APPLICATION OF ANSIBLE CONFIGURATION MANAGEMENT TOOL IN NETWORK SYSTEM ADMINISTRATION

Van Hieu NGUYEN[1], Thai Bao TRAN[1], Chi Nguyen HO[1]
[1]Faculty of Information Technology, Vinh Long University of Technology Education
Contact author: [1]Email: hieunv@vlute.edu.vn; [1]Mobile: 0907969205

Received: 17/12/2025; Reviewed: 3/3/2026; Accepted: 24/3/2026

ABSTRACT

This study applies the Ansible configuration management tool to automate network system administration, addressing the limitations of manual configuration such as time consumption, inconsistency, and error-proneness. A simulated environment was built with Windows Server, Debian Linux, and Cisco Router to evaluate Ansible's automation capabilities through ad-hoc commands and playbooks. The experimental results demonstrate that Ansible significantly reduces configuration time, ensures system consistency, and minimizes operational risks compared to traditional manual methods. Furthermore, the study highlights the potential of integrating Ansible into DevOps workflows and CI/CD pipelines, contributing to the adoption of Infrastructure as Code in modern IT environments.

Keywords: *Ansible, Network Automation, Infrastructure as Code*

1. INTRODUCTION

In the context of the rapid development of information technology, the continuous increase in the number of devices, servers, and online services has made the management, configuration, and operation of network systems increasingly complex. Traditional manual configuration methods reveal many limitations, such as being time-consuming, lacking consistency, and being prone to errors, which lead to increased management costs and operational risks. To overcome these challenges, Configuration Automation has emerged as an inevitable trend, providing a solution to standardize processes and ensure the accuracy and synchronization of the system infrastructure.

The majority of current research and published literature on Ansible primarily focus on automation in pure Linux environments, while the problem of integrating automation in hybrid environments - encompassing various platforms and system versions - has not yet been fully explored. Stemming from this reality, this study proposes a solution using Ansible for the centralized management of cross-platform infrastructure, while demonstrating its effectiveness through configuration synchronization across three common environments today: Windows, Linux, and Cisco network devices. The research results not only contribute to solving technical issues related to compatibility and configuration consistency but also propose a digital transformation model suitable for the information technology infrastructure conditions at educational institutions.

This study focuses on applying the Ansible tool in the automation of cross-platform network administration, aiming to demonstrate Ansible's capability in minimizing errors, saving time, and ensuring consistency among systems. The core of the project is to exploit Ansible's Infrastructure as Code (IaC) philosophy, which allows configurations to be defined using YAML - a language with a clear structure that facilitates version control and reusability.

The scope of the research is implemented in a cross-platform simulated environment built on the VMware Workstation virtualization platform, including servers running Windows Server, Debian Linux, and simulated Cisco Router devices. The experimental process focuses on evaluating Ansible's automation capabilities through the execution of ad-hoc commands,

Playbooks, and Roles in typical scenarios. Three representative scenarios are deployed, including: (1) Installing and configuring DNS services on Windows Server; (2) Synchronizing and changing the Hostname across a cross-platform system comprising Linux, Windows, and Cisco Routers; and (3) Deploying a complete website from source code on a Linux server, which involves installing the Web environment (Apache, PHP, MySQL), importing initial data, and configuring internal access (HTTP) as well as Internet access (HTTPS) via the Ngrok platform.

Experimental results have proven the outstanding effectiveness of Ansible in automating the infrastructure deployment and management processes, helping to shorten configuration time, increase accuracy, and ensure consistency across the entire system. Simultaneously, centralized management through the Infrastructure as Code (IaC) model facilitates expansion, maintenance, and integration into modern workflows such as DevOps and CI/CD. In particular, the application of this model also strengthens the technical foundation for network administration at schools, contributing to the digital transformation process and the modernization of IT infrastructure in the education sector.

2. CONTENT

2.1. Overview of the Ansible Tool

2.1.1. Network Automation with Ansible

Network automation is the process of automating the configuration, management, testing, and operation of devices within a system to enhance availability, ensure consistency, increase control capabilities, and minimize human errors.

Ansible stands out in the field of network automation due to its application of the Infrastructure as Code (IaC) philosophy. IaC allows the definition, management, and provisioning of infrastructure (servers, networks, services) through source code files rather than manual configuration. This facilitates consistent system management, version control, and automated deployment [1].

2.1.2. Architecture and Core Components of Ansible

Ansible is an open-source configuration management tool operating on an Agentless architecture. This is a significant advantage as Ansible does not require the installation of client software (agents) on the target servers (Managed Nodes).

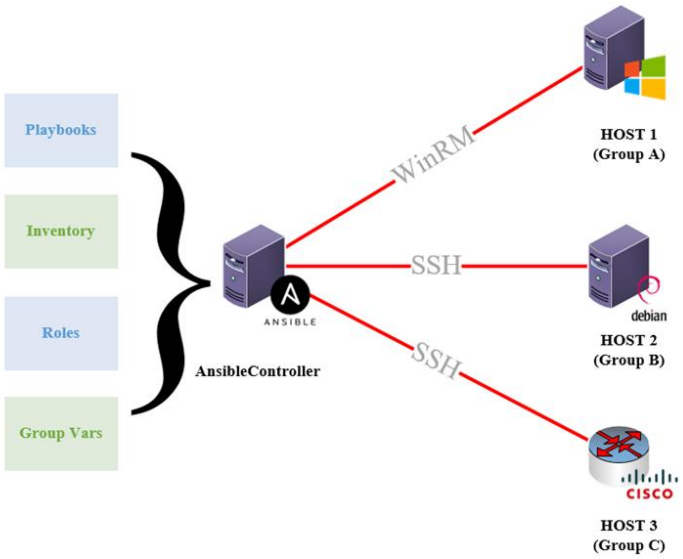

Figure 1: Ansible's Agentless Architecture

Ansible's operating mechanism is built upon a remote control and automation platform following the agentless model, allowing system management without installing intermediary software on the target machines. Specifically, the Ansible Controller uses standard connection protocols: SSH to communicate with Linux systems and network devices such as Cisco, and WinRM to communicate with Windows servers.

Automated processes are described through Playbooks, which are configuration files written in the YAML language, featuring a simple, readable, and easily maintainable syntax. The Ansible system is organized based on several core components, including: Inventory (a list of target servers – Managed Nodes), Ad-hoc Commands (quick execution commands for simple tasks), Playbooks (a collection of defined tasks to deploy complex automation scenarios via Modules), and Roles (a mechanism to organize Playbooks modularly, enhancing reusability and maintainability) [2].

An important feature in Ansible's operating mechanism is Idempotency (state invariance). This mechanism ensures that executing the same task multiple times will not cause changes if the desired state has already been achieved. Consequently, Ansible maintains the stability and consistency of the system while minimizing risks arising from duplicated or misaligned configurations [1].

2.1.3. Advantages of the Ansible Tool

Among the currently popular configuration management tools, Ansible was selected as the primary platform for this project due to its outstanding agentless architecture and deployment simplicity compared to other tools like Puppet and Chef. The analysis and comparison were conducted based on crucial technical criteria, including building language, operating architecture, connection protocols, and prominent features of each tool.

Table 1: Comparison of configuration management tools: Ansible, Puppet, and Chef [3]

Criteria	Ansible	Puppet	Chef
Building language	Python(core), YAML (playbook)	Custom DSL based on Ruby	Custom DSL based on Ruby
Approach	Agentless (no agent required)	Requires agent (Puppet Agent); Can be agentless (Puppet Bolt)	Requires agent (Chef Client)
Protocol	SSH	TCP (port 8140), SSL/TLS	TCP/HTTPS (port 443)
Architecture	Peer-to-Peer	Client-Server	Client-Server

Thanks to its agentless nature and clear, comprehensible YAML syntax, Ansible makes the deployment process flexible and minimizes configuration complexity. This is particularly suitable for multi-platform research environments that require rapid integration capabilities with existing network systems, thereby enhancing the feasibility and effectiveness of the project.

2.2. Design and Preparation of the Experimental Model

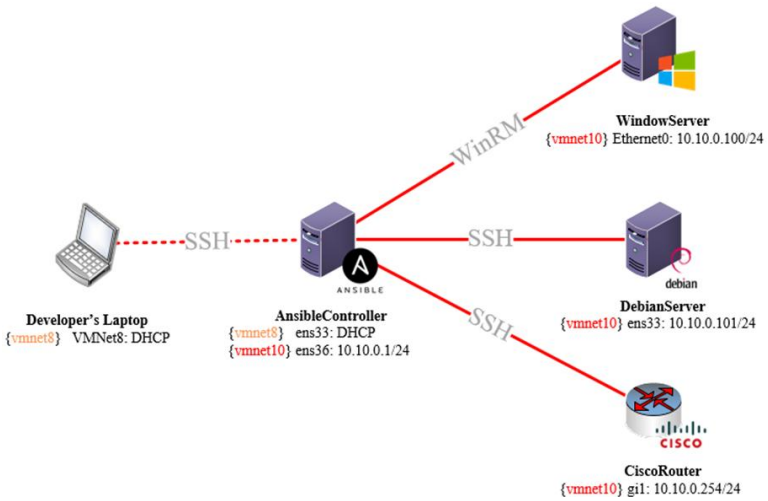

Figure 2: Deployment model of the configuration management system using Ansible

The experimental environment of the study was deployed on the VMware Workstation virtualization platform to simulate a multi-platform network system comprising various servers and network devices, thereby evaluating the flexibility and compatibility of the Ansible tool in practical system administration scenarios.

The system model is structured into two main component groups:

1. Ansible Controller:

Acts as the central control node, installed on the Debian Linux 12.5.0 operating system. This is where Playbook files are stored, Ad-hoc commands are executed, and all automation activities within the system are coordinated.

2. Managed Nodes (Hosts):

Includes target servers and network devices managed by Ansible:

- Windows Server: Installed with Windows Server 2022, connecting to the Controller via the WinRM protocol (port 5985).
- Debian Server: Installed with Debian Linux 12.5.0, communicating with the Controller via SSH.
- Cisco Router: Cisco CSRv1000 simulated network device, managed via the SSH protocol.

The environment preparation process includes the following steps: installing Ansible on the Controller machine, configuring corresponding connection protocols (SSH for Linux and Cisco, WinRM for Windows), and concurrently declaring the server list in the Inventory file. Sensitive information, such as access passwords, is stored in GroupVars and encrypted using Ansible Vault to ensure safety and security during operation.

2.3. Deployment of Automation Scenarios using Ansible

2.3.1. Executing Simple Tasks with Ansible Ad-hoc

In the initial phase of the experimental process, Ad-hoc commands were utilized to test connectivity and execute basic tasks between the Ansible Controller and the Managed Nodes. Specifically, the ping and *win_ping* modules were used to verify the connection status, and the results showed that all target servers responded successfully with a "SUCCESS" state, proving that the SSH and WinRM protocol setups were configured correctly.

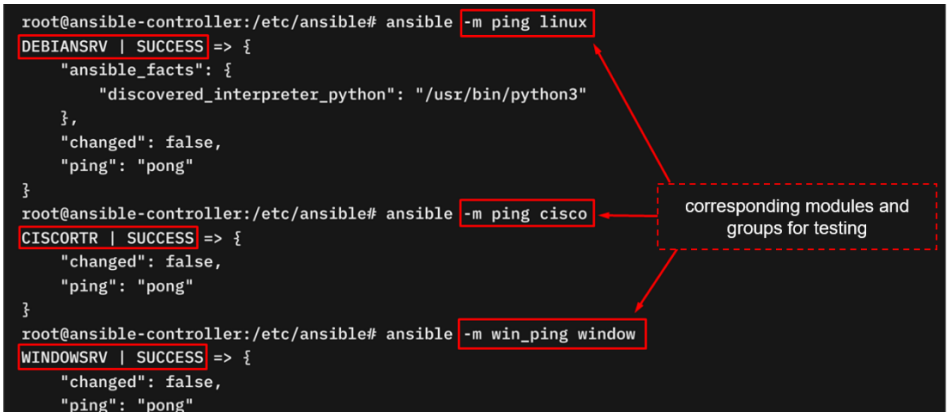

Figure 3: Connection test results using ad-hoc commands to Linux, Cisco, and Windows.

In addition to connection testing, several quick configuration tasks were performed to evaluate system stability, including: updating the time zone on the Windows Server and installing software packages on the Linux server. These tasks were deployed instantly through Ad-hoc commands without the need to build complete Playbooks, saving time and facilitating initial testing [4].

The experimental results also clearly illustrate the Idempotency mechanism – a crucial characteristic of Ansible's operation. When a software package already exists or a configuration has reached the desired state, Ansible automatically skips the redundant operation and returns a changed: false result, ensuring the system maintains stability and consistency without causing unnecessary changes.

2.3.2. Deploying Complex Configuration Scenarios (Playbooks and Roles)

Following the testing phase with Ad-hoc commands, the study continued to expand the evaluation scope by deploying complex configuration scenarios using Ansible Playbooks combined with Roles to organize source code modularly, increasing reusability and ease of maintenance. Three representative scenarios were executed to verify Ansible's multi-platform centralized management and automation capabilities as follows:

Scenario 1: Simultaneous Multi-Platform Hostname Change

This scenario aims to synchronize the hostname across the Managed Nodes on various operating systems, including Linux, Windows, and Cisco network devices, thereby evaluating Ansible's centralized management and multi-platform automation capabilities.

Three independent Roles were built corresponding to each system type – Debian Server, Windows Server, and Cisco Router. A comprehensive Playbook was designed to call and execute the appropriate Roles for each server group defined in the Inventory file. Through this mechanism, Ansible can automatically change the Hostname, synchronize information across the entire system, and restart devices when necessary to apply the new configuration [4].

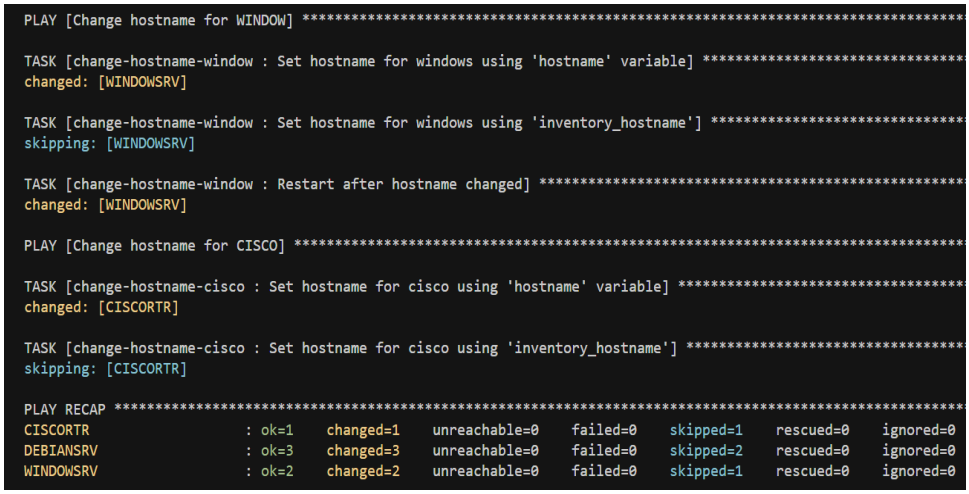

Figure 4: Playbook execution results in Scenario 1

The experimental results demonstrate that Ansible not only ensures centralized coordination and configuration consistency across diverse platforms but also effectively manages system-wide configuration values centrally, making them easy to update, maintain, and scale when the infrastructure changes. This approach contributes to enhancing the consistency, stability, and control of the system in real-world deployment environments.

Scenario 2: Deploying DNS Services on Windows Server

This scenario aims to automate the entire deployment and configuration process of a DNS Server service on the Windows Server 2022 operating system to evaluate Ansible's ability to manage and coordinate complex system services on the Windows platform.

The deployment process is executed through the dns-window Role, combined with configuration variables defined in GroupVars. The Playbook handles multiple sequential configuration steps, including: installing the DNS Server feature, creating the primary DNS zone (example.local), adding necessary DNS records (A, CNAME), and configuring the DNS client to point to the local server. Thanks to the Role mechanism and centralized variable management capabilities, Ansible enables the deployment process to be structured, easily maintainable, and highly reusable [4].

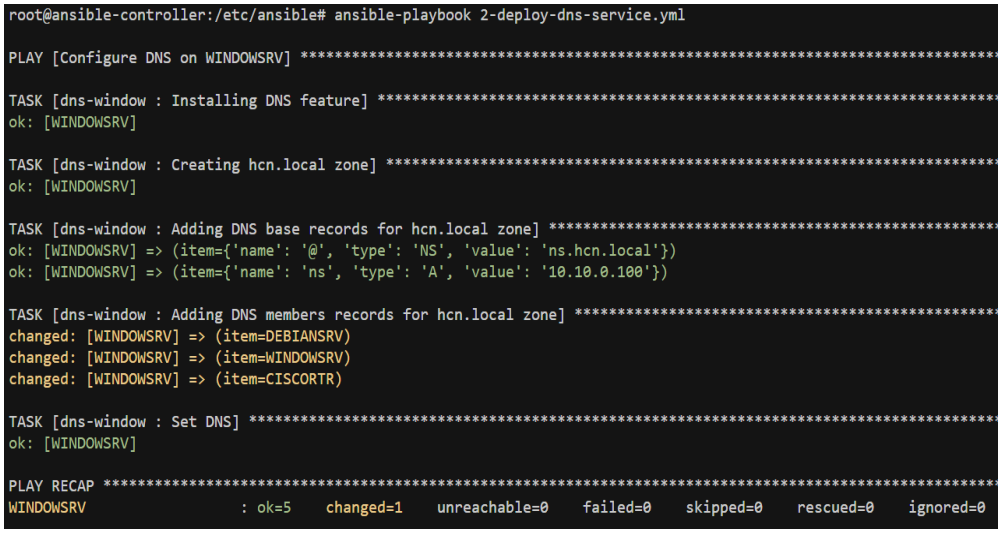

Figure 5: Playbook execution results in Scenario 2

The experimental results show that Ansible effectively automates the multi-step configuration process, ensuring accuracy, consistency, and high repeatability. Storing all configuration parameters in GroupVars facilitates centralized management of system values, making it convenient for future updates, expansions, and maintenance. This affirms the potential application of Ansible in managing complex Windows infrastructure, particularly in models geared toward comprehensive automation and system configuration standardization.

Scenario 3: Deploying a Complete Website from Source Code

This scenario aims to automate the deployment process of a complete website on a Linux server (Debian Server), including the installation of the Web Server, database, and website source code deployment. The system is configured to operate within the internal network (HTTP) using the domain name *www.example.local* (previously deployed in scenario 2) and can be exposed to the Internet (HTTPS) via the pre-registered Ngrok platform.

The process is carried out through automated step-by-step Ansible tasks: installing Apache2, PHP, MySQL, and related tools; deploying the website source code from a central directory to the local directory on the web server; setting up the database by creating the database, user, and importing initial data. Simultaneously, the Playbook configures the Virtual Host and internal DNS, allowing flexible changes to the website's domain name simply by adjusting variable values in GroupVars. Finally, Ansible installs and configures Ngrok to create an HTTPS route for Internet access.

```
TASK [ngrok-setup : Installing ngrok] ******************************************************************************
changed: [DEBIANSRV]

TASK [ngrok-setup : Installing ngrok-online.sh] ********************************************************************
changed: [DEBIANSRV]

TASK [ngrok-setup : Installing ngrok-online.service] ***************************************************************
changed: [DEBIANSRV]

TASK [ngrok-setup : Setup ngrok auth token] ************************************************************************
changed: [DEBIANSRV]

TASK [ngrok-setup : Setup ngrok auth token] ************************************************************************
changed: [DEBIANSRV]

TASK [ngrok-setup : Start ngrok-online service] ********************************************************************
changed: [DEBIANSRV]

TASK [ngrok-setup : Restart ngrok-online service] ******************************************************************
skipping: [DEBIANSRV]

TASK [Deploy infomation] *******************************************************************************************
ok: [DEBIANSRV] =>
  msg: |-
    Local website URL is    : http://www.hcn.local
    Published website URL is: https://adapted-gradually-monitor.ngrok-free.app

PLAY RECAP *********************************************************************************************************
DEBIANSRV                  : ok=34   changed=30   unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
```

Figure 6: Playbook execution results in Scenario 3

The scenario demonstrates that Ansible is capable of automating the deployment of complex Web applications consistently and with high reusability. The website is stably deployed over both HTTP and HTTPS, and its domain name can be easily scaled or altered. This result affirms Ansible's potential in managing and deploying real-world Web services, while simultaneously broadening the application of the Infrastructure as Code (IaC) model in corporate and educational environments.

2.4. Analysis and Evaluation of Experimental Results

2.4.1. Automation Efficiency and Consistency

Experimental results from the three main scenarios have clearly demonstrated the automation efficiency of Ansible in system configuration management. This tool allows the

execution of numerous complex tasks across multiple servers in a short amount of time, eliminating the need for manual operations on each individual device. Consequently, the system deployment and maintenance time is significantly reduced, enhancing work efficiency and alleviating the burden on administrators.

Table 2: Comparison of deployment times for experimental scenarios

Experiment	Deployment time (minutes)		Subjective error rate		Efficiency
	Manual	Ansible	Manual	Ansible	
Scenario 1	05 – 10	0,5 – 1	~15%	~2%	~9 times faster
Scenario 2	15 – 20	02 – 03	~20%	~2%	~7 times faster
Scenario 3	35 – 45	03 – 05	~30%	~2%	~9 times faster

Based on the statistical data in *Table 2*, there is a clear and significant improvement in performance when applying Ansible compared to the manual configuration method. Specifically, the total average time to complete all 3 scenarios was reduced from approximately 65 minutes to just under 9 minutes, corresponding to an overall efficiency increase of ~7 to 9 times depending on task complexity. Most notably, in Scenario 3, the operation time was shortened from 45 minutes to just about 5 minutes. Besides time optimization, the successful execution rate of Ansible consistently approaches 100% due to the Idempotency mechanism, completely eliminating the 15-30% subjective error rate typically encountered in manual command typing.

In terms of scale, the solution's advantage is evident in its system scalability. While the manual configuration method exhibits a near-linear increase in deployment time relative to the number of managed devices, applying Ansible leverages parallel execution mechanisms, thereby significantly reducing the rate of deployment time increase as the number of target servers grows. Consequently, the configuration synchronization process remains highly efficient in large-scale environments. This is particularly significant in the hybrid environment of the study, where manual configuration across Windows, Linux, and Cisco platforms often harbors potential errors due to syntax and management method discrepancies.

Transitioning to Ansible not only optimizes time but also ensures data consistency across the entire system thanks to its robust synchronization capabilities. Configurations are defined through Playbooks with consistent, readable, and controllable YAML syntax, ensuring that every Managed Node receives the exact configuration according to a single standard. This substantially eliminates risks caused by human error and prevents inconsistencies between servers, which frequently occur during manual deployments.

During the experimentation process, Ansible's Idempotency mechanism was also clearly demonstrated. When a software package or configuration already exists, Ansible automatically skips the installation step and responds with a *changed: false* status, proving that the system has achieved the desired state. This mechanism ensures stability, saves resources, and prevents redundant unnecessary changes, thereby maintaining reliability and repeatability in every deployment.

Furthermore, Ansible's multi-platform management capability is demonstrated through the successful deployment of distinct Roles for Linux, Windows, and Cisco network devices. From

a single control point, administrators can configure, monitor, and update the entire system in a centralized, consistent, and easily maintainable manner. These results indicate that Ansible not only excels in automation capabilities but also provides a flexible and modern infrastructure management method suitable for real-world large-scale deployment models.

2.4.2. Discussion on IaC/DevOps Application Potential

The application of Ansible in this study does not stop at technical efficiency but also carries strategic significance in realizing modern infrastructure management models. One of the prominent contributions is the promotion of the Infrastructure as Code (IaC) model. With this approach, all configurations and deployment processes are described using code that can be stored, controlled, and shared. Applying Ansible facilitates easy integration with version control systems like Git, allowing for the tracking of change histories, reverting configurations when necessary, and reusing configuration modules across multiple environments. Consequently, the network infrastructure is not only automated but also tightly managed, transparent, and easily scalable and upgradable.

Additionally, Ansible demonstrates strong potential for integration into CI/CD and DevOps workflows. The essence of Ansible is automation; therefore, this tool is perfectly suited to undertake deployment and configuration roles within the continuous delivery pipeline. When combined with tools like Jenkins or GitLab, Ansible can automate the entire process from building and testing to deployment, significantly shortening the software release cycle. Concurrently, maintaining consistent configurations across development, testing, and production environments contributes to minimizing errors arising from configuration discrepancies – one of the major challenges in traditional system operations.

Another crucial advantage is Ansible’s agentless architecture. Unlike many other configuration management tools that require the installation of monitoring software on each server, Ansible only needs to connect via SSH or WinRM to execute commands remotely. This approach reduces maintenance costs, simplifies deployment, and particularly accelerates system expansion in large-scale or multi-platform environments, such as Windows, Linux, or Cisco network devices. This very characteristic affirms Ansible’s potential as a core tool in realizing the automated and intelligent infrastructure model, aligning with the comprehensive DevOps trend in contemporary information technology management.

3. CONCLUSION

3.1. Contributions of the article

This study has successfully deployed and evaluated the application of Ansible – a configuration management automation tool in a multi-platform network environment. Through experimental scenarios, the project has proven Ansible's ability to overcome the limitations of manual configuration methods, while opening up a modern approach based on the Infrastructure as Code (IaC) model.

First of all, the research affirms the outstanding automation efficiency of Ansible in system administration. Configuration processes that are inherently time-consuming and prone to errors when performed manually are now fully automated through Playbooks and Roles with a clear structure. The simultaneous deployment of changes across multiple servers and network devices is executed quickly, accurately, and with high repeatability, helping to shorten operation time, reduce the workload for administrators, and limit risks caused by human error.

Furthermore, the project has demonstrated consistency, scalability, and high practical applicability in multi-platform environments. The centralized management of Linux systems, Windows Servers, and Cisco Routers from a single control point demonstrates Ansible's flexible

coordination capabilities. In particular, the complete website deployment scenario illustrates the ability to coordinate multiple service layers (Web, Database, DNS, HTTPS) in a unified automated process, meeting the requirements of practical application deployment in corporate or educational environments. The unified configuration system using YAML code along with a flexible variable mechanism helps the entire infrastructure maintain a synchronized state, making it easy to update and maintain without affecting overall operations.

More importantly, the research has consolidated the technical foundation supporting network administration for the educational institution where the authors work. The application of the automation model using Ansible not only improves the operational efficiency of the internal network system but also creates a premise for building a flexible, scalable infrastructure that can easily adapt to new service deployment requirements in the future.

Finally, the research results do not stop at the technical aspect but also propose a digital transformation model suitable for information technology infrastructure at educational institutions. Given the specific characteristics of limited resources, heterogeneous infrastructure, and stable operation requirements, the multi-platform centralized management model using Ansible is proven to be a low-cost, easy-to-deploy, and scalable solution. Describing configurations as code that can be version-controlled, shared, and reused makes Ansible a central tool in the CI/CD pipeline, while establishing a foundation for digital transformation and modernizing school network administration toward sustainability and comprehensive automation.

3.2. Recommendations and future developments

To continue promoting the effectiveness and expanding the practical applicability of the project, several research and development directions in the future are proposed as follows:

First, it is necessary to integrate the Dynamic Inventory mechanism to enhance the flexibility of the automation system. Connecting Ansible with dynamic data sources such as cloud services (AWS, Azure, Google Cloud), identity management systems (LDAP), or internal databases will facilitate the automatic collection and updating of server lists in real-time. Particularly, in the context where universities and enterprises are expanding their infrastructure to the Hybrid Cloud model, the potential of Dynamic Inventory lies in completely eliminating the bottleneck of manual static inventory updates. This mechanism allows the system to automatically recognize, classify device groups (Windows, Linux, Cisco), and apply corresponding configuration policies immediately when a new device is added to the network, ensuring seamless scalability.

In addition, building a complete CI/CD pipeline is an inevitable direction for Ansible to maximize its role in the DevOps ecosystem. Integration with tools such as Jenkins, Docker, or Terraform will allow for the automation of the entire deployment lifecycle – from coding, testing, and packaging to operation. This not only accelerates the software release process but also ensures consistency and high resilience in the production environment.

Another important development direction is upgrading the management interface using Ansible AWX. Deploying AWX will provide an intuitive interface, helping administrators monitor execution status, track task history, and decentralize user permissions more easily. This is a necessary step to expand the application capabilities of Ansible in corporate environments or educational organizations with multiple user levels.

Finally, the project can be expanded into actual deployment environments or public cloud platforms, thereby evaluating the stability, scalability, and performance of Ansible in real-world operational scenarios. Testing on a larger scale will provide more empirical data, serving as a

basis to perfect the model and propose more optimal solutions for system administration in the future.

Overall, these development directions not only contribute to expanding the scope of the research but also aim to build an automated, flexible, and adaptive network infrastructure management platform, aligning with the digital transformation trend in modern educational units and technological organizations.

References:

[1] Phan Van Phung, "Khóa luận tốt nghiệp, Đại học Duy Tân, Đà Nẵng," *Tìm hiểu và triển khai hệ thống quản lý cấu hình sử dụng Ansible,* 2022.

[2] Nguyen L. H. B., Nguyen T. D., "Proceedings of the 2022 RIVF International Conference on Computing and Communication Technologies," *Deploying DevOps Practices in Computer Lab Management at Universities,* 2022.

[3] Rahman S., Raza M. N., "International Conference on Information and Communication Technology for Sustainable Development (ICICT4SD)," *Performance Analysis of Configuration Management Tools: Ansible vs. Puppet vs. Chef,* pp. 150-155, 2021.

[4] Red Hat, "Ansible documentation," 2025. [Online]. Available: https://docs.ansible.com/ansible/latest/. [Accessed 08 03 2025].

[5] Freeman J., Keating J., "Packt Publishing," *Mastering Ansible: Automate configuration management and infrastructure (3rd ed.),* 2019.