JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

COMPARATIVE ANALYSIS OF NETWORK AUTOMATION AND TESTING FRAMEWORKS: EVALUATING PyTEST, PyATS, ANSIBLE AND ROBOT

[1]Savitri Tangirala, [2]Pragya Sen, [3]Sindhu D V, [4]Saba Farheen. N. S

[1]Under Graduate student, Department of Computer Science and Engineering, RV College of Engineering, Bengaluru, India
[2]Under Graduate student, Department of Electronics and Communications, RV College of Engineering, Bengaluru, India
[3]Assistant Professor, Department of Computer Science and Engineering, RV College of Engineering, Bengaluru, India
[4]Assistant Professor Department of Electronics and Communications, RV College of Engineering, Bengaluru, India

Abstract: Automation plays a pivotal role in reducing manual efforts and improving the efficiency of network testing and configuration. This paper explores various automation frameworks, namely PyTest, PyATS, Ansible, and Robot framework, highlighting their features, advantages, and application areas. PyTest offers a versatile testing framework with extensive support for test case design, parameterization, and execution, facilitating flexible and scalable testing scenarios. PyATS specializes in network testing, providing a structured approach to executing test cases tailored for network devices. Ansible addresses the challenges of configuring multiple servers by leveraging playbooks and SSH logins, enabling quick and consistent deployment across diverse environments. Robot framework, with its user-friendly, keyword-driven approach, simplifies the automation of network device connections and validation tasks. This paper aims to guide practitioners in selecting the most suitable framework based on their automation and support requirements, paving the way for efficient network testing and configuration practices.

Index Terms **- automated testing, network-device testing, test development, PyTest, PyATS, Ansible, Robot**

I. INTRODUCTION

The internet comprises various data packets associated with applications such as VoIP calls, video streaming, and gaming. These packets are further categorized, including VoIP calls being classified as Zoom calls or Google Meet calls. Internet Service Providers (ISPs) facilitate the communication of these packets by purchasing bandwidth from upstream ISPs or other sources. Efficient utilization of acquired bandwidth is essential for ISPs, leading them to implement features like subscriber plan control and policy implementation. These features constantly evolve due to the increasing usage of the internet.

To ensure reliable deployment, ISPs need to automate network configuration and testing processes. As organizations grow, they require numerous servers, each with unique configuration requirements like RAM, CPU, and storage capacity. Scalability becomes crucial, with the need to activate new or dormant servers promptly when demand increases. Manual configuration across a global server infrastructure introduces the risk of errors. Automation becomes invaluable in this scenario. By specifying a script with the necessary server configurations, servers can automatically load their operating systems (OS) and other hardware configurations as specified. This automation technique is equally popular in conjunction with network device configuration and testing automation.

Automation can be achieved through various methods, including network configuration automation and server configuration automation [3]. The chosen approach depends on specific requirements. For instance, the Robot Framework employs a keyword-driven approach where tests are structured using predefined keywords that execute specific tasks. This approach enhances modularity and reusability of test cases. Other widely used techniques include pytest and PyATS platforms [4]. Additionally, Ansible serves as a popular framework for automating server configuration tasks, streamlining the process further [2].

In summary, automation plays a critical role in optimizing bandwidth usage, automating network configuration and testing, and efficiently managing server infrastructure with various configuration requirements. Different approaches, such as keyword-driven frameworks and tools like pytest, PyATS, and Ansible, provide flexibility and scalability in achieving automation objectives based on specific needs.

II. BACKGROUND

When faced with the challenges of manual configuration for a substantial number of network devices or servers, various difficulties can arise, including human errors such as misconfigured settings, typos, and overlooking critical parameters. Additionally,

the time-consuming nature of manual configuration and the potential for inconsistencies can lead to troubleshooting and compatibility issues.

To address the continuous addition of new features or configurations and ensure the functionality of existing ones, regression testing becomes crucial. Regression testing is a type of software testing that aims to verify that previously developed and tested functionalities continue to work as expected after introducing changes or updates [5]. This involves retesting existing features to identify any potential issues or regressions resulting from the changes. In some cases, organizations may require nightly regressions, where testing is performed every night to detect any functionality breakage due to the day's additions. However, achieving near-perfection in manual regression testing is nearly impossible.

Automation provides the ideal solution for overcoming these challenges, eliminating errors, inconsistency, and the need for extensive time investments. Automation also enables regression testing during non-peak hours when resources are readily available. However, automation can take various forms. In this study, four frameworks are considered for comparison: pytest, PyATS, Ansible, and Robot Framework. Each framework offers unique strengths and capabilities. Pytest offers extensibility to various devices, making it highly flexible [6]. PyATS specializes in network automation, providing comprehensive tools and APIs for testing and configuration [1]. Ansible is widely recognized for its strength in server configuration management, simplifying the process with declarative YAML-based scripts [21]. Lastly, Robot Framework stands out for its keyword-driven approach, enhancing modularity and test case reusability [11].

These frameworks have gained significant traction, boasting active community support and abundant resources for effective network and server configuration automation and testing. With these considerations in mind, the study focuses on these four frameworks as representative examples of different automation approaches.

LITERATURE REVIEW

In today's highly interconnected world, the stability and reliability of network devices play a crucial role in ensuring seamless communication and efficient data transfer. As networks continue to evolve and grow in complexity, network device testing and automation have become indispensable tools for network engineers and administrators. In this literature survey, we delve into a comparative study of four popular network device testing and automation frameworks: Pytest, PyATS, Ansible, and Robot.

By exploring the strengths, weaknesses and unique features of these frameworks, we aim to provide a comprehensive understanding on several essential factors to evaluate the frameworks comprehensively. It emphasizes the ease of setup, considering how user-friendly and straightforward the initial configuration process is for each framework. Additionally, it examines the ease of coding, assessing the simplicity and intuitiveness of writing code or test scripts using the frameworks. The study also considers the availability, quantity, and support of documentation, as well as the frameworks' main networking applications. Furthermore, it includes a comparison of the programming languages associated with each framework, evaluating their compatibility and suitability for automation tasks. Supported platforms, including operating systems and network versions, are taken into account to assess the frameworks' versatility. Automation-related mechanisms, such as report generation and assertions for test case success or failure, are examined to determine the frameworks' capabilities in ensuring functional correctness. The study also explores regression-related aspects and the frameworks' support for handling regression testing efficiently. Finally, library support is evaluated to gauge the availability and extensibility of additional functionalities provided by the frameworks' libraries. By conducting a thorough literature survey, we aim to provide a comparative analysis of these frameworks, enabling network professionals to select the most suitable option for their specific testing and automation requirements.

A. PyTest

Pytest is a popular and widely used testing framework for Python, known for its ease of installation and minimal configuration requirements [7]. It can be easily installed with a single command or a few clicks, depending on the operating system chosen. The convention-based approach followed by pytest allows users to get started quickly without the need for extensive setup. By simply running the pytest command, the framework automatically discovers and executes tests based on predefined conventions.

With a primary focus on unit testing, pytest empowers users to leverage their Python coding skills to write effective test cases. It provides a comprehensive set of features and a vast collection of plugins that enhance test organization, parametrization, fixture management, and test discovery [7]. These plugins offer additional functionalities such as test coverage reporting, test parallelization, test data generation, and much more, enabling users to tailor their testing approach to specific project requirements.

One notable advantage of Pytest is its extensive documentation, covering various aspects of the framework. The documentation provides clear instructions for installation, usage, fixture creation and usage, plugin integration, and advanced features [26]. This comprehensive resource allows users, both beginners and experienced developers, to quickly grasp the concepts and leverage PyTest’s capabilities effectively.

Although pytest is primarily focused on unit testing, it can be extended to handle network testing and automation [10]. Being entirely in Python, users can leverage the flexibility and extensibility of the language to create custom network testing scenarios and automation workflows. By combining pytest's test organization features, parametrization capabilities, and fixture management, users can build robust and scalable network testing frameworks tailored to their specific needs [8].

Regarding platform support, pytest is a Python-based framework that is compatible with multiple platforms. It can be seamlessly used on Windows, macOS, and Linux systems, making it a versatile choice for testing projects across different environments.

Additionally, pytest provides support for regression testing through its various features. Users can leverage options for test filtering based on markers, test attributes, or patterns to select and run specific subsets of tests [9]. By organizing test cases into test suites or test modules, users can focus on regression testing specific areas of their codebase. These features enable efficient and targeted regression testing, ensuring that changes or updates do not introduce new issues while maintaining code quality and stability.

In summary, pytest is an easy-to-install, convention-based testing framework with minimal dependencies. Its primary focus is on unit testing, but it offers extensive features and plugins that enhance test organization, parametrization, fixture management, and test discovery. With its comprehensive documentation, pytest enables developers to quickly get started and harness the power of Python for testing purposes. While primarily a unit testing framework, pytest can be extended for network testing and automation. It supports multiple platforms and provides features for efficient regression testing.

B. PyATS

PyATS is a powerful framework specifically designed for network automation and testing. While the installation process may require some effort, it is recommended to set up a virtual environment for seamless integration. To run test cases, configuring testbeds and writing YAML files to define the network setup is necessary. PyATS also has additional dependencies on libraries like Genie, which is used for network automation testing [27].

One of PyATS' s key strengths lies in its focus on network automation and testing. It provides APIs that simplify the interaction with network devices using Python. These APIs enable tasks such as device connections, executing CLI commands, parsing command outputs, and performing various network-specific operations [1]. By leveraging these capabilities, developers can streamline the coding process for network automation tasks.

PyATS offers comprehensive documentation, available on the Cisco DevNet website [27], which covers installation guides, tutorials, sample code, API references, and configuration examples. This wealth of resources assists users in quickly getting started with PyATS and exploring its various features. The documentation serves as a valuable reference for understanding the framework's capabilities and implementing network automation and testing workflows effectively [27].

Specifically designed for network automation and testing, PyATS enables users to automate network operations, validate configurations, execute CLI commands, and perform other network-specific tasks. It provides extensive support for protocol testing, network device interaction, and data validation within network automation workflows. This focus on network-specific features makes PyATS a powerful tool for network engineers and automation testers.

PyATS is entirely implemented in Python, making it easy to integrate into existing Python-based projects and take advantage of the language's flexibility and extensive ecosystem of libraries.

PyATS supports a wide range of platforms, including Cisco devices (such as IOS, IOS-XR, NX-OS, ASA), Juniper devices (JunOS), Arista devices, Linux-based systems, and other network devices with SSH and CLI access [1]. This broad platform support allows users to apply PyATS to diverse network environments and devices.

Validation of automation or testing results in PyATS involves using Python scripting to parse the output of network commands, extract relevant information, and validate if the output meets the expected criteria. This can include tasks like string manipulation, regular expressions, or comparing data structures.

PyATS facilitates regression testing through its features for creating reusable test libraries and test scripts. It also provides reporting capabilities that allow users to track and compare test results across different test runs. These features assist in identifying regression issues and ensuring the stability and reliability of network automation workflows.

In summary, PyATS is a comprehensive Python-based framework designed for network automation and testing. While the installation process may require some setup, it offers extensive support for network-specific tasks. The framework's documentation on the Cisco DevNet [27] website serves as a valuable resource. With its focus on network automation and testing, PyATS provides APIs for network device interaction, validation of configurations, and protocol testing. It supports a wide range of platforms and offers features for regression testing through reusable test libraries, test script creation, and result comparison.

C. Ansible

Ansible, a powerful automation framework, follows a similar setup process to pytest, where installation is dependent on the operating system. However, it may require additional steps for configuring inventory and SSH access to target hosts.

An important distinction of Ansible is its declarative approach, allowing users to define desired configurations using YAML files called playbooks [23]. This approach focuses more on specifying the desired state rather than the procedural execution details.

Ansible provides extensive documentation covering installation instructions, conceptual explanations, modules, playbooks, best practices, and specific use cases with examples [28]. This comprehensive resource assists users in understanding the framework's concepts and effectively utilizing its features.

With Ansible's YAML-based playbooks, users can perform simultaneous configuration of multiple network devices. This includes tasks such as configuring interfaces, VLANs, routing, and more, all defined in YAML format. Ansible leverages its core modules, written in Python, to facilitate configuration management and automation [21].

Ansible is a versatile framework that supports a wide range of platforms. It is compatible with various Linux distributions (e.g., Ubuntu, CentOS, Debian), Unix-like systems (e.g., macOS, FreeBSD), Windows, network devices (Cisco, Juniper, Arista, etc.), cloud platforms (AWS, Azure, GCP, etc.), and container orchestration platforms (Docker, Kubernetes).

In Ansible, conditional statements such as "when" can be used to control the execution of tasks based on specific conditions [22]. Ansible also provides modules and plugins for result validation, allowing users to compare the state of configurations, search for patterns in log files, or perform network tests using modules like "uri" or "ping."

Regression testing in Ansible can be achieved by executing specific playbooks or roles that target relevant areas of the system or network. By rerunning these playbooks during regression testing, users can verify that changes have not introduced unexpected behavior or issues. Ansible also allows users to store and compare facts or variables from different runs, facilitating the identification of regressions and providing insights into the system or network state [25].

In summary, Ansible is an automation framework that simplifies configuration management through its declarative approach using YAML-based playbooks. It offers extensive documentation, covering various aspects of the framework. Ansible supports a wide range of platforms, including Linux distributions, Unix-like systems, Windows, network devices, cloud platforms, and containers. With its modules, plugins, and conditional statements, Ansible enables result validation and offers flexibility in automation workflows. Regression testing is supported through the execution of specific playbooks or roles and the ability to store and compare facts or variables from different runs.

D. Robot Framework

The Robot Framework offers a straightforward installation process, typically requiring a single command. It follows a keyword-driven approach, making it easy to configure test cases with minimal effort. Additional libraries may be required based on specific testing requirements, but the framework provides a rich set of built-in keywords for common testing tasks [11].

With its human-readable and tabular syntax, the Robot Framework allows for easy-to-understand test case development. The framework provides a comprehensive set of built-in keywords for actions, assertions, and data manipulations, reducing the need for extensive coding. This approach simplifies the creation of test cases and enhances readability.

Comprehensive documentation is available for the Robot Framework, covering topics such as installation, syntax, keywords, libraries, and advanced usage [29]. This documentation serves as a valuable resource for both beginners and experienced users, providing guidance on various aspects of the framework.

In addition to general testing capabilities, the Robot Framework offers specific features for interacting with network devices, executing CLI commands, parsing output, and performing network-specific tasks [12]. It provides test libraries and keywords designed for network testing, such as SSH connectivity, sending/receiving network packets, and validating network behaviors. These features enable network automation and testing workflows using the Robot Framework.

The Robot Framework supports a domain-specific language (DSL) for writing test cases and keywords. While the framework itself is implemented in Python, it also offers support for other languages like Java and .NET [14]. This flexibility allows users to leverage their preferred programming language to extend the framework's capabilities.

Platform-wise, the Robot Framework is implemented in Python and supports multiple platforms, including Windows, macOS, Linux, Unix-like systems, network devices (via SSH), web browsers (Chrome, Firefox, etc.), and mobile platforms (Android, iOS). This broad platform support ensures that the framework can be used across various environments and devices [29].

For validation purposes, the Robot Framework provides built-in keywords like Should Be Equal, Should Be True, and Should Contain. These keywords allow users to validate values, conditions, or the presence of specific elements in a straightforward manner [17]. Additionally, users can implement custom keywords in Python or other supported languages to perform more complex validations based on specific requirements.

The Robot Framework enables users to generate detailed reports and logs, facilitating the comparison of test results across different test runs [19]. This reporting feature is valuable in identifying regressions or deviations in test outcomes. Furthermore, the framework's integration with version control systems allows users to track changes and incorporate regression testing as part of their continuous integration and deployment (CI/CD) pipeline [25].

In summary, the Robot Framework offers easy installation, a keyword-driven approach, and a rich set of built-in keywords for testing tasks. It provides comprehensive documentation, including installation guides, syntax explanations, and information about keywords and libraries. The framework supports interaction with network devices, CLI commands, and network-specific tasks. It is implemented in Python and supports various platforms. Validation is achieved using built-in keywords and the ability to implement custom validation logic. The Robot Framework also offers reporting capabilities and supports regression testing as part of CI/CD workflows.

	PyTest	PyATS	Ansible	Robot
Ease of setup	Easy	Difficult	Easy	Easy
Ease of Coding	Python-focused	Python-focused	YAML-based	Keyword-driven
Documentation	Extensive	Comprehensive	Extensive	Comprehensive
Validation	Built-in assert	Result parsing	Result validation	Built-in keywords
Regression	Test filtering	Test comparisons	Playbook execution	Reporting
Supported platforms	Windows, macOS, Linux	Cisco, Juniper, Arista, Linux-based systems	Linux distributions, Unix-like systems, Windows, Network devices, Cloud platforms	Windows, macOS, Linux, Unix-like systems, Network devices, Web browsers, Mobile platforms

Table 1: Analysis of testing tools on various metrics

IV. Conclusion

In conclusion, the comparative analysis highlights that there is no one-size-fits-all solution when it comes to choosing a testing framework for network automation and testing. Each framework has its strengths and limitations, making it crucial to carefully evaluate the specific requirements and preferences of the project. By considering factors such as ease of setup, coding convenience, documentation, networking application focus, programming languages, supported platforms, validation capabilities, and regression support, test engineers can make an informed decision to select the most suitable framework for their needs.

V. Future Scope

In the future, it would be beneficial to expand the scope of this project by including a wider range of tools commonly used in the network automation and testing domain. This comparative study provides a valuable starting point for evaluating frameworks such as pytest, PyATS, ansible, and robot. However, there are other notable tools and frameworks in the industry, such as SaltStack, Chef, Puppet, and Jenkins, which could be considered for inclusion in future research.

Furthermore, exploring additional features and functionalities within the existing frameworks would enhance the comprehensiveness of the study. This could involve investigating advanced test organization techniques, integration with CI/CD pipelines, cloud-native testing capabilities, and performance testing aspects. By broadening the scope and delving deeper into various tools and their unique features, the project can serve as a more comprehensive and insightful resource for test engineers and practitioners in the network automation and testing field.

References

[1] Choi, B. (2021). Python Network Automation Labs: Ansible, pyATS, Docker, and the Twilio API. In Introduction to Python Network Automation: The First Journey (pp. 675-732). Berkeley, CA: Apress.

[2] Wågbrant, S., & Dahlén Radic, V. (2022). Automated Network Configuration : A Comparison Between Ansible, Puppet, and SaltStack for Network Configuration (Dissertation). Retrieved from http://urn.kb.se/resolve?urn=urn:nbn:se:mdh:diva-58886

[3] Neidinger, M. (2021). Python Network Programming Techniques: 50 real-world recipes to automate infrastructure networks and overcome networking challenges with Python. Packt Publishing Ltd.

[4] A. M. Mazin, R. A. Rahman, M. Kassim and A. R. Mahmud, 'Performance analysis on network automation interaction with network devices using python,' in 2021 IEEE 11th IEEE Symposium on Computer Applications Industrial Electronics (ISCAIE), 2021.

[5] Liu, Y., Thurston, Z., Han, A., Nie, P., Gligoric, M., & Legunsen, O. (2023). pytest-inline: An Inline Testing Tool for Python. arXiv preprint arXiv:2305.13486.

[6] Hunt, J. (2019). PyTest Testing Framework. In: Advanced Guide to Python 3 Programming. Undergraduate Topics in Computer Science. Springer, Cham. https://doi.org/10.1007/978-3-030-25943-3_15

[7] Okken, B. (2022). Python Testing with Pytest. United States: Pragmatic Bookshelf.

[8] L. Barbosa and A. Hora, "How and Why Developers Migrate Python Tests," 2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER), Honolulu, HI, USA, 2022, pp. 538-548, doi: 10.1109/SANER53432.2022.00071.

[9] Chou, E., Kennedy, M., & Whaley, M. (2020). Mastering Python Networking: Your one-stop solution to using Python for network automation, programmability, and DevOps. Packt Publishing Ltd.

[10] Pajankar, A. (2017). Python Unit Test Automation: Practical Techniques for Python Developers and Testers. Apress.

[11] Neha S Batni and, Jyoti Shetty, "A Comprehensive Study on Automation using Robot Framework", International Journal of Science and Research (IJSR), Volume 9 Issue 7, July 2020.

[12] Shruti Malve and Pradeep Sharma, "Investigation of Manual and Automation Testing using Assorted Approaches", International Journal of Scientific Research, Vol.5, Issue.2, pp.81-87, April (2017).

[13] Umar, Mubarak Albarka & Chen, Zhanfang, "A Study of Automated Software Testing: Automation Tools and Frameworks", 2019.

[14] Sohail, Shaleeza, "Automation of Network Management with Multidisciplinary Concepts", International Journal of Computer Technology and Applications, 2010.

[15] Godasu Swetha, T. Ramaswamy, S.P.V. Subba Rao, "Automation of Telecom Networks using Robot Framework", International Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277-3878, Volume-7, Issue-5S3, February 2019.

[16] Monica, Sanket N Shettar, Survey on Robot framework, International Journal of Engineering & Research & Technology, 2017.

[17] S. Mwanje, G. Decarreau, C. Mannweiler, M. Naseer-ul-Islam and L. C. Schmelz, "Network management automation in 5G: Challenges and opportunities," 2016 IEEE 27th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC), pp. 1-6, doi: 10.1109/PIMRC.2016.7794614.

[18] Mandara Nagendra, C N Chinnaswamy, Dr. T H Sreenivas, "Robot Framework: A boon for Automation", International Journal of Scientific Development and Research (IJSDR), ISSN: 2455-2631, Volume 3, Issue 11, November 2018.

[19] Dr. G. Ranganathan, "Survey on Robot Process Automation Application in Various Industries", Journal of Electrical Engineering and Automation (EEA)2019, ISSN: 2582-3051, Vol.01/ No. 02, DOI: https://doi.org/10.36548/jeea

[20] Pranav T P, Charan S, Darshan M R, Girish L, "DevOps Methods for Automation of Server Management using Ansible International Journal of Advanced Scientific Innovation" Volume 01 Issue 02, May 2021

[21] Pavel Masek , Martin Stusek, Jan Krejci, Krystof Zeman, Jiri Pokorny and Marek Kudlacek, "Unleashing Full Potential of Ansible Framework: University Labs Administration", Software Engineering Companion (ICSE-C), 2021 IEEE/ACM 39th International Conference on, pp. 497–498, IEEE, 2021.

[22] Mohd Faris Mohd Fuzi, K. Abdullah, Iman Hazwam Abd Halim, "Network Automation using Ansible for EIGRP Network", Journal of Computing Research and Innovation, Volume 05 Issue 03, 20 September 2021

[23] Rafiza Ruslan, "Implementation of Network Automation using Ansible to Configure Routing Protocol in Cisco and Mikrotik Router with Raspberry PI", Jurnal Ilmiah Komputasi, Published 29 June 2020

[24] Sriniketan Mysari; Vaibhav Bejgam, "Continuous Integration and Continuous Deployment Pipeline Automation Using Jenkins Ansible", 2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE)

[25] Mysari, S., & Bejgam, V. (2020, February). Continuous Integration and Continuous Deployment Pipeline Automation Using Jenkins Ansible. In 2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE) (pp. 1-4). IEEE

[26] "pytest." Documentation. n.d. Web. Retrieved from https://docs.pytest.org/en/latest/.

[27] "pyATS." Documentation. n.d. Web. Retrieved from https://pubhub.devnetcloud.com/media/pyats/docs/index.html.

[28] "Ansible." Documentation. n.d. Web. Retrieved from https://docs.ansible.com/.

[29] "Robot Framework." User Guide. n.d. Web. Retrieved from https://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html.