Monitoring solution for cloud-native DevSecOps

Arun Sojan
M3S Research Unit
University of Oulu
Oulu, Finland
arun.sojan@oulu.fi

Ranjit Rajan
Consultant
Panasonic Europe
Frankfurt, Germany
ranjit.rajan@opscentric.de

Pasi Kuvaja
M3S Research Unit
University of Oulu
Oulu, Finland
pasi.kuvaja@oulu.fi

2021 IEEE 6th International Conference on Smart Cloud (SmartCloud) | 978-1-6654-4374-6/21/$31.00 ©2021 IEEE | DOI: 10.1109/SMARTCLOUD52277.2021.00029

***Abstract*—Software development and operations are increasingly adopting cloud-native environments. The popularity of development practices such as DevSecOps is one of the reasons for this change. It is identified that monitoring is one essential practice in DevSecOps and currently, a wide variety of tool offerings are available on the market to address this new transformation. However, an automated monitoring solution that covers both the infrastructure and application level is not available yet. We have developed a repeatable solution based on the popular microservice architectural style that monitors the cloud-native infrastructure and application level to address this gap. Furthermore, we have also added automation capability to this monitoring solution for easy deployment and event-triggered alerting. In the future, we plan to do a detailed evaluation and extend the proposed solution with more data collection features in order to enhance the monitoring solution.**

***Index Terms*—Cloud computing, DevSecOps, Security Monitoring, Cloud Monitoring, DevSecOps Monitoring, DevSecOps Cloud, DevOps Cloud**

I. Introduction

In pursuit of agility and velocity to the market with a security focus, the software teams embrace DevSecOps [1]. Companies favour cloud-native environments for their DevSecOps teams. The need for agility and rapid delivery of secure software based on DevSecOps demands the adoption of a cloud-native environment [2]. Besides, the demand for automation is favourable for cloud-native environments [3]. Cloud adoption will encourage the fast pace of development in DevSecOps with a security focus and avoid costly deployment mistakes [4]. Emerging practices such as "fully-baked" security through immutable machines are some of the recent developments that also favour the use of Infrastructure as Service (IaaS) cloud environments [5]. However, there are some challenges in cloud environments. Cloud systems are much more than standards, policies, remote providers and services [6]. Since the infrastructure itself is dynamic, managing such infrastructures demands constant monitoring in addition to the other considerations such as security and infrastructure governance [3], [6].

The adoption of measurement and automation are two crucial elements in DevSecOps [7]. Implementing the measurement solutions will foster the production first mentality in the DevSecOps teams [3]. However, the adoption of measurement alone without having the automation will cause problems in productivity. First, if the measurement adoption happens without automation, the process will become complicated and the scaling up will slow down the overall process. Secondly, without continuous monitoring, the measurement results will not improve the overall situation of the DevSecOps teams since the results will be delayed [1]. Therefore, the organisation has to take concrete steps to reach the goal of automated measurement. In a cloud-native environment such as Infrastructure as a Service (IaaS), the DevSecOps teams should measure the environment's health status along with the application level monitoring [8]–[10]. There are existing DevSecOps monitoring tools and practices for Infrastructure monitoring, security monitoring and application-level monitoring [11]. However, they are mostly focused on one particular area of monitoring, since the standard interface is not present. It is challenging to integrate them. Furthermore, additional complexity and the learning curve will slow down the DevSecOps teams [3].

In this study, we have developed an automated monitoring solution for the cloud-native DevSecOps. Automated monitoring is gathering, reporting and storing system-related information using automated tools [12]. Our proposed solution is a scalable, automated monitoring framework that will monitor the infrastructure and the application in the cloud-native DevSecOps. This proposed monitoring solution for DevSecOps uses open-source software as its building blocks. The cloud-native teams favour more the open-source solutions due to innovation, portability, flexibility so as to prevent vendor-lock [13]. Furthermore, our solution uses the Microservice Architecture.

The rest of the paper is arranged as follows. Section 2 introduces the background of this study. In section 3, we have related research. In section 4, we introduce our software artefact. Finally, in section 5, we conclude.

II. Background

In this section we will define some of the important concepts related to the study.

A. DevOps

DevOps is a relatively new concept and yet widely popular with software organisations. Despite this popularity, DevOps lacks a clear definition [14]. [15] states that one of the reasons

This research has been funded by the ITEA3 project Oxilate (https://itea3.org/project/ oxilate.html)

978-1-6654-4374-6/21/$31.00 ©2021 IEEE
DOI 10.1109/SmartCloud52277.2021.00029

for this lack of definition is the overlapping of continuous practices. However, we have identified several definitions that vary more on continuous practices aspect alone. Regardless of this difference there are also commonalities. According to [15], "DevOps is a paradigm to reduce the disconnect between development and operations by promoting collaboration, communication, and integration". Defining the term as a paradigm is slightly confusing based on the school of thought of IS development [16]. [12] defines "DevOps as a collection of software engineering activities such as continuous planning, a continuous deployment that are supported by the cultural facilitators such as sharing of responsibility and goals, and technical facilitators such as automated building process and automated configuration management". This definition is based on one of the early definitions of DevOps by [17]. Another definition by [18] defines "DevOps as a set of practices that merges software development (Dev) and information technology operations (Ops) to improve the application delivery process, thereby increasing the frequency of production releases of high-quality software". "DevOps is an approach for software development and (IT) systems operations combining best practices from both such domains to improve the overall quality of software systems while reducing the costs and shortening the time to market", according to [19].

Improved agility, reduction in overall software development and shorter time to market are some of the benefits offered by DevOps. DevOps achieves all these advantages by the conceptual and operational merging of the development and operations needs, teams and technology [20]. The merging ensures coupling between the development and operations teams for software development and software deployment to avoid working in silos [21]. The core concept of the DevOps approach is built on four pillars based on the CAMS principle (Culture, Automation, Measurement, Sharing), later this is further refined and coined as a new term called CALMS (Culture, Automation, Lean, Measurement and Sharing) [22]. The technical implementation of DevOps will result in frequent delivery of applications with new features through Continuous Integration (CI) and Continuous Delivery (CD) pipeline [18]. One of the essential features of DevOps is the automation of the CI and CD pipelines via an automated assembly line [19]. However, DevOps is a difficult concept to adopt due to the availability of vast amounts of information and the lack of collaborative culture [14], [22]. Another challenge is the lack of security focus triggering the creation of DevSecOps by integrating security in DevOps [12].

B. DevSecOps

The rapid development pace in DevOps often caused challenges in integrating security. In addition, the security team often worked in silos compared to the Development and Operations, contradicting the DevOps principles [23]. DevSecOps extends the goal of DevOps by integrating the security practices [20], [24]. Similar to DevOps, the standard definition of DevSecOps is also not yet available. In DevSecOps, the evolution from DevOps even caused different aliases such as "SecDevOps", "DevOpsSec", "Rugged Ops", "SecOps" [25], [26]. In this article, we intend to use DevSecOps for DevOps integrated with security. The DevSecOps promotes shifting the security practices to the left, security-by-design and continuous security testing [8], [20], [27]. The shift left and security-by-design means the security team is included in every step and every iteration of the DevSecOps cycle avoiding silos by close collaboration between Development, Security and Operation teams [26], [28]–[30]. In DevOps, security is given less priority and security practices are often considered at a later stage due to the non-functional nature of the security in most systems [15]. However, adding security to DevOps does not mean a reduction in the rate of development. The security team should do the security practices simultaneously as development and operations [29], [31]. In order to achieve the fast pace of development and operations, DevSecOps practices recommend the use of automation. Achieving security automation is often challenging, making the software development organisations reluctant to transform from DevOps to DevSecOps [32].

Security practices in DevSecOps involve the security of the application and the security of infrastructure from the beginning [1]. This practice makes the selection of the right tools necessary from the initial stage itself. However, the DevSecOps transformation means more than just automation. This transformation also affects other pillars of DevOps such as Culture, Measurement and Sharing of knowledge based on the CAMS principle [1], [29], [30]. Furthermore, this change will improve the close collaboration between development, operations and security teams besides the tools, practices and strategies in an organisation [12], [23]. This cross-functional team involvement is challenging and automation and continuous testing are vital for iterative, frequent, repeatable, and reliable unified security practice in DevSecOps [6], [33]. With optimal strategies, the DevSecOps approach has the potential to deliver secure and high-quality features without compromising on time [28].

C. Toolchains in DevOps and DevSecOps

The implementation of DevOps and DevSecOps is achieved using tools [19], [30]. These tools are used in every stage, such as managing the development and operations environment and testing and releasing [19]. The combination tools adopted for the high degree of automation are called toolchains. Even though these tools have automation capability, each tool has its specific function. For example, there are separate tools available for functional requirements and non-functional requirements in addition to the constraints, rules and practices [34]. Further, the infrastructure plays a significant role in the success of DevSecOps and DevOps. For example, tools that have a shared interface between the development and security team to keep track of vulnerabilities play a reasonable role in improving the culture from the CAMS principle [30]. More information on a detailed list of tools used by the DevSecOps and DevOps teams is explained in [11]. The reasons for this open-source tools favouritism are the need to foster innovation,

prevent vendor-lock and support maximum interoperability [13].

D. Cloud native DevSecOps

DevSecOps and DevOps have a customer centrist view bringing the product to the customer faster. This fast-paced development requires the fast deployment of a production environment such as in the cloud [3], [5]. Besides, the automation of infrastructure provisioning and configuration management features provided by the cloud infrastructure benefit the DevSecOps and DevOps software development at different stages [2], [27]. On the other hand, the lack of automation and provisioning features will hinder the fast-paced development in non-cloud environments. The cloud systems also offer encryption, key management, privileged identity management and provisions to centralize security monitoring [3]. In DevSecOps, such security features will help in different stages of development. Another emerging practice related to DevSecOps in cloud infrastructure is the utilization of immutable virtual machines with baked-in security. These immutable virtual machines are easy to provision on a cloud and can be discarded when they become old [5]. Furthermore, cloud computing offers the high availability, lower cost, security and scalability required for the DevSecOps [4], [6], [13]. The combination of all these factors makes cloud-native environment ideal for DevSecOps.

E. Microservices architecture (MSA)

Microservice Architecture (MSA) is one of the prominent architectural styles used by the service-oriented software industry, such as cloud [35]. [36] defines Microservices as "a cohesive independent process interacting via messages" and microservice architecture as "a distributed application where all its modules are microservices". The key benefit of microservice architecture style is the independence of individual service compared to most service-oriented architecture solutions [37]. Furthermore, this architectural style helps in building reusable, modular component designs. The DevOps agility requirements can be satisfied with the microservice due to the modularity and scalability of the microservices [38]. Combined with the cloud, since it is a cloud-native architecture, the microservices architecture style can develop efficient solutions for DevOps. Continuous Deployment is one of the best examples of the use of a microservice in DevOps. With the increase in deployable units, microservice is one of the most effective ways to build CD pipelines. In this study, we are using the microservice architectural style for the monitoring solution. The use of MSA in DevOps is mapped systematically by [39] in their study.

III. Related literature

A. Need for monitoring and measurement in DevSecOps

Measurement is a fundamental software engineering practice [40]. DevSecOps encourages measurement and system matrix [29]. According to CAMS, measurement is one of the core practices in DevSecOps. Accurate measurement is crucial for understanding the system quality attributes and improving the work practices to help practitioners and researchers [40]. In addition, the security focus is one of the critical aspects of DevSecOps. Using the monitoring and metrics, the team can detect security threats and vulnerabilities [29]. Software measurements usually need tools, and it is the same for the case of DevSecOps [40]. Identification of system problems such as security threats via measurements can significantly improve the cost and time in DevSecOps [29]. DevSecOps teams are usually cross-functional teams [41]. The team members need a lot of different skills to solve the challenges throughout the development life-cycle. These people with measurement skills have an important role [2]. These measurement resources are expected to have the math skills to do the measurement, monitoring and performance analysis of team members, non-human resources and software systems in addition to knowledge sharing skills [2], [8]. Including appropriate measuring techniques will foster the production mindset to the culture, improving overall efficiency [3]. Finally, quality is often equated with measuring: this is evident from the importance of measurement from initiatives such as SW-CMM (Capability maturity model of CMM), ISO/IEC 15504 (SPICE software process improvement and capability determination) and CMMI (Capability maturity model integration) [40].

B. Automation of monitoring and measurement in DevSecOps

Manual measurement is time-consuming and also can cause bottleneck issues for the teams. In the DevSecOps approach, automation practice is considered necessary to minimize manual work and reduce bottleneck issues. Automation of continuous monitoring of the application and overall health of infrastructure is considered best practice in DevSecOps measurement [1]. In addition, the software systems can become quite complex over time, and it is virtually impossible to check the metrics manually; this includes information such as logging data generated by the software. The situation becomes much worse in collaboration since members may not be aware of each other's situations [23]. Furthermore, in a cloud environment such as Infrastructure as a Service (IaaS) models, the number of deployed systems will be too high to monitor manually, and the monitoring solution should be capable of handling scalability [2], [5].

C. Existing solutions

There are some existing solutions for cloud-native environments. [11] provides a comprehensive list of tools in this area and some of the existing DevSecOps tools that can be extended for monitoring and measurement. However, these tools are primarily for a single category of monitoring, such as infrastructure monitoring, security monitoring and logging tools. There is also no comprehensive approach to integrate and replicate the system. Additionally, there are also limitations of visualizations techniques and centralized control. [33] demonstrates a rule-based business engine for the application security level vulnerabilities. However, this solution does not

consider infrastructure monitoring. [24] states that there is a lack of DevSecOps tools and mature DevSecOps solutions.

IV. DevSecOps cloud monitoring solution

This section will first explain the overall concepts of the monitoring solution, then the open-source components used in this solution.

A. Overview of the solution

DevSecOps teams usually consist of three groups: Development, Operations and Security teams. All these teams will interact with the cloud-native infrastructure, as shown in Figure 1.

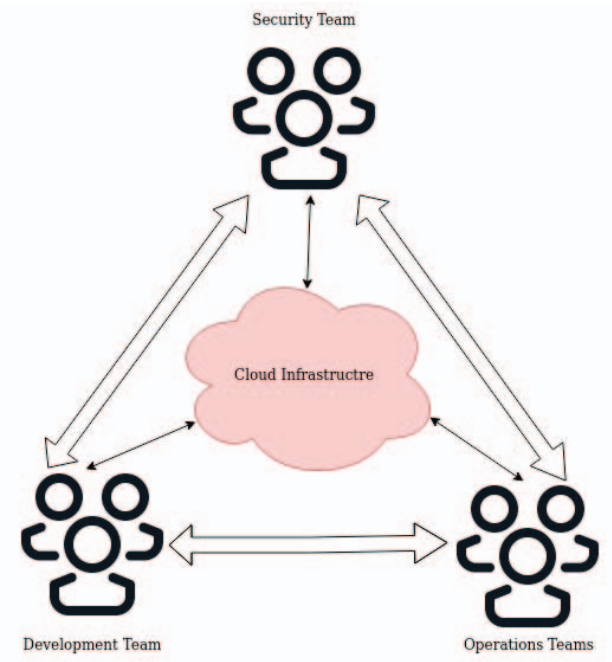

Fig. 1. Cloud interaction with the DevSecOps teams

The monitoring system will monitor two aspects

1) Infrastructure monitoring (cloud monitoring)
2) Application monitoring

The infrastructure part of the monitoring solution will monitor the health of the infrastructure. This infrastructure monitoring is vital for the Operations, Development and Security teams in DevSecOps since they all work in the same environment. The infrastructure monitoring service will monitor the general status of the different deployed environments. Furthermore, users can also select a particular deployed virtual machine based on their requirements. For example, the operations team can see the production server or the development team can see their development servers according to their requirements. Every team member can see the system's overall status and compartmentalization of teams can be avoided, which will facilitate better communication. The infrastructure monitoring will provide four values based on the time and it can also further expand these values for additional information with a click of the button. The general four values are:

1) System health (System is online or not)
2) CPU Rate (The average CPU usage)
3) File system usage (Usage of active disk space)
4) Network Usage (The network activity)

The application monitoring solution shall monitor activities related to the application developed using the DevSecOps practices. Primarily, it shall show all the monitored aspects on a Dashboard. The application monitoring will monitor the continuous integration and continuous deployment pipeline. In addition, for security monitoring an open-source package has been included that can identify the location information of the users from the IP address. The application monitoring solution provides a framework that can be expanded based on the project requirement. The decision was to keep it in this format to ensure that the solution shall stay generic. The monitored data is based on the collected data from data agents installed on the worker nodes.

B. Components of the monitoring solution

In the DevSecOps cloud monitoring, a toolchain was developed based on combining several existing open-source projects. The DevOps community is known for many open source solutions favouring agility, reliability and improved security [5], [42]. The DevOps teams widely use open source tools such as Jenkins, Chef, Ansible, Puppet, Logstash [2], [13]. One of the key practices in DevSecOps is automation. The need for specialized tools is high in automation and the open-source community is addressing this gap effectively [12], [42]. The proposed solution at this stage is primarily aimed at the operations teams after the deployment phase with a focus on security. However, we plan to extend this method to the development teams in the future. The method is designed in a highly extensible format to accommodate future requirements related to the DevSecOps cloud monitoring. For this study, open-source cloud solution was used for Infrastructure as a Service (IaaS) called OpenStack which is also used in similar DevSecOps deployments [25].

We have designed the monitoring solution based on the MSA. The high-level architecture of the system is shown in Figure 2. There are three types of users based on the teams in DevSecOps. Each user shall have a unified dashboard. This unified view approach is beneficial in cross-functional teams. There shall be one main Docker container that has all the critical components. In Figure 2, this container is named the Docker manager, and all the other machines that need monitoring will act as Docker worker nodes. We have used MSA-based architecture using Docker to ensure scalability and easier control of the solution. The team access shall be granted access to the Grafical interface of the manager node. Furthermore, there is a certain level of reliability and it is possible to migrate the manager to another node and reduce the downtime in the event of failure.

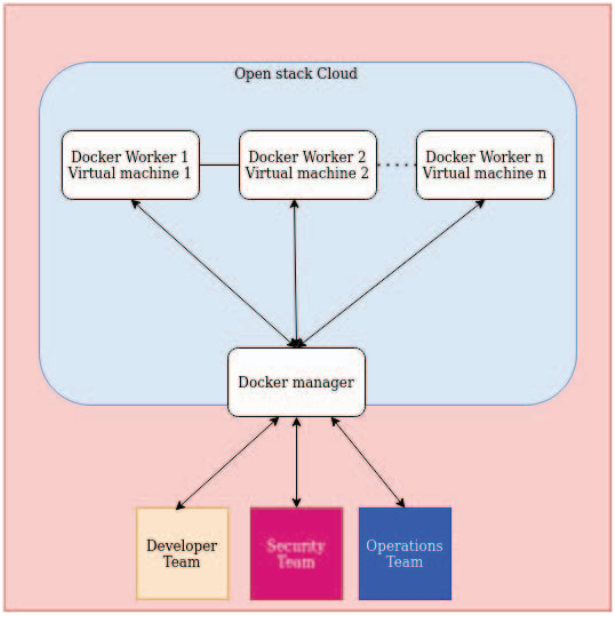

Fig. 2. MSA of the system

The Microservice Architecture Style (MSA) is considered best practice in DevSecOps [7]. In addition, the cloud computing environment demands scalability and the MSA style has improved scalability significantly compared to the monolithic architecture. A simple application is easy to develop using the monolithic architecture style; however, as the complexity increases, the MSA style is more favourable due to the loose coupling. Furthermore, each component in the microservice can be given a more specialized set of tasks [8]. In the DevSecOps environment, these factors promote efficiency and agility. In a distributed systems environment like the cloud, separating individual tasks through specialization will help develop well-defined interfaces, further promoting automation and flexible adoption of the software systems [31]. At the moment, several MSA deployment technologies are in use by several vendors, such as serverless microservice using functions called Amazon Lambda and container-based deployment called Docker [9]. The selection of this deployment technology depends on vendor support and performance requirements. This study has selected container-based MSA deployment using Docker since it is an open-source software and can be easily deployed in almost any cloud environment.

We have deployed Docker in the Linux environment, installed it on the OpenStack cloud. Docker is an open-source container-based virtualisation platform used to provision multiple applications over shared physical hosts in lighter form compared to virtual machines [26]. The standard virtual machines are low-level abstractions that virtualise hardware and requires a full-blown installation of the guest operating systems. Compared to regular virtual machines, the Docker is a lightweight system that virtualise the operating system and eliminates the need for a secondary operating system. Docker is a developer-friendly technology [10]. Docker works based on the container images. The Docker images are packages of all the necessary files such as the application itself, libraries, middleware and network configurations in a layered structure using a union file system (UFS) [10], [26]. Docker platform has a container orchestration mechanism Docker swarm in order to manage the multiple containers running in distributed environments such as cloud. This feature has increased the scalability of the Docker platform. The orchestration helps to have centralised control over containers that help in the distributed environment. Orchestration is yet another feature that increased the Docker platform's desirability in the agile development environments [18].

Traefik is an open-source solution that supports routing, reverse proxy and load balancing [33]. Together with the reverse proxy and the load balancer, Traefik will hide the internal cloud services from the external world and provide better security and refined control. One of the challenges of running a Docker swarm is the reservation of ports. For example, if port 3000 is used by Grafana running on a Docker swarm it will reserve the port 3000 in every cloud machine running the swarm even if it is not running in that specific cloud machine service. Furthermore, as the number of services running in the Docker swarm increases, it will become challenging to manage all the ports in the machine. Traefik, in such a case will also act as a router; based on the received connection request, it forwards the connection to specific endpoints in the system. In addition, there is a middleware layer that will help to extend the capabilities of Traefik. We have used it for internal service authentication to enable better security.

ELK Stack is used for collecting the application level log information in this study. ELK Stack is an open-source solution used for data gathering and analysis. ELK Stack is made up of different components such as Elasticsearch, Logstash and Kibana [40]. This study used Elasticsearch and Logstash along with the data shipper Filebeat. Filebeat is a data shipper to collect information from various sources and send it to Logstash [43]. We have installed Filebeat in every cloud node to collect log related information at the application level. These collected logs are sent to the log stash for transforming it into the format supported by Elasticsearch. Elasticsearch is an analytic engine and also does text-based search. Elasticsearch will help to store and ship the large amounts of data provided by Logstash. One of the primary benefits of using Elasticsearch is generating search results in near real-time. Grafana [24] which is used for visualization retrieves respective metric data from Elasticsearch using its API.

For infrastructure monitoring, an open-source solution called Node Exporter was used. The Node Exporter is a collector agent that collects system-level information such as CPU usage rate, memory usage in a time-series format [40]. Based on this information from the collector agent, the system health can be evaluated. The Node Exporter itself does not have storage capability. It will collect the information and then

make it available over an endpoint for time-series database such as Prometheus to scrape the metrics [40]. Prometheus is an open-source Time Series Database (TSDB) solution. Prometheus gets the data in two models, the pull-based data model and the push-based data model. Prometheus scrapes data from its added data source using the PromQL language [27]. To get the data from Node Exporter, Prometheus uses the pull-based data model.

Grafana is an open-source visualization solution with an interactive dashboard. Grafana can consume data from different data sources such as InfluxDB, Elasticsearch and OpenTSDB and display them using different widgets [24]. Grafana is a resource-efficient tool with several alerting and notification capabilities. In addition, there are several time-series ready templates available in Grafana, making it easier to use. In our study, inputs from two different data sources was populated in Grafana. The data from Prometheus gives system-level information, the data from Elasticsearch is used for displaying information from each of the applications running on respective systems. Grafana dashboard is created dynamically based on the data in real-time. Furthermore, we have also used the alerting and notification capabilities of Grafana to automate the notifications related to the system monitoring.

Many of the automation and the alerting systems are handled by a Python application which was developed to cater the purpose of automation. The Python application provides a user interface for adding emails to the alert monitor. In the event of an alert, the necessary information shall be sent to the configured email addresses. This application also allows adding granular information such as information related to specific nodes, alert events, and details that need to be sent to the email addresses. This application also does API calls to Elasticsearch for data retrieval which are not readily available from Grafana. Any persistent data for the application is stored in Postgresql.

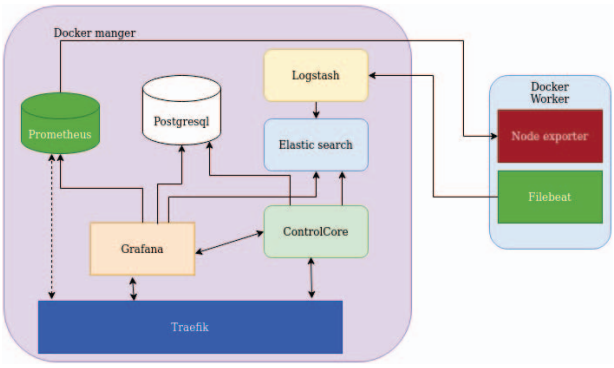

Fig. 3. Components in the Docker stack

Figure 3 shows the different components deployed using Docker and their interfaces in the monitoring solution. We have used agent-based monitoring. The data collection is based on the agents running on the worker node. The worker node has two agents, Node Exporter and Filebeat. Node Exporter shall expose the system level monitoring information, Filebeat collects the data from application logs and other text-based log files. The information from Node Exporter is stored in Prometheus time-series database. This stored information is constantly retrieved by Grafana and stored in its database for visualization. The data exposed by Filebeat will be passed through logstash and stored in Elasticsearch. The data from Elasticsearch is retrieved by Grafana to populate the application level information. Finally, all the external connections shall be handled by Traefik. The primary advantage of using Traefik is security. In addition, Traefik also solves the problem of port allocation in Docker. For example, in Docker swarm, if a Docker service is exposing one port to an external network other than the internal Docker network, this port will be reserved in all the worker nodes. This is particularly challenging when Docker is used in the DevSecOps environment. Furthermore, Traefik has authentication middleware that can be used for additional security.

V. Conclusion and Future Work

Cloud-native DevSecOps monitoring is challenging. There are some tools and practices for cloud-native DevSecOps monitoring. However, they only focus on some of the aspects of monitoring. This results in a lack of comprehensive DevSecOps monitoring solutions. The study addressed this gap by developing a novel monitoring solution for infrastructure and application as an integrated solution. The proposed solution was based on the popular open source solutions that had been already used by DevSecOps community. This study has several limitations. Firstly, we did not present an evaluation based on the proposed solution. In the future, there is a plan to address this limitation by doing a detailed evaluation based on an experimental case. Secondly, the developed solution only considered an agent-based data collection, which shall be iterated with more data collection features in our future study.

Acknowledgment

We want to thank Companies Cybene automation and Opscentric for their valuable contributions. We also would like to thank Polina and Olli-Pekka for their valuable comments.

References

[1] M. Zaydi and B. Nassereddine, "Devsecops practices for an agile and secure it service management," *Journal of Management Information and Decision Sciences*, vol. 23, no. 2, pp. 1–16, 2020.

[2] B. B. N. de França, H. Jeronimo, and G. H. Travassos, "Characterizing devops by hearing multiple voices," in *Proceedings of the 30th Brazilian Symposium on Software Engineering*, ser. SBES '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 53–62. [Online]. Available: https://doi.org/10.1145/2973839.2973845

[3] W. van der Houven MSc, "Security principles for devops and cloud."

[4] B. Somoskői, S. Spahr, E. Rios, O. Ripolles, J. Dominiak, T. Cserveny, P. Bálint, P. Matthews, E. Iturbe, and V. Muntés-Mulero, "Airline application security in the digital economy: Tackling security challenges for distributed applications in lufthansa systems," in *Digitalization Cases*. Springer, 2019, pp. 35–58.

[5] N. Wilde, B. Eddy, K. Patel, N. Cooper, V. Gamboa, B. Mishra, and K. Shah, "Security for devops deployment processes: Defenses risks research directions," *International Journal of Software Engineering & Applications*, vol. 7, no. 6, pp. 01–16, 2016.

[6] S. Carturan and D. Goya, "Major challenges of systems-of-systems with cloud and devops – a financial experience report," in *2019 IEEE/ACM 7th International Workshop on Software Engineering for Systems-of-Systems (SESoS) and 13th Workshop on Distributed Software Development, Software Ecosystems and Systems-of-Systems (WDES)*, 2019, pp. 10–17.

[7] M. Sánchez-Gordón and R. Colomo-Palacios, "Security as culture: A systematic literature review of devsecops," in *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, ser. ICSEW'20. New York, NY, USA: Association for Computing Machinery, 2020, p. 266–269. [Online]. Available: https://doi.org/10.1145/3387940.3392233

[8] N. Tomas, J. Li, and H. Huang, "An empirical study on culture, automation, measurement, and sharing of devsecops," in *2019 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*, 2019, pp. 1–8.

[9] J. Díaz, J. E. Pérez, M. A. Lopez-Peña, G. A. Mena, and A. Yagüe, "Self-service cybersecurity monitoring as enabler for devsecops," *IEEE Access*, vol. 7, pp. 100 283–100 295, 2019.

[10] J. Morales, R. Turner, S. Miller, P. Capell, P. Place, and D. J. Shepard, "Guide to implementing devsecops for a system of systems in highly regulated environments," CARNEGIE-MELLON UNIV PITTSBURGH PA, Tech. Rep., 2020.

[11] V. Mohan and L. B. Othmane, "Secdevops: Is it a marketing buzzword? - mapping research on security in devops," in *2016 11th International Conference on Availability, Reliability and Security (ARES)*, 2016, pp. 542–547.

[12] A. A. U. Rahman and L. Williams, "Software security in devops: Synthesizing practitioners' perceptions and practices," in *2016 IEEE/ACM International Workshop on Continuous Software Evolution and Delivery (CSED)*, 2016, pp. 70–76.

[13] S. Cash, V. Jain, L. Jiang, A. Karve, J. Kidambi, M. Lyons, T. Mathews, S. Mullen, M. Mulsow, and N. Patel, "Managed infrastructure with ibm cloud openstack services," *IBM Journal of Research and Development*, vol. 60, no. 2-3, pp. 6:1–6:12, 2016.

[14] S. Rafi, W. Yu, and M. A. Akbar, "Rmdevops: A road map for improvement in devops activities in context of software organizations," in *Proceedings of the Evaluation and Assessment in Software Engineering*, ser. EASE '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 413–418. [Online]. Available: https://doi.org/10.1145/3383219.3383278

[15] R. N. Rajapakse, M. Zahedi, M. A. Babar, and H. Shen, "Challenges and solutions when adopting devsecops: A systematic review," 2021.

[16] J. Iivari, "A paradigmatic analysis of contemporary schools of is development," *European Journal of Information Systems*, vol. 1, no. 4, pp. 249–272, 1991.

[17] A. Dyck, R. Penners, and H. Lichter, "Towards definitions for release engineering and devops," in *2015 IEEE/ACM 3rd International Workshop on Release Engineering*, 2015, pp. 3–3.

[18] A. Sen, "Devops, devsecops, aiops-paradigms to it operations," in *Evolving Technologies for Computing, Communication and Smart World*. Springer, 2021, pp. 211–221.

[19] A. Capizzi, S. Distefano, and M. Mazzara, "From devops to devdataops: data management in devops processes," in *International Workshop on Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment*. Springer, 2019, pp. 52–62.

[20] H. Myrbakken and R. Colomo-Palacios, "Devsecops: a multivocal literature review," in *International Conference on Software Process Improvement and Capability Determination*. Springer, 2017, pp. 17–29.

[21] K. Carter, "Francois raynaud on devsecops," *IEEE Software*, vol. 34, no. 5, pp. 93–96, 2017.

[22] K. Maroukian and S. R. Gulliver, "Leading devops practice and principle adoption," *9th International Conference on Information Technology Convergence and Services (ITCSE 2020)*, May 2020. [Online]. Available: http://dx.doi.org/10.5121/csit.2020.100504

[23] D. Ashenden and G. Ollis, "Putting the sec in devsecops: Using social practice theory to improve secure software development," in *New Security Paradigms Workshop 2020*, ser. NSPW '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 34–44. [Online]. Available: https://doi.org/10.1145/3442167.3442178

[24] R. Mao, H. Zhang, Q. Dai, H. Huang, G. Rong, H. Shen, L. Chen, and K. Lu, "Preliminary findings about devsecops from grey literature," in *2020 IEEE 20th International Conference on Software Quality, Reliability and Security (QRS)*, 2020, pp. 450–457.

[25] C. Izurieta and M. Prouty, "Leveraging secdevops to tackle the technical debt associated with cybersecurity attack tactics," in *2019 IEEE/ACM International Conference on Technical Debt (TechDebt)*, 2019, pp. 33–37.

[26] S. Wrycza and J. Maślankowski, *Information Systems: Research, Development, Applications, Education: 12th SIGSAND/PLAIS EuroSymposium 2019, Gdansk, Poland, September 19, 2019, Proceedings*. Springer Nature, 2019, vol. 359.

[27] R. Kumar and R. Goyal, "Modeling continuous security: A conceptual model for automated devsecops using open-source software over cloud (adoc)," *Computers & Security*, vol. 97, p. 101967, 2020.

[28] J. S. Lee, "The devsecops and agency theory," in *2018 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, 2018, pp. 243–244.

[29] B. Jammeh, "Devsecops: Security expertise a key to automated testing in ci/cd pipeline."

[30] N. Tomas, J. Li, and H. Huang, "An empirical study on culture, automation, measurement, and sharing of devsecops," in *2019 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*, 2019, pp. 1–8.

[31] Z. Ahmed and S. C. Francis, "Integrating security with devsecops: Techniques and challenges," in *2019 International Conference on Digitization (ICD)*, 2019, pp. 178–182.

[32] V. Mohan, L. ben Othmane, and A. Kres, "Bp: Security concerns and best practices for automation of software deployment processes: An industrial case study," in *2018 IEEE Cybersecurity Development (SecDev)*, 2018, pp. 21–28.

[33] Y. Rouf, J. Mukherjee, M. Fokaefs, M. Shtern, J. Le, and M. Litoiu, "Rule-based security management system for data-intensive applications," in *Proceedings of the 29th Annual International Conference on Computer Science and Software Engineering*, ser. CASCON '19. USA: IBM Corp., 2019, p. 254–263.

[34] S. Rafi, W. Yu, and M. A. Akbar, "Rmdevops: A road map for improvement in devops activities in context of software organizations," in *Proceedings of the Evaluation and Assessment in Software Engineering*, ser. EASE '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 413–418. [Online]. Available: https://doi.org/10.1145/3383219.3383278

[35] N. Alshuqayran, N. Ali, and R. Evans, "A systematic mapping study in microservice architecture," in *2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA)*, 2016, pp. 44–51.

[36] N. Dragoni, S. Giallorenzo, A. L. Lafuente, M. Mazzara, F. Montesi, R. Mustafin, and L. Safina, "Microservices: yesterday, today, and tomorrow," *Present and ulterior software engineering*, pp. 195–216, 2017.

[37] T. Cerny, M. J. Donahoo, and M. Trnka, "Contextual understanding of microservice architecture: Current and future directions," *SIGAPP Appl. Comput. Rev.*, vol. 17, no. 4, p. 29–45, Jan. 2018. [Online]. Available: https://doi.org/10.1145/3183628.3183631

[38] H. Kang, M. Le, and S. Tao, "Container and microservice driven design for cloud infrastructure devops," in *2016 IEEE International Conference on Cloud Engineering (IC2E)*, 2016, pp. 202–211.

[39] M. Waseem, P. Liang, and M. Shahin, "A systematic mapping study on microservices architecture in devops," *Journal of Systems and Software*, vol. 170, p. 110798, 2020.

[40] W. Mallouli, A. R. Cavalli, A. Bagnato, and E. M. De Oca, "Metrics-driven devsecops." in *ICSOFT*, 2020, pp. 228–233.

[41] M. Gokarna and R. Singh, "Devops: A historical review and future works," in *2021 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*, 2021, pp. 366–371.

[42] N. Wilde, B. Eddy, K. Patel, N. Cooper, V. Gamboa, B. Mishra, and K. Shah, "Security for devops deployment processes: Defenses risks research directions," *International Journal of Software Engineering & Applications*, vol. 7, no. 6, pp. 01–16, 2016.

[43] M. Sánchez-Gordón and R. Colomo-Palacios, "Security as culture: A systematic literature review of devsecops," in *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, ser. ICSEW'20. New York, NY, USA: Association for Computing Machinery, 2020, p. 266–269. [Online]. Available: https://doi.org/10.1145/3387940.3392233