

Tuan Anh Nguyen

OPTIMIZING VXLAN NETWORK MANAGEMENT

Master of Science thesis
Faculty of Information Technology and Communication Sciences
Examiners: Lecturer Juha Vihervaara
University Instructor, Jani Urama
December 2023

ABSTRACT

Tuan Anh Nguyen: OPTIMIZING VXLAN NETWORK MANAGEMENT

Master of Science thesis

Tampere University

Communication Systems and Networks - Information Technology

December 2023

This thesis presents an in-depth exploration of Virtual Extensible LAN (VXLAN) technology, a key development in modern networking. VXLAN has emerged as a solution to the limitations of traditional VLANs, particularly in scalability and segmentation, which are essential in the context of large-scale data centers and cloud computing environments. The study focuses on the key components of VXLAN, such as VXLAN Tunnel Endpoints (VTEPs) and VXLAN Network Identifiers (VNI) and their role in the facilitation of advanced networking capabilities. It positions VXLAN within the 'Programmable Age' of networking, linking it with important technologies like Software-Defined Networking (SDN) and Network Function Virtualization (NFV), crucial for supporting containerization and the complexities of contemporary multi-tenant environments.

The thesis explores the integration and management of VXLAN networks, highlighting the need for advanced integrated network management tools to handle the complexities inherent in these advanced networks. The exploration covers various tools and technologies, including SDN, Ansible, Prometheus, Grafana, Zabbix, and the PLG stack, focusing on Grafana Loki and Promtail for log aggregation and analysis. The interaction between these tools and their roles in improving operational efficiency, security, and performance within VXLAN networks is carefully examined.

A crucial component of this study is the Proof-of-Concept (PoC) conducted to demonstrate the practical application and interoperability of these tools within a VXLAN environment. The PoC involved the use of two droplets in Digital Ocean to simulate a VXLAN network, employ Open vSwitch for connections, and deploy network management tools within Docker containers. This practical application not only bridged the gap between theoretical research and real-world implementation, but also provided insightful observations into the challenges and solutions inherent in VXLAN network management.

Furthermore, the thesis points out several avenues for future research, including enhanced traffic flow control through SDN controllers in VXLAN, the incorporation of deep learning strategies to improve QoS, and the application of zero-trust security principles in VXLAN environments. Each of these areas presents exciting opportunities to further develop the field of network management.

In summary, this thesis provides a detailed examination of VXLAN technology, its critical role in modern networking, and the importance of an integrated approach to network management. The insights gained from this study make a significant contribution to the network management domain, offering useful perspectives for academic research and practical applications in the changing landscape of network technology.

Keywords: VXLAN, SDN, cloud, optimize, network automation, Ansible, Prometheus, Grafana, Zabbix, ELK stack

The originality of this thesis has been checked using the Turnitin OriginalityCheck service.

USE OF AI IN THE THESIS

The AI tools used in my thesis and the purpose of their use have been described in a numbered list:

1. ChatGPT 4 (OpenAI): Used for idea verification and occasionally to rephrase sections of the thesis to enhance its academic language. Applied to the development of ideas and to improving the language in specific sections.
2. Writefull (Writefull for Overleaf): Employed to paraphrase and improve the academic tone of the thesis. Provided language suggestions to ensure clarity and precision in scholarly writing. Applied mainly in the editing and refinement of the language.
3. Semantic Scholar (Current Version): Used for research and citation purposes. Assisted in identifying the relevant literature and ensuring accurate citations within the thesis. Applied during the research phase.

I am aware that I am totally responsible for the entire content of the thesis, including the parts generated by AI, and accept responsibility for any violations of the ethical standards of publications.

PREFACE

As I reflect on my academic journey at the Faculty of Communication Systems and Networks at Tampere University, this thesis not only has broadened my intellectual horizons but also provided a profound opportunity to understand and shape my professional trajectory. I extend my heartfelt gratitude to Juha Vihervaara, who supported and guided me from the inception of my initial ideas, and to Jani Urama, who, in his role as an examiner, played a significant part in evaluating and refining this work.

The exploration and application of cutting-edge tools within this thesis reflect my desire to implement the knowledge I have acquired, showcasing a seamless transition between network engineering and software engineering. This journey leans toward nurturing the core competencies for a future in DevOps, where integrating management, operations, and development is key to success.

The facilities and educational resources at Tampere University have played an instrumental role in allowing me to achieve academic goals while also fostering personal growth. My appreciation for the university's support is immeasurable.

Finally, I express my deep gratitude to my family and friends, whose unwavering inspiration and strength have propelled me forward.

Tampere, 1st December 2023

Tuan Anh Nguyen

CONTENTS

1.	Introduction	1
2.	VXLAN Technology and Its Role in Modern Networks	3
2.1	Introduction of VXLAN Network Virtualization.	3
2.1.1	VXLAN Network Components: VTEPs and VNIs	3
2.1.2	VXLAN's Role in Network Virtualization	4
2.1.3	VXLAN Encapsulation Process	5
2.2	Advantages and Challenges of VXLAN	7
2.2.1	Advantages and Applicability of VXLAN	7
2.2.2	Challenges in VXLAN Deployment	9
2.3	Network Management Challenges in VXLAN Deployment.	9
3.	Aspects of VXLAN Management	10
3.1	Software-Defined Networking (SDN) in VXLAN	10
3.1.1	Introduction to SDN	10
3.1.2	Architecture and Functioning of SDN	11
3.1.3	Models of SDN and Their Practical Implementations	16
3.1.4	The Synergy of SDN and VXLAN in Cloud Computing	16
3.2	Network Automation	19
3.2.1	Introduction to Network Automation	19
3.2.2	Comparative Analysis of Network Automation Tools	20
3.3	Monitoring, Logging, and Visualization in VXLAN Management	22
3.3.1	Overview of Monitoring, Alerting, and Visualization.	22
3.3.2	Introduction to Logging and Its Significance	23
3.3.3	Evaluating Modern Monitoring and Logging Solutions: A Comparative Analysis	23
3.3.4	Alerting with Prometheus' Alertmanager	26
3.4	Additional Aspects of VXLAN Management	26
3.4.1	Security and Compliance in VXLAN.	26
3.4.2	Scalability and Performance in VXLAN	27
3.4.3	VXLAN Reliability: Redundancy and failure mechanisms	27
3.5	Gaps and Opportunities in Current VXLAN Management Approaches	27
4.	Tools for VXLAN Network Management Optimization	29
4.1	Ansible in Automated VXLAN Network Configuration.	29
4.2	ELK Stack: The Established Commercial Standard for VXLAN Logging	31

4.3	PLG Stack with Grafana Loki: The Open Source Innovator in Logging Solutions	32
4.4	Prometheus for In-depth VXLAN Monitoring and Alert	33
4.4.1	Prometheus monitoring	33
4.4.2	Prometheus Alertmanager	34
4.5	Grafana: Advanced Dashboards for Network Analysis	35
4.6	Zabbix for Comprehensive Network Monitoring	37
4.7	Unified Approach to VXLAN Management	38
4.8	Proof of Concept (PoC): A Simplified Implementation	40
4.8.1	Implementation and Configuration Details	40
4.8.2	Monitoring and Visualization Outcomes	44
5.	Conclusion	51
	References	53
	Appendix A: YAML configuration files	60
A.1	Alertmanager Configuration	60
A.2	Prometheus Configuration	61
A.3	Zabbix Agent Configuration	63

LIST OF FIGURES

2.1	A detailed schematic representation of VXLAN encapsulation and packet flow [81].	4
2.2	The underlay-overlay paradigm in VXLAN (Modified from [62]).	5
2.3	The 2-tier-spine-leaf architecture adapted for VXLAN.	6
2.4	A detailed depiction of the VXLAN encapsulation process [62].	7
3.1	Traditional vs. SDN: SDN simplifies management and delivers middlebox services through controller applications (Modified from [27]).	11
3.2	Layered view of networking functionality	12
3.3	An illustrative depiction of the SDN architecture emphasizing the role and interaction between the SDN controller, network applications, and SDN device flow tables in the context of OpenFlow protocols [27].	13
3.4	Overview of SDN Architecture.	14
3.5	Comparative Overview of Proactive and Passive VM Migration Mechanisms in SDN-integrated VXLAN Architecture [91]	18
3.6	SDN-based VXLAN Architecture Depicting the VM Migration Procedure and Establishment of New Connections [91]	19
3.7	The Forrester Wave™: Infrastructure Automation, Q1 2023[77].	21
3.8	Differentiation between push and pull based monitoring methodologies. . .	25
4.1	Ansible Architecture.	29
4.2	The ELK data processing workflow in a VXLAN context, illustrating the comprehensive logging capabilities of the stack (Adapted from [73])	31
4.3	The architecture of Grafana Loki	32
4.4	Prometheus architecture (Modified from [54]).	33
4.5	High-Level Overview of Prometheus Alertmanager (Modified from [5]). . . .	35
4.6	Grafana's Gauge Dashboard Representation [17]	36
4.7	Grafana's Time Series Dashboard Visualization [17]	37
4.8	Zabbix architecture (Modified from [88]).	38
4.9	Digital Ocean Droplets configuration for the PoC	41
4.10	PoC Network Infrastructure Diagram with VXLAN and Tools	42
4.11	Graphical visualization of metrics in Prometheus.	44
4.12	Service Discovery in Prometheus.	45
4.13	Monitoring targets in Prometheus.	46
4.14	Zabbix monitoring dashboard.	47

4.15 Host addition in Zabbix monitoring.	48
4.16 Problems and alerts in Zabbix.	48
4.17 Log aggregation in Grafana Loki for container logs.	49
4.18 Grafana dashboard visualizing Prometheus metrics.	50
A.1 Alertmanager YAML configuration file.	61
A.2 Prometheus YAML configuration file.	62
A.3 Zabbix agent YAML configuration file.	64

LIST OF TABLES

2.1	Evolution of Networking Eras with an Emphasis on Technologies Including VXLAN, adapted from [62].	4
2.2	VXLAN Packet Structure	8
3.1	SDN Models and Their Practical Implementations	17
3.2	Comparison of network management approaches, adapted from [9]	22
4.1	Components of Ansible and Their Descriptions.	30
4.2	Containers and Their Functions on AN-DC1	43
4.3	Containers and Their Functions on AN-DC2	43

LIST OF SYMBOLS AND ABBREVIATIONS

AWS	Amazon Web Services
Azure	Microsoft Azure Cloud Service
BUM	Broadcast, Unknown Unicast, and Multicast
CLI	Command Line Interface
CMDB	Configuration Management Database, a repository that acts as a data store for IT installations
CNCF	Cloud Native Computing Foundation, a project that aims to build sustainable ecosystems for cloud native software
DC	Data Center
DSCP	Differentiated Services Code Point
ECMP	Equal-Cost Multi-Path
EIGRP	Enhanced Interior Gateway Routing Protocol
ELK	Elasticsearch, Logstash, Kibana - a set of three open-source products
EVPN	Ethernet VPN - Ethernet Virtual Private Network
FD	Forwarding Devices
ForCES	Forwarding and Control Element Separation
GitHub	A cloud-based hosting service for Git repositories
Go	An open-source programming language also known as Golang
IaaS	Infrastructure as Code
IDS/IPS	Intrusion Detection System / Intrusion Prevention System
IETF	Internet Engineering Task Force
KVM	Kernel-based Virtual Machine
MC-LAG	Multi-chassis Link Aggregation
NAS	Network Automation Server
NetOps	Network Operations
NFV	Network Functions Virtualization
NIB	Network Information Base

NOS	Network Operating System
NV	Network Virtualization
ODL	OpenDaylight, an open-source project that focuses on advancing SDN and NFV with a developer-friendly platform.
OTV	Overlay Transport Virtualization
PoC	Proof of Concept
POF	Protocol-Oblivious Forwarding
Programmable Age	A contemporary era characterized by advanced automation, programmability, and virtualized networking contexts
PromQL	Prometheus Query Language, a robust querying language used to interact with metrics in real-time
REST API	Representational State Transfer API.
SaaS	Software as a Service
SDN	Software-Defined Networking
SDN (OF)	Software-Defined Networking (OpenFlow)
STP	Spanning Tree Protocol
STT	Stateless Transport Tunneling
TICK	Telegraf, InfluxDB, Chronograph, and Kapacitor - an open-source stack of tools
TOR	Top Of Rack
VCS	Version Control System
VLANs	Virtual Local Area Networks
VM	Virtual Machine
VNI	VXLAN Network Identifier
VPC	Virtual Private Cloud
VTEP	VXLAN Tunnel Endpoint
VXLAN	Virtual Extensible Local Area Network
WRED	Weighted Random Early Detection

1. INTRODUCTION

The development of Virtual Extensible LAN (VXLAN) technology marks a significant step in network architecture, addressing the increasing needs for better scalability and segmentation in modern data centers. This advancement goes beyond the limitations of traditional VLANs, introducing a new level of network flexibility and efficiency. However, implementing VXLAN introduces new complexities in network management, which calls for a comprehensive and integrated approach to effectively manage these sophisticated networks.

A key focus of this thesis is the exploration of advanced network management tools in VXLAN infrastructures, emphasizing their role in improving network efficiency, security, and performance. Software-Defined Networking (SDN) is identified as a critical element in this context, offering a new approach that significantly improves network management by separating control and forwarding functions. This separation is especially useful in complex VXLAN setups, allowing more dynamic and adaptable network configurations.

Ansible's role in automating network operations is noteworthy. Its integration into VXLAN networks is crucial to reduce operational complexities and ensure configuration reliability, key factors in large-scale network environments.

Prometheus, Grafana, and Alertmanager are important in the monitoring and visualization aspect of networks. Prometheus stands out by collecting network metrics in real time, laying a strong foundation for network analysis. Grafana complements Prometheus with sophisticated data visualization tools, allowing network administrators to obtain a complete view of network performance, aiding in quick and informed decision-making. Alertmanager, on the contrary, is essential in managing network alerts from Prometheus. Its efficient handling of notifications ensures that network operators can react quickly to issues, maintaining network stability and performance.

Integration of the PLG stack (Promtail, Loki, and Grafana) is also a major part of this study, due to its effectiveness in log aggregation and analysis in VXLAN networks. Grafana Loki is notable for its ability to handle large amounts of log data, a typical challenge in VXLAN environments.

In addition to these tools, the thesis recognizes the potential of the ELK stack (Elasticsearch, Logstash, and Kibana) as a strong logging solution. Despite its capabilities, the

commercial nature of the ELK stack and associated costs have led to its exclusion from the practical component of the thesis. This decision reflects a preference for more accessible and cost-effective tools for the management of the VXLAN network.

The practical application of these tools is shown in a detailed Proof-of-Concept (PoC). This PoC uses two droplets on the Digital Ocean platform to simulate a VXLAN network, establishing VXLAN connections using Open vSwitch. It also includes the deployment of various network management tools within Docker containers, providing a flexible and scalable environment for testing. This PoC validates the theoretical aspects and offers practical information on managing VXLAN networks.

In conclusion, this thesis presents a detailed study of VXLAN technology and its essential role in modern networking. It supports an integrated approach to network management, showing the effectiveness of combining tools such as SDN, Ansible, Prometheus, Grafana, the PLG stack, and Alertmanager. This approach not only addresses the challenges in VXLAN network management, but also prepares the way for future developments in this fast-changing field. The insights from this study are valuable both for academic research and practical applications.

2. VXLAN TECHNOLOGY AND ITS ROLE IN MODERN NETWORKS

2.1 Introduction of VXLAN Network Virtualization

The introduction of VXLAN technology has been a key development in network architecture, providing a solution to the scalability and segmentation limitations that traditional VLANs exhibit. Conceived through the collaborative efforts of industry leaders such as VMware, Arista Networks, and Cisco, VXLAN was designed to extend Layer 2 segments across Layer 3 infrastructures, accommodating the increasing demands of complex, large-scale data centers. This approach has effectively addressed the critical limitations of previous network technologies, such as the Spanning Tree Protocol, and has alleviated issues associated with server virtualization, including the increasing number of MAC addresses in switch tables [62, 86].

2.1.1 VXLAN Network Components: VTEPs and VNIs

At the core of the VXLAN architecture are VXLAN Tunnel Endpoints (VTEPs) and VXLAN Network Identifiers (VNIs). VTEPs are responsible for encapsulating native data frames into VXLAN packets, enabling their transit across the network and their subsequent decapsulation at the destination. This encapsulation and decapsulation process is effectively illustrated in Figure 2.1, which depicts a tunnel between two Top of Rack (ToR) switches transporting data frames via an underlying IP network. In particular, VTEPs can be implemented as standalone network devices, such as Huawei CloudEngine series switches, or as virtual switches on servers.

The depiction in Figure 2.1 clarifies the role of the source VTEP in wrapping original data frames in VXLAN packets for IP network transmission and the function of the destination VTEP in reverting these packets to their initial form to reach the destination server.

VNIs play a crucial role in VXLAN by offering a significantly larger identifier space compared to traditional VLAN IDs. Where VLAN IDs are limited to a 12-bit field, VNIs provide a 24-bit field, which greatly expands the network's segmentation capabilities. This is essential for the modern data center environment that requires the accommodation of nu-

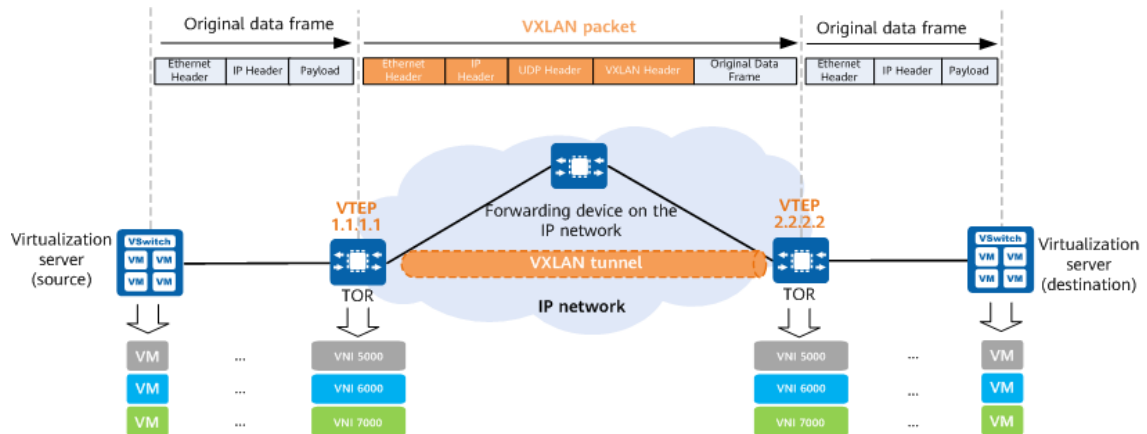


Figure 2.1. A detailed schematic representation of VXLAN encapsulation and packet flow [81].

merous tenants. This process is captured in Figure 2.4, where encapsulation includes the addition of a 24-bit VNI to each VXLAN packet, ensuring isolation of tenant environments. Moreover, VNIs contribute to Layer 2 isolation, preventing inter-tenant communication at this layer. For operational purposes, VNIs are differentiated into two types: Layer 2 VNIs are associated with bridge domains for intra-subnet VXLAN packet forwarding, whereas Layer 3 VNIs are tied to VPN instances to route VXLAN packets across subnets, as detailed in the EVPN documentation.

2.1.2 VXLAN's Role in Network Virtualization

The historical progression of the network architecture is summarized in Table 2.1. This table provides an overview of the evolution from basic VLAN and Spanning Tree protocols to more sophisticated systems such as Software-Defined Networking (SDN), Multiprotocol BGP, and VXLAN. The table delineates the impacts of these technologies on network design and operation, situating VXLAN within the 'Programmable Age' and highlighting its facilitation of key technologies like Network Function Virtualization (NFV) and containerization, which are essential in contemporary networking.

Table 2.1. Evolution of Networking Eras with an Emphasis on Technologies Including VXLAN, adapted from [62].

Era	Stone Age	Bronze Age	The Renaissance	Programmable Age
Technologies / Protocols	Spanning Tree, VLANs	Routing Protocols, WAN Design, IP-magedon	SDN , OpenFlow, Controllers, Overlays, MP-BGP, VXLAN , Micro-Segmentation, White Box	Cloud, Python, REST / APIs, NETCONF / YANG, "Fabrics", Network Function Virtualization (NFV), DevOps , Containers

The role of VXLAN in network virtualization supports both existing functionality and the future development of data centers. By enabling scalable and efficient network design, VXLAN helps in the swift deployment of network services, thus improving management and operational efficiency. The sophisticated encapsulation/de-encapsulation process, along with VNI's extensive isolation capabilities, demonstrates the complexity and critical nature of VXLAN in modern and future network environments [81].

VXLAN is fundamental in network virtualization, maintaining the original frame structure while allowing it to traverse through an IP network. This capability is crucial for allowing separate virtual network segments to co-exist on the same physical network infrastructure, thereby enhancing overall network efficiency.

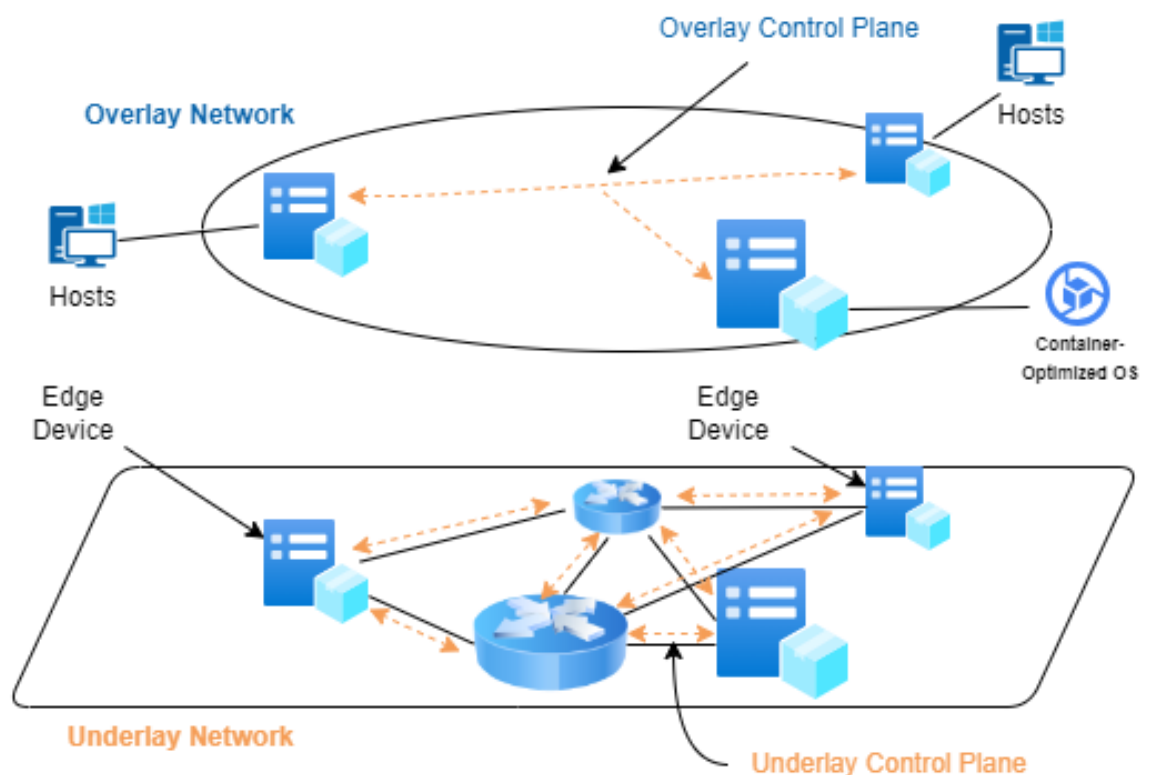


Figure 2.2. The underlay-overlay paradigm in VXLAN (Modified from [62]).

The efficacy of the model is rooted in its underlay-overlay architecture, as outlined in Figure 2.2, which differentiates the physical network (underlay) from the virtual networks (overlay) that carry customer traffic. This dual-layered approach aligns with the Spine-Leaf network architecture, providing a scalable and adaptable data center topology, as depicted in Figure 2.3.

2.1.3 VXLAN Encapsulation Process

In networking, VXLAN represents a significant advancement over traditional VLAN (Virtual LAN) technologies, providing enhanced capabilities in network isolation and scala-

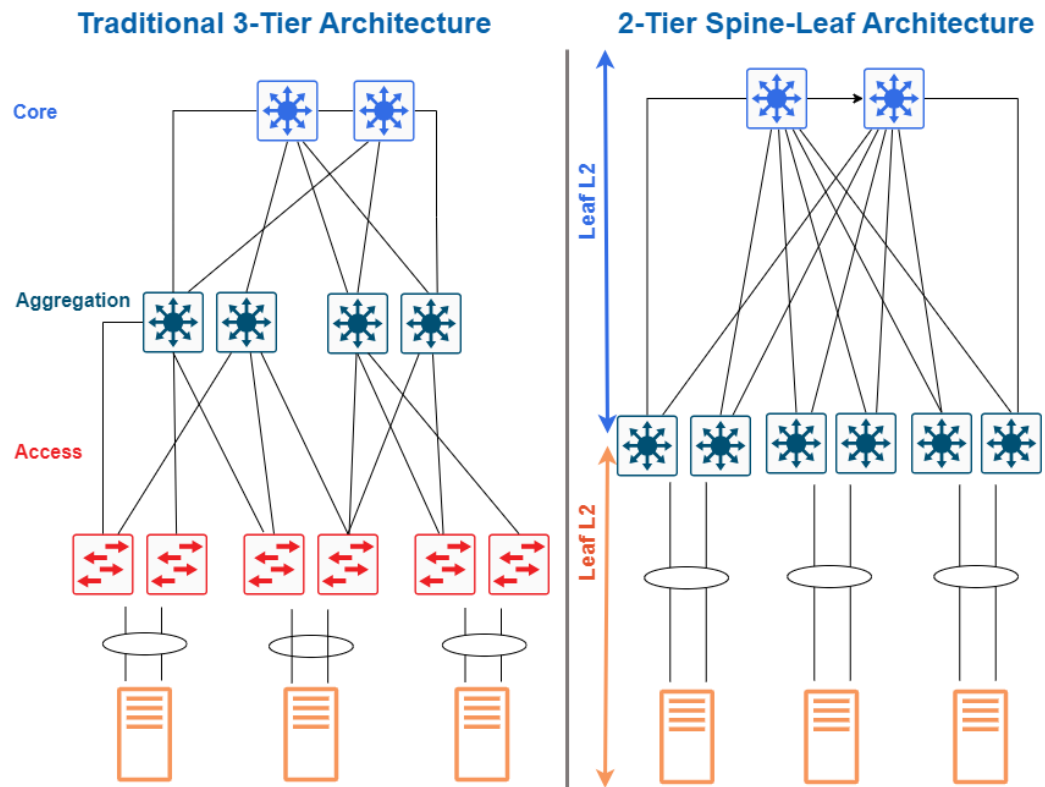


Figure 2.3. The 2-tier-spine-leaf architecture adapted for VXLAN.

bility. The VXLAN encapsulation mechanism extends the scope of the network both in scale and in the flexibility of virtual machine (VM) migrations across Layer 2 networks. This process, depicted in Figure 2.4, involves the addition of headers to the original Ethernet frame, which increases the network's tenant isolation capacity and overcomes the limitations inherent to VLANs.

The VXLAN protocol encapsulates a traditional MAC frame, augmenting it with additional headers to form a VXLAN packet, as detailed in Table 2.2. This process includes the original frame components and adds the layers necessary for the encapsulation of VXLAN, ensuring proper identification and routing within a VXLAN environment [37, 44, 21, 86, 62].

The structured encapsulation method employed by VXLAN, particularly the implementation of the VXLAN header and outer headers, bestows the protocol with marked improvements in terms of scalability and the ability to extend Layer 2 networks across Layer 3 infrastructures. These features allow VXLAN to bypass the limitations associated with traditional VLANs, eliminate the reliance on the Spanning Tree Protocol, and improve efficient network spanning operations [37, 44, 21, 86].

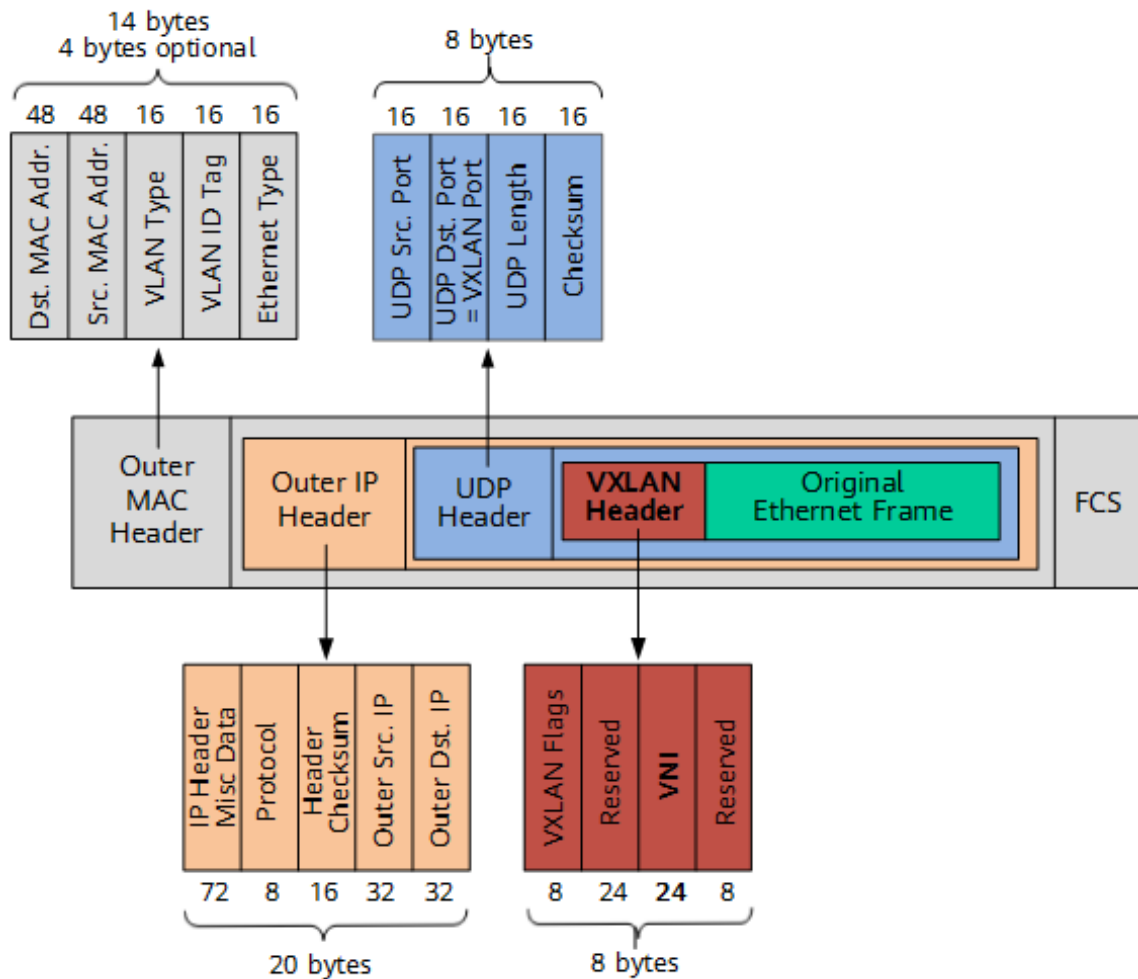


Figure 2.4. A detailed depiction of the VXLAN encapsulation process [62].

2.2 Advantages and Challenges of VXLAN

VXLAN has emerged as a cornerstone in the realm of network virtualization, with its benefits and applications resonating well within the dynamic requirements of contemporary cloud computing and data center environments.

2.2.1 Advantages and Applicability of VXLAN

With its inception, VXLAN introduced an architecture that significantly improved the scale of network segmentation. Supporting up to 16 million logical networks through a 24-bit VNI, VXLAN allows the creation of extensive Layer 2 virtual networks across a common physical network infrastructure [78]. This is particularly beneficial for cloud data centers, enabling smooth migration of virtual machines across different physical hosts and addressing the limitations of traditional VLANs, which are restricted by their 12-bit identifier space [37].

Moreover, VXLAN's optimization of routing and connectivity translates to superior network

Table 2.2. *VXLAN Packet Structure*

Packet Component	Description
VXLAN Header	This header consists of an 8-byte field. The initial 8 bits are designated as flags, where the 'I' flag must be set to 1, indicating the presence of a valid VNI. The next 24 bits represent the VXLAN segment ID or VNI, which uniquely identifies the overlay network within the VXLAN environment. This is succeeded by reserved fields that should be set to zero during transmission and are ignored at the receiving end.
Outer UDP Header	The outer UDP header assigns a well-recognized destination port, which is typically 4789 as specified by the Internet Assigned Numbers Authority (IANA). While this port number should be used by default, the UDP header allows for configurable port numbers to support different operational environments. The UDP source port is dynamically selected, commonly based on the hash of the encapsulated packet's headers. This strategy introduces entropy that is beneficial for equal-cost multipath (ECMP) routing and load balancing. The presence of a UDP checksum is optional; if used, it must be accurately calculated to ensure data integrity.
Outer IP Header	This header specifies the source and destination IP addresses, which correspond to the VTEPs connected to the source and destination VMs, respectively. The destination IP can be unicast, for point-to-point communication, or multicast, to support scenarios that involve multdestination traffic.
Outer Ethernet Header	The outermost header may carry an optional VLAN tag, which facilitates traffic segregation. It also includes the MAC addresses that direct the encapsulated packet either to the destination VTEP or towards an intermediate routing device.

performance, as it circumvents the limitations of traditional MAC address tables in TOR switches by utilizing Layer 3 networks for Layer 2 frame transport. This not only improves network efficiency, but also strengthens security by segmenting traffic, thus providing a robust communication channel for virtual machines regardless of their physical location [78, 21].

Furthermore, the independence of VXLAN from broadcast, unicast unknown, and multicast traffic enhances its adaptability. This independence makes VXLAN a versatile solution for extending virtual machines across large networks, ensuring traffic isolation in multi-tenant architectures, and extending data centers across distributed geographical

locations [78].

2.2.2 Challenges in VXLAN Deployment

Despite these advantages, the deployment of VXLAN is not without challenges. The setup of VXLAN networks is complex and requires a comprehensive strategy for managing MAC and IP address databases. This complexity increases with the scale of the deployment, which requires robust control mechanisms, especially when implementing centralized overlay network controllers [87, 80].

The encapsulation process introduces additional overhead, which can potentially degrade network throughput. The need for underlying multicast support for efficient BUM traffic handling adds to the complexity of deployment. Moreover, since various vendors offer various VXLAN solutions, interoperability issues can arise, which require continuous efforts to validate and comply with security standards [87, 80].

2.3 Network Management Challenges in VXLAN Deployment

The VXLAN deployment heralds a significant shift in networking paradigms, much like the way cloud formations redefine the skyscape. The IETF's endorsement of VXLAN through protocols like RFC 7348 facilitated an evolution in tunneling mechanisms, enabling the encapsulation of data frames with extended headers that encompass outer IP and UDP addresses. This evolution is a testament to VXLAN's ability to cater to an extensive array of tenants within a singular cloud ecosystem [37, 91].

However, the administration of VXLAN networks is accompanied by its own set of complexities. Understanding the intricate dance between Layer 2 and Layer 3 architectures becomes imperative as data center numbers burgeon. The growing number of VNIs requires a more sophisticated approach to network management, an endeavor that conventional monitoring tools are not equipped to handle [91]. Ensuring the security of encapsulated traffic and preventing unauthorized access becomes increasingly critical. Integrating with systems such as SDN is essential to navigate the challenges associated with large-scale VXLAN deployments effectively. Furthermore, the amplification of signaling traffic, especially through IP multicasting, presents a challenge that must be managed with care [43].

To summarize, the transformative potential of VXLAN in data networking is undeniable. However, achieving its full capabilities requires meticulous planning, a deep understanding of the technology, and an adaptive management approach that can meet its sophisticated deployment challenges.

3. ASPECTS OF VXLAN MANAGEMENT

3.1 Software-Defined Networking (SDN) in VXLAN

3.1.1 Introduction to SDN

SDN has gained significant traction in recent times as a transformative approach to designing and managing networks. Its defining characteristic, regardless of its various definitions or applications such as OpenFlow-based SDN, is the clear delineation between the network control and data transmission functions [70].

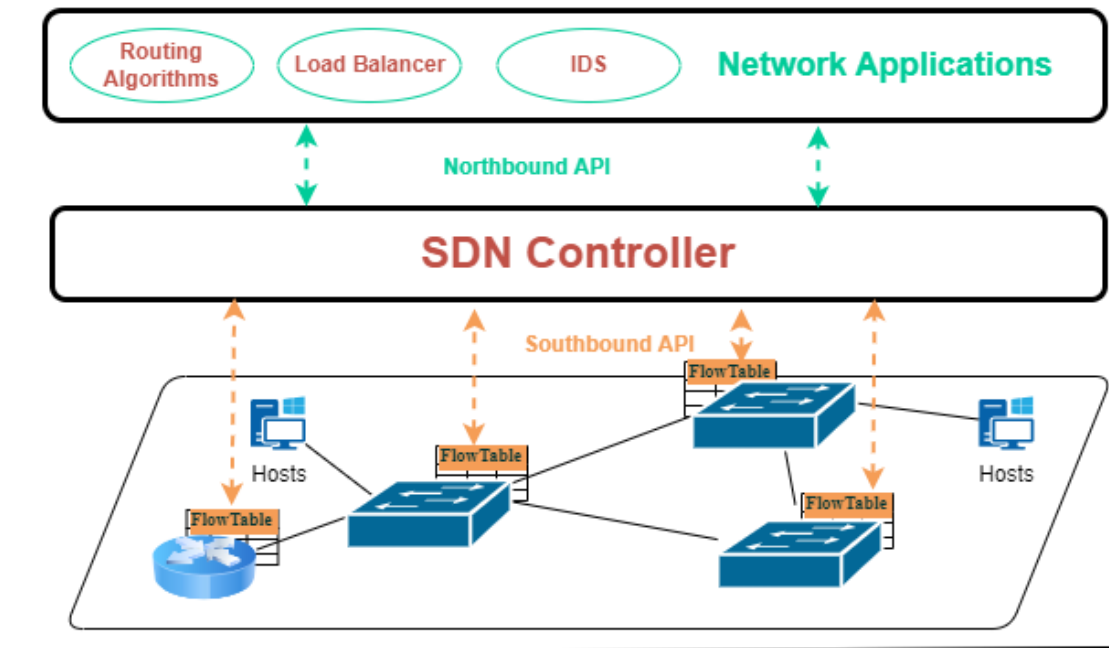
Figure 3.1 provides a comparative visual between conventional networking and software-defined networking. The conventional setup connects devices such as routers, switches, and servers. However, the SDN model places emphasis on a centralized "SDN controller" that communicates directly with these devices. This controller is at the helm of various network applications, including, but not limited to, MAC Learning, Routing Algorithms, Intrusion Detection Systems, and Load Balancers.

A centralized architecture, as seen in SDN, offers multiple advantages. For example, the unified abstraction model, facilitated by the control platform and network programming languages, streamlines software development and guarantees consistency throughout the network [10]. Provides a holistic network perspective, enabling applications to take advantage of consistent network data. This data access helps in informed policy formulation and taps into the software modules of the shared control plane [10].

The visual emphasis on the SDN controller in Figure 3.1 highlights its ability to execute a wide range of actions throughout the network. This includes reconfiguring forwarding devices and bypassing complex strategy formulation for new function placements [10]. The distinct separation of the control and data planes enables the independent optimization of each, giving rise to specialized devices and efficient servers. This structure reduces service providers' capital expenditure (CAPEX) and streamlines network orchestration and monitoring by centralizing network information, applications, and services [79, 27].

In conclusion, the SDN overarching framework improves the cohesion of various applications, fostering a more synchronized networking environment [10]. With the foundational understanding of SDN established, subsequent sections will further delve into SDN within

Software-Defined Networking



Traditional Networking

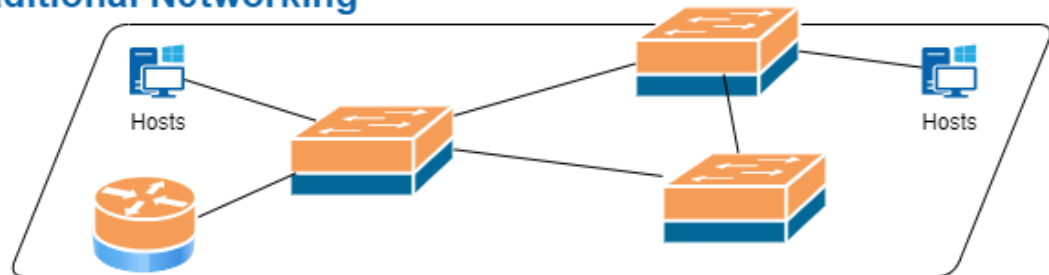


Figure 3.1. Traditional vs. SDN: SDN simplifies management and delivers middlebox services through controller applications (Modified from [27]).

VXLAN ecosystems and its pivotal role in managing VXLAN's challenges and threats.

3.1.2 Architecture and Functioning of SDN

Within computer networking, three critical planes characterize its core functionalities: data, control, and management, depicted in Figure 3.2. The data plane predominantly resides within the domain of networking hardware and is focused on efficient data transmission. On the contrary, the control plane embodies the protocols responsible for shaping the data-centric forwarding databases. The management plane encompasses software services, prominently including those related to the simple network management protocol (SNMP) [49]. Its central role is to facilitate remote supervision and to configure control operations. Drawing a distinction between the planes, the management plane defines the

network policy, the control plane enforces this policy, and the data plane operationalizes it by directing data accordingly. [27]

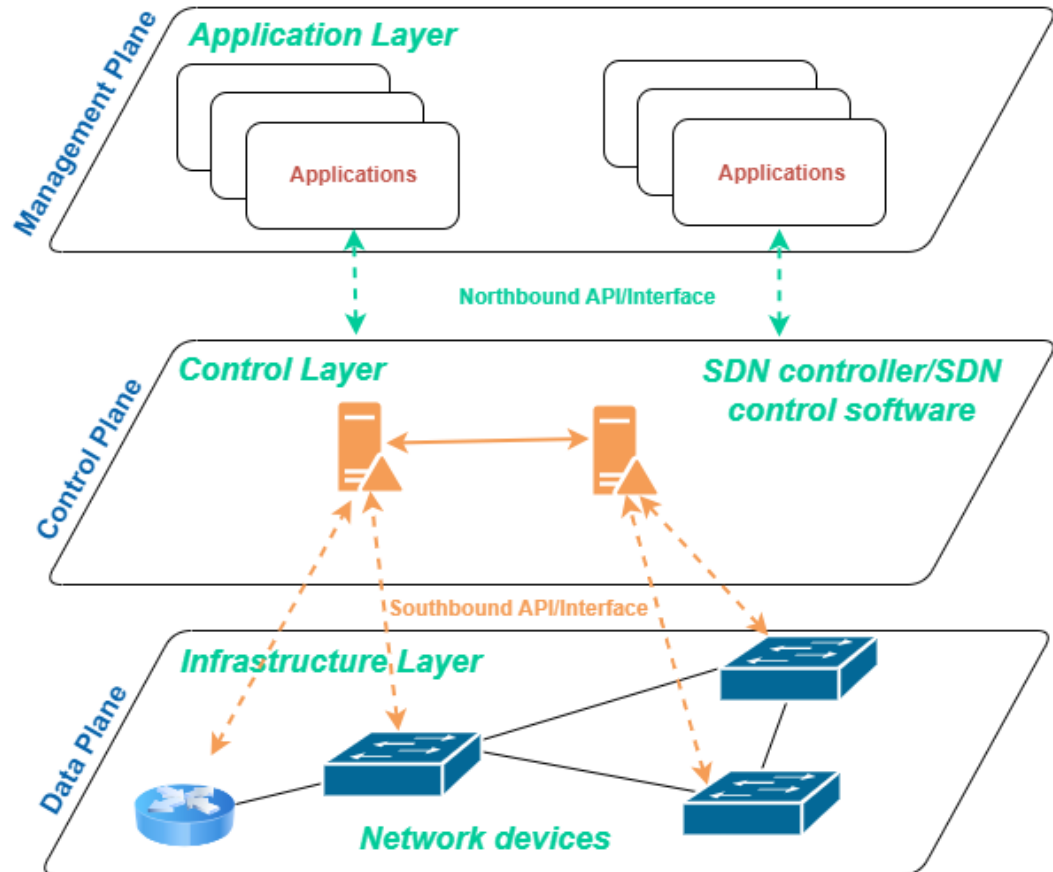


Figure 3.2. Layered view of networking functionality

Control Plane

Central to the SDN framework is the SDN controller, which offers a standardized central programming interface that is crucial for network monitoring and management. The controller facilitates access to the Network Information Base (NIB) for northbound applications. This NIB is enriched with essential data elements, from nodes and links to topology and statistics, empowering applications with the insights required for sound decision making. Operating on commodity servers, the controller guarantees a unified perspective of the network landscape.

Figure 3.3 illustrates the intricate design and competencies of the SDN controller. Serving

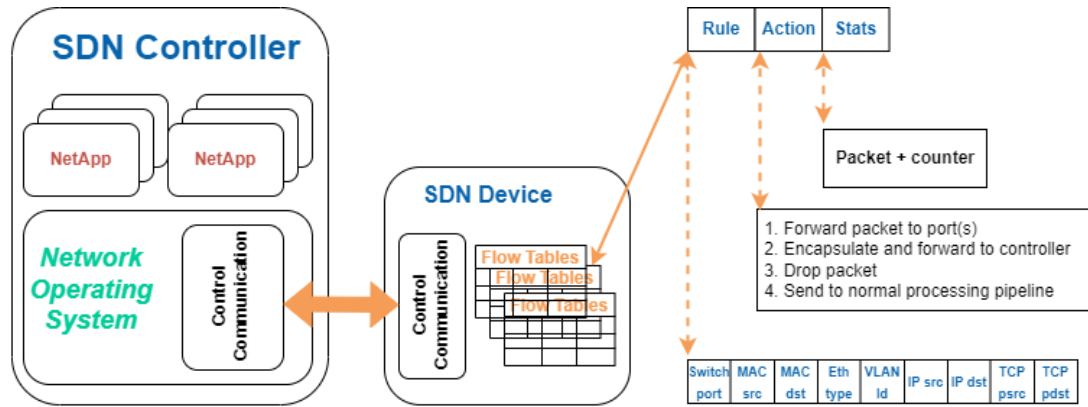


Figure 3.3. An illustrative depiction of the SDN architecture emphasizing the role and interaction between the SDN controller, network applications, and SDN device flow tables in the context of OpenFlow protocols [27].

as an intermediary, the controller bridges high-level application paradigms with detailed network operations. Adapts high-tier terminologies to their respective granular network addresses. A comprehensive SDN controller amalgamates protocol handlers for network protocol management, applications leveraging network information, and a collection of libraries tailored for varying southbound interfaces. Globally, the controller seamlessly interfaces with dedicated network orchestration tools and robust support systems, laying the foundation for pioneering customized solutions [90, 27].

In the SDN context, conforming packets with flow entry induce the refresh of relevant counters and activate the stipulated action. Packets that do not find a match are directed to the controller for nuanced processing. Through these packet transitions and flow initiations, the controller meticulously crafts its perception of the network, orchestrating traffic pathways. It goes beyond packet interpretation to accommodate node and link failures, recalibrating routes to accommodate emerging changes.

Figure 3.4 presents the centrality of the controller, which orchestrates communication with data plane elements using a refined application programming interface (API). The OpenFlow protocol serves as a salient example of such interfaces [41]. Within an OpenFlow switch, flow tables encompass directives for packet processing. These directives, shaped by the controller, identify diverse types of traffic and administer actions, ranging from forwarding and modification to packet dismissal. Under the controller's aegis, an OpenFlow switch can assume manifold responsibilities, be it routers, switches, or specialized roles. The underlying ethos of SDN emphasizes the separation between policy formulation, hardware orchestration, and traffic management. This segregation improves operational nimbleness, simplifies network control, and promotes novel networking abstractions, revitalizing network management, and fostering innovation [27].

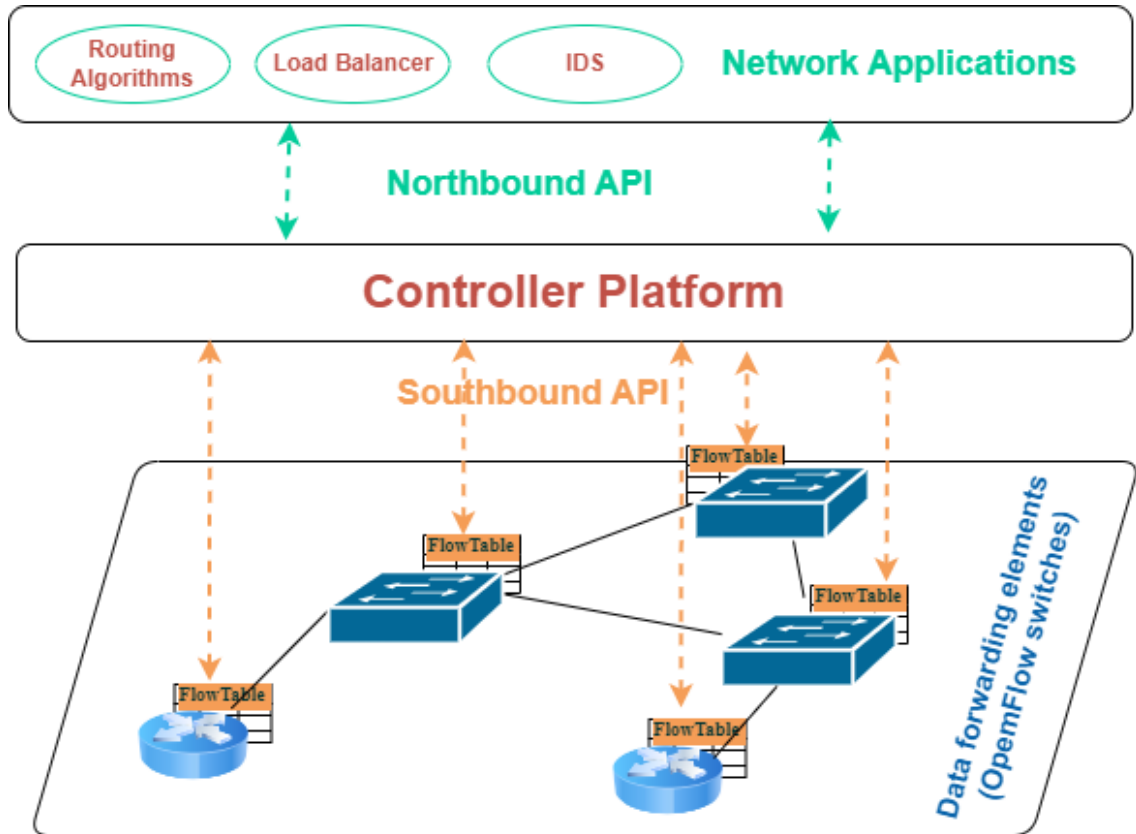


Figure 3.4. Overview of SDN Architecture.

Data Plane

The data plane is an integral layer of the SDN architecture, predominantly represented by Forwarding Devices (FD). Whether manifested as hardware or software constructs, these FDs play a crucial role in the realization of the determinations of the network. They process incoming packets on the basis of a predefined set of instructions, commonly termed flow rules. Upon reception of these packets, FDs might forward them to designated ports, modify certain packet attributes, or undertake other specified actions, all contingent on the instruction they adhere to. [27]

Several protocols facilitate the exchange of instructions between the SDN controller and the FDs. Among these, OpenFlow [41] has gained notable traction due to its pioneering nature in the SDN realm. However, other southbound interfaces, such as ForCES [14] and protocol-oblivious forwarding (POF) [71], also play a crucial role. It is through these channels that the SDN controllers impart directives that guide the operational trajectory of the FDs. [27]

To further elucidate the importance of FDs, one must consider the dynamics of a network, where seamless data transmission and reception are paramount. In traditional network

configurations, static configurations sometimes hindered timely and efficient responses to data flow changes. With the advent of SDN and the decentralized nature of FDs, there is a newfound agility. The FDs, working in tandem with the controllers via protocols like OpenFlow, ensure adaptive, resilient, and efficient data handling, paving the way for robust and flexible network architectures. [27]

In essence, the harmonious collaboration between the controller and the FDs, facilitated by southbound interfaces, is foundational to the SDN model. This synergy ensures that the data plane remains adaptive, efficient, and attuned to the evolving needs of network traffic, marking a transformative phase in network management and operations.

Southbound and Northbound Interfaces

In the SDN paradigm, two pivotal interfaces emerge: the Northbound and Southbound interfaces. These interfaces not only form the crux of SDN's architectural hierarchy, but also epitomize the communication flow within the SDN ecosystem.

The Northbound Interface, often called the Application Layer, bridges the application plane with the SDN controller, encompassing business, networking, and monitoring functionalities [79]. Through this interface, applications convey their requirements, ensuring that SDN aligns with desired service quality, security standards, and managerial efficiency. However, variations in the definitions of the northbound interface across SDN controllers complicate the scenario. While platforms such as Cisco Systems adopt protocols such as RESTconf, Easy-QoS, and PnP [27], others, like OpenDayLight (ODL), prefer RESTCONF, NETCONF, and AMQP [64].

Among the diverse protocols, the Representational State Transfer (REST) API has emerged as a forerunner, chiefly operating through HTTP. By transporting configuration and performance data, and through the integration of RESTful protocols in Open APIs like OpenStack and Virtual Private Cloud (VPC) API, it has facilitated a standardized interaction with network resources, promoting both proprietary and third-party application development.

Conversely, the Southbound Interface, synonymous with the Infrastructure Layer, acts as the conduit connecting the control plane with the data plane. Here, the SDN controller interacts with network nodes such as switches. Protocols that pave the way for this interface include OpenFlow from the Open Networking Foundation (ONF), Cisco OpFlex, and Open vSwitch. OpenFlow, in particular, operating unencrypted over TCP or through the encrypted TLS protocol, reigns supreme in the Southbound interface landscape. [90]

The essence of southbound APIs lies in their ability to offer users unparalleled control over their networks. By enabling the SDN controller to dynamically manage the internal flow table of routers and switches, these APIs optimize real-time traffic responsiveness

[27]. The shift from viewing SDN as merely an academic endeavor to recognizing its commercial potential is profound. The centralization of the network's control logic, distinct from the underlying hardware, underscores SDN's promise for agility and centralized governance.

Highlighting the Open-Flow protocol accentuates the transformative nature of SDN. By strategically decoupling the control and data planes, Open Flow has instigated a significant shift in network management approaches [3]. Central to its success are OF messages that synchronize the SDN controller and OF switches to ensure consistent entry in the flow table [3]. The emergence of the Open Daylight (ODL) SDN controller, backed by industry stalwarts such as IBM, Cisco, and Juniper, serves as a testament to the evolving SDN landscape. The open source Java-based architecture of ODL, characterized by its adaptability and durability, showcases the zenith of SDN excellence. [65]

3.1.3 Models of SDN and Their Practical Implementations

The advent of SDN heralded a transformative era in network design and operations, emphasizing the decoupling of control and data planes. Various SDN models have emerged to meet different requirements and use cases. Table 3.1 summarizes these models, their descriptions and practical implementations [69, 83, 84, 85].

3.1.4 The Synergy of SDN and VXLAN in Cloud Computing

Contemporary cloud computing paradigms have seen a prominent emphasis on the integration of SDN with VXLAN architecture. The profound work of Zhao et al. (2017) introduced a groundbreaking three-tier SDN-based VXLAN model, promising to optimize communications, especially during virtual machine migrations [91]. This model offers an economical solution to IP multicasting challenges while capturing the benefits of SDN-VXLAN integration through improved threshold management and enhanced load balancing [91].

Kumar et al. (2017) provided an exposition on the transformative nature of SDN, emphasizing its potential to transform traditional network operations such as routing and switching into software-oriented challenges, pushing beyond the restrictions of standard network management systems [28].

The union of SDN with VXLAN has truly redefined network architectures. By merging the SDN control plane with VXLAN's data-centric design, this collaboration benefits from global oversight and increased programmability, as detailed by Lee et al. (2015) [31]. As such, it further enhances the capabilities of VXLAN.[91]

In terms of IP multicasting, the SDN-integrated VXLAN differs fundamentally from its tra-

Table 3.1. SDN Models and Their Practical Implementations

SDN Model	Description	Practical Implementations
Open SDN	Based on the OpenFlow protocol, the Open SDN model facilitates direct interactions between the SDN controller and switches through the Southbound API. This approach is a true embodiment of the principles of SDN, focusing on controlled data flows [69, 83].	OpenDaylight (ODL), Floodlight, Ryu
SDN via APIs (API SDN)	Moving beyond the constraints of OpenFlow, the API SDN model seamlessly integrates with conventional networking hardware. It capitalizes on tools like SNMP, CLI, and modern REST APIs. Although the architecture is generally open, certain APIs might be vendor-specific [69, 83, 85].	Cisco APIC-EM, Arista EOS
SDN Overlay Model	Instead of directly engaging hardware, the overlay model overlays a virtual network atop existing infrastructures. By establishing dynamic tunnels, it ensures flexibility in data flow without compromising the integrity of the core network [83, 84].	Nutanix Flow, Cumulus VX, VMware NSX
Hybrid SDN	Combining elements of traditional networking and SDN, the hybrid model operates in an environment marked by synergy. Expertly manages traffic, allowing seamless integration of SDN capabilities into established systems [69, 83, 84, 85].	Junction Contrail, Brocade Vyatta Platform

ditional version. As illustrated in Figure 3.5, the source virtual machines synchronize with the centralized system before initiating multicasting, bypassing ARP-like address resolution protocols. This shift ensures optimized link resource allocation and minimizes potential network disturbances.[13]

Additionally, VM migrations witness enhanced efficiency under the SDN-integrated VXLAN. Unlike traditional VXLAN, which faced setbacks due to manual configurations, the SDN-enabled version benefits from centralized control and intelligent oversight. Figure 3.5 illustrates a network architecture that integrates proactive and passive VM migration strategies within an SDN-enhanced VXLAN framework. The control layer features dual SDN controllers surrounding an intelligent center, tasked with load balancing and migration decision-making. Below, the core layer with two spine switches facilitates VXLAN traffic across the network. The access layer shows hypervisors and VMs before and after migration, marked by dashed and solid lines, indicating passive and proactive migration, respectively. This visual representation emphasizes the SDN's role in automating and optimizing VM placement within a VXLAN network. [40, 33]

Figure 3.6 zooms in on the SDN-integrated VXLAN architecture during a VM migration

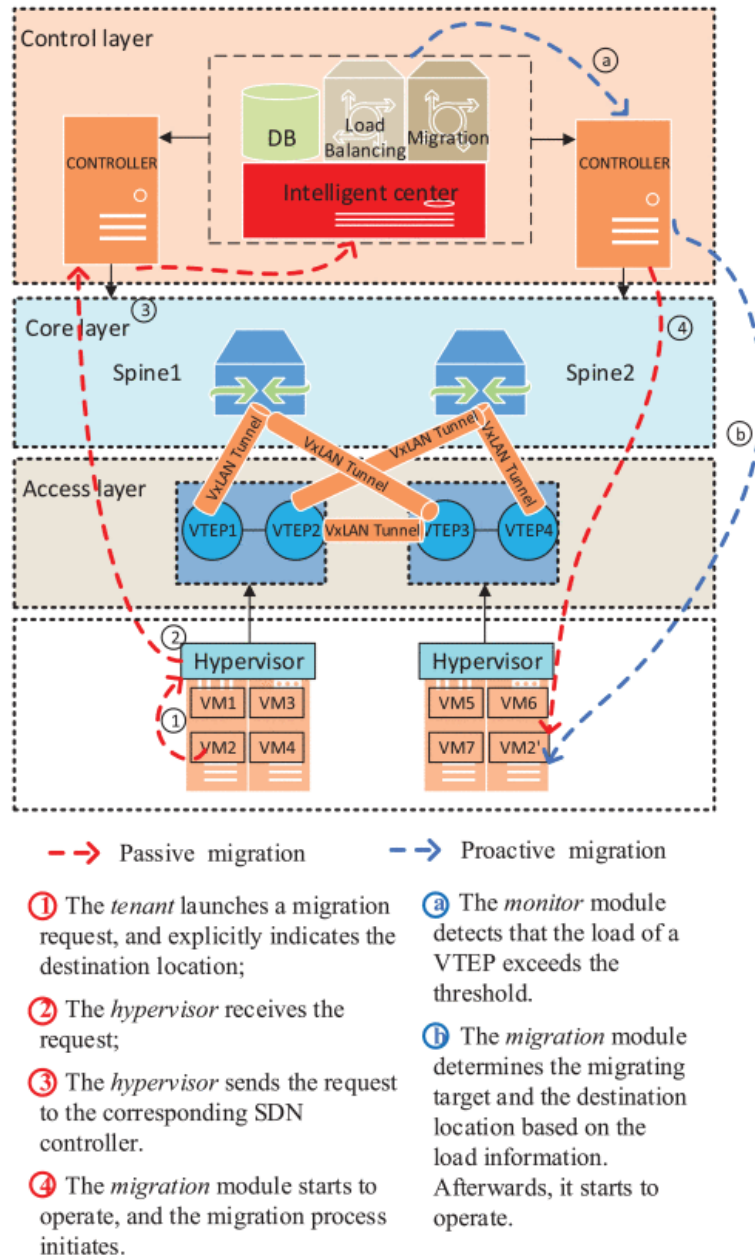


Figure 3.5. Comparative Overview of Proactive and Passive VM Migration Mechanisms in SDN-integrated VXLAN Architecture [91]

process. A single SDN controller at the control layer highlights the centralized management approach. The core layer's spine switches, and the access layer's hypervisors demonstrate the migration's aftermath. The blue dashed line traces the VM's journey, while the red dashed line signals the establishment of new network connections postmigration. This figure underscores the seamless transition and continuity of service facilitated by SDN in VXLAN environments. [40, 33]

Furthermore, the combination of SDN and VXLAN introduces advanced load balancing capabilities. The intelligent center adeptly evaluates network performance and adapts migration thresholds accordingly, optimizing the efficiency of Spine and VTEPs. This

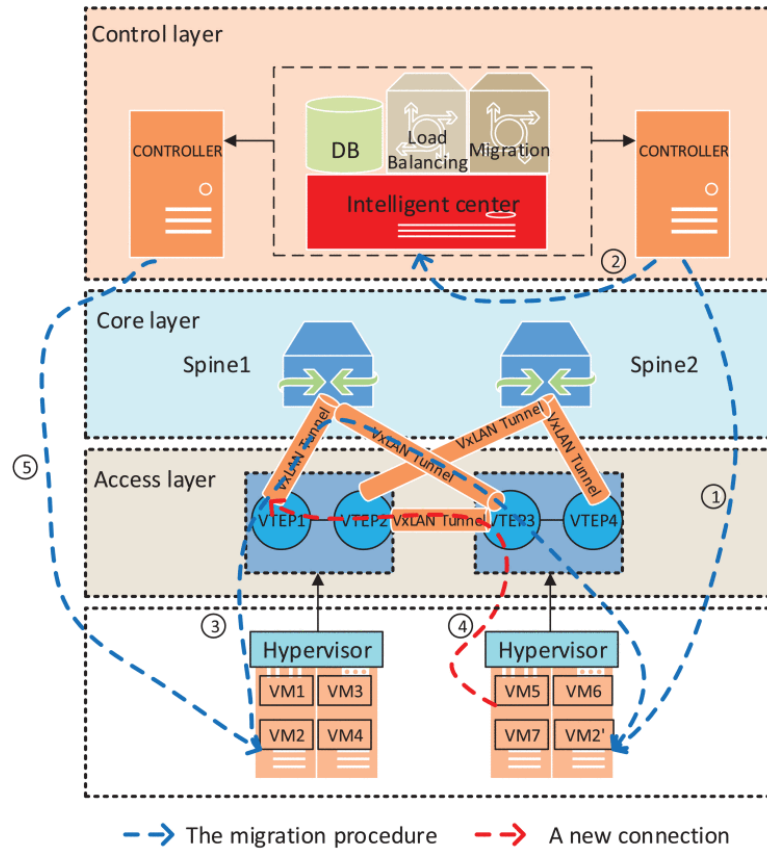


Figure 3.6. SDN-based VXLAN Architecture Depicting the VM Migration Procedure and Establishment of New Connections [91]

adaptability indicates future enhancements through innovative algorithms.[24, 25, 47]

Although SDN-VXLAN integration offers significant potential, developing a complete PoC is challenging due to the extensive resources and time required. Therefore, this thesis will focus on automation tools such as Ansible in conjunction with SDN controllers, providing a narrower but equally valuable exploration.

In conclusion, the fusion of SDN and VXLAN presents a revolutionary architectural approach, reshaping cloud network dynamics through improved IP multicasting, efficient VM migrations, and sophisticated load-balancing techniques. This union not only addresses the inherent challenges of VXLAN, but also sets the stage for the design of powerful, scalable multitenant cloud networks.

3.2 Network Automation

3.2.1 Introduction to Network Automation

The transition to SDN has transformed modern networking, emphasizing the role of VXLAN in enabling layer 2 network connectivity across geographically dispersed locations via Layer 3 links. The transition to SDN has caused increased complexity in networks.

As a result, the need for efficient management and configuration of these networks has become increasingly vital.[7, 27].

Network Automation addresses this challenge. Although automation in technology is not novel, its adoption within SDN-augmented networks is revolutionary. Designed to streamline the intricate tasks associated with the configuration, management, and surveillance of software-focused networks. Through automation, NetOps can realize the full potential of SDN, achieving improved resource allocation and peak network performance.[7]

The merits of Network Automation extend beyond simplifying tasks for network engineers. It plays a crucial role in minimizing human-driven errors, often involved in network disruptions. In an era where data traffic flows from various sources, relying on manual network configurations prone to mistakes and inconsistencies becomes impractical. Hence, the adoption of automation emerges as an essential step, guaranteeing the seamless functionality of SDN-centric networks.[7]

In conclusion, the confluence of SDN, VXLAN, and Network Automation heralds the advent of advanced networking solutions. Subsequent sections will delve into the integral tools and methodologies of Network Automation, underscoring their significance in fortifying and adapting contemporary networks.

3.2.2 Comparative Analysis of Network Automation Tools

In the dynamic domain of network engineering, configuration acts as the cornerstone, ensuring that network infrastructures operate seamlessly. Given the increasing traffic from various data sources, manual configurations often lead to inefficiencies and errors. To address this, network automation has introduced advanced tools to facilitate more effective infrastructure management.

Among the numerous tools at the disposal of NetOps, Ansible, Puppet, and Chef stand out. They are designed to ensure that servers adhere to configurations specified in their automation scripts, such as the playbook in Ansible or the cookbook in Chef. Whenever manual configurations deviate, these tools can revert the system to its predefined state [57].

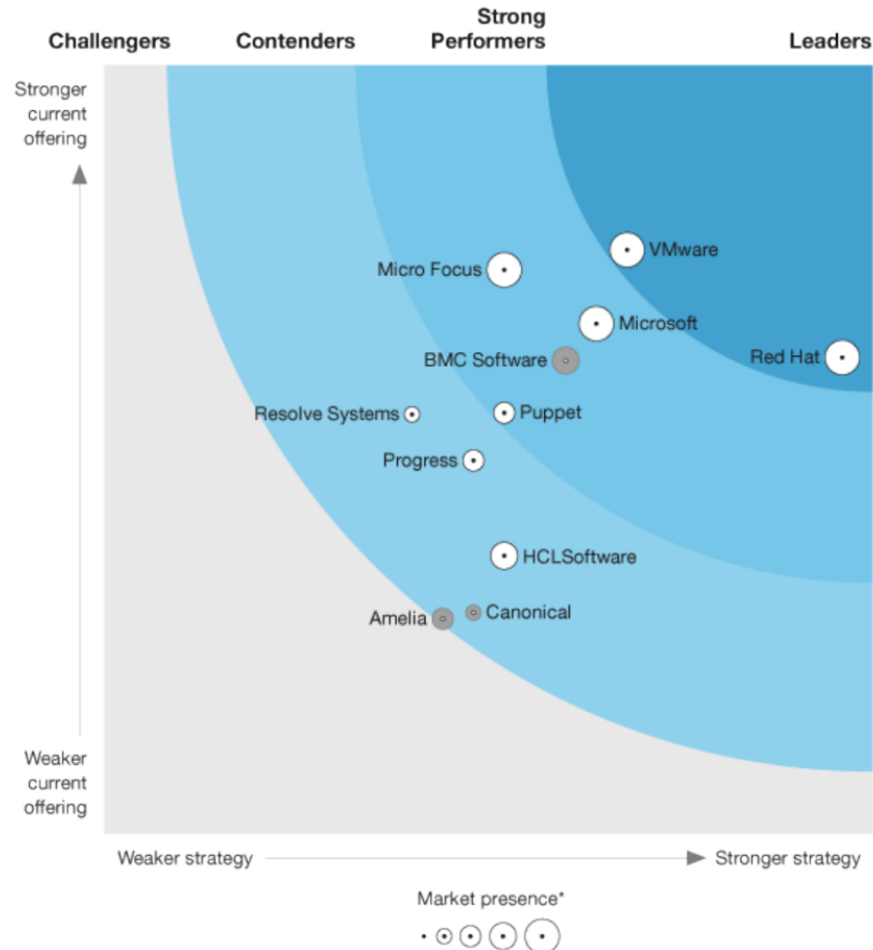
A notable report by Forrester Research sheds light on the competitive landscape in infrastructure automation, as shown in Figure 3.7. The application of Red Hat emerges as a leader in this space, outperforming its peers [77]. However, each tool has its strengths. Puppet's declarative approach outlines the desired result and lets the system handle the rest. In contrast, Chef, known for its imperative approach, provides explicit instructions for system tasks.

Research underscores the impact of Ansible on network automation. Its playbook-driven

THE FORRESTER WAVE™

Infrastructure Automation

Q1 2023



*A gray bubble or open dot indicates a nonparticipating vendor.

Figure 3.7. The Forrester Wave™: Infrastructure Automation, Q1 2023[77].

methodology aligns well with scenarios that require swift automation [20]. Salazar-Chacón and Parra further validate Ansible's utility as a Network Automation Server for managing VXLAN-type networks [63]. Such networks, connecting Layer 2 over distant locations using Layer 3 links, greatly benefit from Ansible's capabilities. Collaborative use with Git as a version control system ensures consistency of configuration and minimization of errors.[63]

Ansible's versatility is evident from its deployment across devices, such as configuring routing protocols on Cisco routers and Mikrotik [22]. Its simple task representation and agent-free nature make it a preferred choice for EIGRP configurations [16].

However, while Ansible sees increasing adoption, the merits of Puppet and Chef remain

Table 3.2. Comparison of network management approaches, adapted from [9]

Approach	Adaptation to Change	Resistance to Config. Drift	Syntax error resilience	Effective Config. Deployment	Scalability
CLI	×	×	×	×	×
SDN (OF)	✓	×	✓	✓	✓
Python and REST APIs	×	×	×	✓	×
Chef	✓	×	✓	✓	×
Puppet	✓	×	✓	✓	×
Chef and Git	✓	✓	✓	✓	×
Puppet and Git	✓	✓	✓	✓	×
Proposed Solution (Ansible and Git)	✓	✓	✓	✓	✓

clear. Luchian et al. delve into Puppet’s architecture, tailored for network devices, ensuring cohesion between ideation and execution [35]. Chef, originally created for software installation automation, now boasts an ecosystem centered on the Chef node to synchronize device configurations with server directives [46].

Blessing and Aaron present a meticulous comparison of these tools [9]. Table 3.2 elucidates attributes such as adaptability, resilience to configuration drift, and deployment efficiency. Such insights are invaluable to professionals in the IT and network domains, guiding their tool selections.

3.3 Monitoring, Logging, and Visualization in VXLAN Management

3.3.1 Overview of Monitoring, Alerting, and Visualization

Monitoring is a pivotal tool in the realm of system maintenance and optimization, encompassing the collection, storage, analysis, and visualization of data from the system landscape [75]. Paint a comprehensive picture of the current state, providing information on potential future behaviors and comparisons with past performance. In SaaS, it is paramount to achieve refined granularity ideally up to the microservice level [75].

The monitoring mechanism extends beyond mere data visualization to include a critical component known as ‘alerting.’ This involves actions initiated by changes in the monitored metrics. Central to alerting are two elements: the threshold, which denotes the value beyond which action is warranted, and the actual action, generally an informative

notification sent to designated personnel, that highlights the location and timing of the event [75]. Alerts serve as sentinels that sound the alarm when metrics exceed thresholds, whether they are historical, projected, or deduced by machine learning techniques [75].

In contemporary system architectures, the demand for adept monitoring and alert solutions such as Zabbix, Prometheus, Grafana, and the Prometheus AlertManager has surged. These tools form a cohesive framework, enabling proactive detection and mitigation of system vulnerabilities, thus boosting performance [12]. Moreover, as network monitoring becomes an indispensable component of this domain, traditional manual oversight is now being overshadowed. Integrating advanced functionalities into these tools heralds a paradigm shift, paving the way for incisive and real-time insights into network dynamics [23].

3.3.2 Introduction to Logging and Its Significance

In today's advanced technological landscape, the swift flow of information signifies the progress we have made. This flow is based on the adept management of network resources by network administrators [60]. Networked devices frequently encounter various anomalies. Each of these, minor or significant, is consistently documented in the system's 'syslog' [60, 32].

As computer networks expand, filled with various devices, the task of identifying the sources and causes of these anomalies becomes a challenge for administrators. Centralized log management emerges as an essential solution in this context. It gathers logs from different devices into one accessible platform, enabling efficient analysis and reporting [60]. Such logs, which capture daily operations and user activities, serve as foundational elements to enhance system security and prevent threats from being addressed [72, 38].

In networking, logs not only monitor, but also proactively detect network anomalies and failures, reinforcing their indispensable role [26].

3.3.3 Evaluating Modern Monitoring and Logging Solutions: A Comparative Analysis

The dynamic world of network management in the context of evolving cyber threats requires a dual focus on both monitoring and log solutions. Efficient network management is based on the ability to not only monitor network activities in real time but also log these activities for analysis, troubleshooting, and predictive insights. In this regard, the choice and integration of the right tools are paramount.

The ELK stack, which includes Elasticsearch, Logstash, and Kibana, has been a prominent solution in this space, offering a powerful combination for both the logging and monitoring aspects of network management. It provides a comprehensive platform for centralizing log processing and enhancing visual analytics, allowing administrators to effectively differentiate between standard and anomalous network activities [60, 72, 74].

In contrast, the PLG stack, which includes Promtail, Loki, and Grafana, has emerged as a formidable alternative, especially noted for its log aggregation and real-time data analysis capabilities in high-performance computing environments [6]. A detailed comparison between the ELK and PLG stacks reveals that while the PLG stack demonstrates greater efficiency in CPU and memory usage, especially for real-time log indexing, the ELK stack stands out for its faster query response times and user-friendly interface [15].

The choice between these two stacks involves weighing their respective strengths. While ELK is a comprehensive commercial software offering robust features, the PLG stack is an open-source solution, providing easy integration and a cost-effective approach for organizations. The Loki component within the PLG stack, specifically, offers efficient storage through its metadata indexing approach, which contrasts with Elasticsearch's method of indexing full document contents. This distinction underlines Loki's storage efficiency, albeit with a trade-off in search capabilities compared to Elasticsearch.

In monitoring, Grafana's expertise in web metrics visualization complements Prometheus's strengths in data source interoperability, showcasing adaptability for diverse monitoring needs [4, 68, 12]. Integration of these tools with Zabbix, which offers an all-encompassing suite of visualization and alert functionality, forms a comprehensive monitoring solution. This integration is further illustrated in Figure 3.8, showing the difference between push and pull-based monitoring methodologies.

Focusing on the monitoring aspect, the role of Prometheus becomes crucial. As a pull-centric framework, Prometheus is particularly suited for large-scale VXLAN networks spread across diverse regions. Its fundamental pull focus, complemented by support for push metrics, showcases its adaptability in contemporary network settings [5, 2]. This adaptability is vital in an era where network scalability and flexibility are key.

Pairing Prometheus with Grafana takes real-time network monitoring to the next level, especially crucial for extensive cloud setups. This combination contrasts sharply with platforms like Graphite, which, although proficient, offer limited functionality to manage time-series data [66, 52].

Additionally, Zabbix's comprehensive suite for visualization and alert functionalities stands out in the monitoring landscape. Its support for a wide range of databases and its adherence to SNMP protocols and ISO / IEC 27002: 2013 standards emphasize its robustness for network monitoring tasks [11, 30, 58]. The user-friendly interface and advanced au-

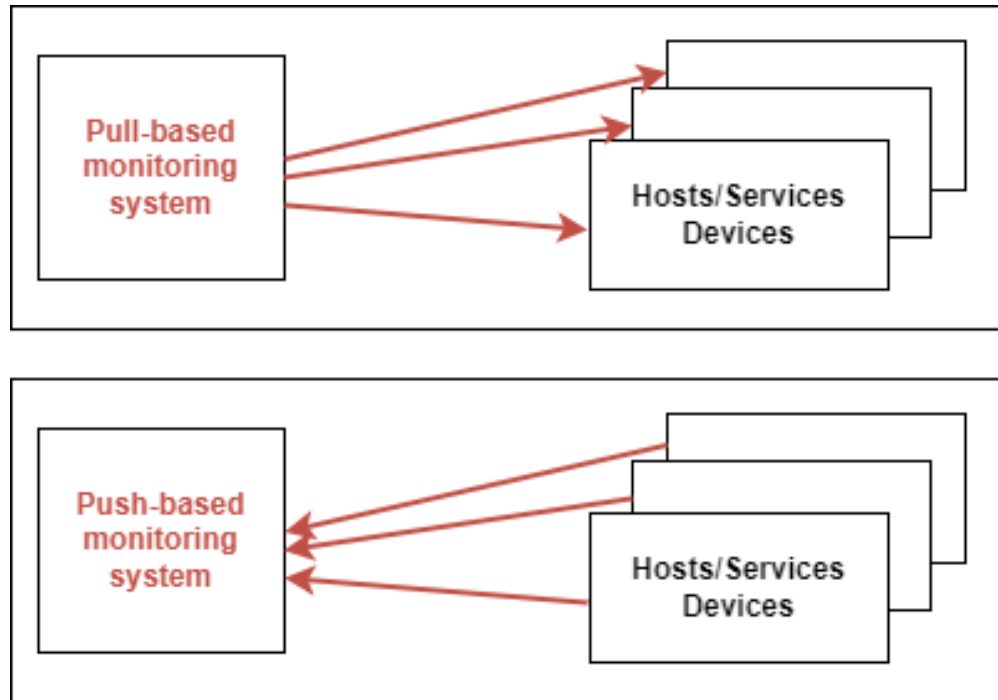


Figure 3.8. Differentiation between push and pull based monitoring methodologies.

tomation capabilities of Zabbix further enhance its effectiveness in managing complex network setups, and its SQL-centric design allows intricate data analyses, even extending into neural network domains [45, 36].

In summary, integration of the ELK or PLG Stack with Prometheus & Grafana and Zabbix presents a robust and versatile solution for VXLAN network management. The combination of these tools offers a comprehensive approach, is adaptable, and resilient, which is crucial for modern network environments. Figure 3.8 further illustrates the differentiation between push and pull-based monitoring methodologies, highlighting the different approaches in network monitoring. In summary, integration of the ELK or PLG stack with Prometheus & Grafana and Zabbix presents a robust and versatile solution for VXLAN network management. The combination of these tools offers a comprehensive approach, is adaptable, and resilient, which is crucial for modern network environments. Figure 3.8 further illustrates the differentiation between push and pull-based monitoring methodologies, highlighting the different approaches in network monitoring.

The choice between ELK and PLG for logging should be based on a thorough assessment of organizational needs, considering factors such as efficiency, usability, scalability, and cost. The perfect blend of real-time monitoring, analytical prowess, and visualization achieved with either stack sets a high standard in VXLAN network administration, catering to a diverse range of network management requirements.

3.3.4 Alerting with Prometheus' Alertmanager

In the complex domain of network management, the importance of advanced alerting systems is paramount, and Prometheus AlertManager embodies this advancement with critical efficiency [51, 61]. Alertmanager is designed to handle alerts sent by client applications, such as the Prometheus server. Its core functionalities involve the intricate processes of deduplication, grouping, and routing of alerts, which are indispensable to maintain a clear operational state during network incidents.[61]

The Alertmanager's **grouping** feature adeptly addresses the issue of alert floods during significant outages [51]. By aggregating similar alerts into a unified notification, it prevents information overload, allowing teams to maintain focus on the resolution of critical events. The **inhibition** mechanism is another innovative feature, in which noncritical alerts can be intelligently silenced in the wake of a related critical alert, ensuring that attention is not diverted from critical issues. The **silences** feature further allows network administrators to preemptively disable alerts, facilitating uninterrupted workflow during planned maintenance windows or known system changes.[51]

The nuanced integration of Prometheus' Alertmanager in network management workflows epitomizes the evolution of alerting systems. Its purpose extends beyond mere anomaly detection; it refines alerts to make them actionable intelligence, helping to speed up the resolution and maintenance of the quality of network service [51, 61].

In essence, Alertmanager is a sophisticated tool that complements the Prometheus ecosystem, providing a robust framework for alert management that is aligned with the high demands of contemporary network infrastructures.

3.4 Additional Aspects of VXLAN Management

3.4.1 Security and Compliance in VXLAN

The rise of VXLAN in modern data centers has generated significant interest in its security and compliance dimensions. At the heart of VXLAN security lies the triad of data confidentiality, integrity, and authentication within the overlay network. This includes protecting VTEP devices, securing VXLAN packets in transit, and ensuring a consistent application of policies.

Currently, several technologies address these concerns. IPsec, for example, is commonly used to encrypt VXLAN traffic during transit. Similarly, firewalls and IDS/IPS solutions are positioned at VTEP gateways to inspect and safeguard VXLAN encapsulated data. On the compliance front, tools like Cisco's TrustSec help to endorse security policies and maintain VXLAN compliance.

For a robust VXLAN security posture, organizations should consider harnessing the synergy of IPSec with potent firewall and IDS/IPS solutions. In addition, regular audits and the adaptation of security protocols using compliance tools can further fortify the security landscape. Given the dynamic nature of cyber threats, it is anticipated that further exploration into VXLAN's security aspects will be invaluable in the future.

3.4.2 Scalability and Performance in VXLAN

VXLAN, by design, is poised to address scalability challenges and bypass the limitations inherent to traditional VLANs. Performance in VXLAN infrastructures usually depends on factors such as packet overhead, latency, and the efficiency of VTEP gateways.

To address these concerns, high-performance VTEP gateways are used, optimized for rapid encapsulation and decapsulation. Additionally, the adoption of jumbo frames, configured to account for the VXLAN header, mitigates packet fragmentation.

For optimal performance, it is prudent to deploy high-end VTEP gateways and support jumbo frames in network configurations. As the demand on networks grows, it is foreseeable that scalability and performance will draw more attention, necessitating deeper insights and solutions.

3.4.3 VXLAN Reliability: Redundancy and failure mechanisms

Reliability remains a cardinal principle in networking, and VXLAN is not exempt. Redundancy in VXLAN refers to duplicating essential components, whereas failover mechanisms ensure continuous network operations even in the face of unexpected failures.

To increase reliability, technologies such as MC-LAG offer redundancy by distributing loads across VTEPs. Similarly, ECMP provides multiple paths for VXLAN traffic, enhancing load balancing.

Adopting both MC-LAG and ECMP can significantly improve network resilience. As network architectures grow in complexity, the domains of redundancy and failover are ready for further exploration and innovation.

3.5 Gaps and Opportunities in Current VXLAN Management Approaches

In the domain of network management, where technological advancements unfold at a breakneck pace, practitioners and academics face a dichotomy of fascination and challenge. The survey of existing literature exposes an array of techniques and tools that have significantly altered the landscape of modern networking. Despite these advances, there

are substantial gaps that warrant focused academic scrutiny and pragmatic resolution.

One of the most pressing challenges identified is the issue of consistency in network configurations. Automation tools like Chef and Puppet have been at the forefront of managing configurations, yet they have not completely eradicated the problem of configuration drift. This phenomenon, a discrepancy between the intended and actual configurations, threatens the integrity of the network and requires constant and vigilant oversight to ensure that the configurations are as intended, as emphasized in the literature.[46]

In addition, the integration of a wide range of tools and platforms remains a formidable challenge. Individual tools possess considerable potential; however, the true test lies in achieving interoperability among these disparate systems. Current solutions often fail in this area, suggesting the need for a more integrated approach to network management.

Scalability is another aspect where current tools and methodologies reveal their limitations. Not every existing tool is capable of adapting to the needs of growing and evolving network infrastructures. The comparison presented in table 3.2 underlines this concern, pointing to a requirement for scalable solutions that do not compromise on performance or efficiency.

Yet, within these challenges lie opportunities. The incorporation of Ansible, known for its automation strengths, alongside the version control capabilities of Git, may offer a resolution to the persistent issue of configuration drift. This combination could improve the consistency and reliability of network configurations.

Furthermore, a unified monitoring suite that brings together Prometheus, Grafana, Zabbix, and now the ELK stack holds the potential for a new era of network management. The ELK Stack provides powerful logging, data processing, and visualization capabilities to enhance VXLAN management. Using these tools, networks transform from a lofty goal to an attainable reality.

The concept of a PoC is also instrumental in this regard. By developing a PoC that takes advantage of the collective capabilities of these tools, both the practicality and the theoretical underpinnings of such an integrated approach can be empirically validated. This validation is crucial not only for academic pursuits but also for paving the way for robust, real-world network management solutions.

In summary, as the intricacies of VXLAN environments grow, it is imperative not only to identify and bridge existing gaps in management approaches but also to take advantage of the opportunities these gaps present. The future of network management lies in the ability to adapt, integrate, and scale with precision, a challenge that is as daunting as it is exciting.

4. TOOLS FOR VXLAN NETWORK MANAGEMENT OPTIMIZATION

In the "Programmable Age", network management has undergone a radical transformation, especially in virtualized contexts. The shift is evident through the integration of modern cloud technologies, automation tools such as Ansible, and monitoring utilities such as Prometheus, Grafana, and Zabbix, as shown in Table 2.1. This integration emphasizes the improved efficiency, scalability, and adaptability of today's networks. Subsequent sections dive into VXLAN management optimization strategies, leveraging these advanced tools and technologies.

4.1 Ansible in Automated VXLAN Network Configuration

Red Hat's Ansible has carved a niche in IT automation. The driving force behind its popularity is its architecture, portrayed in Figure 4.1. The architecture of Ansible is intricate, yet intuitive. It is powered by the Ansible orchestration engine. It also interacts smoothly with both private and public cloud services and a Configuration Management Database (CMDB) [20].

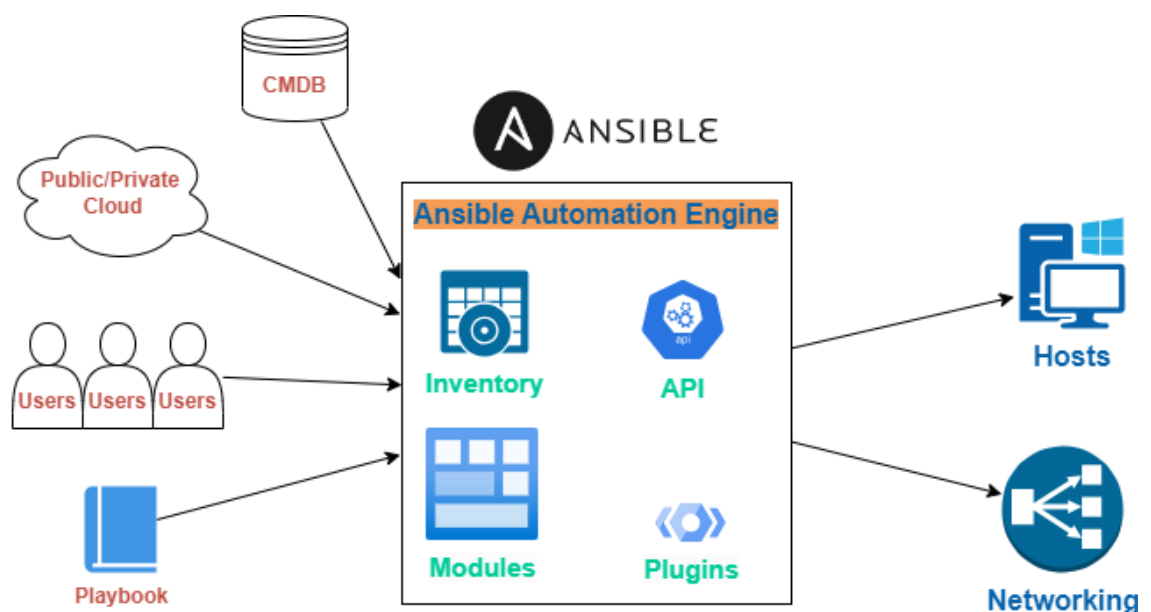


Figure 4.1. Ansible Architecture.

Understanding Ansible's architecture, as presented in Figure 4.1, requires familiarization with its key components. These components are summarized in Table 4.1, which provides a detailed description of each component [20].

Table 4.1. *Components of Ansible and Their Descriptions.*

Components of Ansible	Description
Inventory	This contains a detailed list of nodes or hosts with information such as IP addresses and servers designated for management.
APIs	Crucial to Ansible, these APIs are the bridge between various cloud services.
Modules	Ansible distributes these transient modules across nodes, executes them, and then removes them after task completion. They operate on multiple machines without the need for databases or servers.
Plugins	The components can be standard or custom made. They boost Ansible's primary capabilities.
Playbooks	In YAML format, playbooks outline the tasks for Ansible to execute. They are versatile, supporting both synchronized and independent task starts.
Hosts	These are the node systems within Ansible's range, covering machines from RedHat to Linux and Windows, set for automation.
Networking	Ansible's network automation strength stems from its agentless automation framework.
Cloud	Representing a group of distant servers, the cloud is vital for data handling and processing.
CMDB	This data repository holds information for IT installations.

The rapid evolution of digital domains highlights vulnerabilities due to human errors. Red Hat's Ansible® Automation Platform addresses these by converting intricate manual processes into streamlined workflows. This highlights the vital role of automation tools such as Ansible in present-day network management. [76]

Integration of platforms like Git and GitHub can enhance Ansible's capabilities. Git is a distributed version control system. GitHub serves as its cloud-based hosting counterpart. Together, they aid in efficient code monitoring and cooperative adjustments. When integrated with Ansible, this combination enhances the automation process, making it stronger, more trackable, and more cooperative. [67, 63]. This process is enriched with versioning features and supports a community-centric automation approach. Though this thesis does not explore integration in depth, the combined strengths are evident in current IT landscapes.

4.2 ELK Stack: The Established Commercial Standard for VXLAN Logging

In VXLAN environments, where monitoring and log analysis are critical, the ELK stack stands out as a commercial standard. Integrates Elasticsearch, Logstash, and Kibana - components that harmonize the logarithmic process from ingestion to visualization [82, 59].

Elasticsearch lies at the heart of the stack, serving as a powerful search and analytics engine. It is capable of efficiently indexing a wide array of data types and facilitating near real-time search and analytics. Elasticsearch ensures high availability and fault tolerance through its distributed nature [73].

Logstash's role as a data processing intermediary is indispensable. With its flexible plugin architecture and powerful processing capabilities, Logstash can ingest data from numerous sources, transform it, and ship it to Elasticsearch for indexing [73, 82].

Kibana offers extensive visualization capabilities, enabling users to create dashboards that provide insights into their data. With Kibana, administrators can manage and monitor the health of their Elastic Stack cluster and control user access [73].

Alongside these core components, the Elastic Stack includes **Beats** and the **Elastic Agent** for data collection, **APM** for application performance monitoring and **Fleet** for centralized agent management [73].

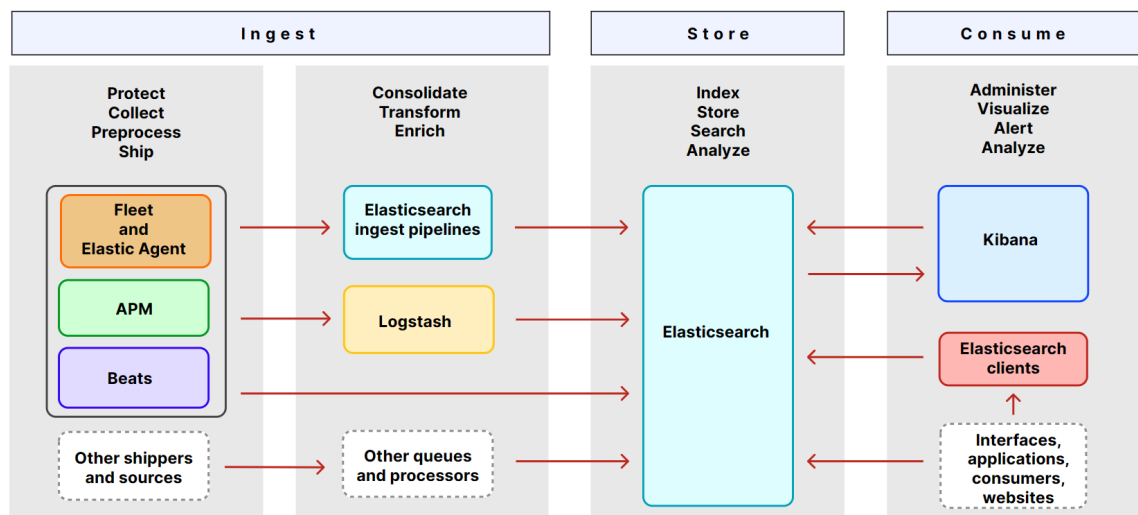


Figure 4.2. The ELK data processing workflow in a VXLAN context, illustrating the comprehensive logging capabilities of the stack (Adapted from [73]).

The ELK stack workflow, as shown in Figure 4.2, shows a structured process from data collection to consumption. Each component plays a critical role in ensuring that data are accurately ingested, processed, stored, and made available for analysis and visualization [73].

In recent developments, the OpenSearch project has emerged in response to licensing changes, ensuring the ongoing availability of an open-source version for Elasticsearch and Kibana. This move underscores the adaptability of the stack and the community's commitment to open source software [73].

The ELK Stack, with its commercial offerings, provides organizations with a dependable and scalable solution for their logging and monitoring needs. Its comprehensive nature not only streamlines VXLAN log management but also fortifies the resilience of the infrastructure against complex challenges such as SIEM and observability [82].

4.3 PLG Stack with Grafana Loki: The Open Source Innovator in Logging Solutions

Following the discussion of the ELK stack as a commercial solution for VXLAN logging, our attention turns to the PLG stack, particularly spotlighting Grafana Loki as a compelling open source alternative. Free to use and developed with the principles of modern observability systems in mind, Grafana Loki prioritizes cost-effectiveness and efficiency.

Figure 4.3 shows the architecture of Grafana Loki, a streamlined and efficient logging system within the PLG stack. In this schematic, logs from two applications are collected by an agent, typically Promtail, which then forwards the logs to the Loki server. Loki is designed to index only the log metadata, optimizing storage and retrieval processes. Users can interact with the log data through Grafana, utilizing LogQL for complex queries, or directly via the command line with LogCLI for simplicity and convenience. This architecture underscores Loki's ability to integrate with existing systems seamlessly, offering a lightweight yet powerful solution for log analysis.



Figure 4.3. The architecture of Grafana Loki

Loki's design prioritizes minimalistic storage requirements by indexing only the metadata of logs. This innovative approach ensures a lightweight and less resource intensive solution, providing a nimble, yet powerful tool for developers and operators who require fast and reliable log analysis capabilities [34].

Integral to the PLG stack is **Promtail**, a log collector specifically designed for Loki. Promtail is responsible for gathering logs and forwarding them to the Loki system, maintaining a crucial link in the observability chain. It adheres to Loki's philosophy of simplicity and efficiency, allowing seamless integration with the logging architecture [56].

Together, Grafana Loki and Promtail offer a streamlined solution that stands out for its open source accessibility and alignment with the evolving needs of scalable infrastructure monitoring and logging [34, 56].

4.4 Prometheus for In-depth VXLAN Monitoring and Alert

4.4.1 Prometheus monitoring

Prometheus was developed using the Go programming language and has attracted attention as a time series monitoring system and database. This powerful tool is adept at scraping metrics from a range of sources, each piece of data being precisely time-stamped to provide clarity on its chronological context. Prometheus was initially developed by SoundCloud, a leading music platform. It soon became a standalone project with a robust developer community, highlighting its significance. Its affiliation with the Cloud Native Computing Foundation (CNCF) in 2016 further emphasized its importance in the IT sector. [55, 50]

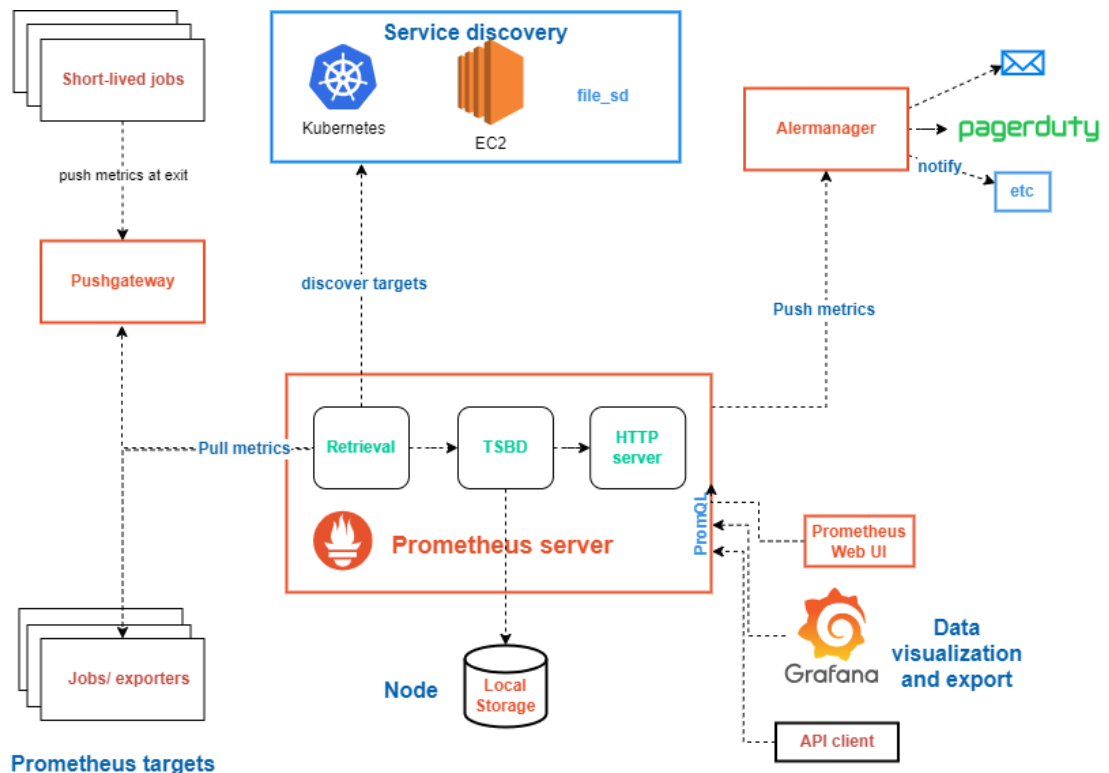


Figure 4.4. Prometheus architecture (Modified from [54]).

A closer examination of its architecture, illustrated in Figure 4.4, reveals the central role of the Prometheus server. This entity is responsible for storing time-series data, leveraging a custom local TSDB (time-series database) for structured data storage. While its default *modus operandi* involves the 'pull' method for data retrieval, it also accommodates the 'push' approach via the Pushgateway, ensuring data consistency even for short-lived processes.

Various exporters support its functioning, transforming data from active applications into a digestible format for Prometheus [53]. Such transformation allows metrics collection either directly from specific exporters, such as node exporter, or through intermediaries such as the Push gateway. Service discovery is another feature that simplifies the monitoring process by automatically identifying candidate monitoring. Simultaneously, Alertmanager integrates with Prometheus to manage alert notifications, ensuring stakeholders are informed through channels such as email, HipChat, PagerDuty, and Slack [51, 1].

Prometheus' distinctiveness in the monitoring landscape is further accentuated by PromQL, its specialized querying language. Through PromQL, users can not only retrieve metrics in real time but also interpret results in various formats. While PromQL offers unparalleled flexibility, it requires a steeper learning curve. In the broader monitoring ecosystem, the collaboration of Prometheus and Alertmanager often sees Grafana as the visualization tool of choice [54, 61].

4.4.2 Prometheus Alertmanager

Within the Prometheus ecosystem, Alertmanager is more than a mere add-on; it is a critical entity tasked with managing notifications originating from the Prometheus server. By handling these alerts, it ensures the seamless transition from anomaly detection in Prometheus to the communication of actionable insights to operation teams [51, 2].

Alertmanager's core offerings include grouping of alerts, which aids in consolidating similar alerts, preventing potential information overload. The tool's ability to filter out less critical alerts during a major incident demonstrates its focus on operational efficiency. Further enriching its user-centric design, Alertmanager provides options for manual alert muting based on certain conditions, ensuring that relevant alerts reach the intended recipients. Its commitment to maintaining a robust monitoring setup is evident in its support for high-availability configurations. [51]

An overview of Alertmanager's workflow, presented in Figure 4.5, portrays it as a service that processes requests from Prometheus servers. After deduplication, these alerts are routed based on established parameters. Its Web interface supports alert visualization, silencing, and the application of inhibition rules. During network partitions within its cluster, Alertmanager's design ensures notifications are sent from both partition sides, reflecting

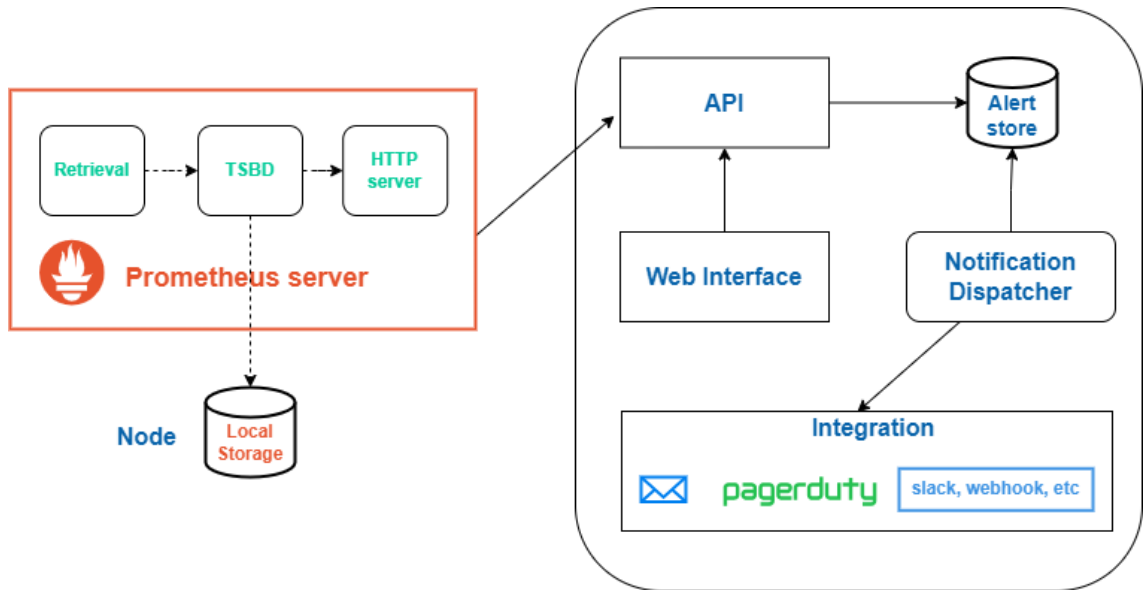


Figure 4.5. High-Level Overview of Prometheus Alertmanager (Modified from [5]).

a priority on delivery. [5]

Alert routes in Alertmanager can be visualized as hierarchical structures. Upon activation of a specific route by an alert, a predetermined integration is set into motion. Several out-of-the-box integrations cater to common communication channels like email and Slack. For custom solutions, webhook integration allows an HTTP POST request to send the firing alert's JSON payload to a specified endpoint. [5]

4.5 Grafana: Advanced Dashboards for Network Analysis

Grafana has emerged as the leading open source platform for data visualization and dashboard creation[18, 17, 75]. Unlike other similar systems, Grafana can unify data from diverse sources such as Kubernetes clusters, cloud services, or even Google Sheets without the need to ingest it into a back-end store[17]. This independence of the database gives Grafana unparalleled flexibility in data visualization.

At its core, Grafana advocates the democratization of data. It aims to break organizational silos, ensure wide access to data, and foster a culture grounded in collaborative data-driven insights[17].

Figure 4.6 provides a detailed visualization of network storage and temperature readings, using Grafana gauge and bar gauge panels. The upper section of the dashboard presents data storage metrics across multiple devices, with color gradients for quick status assessment. LCD-style bar gauges offer an at-a-glance view of storage usage, while basic bar gauges allow for straightforward comparisons across devices. The lower section presents temperature readings from various locations in gradient mode, translating numerical data into a visual format that highlights critical temperature zones with color intensity.

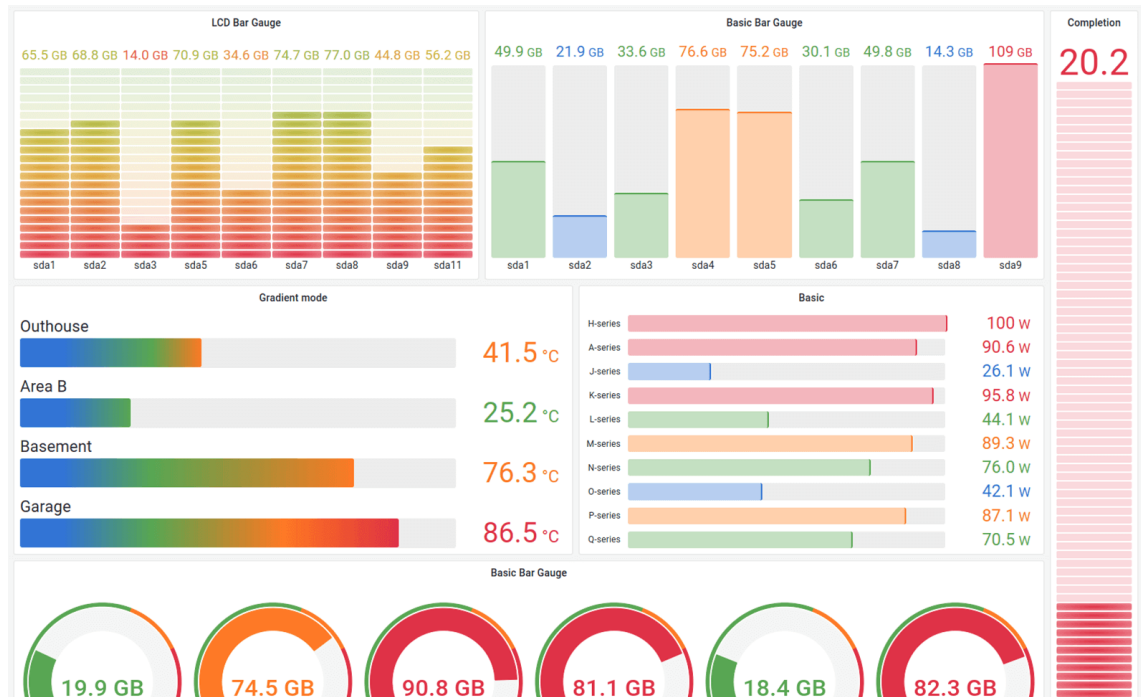


Figure 4.6. Grafana's Gauge Dashboard Representation [17]

Figure 4.7 presents Grafana's time series visualizations, which demonstrate the ability of the platform to represent changes in data over time through line graphs with shaded areas, multiple Y-axes for different units of measurement, and stacked bars for comparative analysis. The dashboard also utilizes bars with opacity gradients and dashed lines with threshold regions to differentiate between normal operating ranges and potential outliers or points of interest.

These figures exemplify Grafana's capability to transform complex datasets into accessible and actionable insights, fostering collaborative data-driven decision-making within network analysis.

Grafana offers a wide set of features, ranging from heat maps and histograms to complex data querying and transformations[17]. Its dynamic dashboards provide a comprehensive environmental view, enabling end users to rapidly gauge the prevailing scenarios. Moreover, these dashboards can be shared and imported using various methods, and the platform's alerting system consolidates notifications. Users can configure these alerts directly within dashboards and notifications can be transmitted to a variety of communication platforms[18]. To augment its inherent capabilities, Grafana supports a wide spectrum of plugins, sourced from both Grafana Labs and the wider community. [17]

Grafana's adaptability stands out. Catering to both novice and expert users, the platform offers customized configurations, from metric exploration to advanced provisioning. This versatility extends to its authentication methods, and the enterprise edition even facilitates team-specific user assignments. [17]

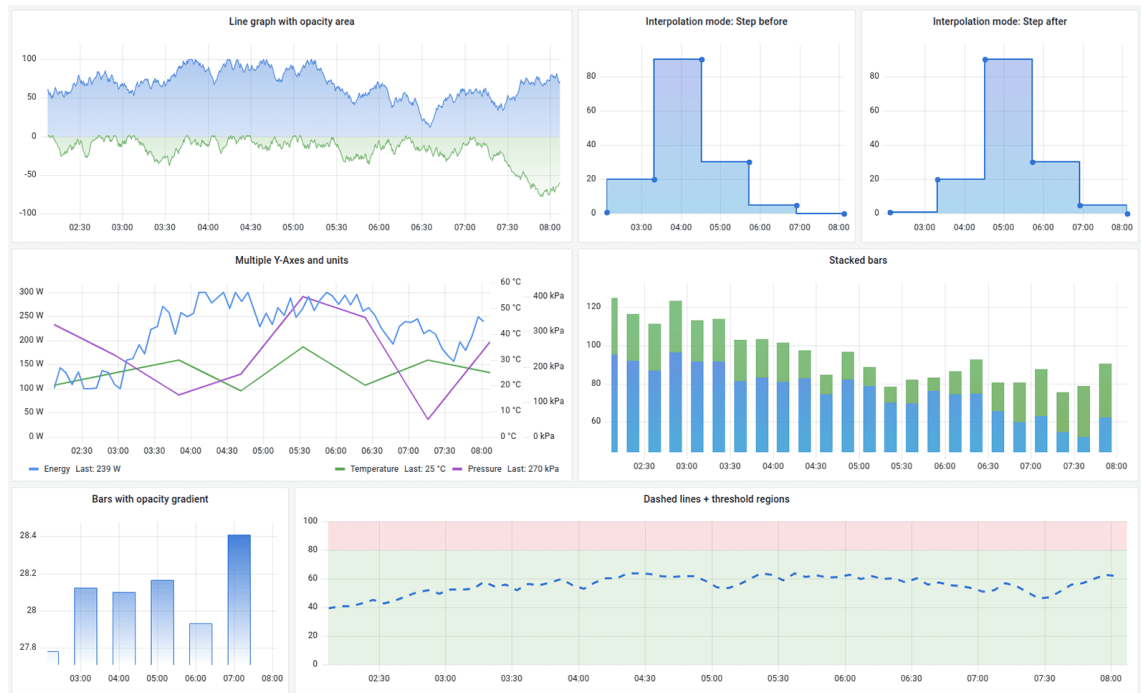


Figure 4.7. Grafana's Time Series Dashboard Visualization [17]

Furthermore, Grafana Labs champions several open-source projects. Among these are Grafana Loki, a logging solution optimized for the Kubernetes ecosystem, and Grafana Tempo, a distributed trace back-end. Projects such as Grafana Mimir and Grafana Phlare exemplify Grafana Labs' unwavering commitment to both the open-source community and the broader realms of data visualization and management. [17]

Grafana's comprehensive approach to data visualization, combined with its compatibility with data sources like Prometheus and Zabbix, positions it as an indispensable tool for organizations striving for data-driven excellence. Such integrations amplify the insights obtainable from data, enhancing operational efficacy and strategic decision-making. When combined with tools like Prometheus and Alertmanager, Grafana becomes part of a formidable infrastructure monitoring solution, optimizing service performance and prompt issue resolution.

4.6 Zabbix for Comprehensive Network Monitoring

Zabbix stands out as an open-source enterprise-class monitoring solution. Aleksei Vladishev developed it and has received support from Zabbix SIA due to its comprehensive network monitoring capabilities [89]. Beyond merely examining network parameters, Zabbix effectively supervises servers, applications, virtual machines, and a host of other IT assets.

A hallmark of Zabbix lies in its alert mechanisms. Its adaptable email notification system, together with its proficiency in metrics collection, anomaly detection, and easy deploy-

ment, puts it in a distinguished position within monitoring solutions. [89, 42]

Deeper into its architecture, as illustrated in Figure 4.8, the Zabbix server emerges as the central coordinating entity. It has responsibilities related to data processing, the formulation of monitoring definitions, the configuration of triggers, and the delivery of notifications [88, 48]. Zabbix, while compatible with a set of databases, integrates a proxy in its architecture. This proxy plays a crucial role in collecting and caching data and liaising with devices and agents operating in both passive and active stances.

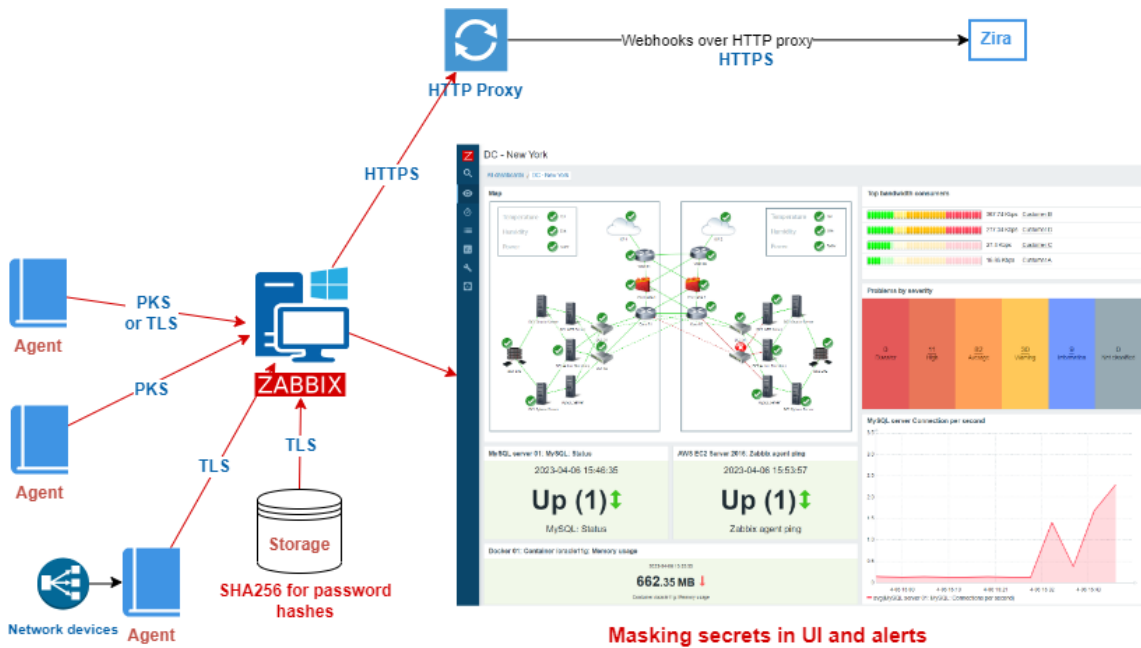


Figure 4.8. Zabbix architecture (Modified from [88]).

Modern improvements in Zabbix accentuate its prowess in detecting network threats. This capability amplifies its contribution to the field of information security [8]. Furthermore, Zabbix's versatility is evident in its potential for integration. A testament to this is its collaboration with Telegram, which allows real-time network monitoring [39]. Research validates Zabbix's stature as an open-source network management tool, highlighting its proactive methodology in maintaining the integrity of diverse IT assets. It constantly adapts to meet the demands of present-day network management [19, 75]. In scenarios that require open source utilities, specific database compatibility, effective device management, and all-encompassing monitoring, Zabbix is the best option. Its ability to integrate with Grafana offers enhanced visualization options. However, its repertoire currently does not encompass an equivalent to that of Prometheus exporters [42, 29].

4.7 Unified Approach to VXLAN Management

The evolution of VXLAN encapsulation reflects rapid advances in network management tools. Integrating tools such as Ansible with advanced logging solutions like the ELK

stack and the emerging PLG stack, featuring Promtail, Loki, and Grafana, showcases the transformative potential achievable in VXLAN management.

Ansible's role in simplifying network configurations is crucial, aligning with modern digital infrastructure needs, although our focus excludes direct SDN experimentation, considering its well-established efficacy. The study introduces a dual approach to log management, presenting both the ELK and PLG stacks as viable options. Although ELK offers a comprehensive solution, PoC emphasizes the PLG stack for its open-source nature, simplicity, and cost-effectiveness, marking a shift towards more resource-efficient solutions.

Prometheus, Grafana, and Alertmanager form a cohesive unit for network monitoring and alerting. Prometheus handles the collection of network metrics, Grafana visualizes these metrics for better understanding, and AlertManager manages the routing of alerts. Similarly, Zabbix provides robust network device monitoring, using predictive analytics to proactively manage network health.

The ELK stack's combination of Elasticsearch, Logstash, and Kibana offers an advanced platform for log management and analytics. However, the PoC's preference for the PLG stack underscores the evolving landscape in VXLAN management, where cost and ease of use become decisive factors.

The orchestration of these tools presents network administrators with a holistic view of network performance, blending Ansible's automation with the chosen stack's analytical capabilities to enhance operational workflows.

Empirical evidence from the PoC highlights the operational and financial benefits of this integrated approach, suggesting a strong return on investment for organizations adopting these strategies. While financial and resource considerations are important, the primary focus remains on validating the effectiveness and efficiency of the chosen tool suite in a VXLAN environment.

In conclusion, this unified management approach, which integrates various tools, heralds a new era in VXLAN management. It equips network administrators with the resources to effectively manage network virtualization complexities, ensuring readiness for future data-centric challenges.

This comprehensive understanding of the tools and methodologies in VXLAN management sets the stage for the PoC, which will empirically demonstrate the practical application and integration of these tools in a VXLAN setup, with a particular focus on the PLG stack for its accessibility and open-source benefits.

4.8 Proof of Concept (PoC): A Simplified Implementation

The discussion of unified VXLAN management leads to the PoC phase. The PoC showcases the efficient integration of select tools into a VXLAN environment, even when resources are constrained. A simplified VXLAN setup is prioritized, featuring the deployment of Prometheus, Grafana, Zabbix, and Grafana Loki within containerized environments. The decision to utilize the PLG stack over the ELK stack is informed by its open source nature, aligning with the aim of an efficient and cost-effective implementation. The PoC is designed to provide a tangible representation of network operations within a VXLAN framework, highlighting the capacity of these tools to enhance the system administration and monitoring capabilities.

In this case, the complexity of SDN is bypassed to mitigate the extensive laboratory requirements and associated expenses. VXLAN configuration spans two droplets, with containers established for the tools mentioned. This method is designed to emulate real-world operations between data centers, offering actionable insights into the integration and functionality of the tools.

Ansible is utilized for remote SSH configuration of the Droplets, directly from a personal laptop, which simplifies the setup and management processes.

The PoC aims to present a clear and pragmatic view of network operations within a VXLAN environment, illustrating how these tools can be leveraged to enhance the efficiency of system administration and monitoring.

4.8.1 Implementation and Configuration Details

The PoC was deployed using the Digital Ocean platform, using two Droplets named AN-DC1 and AN-DC2. Both droplets are configured with substantial computing resources: 16 GB of memory, 4 AMD CPUs, and 200 GB of disk space, running the Ubuntu 22.04 (LTS) x64 operating system within the Frankfurt region (FRA1). This configuration is vital for the VXLAN setup within our PoC. Figure 4.9 provides a detailed illustration of the setup of the Droplets, showing each Droplet with its corresponding IPv4, IPv6 enabled, and private IP within the ANH-VXLAN-FRANKFURT-VPC.

Figure 4.10 presents a comprehensive visualization of the network infrastructure designed for the PoC using VXLAN within a DigitalOcean environment. The diagram delineates a clear and structured overview of the two main components within the network setup, labeled AN-DC1 and AN-DC2, which represent individual nodes or droplets.

Each node is equipped with its own suite of network management tools, demonstrating a Docker-facilitated microservice architecture. For example, AN-DC1 includes containerized instances of the Blackbox exporter for monitoring, Grafana for analytics and visualiza-

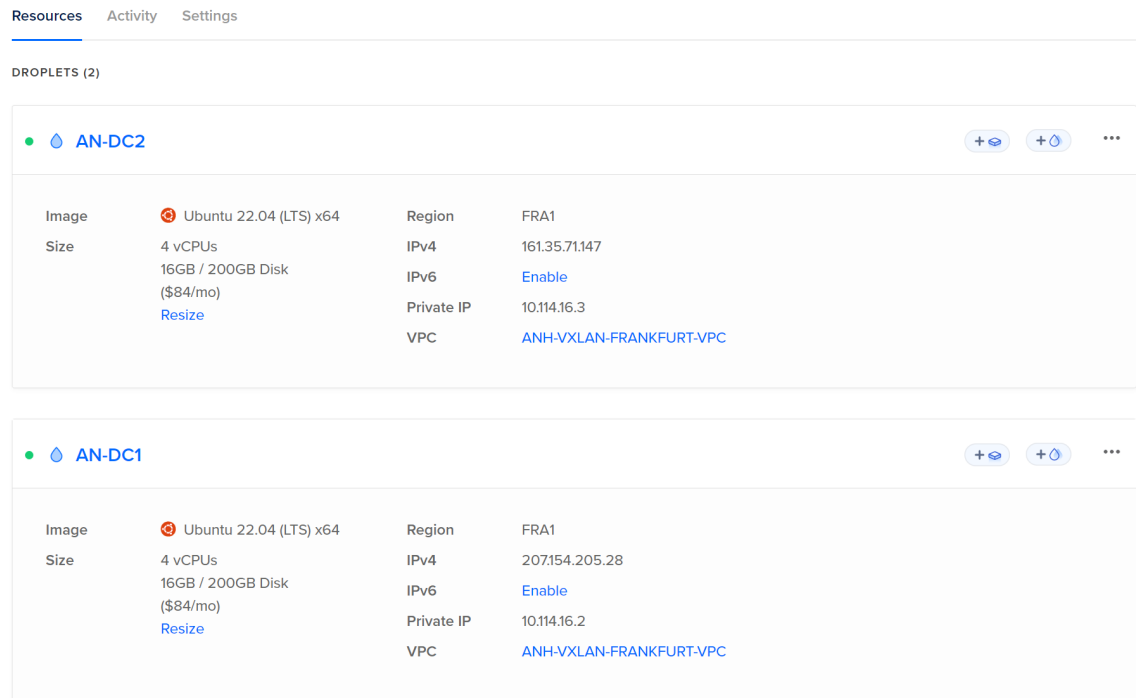


Figure 4.9. Digital Ocean Droplets configuration for the PoC

tion, and Prometheus for time series data monitoring and alerting. This is complemented in AN-DC2 with instances of Zabbix for additional monitoring capabilities, Alertmanager to handle Prometheus alerts, and Grafana Loki for advanced log aggregation.

Central to the network architecture is the VXLAN tunnel connecting AN-DC1 and AN-DC2, denoted by VXLAN1 with a network identifier (VNI) of 2023, which facilitates an overlay network over the physical network infrastructure, referred to as the underlay network. The tunnel encapsulates network frames using a Virtual Tunnel Endpoint (VTEP) on UDP port 4789, enabling secure and efficient communication between nodes.

Open vSwitch (OvS) is depicted on both nodes with the identifier 'br1,' acting as a virtual multilayer switch that provides the binding framework for the containers and the VXLAN tunnel. This setup illustrates how VXLAN can be used to extend layer 2 networks over a layer 3 infrastructure, allowing for enhanced scalability and segmentation.

The establishment of a VXLAN network between the two drops was a critical part of the setup. Open vSwitch, a multi-layer software switch, was installed on both Droplets to enable the VXLAN connections. Alongside, Docker was utilized to create and manage containers needed for the deployment of various tools and services integral to the PoC.

The setup process began with the creation of a virtual switch, termed br1 and used Open vSwitch. This was achieved through the command:

```
sudo ovs-vsctl add-br br1
```

Subsequently, a virtual interface named 'veth1' was added to the 'br1' bridge. This step

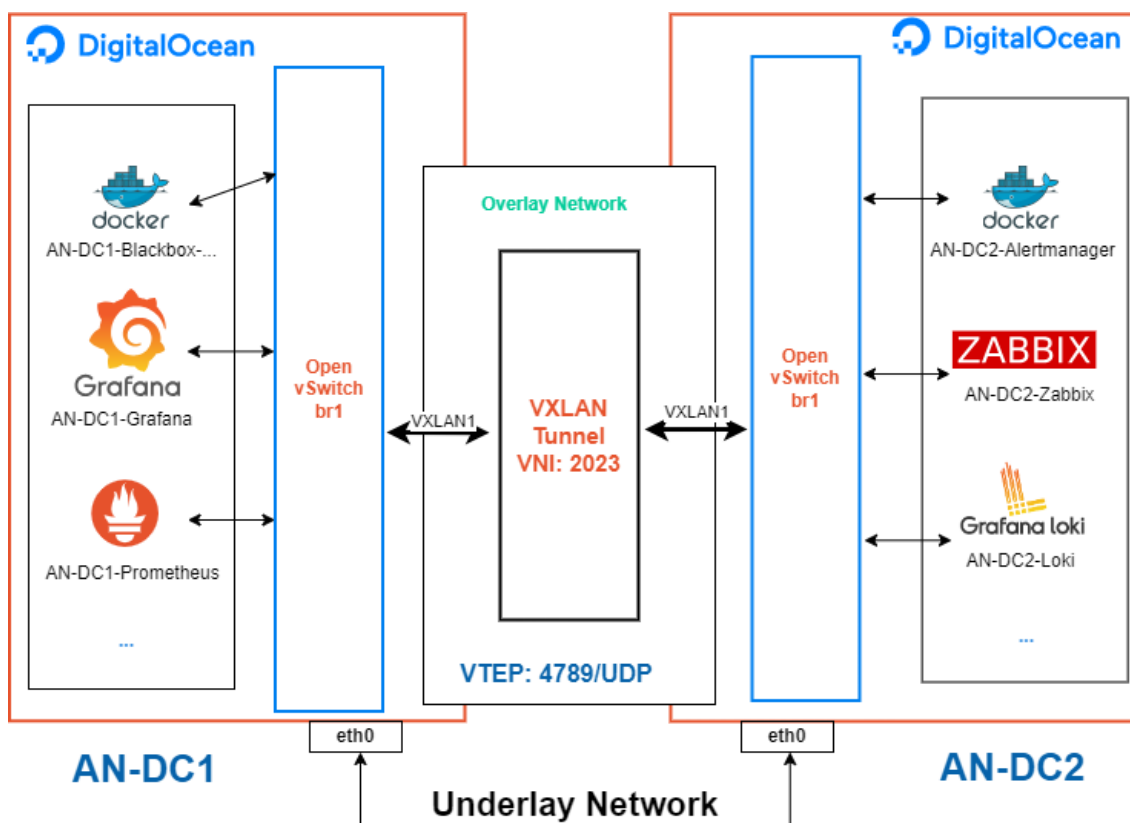


Figure 4.10. PoC Network Infrastructure Diagram with VXLAN and Tools

was crucial for the interconnection within the virtual network. The command executed was:

```
ovs-vsctl add-port br1 veth1 -- set interface veth1 type=internal
```

The next phase involved assigning IP addresses to the bridge interfaces and ensuring their operational status. The commands used were:

```
ip address add 10.x.x.1/24 dev veth1
ip link set dev veth1 up mtu 1450
```

For each container, a unique IP address was assigned to ensure seamless communication within the VXLAN network. This involved running containers and attaching them to the created bridge with specific IP configurations. The command used was:

```
sudo docker run -d --name AN-DC1-Grafana --net=none grafana/grafana
sudo ovs-docker add-port br1 eth0 AN-DC1-Grafana --ipaddress=
10.x.x.5/24 --gateway=10.x.x.1
```

The crucial step of establishing VXLAN tunneling between AN-DC1 & AN-DC2 was carried out. This step involved setting up a VXLAN port on the bridge and configuring the remote IP address, which is the IP address of the opposite droplet. The command used was:

```
ovs-vsctl add-port br1 vxlan1 -- set interface vxlan1 type=vxlan
options:remote_ip=10.114.16.x options:key=2023
```

In setting up the VXLAN environment, specific containers were deployed on two separate Droplets, AN-DC1 and AN-DC2, each serving distinct functions within the network. Below are detailed tables outlining the containers deployed in each Droplet and their respective functionalities.

Table 4.2 provides an overview of the containers deployed in AN-DC1, each of which plays a crucial role in monitoring and data analysis within the VXLAN network.

Table 4.2. Containers and Their Functions on AN-DC1

Container on AN-DC1	Function
AN-DC1-Prometheus	Collects and stores metrics for monitoring and alerting purposes.
AN-DC1-Blackbox-Exporter	Monitors endpoint availability and response times.
AN-DC1-cAdvisor	Provides container-level resource usage and performance data.
AN-DC1-Node-Exporter	Fetches hardware and OS metrics exposed by *NIX kernels.
AN-DC1-Zabbix-Agent	Gathers various system data for Zabbix monitoring.
AN-DC1-Grafana	Visualization tool for metric analysis and alerting.

Table 4.3 lists the containers set up in AN-DC2, focusing on log aggregation, alert management, and monitoring, integral to the efficient functioning of the network.

Table 4.3. Containers and Their Functions on AN-DC2

Container on AN-DC2	Function
AN-DC2-Alertmanager	Manages alerts sent by Prometheus server.
AN-DC2-Loki	Aggregates and stores logs for analysis.
AN-DC2-Zabbix	Centralized monitoring solution.
AN-DC2-Promtail	Collects and ships logs to Loki.
AN-DC2-MySQL	An open-source relational database management system (RDBMS).

The configuration details for the Prometheus server, the Zabbix agent, and Alertmanager are described in YAML files located in the Appendix A (see Figures A.2, A.3 and A.1).

For configuration management, Ansible was used within Windows Subsystem for Linux (WSL) on a Windows 11 system to control Ubuntu-based droplets. AWX provided a web-based interface for easier inventory and playbook operations. Automated tasks included Docker installations and container management, executed securely via SSH.

4.8.2 Monitoring and Visualization Outcomes

In the PoC final stage, our monitoring and alerting configurations come to fruition. Detailed logs and metrics, as visualized in the ensuing figures, illustrate the efficacy and synergy of our monitoring tools. These visual outcomes not only affirm the tools' integration, but also display the practical results of our VXLAN configuration, encapsulating the core objectives of the PoC.

In Figure 4.11, Prometheus shows the graphical representation of the metric `access_evaluation_duration_bucket`, which offers insight into the performance of the AN-DC1-Grafana server. This metric is indicative of the duration distribution for access evaluations. The count of 4587 within a specific duration bucket reflects the cumulative number of events, providing a measure of server responsiveness. Analyzing this metric helps to recognize typical response time patterns and identify performance deviations.

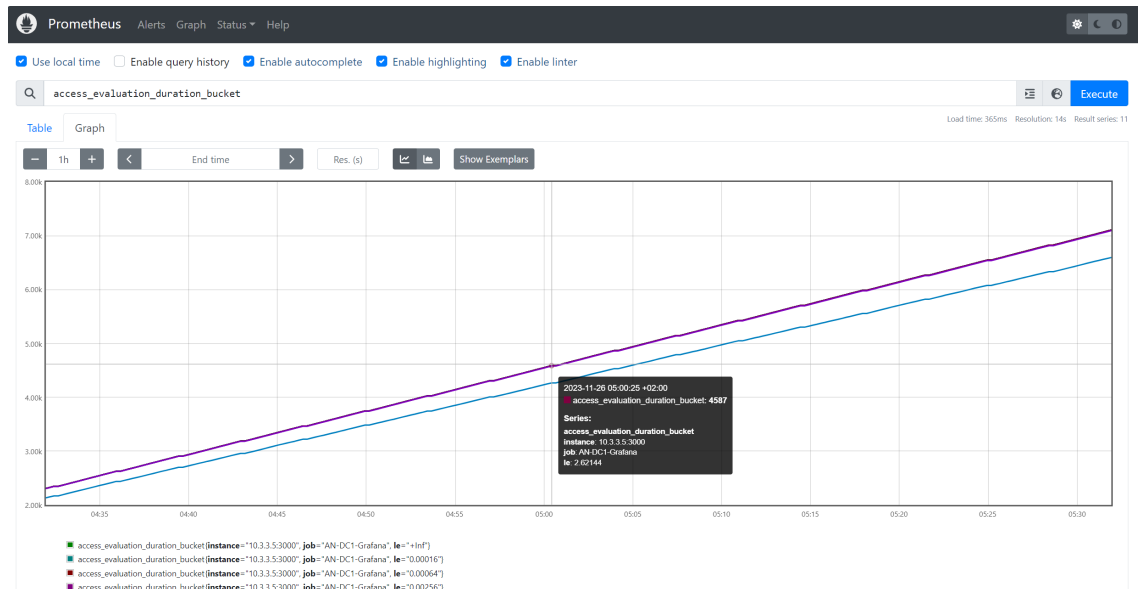


Figure 4.11. Graphical visualization of metrics in Prometheus.

In Figure 4.12, the Prometheus Service Discovery feature is shown, illustrating an automated process that significantly streamlines the monitoring infrastructure within network systems. Service Discovery dynamically scans for active services across the network, automatically identifying and registering their endpoints. This capability is crucial for maintaining an up-to-date monitoring landscape and ensuring that new services are promptly incorporated into the monitoring framework without the need for manual updates or configurations.

The figure shows a succinct interface of Prometheus with a list of services, such as AN-DC1-Grafana, AN-DC1-Node-Exporter, and AN-DC2-Zabbix-Agent, each with an active target status indicating their current monitoring state. Automation of this discovery process is exemplified by the detailed labels assigned to each service, such as `'_address'`

, `'_metrics_path'` and `'scheme'`, which provide essential information for Prometheus to accurately scrape and collect metrics.

For example, the service labeled AN-DC1-Grafana is assigned a specific address and a metrics path, denoting its location and the endpoint from which Prometheus can retrieve metrics. The inclusion of a `'scrape_interval'` label specifies the frequency at which Prometheus collects data, and a `'scrape_timeout'` label defines the maximum duration to wait for a scrape request before timing out, highlighting the customizable nature of monitoring intervals to suit various network demands.

This level of detail in Service Discovery not only facilitates the organization and querying of service metrics but also contributes to a more efficient and resilient monitoring system. It allows network administrators to swiftly adapt to changes in the network infrastructure, ensuring continuous and comprehensive coverage of all services for performance analysis and alerting. The automated detection and labeling process depicted in the figure represent a small yet integral part of Prometheus's robust monitoring capabilities, enabling scalable and flexible network management.

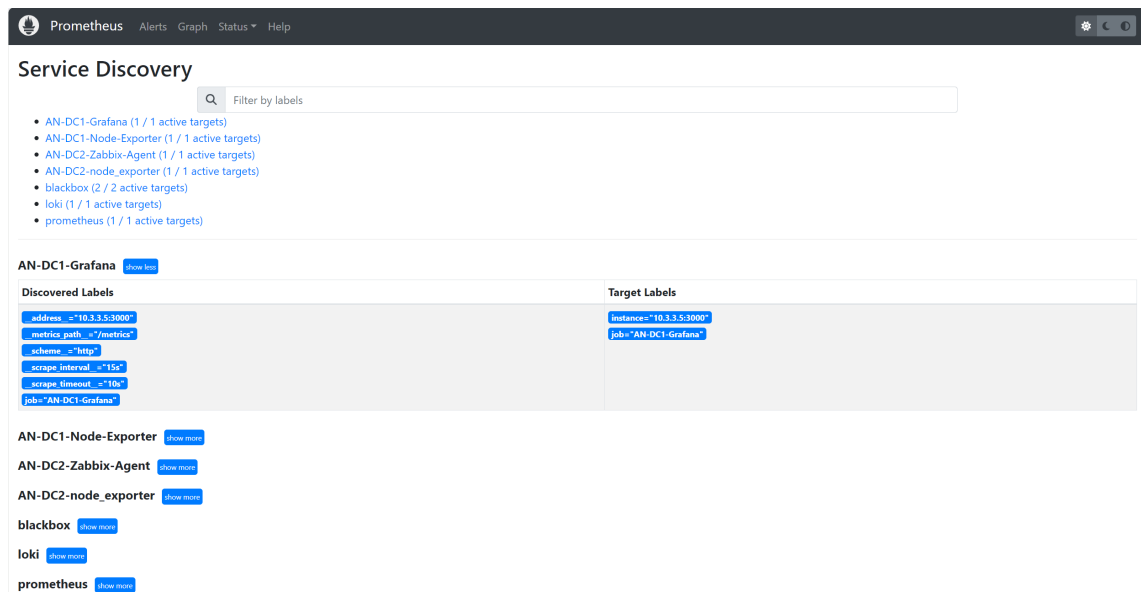


Figure 4.12. Service Discovery in Prometheus.

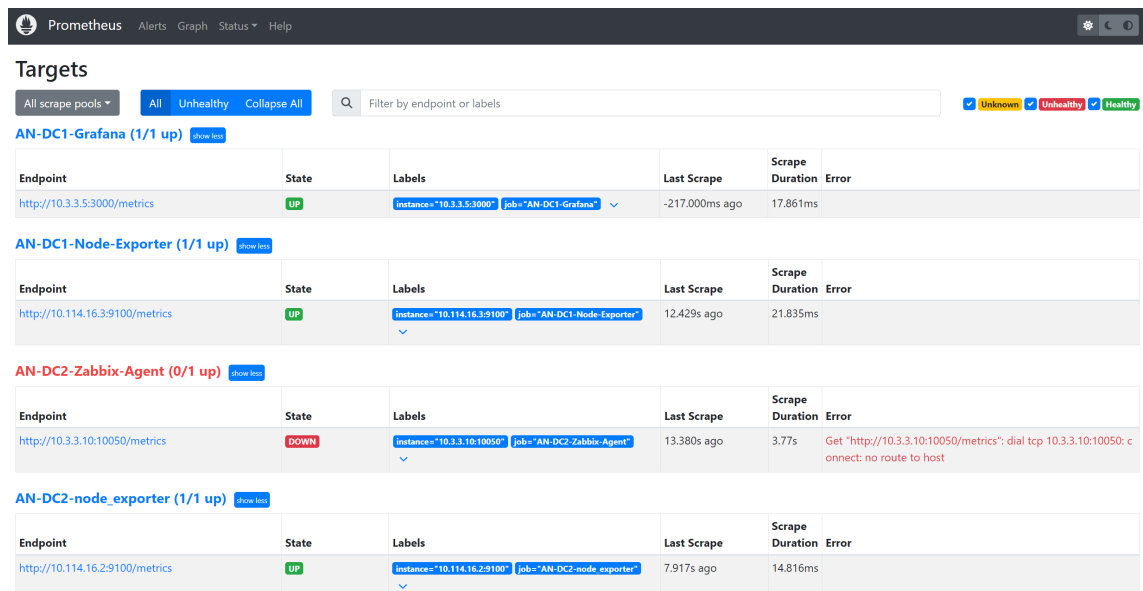
In Figure 4.13, Prometheus's Targets page is depicted, providing a crucial interface for network administrators to monitor the status of various endpoints within the network. Each row represents a monitored target, detailed with endpoint addresses and current operational states, such as 'UP' for active and 'DOWN' for inactive, which allows for immediate assessment of service availability.

The 'Labels' column offers a quick reference to the specific characteristics of each endpoint, such as the instance and job name, enhancing the manageability of the network. This labeling system is integral to the Prometheus setup as it categorizes the metrics

collected from each endpoint, enabling efficient filtering and querying.

Crucially, the 'Last Scrape' timestamp informs administrators of the most recent successful data collection attempt, providing insights into the periodicity and reliability of monitoring operations. Correspondingly, the 'Scrape Duration' indicates how long the last scrape took, serving as an indicator of the performance and health of the monitoring system itself. Any errors encountered during the scraping process are also recorded, offering immediate diagnostic information to troubleshoot potential issues.

The figure presents a clear visual summary of the health and status of network services, with color-coded indicators that provide at-a-glance recognition of system health. This real-time overview is vital for maintaining network reliability and performance, as it enables prompt response to any identified issues. The Targets page, therefore, is not just a monitoring tool, but a critical component of proactive network management, ensuring high availability and minimal service disruption.



The screenshot shows the Prometheus 'Targets' page. At the top, there are navigation links for 'Alerts', 'Graph', 'Status', and 'Help'. Below the navigation, there are filters for 'All scrape pools', 'All', 'Unhealthy', and 'Collapse All'. A search bar is present with the text 'Filter by endpoint or labels'. On the right, there are status indicators for 'Unknown', 'Unhealthy', and 'Healthy'. The main content area displays four target groups, each with a table of endpoints. The first group is 'AN-DC1-Grafana (1/1 up)', the second is 'AN-DC1-Node-Exporter (1/1 up)', the third is 'AN-DC2-Zabbix-Agent (0/1 up)', and the fourth is 'AN-DC2-node_exporter (1/1 up)'. Each table has columns for 'Endpoint', 'State', 'Labels', 'Last Scrape', 'Scrape Duration', and 'Error'.

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://10.3.3.5:3000/metrics	UP	instance="10.3.3.5:3000" job="AN-DC1-Grafana"	-217.000ms ago	17.861ms	
http://10.114.16.3:9100/metrics	UP	instance="10.114.16.3:9100" job="AN-DC1-Node-Exporter"	12.429s ago	21.835ms	
http://10.3.3.10:10050/metrics	DOWN	instance="10.3.3.10:10050" job="AN-DC2-Zabbix-Agent"	13.380s ago	3.77s	Get "http://10.3.3.10:10050/metrics": dial tcp 10.3.3.10:10050: connect: no route to host
http://10.114.16.2:9100/metrics	UP	instance="10.114.16.2:9100" job="AN-DC2-node_exporter"	7.917s ago	14.816ms	

Figure 4.13. Monitoring targets in Prometheus.

Figure 4.14 presents the Zabbix Dashboard, a critical interface for network administrators, providing a real-time overview and health status of the entire monitored IT environment. The dashboard is thoughtfully laid out, offering immediate visibility into critical network metrics, including the number of hosts and the count of active monitoring items and triggers, which are essential parameters for assessing the network's robustness.

Central to the dashboard is the 'Problems' section, which meticulously catalogs every issue detected within the network, arranged by severity level from 'Disaster' to 'Information.' This hierarchy of problems enables administrators to quickly identify and prioritize problems for troubleshooting. Each entry within this section is timestamped for temporal context and includes identification tags for the affected host, ensuring that the origin of

the issue is readily apparent. Detailed descriptions accompanying each problem entry provide actionable intelligence, allowing swift, informed response actions.

This holistic approach to network health visualization, encapsulated in the Zabbix dashboard, exemplifies the platform's utility in ensuring network resilience and operational continuity. The intuitive design of the dashboard, combined with its depth of information, makes it an indispensable tool for contemporary network management.

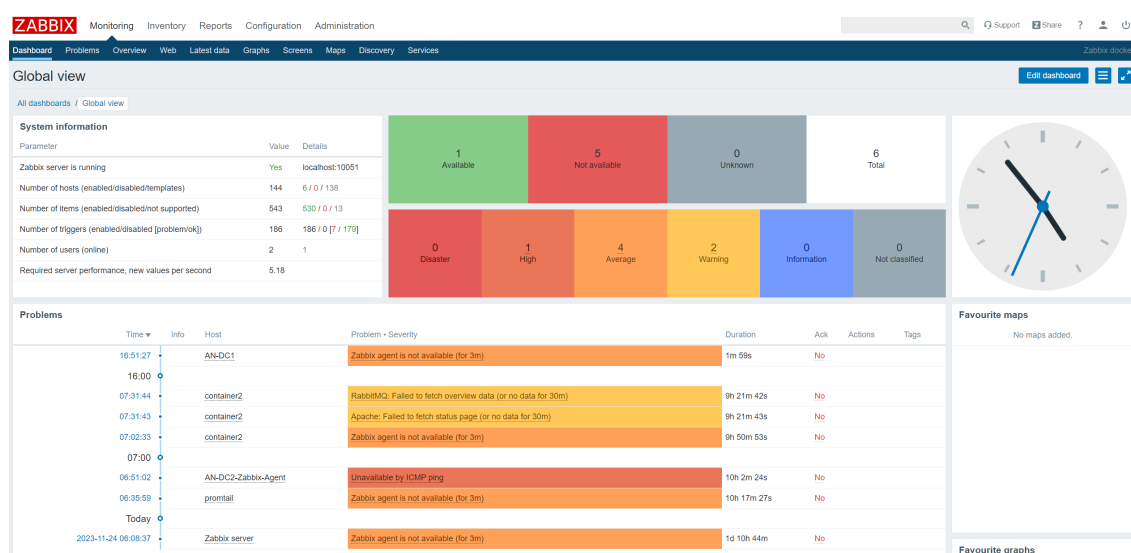


Figure 4.14. Zabbix monitoring dashboard.

Figure 4.15 presents a screenshot of the Zabbix interface during the crucial process of adding the host. This interface is integral to the Zabbix monitoring system, allowing administrators to seamlessly incorporate new network nodes. It displays a comprehensive list of hosts, each with associated details such as the number of linked applications, the number of monitoring items, and the count of defined triggers which are imperative for the operational awareness of the network.

Each host entry shows essential connection parameters like IP address and port number, offering a snapshot of the node's network footprint. The status column indicates whether the host is actively being monitored (denoted "Enabled"), and the availability column provides immediate visibility into the host's current monitoring state, with color-coded indicators reflecting the operational status.

The integration of hosts within the Zabbix platform is a testament to the system's ability to scale and adapt to new network entities. The interface is meticulously designed to ensure that administrators can efficiently manage the network's expanding scale, reflecting Zabbix's commitment to providing a dynamic, responsive network monitoring solution.

Figure 4.16 shows Zabbix's 'Problems' tab, a centralized hub for tracking network issues. Problems are listed chronologically, with color-coded severity levels from 'Information' to 'Disaster,' enabling quick prioritization. Each entry specifies the affected host and pro-

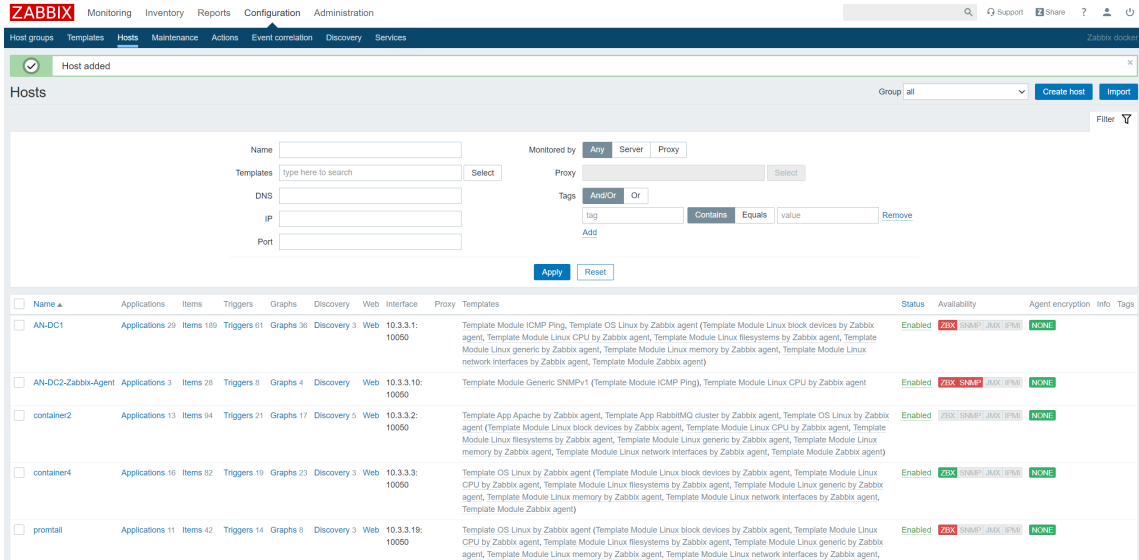


Figure 4.15. Host addition in Zabbix monitoring.

vides a brief description of the issue, such as 'Zabbix agent is not available', indicating where attention is needed. 'Recovery time' and 'Duration' fields offer temporal context to each problem, suggesting how long an issue has persisted and if it has been resolved. The dashboard also allows for problem acknowledgment and displays available actions, streamlined the troubleshooting process. The tags associated with each problem enable filtering, helping administrators categorize and navigate through issues efficiently.

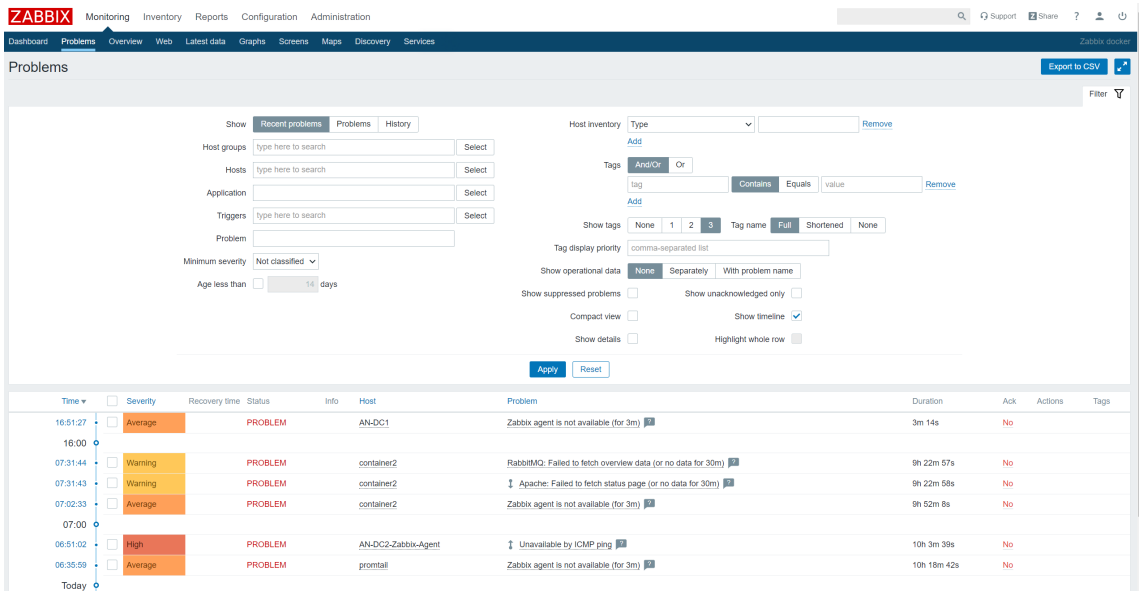


Figure 4.16. Problems and alerts in Zabbix.

Figure 4.17 illustrates Grafana Loki log aggregation in action, displaying a clear layout of log entries from multiple containers. This dashboard is instrumental for administrators, providing a time-sequenced visualization of logs, which is essential for chronological analysis. Each log is meticulously detailed, with labels that include the 'filename', indicating

the precise location of the log file within the container's filesystem. These filenames serve as a direct reference point for administrators to trace back logs to their source, which is particularly helpful in environments with extensive container setups.

The entries also include unique identifiers ('id'), which are critical for pinpointing specific logs during in-depth system analysis or debugging. Alongside the log details, one can see metadata fields such as timestamps that are precise to the nanosecond ('tsNs'), offering high-resolution insights into the timing of each logged event.

These structured log data, when used in conjunction with Loki's querying capabilities, allow for efficient searching and analysis, facilitating swift operational responses to maintain the integrity of VXLAN network operations. The integration of such detailed logs with Grafana's visualization tools exemplifies the stack's utility in real-time monitoring and troubleshooting within complex network infrastructures.

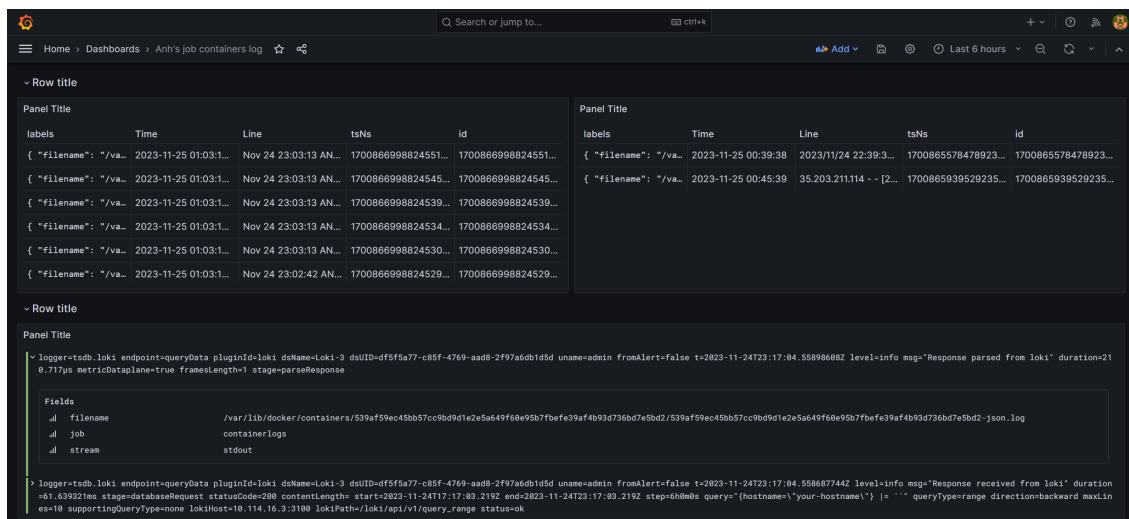


Figure 4.17. Log aggregation in Grafana Loki for container logs.

The Grafana dashboard depicted in Figure 4.18 serves as a powerful interface that synthesizes complex time series data from Prometheus in a user-friendly and interactive graphical format. Visualization likely includes various metrics such as response times, server requests, and system utilization rates, all of which are vital for real-time monitoring. The dashboard enhances the user's ability to interpret extensive Prometheus metrics quickly, pinpoint trends, identify anomalies, and make informed decisions based on the comprehensive analysis that Grafana facilitates. This integration exemplifies how Grafana acts as a window to Prometheus's extensive data, enhancing observability and operational oversight within network environments.

The PoC effectively demonstrated the functional interplay and compatibility of a comprehensive set of network management tools within a VXLAN environment. The successful implementation highlighted not only the PLG stack, but also a suite of other vital tools, including SDN solutions, Prometheus, Grafana, Zabbix, and Alertmanager. These tools

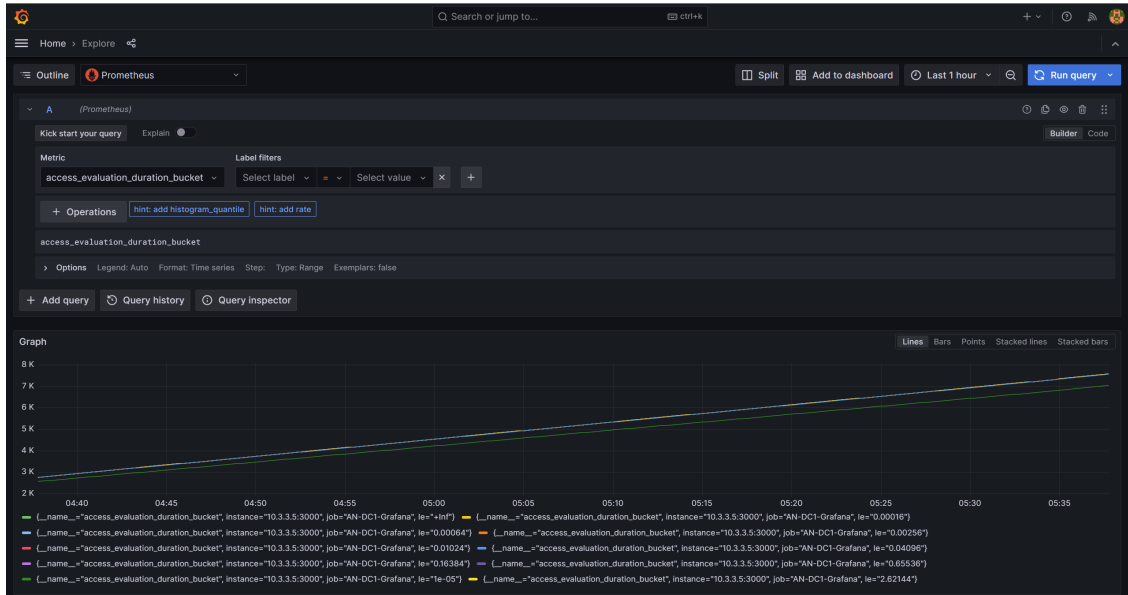


Figure 4.18. Grafana dashboard visualizing Prometheus metrics.

collectively contribute to a robust and dynamic network management system, underscoring the potential for enhanced system administration and real-time monitoring in complex network architectures.

5. CONCLUSION

As this extensive study of VXLAN technology concludes, it is clear that the journey through the complexities of VXLAN has revealed pivotal advancements in network architecture. VXLAN's emergence as a transformative solution for scalability and segmentation challenges in traditional VLANs represents a significant leap in network design. This advancement not only has enhanced network flexibility and efficiency, particularly for large-scale data centers and cloud environments, but has also introduced new layers of complexity in network management. This requires a sophisticated, integrated approach to effectively handle these advanced networks.

The exploration of SDN within this context has revealed its dynamic and transformative impact. The ability of SDN to offer flexibility and adaptability, critical in managing VXLAN infrastructures, cannot be overstated. Although SDN was not implemented in the practical component of this thesis, its theoretical importance in VXLAN network optimization is undeniably profound.

The integration of Ansible into VXLAN management has proven indispensable. Ansible has simplified network operations, enhancing reliability and efficiency in network configurations. Its automation capabilities have reduced operational complexities, enabling a smoother management of extensive network infrastructures.

The combined use of Prometheus and Grafana has emerged as a crucial aspect of this study. Prometheus, with its proficiency in collecting real-time network metrics, has become the backbone of network analysis. Grafana, with its advanced visualization tools, has brought these metrics to life, enabling comprehensive insight into network performance. This synergy has enabled detailed monitoring and proactive management within VXLAN networks.

The role of Alertmanager in network management has been vital. As part of the Prometheus ecosystem, Alertmanager has efficiently managed and routed network alerts, ensuring operational clarity and responsiveness. Its ability to deduplicate, group, and route alerts has provided a clear and organized view of network health.

The PLG stack, specifically Grafana Loki and Promtail, has played a key role in log aggregation and analysis within VXLAN networks. Grafana Loki's ability to manage large volumes of log data, coupled with Promtail's log collection capabilities, has effectively

addressed common challenges in VXLAN environments.

The PoC conducted in this study was instrumental in demonstrating the practical application and successful integration of these tools into a VXLAN setup. Using the Digital Ocean platform with two droplets named AN-DC1 and AN-DC2, each equipped with substantial computing resources, a VXLAN network was established using Open vSwitch. This setup illustrated a practical implementation of VXLAN technology, including the configuration of VXLAN tunnels and the deployment of various network management tools. The PoC provided a step-by-step guide that included the process of setting up Ansible for remote management, configuring Prometheus for monitoring, and using Grafana for visualization. This detailed depiction allows for the replication of the PoC in similar environments, thereby showcasing a practical model for the integration and utilization of these tools within VXLAN networks.

In addition to the primary tools, the thesis acknowledged the potential of the ELK stack as a robust logging solution. Although the ELK stack boasts strong capabilities, its commercial nature and associated costs led to its exclusion from the PoC. However, its inclusion in the study reflects a comprehensive evaluation of various tools suitable for different organizational needs and resources.

Looking ahead, the thesis identifies several promising areas for future research. Enhanced traffic flow control via SDN controllers in VXLAN presents an exciting avenue for exploration. The potential of deep learning to improve network performance within VXLAN, particularly in QoS enhancement, is another area that deserves investigation. Furthermore, the application of Zero Trust security principles to VXLAN networks, in response to increasing cybersecurity threats, offers challenges and opportunities for future research.

In conclusion, this thesis has provided a thorough examination of VXLAN technology and its crucial role in modern networking. The integration of various network management tools within VXLAN networks has set a new benchmark, providing valuable information for academic research and practical applications. The journey of VXLAN and its management continues, filled with opportunities for innovation and advancement in the ever-evolving field of network technology.

REFERENCES

- [1] *Alertmanager*. original-date: 2013-07-16T15:06:31Z. Oct. 27, 2023. URL: <https://github.com/prometheus/alertmanager> (visited on 10/27/2023).
- [2] Cosimo Anglano, Massimo Canonico, and Marco Guazzone. “Prometheus: A flexible toolkit for the experimentation with virtualized infrastructures”. In: *Concurrency and Computation: Practice and Experience* 30.11 (2018), e4400.
- [3] Panagiotis Apostolidis. “Network management aspects in SDN”. In: (2016).
- [4] G. Avolio et al. “A web-based solution to visualize operational monitoring data in the Trigger and Data Acquisition system of the ATLAS experiment at the LHC”. In: *Journal of Physics: Conference Series* 898 (2017). DOI: 10.1088/1742-6596/898/3/032010.
- [5] Joel Bastos and Pedro Araújo. *Hands-On Infrastructure Monitoring with Prometheus: Implement and scale queries, dashboards, and alerting across machines and containers*. Packt Publishing Ltd, 2019.
- [6] Elizabeth Bautista, Nitin Sukhija, and Siqi Deng. “Shasta Log Aggregation, Monitoring and Alerting in HPC Environments with Grafana Loki and ServiceNow”. In: *2022 IEEE International Conference on Cluster Computing (CLUSTER)*. 2022, pp. 602–610. DOI: 10.1109/CLUSTER51413.2022.00079.
- [7] Mehmet Beyaz. “Development Methodologies for Network Softwarization: A Comparison of DevOps, NetOps, and Verification”. In: *International Journal of Communications, Network and System Sciences* 16.5 (May 11, 2023), pp. 97–104. DOI: 10.4236/ijcns.2023.165007. URL: <https://www.scirp.org/journal/paperinformation.aspx?paperid=125397> (visited on 10/20/2023).
- [8] I. V. Bilobrovets. “Network threat detection technology using Zabbix software”. In: *Modern Information Security* (2023). URL: <https://api.semanticscholar.org/CorpusID:259795828>.
- [9] Blessing Bwalya and Aaron Zimba. “An SDN approach to mitigating network management challenges in traditional networks”. In: *International Journal on Information Technologies and Security* 13.4 (2021), pp. 3–14.
- [10] Martin Casado, Nate Foster, and Arjun Guha. “Abstractions for software-defined networks”. In: *Communications of the ACM* 57.10 (2014), pp. 86–95.
- [11] Deepak Chahal, Latika Kharb, and Deepanshu Choudhary. “Performance analytics of network monitoring tools”. In: *Int. J. Innov. Technol. Explor. Eng.(IJITEE)* 8.8 (2019).

- [12] Mainak Chakraborty and Ajit Pratap Kundan. "Grafana". In: *Monitoring Cloud-Native Applications: Lead Agile Operations Confidently Using Open Source Software*. Berkeley, CA: Apress, 2021, pp. 187–240. ISBN: 978-1-4842-6888-9. DOI: 10.1007/978-1-4842-6888-9_6. URL: https://doi.org/10.1007/978-1-4842-6888-9_6.
- [13] Alexander Craig et al. "Load balancing for multicast traffic in SDN using real-time link cost modification". In: *2015 IEEE international conference on communications (ICC)*. IEEE. 2015, pp. 5789–5795.
- [14] Avri Doria et al. *Forwarding and control element separation (ForCES) protocol specification*. Tech. rep. 2010.
- [15] Joakim Eriksson and Anawil Karavek. *A comparative analysis of log management solutions: ELK stack versus PLG stack*. 2023.
- [16] Mohd Faris Mohd Fuzi et al. "Network Automation using Ansible for EIGRP Network". In: *Journal of Computing Research and Innovation* (2021). URL: <https://api.semanticscholar.org/CorpusID:240599269>.
- [17] *Grafana | Query, visualize, alerting observability platform*. Grafana Labs. URL: <https://grafana.com/grafana/> (visited on 10/28/2023).
- [18] *Grafana Labs*. GitHub. URL: <https://github.com/grafana> (visited on 10/28/2023).
- [19] Cáo Tiân Guì. "Application of Network Systems Management Cloud Service Based on Open Source Zabbix". In: 2018. URL: <https://api.semanticscholar.org/CorpusID:64672536>.
- [20] Ansible Hat Red. *Ansible is Simple IT Automation*. URL: <https://www.ansible.com> (visited on 10/20/2023).
- [21] Sanjay Hooda, Shyam Kapadia, and Padmanabhan Krishnan. *Using Trill, Fabric-Path, and VXLAN: designing massively scalable data centers (MSDC) with overlays*. Cisco Press, 2014.
- [22] Muhammad Islami, Purnawarman Musa, and Missa Lamsani. "Implementation of Network Automation Using Ansible to Congure Routing Protocol in Cisco and Mikrotik Router with Raspberry Pi". In: 19 (June 2020). DOI: 10.32409/jikstik.19.2.80.
- [23] Svetoslav Sergeev Kamenov. "Experimental Monitoring on Network Based Tactile Sensing System". In: *2019 IEEE XXVIII International Scientific Conference Electronics (ET)*. 2019, pp. 1–4. DOI: 10.1109/ET.2019.8878661.
- [24] Karamjeet Kaur, Japinder Singh, and Navtej Singh Ghumman. "Mininet as software defined networking testing platform". In: *International conference on communication, computing & systems (ICCCS)*. 2014, pp. 139–42.
- [25] Rahamatullah Khondoker et al. "Feature-based comparison and selection of Software Defined Networking (SDN) controllers". In: *2014 world congress on computer applications and information systems (WCCAIS)*. IEEE. 2014, pp. 1–7.
- [26] Tatsuaki Kimura et al. "Proactive failure detection learning generation patterns of large-scale network logs". In: *IEICE Transactions on Communications* 102.2 (2019), pp. 306–316.

- [27] Diego Kreutz et al. "Software-defined networking: A comprehensive survey". In: *Proceedings of the IEEE* 103.1 (2014), pp. 14–76.
- [28] Priyesh Kumar et al. "A programmable and managed software defined network". In: *International Journal of Computer Network and Information Security* 10.12 (2017), p. 11.
- [29] Dmitry Lambert. *Multi-tenant monitoring: how to achieve that using free Zabbix open-source monitoring software*. Zabbix Blog. Oct. 1, 2020. URL: <https://blog.zabbix.com/multi-tenant-monitoring-how-to-achieve-that-using-free-zabbix-open-source-monitoring-software/12024/> (visited on 10/27/2023).
- [30] Jin-Shyan Lee and Pau-Lo Hsu. "Design and implementation of the SNMP agents for remote monitoring and control via UML and Petri nets". In: *IEEE Transactions on Control Systems Technology* 12.2 (2004), pp. 293–302. DOI: 10.1109/TCST.2004.824287.
- [31] Steven SW Lee et al. "Design of SDN based large multi-tenant data center networks". In: *2015 IEEE 4th International Conference on Cloud Networking (Cloud-Net)*. IEEE. 2015, pp. 44–50.
- [32] Aaron Leskiw. *Syslog: Servers, Messages & Security - Tutorial & Defintion [Free Tool!]* Network Management Software - Reviews & Network Monitoring Tools. Sept. 13, 2018. URL: <https://www.networkmanagementsoftware.com/what-is-syslog/> (visited on 11/04/2023).
- [33] Jiaqiang Liu, Yong Li, and Depeng Jin. "SDN-based live VM migration across data-centers". In: *ACM SIGCOMM Computer Communication Review* 44.4 (2014), pp. 583–584.
- [34] *Loki overview | Grafana Loki documentation*. en. URL: <https://grafana.com/docs/loki/latest/get-started/overview/> (visited on 11/26/2023).
- [35] Eduard Luchian et al. "Automation of the infrastructure and services for an open-stack deployment using chef tool". In: *2016 15th RoEduNet Conference: Networking in Education and Research*. 2016, pp. 1–5. DOI: 10.1109/RoEduNet.2016.7753200.
- [36] Lukas Macura and M. Voznák. "Multi-Criteria Analysis and Prediction of Network Incidents Using Monitoring System". In: *J. Adv. Eng. Comput.* 1 (2017), pp. 29–38. DOI: 10.25073/JAEC.201711.47.
- [37] Mallik Mahalingam et al. *Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks*. Request for Comments RFC 7348. Num Pages: 22. Internet Engineering Task Force, Aug. 2014. DOI: 10.17487/RFC7348. URL: <https://datatracker.ietf.org/doc/rfc7348> (visited on 10/29/2023).
- [38] Ibrahim Yahya Mohammed Al-Mahbashi, M. B. Potdar, and Prashant Chauhan. "Network security enhancement through effective log analysis using ELK". In: *2017*

- International Conference on Computing Methodologies and Communication (IC-CMC)*. 2017, pp. 566–570. DOI: 10.1109/ICCMC.2017.8282530.
- [39] Anggi Mardiyono, Walidatush Sholihah, and Faisal Hakim. “Mobile-based Network Monitoring System Using Zabbix and Telegram”. In: *2020 3rd International Conference on Computer and Informatics Engineering (IC2IE)*. 2020, pp. 473–477. DOI: 10.1109/IC2IE50715.2020.9274582.
- [40] Arturo Mayoral et al. “Experimental seamless virtual machine migration using an integrated SDN IT and network orchestrator”. In: *2015 Optical Fiber Communications Conference and Exhibition (OFC)*. IEEE. 2015, pp. 1–3.
- [41] Nick McKeown et al. “OpenFlow: enabling innovation in campus networks”. In: *ACM SIGCOMM computer communication review* 38.2 (2008), pp. 69–74.
- [42] MetricFire. *Prometheus vs. Zabbix | MetricFire Blog*. URL: <https://www.metricfire.com/blog/prometheus-vs-zabbix/> (visited on 10/27/2023).
- [43] Yukihiro Nakagawa, Kazuki Hyoudou, and Takeshi Shimizu. “A Management Method of IP Multicast in Overlay Networks Using Openflow”. In: *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*. HotSDN ’12. Helsinki, Finland: Association for Computing Machinery, 2012, pp. 91–96. ISBN: 9781450314770. DOI: 10.1145/2342441.2342460. URL: <https://doi.org/10.1145/2342441.2342460>.
- [44] Edison F. Naranjo and Gustavo D. Salazar Ch. “Underlay and overlay networks: The approach to solve addressing and segmentation problems in the new networking era: VXLAN encapsulation with Cisco and open source networks”. In: *2017 IEEE Second Ecuador Technical Chapters Meeting (ETCM)*. 2017, pp. 1–6. DOI: 10.1109/ETCM.2017.8247505.
- [45] “Network monitoring tools: enabling the management of network assets, ensuring high availability”. In: *International Journal of Development Research* (2020). DOI: 10.37118/ijdr.19224.06.2020.
- [46] Wendell Odom. *CCNA 200-301 Official Cert Guide, Volume 2*. Cisco Press, 2019.
- [47] Ben Pfaff et al. “The design and implementation of open {vSwitch}”. In: *12th USENIX symposium on networked systems design and implementation (NSDI 15)*. 2015, pp. 117–130.
- [48] *PostgreSQL monitoring using Zabbix Agent 2: easy and extensible*. Nov. 24, 2020. URL: <https://postgrespro.com/blog/pgsql/5967895> (visited on 10/27/2023).
- [49] Randy Presuhn. *Version 2 of the protocol operations for the simple network management protocol (SNMP)*. Tech. rep. 2002.
- [50] *Prometheus*. original-date: 2012-11-24T11:14:12Z. Oct. 27, 2023. URL: <https://github.com/prometheus/prometheus> (visited on 10/27/2023).
- [51] Prometheus. *Alertmanager | Prometheus*. URL: <https://prometheus.io/docs/alerting/latest/alertmanager/> (visited on 10/27/2023).
- [52] Prometheus. *Comparison to alternatives | Prometheus*. URL: <https://prometheus.io/docs/introduction/comparison/> (visited on 10/27/2023).

- [53] Prometheus. *Exporters and integrations | Prometheus*. URL: <https://prometheus.io/docs/instrumenting/exporters/> (visited on 10/27/2023).
- [54] Prometheus. *Overview | Prometheus*. URL: <https://prometheus.io/docs/introduction/overview/> (visited on 10/27/2023).
- [55] Prometheus. *Querying basics | Prometheus*. URL: <https://prometheus.io/docs/prometheus/latest/querying/basics/> (visited on 10/27/2023).
- [56] *Promtail agent | Grafana Loki documentation*. en. URL: <https://grafana.com/docs/loki/latest/send-data/promtail/> (visited on 11/26/2023).
- [57] S. Rajalakshmi and A. Kannaki@VasanthaAzagu. "Autonomous Infrastructure Provisioning In AWS". In: *International Journal of Innovative Research in Engineering* 4.1 (2023), pp. 110–112.
- [58] Kartika Rianafirin and Mochamad Teguh Kurniawan. "Design network security infrastructure cabling using network development life cycle methodology and ISO/IEC 27000 series in Yayasan Kesehatan (Yakes) Telkom Bandung". In: *2017 4th International Conference on Computer Applications and Information Processing Technology (CAIPT)*. 2017, pp. 1–6. DOI: 10.1109/CAIPT.2017.8320681.
- [59] Juan Rios. "MAINTAINING ZERO TRUST WITH ELK". PhD thesis. CALIFORNIA STATE UNIVERSITY SAN MARCOS, 2023.
- [60] Adian F. Rochim, Mukhlis A. Aziz, and Adnan Fauzi. "Design Log Management System of Computer Network Devices Infrastructures Based on ELK Stack". In: *2019 International Conference on Electrical Engineering and Computer Science (ICECOS)*. 2019, pp. 338–342. DOI: 10.1109/ICECOS47637.2019.8984494.
- [61] Navin Sabharwal and Piyush Pandey. "Getting Started with Prometheus and Alert Manager". In: *Monitoring Microservices and Containerized Applications: Deployment, Configuration, and Best Practices for Prometheus and Alert Manager*. Berkeley, CA: Apress, 2020, pp. 43–83. ISBN: 978-1-4842-6216-0. DOI: 10.1007/978-1-4842-6216-0_3. URL: https://doi.org/10.1007/978-1-4842-6216-0_3.
- [62] Gustavo Salazar-Chacon and Luis Marrone. "Open Networking for Modern Data Centers Infrastructures: VXLAN Proof-of-Concept Emulation using LNV and EVPN under Cumulus Linux". In: *2022 IEEE Sixth Ecuador Technical Chapters Meeting (ETCM)*. 2022, pp. 1–6. DOI: 10.1109/ETCM56276.2022.9935681.
- [63] Gustavo Salazar-Chacón and Diego Marcillo Parra. "Infrastructure-as-Code in Open Networking: Git, Ansible, and Cumulus-Linux Case Study". In: *2023 IEEE 13th Annual Computing and Communication Workshop and Conference (CCWC)*. 2023, pp. 1126–1131. DOI: 10.1109/CCWC57344.2023.10099084.
- [64] Gustavo D. Salazar-Chacón and Luis Marrone. "OpenSDN Southbound Traffic Characterization: Proof-of-Concept Virtualized SDN-Infrastructure". In: *2020 11th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*. 2020, pp. 0282–0287. DOI: 10.1109/IEMCON51383.2020.9284938.

- [65] Dr Bhargavi Goswami Saleh Asadollahi and Atul M Gonsai. "Implementation of SDN using OpenDayLight Controller". In: ().
- [66] S Savitha et al. "Auto Scaling Infrastructure with Monitoring Tools using Linux Server on Cloud". In: *2023 7th International Conference on Computing Methodologies and Communication (ICCMC)*. IEEE. 2023, pp. 45–52.
- [67] Jay Shah, Dushyant Dubaria, and John Widhalm. "A Survey of DevOps tools for Networking". In: *2018 9th IEEE Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON)*. 2018, pp. 185–188. DOI: 10.1109/UEMCON.2018.8796814.
- [68] J. Sierra-Fernández et al. "Online System for Power Quality Operational Data Management in Frequency Monitoring using Python and Grafana". In: *Energies* (2021). DOI: 10.20944/preprints202110.0260.v1.
- [69] *Software defined Networking (SDN)*. GeeksforGeeks. Section: Computer Networks. July 2, 2019. URL: <https://www.geeksforgeeks.org/software-defined-networking/> (visited on 10/30/2023).
- [70] *Software-Defined Networking (SDN) Definition*. Open Networking Foundation. URL: <https://opennetworking.org/sdn-definition/> (visited on 10/20/2023).
- [71] Haoyu Song. "Protocol-oblivious forwarding: Unleash the power of SDN through a future-proof forwarding plane". In: *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*. 2013, pp. 127–132.
- [72] Mihail Alexandru STAN. "Automation of Log Analysis Using the Hunting ELK Stack". In: *Romanian Cyber Security Journal* 3.1 (2021).
- [73] *Starting with the Elasticsearch Platform and its Solutions | Elastic*. en-us. Learn/Docs/. URL: <https://www.elastic.co/guide/index.html> (visited on 11/26/2023).
- [74] Răzvan Stoleriu, Alin Puncioiu, and Ion Bica. "Cyber Attacks Detection Using Open Source ELK Stack". In: *2021 13th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*. 2021, pp. 1–6. DOI: 10.1109/ECAI52376.2021.9515120.
- [75] Patricie Suppala. "FinOps in SaaS platform within hybrid, multi-cloud, multi-tenant, multi-region environments". In: (2022).
- [76] *The cost of human error and the advantages of security automation*. URL: <https://www.redhat.com/en/resources/cost-human-error-advantage-security-automation-ebook> (visited on 10/20/2023).
- [77] *The Forrester Wave™: Infrastructure Automation, Q1 2023*. URL: <https://www.redhat.com/en/resources/forrester-wave-infrastructure-automation-analyst-asset> (visited on 10/20/2023).
- [78] Claudiu Trăistaru. "VXLAN - A practical approach to cloud computing scalability". In: *2023 22nd RoEduNet Conference: Networking in Education and Research (RoEduNet)*. 2023, pp. 1–4. DOI: 10.1109/RoEduNet60162.2023.10274934.

- [79] Juha Vihervaara. *Computer Networking II Textbook*. 2022. URL: https://moodle.tuni.fi/pluginfile.php/1282992/mod_resource/content/16/ComputerNetworking2_2022.pdf.
- [80] *VXLAN vs VLAN: Differences, Advantages & Disadvantages*. Windows Report - Your go-to source for PC tutorials. July 13, 2023. URL: <https://windowsreport.com/VXLAN-vs-vlan/> (visited on 10/30/2023).
- [81] *VXLAN: the Future for Data Center Networks | FS Community*. Knowledge. May 19, 2022. URL: <https://community.fs.com:7003/article/VXLAN-the-future-for-data-center-networks.html> (visited on 10/29/2023).
- [82] *What is ELK stack? - Elasticsearch, Logstash, Kibana Stack Explained - AWS*. Amazon Web Services, Inc. URL: <https://aws.amazon.com/what-is/elk-stack/> (visited on 11/04/2023).
- [83] *What Is Software-Defined Networking (SDN)? | FS Community*. Knowledge. May 24, 2022. URL: <https://community.fs.com:7003/article/what-is-software-defined-networking-sdn.html> (visited on 10/30/2023).
- [84] *What is Software-Defined Networking (SDN)? | VMware Glossary*. VMware. URL: <https://www.vmware.com/topics/glossary/content/software-defined-networking.html> (visited on 10/30/2023).
- [85] *What is Software-Defined Networking?* URL: <https://www.ibm.com/topics/sdn> (visited on 10/30/2023).
- [86] *What Is VXLAN*. July 12, 2018. URL: <https://support.huawei.com/enterprise/en/doc/EDOC1100086966> (visited on 07/12/2018).
- [87] *What is VXLAN and VLAN: Advantages and Implementation*. NAKIVO. May 10, 2022. URL: <https://www.nakivo.com/blog/VXLAN-vmware-basics/> (visited on 10/30/2023).
- [88] *What's in Zabbix 5.0 LTS*. URL: https://www.zabbix.com/whats_new_5_0 (cit. 27. 10. 2023).
- [89] *Zabbix Manual*. URL: <https://www.zabbix.com/documentation/current/en/manual> (visited on 10/19/2023).
- [90] Ying Zhang. *Network Function Virtualization: Concepts and Applicability in 5G Networks*. Wiley-IEEE Press, 2018. 192 pp.
- [91] Zhifeng Zhao, Feng Hong, and Rongpeng Li. "SDN Based VXLAN Optimization in Cloud Computing Networks". In: *IEEE Access* 5 (2017), pp. 23312–23319. DOI: 10.1109/ACCESS.2017.2762362.

APPENDIX A: YAML CONFIGURATION FILES

This chapter provides the configuration details for AlertManager, Prometheus, and Zabbix, which are integral to the PoC presented in this document.

A.1 Alertmanager Configuration

The configuration file for Alertmanager, depicted in Figure A.1, serves as a fundamental blueprint for implementing alert routing, notification management, and defining how alerts are grouped, acknowledged, and silenced. In the file, we see the `global` block specifying the Simple Mail Transfer Protocol (SMTP) settings, which are critical for the dispatch of email alerts. This includes the SMTP server address and port, the email address from which alerts will be sent, and the authentication credentials necessary for server access.

The `route` block beneath defines the grouping parameters of the alerts, such as grouping them by alert names, which is an essential feature to reduce noise and prevent alert fatigue. It also details the waiting times before a new group of alerts is sent (`group_wait`), the interval between repeated alerts (`group_interval`), and how often a repeated alert should be sent (`repeat_interval`). These timings are crucial for managing the flow of alert notifications without overwhelming the recipients.

The `receiver` line within the `route` block designates the receiver for the alerts, which in this case is specified as `email-notifications`. Below this, the `receivers` block provides configurations for defining the receiver, including the email address to which notifications will be sent. The `send_resolved` flag, when set to `true`, ensures that notifications are also sent out when an alert has been resolved, closing the loop on an alert lifecycle.

The `inhibit_rules` section allows the suppression of notifications for certain alerts if another alert with higher severity is already running. This is configured using `source_match` and `target_match` labels that align with severity levels, preventing less critical alerts from overshadowing more urgent ones.

This YAML configuration file is crucial for the Alertmanager's operation, translating these declarative configurations into operational behaviors that ensure alerts are handled efficiently and effectively. It reflects the operational logic for managing alerts and embodies

```

GNU nano 6.2
# Global settings for the alertmanager
global:
  # The SMTP smarthost and port used for sending emails
  smtp_smarthost: 'smtp.gmail.com:587'
  # The email address to send emails from
  smtp_from: '[redacted]@gmail.com'
  # The SMTP user authentication details
  smtp_auth_username: '[redacted]@gmail.com'
  smtp_auth_password: '[redacted]'

# The route subsection defines how alerts are grouped and routed to receivers
route:
  # Group alerts by alertname
  group_by: ['alertname']
  # Wait for 30 seconds to group alerts before sending them
  group_wait: 30s
  # Time to wait before sending a new group of alerts
  group_interval: 5m
  # Time to wait before sending the same alert again
  repeat_interval: 1h
  # Designate the receiver for these alerts
  receiver: 'email-notifications'

# Define a list of receivers
receivers:
- name: 'email-notifications'
  # Email configuration for the receiver
  email_configs:
  - to: '[redacted]@tuni.fi'
    # Send a notification when an alert is resolved
    send_resolved: true

# Inhibit rules to suppress notifications for certain alerts
inhibit_rules:
- source_match:
  severity: 'critical'
  target_match:
  severity: 'warning'
  # The labels that must be equal for the inhibition to take place
  equal: ['alertname', 'dev', 'instance']

```

Figure A.1. Alertmanager YAML configuration file.

the Alertmanager's capability to streamline incident response through structured and automated alert handling.

A.2 Prometheus Configuration

As shown in Figure A.2, the configuration file for the Prometheus server delineates a host of critical parameters that guide the monitoring service's behavior. This YAML file is divided into several sections, each with a distinct purpose. The `rule_files` block lists the rule files to be loaded by Prometheus, allowing the service to periodically evaluate the rules based on the `global evaluation_interval`.


```

cloud.digitalocean.com/droplets/386591580/terminal/ui/
# - alertmanager:9093

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

# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
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: "AN-DC1-Grafana"
    static_configs:
      - targets: ["10.3.3.5:3000"]
  - job_name: "AN-DC2-node_exporter"
    static_configs:
      - targets: ["10.114.16.2:9100"]
  - job_name: 'AN-DC2-Zabbix-Agent'
    static_configs:
      - targets: ["10.3.3.10:10050"][]
  - job_name: "AN-DC1-Node-Exporter"
    static_configs:
      - targets: ["10.114.16.3:9100"]
  - job_name: 'blackbox'
    metrics_path: /probe
    params:
      module: [http_2xx]
    static_configs:
      - targets:
        - https://www.google.com.vn/
        - http://10.3.3.5:3000
        - http://10.3.3.4:9090
  relabel_configs:
    - source_labels: [__address__]
      target_label: __param_target
    - source_labels: [__param_target]
      target_label: instance
    - target_label: __address__
      replacement: 10.3.3.101:9115
  - job_name: 'loki'
    static_configs:
      - targets: ['http://10.114.16.3:3100']
~
~
~
I /etc/prometheus/prometheus.yml [Modified] 38/63 60%

```

Figure A.2. Prometheus YAML configuration file.

Following this, the `scrape_configs` section comprises the core of the monitoring configuration, where each scrape target is meticulously defined. For instance, the job name 'AN-DC1-Grafana' corresponds to a specific scrape target, which Prometheus will poll to retrieve metrics. These targets are not just limited to internal endpoints, but may also include external ones, as indicated by the various URLs listed.

This section of the configuration file also specifies the job names for each set of targets, which become part of the time series data. The `metrics_path` and `scheme` parameters within this section default to `/metrics` and `http`, respectively, but can be tailored as needed for different endpoints.

The `static_configs` blocks within `scrape_configs` detail the actual endpoints to be monitored, each associated with a respective job name. Here, parameters such as `targets` list the individual scrape endpoints, while `labels` provide additional context for the data being collected, enhancing the granularity and usability of the metrics.

Moreover, the file contains advanced configurations such as `relabel_configs`, which allow for the alteration of labels before they are ingested into the time series database. This enables more sophisticated control over how metrics are categorized and handled.

This carefully structured configuration ensures that Prometheus can not only collect and store meaningful metrics efficiently, but also allows for significant customization to adapt to the varied needs of a complex monitoring environment.

A.3 Zabbix Agent Configuration

Figure A.3 displays the Zabbix agent configuration file opened for editing. This file is a cornerstone of the agent's functionality, as it dictates the behavior of the agent in the network monitoring setup. The 'ServerActive' parameter is set to the IP address of the Zabbix server, which is crucial for establishing a connection between the agent and the server for active checks.

The 'Hostname' parameter is left blank, which implies that the hostname will be autodetected or is defined elsewhere, typically within the Zabbix server's configuration. The 'HostnameItem' setting, defaulted to 'system.hostname', specifies that the agent should use the system's hostname as its identifier if no explicit 'Hostname' is provided.

Further down, the configuration file includes parameters 'HostMetadata' and 'HostMetadataItem', which are used in the host auto-registration process. The metadata provided here enriches the data collected by the Zabbix server, allowing for more nuanced identification and categorization of the host within the monitoring system.

This configuration snapshot is pivotal as it encapsulates the specific settings that enable the Zabbix agent to report metrics to the server, thus playing a significant role in the operational integrity of the monitoring infrastructure outlined in the PoC.

```

GNU nano 6.2 /etc/zabbix/zabbix_agentd.conf
# Default:
# ServerActive=

ServerActive=10.3.3.9

### Option: Hostname
#     Unique, case sensitive hostname.
#     Required for active checks and must match hostname as configured on the server.
#     Value is acquired from HostnameItem if undefined.
#
# Mandatory: no
# Default:
# Hostname=

### Option: HostnameItem
#     Item used for generating Hostname if it is undefined. Ignored if Hostname is defined.
#     Does not support UserParameters or aliases.
#
# Mandatory: no
# Default:
# HostnameItem=system.hostname

### Option: HostMetadata
#     Optional parameter that defines host metadata.
#     Host metadata is used at host auto-registration process.
#     An agent will issue an error and not start if the value is over limit of 255 characters.
#     If not defined, value will be acquired from HostMetadataItem.
#
# Mandatory: no
# Range: 0-255 characters
# Default:
# HostMetadata=

### Option: HostMetadataItem
#     Optional parameter that defines an item used for getting host metadata.
#     Host metadata is used at host auto-registration process.
#     During an auto-registration request an agent will log a warning message if
#     the value returned by specified item is over limit of 255 characters.
#     This option is only used when HostMetadata is not defined.
#
#
^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^I Execute    ^C Location   M-U Undo
^X Exit      ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^_ Go To Line  M-E Redo

```

Figure A.3. Zabbix agent YAML configuration file.