International Journal of Electronics and Communications System
Volume 4, Issue 2, 87-97.
ISSN: 2798-2610
http://ejournal.radenintan.ac.id/index.php/IJECS/index
DOI: 10.24042/ijecs.v4i2.23821

Automation of Software Defined Network (SDN) Configuration Management Based on Proxmox with Ansible

Khairan Marzuki
Universitas Bumigora, Mataram, INDONESIA

Husain*
Universitas Bumigora, Mataram, INDONESIA

Zikriati Apriliana
Universitas Bumigora, Mataram, INDONESIA

Article Info

Article history:

Received: August 18, 2024
Revised: October 10, 2024
Accepted: December 02, 2024
Published: December 15, 2024

Keywords:

Network Automation;
Software-Defined Networking (SDN);
Proxmox VE;
Virtual Network Management.

Abstract

Automation of software-defined networking (SDN) in Proxmox VE 8.1 enhances virtual network management by improving efficiency, scalability, and the speed of network configuration changes. This study aims to automate SDN configuration management in Proxmox VE using Ansible and evaluate its performance. The research follows the Network Development Life Cycle (NDLC) method, consisting of six stages: Analysis, Design, Simulation Prototyping, Implementation, Monitoring, and Management. Automation was implemented successfully using an Ansible playbook to manage creating and deleting zones, VNets, subnets, IP gateways, and DHCP ranges. The automated process was tested over three trials, with creation times of 23, 20, and 21 seconds and deletion times of 20, 19, and 20 seconds, respectively. By contrast, the manual process required 6 minutes 6 seconds for creation and 2 minutes 37 seconds for deletion. These results demonstrate that automation using Ansible significantly reduces configuration time, offering a more efficient and reliable approach than manual methods. The findings highlight the potential of Ansible to streamline SDN management in Proxmox VE, saving time, energy, and resources while ensuring scalability and consistency in virtualized network environments.

To cite this article: K. Marzuki, H. Husain, and Z. Apriliana. "Automation of Software Defined Network (SDN) Configuration Management Based on Proxmox with Ansible," *Int. J. Electron. Commun. Syst.*, vol. 4, no. 2, pp. 87-97, 2024.

INTRODUCTION

The rapid development of information technology continues to enhance various aspects of human life. Among these advancements, virtualization technology has emerged as a transformative innovation, enabling a single physical machine to simultaneously act as a shared resource for multiple services [1], [2]. Proxmox Virtual Environment (VE), an open-source virtualization platform based on Linux Debian, is widely utilized for managing hardware and operating system virtualizations. It facilitates container management, virtual machines, storage, virtual networks, and high-availability clusters through a web interface and command line [3]. However, manual management of Proxmox configurations often leads to human errors, such as incorrect data or code input, potentially causing system errors or downtime [4].

Software Defined Networking (SDN) is a significant advancement in network virtualization, particularly in Proxmox VE. SDN in Proxmox VE offers enhanced flexibility by separating control and data planes, enabling comprehensive control over virtual network guests through software-defined configurations [5]. Among the automation tools available, Ansible has become an open-source platform for provisioning servers, managing configurations, and deploying applications [6].

Previous studies have addressed various aspects of network automation. For example, Ikhsan [7] focused on implementing Network Controllers for centralized configuration management of AAA protocols, DNS, and

• **Corresponding author:**
Universitas Bumigora, Mataram, INDONESIA. ✉ husain@universitassbumigora.ac.id

monitoring infrastructure. Similarly, Marzuki et al. [8] demonstrated the automation of virtual network configurations using VSwitch and Ansible. Chandrawaty and Hariyadi [9] investigated using Ansible Playbooks to automate VLAN configurations on Cisco devices. However, these studies focus on Cisco-based network controllers, quality of service (QoS) applications, and VLAN or VXLAN automation.

Despite these advancements, significant gaps remain in managing Proxmox VE SDN configurations. Existing studies fail to address comprehensive control over virtual network guests, the separation of features and limitations for virtual networks, and the integration of monitoring capabilities for VNets and connected guests. Moreover, managing configurations for multiple zones, Vnets, and subnets remains time-consuming and resource-intensive [10], [11].

This study proposes automating SDN network configuration management on Proxmox VE using Ansible as an automation tool to address these limitations. The proposed approach minimizes configuration errors and accelerates processes by automating tasks such as creating and deleting Simple Zones, Vnets, Subnets, and DHCP ranges. Unlike previous research, this study provides a detailed design and implementation of automated Proxmox VE SDN configuration management, including monitoring capabilities through the IP Address Management (IPAM) menu. The novelty of this research lies in its integration of comprehensive automation and monitoring functionalities to streamline network management while reducing human error. This innovation is expected to significantly improve the efficiency and reliability of SDN management in Proxmox VE environments.

METHOD

This research employs the Network Development Life Cycle (NDLC) method, a structured framework widely used in network-related projects to streamline system development and ensure reliability. The NDLC method builds upon established processes, such as business strategy planning, application development lifecycle, and data distribution analysis [12], [13]. It comprises six stages: analysis, design, simulation prototyping, implementation, monitoring, and management. However, this study focuses on three primary stages: Analysis, Design, Simulation, and Prototyping, described in detail below.

Analysis Stage

The authors identified inefficiencies and challenges associated with manually configuring Proxmox-based SDN environments in the analysis stage. Common issues include high susceptibility to human error—such as misconfigurations—that can disrupt system functionality and require significant rectification time. Data on frequently encountered problems in manual SDN configuration were collected and analyzed [14], [15]. This analysis formed the basis for designing an automation system using Ansible, addressing key requirements such as minimizing configuration errors and enhancing operational efficiency. By focusing on these issues, this stage provided a strong foundation for tailoring automation solutions specifically for Proxmox VE environments.

Design Stage

The design stage centered on developing a simulation for automating Proxmox-based SDN configurations using Ansible. This process involved designing IP address configurations to establish a structured and efficient virtual network environment. The automation workflow was developed using Ansible Playbooks to streamline tasks such as creating and deleting zones, VNets, subnets, and DHCP ranges [6], [16]. These automated configurations were designed to ensure scalability and reduce manual interventions, improving reliability.

Network Design

The network topology utilized in this study is designed to optimize the automation process for SDN configurations [17], [18], [19]. The setup consists of two primary components: an Ansible server and a Proxmox VE server. The Ansible server is connected to the internet via port ens33, with its IP address dynamically allocated through NAT and DHCP, resulting in a configuration of 192.168.169.1. Meanwhile, the Proxmox VE server establishes a static connection with the Ansible server through port vmbr0, using the static IP address 192.168.169.10. This direct and

structured connection between the two servers facilitates seamless communication, ensuring efficient execution of automated SDN configuration tasks. The network topology is depicted in Figure 1, showcasing the streamlined design that underpins the automation framework.

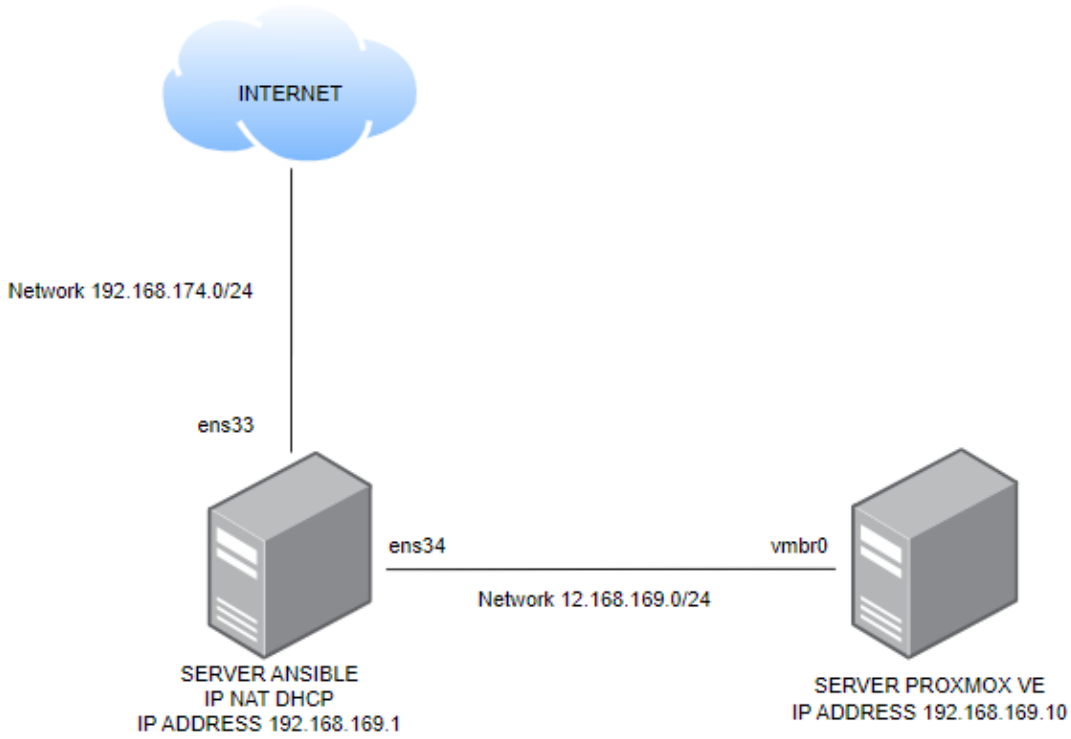

Figure 1. The Network Design

Test Network Design

The test network design, illustrated in Figure 2, was developed using virtualization tools to establish a controlled and flexible testing environment [20], [21], [22]. A Windows 11 host, connected directly to the internet, served as the base system for the setup. VMware Workstation version 16 Pro was utilized on this host to create two virtual machines. The first virtual machine functioned as an Ansible server, responsible for executing automation scripts. The second virtual machine operated as a Proxmox VE server, managing Software Defined Networking (SDN) configurations, including creating and deleting zones, VNets, subnets, and DHCP ranges. As emphasized in related studies, this virtualized setup provided a robust platform for testing the automation processes, ensuring reproducibility and adaptability to various network scenarios.

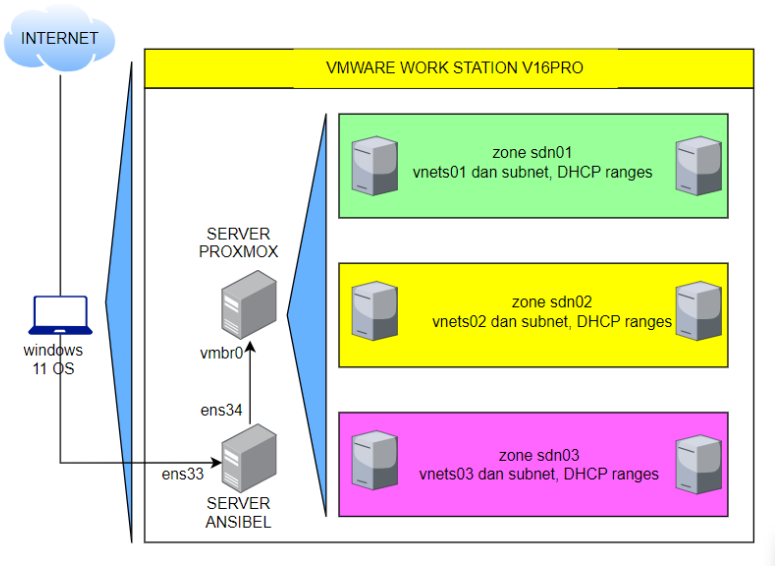

Figure 2. Test Network Design

IP Addressing Design

The IP addressing for this study uses Class C networks with five address blocks: 192.168.174.0/24, 192.168.169.0/24, 192.168.100.0/24, 192.168.101.0/24, and 192.168.102.0/24. Each network is configured to support specific SDN zones and DHCP ranges, as shown in Table 1.

Table 1. IP Addressing Design

Device	int	Description	IP Address	Gateway
Server ansible	ens 33	DHCP from NAT cloud	192.168.174.128/24	192.168.174.1
	Ens 34	*Host-only*	192.168.169.1	
Server Proxmox	Vmbr0	*Host-only*	192.168.169.10	
Zone sdn01	bridge	Zones created on a proxmox-based SDN	192.168.100.0/24	
Zone sdn02			192.168.101.0/24	
Zone sdn03			192.168.102.0/24	
100 *srv.ubg.local*		Which CT in *zone sdn01*		
			DHCP	192.168.100.1
101 *srv.ubg.local*				
102 *srv.ubg.local*		CT that is on *zone* sdn02		
			DHCP	192.168.101.1
103 *srv.ubg.local*				
104 *srv.ubg.local*		CT that is on *zone* sdn03		
			DHCP	192.168.102.1
105 *srv.ubg.local*				

Automation System Design

The automation system design focuses on developing an efficient workflow using Ansible to manage SDN features in Proxmox VE. The process begins with a network administrator creating and deploying Ansible Playbooks and YAML scripts to automate various SDN configurations. These scripts are designed to handle creating and deleting zones, VNets, subnets, DHCP ranges, and containers, ensuring that all tasks are executed consistently and without manual intervention. Once the automation scripts are deployed, the system undergoes rigorous testing to verify that all configurations function seamlessly, enabling reliable and streamlined network management. The complete automation workflow is illustrated in Figure 3, which outlines the sequence of tasks and their integration within the Proxmox VE environment.

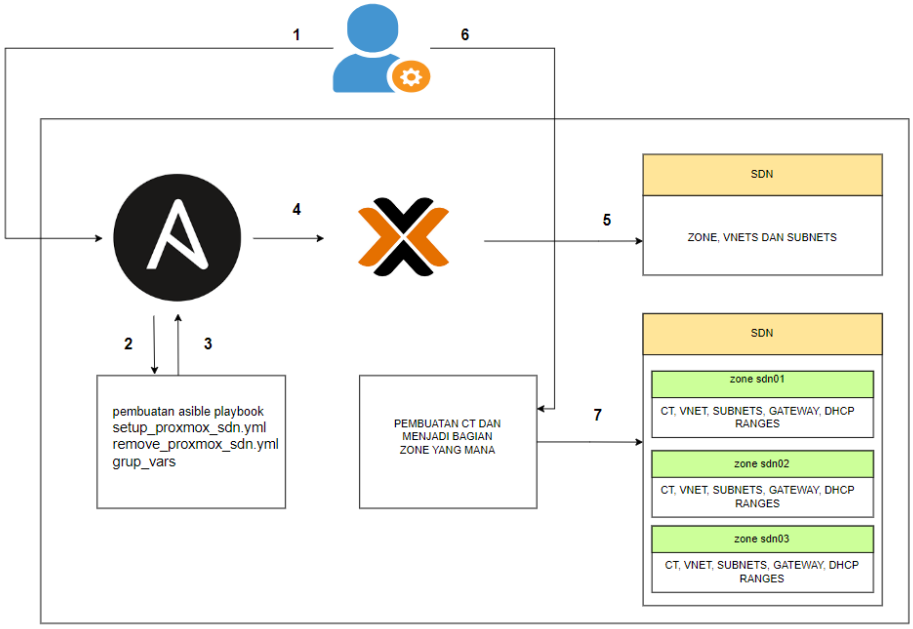

Figure 3. Automation System Flow Design

Prototyping Simulation Stage

In the prototyping simulation stage, a system simulation was developed to validate the automation process. The authors configured virtualization devices using Proxmox VE and Ansible servers, ensuring accurate system behavior before deployment. The setup included testing connectivity between containers (CTs) in the same and different zones and verifying IP address allocation in the IPAM menu.

Installation and Configuration Stage

The installation and configuration stage involved setting up all necessary components to enable automation and management of the SDN environment. First, the Proxmox VE server was installed and configured to handle the creation of zones, VNets, subnets, and DHCP ranges, serving as the primary virtualization platform. Next, the Ansible server was established with a collection of Playbooks and YAML scripts specifically designed to automate SDN management tasks, including creating and deleting network configurations. Finally, a Windows 11 client was configured to establish connections with the Proxmox VE and Ansible servers, enabling simulation and testing of the automated processes. This comprehensive setup ensured a functional and efficient environment for implementing and verifying the automation workflow.

Test Scenario

The test scenarios conducted in this study aimed to evaluate the efficiency and reliability of SDN configuration management in Proxmox VE. First, manual configuration testing was performed to measure the time required to create and delete three zones and VNets manually. This process provided a baseline for comparing the efficiency of manual configurations. Next, automated configuration testing was conducted using Ansible to automate the creation and deletion of zones and VNets. This step highlighted the time-saving advantages of automation compared to manual efforts. Finally, connectivity and IP allocation testing was carried out to verify the connectivity of containers (CTs) both within the same zone and across different zones. The tests also included validating internet access and ensuring accurate IP address allocation via the IPAM menu, reinforcing the reliability of automated configurations. These scenarios provided comprehensive insights into the benefits of automation in managing SDN configurations effectively.

RESULTS AND DISCUSSION

This section provides a detailed analysis of the results obtained from the manual and automated configuration tests, focusing on

time efficiency, configuration success rates, and connectivity validation. The findings are critically discussed to highlight the implications and relevance of automation in Software-Defined Network (SDN) management.

1. Manual Configuration Analysis

Table 2 summarizes the results of the manual configuration process in Proxmox-based SDN. This includes creating and deleting zones, VNets, subnets, gateways, and DHCP ranges for each zone.

Table 2. Manual Creation and Deletion Analysis Table

Zone	Vnet			Success	Time
	subnets	gateway	DHCPranges		
Creation					
Sdn01	✓	✓	✓	✓	6 minutes 6 seconds
Sdn02	✓	✓	✓	✓	
Sdn03	✓	✓	✓	✓	
Deletion					
Sdn01	✓	✓	✓	✓	2 minutes 37 seconds
Sdn02	✓	✓	✓	✓	
Sdn03	✓	✓	✓	✓	

The results indicate that the manual configuration process, although successful, is time-consuming. On average, manually creating configurations took 6 minutes and 6 seconds, while deletion required 2 and 37 seconds. These findings are consistent with Madamidola [23], who identified manual processes as prone to inefficiencies and human error, particularly in larger-scale networks.

The prolonged duration of manual configuration is a significant limitation, especially in dynamic environments where frequent updates and reconfigurations are necessary. As network complexity increases, manual processes can introduce operational delays and inconsistencies. This reinforces Ajiga et al. [24] 's argument, which emphasizes the need for automation to mitigate these challenges.

2. Automated Configuration Analysis

The automation results using Ansible for creating and deleting configurations are presented in Tables 3 and 4. The data highlights the efficiency of automation compared to manual processes

Table 3. Analysis of Automated Creation Results

zone	Creation			Success	Time
	vnets				
	subnets	gateway	DHCP ranges		
First Test					
Sdn01	✓	✓	✓	✓	23 seconds
Sdn02	✓	✓	✓	✓	
Sdn03	✓	✓	✓	✓	
Second Test					
Sdn01	✓	✓	✓	✓	20 seconds
Sdn02	✓	✓	✓	✓	
Sdn03	✓	✓	✓	✓	
Third Test					
Sdn01	✓	✓	✓	✓	21 seconds
Sdn02	✓	✓	✓	✓	
Sdn03	✓	✓	✓	✓	

Table 3 provides information on the configuration management process for creating zones, vnets, subnets, IP gateways, and DHCP ranges on an SDN based on Proxmox VE. During the automation process, the creation time was measured across three

tests. The automation for creating zones, vnets, subnets, IP gateways, and DHCP ranges was successful, with the first test taking 23 seconds, the second 20 seconds, and the third 21 seconds. The analysis of the automated deletion process is shown in Table 4.

Table 4. Analysis of Automated Deletion Results

Deletion					
zone	vnets			Success	Time
	subnets	gateway	DHCP ranges		
First Test					
Sdn01	✓	✓	✓	✓	
Sdn02	✓	✓	✓	✓	20 seconds
Sdn03	✓	✓	✓	✓	
Second Test					
Sdn01	✓	✓	✓	✓	
Sdn02	✓	✓	✓	✓	19 seconds
Sdn03	✓	✓	✓	✓	
Third Test					
Sdn01	✓	✓	✓	✓	
Sdn02	✓	✓	✓	✓	20 seconds
Sdn03	✓	✓	✓	✓	

Table 4 provides information on the configuration management process for deleting zones, vnets, subnets, IP gateways, and DHCP ranges on an SDN based on Proxmox VE. During the automation process, the deletion time was measured over three tests. The automated deletion of zones, vnets, subnets, IP gateways, and DHCP ranges was successful, with the first test taking 20 seconds, the second 19 seconds, and the third 20 seconds. At this stage, an analysis is conducted on creating CTs for each zone with two CTs. Connectivity between CTs within the same zone and between CTs in different zones is checked. Additionally, an analysis is performed to ensure that the CTs have internet access and the correct IP address allocation in the IPAM menu. These findings corroborate prior research by Chandrawaty and Hariyadi [25], who demonstrated similar time savings using Ansible for VLAN configuration management.

Moreover, the consistency observed in automation results eliminates the risks associated with human error. As El Rajab et al. [26] and Ramesh et al. [27] noted, automation ensures that configurations are applied uniformly across all network components, improving overall reliability and scalability.

3. Connectivity and IP Allocation Validation

Table 5 illustrates the results of connectivity tests between containers (CTs) within the same and different zones, along with IP allocation and internet access validation.

Table 5. CT Connectivity Analysis Results in Zone

CT ID	ZONE	Internet Access	CT Connectivity on Different Zones	CT Connectivity to Different Zones	IPAM IP Allocation
100	Sdn01	✓	✓	✓	✓
101		✓	✓	✓	✓
102	Sdn02	✓	✓	✓	✓
103		✓	✓	✓	✓
104	Sdn03	✓	✓	✓	✓
105		✓	✓	✓	✓

Table 5 presents information related to the results of CT connectivity checks that have been made where tests are carried out related to the success of each CT in the zone that has been created by conducting connectivity tests between CTs in the same zone, testing connectivity between CTs in different zones and testing each CT to access the internet so

that CT gets internet services and finally conducting trials related to IP allocation at IPAM so that it can be seen in the table that it was successfully carried out related to checking connectivity between CTs in the same zone where in this trial pinging was carried out between CT IDs 100 and 101 in zone sdn01 and carried out to all CTs in each zone, after that, a connectivity check between CTs in different zones was carried out on CT IDs 102 and 103 which pinged the CT IP Address in each zone and was successfully connected, CT trials got internet access by pinging google.com on each CT and successfully get internet access. Finally, the IPAM section will be tested to see where each CT ID is successfully in the IPAM section in each zone.

The connectivity and IP allocation tests demonstrate the effectiveness of Ansible automation in maintaining network integrity. All CTs successfully communicated within and across zones and accessed the internet without issues. Furthermore, IP allocation through the IPAM module was accurate and consistent.

These results align with findings from Aleem et al. [28], Kulkarni et al. [29], and Gupta [30], who highlighted the importance of automated IP management systems in maintaining seamless connectivity in SDN environments. The results also validate the robustness of Ansible as a tool for managing complex network configurations.

The manual and automated configuration results demonstrate the advantages of automation in SDN management. Automation not only reduces configuration times but also minimizes errors and enhances reliability. As Muhammad and Munir [31] highlight, automation is essential for achieving scalability and consistency in dynamic network environments.

Future research could explore the integration of advanced monitoring and analytics tools into the automation process. Additionally, incorporating Quality of Service (QoS) features and real-time performance metrics could further enhance automation's utility in SDN management.

LIMITATIONS

Complexity of SDN Infrastructure, Possible Limitations in SDN, Proxmox as a Limited Hypervisor, Dependency on Tool Versions, and Security and Management.

CONCLUSION

The design and implementation of SDN virtualization configuration management automation based on Proxmox VE using Ansible as an automation tool was successfully applied through an Ansible playbook. The playbook handled the configuration for creating and deleting zones, vnets, subnets, IP gateways, and DHCP ranges. The process involved creating and deleting three zones, three vnets, and subnets, along with their corresponding gateways and DHCP ranges for each vnet.

Based on three automated tests using Ansible, the automation process took 23 seconds for the first test, 20 seconds for the second, and 21 seconds for the third. Similarly, deletion tests were conducted three times, with the first test taking 20 seconds, the second 19 seconds, and the third 20 seconds. Additionally, manual testing for the creation process took 6 minutes and 6 seconds, while manual deletion took 2 minutes and 37 seconds. Therefore, automation using Ansible for SDN configuration management based on Proxmox VE is faster than manual configuration, making the automation process more efficient in terms of time, effort, and resources.

ACKNOWLEDGEMENTS

We extend our gratitude to all parties involved in this research for their support and contributions. Special thanks to the Department of Computer Science for their assistance with data collection and analysis.

AUTHORS CONTRIBUTION

HH was responsible for designing the network architecture and overseeing its implementation. They also coordinated the technical development of the project. **KM** provided supervision and offered guidance throughout the research process, reviewed the methodology and ensured alignment with the research objectives. **ZA** contributed by providing advice on system optimization and component integration, ensuring that the research outcomes were achieved.

REFERENCES

[1] T. Borangiu, D. Trentesaux, A. Thomas, P. Leitão, and J. Barata, "Digital transformation of manufacturing through cloud services and resource virtualization," *Computers in Industry*, vol. 108. Elsevier, pp. 150–162, 2019, doi: https://doi.org/10.1016/j.compind.2019.01.006.

[2] A. Benlian, W. J. Kettinger, A. Sunyaev, T. J. Winkler, and GUEST EDITORS, "Special Section: The Transformative Value of Cloud Computing: A Decoupling, Platformization, and Recombination Theoretical Framework," *J. Manag. Inf. Syst.*, vol. 35, no. 3, pp. 719–739, Jul. 2018, doi: https://doi.org/10.1080/07421222.2018.1481634.

[3] E. Casalicchio and S. Iannucci, "The state-of-the-art in container technologies: Application, orchestration and security," *Concurr. Comput. Pract. Exp.*, vol. 32, no. 17, p. e5668, Sep. 2020, doi: https://doi.org/10.1002/cpe.5668.

[4] M. Šimon, L. Huraj, and N. Búčik, "A Comparative Analysis of High Availability for Linux Container Infrastructures," *Future Internet*, vol. 15, no. 8, p. 253, 2023, doi: https://doi.org/10.3390/fi15080253.

[5] Π. Λιοταλ\acute\etaς, "Network Function Virtualization (NFV) technologies and their impact on energy consumption in large-scale data centers," Master's Thesis, Πανεπιοταστ\acute\etaμιοτao Πειοταραιοτα\acuteømegaς, 2024, doi: http://dx.doi.org/10.26267/unipi_dione/4129.

[6] M. D. Elradi, "Ansible: A Reliable Tool for Automation," *Electr. Comput. Eng. Stud.*, vol. 2, no. 1, 2023.

[7] N. Ikhsan, A. A. Sukmandhani, J. Ohliati, and Y. D. Prabowo, "Design and Build AAA Server using Free Radius Study Case Network Security Management at PT. XYZ," in *2023 IEEE 9th International Conference on Computing, Engineering and Design (ICCED)*, IEEE, 2023, pp. 1–6, doi: https://doi.org/10.1109/ICCED60214.2023.10425645.

[8] K. Marzuki, M. I. Kholid, I. P. Hariyadi, and L. Z. A. Mardedi, "Automation of Open VSwitch-Based Virtual Network Configuration Using Ansible on Proxmox Virtual Environment," *Int. J. Electron. Commun. Syst.*, vol. 3, no. 1, pp. 11–20, 2023, doi: https://doi.org/10.24042/ijecs.v3i1.16524.

[9] N. M. A. Y. Chandrawaty and I. P. Hariyadi, "Implementasi Ansible Playbook Untuk Mengotomatisasi Manajemen Konfigurasi VLAN Berbasis VTP Dan Layanan DHCP," *J. Bumigora Inf. Technol. BITe*, vol. 3, no. 2, pp. 107–122, 2021, doi: https://doi.org/10.30812/bite.v3i2.1577.

[10] K. Zheng, Q. Zheng, P. Chatzimisios, W. Xiang, and Y. Zhou, "Heterogeneous vehicular networking: A survey on architecture, challenges, and solutions," *IEEE Commun. Surv. Tutor.*, vol. 17, no. 4, pp. 2377–2396, 2015, doi: https://doi.org/10.1109/COMST.2015.2440103.

[11] S. Zhu *et al.*, "Proactive Telemetry in Large-Scale Multi-Tenant Cloud Overlay Networks," *IEEEACM Trans. Netw.*, 2024, doi: 10.1109/TNET.2024.3381786.

[12] F. Naim, R. R. Saedudin, and U. Y. K. S. Hediyanto, "Analysis of wireless and cable network quality-of-service performance at Telkom University landmark tower using network development life cycle (ndlc) method," *JIPI J. Ilm. Penelit. Dan Pembelajaran Inform.*, vol. 7, no. 4, pp. 1033–1044, 2022, doi: https://doi.org/10.29100/jipi.v7i4.3192.

[13] M. E. Arass, K. Ouazzani-Touhami, and N. Souissi, "Data life cycle: towards a reference architecture," *Int J*, vol. 9, 2020, doi: 10.30534/ijatcse/2020/215942020.

[14] N. Ahmed *et al.*, "Network threat detection using machine/deep learning in sdn-based platforms: a comprehensive analysis of state-of-the-art solutions, discussion, challenges, and future research direction," *Sensors*, vol. 22, no. 20, p. 7896, 2022, doi: https://doi.org/10.3390/s22207896.

[15] L. Zhu *et al.*, "SDN Controllers: A Comprehensive Analysis and Performance Evaluation Study," *ACM Comput. Surv.*, vol. 53, no. 6, pp. 1–40, Nov. 2021, doi: https://doi.org/10.1145/3421764.

[16] B. Choi and E. Medina, "Is Ansible Good for Network Automation?," in *Introduction to Ansible Network Automation*, Berkeley, CA: Apress, pp. 3-30, 2023, doi: https://doi.org/10.1007/978-1-4842-9624-0_1.

[17] R. Amin, E. Rojas, A. Aqdus, S. Ramzan, D. Casillas-Perez, and J. M. Arco, "A survey on machine learning techniques for routing optimization in SDN," *IEEE Access*, vol. 9, pp. 104582–104611, 2021, doi: https://doi.org/10.1109/ACCESS.2021.3099092.

[18] A. A. Ibrahim, F. Hashim, A. Sali, N. K. Noordin, and S. M. Fadul, "A multi-objective routing mechanism for energy management optimization in SDN multi-control architecture," *IEEE Access*, vol. 10, pp. 20312–20327, 2022, doi: https://doi.org/10.1109/ACCESS.2022.3149795.

[19] D. Bringhenti *et al.*, "Automatic, verifiable and optimized policy-based security enforcement for SDN-aware IoT networks," *Comput. Netw.*, vol. 213, p. 109123, 2022, doi: https://doi.org/10.1016/j.comnet.2022.109123.

[20] R. Wójtowicz, R. Kowalik, D. D. Rasolomampionona, and K. Kurek, "Virtualization of protection systems-tests performed on a large environment based on data center solutions," *IEEE Trans. Power Deliv.*, vol. 37, no. 4, pp. 3401–3411, 2021, doi: https://doi.org/10.1109/TPWRD.2021.3128993.

[21] I. Alam *et al.*, "A Survey of Network Virtualization Techniques for Internet of Things Using SDN and NFV," *ACM Comput. Surv.*, vol. 53, no. 2, pp. 1–40, Mar. 2021, doi: https://doi.org/10.1145/3379444.

[22] A. S. Zamani, A. S. A. Shatat, I. A. Khan, M. M. Akhtar, R. Ayub, and F. Samdani, "Cloud Network Design and Requirements for the Virtualization System for IoT Networks," *IJCSNS*, vol. 22, no. 11, p. 727, 2022, doi: 10.22937/IJCSNS.2022.22.11.101.

[23] O. A. Madamidola, O. A. Daramola, K. G. Akintola, and O. T. Adeboje, "A Review of existing inventory management systems," *Int. J. Res. Eng. Sci. IJRES*, vol. 12, no. 9, pp. 40–50, 2024.

[24] D. Ajiga, P. A. Okeleke, S. O. Folorunsho, and C. Ezeigweneme, "The role of software automation in improving industrial operations and efficiency," *Int. J. Eng. Res. Updat.*, vol. 7, no. 1, pp. 22–35, 2024, doi: https://doi.org/10.53430/ijeru.2024.7.1.0031.

[25] B. Choi and E. Medina, "Cisco Router and Switch Configuration with Ansible," in *Introduction to Ansible Network Automation*, Berkeley, CA: Apress, 2023, pp. 595–660, doi: https://doi.org/10.1007/978-1-4842-9624-0_12.

[26] M. El Rajab, L. Yang, and A. Shami, "Zero-touch networks: Towards next-generation network automation," *Comput. Netw.*, vol. 243, p. 110294, 2024, doi: https://doi.org/10.1016/j.comnet.2024.110294.

[27] G. Ramesh, J. Logeshwaran, and A. P. Kumar, "The Smart Network Management Automation Algorithm for Administration of Reliable 5G Communication Networks," *Wirel. Commun. Mob. Comput.*, vol. 2023, pp. 1–13, Apr. 2023, doi: https://doi.org/10.1155/2023/7626803.

[28] S. Aleem and S. Ahmed, "Unlocking Network Security and QoS: The Fusion of SDN, IoT, and Machine Learning: A Comprehensive Analysis," *Int J Sci Res Netw. Secur. Commun. Vol*, vol. 11, p. 6, 2023.

[29] M. Kulkarni, B. Goswami, J. Paulose, and L. Malakalapalli, "Unlocking the Power of Software-Defined Networking (SDN) in Revolutionizing Network Management," in *Advanced Cyber Security Techniques for Data, Blockchain, IoT, and Network Protection*, IGI Global Scientific Publishing, pp- 309-336, 2025, doi:

https://doi.org/10.4018/979-8-3693-9225-6.ch012.

[30] P. M. Gupta, "Software-Defined Networking (SDN): Revolutionizing Network Infrastructure for the Future," in *Software-Defined Network Frameworks*, CRC Press, pp 89-108, 2024.

[31] T. Muhammad and M. Munir, "Network Automation," *Eur. J. Technol.*, vol. 7, no. 2, pp. 23–42, 2023, doi: https://doi.org/10.47672/ejt.1547.