IIETA International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Ingénierie des Systèmes d'Information
Vol. 28, No. 5, October, 2023, pp. 1127-1134
Journal homepage: http://iieta.org/journals/isi

An Empirical Evaluation of Automated Configuration Tools for Software-Defined Networking: A Usability and Performance Perspective

Fabio Sergio Bruschetti, Javier Guevara, María Claudia Abeledo*, Daniel Alberto Priano

Centro de Investigación y Desarrollos en Informática (CIDI), Instituto de Tecnologías Emergentesy Ciencias Aplicadas (ITECA), Universidad Nacional de San Martín (UNSAM), San Martín 1650, Argentina

Corresponding Author Email: mabeledo@unsam.edu.ar

https://doi.org/10.18280/isi.280502

Received: 19 June 2023
Revised: 7 October 2023
Accepted: 13 October 2023
Available online: 31 October 2023

Keywords:
SDN, computer networks, software-defined networking, network management, software tools, network configuration scripting, configuration management

ABSTRACT

The advent of Software Defined Networking (SDN) has ushered in an era where the functions of interconnected devices are no longer constrained by their original design. Instead, these devices, now transformed into "general-purpose" nodes within the network, have roles that are defined by their configuration settings. Given that these configurations can be compiled into a computer file, software tools have been developed to consolidate and automate the administration of configuration parameters across all devices in an SDN network. These tools, akin to source code control tools used in programming languages, are capable of managing configurations for individual or groups of devices simultaneously. This study presents an evaluation of three such tools—Ansible, Puppet, and Chef—assessing their merits and demerits across various performance and usability dimensions, including configuration, installation, ease of use, and management capabilities. The comparative analysis reveals Ansible as a remarkably versatile tool, offering a wealth of advantages that make it a compelling choice for a majority of automation and configuration management tasks.

1. INTRODUCTION

In the realm of OpenFlow-enabled Software-Defined Networking (SDN) [1, 2], changes to the network may be instantaneously implemented, yet often necessitate human approval and intervention. The average latency for tangible results to materialize from such modifications is approximately one to two weeks-a delay that poses significant management challenges in legacy systems, as exemplified by Voice over IP (VoIP) applications [3]. This predicament has stimulated the emergence of automation tools like Ansible [4], Puppet [5], and Chef [6].

Ansible, a free software tool, offers robust cross-platform automation capabilities. Primarily designed for IT professionals, it facilitates the deployment of applications, updating of workstations and servers, configuration of cloud resources, and management of software and hardware configurations [7]. With no dependency on agent software and no need for additional security infrastructure, Ansible provides a straightforward implementation process.

Meanwhile, Puppet, a software tool, empowers its users to define infrastructure through code (SDN) and effectively manage multiple servers. It also enforces system settings and configurations. As a part of the DevOps [8, 9] platform, Puppet holds significant value for managing multiple servers [10].

Chef, an open-source cloud configuration management and deployment software, is devised to orchestrate servers in the cloud or within a departmental data center. It enables DevOps to rapidly instantiate any server as needed, obviating the need for separate management tools for each individual or standalone server [11].

This study undertakes a comprehensive comparison of these three tools, delineating their relative strengths and weaknesses. It includes simulation tests and analyses of their evolution over time. The evolution of simulation testing in Ansible, Puppet, and Chef has been largely driven by the need to cater to increasingly complex systems, diverse environments, and stricter security demands. These tools have progressively adopted more sophisticated techniques for automation, testing, and configuration validation-enhancing the reliability and efficiency of configuration automation solutions. While the tests for the current study are yet to be completed due to the diversity of proposed scenarios, the findings will be discussed in future work. The study concludes by detailing the factors that contribute to the prevailing acceptance of one tool over the others.

2. NETWORK ADMINISTRATION

Network automation is the process of automating the configuration, management, testing, deployment, and operation of physical and virtual devices within a network. Network services can substantially be enhanced through automated tasks, functions, and repetitive processes.

Any type of network can use this kind of automation. Implementing hardware and software-based network automation will improve efficiency, reduce human error, and lower operating expenses for data centers, service providers, and businesses operations.

DevOps is a current cultural and technical trend in most companies focused on digital technologies. The purpose of DevOps is to eliminate the differences between the software developers and the operations that run on the infrastructure.

Once this goal is achieved, the development, testing and deployment processes will be more frequently, reliable and faster. This leads a cultural change where teams will evolve from stand-alone to cross-functional. DevOps also leads a change in the security arena because rules and parameters of automated processes must be adapted to ensure quick and continuous releases to patch bugs or provide updates. DevOps tools also support cross-functional interoperability and automated workflows to enable multi-platform integration [12].

On the other hand, NetDevOps [13] takes the collaboration, tools and automation approaches and extends them to network architects and operators. NetDevOps (named by Cisco) or DevNetOps combine network engineers and their tools.

Cisco, a leading provider of networking hardware, appears to be leading the NetDevOps shift from theory to practice with some of its customers. In other words, one of the main leaders in the networking market tries to move its clients to use of "infrastructure-as-code" and go towards automation and scripting, applying some of those ideas [14]. These are the foundations of the Cisco developer community, where members can learn all about theirs API [15] and how they can relate to a special kind of networks called intent-based networks [16].

In this way, companies consolidate networks operation in the DevOps culture and automation ensuring that their hardware is connected properly. Routers and switches will be programmed and configured trough an API which also will manage security and provide the analytics capability according to DevOps principles and processes for networks.

To implement changes in a test network and then move them into production networks in a consistent and secure way, CI/CD pipelines (Continuous Integration/Continuous Delivery) can storage the network configurations as code in a source code control program.

Although these changes are relatively new, service providers where the first to promote some of these NetDevOps transformations. In addition of this, many changes have emerged from the corporate IT areas where there is a greater need to automate and manage more and more network devices.

Ansible, Puppet and Chef are widely used tools for network automation and DevOps implementation practices. These tools help streamline and simplify infrastructure and application management, enabling greater efficiency, reliability and scalability in development and operations environments. The relationship for each tool mentioned is detailed below:

1. Ansible is a central tool in DevOps practices, as it can be used to automate everything from infrastructure provisioning to application deployment and configuration management. Ansible playbooks allow infrastructure and applications to be defined and maintained as code, facilitating collaboration between development and operations teams. Ansible also integrates with CI/CD tools to enable automated deployments.

2. Puppet plays a key role in DevOps by enabling automated management and configuration of infrastructure and applications. It allows you to define the desired state of systems and ensure that it is maintained at all times. This facilitates consistent and reproducible application deployment, and also fosters collaboration across teams throughout the development and operational lifecycle.

3. Chef is very popular in DevOps environments because of its focus on infrastructure as code. Chef cookbooks allow you to define, maintain and version infrastructure and application configuration in a similar way to software source code. This facilitates collaboration, automation and continuous deployment, which are key pillars of DevOps.

3. NETDEVOPS IN COMPANIES

In companies experimenting with NetDevOps, networking teams are focusing on establishing a culture and environment where creating, testing, and publishing changes to the network can happen more quickly, frequently, and reliably. Developers and operators working together are primarily interested in launching fast and stable releases. However, there is a lot of work to be done regarding communication between the network and the DevOps teams and their tools.

But what are the benefits of NetDevOps? Given its roots in DevOps, it makes sense that NetDevOps adopts many of the same goals. "DevOps is described by four principles: a holistic system thinking approach (see the whole system, not just one part), no organisational silos, rapid feedback, and automation to reduce work," says Joel King, an independent network automation architect.

Network operations has lagged behind other functional areas that support an organisation's IT infrastructure, King points out, especially when it comes to automation through network programmability.

"To implement a new application or service that supports the business, compute, storage and network components require configuration changes or the implementation of new hardware," King explains. "Often these changes are made manually by an engineer typing in a terminal window, and they may need to be reviewed through a change control process and implemented during an off-hours time slot.

NetDevOps "helps improve agility and is particularly valuable for organisations that deploy infrastructure as code, because the network is often a bottleneck," says Andrew Lerner, vice president of networking research at Gartner Inc (USA) "NetDevOps practices drive clear workflows and documentation, which helps with auditing, governance and troubleshooting" [17].

4. THE NEED FOR NEW TOOLS

Nowadays, this is an imperative, not only from the perspective of the management and operation of physical devices but also from the perspective of the analysis and control of network traffic. Manual or "human" verification will no longer work properly on current scenarios. Using a manual approach, for example, the number of bits in network traffic difficult to find specific binary flows in network connections.

When we talk about network devices configuration, the CLI (Command-Line Interface) method is the most used to make changes to configurations. This method allows access to devices through the console port, the auxiliary port, or through Telnet or SSH. Once connected to the CLI, network technicians can make changes to device settings. However, the CLI method has several drawbacks. First, it offers the wrong level of abstraction by allowing human operators to operate the console without being able to validate that the proper procedures are being followed. Also, different vendors do not use a standard CLI language.

CLI (Command Line Interface) method is a common way of interacting with computer systems and executing commands to perform tasks. Although it is widely used and has many advantages, it also has certain limitations. Some of these limitations include:

Complexity of commands: Some commands can be complicated and have a syntax that is difficult to remember. This can lead to human error if commands are entered incorrectly.

Learning curve: Learning to use a CLI effectively can take time, especially for novice or non-technical users. The need to memorise commands and their syntax can be challenging.

Visualisation limitations: Command line interfaces often do not provide complete visual representations of information, which can make it difficult to understand certain data, especially in complex systems.

Difficulty in automation: Automating tasks through the CLI can be complex, as it often requires the use of scripts or scripting to achieve this. This can be more difficult to maintain and less flexible than more robust automated solutions.

Need for technical knowledge: Using the CLI generally requires a solid knowledge of the terminology and structure of the system or software in question. This may exclude non-technical users or those less familiar with the technology.

Difficulty in graphical environments: Compared to graphical user interfaces (GUIs), CLIs may be less intuitive for some people, especially those who are more accustomed to visual interactions.

Feedback limitations: Some CLIs may provide limited or insufficiently descriptive information about errors that occur, which can make troubleshooting difficult.

Cross-platform incompatibility: Some CLI commands may be specific to certain platforms or operating systems, which may require adjustment if changing environments.

Difficulty with complex tasks: Performing complex, multi-faceted tasks through the CLI can be more complicated and error-prone than doing so through a specially designed GUI.

Industry reacted and introduced NETCONF [18]. NETCONF is the standard to install, modify and remove any configuration of network devices, while YANG [19] is used to model both the configuration and the status of the network elements. YANG organizes data definitions in tree structures and provides modeling features such as extensible types, separation of status and configuration data, handling of syntactic and semantic constraints, among others. These definitions are organized in modules that allow their extensibility and reuse. On the other hand, NETCONF has various constrains to use a same version on different vendor operating systems. Many of them use a proprietary version which makes it difficult to write NETCONF applications for use in multi-vendor networks.

NETCONF was basically created to make automation easier, but the difficulties it presented made automation even more difficult. Also, old troubleshooting tools like Ping and Traceroute did not provide a holistic assessment of how the network is performing. For example, Traceroute has problems with unnumbered IP links; these types of links are best for automated network environments. On the other hand, Ping does not provide information about network performance. These tools were originally created for simpler environments.

For this reason, the need arises to progress towards a vendor-independent solution that allows configuring the rules and policies of any network and being able to verify if they are aligned with what is expected of them, that is, their intention. The solution must be regardless of the number of devices, the operating system installed, the traffic rules and any other type of policy that should be configured. There is a need for the networks to be automated and predictable. The existing tools did not add value. A new model is required to trace all traffic and device interactions, not just at the device level, but at the entire network level.

5. NETWORK AUTOMATION TOOLS

As we have mentioned, network automation is the process of setting up software to automatically manage, configure, test, deploy, and operate network devices (whether physical or virtual). For this reason, SDN networks will make it possible to implement in a much simpler way the automation before described and network automation tools will be the key.

These tools discover and map all connected devices, manage different network configuration, provide resources, and allow to plan their network capacity. The automation of the network can be implemented through script languages (lines of code of a programming language that will be executed on a trigger event) or based on software (conventional languages with source code that must be interpreted or compiled for use). The last ones are also known as smart network automation tools. Software-based tools will be discussed below as they can almost eliminate the performance of manual tasks.

The following is a summary analysis of each tool, its advantages and disadvantages:

(1) Ansible:

1) Advantages:

Ease of use: Ansible uses a simple, readable YAML-based syntax, making it easier to learn and use for those new to configuration automation.

No agents required: Ansible operates over SSH or WinRM, which means there is no need to install agents on managed nodes, simplifying the deployment and administration process.

Dry-run mode: Ansible allows you to simulate changes before applying them, which helps prevent errors.

Cross-platform support: It can manage operating systems and devices from different platforms, including Linux, Windows and network devices.

Extensive community and documentation: Ansible has an active community and a wealth of online resources available.

2) Disadvantages:

Scalability: While Ansible can handle large deployments, it can face performance issues compared to more scalability-oriented tools.

Runtime: Ansible can be slower compared to other tools in large-scale deployments due to its SSH-based execution model.

(2) Puppet:

1) Advantages:

Resource management model: Puppet uses a declarative model to manage resources, allowing you to define the desired state and Puppet takes care of enforcing it.

Broad ecosystem: Puppet has a large number of predefined modules that make it easy to configure and manage various components.

Long-term management: Puppet is suitable for environments that require long-term configuration management and constant maintenance of the desired state.

Role and profile management: Puppet allows a clear separation between role definition, profiles and node-specific configuration.

2) Disadvantages:

Learning curve: Puppet can be more complex for beginners to learn due to its terminology and structure.

Requires agents: Managed nodes must have a Puppet agent installed, which can increase complexity and maintenance requirements.

(3) Chef:

1) Advantages:

Infrastructure as code: Chef allows infrastructure to be defined as code, which facilitates automation and consistent deployment.

Flexibility: Chef is highly configurable and adapts well to diverse environments and use cases.

Large community: Chef has an active community and a variety of resources available online.

2) Disadvantages:

Learning curve: Chef can be complex to learn for beginners due to its terminology and approach.

Requires agents: Like Puppet, Chef requires agents installed on managed nodes, which can add complexity and maintenance requirements.

More initial configuration: Chef may require more initial configuration compared to other tools.

6. ANSIBLE TOOL

This tool is designed for multi-tier deployments [20], modeling the interrelationship of all IT systems across the infrastructure rather than just managing one system at a time. It does not use agents or additional custom security scheme, so it is easy to implement. Most importantly, it uses a quite simple language (YAML, in the form of Ansible Playbooks) that enables automation tasks to be described in an almost plain English.

Ansible connects to your nodes and distributes small programs called "Ansible modules". Ansible then runs these modules using SSH [21] by default and removes them when done.

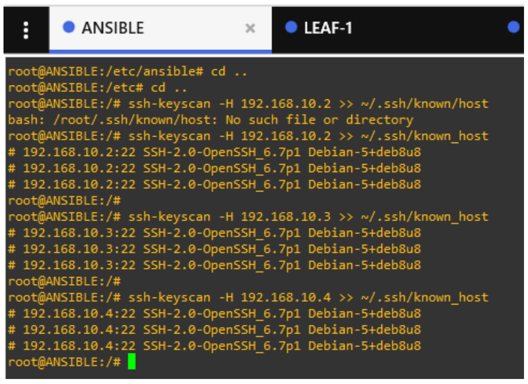
Your module library can reside on any machine and no servers, daemons, or databases are required. Generally, it will work with the user's favorite programs such as remote terminal access, text editor and a version control system for tracking changes.

Ansible supports passwords, and the best way to use it is via SSH keys with the SSH-agent as shown in Figure 1. Any user can log in, it does not have to be root. Then the user can use the SU or SUDO commands without problems. Ansible's "authorized_key" [22] module is a great way to use it to control which machines can access which hosts.

Through Ansible, multiple inventories can be configured statically or dynamically, composed of hosts, groups of hosts and groups of groups, host variables, group variables, non-SSH connections and many more. This parameterization is stored in a text file as shown in Figure 2.

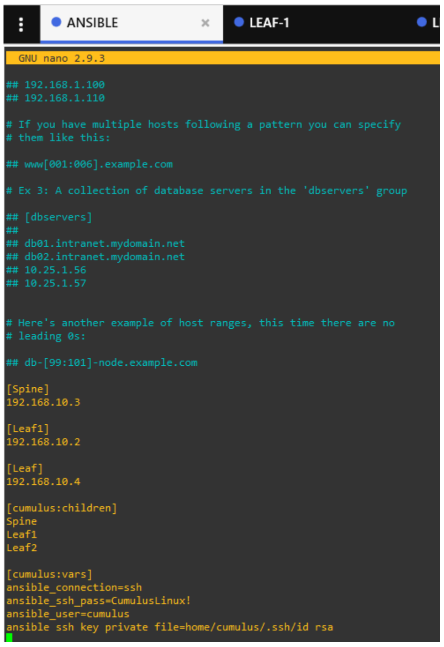
To add new hosts to the network, it is not necessary an additional server to generate the SSH keys, with Ansible it will be possible to generate these keys on each host.

As shown in Figure 3, all components of the infrastructure topology can be orchestrated precisely with Ansible playbooks [24]; it will allow a detailed control over how many nodes can be tackle at once.



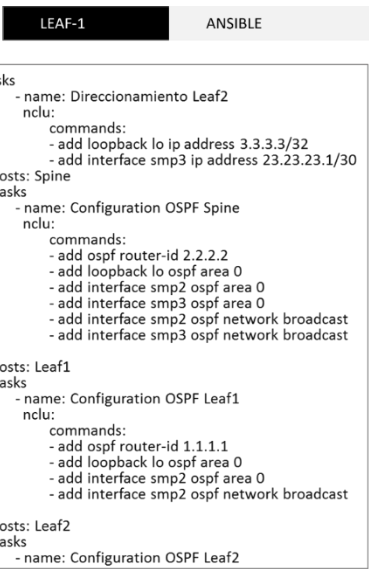
```
root@ANSIBLE:/etc/ansible# cd ..
root@ANSIBLE:/etc# cd ..
root@ANSIBLE:/# ssh-keyscan -H 192.168.10.2 >> ~/.ssh/known/host
bash: /root/.ssh/known/host: No such file or directory
root@ANSIBLE:/# ssh-keyscan -H 192.168.10.2 >> ~/.ssh/known_host
# 192.168.10.2:22 SSH-2.0-OpenSSH_6.7p1 Debian-5+deb8u8
# 192.168.10.2:22 SSH-2.0-OpenSSH_6.7p1 Debian-5+deb8u8
# 192.168.10.2:22 SSH-2.0-OpenSSH_6.7p1 Debian-5+deb8u8
root@ANSIBLE:/#
root@ANSIBLE:/# ssh-keyscan -H 192.168.10.3 >> ~/.ssh/known_host
# 192.168.10.3:22 SSH-2.0-OpenSSH_6.7p1 Debian-5+deb8u8
# 192.168.10.3:22 SSH-2.0-OpenSSH_6.7p1 Debian-5+deb8u8
# 192.168.10.3:22 SSH-2.0-OpenSSH_6.7p1 Debian-5+deb8u8
root@ANSIBLE:/#
root@ANSIBLE:/# ssh-keyscan -H 192.168.10.4 >> ~/.ssh/known_host
# 192.168.10.4:22 SSH-2.0-OpenSSH_6.7p1 Debian-5+deb8u8
# 192.168.10.4:22 SSH-2.0-OpenSSH_6.7p1 Debian-5+deb8u8
# 192.168.10.4:22 SSH-2.0-OpenSSH_6.7p1 Debian-5+deb8u8
root@ANSIBLE:/#
```

Figure 1. An example of an Ansible implementation, where the generation of SSH keys was done-GNS3 2.2.8 simulator [23]



```
  GNU nano 2.9.3

## 192.168.1.100
## 192.168.1.110

# If you have multiple hosts following a pattern you can specify
# them like this:

## www[001:006].example.com

# Ex 3: A collection of database servers in the 'dbservers' group

## [dbservers]
##
## db01.intranet.mydomain.net
## db02.intranet.mydomain.net
## 10.25.1.56
## 10.25.1.57


# Here's another example of host ranges, this time there are no
# leading 0s:

## db-[99:101]-node.example.com

[Spine]
192.168.10.3

[Leaf1]
192.168.10.2

[Leaf]
192.168.10.4

[cumulus:children]
Spine
Leaf1
Leaf2

[cumulus:vars]
ansible_connection=ssh
ansible_ssh_pass=CumulusLinux!
ansible_user=cumulus
ansible ssh key private file=home/cumulus/.ssh/id rsa
```

Figure 2. Example of the file for inventory management in Ansible - GNS3 2.2.8 simulator



```
tasks
        - name: Direccionamiento Leaf2
          nclu:
                commands:
                - add loopback lo ip address 3.3.3.3/32
                - add interface smp3 ip address 23.23.23.1/30
- hosts: Spine
  tasks
        - name: Configuration OSPF Spine
          nclu:
                commands:
                - add ospf router-id 2.2.2.2
                - add loopback lo ospf area 0
                - add interface smp2 ospf area 0
                - add interface smp3 ospf area 0
                - add interface smp2 ospf network broadcast
                - add interface smp3 ospf network broadcast

- hosts: Leaf1
  tasks
        - name: Configuration OSPF Leaf1
          nclu:
                commands:
                - add ospf router-id 1.1.1.1
                - add loopback lo ospf area 0
                - add interface smp2 ospf area 0
                - add interface smp2 ospf network broadcast

- hosts: Leaf2
  tasks
        - name: Configuration OSPF Leaf2
```

Figure 3. An example of an Ansible playbook - GNS3 2.2.8 simulator

7. PUPPET TOOL

Puppet is an open-source software tool for the deployment and management of network configurations. It is most used in Linux and Windows to move configurations to multiple application servers at the same time. Puppet can also be used on various platforms, including IBM mainframes, Cisco switches, and Mac OS servers.

Like other DevOps applications, Puppet does more than just automate system administration. It changes the human workflow and allows developers and sysadmins to work together. Developers can write, test, and launch applications without waiting for Operations staff to provide the necessary resources.

Puppet is available in both open source and commercial versions. It has its own language called the eponymous Puppet [25]. Puppet automates configuration changes and removes manual script-based changes. However, Puppet is not just another shell language like PowerShell for Windows or Unix, or Linux Bash shells. Also, Puppet is not a pure programming language like PHP. Puppet uses a declarative, model-based approach to automation of the IT infrastructure. This allows Puppet to define the infrastructure through lines of code and to enforce the system configuration through specific applications.

7.1 Puppet modeling capabilities

Puppet identifies the current state of a node, defines the model of the desired end state, and describes the actions required to move from one to the other. Each server instance managed by Puppet receives a catalog of resources and relationships with its current state, compares it to the desired state, and then defines and makes the necessary changes for the system to meet that desired state. Puppet will be able to manage the software and its services by creating the complete configurations through lines of code.

Puppet encourages users to control the different levels of complexity of the settings. Users will be able to write code that is reusable, easy to configure, understand and refactor. To achieve this, Puppet uses profiles and roles. The code can be separated into the following three levels:

(1) *Component modules*: They manage each technology, such as puppetlabs-apache [26].

(2) *Profiles* [27]: Container classes that use multiple component modules to configure a layered stack of technologies. For example, you can create a profile to configure Jenkins, the integration application, its web interface, and automated tasks.

(3) *Roles* [27]: Container classes that use multiple profiles to build a complete system configuration. For example, a server will have standard profiles, such as "base operating system profile" and "base web server profile". The first one could declare that the server must run Ubuntu 16.04.2, while the other would declare that it must use NGINX [28].

All this stuff (tools, languages, profiles, roles, processes, etc.) can seem to add additional complexity. In fact, it provides the flexibility and potential to create practical and specific interfaces to automate the configurations of each system within an organization. This will make, for example, hierarchical data easier to use, system configurations easier to read, and refactoring easier to perform as shown in Figure 4.

Ansible and Puppet are two popular configuration management tools, and the choice between them depends on several factors.

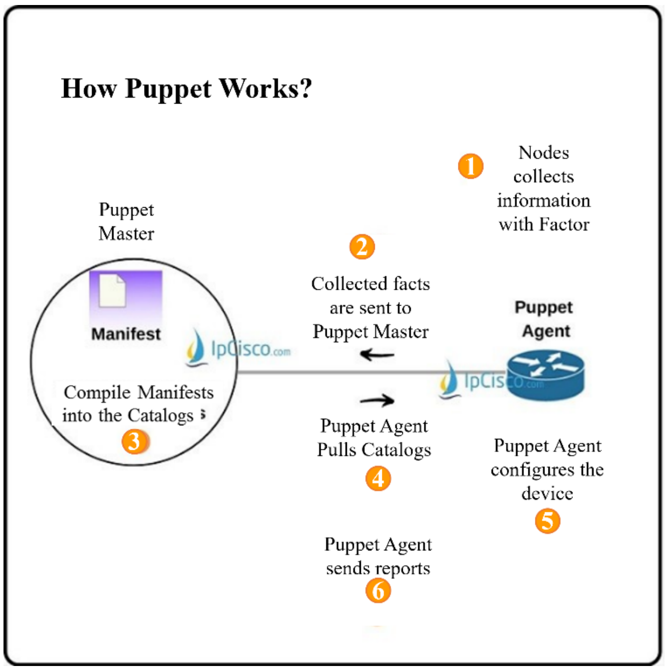

How Puppet Works?
1
Nodes collects information with Factor
Puppet Master
2
Collected facts are sent to Puppet Master
Manifest
Puppet Agent
Compile Manifests into the Catalogs
3
Puppet Agent Pulls Catalogs
4
Puppet Agent configures the device
5
Puppet Agent sends reports
6
IpCisco.com

Figure 4. Puppet working model
Source: Cisco System

Ease of use and quick learning: Ansible tends to have a smoother learning curve than Puppet.

It doesn't require agents: Ansible operates over SSH or WinRM and does not require the installation of agents on managed nodes. This simplifies the deployment and administration process, which can be an advantage if you want to avoid the effort of maintaining agents on all systems.

Ad hoc task automation: Ansible is particularly effective for automating one-off and ad hoc tasks. Ansible makes it easy to run commands on multiple servers on an occasional basis or to perform specific tasks without complex configuration.

Focus on simplicity: Ansible uses YAML to define configurations and tasks, which can be more intuitive for those familiar with markup languages or text-based configuration.

Application deployment and orchestration: Ansible can be a solid choice because of its focus on infrastructure as code and its ability to work with a variety of platforms.

Smaller or medium-sized deployments: Ansible may be better suited for medium-sized or smaller deployments, where simplicity and flexibility may outweigh the scalability and complexity needs that Puppet could better handle.

On the other hand, the choice between Puppet and Ansible depends on a number of factors, and there are situations where Puppet might be preferable over Ansible. Here are some considerations for deciding when it's better to use Puppet over Ansible:

Long-term management and constant maintenance: Puppet is especially well-suited for environments that require long-term configuration management and constant maintenance of the desired state. If you are looking for a tool that is effective at maintaining configuration consistency over time and ensuring that systems comply with policies on an ongoing basis, Puppet could be a solid choice.

Role and profile management: Puppet is known for its role and profile management approach. If you want to clearly and structurally define the configuration of different types of systems (roles) and node-specific customisations (profiles), Puppet offers a robust approach to achieve this.

Complex configurations and multiple states: If the infrastructure is large and complex with multiple states and node-specific configurations, Puppet may be preferable. Its declarative model allows you to define the desired state and Puppet takes care of applying it consistently across all nodes.

8. CHEF TOOL

Chef converts and models "cloud configuration management" in lines of code through a flexible and versatile process which is human-readable and easily verifiable. Infrastructure-as-code enables both the management of local and cloud resources.

Chef is also a framework for automating and managing infrastructure and applications. Specifically, Chef translates system administration tasks into reusable definitions, known as cookbooks and recipes. In a recipe, Chef's authors define the desired state of a system by writing its configuration in lines of code. Chef then processes that setting along with data about the specific node where the code is running to ensure that the desired state matches the final state of the system.

8.1 Chef automation

Automation makes the process much more scalable. With Chef, you simply clone your existing platform into a test platform. You do not need to configure servers or clusters manually. Your test platform can be the public cloud, the same used for the production environment. The entire setup process can take minutes instead of hours, days, or weeks.

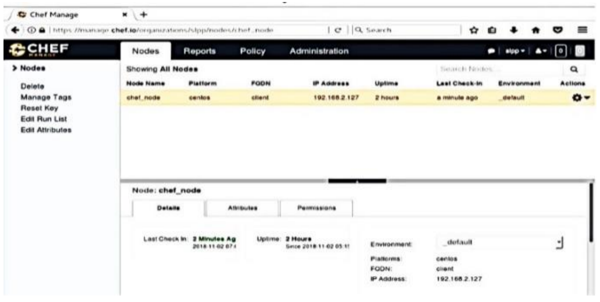
The following is an implementation of the Chef tool. In this example, if a new website configuration does not work, it does not need to be manually reconfigured. Chef is used to automatically revert to the previous version of the application code in the production environment and for all users. In Figure 5 we present a screen capture of the Chef tool [29]. Figure 6 shows an Oracle VirtualBox screen capture on workstation side.

Chef also allows cloud deployments quite easy. An example of this can be the implementation of NGINX to analyze the performance of an Apache web server. This will be done using the NGINX cookbook [30] for Linux servers.

You do not need to become a NGINX expert. You just need to implement NGINIX on a test server, transfer the web programs using Chef's Recipes [31], and start making comparisons.

(1) Listed below are reasons to use Ansible instead of Chef:

Simplicity and speed of deployment: Ansible is known for its ease of use and smoother learning curve. If you need to deploy an automation solution quickly or if you have a team that is new to configuration automation, Ansible may be a better choice.

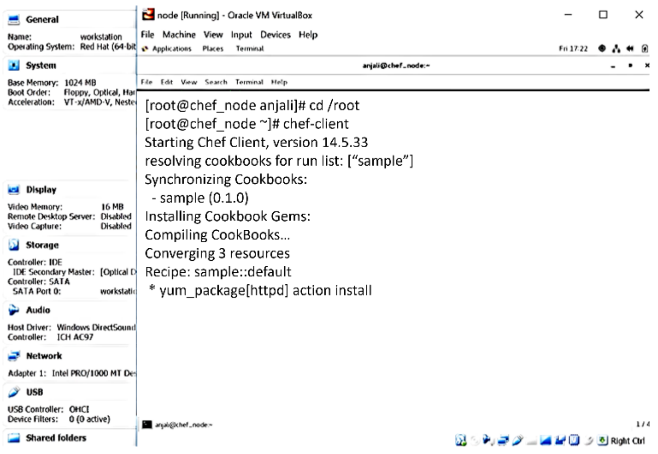


Node Name	Platform	FQDN	IP Address	Uptime	Last Check-In	Environment	Actions
chef_node	centos	client	192.168.2.127	2 hours	a minute ago	_default	

Node: chef_node

Last Check In: 2 Minutes Ag 2018-11-02 07:(
Uptime: 2 Hours Since 2018-11-02 05:1
Environment: _default
Platforms: centos
FQDN: client
IP Address: 192.168.2.127

Figure 5. Node configuration using Chef tool


General
Name: workstation
Operating System: Red Hat (64-bit
System
Base Memory: 1024 MB
Boot Order: Floppy, Optical, Har
Acceleration: VT-x/AMD-V, Neste
Display
Video Memory: 16 MB
Remote Desktop Server: Disabled
Video Capture: Disabled
Storage
Controller: IDE
IDE Secondary Master: [Optical D
Controller: SATA
SATA Port 0: workstati
Audio
Host Driver: Windows DirectSound
Controller: ICH AC97
Network
Adapter 1: Intel PRO/1000 MT De
USB
USB Controller: OHCI
Device Filters: 0 (0 active)
Shared folders
node [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places Terminal
anjali@chef_node:~
File Edit View Search Terminal Help

```
[root@chef_node anjali]# cd /root
[root@chef_node ~]# chef-client
Starting Chef Client, version 14.5.33
resolving cookbooks for run list: ["sample"]
Synchronizing Cookbooks:
  - sample (0.1.0)
Installing Cookbook Gems:
Compiling CookBooks...
Converging 3 resources
Recipe: sample::default
 * yum_package[httpd] action install
```

Figure 6. Workstation configuration running on Oracle VM VirtualBox

Ad hoc task automation: Ansible shines at automating one-off, ad hoc tasks, such as running commands on multiple servers or performing specific tasks without complex configuration. If you're looking for a solution for quick administration tasks, Ansible is a solid choice.

No agents required: Ansible operates without the need to install agents on managed nodes. If you prefer to avoid installing and maintaining agents on your systems, Ansible is an attractive option.

(2) Reasons to use Puppet instead of Chef are also listed below:

Long-term management-Puppet is suitable for environments that require long-term configuration management and consistent maintenance of the desired state. If you are looking for a tool that is effective at maintaining configuration consistency over time, Puppet may be more appropriate.

Role and profile management: Puppet excels at defining roles, profiles and node-specific configuration. If you want to clearly separate configuration logic from node-specific details, Puppet may be more appropriate.

Large-scale infrastructure: If you have a large and complex infrastructure, Puppet may be more scalable and effective for managing the configuration of multiple large-scale systems.

To conclude with Chef's analysis, the following are real-world scenarios for the use of Chef:

Simplicity and speed of deployment: Ansible is known for its ease of use and smoother learning curve. If you need to deploy an automation solution quickly or if you have a team that is new to configuration automation, Ansible may be a better choice.

Ad hoc task automation: Ansible shines at automating one-off, ad hoc tasks, such as running commands on multiple servers or performing specific tasks without complex configuration. If you're looking for a solution for quick administration tasks, Ansible is a solid choice.

No agents required: Ansible operates without the need to install agents on managed nodes. If you prefer to avoid installing and maintaining agents on your systems, Ansible is an attractive option.

9. CONCLUSIONS

In simple terms, the tools described above provide an abstraction layer between the existing configuration of a server and its desired state. Optimum configuration state for this network devices will be achieved by focusing more on the desired result than on the detailed tasks required to achieve it.

In this review we can see that Ansible has several advantages over the other tools since it is more oriented to SysOps due to its structure and paradigm. Instead, Puppet and Chef are focused on developers. Ansible is the easiest option to configure and then you would be starting to use it immediately. The tool has a detailed and structured documentation.

As we can see, the Ansible tool has gained popularity because it is oriented to uniform infrastructure and there are several add-on components available to improve its user interface capabilities and functionality. There is a huge group of administrators who have chosen it as a configuration management tool among the other competitors mentioned hereto.

Puppet is a reliable tool with particularly good usability. Taking full advantage of its wide range of features, structure, and scalability requires some practical knowledge of Ruby. Puppet setup maybe much detailed and sometimes complicated, but it is the safest choice if you are looking for a homogeneous software environment. The Puppet user will need to learn new procedures and functions to program the tool.

Table 1. SDN automation configuration tools comparison

Item\Tool	ANSIBLE	PUPPET	CHEF
Language	Python, YAML	Ruby, Puppet DSL, Embedded Ruby (ERB), DSL	Chef DSL Ruby
Usage	Easy	Complex	Complex
Architecture	Only master (Agentless)	Master-Agent	Master-Agent
Installation / Setup	Easy	Complex	Complex
Configuration	Only Pull	Only Pull	Both Push and Pull
Management	Easy (push model)	Complex (master-agent)	Complex (Chef server – Managed systems)

Chef is a simple, well-designed tool and much more usable than Puppet. It has a demanding learning curve for SysOps who lack experience in application development and coding as it requires a broader knowledge of programming languages and more experience.

In general, all three tools are expected to continue to be part of the automation and configuration management landscape. However, the choice of which one to use will depend on the specific needs and goals of each particular organisation, as well as how each tool fits with emerging trends in technology and DevOps practices.

In Table 1 we expose a comparison of main characteristic of each tool.

REFERENCES

[1] Nguyen, T.H., Bonnet, C. (2016). 5-IP mobility management for future public safety networks. Wireless Public Safety Networks 2, Buenos Aires, Argentina, pp. 127-172. https://doi.org/10.1016/B978-1-78548-052-2.50005-0

[2] Abeledo, M.C., Priano, D.A., Guevara, J., Bruschetti, F.S. (2023). Comparison of flow forwarding between software-defined and legacy networks based on fixed routing and QoS conditions. Review of Computer Engineering Studies, 10(1): 7-13. https://doi.org/10.18280/rces.100102

[3] Zanuttini, D., Andres, C., Gonzalez, J., Lacapmesure, A., Priano, D., Pucci, N., Bullian, P. (2018). Using Software Defined Networking for Call Admission Control and VoIP applications. In 2018 Congreso Argentino de Ciencias de la Informática y Desarrollos de Investigación (CACIDI), IEEE, Buenos Aires, Argentina, 1-5. https://doi.org/10.1109/CACIDI.2018.8584371

[4] Masek, P., Stusek, M., Krejci, J., Zeman, K., Pokorny, J., Kudlacek, M. (2018). Unleashing full potential of ansible framework: University labs administration. In 2018 22nd conference of open innovations association (FRUCT), Jyvaskyla, Finland, pp. 144-150. https://doi.org/10.23919/FRUCT.2018.8468270

[5] Puppet Infrastruture Automation & Compliance at Entrepise scale, 2023, https://puppet.com/.

[6] Singh, R., Purwar, R.K. (2019). Cloud automation with configuration management using CHEF Tool. International Journal of Engineering Research and Technology (IJERT). https://doi.org/10.17577/IJERTV8IS040166

[7] How Ansible works (2023). Learn the fundamentals of Ansible, powerful IT automation software that emphasizes simplicity and ease of use. https://www.ansible.com/overview/how-ansible-works

[8] Ebert, C., Gallardo, G., Hernantes, J., Serrano, N. (2016). DevOps. IEEE Software, 33(3): 94-100. https://doi.org/10.1109/MS.2016.68

[9] Zhu, L., Bass, L., Champlin-Scharff, G. (2016). DevOps and its practices. IEEE software, 33(3): 32-34. https://doi.org/10.1109/MS.2016.81

[10] Hintsch, J., Görling, C., Turowski, K. (2015). Modularization of software as a service products: A case study of the configuration management tool Puppet. In 2015 International Conference on Enterprise Systems (ES), Basel, Switzerland, pp. 184-191. https://doi.org/10.1109/ES.2015.25

[11] Luchian, E., Filip, C., Rus, A.B., Ivanciu, I.A., Dobrota, V. (2016). Automation of the infrastructure and services for an openstack deployment using chef tool. In 2016 15th RoEduNet Conference: Networking in Education and Research, Bucharest, Romania, pp. 1-5, https://doi.org/10.1109/RoEduNet.2016.7753200

[12] Shah, J.A., Dubaria, D. (2019). NetDevOps: A new era towards networking & DevOps. In 2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), New York, NY, USA, pp. 0775-0779. https://doi.org/10.1109/UEMCON47517.2019.8992969

[13] Palermo, J. (2019). Building Code. In: .NET DevOps for Azure. Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-5343-4_6

[14] Artac, M., Borovssak, T., Di Nitto, E., Guerriero, M., Tamburri, D.A. (2017). DevOps: Introducing infrastructure-as-code. In 2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C), Buenos Aires, Argentina, pp. 497-498. https://doi.org/10.1109/ICSE-C.2017.162

[15] Madden, N. (2022). API security in action. Manning Publisher ISBN: 9781617296024/1617296023

[16] Fantom, W., Alcock, P., Simms, B., Rotsos, C., Race, N. (2022). A NEAT way to test-driven network management. In NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium, Budapest, Hungary, pp. 1-5. https://doi.org/10.1109/NOMS54207.2022.9789909

[17] Salazar-Chacón, G. (2022). Hybrid Networking SDN and SD-WAN: Traditional Network Architectures and Software-Defined Networks Interoperability in digitization era. Journal of Computer Science and Technology, 22(1): e07-e07. https://doi.org/10.24215/16666038.22.e07

[18] Bjorklund, M., Schönwälder, J., Shafer, P., Watsen, K., Wilton, R. (2019). NETCONF extensions to support the network management datastore architecture (No. rfc8526). https://www.rfc-editor.org/rfc/rfc8526.html.

[19] Clemm, A., Medved, J., Varga, R., Bahadur, N., Ananthakrishnan, H., Liu, X. (2018). A yang data model for network topologies. https://www.rfc-editor.org/rfc/rfc8345.

[20] Singh, N., Singh, A., Rawat, V. (2022). Deploying jenkins, ansible and kubernetes to automate continuous integration and continuous deployment pipeline. 2022 IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI), Delhi, India, pp. 1-5. https://doi.org/10.1109/SOLI57430.2022.10294378

[21] Bilder, D. (2018). " Extension Negotiation in the Secure Shell (SSH) Protocol", https://www.rfc-editor.org/rfc/rfc8308.

[22] Albrecht, M.R., Degabriele, J.P., Hansen, T.B., Paterson, K.G. (2016). A surfeit of SSH cipher suites. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 1480-1491. https://doi.org/10.1145/2976749.2978364

[23] Alberto Priano, D., Abeledo, M. C., Guevara, J., Marsicano, M., Sergio Bruschetti, F., Giniger, I. (2023). Comparative analysis of sdn controllers: a study on installation, protocols interaction, network topologies monitoring, and GUI experience. Review of Computer Engineering Studies, 10(3): 41-47. https://doi.org/10.18280/rces.100302

[24] Hochstein, L., Moser, R. (2017). Ansible: Up and running: Automating configuration management and deployment the easy way. O'Reilly Media, Inc. ISBN-13: 978-1491979808, ISBN-10:1491979801

[25] Van der Bent, E., Hage, J., Visser, J., Gousios, G. (2018). How good is your puppet? An empirically defined and validated quality model for puppet. In 2018 IEEE 25th international conference on software analysis, evolution and reengineering (SANER), Campobasso, Italy, pp. 164-174. https://doi.org/10.1109/SANER.2018.8330206

[26] Heinonen, J. (2015). Learning puppet: build intelligent software stacks with the puppet configuration management suite. ISBN-13: 978-1784399832, ISBN-10 1784399833.

[27] Plummer, S., Warden, D. (2016). Puppet: Introduction, Implementation, & the Inevitable Refactoring. In Proceedings of the 2016 ACM SIGUCCS Annual Conference, pp. 131-134. https://doi.org/10.1145/2974927.2974950

[28] Vujović, M., Savić, M., Stefanović, D., Pap, I. (2015). Usage of NGINX and websocket in IoT. In 2015 23rd Telecommunications Forum Telfor (TELFOR), Belgrade, Serbia, pp. 289-292. https://doi.org/10.1109/TELFOR.2015.7377467

[29] Das, D. (2014). Integrating cloud service deployment automation with software-defined environments (Master's thesis), Universität Stuttgart. https://doi.org/10.18419/opus-3297

[30] DeJonghe, D. (2020). Nginx cookbook. O'Reilly Media.

[31] Sabharwal, N., Wadhwa, M. (2014). Developing a cookbook. In: Automation through chef opscode. Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4302-6296-1_9

NOMENCLATURE

CLI	Command Line Interface
DevOps	DevOps Methodology
DSL	Domain-Specific Languaje
ERB	Embedded Ruby
GNS3	Graphical Network Simulator 3 application software.
GUI	Graphical User Interface
IBM	International Business Machines Corporation
IT	Information Technology
Mac OS	Apple Computer´s Macintosh Operating System
NETCONF	Network Configuration Protocol
NetDevOps	Networking operations using DevOp tools
NGINX	Web server application
PFP	Hypertext Scripting Preprocessor
SDN	Software Developed Networks
SSH	Secure Shell Protocol
SU	Linux command
SUDO	Linux command
SysOps	System Operator
VM VirtualBox	Oracle Corporation´s Virtual Machine software
VoIP	Voice over Internet Protocol
WinRM	Windows Remote Management
WS	Workstation
YAML	Configuration languaje
YANG	Data Modeling languaje