ASEC 2025 Conference

The 6th International Electronic Conference on Applied Sciences

09–11 December 2025 | Online

MDPI

Automated Provisioning of FTP Services Using Open-Source Tools: A Comparative Analysis of Shell, Ansible, and Chef

Pedro Escudero-Villa[1]*, Darwin Armijos-Guillen[1], Jenny Paredes-Fierro[1]

1 Facultad de Ingeniería, Universidad Nacional de Chimborazo, Riobamba 060108, Ecuador

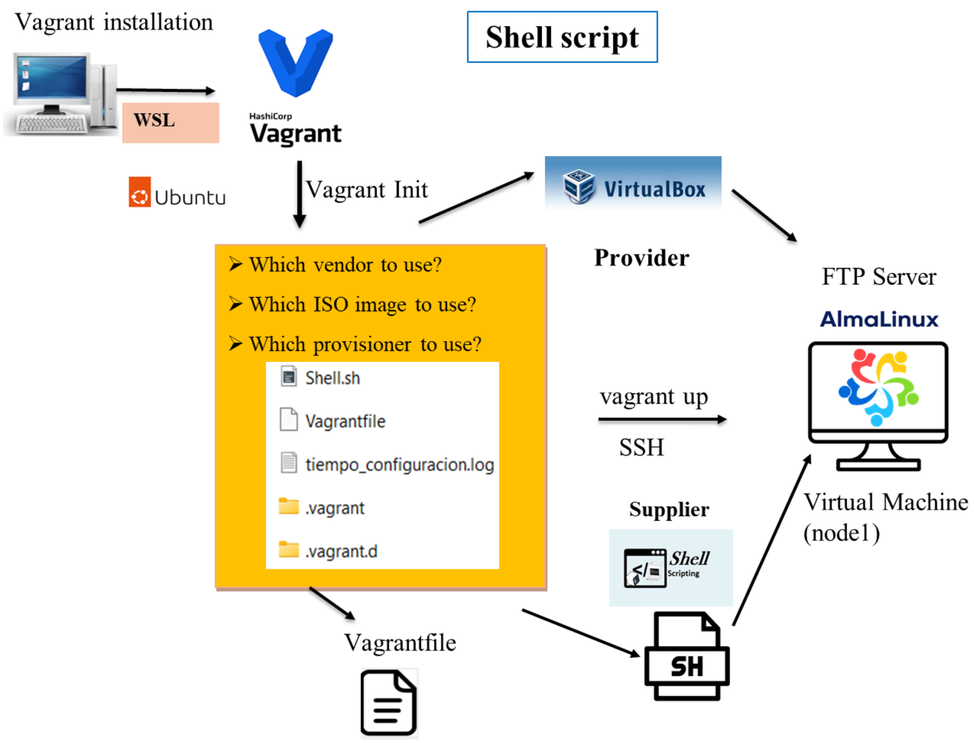
INTRODUCTION & AIM

This work addresses the growing need to optimize server configuration processes, particularly for FTP services, in environments that demand fast, secure, and scalable file transfer solutions. Manual configuration is increasingly inefficient and prone to errors, accounting for up to 60% of FTP security breaches. Infrastructure as Code tools—such as Ansible, Chef, and Shell—have demonstrated their ability to automate deployments, reduce human intervention, and significantly cut configuration time by more than 80%. Despite their widespread use in web and database servers, there is limited research focused on FTP servers. This study proposes developing an automated provisioning system using open-source tools to streamline installation, configuration, and management tasks. The project aims to reduce operational workload, enhance reliability, and evaluate performance gains compared to traditional manual methods.

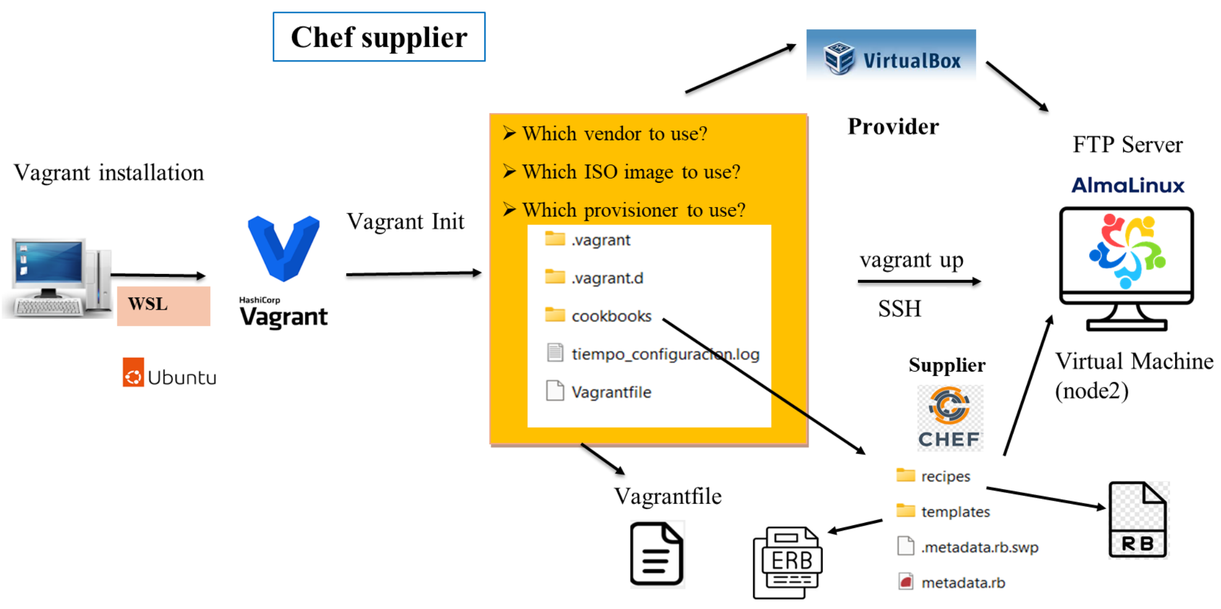
METHOD

Vagrant was used to provision an FTP server on a virtual machine hosted in VirtualBox, incorporating AlmaLinux, Ubuntu, Shell Script, Chef, and Ansible. The provisioning process began with installing Vagrant on Ubuntu or Windows and initializing the environment with vagrant init, which generated configuration files such as the Vagrantfile, Shell.sh, default.rb, vsftpd.conf.erb, metadata.rb, vsftpconfiguracion.yml, and the Ansible hosts file. The Vagrantfile specified VirtualBox as the hypervisor and AlmaLinux as the operating system. Executing vagrant up created and provisioned the virtual machine.

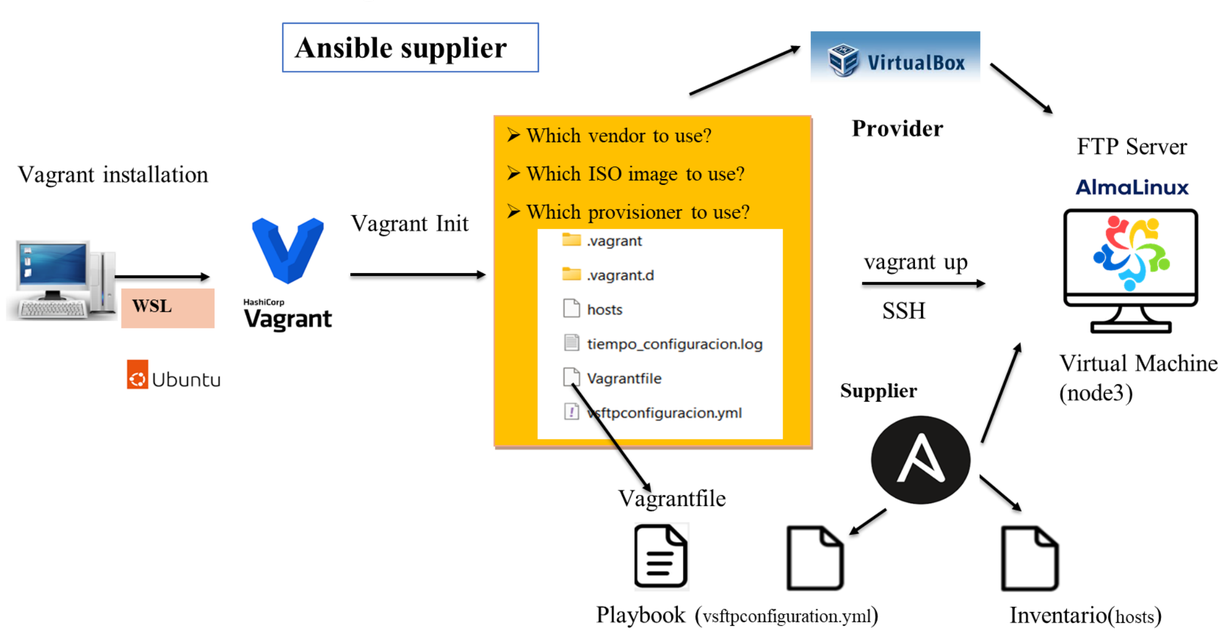
For the Shell case, Shell.sh automated installation and configuration commands.



In Chef, default.rb defined the recipe logic, vsftpd.conf.erb provided the configuration template, and metadata.rb handled dependencies.



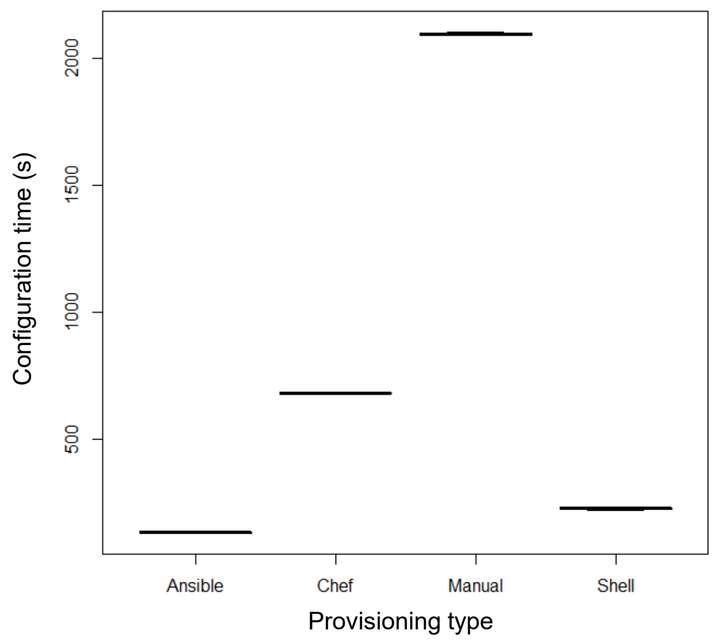
In Ansible, the playbook vsftpconfiguracion.yml and the hosts inventory defined tasks and target servers.



The workflow arrows illustrate how each process originated from WSL, passed through Vagrant, reached VirtualBox, and finalized configuration within the AlmaLinux virtual machine.

RESULTS & DISCUSSION

The results demonstrate a clear reduction in configuration time using automated provisioning tools compared to the manual method. the distribution of execution times, showing that Ansible achieved the lowest and most stable values, followed by Shell Script and Chef.



Using Vagrant with Ansible, Shell Script, and Chef, configuration times decreased from 2073.6 seconds manually to 129 seconds, 209.4 seconds, and 673.2 seconds, respectively. Each tool automated essential tasks—service setup, firewall configuration, and user creation—though default Vagrant credentials required security adjustments.

CONCLUSION

The automated provisioning system significantly reduced FTP server configuration time, proving the efficiency of open-source automation tools. Among the evaluated methods, Ansible delivered the fastest and most consistent performance due to its agentless architecture and YAML-based structure. Automation also minimized human errors and ensured standardized, repeatable configurations across virtualized environments, improving reliability and administrative simplicity. Overall, the system is scalable and suitable for small organizations, offering a strong foundation for future extensions to other services and infrastructures.

FUTURE WORK / REFERENCES

Future work may extend the provisioning system to other network services, improve scalability across multiple virtual machines, integrate monitoring and security automation, and evaluate additional IaC tools. Further testing in real organizational environments could validate performance, reliability, and adaptability for broader infrastructure deployment.