INPG Stack: An automated way of maintaining and testing a network

Marco Martinez
Eastern Switzerland University
of Applied Sciences
Rapperswil, Switzerland
Email: marco.martinez@ost.ch

***Abstract*—The INPG Stack (Infrahub, NUTS, Prometheus and Grafana) provides an automated approach to network testing, offering visualised results for enhanced visibility. The INPG Stack automates the generation and execution of tests for all network devices, thereby addressing a significant challenge in modern network management where manual testing remains prevalent. By reducing the time required for testing and ensuring networks are continuously updated and tested, the INPG Stack enhances the efficiency and reliability of network operations. This system represents a notable advancement over traditional manual methods, offering scalability, maintainability, and improved performance visibility through its automated processes and integration of open-source tools.**

I. Introduction

Networks are a critical component of any business, providing the backbone for effective communications and operations. It is therefore desirable to have stable, scalable and reliable networks in order to ensure seamless functionality and growth. In order to achieve such a stable network, it is necessary to have a robust validation mechanism in place to confirm its reliability. In addition, it is essential to have visibility into the network's performance and operations in order to maintain its stability and address any potential issues in a timely manner.

In the ever-changing business environment, companies have become highly proficient in the construction of resilient networks, with a significant number of them integrating semi-automatic or fully automatic deployment strategies to improve efficiency and reliability. However, the landscape is in a state of constant evolution, and the requirements of networks frequently change, thereby necessitating ongoing adjustments. In spite of these advancements, the majority of companies have yet to implement automatic network testing, instead relying on engineers who utilise tools such as ping or traceroute for manual validation. [1] Despite the presence of monitoring systems to track network performance, changes can still introduce issues that remain undetected until they manifest as problems. Furthermore, components that are not directly altered are sometimes overlooked in the testing process, potentially compromising overall network stability.

II. Related Work

Although automated network testing is a relatively mature field, it has not yet become fully established within the industry. While there are already open-source network testing tools available, such as NUTS[2], Ondatra[3] and ANTA[4], they suffer from significant usability issues. These tools typically require users to maintain an inventory in YAML[5] and manually define the tests to be conducted, which adds further complexity and effort, to the ever-growing complex networks.[6][7] This cumbersome setup makes it difficult for businesses to keep pace with the fast-moving demands of today's market. Furthermore, current solutions often lack visual feedback, which is crucial for quickly identifying and addressing issues. These limitations hinder the widespread adoption of automated network testing and prevent companies from fully leveraging its potential.

III. System Architecture and Implementation

The system is designed to automate as many steps as possible, thereby reducing the barrier to entry. It aims to maintain synchronisation with the actual state of the network, ensuring that all aspects are continuously and comprehensively tested. Routine processes, such as writing the inventory or tests, are automated and maintained up to date, reflecting the network's state.

The solution with the aforementioned desired capabilities is created through the utilisation of Infrahub[8], NUTS, Prometheus[9], and Grafana[10]. The network testing framework NUTS is used as a basis and its usability issues are addressed, facilitating the construction of a robust, scalable, and dependable network testing automation system. This solution offers comprehensive validation mechanisms and extensive visibility into network testing performance and operations, ensuring that businesses can maintain stability and swiftly address any potential issues.

The four main open-source components utilised in the network testing automation system are listed below.

Fig. 1. Architecture with corresponding components

A. Infrahub

Infrahub serves as the source of truth, acting as the inventory of devices and containing all necessary information to provide

a view of the network and what needs to be tested in an environment. Its flexible schema allows for easy extension to meet specific needs.

Infrahub enables the generation of artifacts on any device. These artifacts can be created using either Jinja[11] or Python. This approach allows for the generation of all tests required to ensure the device is fully functional.

```
- test_class: TestNapalmBgpNeighbors
  test_data:
  {% for node in data.InfraBGPSession.edges  %}
    - host: {{ node.node.device.node.name.value }}
      local_id: {{ ns.loopback_ip }}
      local_as: {{ node.node.local_as.node.asn.value }}
      peer: {{ node.node.remote_ip.node.address.ip }}
      remote_as: {{ node.node.remote_as.node.asn.value }}
      is_enabled: {{ 'True' if node.node.status.value == 'active' else 'False' }}
      is_up: {{ 'True' if node.node.status.value == 'active' else 'False' }}
  {% endfor %}
```

Fig. 2. Jinja code to generate NUTS tests in Infrahub

B. NUTS

NUTS, a plugin for the pytest[12] Testing Framework, offers considerable flexibility for network testing. With a comprehensive set of pre-written tests available, NUTS streamlines the initiation process, requiring only the definition of the target devices to be tested.

Infrahub provides the inventory data, which is queried using the nornir-infrahub[13][14] inventory plugin. Currently, the NUTS tests generated by Infrahub are pulled from their API using a Python script, ensuring seamless integration and up-to-date test definitions.

NUTS currently provides support for a variety of tests, including but not limited to, the following: “interface information, BGP neighbours and config drift detection”. [15]

As a pytest plugin, NUTS can be combined with the pytest-prometheus-pushgateway[16] plugin, which enables the collection of the testing metrics and their transmission to a Prometheus instance for further analysis.

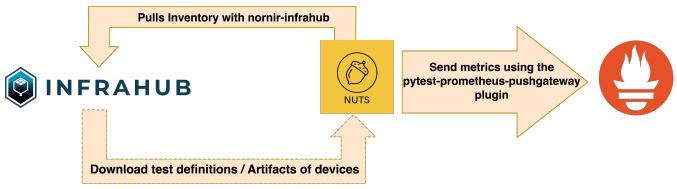

Fig. 3. NUTS interaction with other systems

C. Prometheus

The selection of Prometheus was based on three key factors: its popularity, ease of use, and comprehensive feature set. Its seamless integration with Pytest via the Prometheus Pushgateway enables the efficient collection and monitoring of testing metrics.

It is necessary to implement the Prometheus Pushgateway because the NUTS test run is terminated upon completion, which prevents it from functioning as a queryable endpoint for metrics. Consequently, the approach is to push metrics to the Pushgateway upon termination.

D. Grafana

The selection of Grafana as the visualization option was based on its robust and adaptable capabilities in data visualization and monitoring. Its user-friendly interface and comprehensive plugin ecosystem facilitate the development of dynamic and personalized dashboards, enabling efficient analysis and interpretation of data sets generated by NUTS. Grafana’s seamless integration with Prometheus guarantees real-time visibility into network performance metrics.

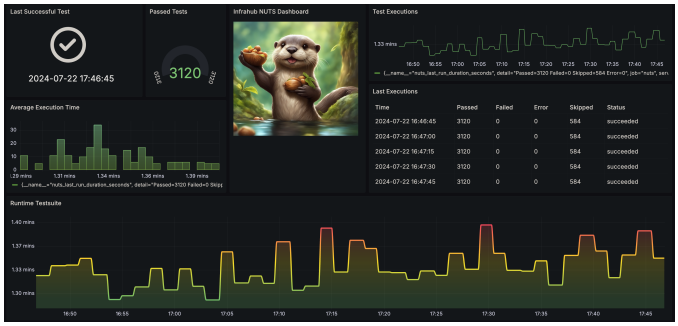

Fig. 4. Grafana Dashboard used to monitor Performance and Events

IV. Discussion, Examples & Demo

In dialogue with network operators, it became apparent that they struggle with four main challenges: (i) maintaining an up-to-date inventory, (ii) ensuring their test suite is in sync with their network, (iii) keeping the network stable and substantiating this stability with metrics, and (iv) providing insights for non-technical stakeholders. Addressing these issues is crucial for effective network management and performance validation.

Therefore, the system has been designed with the objective of being time-friendly and maintainable, as the inventory and test definitions are generated dynamically. The test definitions are crafted only once, using either Jinja or Python, and are then automatically generated for each device. New devices added to Infrahub are automatically included in the testing process, which serves to minimise the barrier to entry and initial effort required. Furthermore, Grafana provides automated visibility into the network, thereby eliminating the need to manually collect metrics or check a CLI, thus streamlining network monitoring and analysis.

Figure 4 illustrates the visibility of a medium-sized datacenter lab. The network topology consisting of 8 spine switches and 16 leaf switches. All devices were registered with their respective information in Infrahub, and the corresponding tests were generated. NUTS was able to execute 3120 tests within 1.33 minutes, based on the four test definition types which were created. This demonstrates the scalability of this approach where within seconds of the lab generation there is a fully functional testing infrastructure present and able to prove the correct working of the network infrastructure. It is also important to consider the non-technical stakeholders, who can use the dashboard to gain insight into the operational status of the network and identify potential issues.

The INPG Stack has been engineered with flexibility as a core design principle, enabling the seamless exchange of all components when required due to its inherent loose coupling. The Stack operates based on a combination of four basic components: a source of truth, a testing framework that integrates with this source for inventory and test generation, a time-series database, and a visualisation framework. This modular approach ensures that the stack can be adapted to various requirements and technologies. For example, Prometheus could be exchanged with InfluxDB[17] without changing the core behaviour of the solution.

For the demonstration, the INPG stack deployment is fully automated by populating infrahub and spinning up a virtual network topology using containerlab from the infrahub artifacts.[18]

V. Conclusion and Future Work

The INPG Stack represents an optimal foundation for environments that currently rely on manual testing, offering a robust basis for the automation and enhancement of network validation processes. It can be integrated with existing testing infrastructure, providing a reliable means of tracking the current state of the network. The repeatability of the process makes it suitable for continuous use, thereby enabling the aggregation of historical data and performance metrics, and facilitating the implementation of alerting mechanisms based on discernible patterns. The generation of NUTS tests is a relatively straightforward process, requiring only the presence of information in Infrahub and the formulation of test definitions in Jinja. The versatility of NUTS allows for the incorporation of new tests with ease, enabling the adaptation of test definitions to meet specific requirements. Overall, the solution remains uncomplicated and scalable, even for larger networks, without increasing complexity.

This solution shows considerable promise, but further testing in production environments is required to fully assess its effectiveness and reliability. A number of options are being considered to determine the best way forward. One possible future enhancement is the improvement of NUTS to facilitate the automatic retrieval of tests directly from an API. This would streamline the process by eliminating the necessity for a Python script to download artifacts as local files. Currently, NUTS has a restricted set of predefined tests; augmenting this collection with a more expansive range of sophisticated tests would facilitate a more comprehensive assessment of diverse networks and their conditions. The built-in version control functionality of Infrahub offers a considerable opportunity for pre-implementation testing of network changes. This may be achieved by deploying a containerlab environment for the purpose of testing configurations, or alternatively by booting up a physical staging environment for the assessment of changes prior to their integration into the main branch. Furthermore, the process of network testing could be integrated into the regular monitoring routine. By scheduling these tests to run as a cron job, continuous validation and real-time monitoring of network performance can be achieved, thus ensuring that any emerging issues are promptly identified and addressed. In addition, the potential for implementing automated alerting based on historical data patterns and metrics collected through Grafana should be investigated. This would enhance the solution's ability to preemptively address potential network issues, ensuring that network operators are informed of any anomalies or performance degradation, and enabling swift corrective action.

In conclusion, although the existing solution is encouraging, the forthcoming enhancements and extended testing in genuine operational contexts will be pivotal to fully actualise its potential and guarantee that it aligns with the evolving requirements of today's network operations.

Acknowledgment

The author would like to thank OpsMill, in particular Damien Garros and Patrick Ogenstad, for providing early access to Infrahub and for the invaluable feedback that made this work possible. The author would also like to thank Urs Baumann for his guidance and insightful feedback throughout the research process.

References

[1] M. Martinez and S. Grimm, "NUTS", 2021, last accessed: 2024-06-18. [Online] Available: https://eprints.ost.ch/id/eprint/1013/

[2] U. Baumann and M. Martinez, "Network Unit Testing System (NUTS)", 2024, last accessed: 2024-06-18. [Online] Available: https://nuts.readthedocs.io/en/latest/

[3] Ondatra, "Ondatra: Open Network Device Automated Test Runner and API ", 2024, last accessed: 2024-06-18. [Online] Available: https://github.com/openconfig/ondatra

[4] Arista Networks, "Arista Network Test Automation (ANTA) Framework", 2024, last accessed: 2024-06-18. [Online] Available: https://anta.arista.com/stable/

[5] O. Ben-Kiki, C. Evans and I. döt Net, "YAML", 2024, last accessed: 2024-06-18. [Online] Available: https://yaml.org/

[6] H. Kim, T. Benson, A. Akella and N. Feamster, "The evolution of network configuration: A tale of two campuses", in: Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference, 2011, pp. 499-514.

[7] S. Lee, T. Wong and H.S. Kim, "To automate or not to automate: on the complexity of network configuration", in: 2008 IEEE International Conference on Communications, IEEE, 2008, pp. 5726-5731.

[8] Opsmill, "Infrahub", 2024, last accessed: 2024-06-18. [Online] Available: https://www.opsmill.com/

[9] The Linux Foundation, "Prometheus", 2024, last accessed: 2024-06-18. [Online] Available: https://prometheus.io/

[10] Grafana Labs, "Grafana", 2024, last accessed: 2024-06-18. [Online] Available: https://grafana.com/

[11] Pallets, "Jinja", 2024, last accessed: 2024-06-23. [Online] Available: https://jinja.palletsprojects.com/en/3.1.x/

[12] H. Krekel and Pytest-dev team, "pytest", 2024, last accessed: 2024-06-23. [Online] Available: https://docs.pytest.org/

[13] OpsMill, "Nornir plugin for Infrahub", 2024, last accessed: 2024-06-23. [Online] Available: https://github.com/opsmill/nornir-infrahub

[14] D. Barroso, "Nornir", 2024, last accessed: 2024-06-23. [Online] Available: https://nornir.readthedocs.io/en/latest/

[15] U. Baumann and M. Martinez, "NUTS Test Bundles", 2024, last accessed: 2024-06-23. [Online] Available: https://nuts.readthedocs.io/en/latest/testbundles/alltestbundles.html

[16] M. Treussart, "pytest-prometheus-pushgateway", 2024, last accessed: 2024-06-23. [Online] Available: https://github.com/treussart/pytest-prometheus-pushgateway

[17] InfluxData Inc, "InfluxDB. It's About Time.", 2024, last accessed: 2024-06-23. [Online] Available: https://www.influxdata.com/

[18] M. Martinez, "INPG Demonstration", 2024, last accessed: 2024-06-23. [Online] Available: https://github.com/marcom4rtinez/INPG_demo