On-Demand Network Management with NMaaS: Network Management as a Service

Vojdan Kjorveziroski
Faculty of Computer Science and Engineering
Ss. Cyril and Methodius University Skopje
Skopje, North Macedonia
vojdan.kjorveziroski@finki.ukim.mk

Pavle V. Vuletic
Department of Computer Science & Information Technology
University of Belgrade
Belgrade, Serbia
pavle.vuletic@etf.bg.ac.rs

Łukasz Łopatowski
Poznan Supercomputing and Networking Center
Poznan, Poland
llopat@man.poznan.pl

Frédéric Loui
RENATER
Paris, France
frederic.loui@renater.fr

***Abstract*—The need for network management is universal, no matter the size of the network. Unfortunately, monitoring is often burdensome for infrastructure teams, because despite catering to their production services, additional supporting systems need to be maintained, continuously updated, and configured. As a result of the ever-growing complexity of new network management solutions, the time required for testing a new tool is increasing, thus disincentivizing exploration of new alternatives. Network Management as a Service (NMaaS) is a software suite which allows teams to centrally manage multiple remote infrastructures through effortless, fast deployment and configuration of network management applications as well as supporting tools. Using the GitOps approach, all application configuration is versioned and automatically synced to running instances, allowing easy migration from existing installations, as well as roll-back of recent configuration changes. In this demonstration we outline the NMaaS architecture, discuss possible use-cases, and showcase the application deployment process together with the steps required for extending the existing application catalog.**

***Keywords*—network management, containers, Kubernetes, GitOps, orchestration**

I. Introduction

All production computer systems, no matter their size, require persistent monitoring and maintenance. The benefits of a comprehensive network management infrastructure are wide-ranging and such setups should offer both proactive and reactive problem detection. Using a proactive monitoring approach [1], potential issues can be detected even before they have an impact on normal operations, thus maintaining the availability of the system. On the other hand, reactive monitoring aims to minimize the incurred downtime once a failure occurs, by detecting anomalies as fast as possible, after they happen. Even though proactive monitoring is the desired approach, not all failures can be anticipated, and reactive problem detection coupled with on-time alerting is just as important.

Nowadays a number of full-fledged monitoring suites exist [2], both commercial and open source, supporting the two strategies discussed above [3]. However, no matter their license or the associated price tag, they all have one thing in common – the need for ongoing maintenance, and regular updates. As with any other software component, monitoring systems also require version upgrades, vulnerability monitoring, configuration changes, and data backup, reminiscent of any production service. Furthermore, the sheer number of available options and their different architectures, makes the selection process for the right tool difficult. Administrators need to provision additional infrastructure such as virtual machines and test devices for each potential solution in which they might be interested, before even being able to install it and determine whether it fulfills their requirements. It is not uncommon for infrastructure teams to not have the required computing resources or manpower to undertake such operations [4], thus disincentivizing the testing of new solutions which might offer a wider feature set compared to the existing ones.

To alleviate these problems, we have developed NMaaS (Network Management as a Service), a software suite which allows easy deployment of applications on the infrastructure where it is installed. Even though the NM in NMaaS stands for Network Management, it is not limited only to such applications, and instead it can be used to deploy any containerized software, no matter its purpose. Utilizing the Kubernetes container orchestrator and adopting a standardized application packaging format, NMaaS allows easy deployment, automatic upgrades, and seamless configuration updates of deployed applications using the GitOps [5] approach. Easily deployable, NMaaS can be installed on any current computing infrastructure and integrated with existing storage and networking systems, providing users with an extensible software catalog. The primary target group of NMaaS are organizations which have many distributed locations, lacking dedicated engineering teams, allowing their limited manpower to be focused on adding value to their core services, instead of infrastructure management and monitoring.

The goal of this paper is to present the underlying NMaaS architecture, the set of applications which are already supported by the NMaaS application catalog, as well as showcase the process of adding new applications. The rest of this paper is organized as follows: in section II we discuss the NMaaS architecture in detail, explaining the various components and their purpose. We then proceed with section III where we explore the use-cases supported by NMaaS and the various configuration modes for deployed applications, before moving to section IV where we discuss the demo content to be showcased.

II. NMaaS Architecture

NMaaS is comprised of 5 different components in total, 4 of which are necessary for its operation, no matter the environment where it is installed. The NMaaS Shibboleth Service Provider (SP) is optional and used only for integration with third-party identity providers (IdP). Fig. 1 provides a visual representation of the NMaaS architecture, including all third-party open-source components required

978-1-6654-0601-7/22/$31.00 ©2022 IEEE

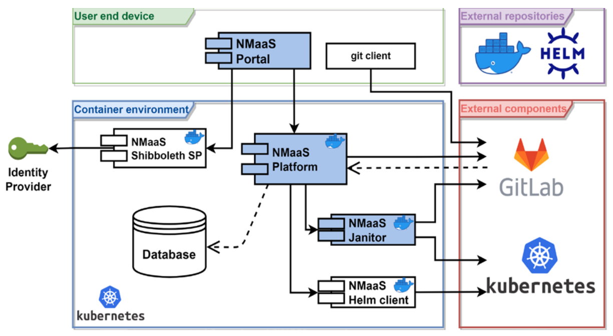

Fig. 1 NMaaS Architecture Diagram

for an NMaaS deployment. In the subsections that follow we discuss each of these components, with the addition of the Kubernetes container orchestrator, which plays a central role in the architecture.

A. *Kubernetes as a Container Orchestrator*

NMaaS leverages the Kubernetes container orchestrator for deployment of the applications present in the catalog on the actual computing nodes. Kubernetes is the most popular container orchestrator today [6] and is suitable for various infrastructures such as: on-premises, cloud, or even on resource constrained edge devices [7]. This approach allows NMaaS to be easily scalable and infrastructure agnostic, capable of integrating with existing storage systems which are compliant with the container storage interface (CSI) [8], as well as existing software defined networking plugins using the container networking interface (CNI) [9]. All core NMaaS components, together with the instantiated applications from the catalog run within the same Kubernetes cluster.

B. *NMaaS Portal*

The NMaaS Portal is the web frontend application which allows end users to login, manage, and configure their existing application instances as well as deploy new ones. It interacts with the NMaaS Platform, the backend, using a representational state transfer (REST) application programming interface (API). It offers a multi-step application instance deployment wizard which provides a visual walkthrough for the users during the deployment process, allowing them to specify context specific parameters and optional configuration. Depending on the assigned user role, it is also possible to leverage the new application wizard available in the Portal in order to add a new application to the catalog. Such an application can later be deployed by any registered user of the given NMaaS instance.

C. *NMaaS Platform*

The NMaaS Platform represents the backend and is the central component of the whole system. It uses a relational database for state management and interacts with the Kubernetes API via the NMaaS Helm Client for application deployment, status monitoring, and management. The REST API exposed by the Platform enables users to programmatically access the system, bypassing the Portal, allowing easy integration with external, third-party, components. Depending on the way in which a given instantiated application manages its configuration, the Platform also interacts with the in-cluster GitLab instance for user syncing and new Git repository provisioning. Each application instance which supports the GitOps configuration workflow has its own GitLab repository. Access to it can also be shared among multiple users.

D. *NMaaS Helm Client*

The NMaaS Helm Client is the component responsible for direct interaction with the underlying Kubernetes cluster. NMaaS uses the concept of Helm charts for application packaging and easy instantiation. Helm [10] is a widely used Kubernetes application manager which allows easy development, templating, and sharing of the necessary Kubernetes manifest files for deploying a given application to an existing cluster. The set of all Helm templates required for instantiating a given application is called a Helm chart. In this way, any software which already has a Helm chart available, either provided by the developers themselves, or by the large Helm open-source community, can be directly added as a supported application to the NMaaS catalog though the new application wizard via the Portal.

E. *NMaaS Janitor*

The NMaaS Janitor is responsible for various maintenance tasks including deletion of left-over volumes and manifests, as well as syncing configuration file changes between the GitLab repositories and the user instantiated applications running in the Kubernetes cluster. Whenever a change is detected within the relevant Git repository for an application, the Janitor updates the associated ConfigMap [11] and Secret [12] objects within the Kubernetes cluster with the contents of the newly pushed files. In this manner, any configuration change made via Git is immediately synced to the running container instance of the application within the cluster.

III. Available Tools and Use-Cases

The default NMaaS catalog available for any new installation of the software comes prepopulated with 26 different applications, primarily centered around network management, but encompassing other areas as well, such as resource scheduling (Booked), inventory and IP address management (NetBox) or even team collaboration (Synapse).

It should also be noted that the list of applications can be customized by each installation independently, to better address the needs of the institution where NMaaS is deployed. NMaaS is being actively developed and additional applications are continuously being added to the default catalog. Additionally, NMaaS also tries to foster a community around it, allowing anyone to submit and then redistribute their application through it.

A. *Installation Options*

NMaaS can be installed by anyone with access to a functioning Kubernetes cluster. All of the NMaaS components can be freely downloaded and used, and we have also prepared additional documentation material helping new users deploy their NMaaS clusters [13]. A ready-to-use development environment in the form of a virtual machine is also available for users simply wanting to try NMaaS out on their workstations, before proceeding with a full-fledged deployment.

B. *Remote Access*

The main use-case of NMaaS is its deployment on a centralized infrastructure which can later be used for monitoring various other distributed locations. A real-world example of such a scenario is a national research and education network (NREN) which is responsible for the provisioning and well-functioning of network equipment in various locations. However, one of the main questions that need to be answered is that of remote access, from the NMaaS infrastructure towards the network elements and vice versa. Even though NMaaS can work with any existing network technology, it supports the concept of client-access and site-to-site VPNs. The site-to-site VPNs are used for establishing secure end-to-end connection to the monitored remote infrastructure, using either a hardware or software-based VPN concentrator. This VPN connection is then used by the deployed applications on NMaaS for monitoring the remote network elements. On the other hand, using the client-access VPNs, NMaaS users are able to access the frontends of their deployed application instances, as well as further configure them. It is up to the administrator to choose the desired connection strategy, and NMaaS components are completely agnostic in that regard.

C. *Application Configuration*

Taking into account the fact that NMaaS supports the inclusion of diverse applications in its catalog, a wide-ranging configuration strategy is required. The goal is to support as many different applications and their configuration as possible, without requiring changes to the underlying NMaaS source code itself. To this effect, the following configuration scenarios are supported: a) Configuration of the instantiated application via the NMaaS Portal – the user is allowed to specify various parameters during deployment time, and these are then passed to additional initialization scripts running within the application's container; b) Configuration using the GitOps approach – all configuration is hosted within a dedicated Git repository and the Janitor component is responsible for syncing any configuration changes to the appropriate Kubernetes objects, and thus the containers where the application itself is being run; c) SSH access to the application container – some applications require elaborate configuration steps which can only be performed using direct console access. In such cases users can directly connect to the container which acts as a runtime environment for their application; d) Application dependent configuration workflow – some applications have elaborate web-based configuration interfaces which can be accessed by users, foregoing the need for a different configuration strategy.

D. *Tenant Isolation*

NMaaS natively supports multi-tenancy. Using the concept of an NMaaS domain, a given NMaaS deployment can be used by multiple organizations, each having their own isolated environment and even different sets of applications in their respective catalogs. Each domain can have an unlimited number of users with various roles, mirroring the real-world hierarchy of the organization. Isolation is ensured both at the application level and the network level, and each tenant can access only their own network elements.

IV. Demonstration

In the proposed demo we plan to discuss NMaaS deployment options, showcasing how the development virtual machine can be used for quick evaluation of the NMaaS software. We would then proceed with the new user registration process on an existing NMaaS deployment, explaining the different supported user roles within a given NMaaS domain. We will also describe the procedure for deploying new application instances using the deployment wizard, and the required steps for configuring the instantiated applications using the GitOps approach. By monitoring a demo network environment, we will present how a network operator can access the web user interface of the deployed applications on NMaaS using a client-access VPN connection. Finally, we will demonstrate the extensibility of NMaaS by adding a previously unsupported application to the NMaaS catalog using the new application wizard directly from the Portal. We will conclude the demo by testing the newly added application and deploying an instance.

Acknowledgment

This work was supported by the GN4 - phase 3 project under GÉANT 2020 Framework Partnership Agreement (FPA). The project is co-funded by the European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 856726 (GN4-3) and National Research and Education Networks in Europe, members of the GN4-3 consortium. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the GN4 - phase 3 consortium or the European Commission.

References

[1] G. Nguyen, S. Dlugolinsky, V. Tran, and Á. López García, 'Deep Learning for Proactive Network Monitoring and Security Protection', *IEEE Access*, vol. 8, pp. 19696–19716, 2020, doi: 10.1109/ACCESS.2020.2968718.

[2] S. P. F. Persis and S. Bindiya, 'A Survey on Open Source Tools - for Server Monitoring using SNMP', *International Journal of Engineering Research & Technology*, vol. 3, no. 15, Apr. 2018.

[3] P.-W. Tsai, C.-W. Tsai, C.-W. Hsu, and C.-S. Yang, 'Network Monitoring in Software-Defined Networking: A Review', *IEEE Systems Journal*, vol. 12, no. 4, pp. 3958–3969, Dec. 2018, doi: 10.1109/JSYST.2018.2798060.

[4] 'IT skills shortage opens chances for growth and diversity', *SearchITOperations*. https://searchitoperations.techtarget.com/feature/IT-skills-shortage-opens-chances-for-growth-and-diversity (accessed Jan. 21, 2022).

[5] T. A. Limoncelli, 'GitOps: A Path to More Self-service IT: IaC + PR = GitOps', *Queue*, vol. 16, no. 3, pp. 13–26, Jun. 2018, doi: 10.1145/3236386.3237207.

[6] '2021 Kubernetes Adoption Survey'. https://www.purestorage.com/content/dam/pdf/en/analyst-reports/ar-portworx-pure-storage-2021-kubernetes-adoption-survey.pdf (accessed Dec. 26, 2021).

[7] S. Böhm and G. Wirtz, 'Profiling Lightweight Container Platforms: MicroK8s and K3s in Comparison to Kubernetes', presented at the 13th Central European Workshop on Services and their Composition, Bamberg, Germany, Mar. 2021.

[8] 'Container Storage Interface (CSI) for Kubernetes GA', *Kubernetes*, Jan. 15, 2019. https://kubernetes.io/blog/2019/01/15/container-storage-interface-ga/ (accessed Dec. 26, 2021).

[9] R. Kumar and M. C. Trivedi, 'Networking Analysis and Performance Comparison of Kubernetes CNI Plugins', in *Advances in Computer, Communication and Computational Sciences*, Singapore, 2021, pp. 99–109. doi: 10.1007/978-981-15-4409-5_9.

[10] 'Helm - The Package Manager for Kubernetes'. https://helm.sh/ (accessed Jan. 18, 2022).

[11] 'Kubernetes ConfigMaps', *Kubernetes*. https://kubernetes.io/docs/concepts/configuration/configmap/ (accessed Jan. 18, 2022).

[12] 'Kubernetes Secrets', *Kubernetes*. https://kubernetes.io/docs/concepts/configuration/secret/ (accessed Jan. 18, 2022).

[13] V. Kjorveziroski and Ł. Łopatowski, 'Local NMaaS Testing Environment - NMaaS - GÉANT federated confluence'. https://wiki.geant.org/display/NMAAS/Local+NMaaS+Testing+Environment (accessed Jan. 19, 2022).