UNIVERSIDAD TÉCNICA DE MANABÍ
Fundada en 1952

INFORMÁTICA Y SISTEMAS
Revista de Tecnologías de la Informática y las Comunicaciones

# Diseño de un sistema automatizado para la configuración y monitoreo de dispositivos de red usando python y ansible en entornos virtuales

# Design of an automated ystem for configuring and monitoring network devices using python and ansible in virtual environments

**Autores**

***Marlon Steven Sisalima Ulloa****
msisalima3@utmachala.edu.ec

***Leyton Lenin Veintimilla Guzmán***
lveintimi1@utmachala.edu.ec

***Oscar Efrén Cárdenas Villavicencio***
oecardenas@utmachala.edu.ec

***Rodrigo Fernando Morocho Román***
rmorocho@utmachala.edu.ec

Universidad Técnica de Machala, Facultad de Ingeniería Civil, Carrera de Tecnologías de la Información, Machala, El Oro, Ecuador.
*Autor para correspondencia

**Comó citar el artículo:**
Sisalima Ulloa, M.S., Veintimilla Guzmán, L.L., Cárdenas Villavicencio, O.E. & Morocho Román, R.F. (2026). Diseño de un Sistema Automatizado para la Configuración y Monitoreo de Dispositivos de Red Usando Python y Ansible en Entornos Virtuales. Informática y Sistemas, 10(2), 197–211. https://doi.org/10.33936/isrtic.v10i2.8666

Enviado: 24/08/2026
Aceptado: 10/11/2026
Publicado: 11/11/2026

**Resumen**

La administración manual de redes mediante interfaces de línea de comandos genera errores de configuración, tiempos de inactividad y una carga operativa considerable, particularmente en organizaciones con recursos limitados para adoptar soluciones comerciales de automatización. Este estudio desarrolló y evaluó, en un entorno de laboratorio virtualizado mediante EVE-NG, un prototipo de prueba de concepto basado en Python, Ansible y FastAPI para la configuración y el monitoreo de dispositivos de red. El prototipo integra playbooks de Ansible para el aprovisionamiento de redundancia VRRP y un módulo de monitoreo basado en SNMP y Syslog. Los procedimientos manual y automatizado se compararon mediante veinte repeticiones por método, tanto en configuración como en consulta de estado operativo, sobre una topología de dos switches de capa 3. Los resultados evidenciaron una reducción del 84.13% en el tiempo de aprovisionamiento y del 71.88% en el tiempo de obtención de estado operativo, una tasa de éxito del 100% frente al 90% manual, y concordancia dentro del margen de tolerancia definido en los datos vía SNMP. Estos hallazgos confirman la viabilidad técnica del prototipo en la topología de dos switches de capa 3 evaluada, dentro de un entorno controlado de laboratorio, con potencial de aplicación en contextos educativos; su escalabilidad a un mayor número de dispositivos y su eventual adopción en producción quedan planteadas como trabajo futuro, sujetas a la incorporación de mecanismos de seguridad actualmente fuera del alcance de este prototipo.

**Palabras clave:** automatización de redes; Ansible; SNMP; Infraestructura como Código; monitoreo de red.

**Abstract**

Manual administration of networks through command-line interfaces generates configuration errors, downtime, and a considerable operational burden, particularly for organizations with limited resources to adopt commercial automation solutions. This study developed and evaluated, in a virtualized laboratory environment using EVE-NG, a proof-of-concept prototype based on Python, Ansible, and FastAPI for network device configuration and monitoring. The prototype integrates Ansible playbooks for VRRP redundancy provisioning and a monitoring module based on SNMP and Syslog. Manual and automated procedures were compared through twenty repetitions per method, both for configuration and operational status queries, over a two-layer-3-switch topology. Results showed an 84.13% reduction in provisioning time and a 71.88% reduction in operational status retrieval time, a 100% success rate compared to 90% manual, and agreement within the defined tolerance margin in the SNMP data. These findings confirm the technical viability of the prototype in the evaluated two-layer-3-switch topology, within a controlled laboratory setting, with potential application in educational contexts; its scalability to a larger number of devices and eventual production adoption remain as future work, subject to incorporating security mechanisms currently outside the scope of this prototype.

**Keywords**: network automation; Ansible; SNMP; Infrastructure as Code; network monitoring.

CC BY NC ND

INFORMÁTICA Y SISTEMAS
Revista de Tecnologías de la Informática y las Comunicaciones

Facultad de Ciencias Informáticas
Universidad Técnica de Manabí
Av. Urbina y Che Guevara, Portoviejo, Ecuador
revista.iys@utm.edu.ec

## 1. Introducción

La administración manual de redes mediante interfaces de línea de comandos (CLI) genera, en promedio, que el 95% de las tareas operativas se ejecuten sin ningún grado de automatización, derivando en errores de configuración, tiempos de inactividad y una carga administrativa que compromete la continuidad operativa (El-Mokadem et al., 2026). Esta problemática afecta tanto a redes empresariales como a entornos educativos y organizaciones de menor escala, donde la adopción de soluciones comerciales de automatización suele estar fuera de alcance presupuestario. La transformación digital ha intensificado estas demandas: redes heterogéneas de múltiples fabricantes, entornos híbridos y la adopción masiva de dispositivos IoT han convertido la gestión manual en una práctica insostenible (Suresh Reddy Thati, 2025). Frente a este escenario, la literatura ha documentado avances en automatización de redes mediante paradigmas como NetDevOps (Network Development and Operations) e Infraestructura como Código (IaC); no obstante, persiste una brecha técnica relacionada con la integración de mecanismos de automatización de configuraciones, monitoreo operacional y validación previa al despliegue dentro de plataformas de código abierto orientadas a entornos virtualizados de experimentación.

Diversos estudios han señalado que la automatización de redes constituye una estrategia efectiva para reducir tareas repetitivas, minimizar errores operativos y mejorar la eficiencia de la administración de infraestructuras modernas (Muhammad & Munir, 2023). Asimismo, evaluaciones recientes de herramientas de automatización han evidenciado mejoras significativas tanto en el rendimiento operativo como en la experiencia de gestión de los administradores de red (Bruschetti et al., 2023).

Si bien existen plataformas consolidadas para la administración de redes, estas presentan limitaciones en contextos de pequeña y mediana escala. Herramientas como AWX y Nornir ofrecen potentes capacidades de orquestación, pero pueden implicar mayores requerimientos de infraestructura y administración según el contexto de implementación. Por otro lado, soluciones como NetBox (orientada a actuar como fuente de verdad) o Zabbix (enfocada en el monitoreo integral) pueden implicar mayores requerimientos de infraestructura y recursos para determinados escenarios de pequeña escala. Frente a estas alternativas, el presente estudio propone un prototipo unificado y ligero mediante la integración de automatización programática con Python y Ansible, acoplado a un motor de telemetría directo (SNMP/Syslog) a través de FastAPI. Este enfoque integra en una única interfaz las funciones de configuración, monitoreo y gestión de eventos evaluadas en este estudio, sin requerir licenciamiento comercial. Cabe precisar que la validación experimental del presente estudio se limitó a un único fabricante de dispositivos, por lo que la compatibilidad multivendor descrita responde al diseño arquitectónico del sistema (uso de protocolos estándar como SSH, SNMP y Syslog) y no ha sido verificada empíricamente frente a equipos de distintos proveedores.

Esta carencia plantea la siguiente pregunta de investigación: ¿En qué medida la implementación de un sistema automatizado basado en Python y Ansible mejora la eficiencia operativa en la configuración y monitoreo de dispositivos de red emulados en un entorno virtualizado, en comparación con la administración manual tradicional? En respuesta, el objetivo general es desarrollar un sistema automatizado basado en Python y Ansible orientado a optimizar la eficiencia operativa en tareas de configuración y monitoreo de dispositivos de red.

Para alcanzar este propósito, el estudio se desarrolló a partir de tres objetivos específicos. En primer lugar, fundamentar teóricamente el paradigma de Infraestructura como Código (IaC), el uso de herramientas de orquestación como Ansible y APIs programables como FastAPI mediante una revisión bibliográfica de fuentes de alto impacto, para sustentar el modelo integrado de automatización y monitoreo de red. En segundo lugar, desarrollar un prototipo funcional en un entorno controlado mediante EVE-NG, integrando playbooks de Ansible para el aprovisionamiento de dispositivos y scripts en Python para la recolección de métricas operativas vía SNMP, con visualización mediante una API desarrollada en FastAPI. Finalmente, evaluar la eficiencia operativa del modelo propuesto mediante pruebas comparativas de rendimiento frente a métodos tradicionales de gestión manual, en una topología jerárquica de red de tres capas.

La propuesta fue validada mediante un enfoque experimental-cuantitativo desarrollado sobre una infraestructura virtualizada en EVE-NG, comparando el tiempo requerido para la ejecución de configuraciones manuales y automatizadas, así como la correcta implementación de los parámetros de redundancia y conectividad definidos para el escenario experimental.

### 1.1 Estado del arte

El paradigma NetDevOps y el enfoque de Infraestructura como Código (IaC) proponen tratar la configuración de dispositivos de red como código fuente: versionable, reproducible y auditable (Dsilva et al., 2025). En este marco, Ansible se ha consolidado como herramienta de orquestación de referencia gracias a su arquitectura sin agentes y compatibilidad multivendor (Datta et al., 2023; Khumaidi, 2021; Marzuki et al., 2023; Mohammed Daffalla, 2023), mientras que Python constituye la interfaz programática principal del ecosistema mediante librerías especializadas como Netmiko y NAPALM (Mazin et al., 2021). La eficacia de esta



CC BY NC ND

e-ISNN 2550-6730 | Vol. 10, Núm. 2 (197-211): Julio-Diciembre, 2026 | DOI: 10.33936/isrtic.v10i2.8666

combinación ha sido cuantificada consistentemente: Prasetyo et al. (2024) reportaron una reducción del tiempo de despliegue de 50 a 6 minutos con Ansible; Abdellatif et al. (2026) demostraron que la automatización de topologías en entornos emulados es 4,5 veces más rápida y las pruebas 11 veces más eficientes; y El-Mokadem et al. (2026) evidenciaron reducciones de hasta el 67% en tiempos de ejecución con técnicas de concurrencia en Python, manteniendo una precisión del 100%. Avances como Ansible Lightspeed extienden este ecosistema incorporando generación de playbooks mediante lenguaje natural (Sahoo et al., 2024), mientras que la aplicabilidad en protocolos específicos como EIGRP y escenarios de hardening multivendor ha sido igualmente validada (Imoukhuede et al., 2025; Mohd Fuzi et al., 2021). En el contexto local, trabajos publicados en Informática y Sistemas como el de Ponce Loor y Herrera Tapia (2023) han empleado igualmente entornos de simulación de redes para evaluar comparativamente el desempeño de configuraciones de red bajo distintos escenarios, lo cual refuerza la pertinencia de la validación experimental en infraestructuras virtualizadas como enfoque metodológico en este campo.

Sin embargo, las soluciones existentes presentan limitaciones estructurales en tres dimensiones. En cuanto a la calidad del código, el 49,3% de los proyectos de automatización en Ansible incumple principios de idempotencia y modularidad, comprometiendo su mantenibilidad y escalabilidad (Dalla Palma et al., 2020; Kokuryo et al., 2020; Kumara et al., 2021). Respecto a la observabilidad, los sistemas basados en SNMP dependen de software propietario de alto costo (Kim et al., 2016), la telemetría programable profunda resulta inviable en equipos legados (Yaseen, 2025) y los enfoques multiagente introducen sobrecargas severas de procesamiento (Espinel Villalobos et al., 2021). Finalmente, en materia de validación, plataformas como ConfigHub (Han et al., 2024) omiten mecanismos de verificación previa al despliegue, y trabajos como el de Marzuki et al. (2024) resuelven la orquestación de redes virtuales sin integrar monitoreo en tiempo real ni auditoría de configuraciones. La tabla 1 sintetiza los trabajos más representativos, sus enfoques y su relación con la propuesta del presente estudio. Es importante precisar que existen soluciones consolidadas (como LibreNMS, Zabbix integrado con Ansible, o NetBox como fuente de verdad con automatización mediante plugins) que también combinan automatización, monitoreo y validación en distintos grados. La contribución del presente estudio no radica en integrar por primera vez estas funciones, sino en ofrecer un prototipo ligero, unificado y de código abierto, orientado específicamente a la validación previa al despliegue en entornos de laboratorio y educativos, con un alcance funcional y unos requerimientos de infraestructura deliberadamente acotados respecto de esas plataformas.

El análisis comparativo de la Tabla 1 es consistente con lo señalado previamente: soluciones como LibreNMS, Zabbix o NetBox ya combinan, en distintos grados, automatización, monitoreo y validación. La contribución del presente estudio no es la combinación en sí, sino su implementación como un prototipo ligero, unificado y de código abierto, orientado a la validación previa al despliegue mediante revisión humana en entornos de laboratorio y educativos, con un alcance funcional y una arquitectura deliberadamente acotados respecto de plataformas integrales de gestión de redes.

Con base en lo anterior, el presente estudio propone y evalúa un prototipo de prueba de concepto que integra estas tres dimensiones en una solución ligera y unificada. La sección siguiente describe

**Tabla 1.** Trabajos relacionados y aporte con la propuesta.
Fuente: Los autores.

| Autor / Año | Tema | Metodología | Resultado principal | Relación con el estudio |
|---|---|---|---|---|
| (Mazin et al., 2021) | Automatización con Python en Cisco | Experimental / Comparativa | Reducción de tiempo y eliminación de errores humanos | Fundamenta el uso de Python como capa de automatización |
| (Abdellatif et al., 2026) | Automatización en redes IP densas | Experimental / Simulación (EVE-NG) | Automatización 4,5× más rápida; pruebas 11× más eficientes | Sustenta el uso de EVE-NG para validación en entornos virtualizados |
| (Espinel Villalobos et al., 2021) | Monitoreo de redes empresariales | Arquitectura multiagente con SNMP | Mejora en monitoreo; sobrecarga severa en equipos legados | Justifica un enfoque de telemetría ligera, sin sobrecarga |
| (Kumara et al., 2021) | Buenas prácticas en IaC | Revisión sistemática de literatura gris | Modularidad e idempotencia como pilares en DevOps | Orienta el diseño de playbooks modulares y reutilizables |
| (Dsilva et al., 2025) | Automatización NetDevOps | Conceptual / Descriptivo | Integración de automatización y despliegue continuo CI/CD | Fundamenta el enfoque NetDevOps del sistema |
| (Imoukhuede et al., 2025) | Hardening automatizado multivendor | Experimental / Simulación | Reducción de tiempo y errores en entornos heterogéneos | Respalda el uso de scripts automáticos en topologías heterogéneas |
| (Kokuryo et al., 2020) | Módulos imperativos en Ansible | Estudio empírico cuantitativo | 49,3% de proyectos incumple idempotencia y buenas prácticas | Justifica el diseño idempotente de los playbooks |
| (Sahoo et al., 2024) | Generación automática de código Ansible | Evolución tecnológica con IA | Generación de playbooks por lenguaje natural; menor barrera técnica | Sitúa el sistema frente a tendencias emergentes de IA aplicada a redes |


e-ISNN 2550-6730 | Vol. 10, Núm. 2 (197-211): Julio-Diciembre, 2026 | DOI: 10.33936/isrtic.v10i2.8666

INFORMÁTICA Y SISTEMAS
Revista de Tecnologías de la Informática y las Comunicaciones

Facultad de Ciencias Informáticas
Universidad Técnica de Manabí
Av. Urbina y Che Guevara, Portoviejo, Ecuador
revista.iys@utm.edu.ec

el enfoque adoptado para su desarrollo y evaluación.

## 2. Materiales y Métodos

### 2.1 Diseño metodológico

La presente investigación se desarrolló bajo un enfoque cuantitativo de carácter aplicado, debido a que estuvo orientada al desarrollo de una solución tecnológica destinada a optimizar procesos de configuración y monitoreo en infraestructuras de red. El análisis se fundamentó en la obtención de métricas objetivas relacionadas con el desempeño operativo del sistema propuesto, permitiendo comparar procedimientos de administración manual y automatizada sobre la misma infraestructura y bajo condiciones operativas controladas. Debido a que las repeticiones se ejecutaron secuencialmente por bloques, primero el procedimiento manual y posteriormente el automatizado, esta comparación debe interpretarse considerando la posible influencia del orden de ejecución y la familiarización de los operadores con la topología.

El diseño de investigación fue experimental, puesto que se manipuló la variable independiente correspondiente al método de administración de red empleado. Para ello se definieron dos escenarios de evaluación: un escenario basado en la configuración manual mediante CLI y un escenario basado en la automatización mediante el sistema desarrollado utilizando Python y Ansible.

La unidad experimental estuvo constituida por el proceso de generación, validación y despliegue de configuraciones de red requeridas para implementar mecanismos de redundancia sobre una infraestructura virtualizada. Como caso de estudio se utilizó la configuración del protocolo de Redundancia de Router Virtual (VRRP) entre dispositivos de capa 3, debido a que este procedimiento involucra múltiples parámetros de direccionamiento, redundancia y consistencia operativa.

La variable independiente correspondió al método de administración utilizado durante las pruebas experimentales. Como variables dependientes se consideraron el tiempo requerido para la ejecución de las configuraciones, la correcta aplicación de los parámetros definidos y la disponibilidad de información de monitoreo obtenida mediante los mecanismos implementados en el sistema.

El desarrollo del sistema se alineó con los principios del paradigma NetDevOps, el cual integra prácticas de desarrollo ágil, operaciones de red y automatización con el propósito de habilitar infraestructuras más confiables, escalables y gestionables. En este marco, las configuraciones de red fueron tratadas como código reutilizable y versionable bajo el enfoque de IaC, mientras que la automatización mediante Python y Ansible, el monitoreo centralizado con SNMP y PySNMP, y la virtualización del entorno en EVE-NG constituyeron las prácticas NetDevOps aplicadas de forma transversal durante todo el proyecto.

### 2.2 Entorno experimental y recursos utilizados

Los experimentos fueron desarrollados en un entorno de laboratorio virtualizado implementado mediante la plataforma EVE-NG Community Edition, utilizada como infraestructura de emulación de redes para la validación de las funcionalidades de automatización y monitoreo propuestas. La elección de esta plataforma permitió recrear escenarios de red de mediana complejidad, representativos de un contexto de laboratorio o educativo, de forma controlada, facilitando la ejecución repetitiva de pruebas sin necesidad de infraestructura física dedicada.

El sistema de automatización fue desplegado sobre un servidor con sistema operativo Ubuntu Server 24.04.3 LTS, empleando Python 3.12.3 como lenguaje principal para el desarrollo de la aplicación y Ansible Core 2.18.10 como motor de automatización. La interfaz web fue desarrollada con HTML, CSS y JavaScript, y servida mediante el framework FastAPI. Las tareas de monitoreo combinaron múltiples mecanismos según el tipo de dato: el protocolo SNMP para la obtención de identificación del dispositivo, tiempo de actividad, uso de CPU y tráfico de interfaces; playbooks de Ansible para la consulta del estado y configuración de interfaces de red; y un servidor Syslog embebido, basado en sockets UDP, para la captura pasiva de eventos generados por los dispositivos en tiempo real.

La selección de estas herramientas se fundamentó en criterios de compatibilidad multivendor, ausencia de agentes en dispositivos gestionados y respaldo en la literatura especializada. Ansible fue preferido frente a alternativas como Puppet, Chef o SaltStack debido a su arquitectura agentless, que evita instalar software adicional en los dispositivos de red administrados, y a su amplia adopción documentada en automatización de infraestructuras multivendor (Khumaidi, 2021; Mohammed Daffalla, 2023). Python se eligió como lenguaje de integración por su ecosistema maduro de librerías especializadas en redes, como Netmiko, NAPALM y PySNMP, frente a alternativas como Go o PowerShell, que ofrecen menor soporte nativo para automatización de dispositivos de red multivendor (Mazin et al., 2021). Para el monitoreo se optó por SNMP en lugar de protocolos de telemetría programable (por ejemplo, gNMI o streaming telemetry), debido a su amplia compatibilidad con dispositivos que implementan SNMP, incluidos equipos legados, característica pertinente para la topología experimental. Finalmente, FastAPI fue seleccionado frente a frameworks como Flask o Django por su desempeño asíncrono nativo y su menor



CC BY NC ND

e-ISNN 2550-6730 | Vol. 10, Núm. 2 (197-211): Julio-Diciembre, 2026 | DOI: 10.33936/isrtic.v10i2.8666

sobrecarga, características relevantes para una interfaz que debía consultar métricas operacionales en tiempo real sin introducir latencia adicional. La Tabla 2 resume la justificación de cada herramienta seleccionada.

**Table 2.** Justificación de herramientas y tecnologías seleccionadas. Fuente: Los autores.

| Herramientas | Alternativas consideradas | Justificación de selección |
|---|---|---|
| Ansible | Puppet, Chef, SaltStack | Arquitectura agentless; no requiere instalar software en los dispositivos gestionados; capacidad de gestionar dispositivos mediante módulos y protocolos compatibles. |
| Python | Go, PowerShell | Ecosistema maduro de librerías especializadas para automatización y monitoreo de redes, incluyendo Netmiko, NAPALM y PySNMP. |
| SNMP | gNMI, streaming telemetry | Compatibilidad con dispositivos que implementan SNMP, incluidos equipos legados; esta característica fue pertinente para la topología experimental. |
| FastAPI | Flask, Django | Desempeño asíncrono nativo; menor sobrecarga para consultas de monitoreo en tiempo real. |
| EVE-NG | GNS3, Containerlab | Soporte para imágenes reales de fabricantes y mayor fidelidad de emulación frente a simuladores puramente conceptuales. |

La topología experimental estuvo compuesta por nueve elementos principales: un router, dos switches multicapa (capa 3), dos switches de acceso (capa 2), cuatro equipos terminales virtuales (Virtual PC Simulator, VPCS) y una interfaz Cloud utilizada para proporcionar conectividad entre el entorno virtualizado y el equipo físico anfitrión. La arquitectura observable en la Figura 1 permitió evaluar simultáneamente funciones de enrutamiento, conmutación, redundancia de gateway mediante VRRP, monitoreo de interfaces y automatización de configuraciones.

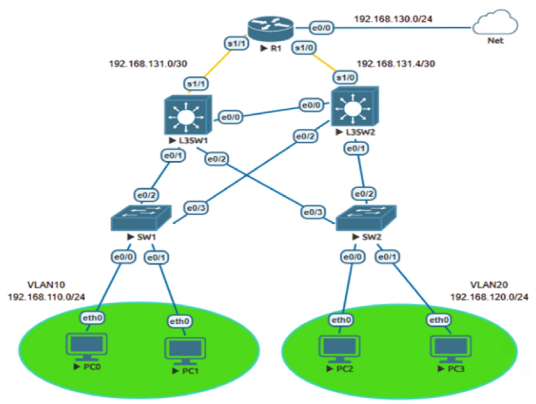



**Figura 1.** Topología experimental implementada en EVE-NG para la validación del sistema propuesto. Fuente: Los autores.

Cabe señalar que los resultados obtenidos corresponden a un entorno experimental controlado y constituyen evidencia de viabilidad técnica. Por tanto, no pretenden representar directamente el comportamiento de redes empresariales en producción, donde factores como la latencia real, la heterogeneidad de firmware y la carga operativa pueden introducir variaciones adicionales.

La Tabla 3 resume los principales recursos de hardware y software utilizados durante el desarrollo y evaluación de la propuesta.

**Tabla 3.** Recursos utilizados en el entorno experimental. Fuente: Los autores

| Componente | Descripción |
|---|---|
| Plataforma de emulación | EVE-NG Community Edition |
| Sistema operativo host | Ubuntu Server 24.04.3 LTS |
| Lenguaje de programación | Python 3.12.3 |
| Framework web | FastAPI + HTML/CSS/JavaScript |
| Motor de automatización | Ansible Core 2.18.10 |
| Protocolo de monitoreo | SNMP v2c |
| Protocolo de registro de eventos | Syslog |
| Router virtual | 1 |
| Switches capa 3 | 2 |
| Switches capa 2 | 2 |
| Equipos terminales (VPCS) | 4 |
| Interfaz Cloud | 1 |

### 2.3 Arquitectura del sistema

Con el propósito de automatizar tareas de configuración y monitoreo en entornos de red virtualizados, se desarrolló una plataforma centralizada basada en tecnologías de código abierto. La arquitectura propuesta integra capacidades de automatización de configuraciones, supervisión operacional y gestión de eventos mediante una estructura modular implementada en Python y Ansible.

El diseño del prototipo respondió directamente a las limitaciones identificadas en la literatura: la dependencia de software propietario de alto costo para tareas de monitoreo (Kim et al., 2016) y la escasa disponibilidad, en plataformas ligeras de código abierto, de mecanismos accesibles de validación previa al despliegue mediante revisión humana. Por ello, la arquitectura se orientó específicamente a entornos de laboratorio y educativos de pequeña escala, donde resulta necesario experimentar con automatización, configuración y monitoreo de dispositivos de red sin requerir infraestructura comercial especializada; la


e-ISNN 2550-6730 | Vol. 10, Núm. 2 (197-211): Julio-Diciembre, 2026 | DOI: 10.33936/isrtic.v10i2.8666

INFORMÁTICA Y SISTEMAS
Revista de Tecnologías de la Informática y las Comunicaciones

Facultad de Ciencias Informáticas
Universidad Técnica de Manabí
Av. Urbina y Che Guevara, Portoviejo, Ecuador
revista.iys@utm.edu.ec

extensión de este enfoque a redes empresariales o de mayor escala queda planteada como línea de trabajo futuro, sujeta a la incorporación de los mecanismos de seguridad descritos en las limitaciones del estudio. Bajo esta premisa, el sistema prioriza tecnologías de código abierto, bajo acoplamiento entre módulos y compatibilidad con protocolos estándar (SNMP, Syslog) ampliamente soportados incluso en equipos legados, de manera que su adopción no dependa de actualizaciones costosas de infraestructura.

El sistema fue diseñado siguiendo una arquitectura cliente-servidor, donde una aplicación web desarrollada con FastAPI actúa como punto central de interacción entre el administrador de red y los dispositivos gestionados. Desde la interfaz web el administrador puede ejecutar procesos de automatización, consultar información operacional y supervisar el estado de la infraestructura en tiempo real.

La arquitectura se encuentra compuesta por cinco módulos principales: módulo de gestión web, módulo de automatización de configuraciones, módulo de monitoreo SNMP, módulo de recolección de eventos Syslog y módulo de validación y despliegue de configuraciones.

### 2.3.1 Módulo de gestión web

Este módulo constituye la capa de presentación del sistema, desarrollada mediante una interfaz web estática implementada con HTML, CSS y JavaScript, servida directamente por FastAPI. Su función principal consiste en proporcionar una interfaz gráfica accesible desde navegador web para centralizar las tareas de administración de la infraestructura. A través de esta interfaz el administrador puede visualizar el inventario de dispositivos, consultar métricas operacionales, monitorear eventos Syslog y ejecutar procesos de automatización de manera centralizada.

### 2.3.2 Módulo de automatización de configuraciones

La automatización de configuraciones se implementó mediante Ansible, utilizando un inventario centralizado de dispositivos y playbooks específicos para cada tarea de administración. Este módulo permite generar automáticamente configuraciones de red a partir de parámetros suministrados por el usuario, garantizando uniformidad en los despliegues y reduciendo el riesgo de errores derivados de la intervención manual.

Como caso de aplicación, el sistema incorpora la generación automática de configuraciones VRRP. A partir de la información proporcionada por el administrador, el sistema calcula direcciones IP, prioridades y parámetros de redundancia, generando las líneas de configuración correspondientes para cada dispositivo participante.

### 2.3.3 Módulo de monitoreo SNMP

El monitoreo operacional se implementó mediante consultas SNMP realizadas desde Python utilizando la biblioteca PySNMP. Este módulo permite obtener información de los dispositivos de red sin necesidad de instalar agentes adicionales, aprovechando capacidades nativas ampliamente soportadas por equipos empresariales.

Entre los parámetros monitorizados se incluyen el nombre del dispositivo, tiempo de actividad (uptime), utilización de CPU y estadísticas de tráfico de interfaces. Para el cálculo del tráfico se emplearon los contadores estándar de bytes transmitidos y recibidos definidos en la Base de Información de Gestión versión 2 (MIB-II), permitiendo estimar la tasa de transferencia en tiempo real. Los datos obtenidos son retornados en tiempo real a la interfaz web sin persistencia en almacenamiento secundario, dado que el sistema está orientado a la consulta operacional inmediata. Esta decisión de diseño responde a que el entorno experimental fue concebido como una prueba de concepto en un entorno virtualizado, donde la persistencia continua de métricas habría implicado un consumo adicional de recursos de cómputo y almacenamiento del servidor sin aportar valor a los objetivos de validación planteados en este estudio.

### 2.3.4 Módulo de recolección de eventos Syslog

Con el fin de complementar la observabilidad de la infraestructura, se implementó un servidor Syslog integrado capaz de recibir y almacenar eventos generados por los dispositivos de red. Este módulo opera mediante sockets UDP sobre un puerto dedicado y registra la dirección IP de origen, el identificador del dispositivo, la marca temporal y el contenido completo del mensaje recibido. Los eventos recibidos son gestionados en memoria durante la sesión activa y presentados en la interfaz web para su consulta y auditoría en tiempo real, sin persistencia en almacenamiento secundario.

La centralización de eventos permite disponer de un mecanismo de auditoría y diagnóstico que facilita la identificación de cambios operacionales, fallos de conectividad y eventos relacionados con protocolos de red.

### 2.3.5 Módulo de validación y despliegue

Antes de aplicar cualquier configuración sobre los dispositivos administrados, el sistema genera una vista previa completa de los comandos que serán ejecutados. Este mecanismo permite que el administrador revise y valide manualmente los cambios propuestos antes de iniciar el proceso de despliegue.

Una vez aprobada la configuración, el sistema utiliza módulos de Ansible para transferir y aplicar los cambios sobre los dispositivos



CC BY NC ND

e-ISNN 2550-6730 | Vol. 10, Núm. 2 (197-211): Julio-Diciembre, 2026 | DOI: 10.33936/isrtic.v10i2.8666

seleccionados. Este enfoque reduce el riesgo de errores de aprovisionamiento derivados de configuraciones incorrectas y proporciona un control adicional sobre las modificaciones realizadas en la infraestructura. Cabe precisar que, en el resto del artículo, el término "validación" referido a este módulo designa específicamente esta previsualización de comandos sujeta a aprobación humana, y no una verificación automática de sintaxis, políticas o cumplimiento normativo.

La integración de estos cinco módulos constituye la base funcional del sistema cuya implementación se describe a continuación. La interacción entre estos componentes se ilustra en la Figura 2.

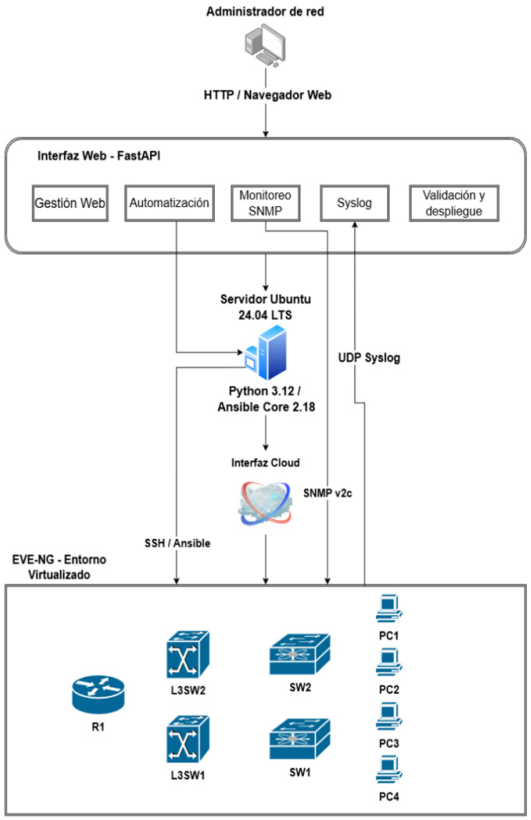



**Figura 2.** Arquitectura general del sistema propuesto.
Fuente: Los autores.

### 2.4 Procedimiento experimental y métricas de evaluación

La evaluación del sistema propuesto se realizó mediante un procedimiento experimental estructurado en dos líneas complementarias: la evaluación del módulo de aprovisionamiento, orientada a comparar la ejecución manual de tareas de configuración de red frente a su automatización mediante el sistema desarrollado, y la evaluación del módulo de monitoreo, orientada a verificar el desempeño y la concordancia de las consultas SNMP y Syslog. Ambas pruebas se efectuaron sobre la topología virtual descrita en la sección 2.2, utilizando los mismos dispositivos y parámetros de configuración, con el fin de garantizar la comparabilidad de los resultados. El procedimiento completo, desde la preparación de la topología hasta el análisis estadístico, se ilustra en la Figura 3. El código fuente completo, los playbooks de Ansible y los datos recolectados durante la experimentación están disponibles en el repositorio del proyecto (ver subsección de Disponibilidad de datos y materiales).




**Figura 3.** Procedimiento experimental seguido para la comparación entre método manual y automatizado.
Fuente: Los autores.

#### 2.4.1 Evaluación del módulo de aprovisionamiento

El escenario de evaluación consistió en la implementación de configuraciones de redundancia mediante el protocolo VRRP sobre dispositivos de capa 3, incluyendo la creación de Redes de Área Local Virtuales (VLAN), asignación de direcciones IP, definición de prioridades de respaldo y habilitación de mecanismos de recuperación automática. Estas tareas fueron ejecutadas inicialmente mediante procedimientos manuales a través de la CLI y posteriormente mediante el sistema desarrollado, el cual genera y despliega automáticamente las configuraciones utilizando Ansible.

Con el propósito de reducir la influencia de variaciones temporales y obtener resultados representativos, cada procedimiento fue ejecutado veinte veces bajo las mismas condiciones experimentales. Las veinte repeticiones de cada método se ejecutaron durante la misma sesión de trabajo. Una vez completadas las veinte repeticiones manuales, se restableció la configuración inicial de los dispositivos y se procedió a ejecutar las veinte repeticiones automatizadas sobre el mismo punto de partida. De este modo, ambos métodos partieron del mismo estado inicial de la infraestructura, aunque la ejecución secuencial de los bloques introduce la posible influencia del orden y la familiarización descrita en las limitaciones metodológicas. En cada repetición se registró el tiempo total requerido para completar

INFORMÁTICA Y SISTEMAS
Revista de Tecnologías de la Informática y las Comunicaciones

Facultad de Ciencias Informáticas
Universidad Técnica de Manabí
Av. Urbina y Che Guevara, Portoviejo, Ecuador
revista.iys@utm.edu.ec

la configuración de los dispositivos involucrados, medido desde el inicio de la operación hasta la correcta aplicación de todos los parámetros definidos. En el procedimiento manual, el tiempo se midió mediante cronómetro externo, con una resolución de un segundo; en el procedimiento automatizado, el tiempo se registró internamente en el backend del sistema desarrollado, con una resolución de milisegundos. Esta diferencia de resolución entre instrumentos se retoma como limitación metodológica en la sección de Resultados y Discusión. Las pruebas fueron ejecutadas por los mismos autores responsables del desarrollo del prototipo, quienes poseían un nivel técnico equivalente en la ejecución de comandos de configuración y monitoreo, lo cual permitió controlar la variabilidad asociada a la experiencia del operador entre ambos escenarios de evaluación.

Adicionalmente, se verificó la correcta implementación de las configuraciones mediante consultas posteriores a los dispositivos de red, comprobando la creación de las VLAN correspondientes, la asignación de direcciones IP, la formación de grupos VRRP y la elección adecuada de los dispositivos maestro y respaldo. Esta validación permitió confirmar que las configuraciones generadas por el sistema producían resultados funcionalmente equivalentes a los obtenidos mediante administración manual.

Como complemento a la validación funcional, se realizó una prueba adicional de conmutación por falla (failover) para evaluar la capacidad de recuperación automática del protocolo VRRP ante la caída de un dispositivo maestro. La prueba se ejecutó en dos escenarios: en el primero se detuvo el dispositivo L3SW1, maestro del grupo VRRP correspondiente a la VLAN10, mientras se mantenía un flujo continuo de paquetes ICMP hacia la dirección IP virtual del gateway; en el segundo se aplicó el mismo procedimiento sobre L3SW2, maestro del grupo correspondiente a la VLAN20. En ambos casos se verificó, mediante el comando show vrrp brief, que el dispositivo de respaldo asumiera correctamente el rol de maestro, y se registró el tiempo de recuperación mediante un cronómetro externo, dado que el servidor NTP del entorno de laboratorio no se encontraba sincronizado y las marcas de tiempo del registro Syslog no se consideraron confiables para este cálculo.

Las métricas de evaluación fueron seleccionadas con base en su capacidad para cuantificar directamente el impacto operativo de la automatización: el tiempo de configuración y su reducción porcentual permiten medir la eficiencia del proceso, mientras que la correctitud y la tasa de éxito de despliegue garantizan que la velocidad no se obtenga a costa de la fiabilidad. Estas métricas se resumen en la Tabla 4, junto con su descripción correspondiente.

### 2.4.2 Evaluación del módulo de monitoreo

La evaluación del módulo de monitoreo se centró en los dos dispositivos de capa 3 involucrados en la configuración VRRP evaluada en la sección 2.4.1, comparando la obtención manual de información operativa frente a su recuperación mediante el sistema desarrollado.

**Tabla 4.** Métricas de evaluación del módulo de aprovisionamiento. Fuente: Los autores.

| Métrica | Descripción |
|---|---|
| Tiempo de configuración (s) | Tiempo total desde inicio hasta correcta aplicación de parámetros |
| Reducción porcentual (%) | Diferencia relativa entre tiempo manual y automatizado |
| Correctitud de la configuración | Verificación del cumplimiento de parámetros definidos |
| Tasa de éxito de despliegue (%) | Proporción de configuraciones aplicadas correctamente |

En la modalidad manual, los datos de cada dispositivo fueron obtenidos a través de comandos ejecutados directamente en su interfaz de línea de comandos, incluyendo el tiempo de actividad, el uso de procesador, el estado de las interfaces de red y los eventos de registro recientes. El tiempo requerido para completar esta secuencia se midió mediante cronómetro, desde el inicio de la sesión de administración hasta la obtención de la totalidad de los datos solicitados. El tráfico cursado por las interfaces, al igual que en la modalidad automatizada, se consultó de manera independiente y no fue incluido en esta medición, con el fin de mantener la comparabilidad entre ambos métodos.

En la modalidad automatizada, la misma información fue recuperada mediante la función de consulta integrada en la interfaz del sistema desarrollado, la cual recupera de forma consolidada la identificación del dispositivo, el tiempo de actividad y el uso de CPU mediante SNMP, y el estado de sus interfaces de red mediante un playbook de Ansible, complementados con los eventos Syslog capturados previamente por el servidor embebido del sistema. El tiempo de respuesta se registró internamente en el backend del sistema. El tráfico de interfaces, al depender de dos lecturas SNMP sucesivas para calcular la tasa de transferencia, se consultó de manera independiente y no fue incluido en la medición del tiempo de respuesta.

Ambos procedimientos, manual y automatizado, fueron ejecutados veinte veces por dispositivo, siguiendo el mismo criterio de repetición aplicado en la evaluación del módulo de aprovisionamiento, y a cargo de los mismos autores responsables del desarrollo del prototipo.



e-ISNN 2550-6730 | Vol. 10, Núm. 2 (197-211): Julio-Diciembre, 2026 | DOI: 10.33936/isrtic.v10i2.8666

Adicionalmente, para una submuestra de cinco consultas automatizadas por dispositivo, los valores reportados por el sistema fueron contrastados con los obtenidos manualmente por CLI en el mismo instante con el propósito de verificar la concordancia de los datos recolectados. Para validar la captura de eventos Syslog, se generaron eventos controlados mediante la desactivación y reactivación de una interfaz de red antes de ejecutar la consulta automatizada, verificando posteriormente que dichos eventos aparecieran correctamente registrados en el panel de monitoreo del sistema. Para la comparación de valores entre el sistema y la CLI se definió como criterio de concordancia una diferencia de tiempo de actividad inferior a un minuto (dado el truncamiento a minutos que aplica la CLI de Cisco IOS) y una diferencia de uso de CPU inferior al 0.05%.

Las métricas seleccionadas para evaluar el módulo de monitoreo se resumen en la Tabla 5.

**Tabla 5.** Métricas de evaluación del módulo de monitoreo.
Fuente: Los autores.

| Métrica | Descripción |
|---|---|
| Tiempo de obtención de estado operativo (s) | Tiempo total desde el inicio de la consulta hasta disponer de uptime, CPU, interfaces y logs, manual vs. automatizado |
| Reducción porcentual (%) | Diferencia relativa entre tiempo manual y automatizado |
| Concordancia de datos reportados (%) | Coincidencia entre valores del sistema automatizado y valores obtenidos por CLI, sobre una submuestra |
| Captura de eventos Syslog | Verificación cualitativa de la correcta recepción de un evento generado de forma controlada |

Los datos obtenidos en ambas evaluaciones fueron analizados mediante estadística descriptiva, calculando valores de promedio, mínimo, máximo y desviación estándar para cada conjunto de mediciones. Los resultados permitieron cuantificar tanto el impacto de la automatización sobre la eficiencia operativa de las tareas de administración de red, como el desempeño del módulo de monitoreo y la concordancia de los datos obtenidos frente a los valores de referencia.

### 2.5 Análisis estadístico

Para contrastar las diferencias observadas entre los métodos manual y automatizado, se realizaron análisis estadísticos sobre los tres escenarios evaluados: tiempo de aprovisionamiento VRRP, tiempo de obtención de estado operativo en L3SW1 y tiempo de obtención de estado operativo en L3SW2. Previamente se evaluó el supuesto de normalidad mediante la prueba de Shapiro-Wilk y la homogeneidad de varianzas mediante la prueba de Levene. Dado que los grupos mostraron una distribución aproximadamente normal ($p > .05$) y varianzas significativamente distintas ($p < .001$) en las comparaciones evaluadas, se empleó la prueba t de Welch para contrastar las diferencias entre las distribuciones de tiempos. Se calculó adicionalmente el tamaño del efecto mediante la d de Cohen y, como contraste no paramétrico complementario, la prueba U de Mann-Whitney. Debido a que las repeticiones corresponden a ejecuciones sucesivas sobre una misma infraestructura experimental y no a unidades experimentales independientes en sentido estricto, los resultados de estos contrastes se interpretan dentro de las condiciones específicas del experimento y no como una estimación generalizable del comportamiento en otras topologías.

Se realizaron tres comparaciones principales entre métodos. Los valores de significancia se reportan individualmente para cada contraste y se interpretan conjuntamente con los tamaños de efecto y las limitaciones derivadas del diseño experimental.

### 2.6 Disponibilidad de datos y materiales

Los datos individuales de las repeticiones experimentales, los playbooks de Ansible, el código del sistema desarrollado (FastAPI) y la topología de laboratorio utilizada se encuentran disponibles públicamente en:

https://github.com/Sisalima28/netdevops-vrrp-monitoring.

## 3. Resultados y Discusión

### 3.1 Resultados de la evaluación del módulo de aprovisionamiento

La evaluación del módulo de aprovisionamiento se realizó sobre veinte repeticiones por método, ejecutando la configuración de redundancia VRRP entre los dos switches de capa 3 de la topología experimental. La Tabla 6 resume los resultados obtenidos.

**Tabla 6.** Estadística descriptiva del tiempo de configuración VRRP: manual vs. automatizado.
Fuente: Los autores.

| Método | Media (s) | Mínimo (s) | Máximo (s) | Desviación Estándar |
|---|---|---|---|---|
| Automatizado | 10.18 | 9.78 | 10.38 | 0.143 |
| Manual | 64.15 | 55.00 | 76.00 | 5.112 |

El sistema automatizado registró un tiempo promedio de 10.18 segundos, mientras que el procedimiento manual presentó un promedio de 64.15 segundos. Las desviaciones estándar registradas fueron de 0.143 y 5.112 segundos, respectivamente. Sin embargo, estas medidas de dispersión deben interpretarse con cautela debido a la diferente resolución de los instrumentos empleados: el procedimiento manual fue medido mediante un cronómetro externo con resolución de un segundo, mientras que el procedimiento automatizado utilizó una medición interna con resolución de milisegundos. Por tanto, la diferencia entre las desviaciones estándar no constituye evidencia directa de una mayor estabilidad o precisión del procedimiento automatizado.


e-ISNN 2550-6730 | Vol. 10, Núm. 2 (197-211): Julio-Diciembre, 2026 | DOI: 10.33936/isrtic.v10i2.8666

INFORMÁTICA Y SISTEMAS
Revista de Tecnologías de la Informática y las Comunicaciones

Facultad de Ciencias Informáticas
Universidad Técnica de Manabí
Av. Urbina y Che Guevara, Portoviejo, Ecuador
revista.iys@utm.edu.ec

La comparación principal se centra en los tiempos registrados y en la reducción relativa observada bajo las condiciones experimentales evaluadas.

Estos resultados representan una reducción del 84.13% en el tiempo requerido para completar la configuración de redundancia mediante el sistema propuesto, en comparación con el procedimiento manual tradicional. La diferencia entre métodos resultó estadísticamente significativa (prueba t de Welch, $t(19.0) = 47.20$, $p < .001$, $d = 14.92$), con un tamaño del efecto muy elevado, coherente con la separación casi completa observada entre las distribuciones de ambos grupos (U de Mann-Whitney = 0, $p < .001$) y no atribuible a un artefacto de cálculo. La Figura 4(a) ilustra esta diferencia junto con la variabilidad observada en cada método.

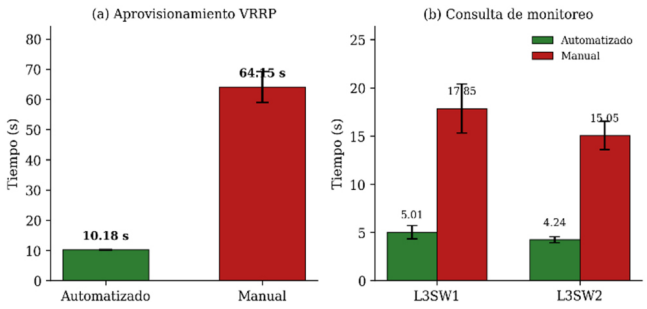



**Figura 4.** Comparación de tiempos entre método manual y automatizado. (a) Aprovisionamiento de configuración VRRP. (b) Consulta de estado operativo por dispositivo, en el módulo de monitoreo. Las barras de error representan la desviación estándar registrada en las veinte repeticiones de cada condición. Debido a la diferente resolución de los instrumentos empleados, estas medidas de dispersión deben interpretarse únicamente con carácter descriptivo.
Fuente: Los autores.

En cuanto a la correctitud de las configuraciones aplicadas, el sistema automatizado alcanzó una tasa de éxito del 100% (20/20 configuraciones correctas), mientras que el procedimiento manual presentó una tasa de éxito del 90% (18/20), registrándose dos errores de digitación en la asignación de direcciones IP de la interfaz VLAN. Este hallazgo respalda uno de los argumentos centrales del presente trabajo: además de la reducción en el tiempo de ejecución, la automatización mitiga el riesgo de errores humanos inherentes a la configuración manual repetitiva.

En cuanto a la prueba de conmutación por falla del protocolo VRRP, el primer escenario (caída de L3SW1) registró un tiempo de recuperación de 5.16 segundos, con una pérdida de 2 paquetes ICMP. El segundo escenario (caída de L3SW2) registró un tiempo de recuperación de 49.32 segundos, con una pérdida de 24 paquetes ICMP (Figura 5). En ambos casos, al restablecer el dispositivo original, este recuperó automáticamente su rol de maestro mediante el mecanismo de preemption habilitado por defecto.

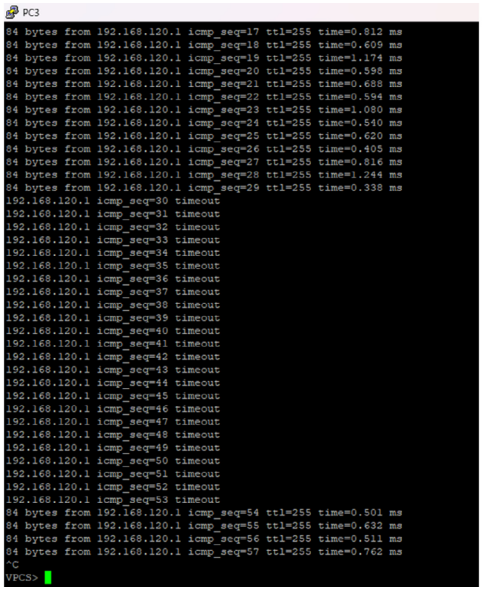



**Figura 5.** Pérdida de paquetes ICMP durante la conmutación por falla del dispositivo maestro L3SW2.
Fuente: Los autores.

### 3.2 Resultados de la evaluación del módulo de monitoreo

La evaluación del módulo de monitoreo se realizó sobre veinte repeticiones por método y por dispositivo, comparando la obtención de información operativa (identificación del dispositivo, tiempo de actividad, uso de CPU, estado de interfaces y eventos Syslog) de forma manual frente a su recuperación mediante el sistema desarrollado. La Tabla 7 resume los resultados obtenidos para ambos switches de capa 3.

**Tabla 7.** Estadística descriptiva del tiempo de obtención de estado operativo: manual vs. automatizado, por dispositivo.
Fuente: Los autores.

| Dispositivo | Método | Media (s) | Mínimo (s) | Máximo (s) | Desviación Estándar |
|---|---|---|---|---|---|
| L3SW1 | Auto | 5.01 | 3.86 | 6.44 | 0.694 |
| L3SW1 | Manual | 17.85 | 14.00 | 22.00 | 2.540 |
| L3SW2 | Auto | 4.24 | 3.60 | 4.99 | 0.308 |
| L3SW2 | Manual | 15.05 | 13.00 | 18.00 | 1.468 |



e-ISNN 2550-6730 | Vol. 10, Núm. 2 (197-211): Julio-Diciembre, 2026 | DOI: 10.33936/isrtic.v10i2.8666

En L3SW1, el sistema automatizado registró un tiempo promedio de 5.01 segundos frente a 17.85 segundos del procedimiento manual, lo que representa una reducción del 71.93%. En L3SW2, los resultados fueron consistentes: 4.24 segundos automatizado frente a 15.05 segundos manual, con una reducción del 71.83%. La cercanía entre ambos porcentajes de reducción (71.93% y 71.83%) sugiere un comportamiento consistente entre los dos dispositivos evaluados. Considerando ambos dispositivos de forma conjunta, la reducción promedio del módulo de monitoreo fue del 71.88%. Ambas diferencias resultaron estadísticamente significativas (L3SW1: t de Welch(21.8) = 21.81, $p < .001$, $d = 6.90$; L3SW2: t de Welch(20.7) = 32.23, $p < .001$, $d = 10.19$), lo que proporciona evidencia estadística de una diferencia entre las distribuciones de tiempos bajo las condiciones experimentales evaluadas. La Figura 4(b) presenta esta comparación de forma desagregada por dispositivo.

Se observó una diferencia moderada en los tiempos absolutos entre ambos switches (5.01 s frente a 4.24 s en modo automatizado). Esta diferencia puede estar asociada a características específicas de la ejecución y de la topología virtualizada; sin embargo, el diseño experimental no permite determinar una causa específica para esta variación.

Respecto a la concordancia de los datos recolectados por el sistema, se verificaron diez muestras (cinco por dispositivo), contrastando los valores de tiempo de actividad, uso de CPU y estado de interfaces reportados por el sistema contra los obtenidos directamente mediante comandos de verificación en la CLI de cada dispositivo. Los resultados de esta verificación se presentan en la Tabla 8. La totalidad de las muestras evaluadas presentó concordancia dentro del margen de tolerancia definido, en los tres tipos de dato verificados, incluyendo las once interfaces de red inspeccionadas por dispositivo. Cabe señalar que, al tratarse de un entorno emulado en reposo, el uso de CPU se mantuvo prácticamente nulo: en nueve de las diez muestras ambas fuentes reportaron 0%, y en la muestra restante (L3SW2, primera consulta) el sistema reportó 0.01% frente a 0% en la CLI, diferencia que se mantiene dentro del margen de tolerancia definido (<0.05%). Dado que el uso de CPU se mantuvo prácticamente nulo en todas las muestras, ofrece evidencia limitada sobre el comportamiento del sistema bajo condiciones de carga real; lo que sustenta es la concordancia observada entre el sistema y la CLI en condiciones de reposo, sin que ello implique necesariamente que ambas fuentes obtienen el dato desde el mismo origen interno.

La CLI de Cisco IOS reporta el uptime truncado a la unidad de minuto, mientras que el sistema reporta el valor exacto derivado del contador SNMP en centésimas de segundo; las diferencias decimales observadas corresponden a este truncamiento y no representan discrepancias entre las fuentes de referencia y el sistema.

Con el propósito de sintetizar los resultados de las pruebas aplicadas a las tres comparaciones evaluadas, la Tabla 9 resume los estadísticos de contraste, su significancia y el tamaño del efecto correspondiente.

**Tabla 8.** Verificación de concordancia de datos: sistema vs. CLI (submuestra de diez consultas).
Fuente: Los autores.

| Dispositivo - Muestra | Uptime sistema | Uptime CLI | CPU sistema (%) | CPU CLI (%) | Interfaces coinciden-tes |
|---|---|---|---|---|---|
| L3SW1 – 1 | 26.44 min | 26 min | 0 | 0 | 11/11 |
| L3SW1 – 2 | 40.02 min | 40 min | 0 | 0 | 11/11 |
| L3SW1 – 3 | 42.07 min | 42 min | 0 | 0 | 11/11 |
| L3SW1 – 4 | 43.25 min | 43 min | 0 | 0 | 11/11 |
| L3SW1 – 5 | 44.42 min | 44 min | 0 | 0 | 11/11 |
| L3SW2 – 1 | 4.71 min | 4 min | 0.01 | 0 | 11/11 |
| L3SW2 – 2 | 5.33 min | 5 min | 0 | 0 | 11/11 |
| L3SW2 – 3 | 7.11 min | 7 min | 0 | 0 | 11/11 |
| L3SW2 – 4 | 8.01 min | 8 min | 0 | 0 | 11/11 |
| L3SW2 – 5 | 10.89 min | 10 min | 0 | 0 | 11/11 |

**Tabla 9.** Resultados de las pruebas para la comparación de tiempos entre método manual y automatizado.
Fuente: Los autores.

| Comparación | t de Welch | gl | p | d de Cohen | U de Mann-Whitney | p (U) |
|---|---|---|---|---|---|---|
| Aprovisiona-miento VRRP | 47.20 | 19.0 | < .001 | 14.92 | 0.00 | < .001 |
| Monitoreo L3SW1 | 21.81 | 21.8 | < .001 | 6.90 | 0.00 | < .001 |
| Monitoreo L3SW2 | 32.23 | 20.7 | < .001 | 10.19 | 0.00 | < .001 |

Finalmente, la captura de eventos Syslog fue validada generando un evento controlado mediante la desactivación y reactivación de una interfaz de red. El evento fue correctamente registrado por el servidor Syslog embebido del sistema, confirmando la capacidad de captura de eventos del módulo implementado. Dado que el servidor NTP del entorno de laboratorio no se encontraba sincronizado (ver sección 2.4.1), no se reporta aquí una cifra de latencia derivada de las marcas de tiempo del registro, ya que estas no se consideran confiables para ese cálculo. La Figura 6 evidencia la correspondencia entre los eventos registrados en el dispositivo y los capturados por el sistema, incluyendo la coincidencia en el tipo de evento, la interfaz afectada y la secuencia temporal reportada.



**e-ISNN 2550-6730** | **Vol. 10, Núm. 2 (197-211): Julio-Diciembre, 2026** | **DOI: 10.33936/isrtic.v10i2.8666**

INFORMÁTICA Y SISTEMAS
Revista de Tecnologías de la Informática y las Comunicaciones

Facultad de Ciencias Informáticas
Universidad Técnica de Manabí
Av. Urbina y Che Guevara, Portoviejo, Ecuador
revista.iys@utm.edu.ec

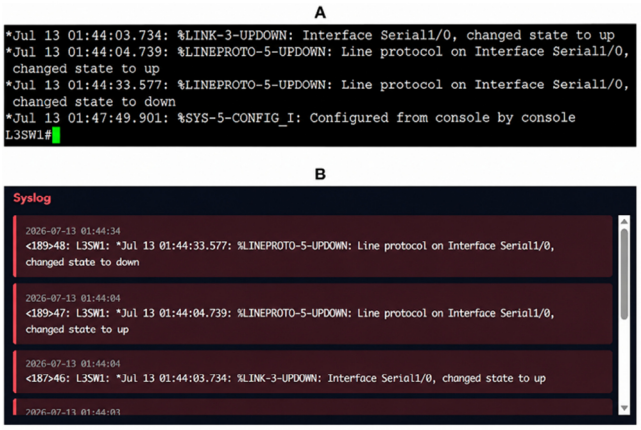



**Figura 6.** Verificación de captura de eventos Syslog generados de forma controlada. (a) Registro de eventos en la CLI del dispositivo L3SW1. (b) Registro correspondiente capturado por el módulo de monitoreo del sistema propuesto.
Fuente: Los autores.

### 3.3 Discusión

Los resultados obtenidos muestran una reducción de los tiempos registrados mediante el procedimiento automatizado respecto del procedimiento manual, con reducciones del 84.13% y 71.88% en las tareas de aprovisionamiento y obtención de estado operativo, respectivamente. Estas cifras son consistentes con resultados reportados previamente sobre automatización de tareas de red; Abdellatif et al. (2026) documentaron una automatización de topologías 4.5 veces más eficiente que la gestión manual, mientras que El-Mokadem et al. (2026) evidenciaron reducciones de hasta el 67% en tiempos de ejecución mediante técnicas de concurrencia en Python. No obstante, la comparación debe interpretarse dentro del alcance del presente experimento, debido a las diferencias en topología, escala, procedimiento de medición y condiciones experimentales, que impiden establecer una comparación directa de magnitud entre los estudios.

En cuanto al módulo de monitoreo, la reducción del 71.88% en el tiempo de obtención de estado operativo, junto con concordancia dentro del margen de tolerancia definido frente a los valores de referencia obtenidos por CLI, respalda uno de los planteamientos centrales de este trabajo: la viabilidad técnica del prototipo como base para explorar, en trabajos futuros y tras su correspondiente hardening de seguridad, alternativas de monitoreo de código abierto frente a soluciones propietarias (Kim et al., 2016), cuya comparación directa de costos no fue evaluada en este estudio, por lo que los resultados obtenidos no permiten establecer una ventaja económica ni extrapolar esta afirmación a entornos de producción. La verificación de la captura de eventos Syslog, mediante generación controlada de eventos, refuerza, dentro del entorno de laboratorio evaluado, la viabilidad del enfoque para escenarios que requieren observabilidad de eventos operativos.

Un hallazgo relevante fue la diferencia en la tasa de éxito entre la configuración manual (90%) y la automatizada (100%), originada en errores de digitación durante la asignación de direcciones IP. Este resultado es consistente con la premisa planteada en la introducción respecto a que la administración manual de redes empresariales está expuesta a errores de configuración que comprometen la continuidad operativa (El-Mokadem et al., 2026), este resultado sugiere una reducción de errores de configuración en las ejecuciones automatizadas evaluadas, aunque su generalización a otros escenarios requiere estudios adicionales.

No obstante, estos resultados deben interpretarse considerando las limitaciones del estudio. Como se indicó previamente, la evaluación se realizó en un entorno virtualizado y controlado, por lo que los tiempos obtenidos no representan directamente el comportamiento de redes en producción, donde la latencia real y la heterogeneidad de firmware podrían introducir variaciones adicionales. Asimismo, el tamaño de la muestra (veinte repeticiones por método) y el número reducido de dispositivos evaluados (dos switches de capa 3) limitan la generalización de los resultados a topologías de mayor escala, aspecto que también motivó la decisión de diseño de no persistir los datos de monitoreo en almacenamiento secundario, orientada a una prueba de concepto y no a un despliegue productivo continuo. De manera similar, la selección de SNMP versión 2c (SNMPv2c) para el módulo de monitoreo respondió a fines de simplicidad y validación funcional propias de un prototipo, dado que esta versión no incorpora mecanismos de autenticación de origen ni cifrado en el transporte de la información, apoyándose únicamente en un community string de texto plano. De cara a un eventual despliegue en producción, se recomienda migrar a SNMP versión 3 (SNMPv3), que incorpora autenticación basada en usuario y cifrado de la carga útil (USM), garantizando así la confidencialidad e integridad de los datos de gestión de red frente a posibles interceptaciones o accesos no autorizados. De manera similar, dado que el prototipo se desplegó y validó en un entorno de laboratorio aislado, sin exposición a redes externas, se priorizó la validación funcional del sistema sobre la implementación de mecanismos de autenticación y control de acceso. La interfaz web de FastAPI no incorporó autenticación de usuarios ni cifrado de transporte (HTTPS), y las credenciales utilizadas por Ansible para acceder a los dispositivos se almacenaron en el inventario sin cifrado. Para un eventual despliegue en producción, se recomienda habilitar autenticación basada en tokens y HTTPS en la interfaz FastAPI, así como el cifrado de credenciales mediante Ansible Vault y



CC BY NC ND

e-ISNN 2550-6730 | Vol. 10, Núm. 2 (197-211): Julio-Diciembre, 2026 | DOI: 10.33936/isrtic.v10i2.8666

la restricción de privilegios de la cuenta de automatización al mínimo necesario. Asimismo, dado que el servidor Syslog embebido opera sobre UDP, protocolo que no garantiza entrega ni autenticidad del origen, en producción se recomienda complementar la recepción de eventos con mecanismos de verificación de origen o migrar a transporte sobre TLS (RFC 5425). Adicionalmente, las repeticiones no se ejecutaron de forma alternada entre métodos (manual, automatizado, manual, automatizado), sino de forma secuencial por bloques (primero las veinte repeticiones manuales y, tras restablecer la configuración inicial, las veinte automatizadas). Este diseño no permite descartar un efecto de orden o aprendizaje entre bloques: al ejecutar primero el bloque manual, los operadores llegaron a la fase automatizada ya familiarizados con la topología y los comandos de verificación, por lo que los tiempos automatizados podrían estar ligeramente subestimados, o los tiempos manuales ligeramente sobreestimados, respecto de lo que se obtendría con un diseño cruzado o aleatorizado por bloques. La magnitud del efecto observado no permite determinar, con el diseño empleado, qué proporción de la diferencia corresponde exclusivamente al método de administración y qué proporción podría estar asociada al efecto de orden o familiarización; estudios posteriores deberán emplear un diseño aleatorizado o cruzado para aislar este factor. Cabe señalar también que el tiempo manual se midió con cronómetro externo (resolución de un segundo) y el tiempo automatizado se registró internamente por el sistema (resolución de milisegundos). Esta diferencia de resolución limita principalmente la comparación directa de las medidas de dispersión entre métodos y puede introducir cuantización en las mediciones manuales, por lo que los resultados relativos a las desviaciones estándar reportadas en las Tablas 6 y 7, y visualizadas en la Figura 4, deben interpretarse con cautela. La diferencia observada entre las medias debe considerarse principalmente como una comparación de los tiempos registrados bajo los instrumentos disponibles, sin interpretar la precisión de ambas mediciones como equivalente. Asimismo, la prueba de conmutación por falla de VRRP evidenció una diferencia notable entre ambos escenarios evaluados, la cual se explica por el hecho de que L3SW2 actuaba adicionalmente como puente raíz (Root Bridge) del protocolo Spanning Tree (STP) para la VLAN20, condición verificada mediante el comando show spanning-tree vlan 20 (Figura 7).

```
L3SW2#show spanning-tree vlan 20

VLAN0020
  Spanning tree enabled protocol ieee
  Root ID    Priority    32788
             Address     aabb.cc00.4000
             This bridge is the root
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec

  Bridge ID  Priority    32788  (priority 32768 sys-id-ext 20)
             Address     aabb.cc00.4000
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
             Aging Time  300 sec

Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- -------- --------------------------------
Et0/0               Desg FWD 100       128.1    P2p
Et0/1               Desg FWD 100       128.2    P2p
Et0/2               Desg FWD 100       128.3    P2p


L3SW2#
```

**Figura 7.** Verificación del puente raíz (Root Bridge) de Spanning Tree en L3SW2 para la VLAN20.
Fuente: Los autores.

Al tratarse de una implementación de STP clásico (IEEE 802.1D), la caída del puente raíz obligó a reconstruir completamente el árbol de expansión, proceso que, bajo los temporizadores por defecto de este protocolo (Max Age de 20 segundos y Forward Delay de 15 segundos), puede tardar entre 30 y 50 segundos en completarse, lo cual es consistente con el tiempo de recuperación observado. Este hallazgo evidencia que, si bien VRRP proporciona redundancia efectiva a nivel de enrutamiento, el tiempo de recuperación percibido por el usuario final puede verse afectado significativamente por la interacción con otros protocolos de la capa de enlace, especialmente cuando coincide con roles críticos de la topología. Dado que las pruebas fueron ejecutadas por los mismos autores responsables del desarrollo del prototipo, no puede descartarse un sesgo de familiaridad: los operadores contaban con un conocimiento profundo del funcionamiento interno del sistema automatizado, lo cual pudo optimizar su interacción con este frente a un usuario externo sin ese conocimiento previo. En contraste, la ejecución manual, aunque realizada por personal técnicamente competente, no se benefició de una preparación equivalente específica al escenario, dado que representa procedimientos estándar de CLI. Futuros estudios deberían incorporar operadores externos al desarrollo del sistema para controlar este posible sesgo.

Como líneas de trabajo futuro, se recomienda extender la evaluación a topologías con un mayor número de dispositivos y de mayor heterogeneidad de fabricantes, incorporar persistencia de métricas históricas para análisis de tendencias, y explorar la integración de protocolos de telemetría programable como complemento a SNMP en escenarios que lo permitan. De igual forma, se sugiere mejorar la presentación de los datos crudos obtenidos vía SNMP, como el formateo del tiempo de actividad a un formato legible, con el fin de optimizar la experiencia del administrador de red que utilice el sistema, así como evaluar el uso de Rapid PVST+ en lugar de STP clásico para reducir los tiempos de convergencia a nivel de capa 2 ante escenarios de conmutación por falla.

Asimismo, futuras evaluaciones deberían emplear un diseño experimental aleatorizado o cruzado, alternando el orden de aplicación de los métodos manual y automatizado, y utilizar instrumentos de medición con resolución equivalente para reducir el efecto del orden de ejecución y de la cuantización de las mediciones sobre la comparación de los resultados.

## 4. Conclusiones

El presente estudio evaluó, mediante un prototipo de prueba de concepto en un entorno de laboratorio virtualizado, la automatización basada en Python, Ansible y FastAPI para la configuración y el monitoreo de dispositivos de red, en una topología de pequeña escala (dos switches de capa 3).



**e-ISNN 2550-6730** | **Vol. 10, Núm. 2 (197-211): Julio-Diciembre, 2026** | **DOI: 10.33936/isrtic.v10i2.8666**

INFORMÁTICA Y SISTEMAS
Revista de Tecnologías de la Informática y las Comunicaciones

Facultad de Ciencias Informáticas
Universidad Técnica de Manabí
Av. Urbina y Che Guevara, Portoviejo, Ecuador
revista.iys@utm.edu.ec

Bajo el diseño experimental empleado (y con las limitaciones metodológicas discutidas en la sección 3.3), se observó una reducción del 84.13% en el tiempo de aprovisionamiento y del 71.88% en el tiempo de obtención de estado operativo, junto con una tasa de éxito del 100% frente al 90% del procedimiento manual. Estos resultados son coherentes con la pregunta de investigación y los objetivos planteados y muestran una mayor tasa de configuraciones correctas en las ejecuciones automatizadas bajo las condiciones experimentales evaluadas; sin embargo, el diseño secuencial de las pruebas y la diferente resolución de los instrumentos de medición limitan la atribución exclusiva de estas diferencias al método de administración.

Los resultados obtenidos respaldan la viabilidad técnica del prototipo como prueba de concepto en entornos de laboratorio y educativos de pequeña escala. No obstante, los resultados no permiten inferir su comportamiento ni su seguridad en redes empresariales de producción. La concordancia dentro del margen de tolerancia definido obtenida en la verificación de los datos recolectados vía SNMP, junto con la validación de la captura de eventos Syslog en tiempo real, respalda esta viabilidad técnica dentro del alcance evaluado. Como línea de trabajo futuro, el prototipo podría explorarse como base para soluciones de automatización y monitoreo de código abierto en organizaciones con restricciones presupuestarias, siempre que se incorporen previamente los mecanismos de seguridad descritos en las limitaciones y se valide su comportamiento en topologías de mayor escala e infraestructura multivendor.

No obstante, dado que la validación se realizó en un entorno virtualizado con un número reducido de dispositivos, futuras investigaciones deberían orientarse a evaluar el comportamiento del sistema en topologías de mayor escala y en condiciones de red de producción, así como a incorporar mecanismos de persistencia de métricas históricas que permitan un análisis de tendencias a largo plazo, ampliando así el alcance práctico del modelo de automatización y monitoreo integrado propuesto en este trabajo. Asimismo, dada su arquitectura modular y su base en protocolos de red estándar, el modelo propuesto podría adaptarse a ecosistemas de Internet de las Cosas (IoT), en los que el elevado número de dispositivos interconectados incrementa considerablemente la complejidad de su configuración y supervisión manual; en dichos escenarios, la automatización del aprovisionamiento y el monitoreo centralizado podrían contribuir a reducir la complejidad asociada a la configuración y supervisión de un número elevado de dispositivos.

**Agradecimientos**

Agradecemos a la Universidad Técnica de Machala por brindarnos los recursos y el respaldo institucional necesarios para el desarrollo de esta investigación. Asimismo, expresamos nuestro agradecimiento a nuestros tutores por el acompañamiento académico y técnico brindado a lo largo de todo el proceso. Finalmente, agradecemos a nuestras familias por su apoyo, paciencia y motivación constante durante el desarrollo de este trabajo.

**Contribución de los autores**

**Marlon Steven Sisalima Ulloa:** Investigación, Metodología, Software, Redacción. **Leyton Lenin Veintimilla Guzmán:** Investigación, Metodología, Software, Redacción. **Oscar Efrén Cárdenas Villavicencio:** Validación, Redacción - revisión y edición. **Rodrigo Fernando Morocho Román:** Supervisión, Redacción - revisión y edición.

**Conflictos de interés**

Los autores declaran no tener ningún conflicto de interés.

**Referencias bibliográficas**

Abdellatif, M. M., Desouki, O., & AbdelRaheem, M. (2026). Performance analysis of network automation techniques for dense IP networks. Scientific Reports, 16(1), 9532. https://doi.org/10.1038/s41598-026-40975-9

Bruschetti, F. S., Guevara, J., Abeledo, M. C., & Priano, D. A. (2023). An Empirical Evaluation of Automated Configuration Tools for Software-Defined Networking: A Usability and Performance Perspective. Ingénierie des systèmes d information, 28(5), 1127-1134. https://doi.org/10.18280/isi.280502

Dalla Palma, S., Di Nucci, D., & Tamburri, D. A. (2020). AnsibleMetrics: A Python library for measuring Infrastructure-as-Code blueprints in Ansible. SoftwareX, 12, 100633. https://doi.org/10.1016/j.softx.2020.100633

Datta, A., Imran, A. T. M. A., & Biswas, C. (2023). Network Automation: Enhancing Operational Efficiency Across the Network Environment. ICRRD Quality Index Research Journal, 4(1). https://doi.org/10.53272/icrrd.v4i1.1

Dsilva, N., Karunakaran, V., Autenrieth, A., Elbers, J.-P., & Bauschert, T. (2025). Network Automation using



e-ISNN 2550-6730 | Vol. 10, Núm. 2 (197-211): Julio-Diciembre, 2026 | DOI: 10.33936/isrtic.v10i2.8666

NetDevOps in Optical Networks. 2025 International Conference on Optical Network Design and Modeling (ONDM), 1-3. https://doi.org/10.23919/ONDM65745.2025.11029370

El-Mokadem, E. S., Bataineh, B., El-Mokadem, S., M. Ahmed, A., & A. Torad, M. (2026). Design and evaluation of a Python-based network automation system for internet of things devices. Bulletin of Electrical Engineering and Informatics, 15(1), 237-252. https://doi.org/10.11591/eei.v15i1.9562

Espinel Villalobos, R. I., Ardila Triana, E., Zarate Ceballos, H., & Ortiz Triviño, J. E. (2021). Design and Implementation of Network Monitoring System for Campus Infrastructure Using Software Agents. Ingeniería e Investigación, 42(1), e87564. https://doi.org/10.15446/ing.investig.v42n1.87564

Han, G., Shao, H., Duan, R., Zhuang, C., & Xiang, Q. (2024). ConfigHub: A Network Configuration Sharing Platform. Proceedings of the SIGCOMM Workshop on Formal Methods Aided Network Operation, 33-38. https://doi.org/10.1145/3672199.3673885

Imoukhuede, A. B., Sheltami, T. R., Mahmoud, A. H., & Barnawi, A. Y. (2025). Optimization of network device hardening in a multivendor environment. Scientific Reports, 15(1), 15042. https://doi.org/10.1038/s41598-025-97894-4

Khumaidi, A. (2021). IMPLEMENTATION OF DEVOPS METHOD FOR AUTOMATION OF SERVER MANAGEMENT USING ANSIBLE. Jurnal Transformatika, 18(2), 199-209. https://doi.org/10.26623/transformatika.v18i2.2447

Kim, Y.-I., Park, S.-J., Jung, N.-J., Choi, M.-S., & Park, B.-S. (2016). Design and implementation of NMS using SNMP for AMI network device monitoring. 2016 IEEE International Conference on Power System Technology (POWERCON), 1-6. https://doi.org/10.1109/POWERCON.2016.7754021

Kokuryo, S., Kondo, M., & Mizuno, O. (2020). An Empirical Study of Utilization of Imperative Modules in Ansible. 2020 IEEE 20th International Conference on Software Quality, Reliability and Security (QRS), 442-449. https://doi.org/10.1109/QRS51102.2020.00063

Kumara, I., Garriga, M., Romeu, A. U., Di Nucci, D., Palomba, F., Tamburri, D. A., & Van Den Heuvel, W.-J. (2021). The do's and don'ts of infrastructure code: A systematic gray literature review. Information and Software Technology, 137, 106593. https://doi.org/10.1016/j.infsof.2021.106593

Marzuki, K., Husain, H., & Apriliana, Z. (2024). Automation of Software Defined Network (SDN) Configuration Management Based on Proxmox with Ansible. International Journal of Electronics and Communications Systems, 4(2), 87-97. https://doi.org/10.24042/ijecs.v4i2.23821

Marzuki, K., Kholid, M. I., Hariyadi, I. P., & Mardedi, L. Z. A. (2023). Automation of Open VSwitch-Based Virtual Network Configuration Using Ansible on Proxmox Virtual Environment. International Journal of Electronics and Communications Systems, 3(1), 11-20. https://doi.org/10.24042/ijecs.v3i1.16524

Mazin, A. M., Rahman, R. A., Kassim, M., & Mahmud, A. R. (2021). Performance Analysis on Network Automation Interaction with Network Devices Using Python. 2021 IEEE 11th IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE), 360-366. https://doi.org/10.1109/ISCAIE51753.2021.9431823

Mohammed Daffalla, E. (2023). Ansible: A Reliable Tool for Automation. Electrical and Computer Engineering Studies, 2. https://doi.org/10.58396/eces020104

Mohd Fuzi, M. F., Abdullah, K., Abd Halim, I. H., & Ruslan, R. (2021). Network Automation using Ansible for EIGRP Network. Journal of Computing Research and Innovation, 6(4), 59-69. https://doi.org/10.24191/jcrinn.v6i4.237

Muhammad, T., & Munir, M. (2023). Network Automation. European Journal of Technology, 7(3), 23-42. https://doi.org/10.47672/ejt.1547

Ponce Loor, Y. A., & Herrera Tapia, J. S. (2023). Evaluación del Rendimiento de una red IPv6 utilizando IPSec en Modo Túnel. Informática y Sistemas: Revista de Tecnologías de la Informática y las Comunicaciones, 7(2). https://doi.org/10.33936/isrtic.v7i2.5710

Prasetyo, L., Prihandi, I., Rifqi, M., & Budiarto, R. (2024). Implementation of automation configuration of enterprise networks as software defined network. Computer Science and Information Technologies, 5(2), 99-111. https://doi.org/10.11591/csit.v5i2.pp99-111

Sahoo, P., Pujar, S., Nalawade, G., Genhardt, R., Mandel, L., & Buratti, L. (2024). Ansible Lightspeed: A Code Generation Service for IT Automation. Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering, 2148-2158. https://doi.org/10.1145/3691620.3695277

Suresh Reddy Thati. (2025). Configuration and compliance automation in modern networks: A framework for enhanced security and operational efficiency. World Journal of Advanced Engineering Technology and Sciences, 15(2), 2513-2519. https://doi.org/10.30574/wjaets.2025.15.2.0613

Yaseen, N. (2025). From Counters to Telemetry: A Survey of Programmable Network-Wide Monitoring. Network, 5(3), 38. https://doi.org/10.3390/network5030038