АНАЛІЗ І ПОРІВНЯННЯ ЗАСОБІВ АВТОМАТИЗОВАНОГО КОНФІГУРУВАННЯ ПРИСТРОЇВ ІНФОКОМУНІКАЦІЙНИХ МЕРЕЖ

Яковенко К.О.

Харківський національний університет радіоелектроніки,
кафедра інфокомунікаційної інженерії ім. В.В. Поповського,
Україна.

E-mail: kyrylo.yakovenko@nure.ua

Abstract

The automation of network equipment configuration has become essential for efficient infrastructure management in contemporary information and communication networks. Manual device configuration, particularly in large-scale networks, is time-consuming and labor-intensive, significantly increasing the likelihood of human error. Automation tools such as Ansible, Puppet, and SaltStack streamline the configuration and management of hundreds of devices simultaneously, reducing the time and effort needed to sustain infrastructure. These tools not only enhance configuration reliability but also support easier maintenance and scalability of networks. This work provides a comparative analysis of key automation tools to highlight their strengths and weaknesses and offers recommendations for selecting the most suitable solution.

Автоматизація мереж є ключовим елементом концепції «Інфраструктура як код» (Infrastructure as a Code, IaC), що передбачає передачу процесів управління та налаштування програмним рішенням. Важливість автоматизації полягає не лише у прискоренні виконання завдань, а й у підвищенні надійності та стабільності роботи мереж. Інструменти автоматизації, такі як Ansible, Puppet та SaltStack, дозволяють централізовано керувати конфігураціями, забезпечуючи швидке масштабування та оперативне внесення змін. Ці інструменти широко використовуються в галузі інформаційних технологій для спрощення та оптимізації процесів управління конфігурацією мережної інфраструктури, проте мають свої особливості, що впливає на їх вибір та застосування.

Отже, автоматизація конфігурації мережного обладнання дозволяє значно прискорити процес налаштування, знижуючи ризики, пов'язані з людськими помилками. Ручне конфігурування мереж, особливо коли потрібно налаштувати велику кількість пристроїв, може зайняти кілька днів і потребує ретельної перевірки, щоб уникнути помилок, які можуть призвести до порушення роботи мережі. Завдяки автоматизації можна створити єдину конфігурацію, яку згодом застосовують до всіх пристроїв, тим самим зменшуючи потребу у повторенні однакових операцій і підвищуючи узгодженість налаштувань [1 – 4].

Ansible – популярний інструмент з відкритим кодом, що використовує безагентну архітектуру. Він дозволяє інженерам швидко створювати конфігурації у форматі YAML, який є легким для читання та розуміння. Наприклад, для налаштування інтерфейсу можна створити простий playbook, який підключається до групи пристроїв та конфігурує IP-адресу інтерфейсу. Це дозволяє забезпечити одноманітність конфігурацій усіх пристроїв у мережі (рис. 1) [2].

```
- name: Налаштування інтерфейсу
  hosts: routers
  tasks:
    - name: Встановити IP-адресу на GigabitEthernet0/1
      ios_config:
        lines:
          - interface GigabitEthernet0/1
          - ip address 192.168.1.1 255.255.255.0
          - no shutdown
```

Рис. 1. Приклад Playbook в Ansible

```
node 'router1' {
  network_interface { 'GigabitEthernet0/1':
    ensure      => 'up',
    ipaddress   => '192.168.1.1',
    netmask     => '255.255.255.0',
    description  => 'Основний інтерфейс',
  }
}
```

Рис. 2. Приклад маніфесту в Puppet

Puppet використовує декларативний підхід, що дозволяє вказати бажаний стан пристрою, а система автоматично досягає цього стану. Puppet базується на агент-базованій архітектурі, що дозволяє пристроям взаємодіяти з центральним сервером. Однак, з появою Puppet Bolt, стало можливим гнучке та ефективне безагентне налаштування мережних пристроїв. У декларативній мові Puppet описується, що інтерфейс повинен бути активним і мати певну IP-адресу, що показано на (рис. 2) [3].

SaltStack відомий своєю швидкістю і гнучкістю. Він може працювати як з агентами, так і без них, використовуючи proxy-агенти для керування мережними пристроями. SaltStack також використовує YAML для опису конфігурацій, що робить його схожим на Ansible, але з вищою продуктивністю в асинхронних операціях. Наприклад, для налаштування інтерфейсу у SaltStack можна використовувати стейт-файли, які дозволяють конфігурувати параметри інтерфейсу (рис. 3) та файл конфігурації для інтерфейсу (рис. 4) [4].

```
configure_interface:
  netconfig.managed:
    - filename: interface.conf
```

Рис. 3. Приклад стейт-файлу в SaltStack

```
interface GigabitEthernet0/1
 ip address 192.168.1.1 255.255.255.0
 description Основний інтерфейс
 no shutdown
```

Рис. 4. Приклад файлу конфігурації interface.conf

Щоб забезпечити максимальну ефективність від використання автоматизованих засобів, важливо враховувати не лише технічні характеристики кожного інструмента (таблиця 1), але й можливості інтеграції з існуючими системами управління та моніторингу. Наприклад, Ansible відзначається сумісністю з різними системами контролю версій, такими як Git, що дозволяє організувати централізоване управління конфігураціями і забезпечує відстеження змін. Puppet, завдяки своєму декларативному підходу, забезпечує високу точність та узгодженість конфігурацій, що є особливо важливим для організацій, де кожен збій може призвести до значних втрат. SaltStack зі свого боку пропонує асинхронну архітектуру, яка дозволяє одночасно обробляти великий обсяг запитів, що забезпечує швидке масштабування навіть у складних і розподілених мережах.

Вибір відповідного інструменту для автоматизації конфігурації мережних пристроїв є критичним рішенням, яке залежить від специфічних потреб та умов вашої організації. Якщо ви шукаєте рішення, яке забезпечить простоту використання та швидкий старт, Ansible може стати ідеальним варіантом. Його інтуїтивно зрозумілий синтаксис, заснований на YAML, та агент-менше архітектура дозволяють швидко розпочати автоматизацію без необхідності встановлення додаткового програмного забезпечення на цільових системах. Це особливо корисно для невеликих команд або проєктів, де час впровадження та навчання є критичними факторами.

У випадку, коли ваша інфраструктура є великою та складною, і ви готові інвестувати час та ресурси у навчання персоналу, варто розглянути Puppet. Цей інструмент пропонує потужні можливості для управління складними конфігураціями та залежностями між компонентами системи. Puppet використовує власну декларативну мову, яка дозволяє точно описувати бажаний стан інфраструктури. Це забезпечує високий рівень контрольованості та узгодженості в масштабних середовищах, де помилки можуть мати значні наслідки. Інвестиції в навчання та впровадження Puppet можуть окупити-

ся за рахунок підвищення ефективності та надійності управління конфігурацією в довгостроковій перспективі.

Таблиця 1. Порівняння Ansible, Puppet Bolt та SaltStack

Характеристика	Ansible	Puppet Bolt	SaltStack
Операційна система	Linux, macOS, Windows	Linux, macOS, Windows	Linux, macOS, Windows
Перший реліз	Лютий 2012 року	Жовтень 2017 року	Березень 2011 року
Поточна версія	10.5.0 (8 жовтня 2024 року)	3.30.0 (24 травня 2024 року)	3007.1 (22 травня 2024 року)
Написаний на	Python	Ruby	Python
Вихідний код	Відкритий	Відкритий	Відкритий
Декларативний/Імперативний	Обидва	Обидва	Обидва
Агент/Без агентів	Без агентів	Без агентів	Обидва (агент-базований та без агентів)
Push/Pull модель	Push	Push	Обидва
Протокол комунікації	SSH, WinRM	SSH, WinRM	ZeroMQ, SSH
Мова скриптів	YAML (Ansible Play-books)	YAML, Puppet Plans	YAML (SLS), Jinja
Ліцензія	GNU GPL-3.0	Apache-2.0	Apache-2.0
Термінологія	Playbook, Inventory	Manifest, Targets	States, Pillars

Якщо для вашої організації критичною є швидкість виконання завдань, і ви готові мати справу зі складнішим процесом налаштування, SaltStack може бути відповідним вибором. Завдяки своїй асинхронній архітектурі та можливості працювати як в агент-базованому, так і в безагентному режимах, SaltStack забезпечує високу продуктивність і масштабованість. Він здатний одночасно обробляти велику кількість запитів, що є важливим у динамічних середовищах з високими вимогами до швидкодії. Проте слід бути готовим до більш стрімкої кривої навчання та більш комплексного налаштування, оскільки SaltStack може вимагати більш глибокого розуміння його внутрішніх механізмів та архітектури.

Отже, при виборі інструменту для автоматизації конфігурації мережних пристроїв необхідно враховувати такі фактори, як розмір та складність інфраструктури, готовність інвестувати в навчання персоналу, а також пріоритети щодо швидкості та простоти використання. Ретельний аналіз цих аспектів допоможе обрати рішення, яке найкраще відповідатиме вашим бізнес-потребам і сприятиме ефективному та надійному управлінню IT-інфраструктурою.

Крім того, варто враховувати можливість інтеграції вибраного інструмента з існуючими системами моніторингу та управління. Це дозволяє отримувати актуальну інформацію про стан мережі в реальному часі та швидко реагувати на зміни або потенційні проблеми. Ефективне поєднання інструментів автоматизації з моніторинговими платформами може забезпечити проактивний підхід до підтримки інфраструктури, що є особливо важливим у складних і динамічних середовищах. Також до уваги слід брати сумісність із засобами контролю версій, такими як Git, що спрощує управління конфігураціями та дозволяє відстежувати зміни. Це створює можливості для більш прозорого контролю над інфраструктурою, сприяючи її стабільності й прогнозованості.

Література

1. Network Automation Trends and Strategy [Електрон. ресурс]. – Режим доступу: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/network-automation-strategy-wp.html.
2. Ansible Documentation. Red Hat. URL: https://docs.ansible.com.
3. Puppet Documentation. Puppet Labs. URL: https://puppet.com/docs.
4. SaltStack Documentation. Salt Project. URL: https://docs.saltproject.io.