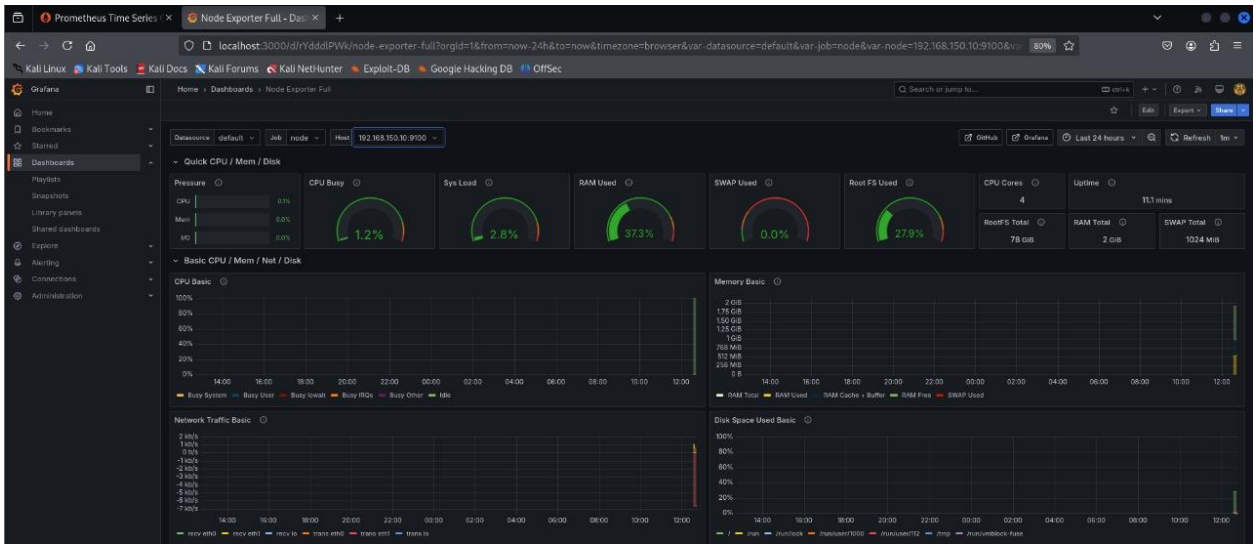


Figure 4-44 node exporter

.

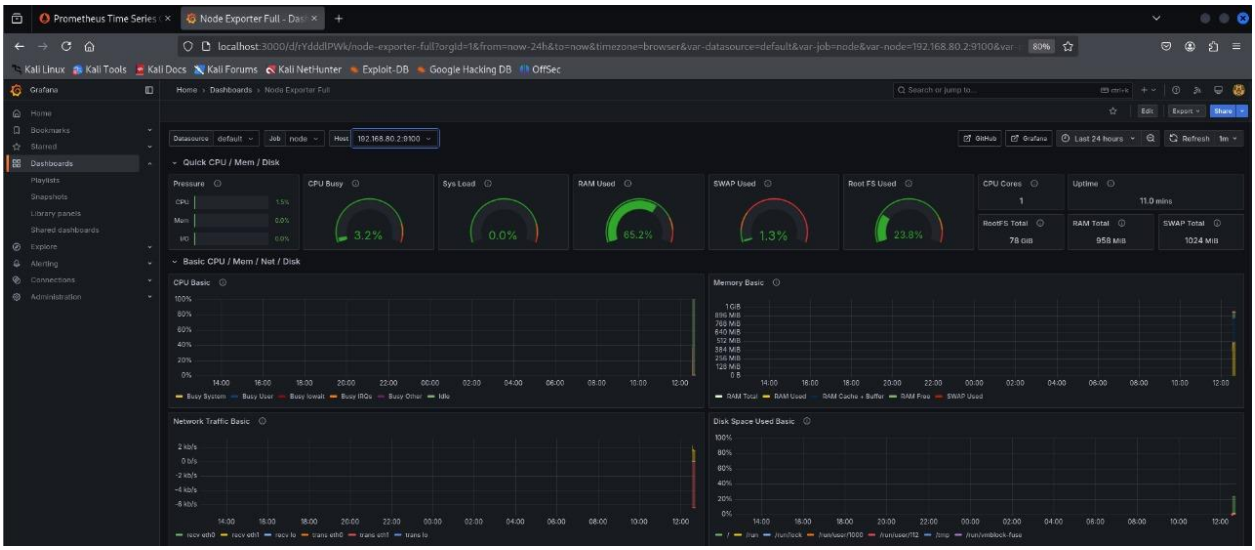


.

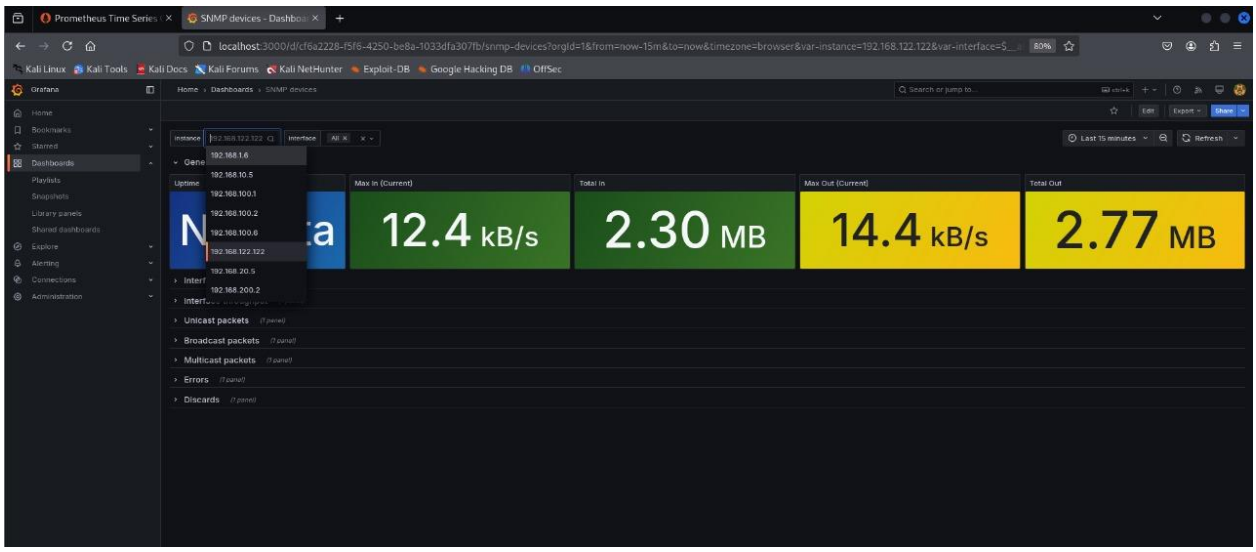


.

This figure shows the metrics exposed by Node Exporter, which provides system-level statistics like CPU usage, memory status, disk I/O, and network interfaces for Linux-based nodes.



4.10.4 snmp exporter dashboard



.

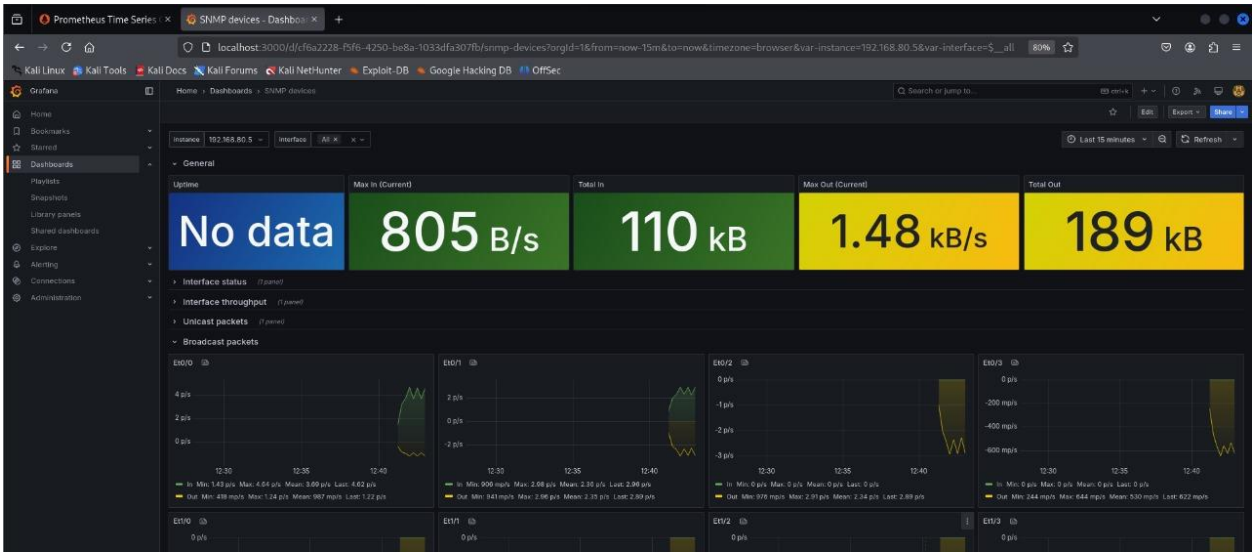
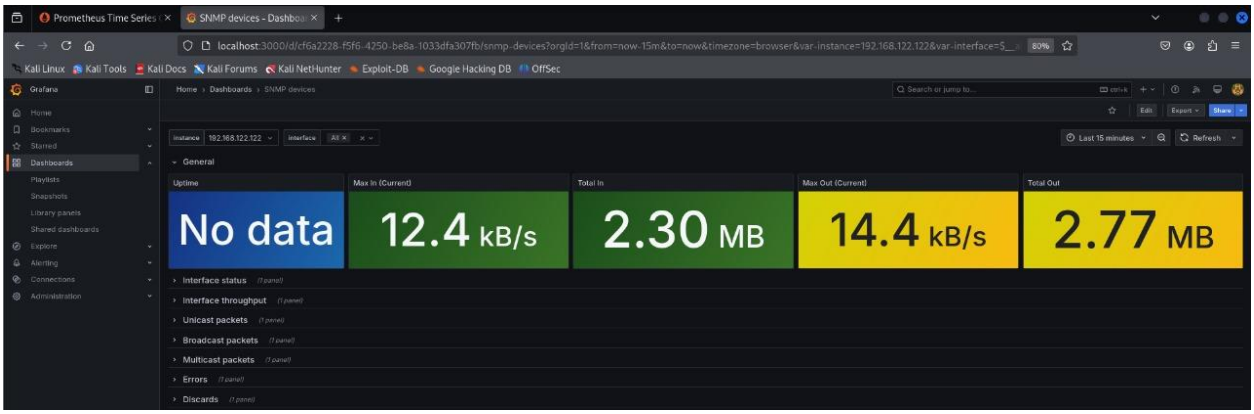
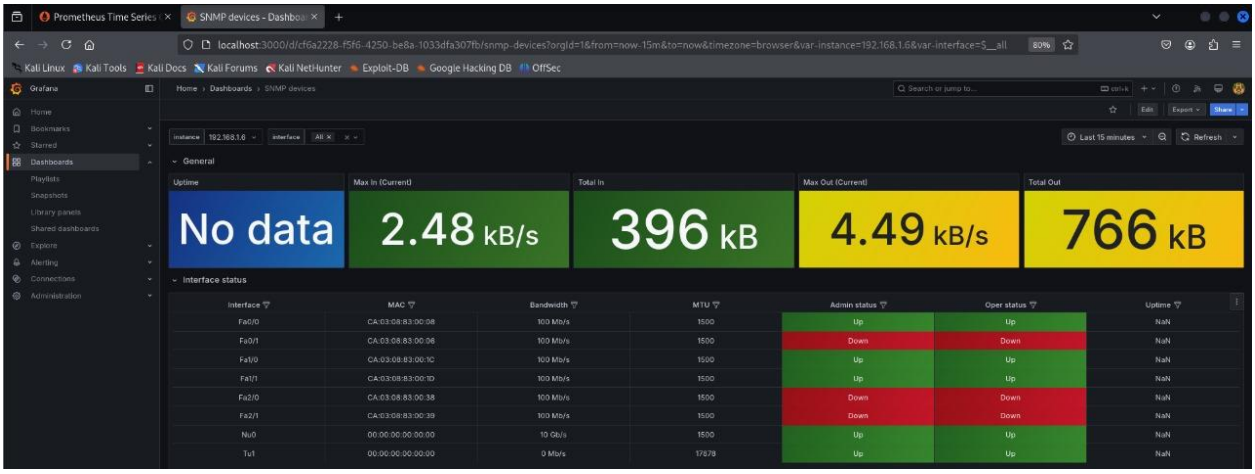


Figure 4-45 snmp exporter

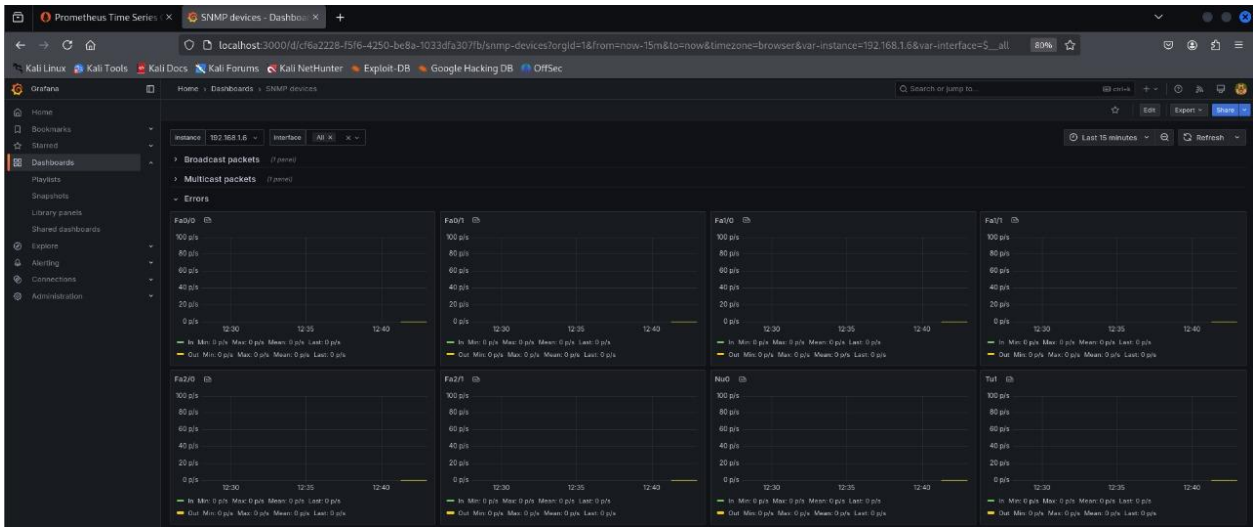
.


Prometheus Time Series
SNMP devices - Dashboa
localhost:3000/d/cf6a2228-f5f6-4250-be8a-1033dfa307fb/snmp-devices?orgId=1&from=now-15m&to=now&timezone=browser&var-instance=192.168.122.122&var-interface=$__
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec
Grafana
Home
Bookmarks
Starred
Dashboards
Playlists
Snapshots
Library panels
Shared dashboards
Explore
Alerting
Connections
Administration
Home › Dashboards › SNMP devices
Edit
Export
Share
instance
192.168.122.122
interface
All
Last 15 minutes
Refresh
General
Uptime
No data
Max In (Current)
12.4 kB/s
Total In
2.30 MB
Max Out (Current)
14.4 kB/s
Total Out
2.77 MB
Interface status
Interface throughput
Unicast packets
Broadcast packets
Multicast packets
Errors
Discards

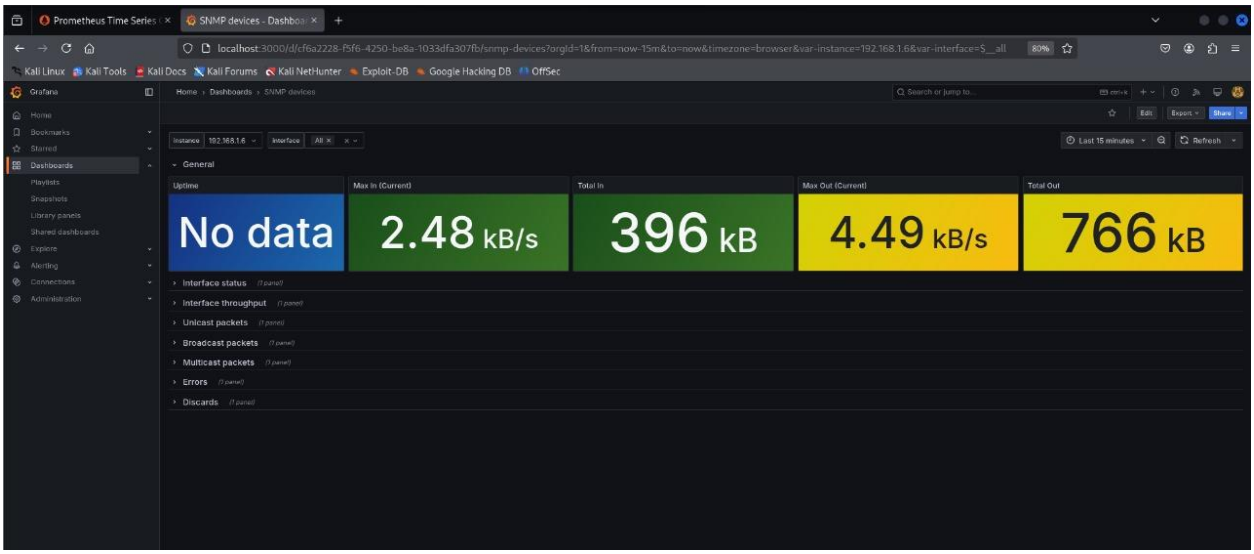
.


Prometheus Time Series
SNMP devices - Dashboa
localhost:3000/d/cf6a2228-f5f6-4250-be8a-1033dfa307fb/snmp-devices?orgId=1&from=now-15m&to=now&timezone=browser&var-instance=192.168.1.6&var-interface=$__all
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec
Grafana
Home
Bookmarks
Starred
Dashboards
Playlists
Snapshots
Library panels
Shared dashboards
Explore
Alerting
Connections
Administration
Home › Dashboards › SNMP devices
Edit
Export
Share
instance
192.168.1.6
interface
All
Last 15 minutes
Refresh
General
Uptime
No data
Max In (Current)
2.48 kB/s
Total In
396 kB
Max Out (Current)
4.49 kB/s
Total Out
766 kB
Interface status
Interface
MAC
Bandwidth
MTU
Admin status
Oper status
Uptime
Fa0/0 CA:03:08:83:00:08 100 Mb/s 1500 Up Up NaN
Fa0/1 CA:03:08:83:00:06 100 Mb/s 1500 Down Down NaN
Fa1/0 CA:03:08:83:00:1C 100 Mb/s 1500 Up Up NaN
Fa1/1 CA:03:08:83:00:1D 100 Mb/s 1500 Up Up NaN
Fa2/0 CA:03:08:83:00:38 100 Mb/s 1500 Down Down NaN
Fa2/1 CA:03:08:83:00:39 100 Mb/s 1500 Down Down NaN
Nu0 00:00:00:00:00:00 10 Gb/s 1500 Up Up NaN
Tu1 00:00:00:00:00:00 0 Mb/s 17878 Up Up NaN

.



.



This figure illustrates how SNMP Exporter enables Prometheus to collect metrics from network devices using the SNMP protocol, such as interface traffic, device uptime, and error counters.

4.10.5 loki dashboard

This dashboard visualizes system and network logs collected via Loki, allowing administrators to

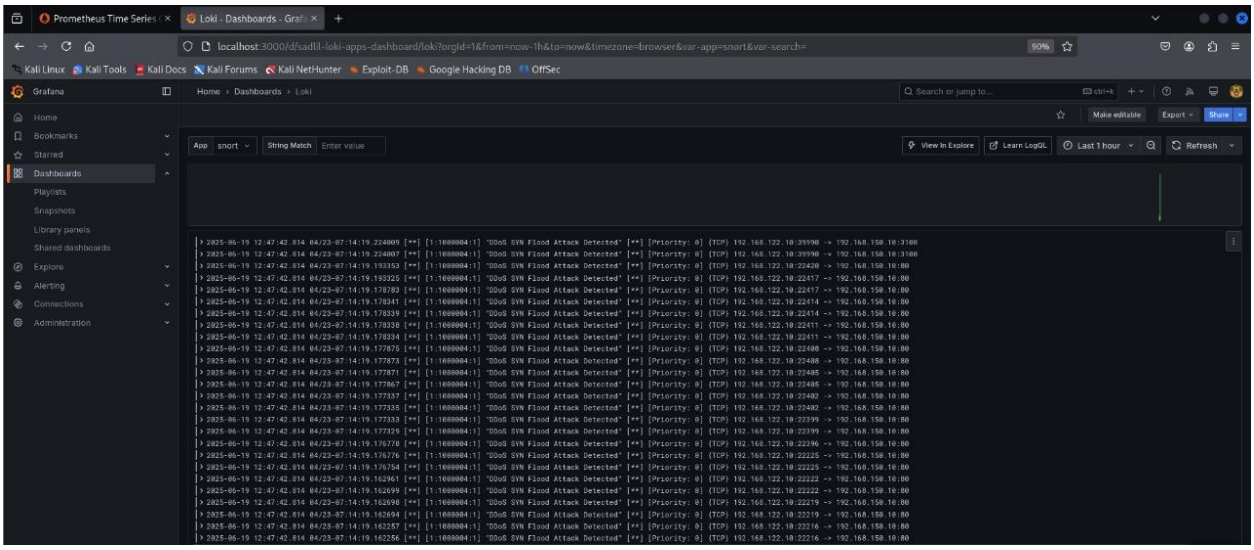
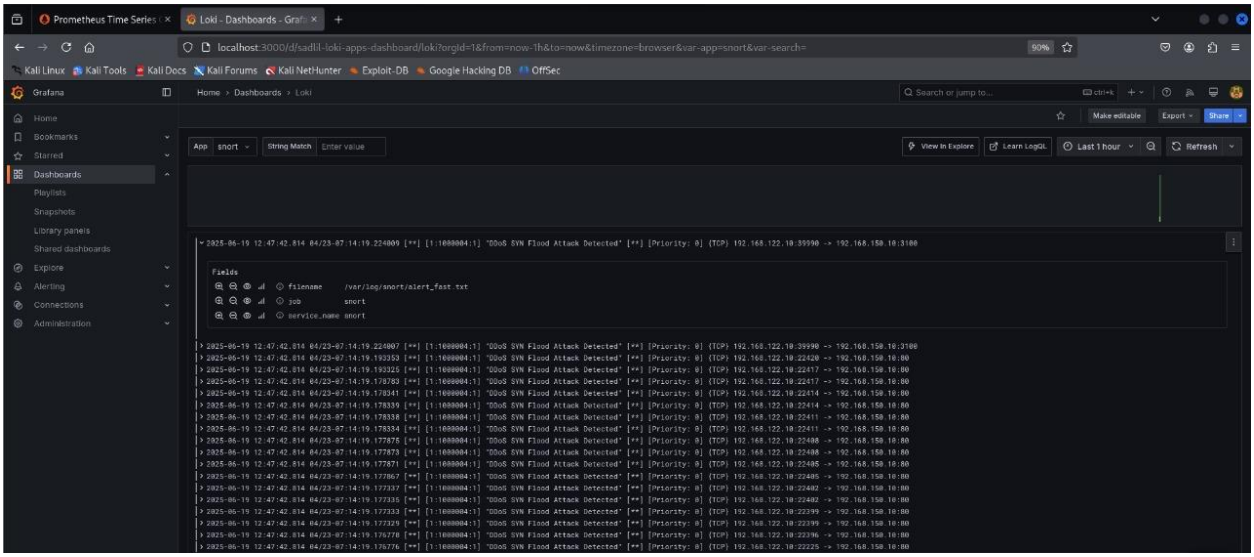


Figure 4-46 loki dashboard

track events, troubleshoot issues, and monitor system behavior over time.



CHAPTER 5 : CONCLUSION

The project successfully demonstrated how DevOps tools can enhance network management through automation, monitoring, and security. Each tool played a critical role in building a robust and efficient network

5.1 PROJECT OUTCOMES

The implementation of this project led to several valuable technical and practical outcomes:

- **Automated Network** Configuration: By using Ansible, we successfully automated repetitive network configuration tasks, which reduced human error and accelerated deployment time.
- **Real-time Network Monitoring**: Prometheus and Grafana enabled continuous monitoring and visualization of key network metrics, improving situational awareness and proactive maintenance.
- **Intrusion Detection and Alerting:** The integration of Snort with Alert Manager allowed the system to detect and respond to suspicious traffic patterns, significantly enhancing network security.
- **Simulated Enterprise Environment:** The network topology built using GNS3 and VMware simulated a real-world, multi-department enterprise network, enabling accurate testing and training scenarios.
- **DevOps in Networking**: The project demonstrated the applicability of DevOps principles in networking (NetDevOps), bridging the gap between development, operations, and network administration.

5.2 LIMITATIONS

Despite the successful implementation, the project faced some limitations:

- The testing environment was based on simulation tools (GNS3 and VMware), which may not fully represent real-world network conditions.
- Limited resources constrained the performance of the virtual machines and network devices.

- Some advanced features of monitoring and security tools were not explored due to time and resource constraints.

5.3 CHALLENGES AND LESSONS LEARNED

Throughout the development of this project, the team encountered various technical and logistical challenges that provided valuable learning experiences. These challenges helped shape our understanding of network automation, security, and monitoring in real-world scenarios. Below is a summary of the key obstacles faced and the lessons learned from overcoming them:

- **Tool Compatibility and Integration**

➢ **Challenge:** Integrating open-source DevOps tools like Ansible, Prometheus, Grafana, Snort, and Alertmanager in a simulated GNS3/VMware environment was not straightforward. Each tool had different configuration requirements and dependencies.

➢ **Lesson Learned:** Careful planning and reading official documentation is critical. Understanding the architecture of each tool helped us configure them correctly and troubleshoot integration issues efficiently.

- **.Snort Sensitivity and Tuning**

➢ **Challenge**: Initially, Snort generated many false positives and irrelevant alerts, making it difficult to distinguish real threats from normal traffic.

➢ **Lesson Learned**: Intrusion detection systems require fine-tuning. Writing accurate Snort rules and adjusting sensitivity thresholds is essential to avoid alert fatigue and ensure meaningful detection.

- **Network Connectivity and Routing Errors**

➢ **Challenge:** Several routing misconfigurations occurred during testing phases, leading to communication failures between subnets and devices.

- **Lesson Learned**: Proper IP addressing schemes, routing protocols (like OSPF), and subnet planning are crucial. Visual topology maps and continuous testing helped isolate and fix misconfigurations quickly.

- **Automation Logic Complexity**
- **Challenge**: Creating efficient Ansible playbooks for different network devices with varying CLI syntaxes was more complex than anticipated.

- **Lesson Learned:** Standardizing configuration templates and using variables/modules in Ansible improved reusability and minimized errors. Automation must be modular and tested incrementally.
- **Real-Time Monitoring Setup**

- **Challenge:** Configuring Prometheus to scrape metrics from all network nodes and exporters, while maintaining performance and data accuracy, was challenging.

- **Lesson Learned**: Exporters must be correctly installed and exposed, and scrape intervals balanced to avoid overloading the monitoring system. Grafana dashboards also require thoughtful design for clarity.
- **Team Coordination and Task Division**

- **Challenge:** Coordinating tasks among three roles (Automation, Network, Security) required clear communication and time management.

- **Lesson Learned**: Using version control tools and maintaining documentation ensured consistency. Weekly planning sessions helped the team stay on track and address blockers together.

Conclusion

Facing and overcoming these challenges enriched our technical skills and problem-solving capabilities. More importantly, they emphasized the importance of planning, collaboration, and adaptability in complex IT projects. The lessons

learned here will undoubtedly serve as a strong foundation for future careers in networking, cybersecurity, and DevOps.

5.4 SUMMARY

This project presented a comprehensive approach to transforming traditional network management into a modern, automated, and secure infrastructure through the integration of DevOps methodologies.

A realistic enterprise network was successfully designed and simulated using GNS3 and VMware Workstation, where Ansible was utilized to automate configuration tasks, significantly minimizing human error and increasing operational efficiency.

The deployment of Prometheus and Grafana provided real-time monitoring and visualization of network performance, enhancing the network's transparency and facilitating proactive maintenance.

In parallel, the integration of Snort and Alert Manager established a robust security framework capable of detecting and responding to network threats in real time.

Through rigorous testing, including simulated attacks using Kali Linux, the system’s resilience and responsiveness were validated.

The project conclusively demonstrates that the adoption of automation, centralized monitoring, and security integration is essential for building scalable, reliable, and future-proof network infrastructures.

5.5 FUTURE WORK AND RECOMMENDATIONS

While this project achieved its primary objectives in demonstrating the power of network automation, monitoring, and security integration, there remain several opportunities for future enhancements and expansions.

Potential future work includes:

- Scaling the Network: Expanding the simulated environment to include more departments, larger topologies, and more complex routing protocols to better mimic enterprise-level networks.
- Advanced Security Integration: Incorporating more sophisticated Intrusion Prevention Systems (IPS) alongside Snort, and deploying Security Information and Event Management (SIEM) solutions for enhanced threat analysis.
- Machine Learning for Anomaly Detection: Implementing AI/ML algorithms that can analyze network traffic patterns and predict potential threats before they occur.
- Cloud Integration: Extending the current network model to hybrid cloud infrastructures using platforms like AWS, Azure, or GCP, to simulate modern enterprise architectures.
- Continuous Deployment and Testing: Integrating CI/CD pipelines for automatic network updates, configuration testing, and rollback procedures to further enhance reliability.
- Zero Trust Security Model: Researching and implementing a Zero Trust framework to strengthen internal network security measures.
- By pursuing these advancements, the project can evolve from a simulated environment to a real-world, enterprise-grade solution capable of meeting the demands of next-generation IT infrastructures.

REFERENCES

1. Wikipedia. "Packet Tracer." https://en.wikipedia.org/wiki/Packet_Tracer
2. Paessler. "What is a Server?" https://www.paessler.com/it-explained/server
3. Computer Networking Blog. "SSH Configuration on a Switch and Router in Packet Tracer." Link
4. Router Over Blog. "How to Configure Cisco Router Password."
5. http://router.over-blog.com/article-how-to-configure-cisco-router-password-106850439.html
6. Cognoscape. "Benefits of Going Wireless." https://www.cognoscape.com/benefits-going-wireless/
7. IPCisco. "DNS Configuration on Cisco Routers." https://ipcisco.com/lesson/dns-configuration-on-cisco-routers/
8. Firewall.cx. "Cisco Router Site-to-Site IPSec VPN." https://www.firewall.cx/cisco/cisco-routers/cisco-router-site-to-site-ipsec-vpn.html
9. Cisco Learning Network. "DHCP Client in GNS3." https://learningnetwork.cisco.com/s/question/0D53i00000Kt7TpCAJ/dhcp-client-in-gns3
10. ManageEngine. "Configure DHCP Server - Cisco." https://www.manageengine.com/network-configuration-manager/configlets/configure-dhcp-server-cisco.html
11. Thra3. "What is IPSEC." https://www.thra3.com/2021/11/what-is-IPSEC.html

12. SpeakNetworks. "Basic Cisco ASA 5506-X Configuration Example." https://www.speaknetworks.com/basic-cisco-asa-5506-x-configuration-example
13. Cisco Support. "Cisco Router IPSec VPN Configuration." https://www.cisco.com/c/ar_ae/support/docs/routers/1700-series-modular-access-routers/71462-rtr-l2l-ipsec-split.html
14. Amazon AWS. "What is IPSec." https://aws.amazon.com/ar/what-is/ipsec
15. Hsoub Academy. https://academy.hsoub.com
16. Cisco Networking Academy. "Networking Essentials."
17. Ansible Official Documentation. https://docs.ansible.com
18. Prometheus Documentation. https://prometheus.io/docs/
19. Grafana Documentation. https://grafana.com/docs/
20. Snort Official Documentation. https://snort.org/documents
21. Loki & Grafana Labs Documentation. https://grafana.com/oss/loki/