НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
май–июнь 2023 Том 23 № 3 http://ntv.ifmo.ru/
SCIENTIFIC AND TECHNICAL JOURNAL OF INFORMATION TECHNOLOGIES, MECHANICS AND OPTICS
May–June 2023 Vol. 23 No 3 http://ntv.ifmo.ru/en/
ISSN 2226-1494 (print) ISSN 2500-0373 (online)

doi: 10.17586/2226-1494-2023-23-3-538-546

A novel approach to feature collection for anomaly detection in Kubernetes environment and agent for metrics collection from Kubernetes nodes

Ghadeer Darwesh[1], Jaafar Hammoud[2], Alisa A. Vorobeva[3]✉

[1,2,3] ITMO University, Saint Petersburg, 197101, Russian Federation

[1] ghadeerdarwesh32@gmail.com, https://orcid.org/0000-0003-1116-9410
[2] hammoudgj@gmail.com, https://orcid.org/0000-0002-2033-0838
[3] alice_w@mail.ru✉, https://orcid.org/0000-0001-6691-6167

Abstract

Kubernetes is a widely adopted open-source platform for managing containerized workloads and deploying applications in a microservices architecture. Despite its popularity, Kubernetes has faced numerous security challenges; deployments using Kubernetes are vulnerable to security risks. The current solutions for detecting anomalous behavior within a Kubernetes cluster lack real-time detection capabilities allowing hackers to exploit vulnerabilities and cause damage to production assets. This study aims to address these security concerns by proposing a new approach and novel agent to feature collection for anomaly detection in Kubernetes environment. It is proposed to use metrics (related to disk usage, CPU and network) collected by node exporter (Prometeus) directly from Kubernetes nodes. The simulation was conducted in a real-world production Kubernetes environment hosted on the Microsoft Azure, with results indicating the agent success in collecting 24 security metrics in a short amount of time. These metrics can be used to create a labeled time-series dataset of anomalies produced by microservices, enabling real-time detection of attacks based on the behavior of compromised nodes within the Kubernetes cluster. The proposed approach and developed agent for monitoring can be used to generate datasets for training anomaly detection models in the Kubernetes environment, based on artificial intelligence technologies, in real-time mode. The obtained results will be useful for researchers and specialists in the field of Kubernetes cybersecurity.

Keywords

Kubernetes, security, Kubernetes monitoring, attack detection, anomalies detection

For citation: Darwesh G., Hammoud J., Vorobeva A.A. A novel approach to feature collection for anomaly detection in Kubernetes environment and agent for metrics collection from Kubernetes nodes. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, 2023, vol. 23, no. 3, pp. 538–546. doi: 10.17586/2226-1494-2023-23-3-538-546

УДК 004.056

Новый способ сбора данных для обнаружения аномального поведения в среде Kubernetes и агент для сбора метрик с узлов

Гадир Дарвиш[1], Жаафар Хаммуд[2], Алиса Андреевна Воробьева[3]✉

[1,2,3] Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация

[1] ghadeerdarwesh32@gmail.com, https://orcid.org/0000-0003-1116-9410
[2] hammoudgj@gmail.com, https://orcid.org/0000-0002-2033-0838
[3] alice_w@mail.ru✉, https://orcid.org/0000-0001-6691-6167

Аннотация

Введение. Kubernetes — широко используемая платформа с открытым исходным кодом для управления контейнеризированными нагрузками и развертывания приложений в микросервисной архитектуре. Несмотря на популярность, платформа Kubernetes имеет многочисленные проблемы, связанные с безопасностью. Существующие решения для обнаружения аномального поведения в среде Kubernetes не позволяют детектировать аномальную активность, связанную с атаками злоумышленников, в режиме реального времени. **Метод.** Представлен новый способ сбора характеристик с узлов платформы Kubernetes для обнаружения

аномалий. Предложен новый агент мониторинга с собственными экстракторами и настраиваемыми правилами, которые собирают важные метрики с узлов реальной системы Kubernetes и экспортируют их в центральный набор данных. Применены метрики (связанные с использованием диска, процессора и сети), полученные от экспортеров Prometheus. **Основные результаты.** Выполнена симуляция в реальной среде Kubernetes на облачной платформе Microsoft Azure. Полученные результаты показали, что предложенный агент успешно собрал 24 метрики в централизованную базу данных за короткое время. Отобранные метрики могут быть использованы для создания размеченного набора данных временных рядов с аномалиями, создаваемыми микросервисом. Данное решение позволит обнаруживать атаки в реальном времени в среде Kubernetes на основе поведения скомпрометированных узлов в ее кластере. **Обсуждение.** Предложенный способ и разработанный агент мониторинга могут быть применены для формирования наборов данных для обучения моделей детектирования аномалий в среде Kubernetes, основанных на технологиях искусственного интеллекта, в режиме реального времени. Полученные результаты будут полезны исследователям и специалистам в области кибербезопасности приложения Kubernetes.

Ключевые слова

Kubernetes, безопасность, мониторинг Kubernetes, обнаружение атак, выявление аномалий

Ссылка для цитирования: Дарвиш Г., Хаммуд Ж., Воробьева А.А. Новый способ сбора данных для обнаружения аномального поведения в среде Kubernetes и агент для сбора метрик с узлов // Научно-технический вестник информационных технологий, механики и оптики. 2023. Т. 23, № 3. С. 538–546 (на англ. яз.). doi: 10.17586/2226-1494-2023-23-3-538-546

Introduction

The popularity of containerization technology has increased in recent years, with developers adopting it to address various real-world challenges, such as optimizing fetched, virtual machines auto-scaling, stack adjusting, performance misfortune issues, and various others [1]. Kubernetes has emerged as a popular platform for managing containerized workloads and deploying applications in a microservices architecture. However, Kubernetes deployments are vulnerable to security risks and existing solutions for detecting anomalous behavior within the system lack real-time detection capabilities.

In this study, we present a new monitoring agent for the Kubernetes system that collects important security and performance metrics from nodes in real-time. The agent features custom extractors and rules to ensure the collection of relevant security metrics, and exports them to a centralized database. The collected dataset will be used in future research to develop a machine learning module for anomaly detection.

The study begins with a background section that clarifies the significance of the proposed work. This section covers several topics including the current state of monitoring solutions in Kubernetes, the importance of the node exporter project, and the security challenges within the system. The study also includes a discussion of recent security challenges and threats in the Kubernetes environment to provide context for the proposed work and explain why it is necessary to address these security issues. The next section highlights the recent security challenges and threats in the Kubernetes environment. Finally, we describe the main contribution of this work, the new monitoring service that we have developed, and its ability to collect security metrics and export them to a central database provisioned as a Postgres deployment in the Kubernetes system.

Background

Kubernetes is a widely adopted open-source platform for the orchestration of containerized services across a dispersed cluster of nodes. It offers a robust infrastructure with the capability for zero-downtime deployment, scaling, automatic rollback, self-healing, load balancing, and service discovery [2]. This platform is utilized for the deployment, management, scaling, and composition of application containers over a cluster of hosts.

At a high level, a Kubernetes environment consists of a control plane (master), a distributed storage system for maintaining the stability of the cluster state (etcd), and multiple cluster nodes (Kubelets)[1]. Fig. 1 describes the Kubernetes architecture overview.

The control plane is comprised of various components to manage object states, including the Kube-apiserver which implements a RESTful interface to interact with libraries and tools for cluster administration, the Kube-controller-manager which maintains the state of the cluster, and the Kube-scheduler which plans and schedules workloads across nodes in the cluster [3].

Cluster nodes are machines managed by the master nodes and are equipped with the Kubelet and Kube-proxy components on top of Docker.

The fundamental Kubernetes objects are[1]:

— The Services used to characterize policies and logical set of Pods accessed by these policies.
— Volume is a directory accessible to all containers running inside the Pod.
— The pod is the smallest execution unit in the Kubernetes environment. It presents a single application that can consist of multiple storage volumes and containers. There are different sorts of pods [4]:
  — ReplicaSet, the default, is a simple type relatively. It guarantees the required number of pods are running.
  — Deployment is an explanatory way of managing pods through ReplicaSets. Incorporates rollback and rolling update mechanisms.
  — Daemonset guarantees that each node will run an instance of a pod. Utilized for cluster services like log forwarding and monitoring.

[1] Kubernetes Documentation | Kubernetes. Available at: https://kubernetes.io/docs/home/ (accessed: 31.08.2022).

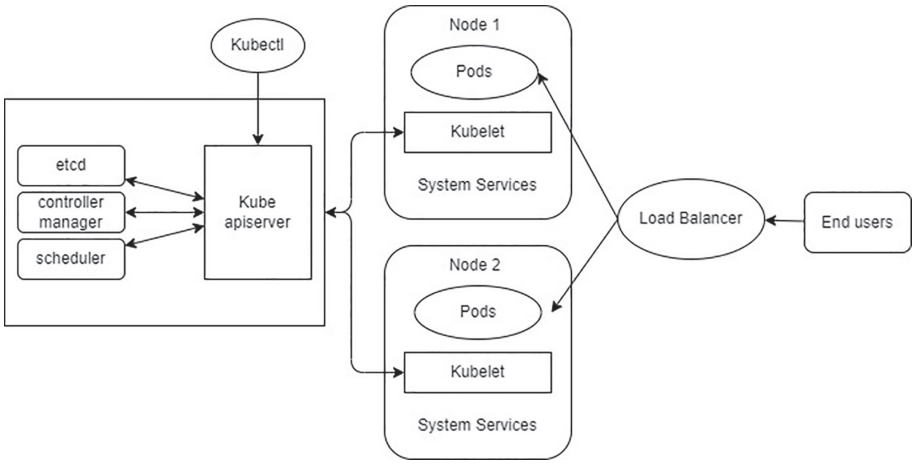

Fig. 1. Kubernetes Architecture Overview

— StatefulSet is custom fitted to managing pods that must hold on or maintain state.
— CronJob and Job run short-lived jobs as a one-off or on a schedule.

Kubernetes Monitoring

The application of containerization in a Kubernetes environment presents a unique challenge for monitoring. Monitoring the performance and resolving issues in a system that consists of hundreds of microservices deployed across thousands, or even millions, of ephemeral containers, requires a robust approach.

Kubernetes is a highly decentralized system comprised of various nested components. Monitoring in a Kubernetes environment involves monitoring the application and hosts as well as the containers and clusters. Additionally, the use of automated scheduling in Kubernetes introduces an additional layer of complexity, as the workloads and resources are managed optimally and dynamically, making it difficult to determine the identity or number of nodes running on the pods. To address this, a robust tagging system can be employed alongside logging to collect information from the clusters, which can then be exposed to an endpoint for a service to collect and analyze the metrics.

A comprehensive monitoring solution for Kubernetes must meet certain critical requirements, including:
— monitoring of all layers of the technology stack, including the host infrastructure on which Kubernetes runs, core components, pods, nodes, and containers within the cluster as well as all applications and services running in Kubernetes containers;
— dynamic identification and monitoring of services as they arise;
— providing a means of connecting relevant information for grouping and analysis of related logs, metrics, and other observability data.

In the Kubernetes environment, the provision of all required tools for monitoring resources, such as the collection of metrics from nodes within the cluster, is not natively supported. To address this requirement, the use of an external node-exporter service is proposed. We used the Prometheus Exporter designed specifically for collecting metrics and monitoring the host system. It is a widely used service for exposing hardware and operating system metrics from *NIX-based Bits[1], and can also be used for tagging and exporting metrics in the Kubernetes environment.

As depicted in Fig. 2, the Node Exporter component is integrated into the Prometheus Operator which is responsible for the Kubernetes-native deployment and management of Prometheus and related monitoring components. The Node Exporter operates as an agent on each node within the cluster which may also host the Kubernetes cluster. Its function is to gather monitoring data, such as memory usage, CPU utilization, disk space, inodes, and network statistics, from each node and forward it to the Prometheus server[2]. The Prometheus server stores this data as time-series data in the form of (timestamp, metric) pairs which can then be visualized using tools such as Grafana.

Kubernetes security

The deployment of Kubernetes can be performed in various settings, including on-premises, on bare metal, and within public clouds (either by creating a custom Kubernetes construct on virtual machines or by utilizing a managed service). Kubernetes was designed to be highly portable enabling users to effectively switch between these environments and move their workloads.

This high level of customization in Kubernetes makes it adaptable to a wide range of scenarios; however, it also represents a significant weakness in terms of security [5]. Kubernetes is designed to be highly customizable and requires users to enable specific functionalities to secure their cluster. This means that the engineers responsible for deploying the Kubernetes platform must have an in-depth understanding of all potential attack vectors and vulnerabilities that can result from inadequate configuration.

[1] GitHub — prometheus/node_exporter: Exporter for machine metrics. Available at: https://github.com/prometheus/node_ exporter (accessed: 10.04.2023).

[2] Getting started | Prometheus. Available at: https://prometheus.io/docs/prometheus/latest/getting_started/ (accessed: 10.04.2023).

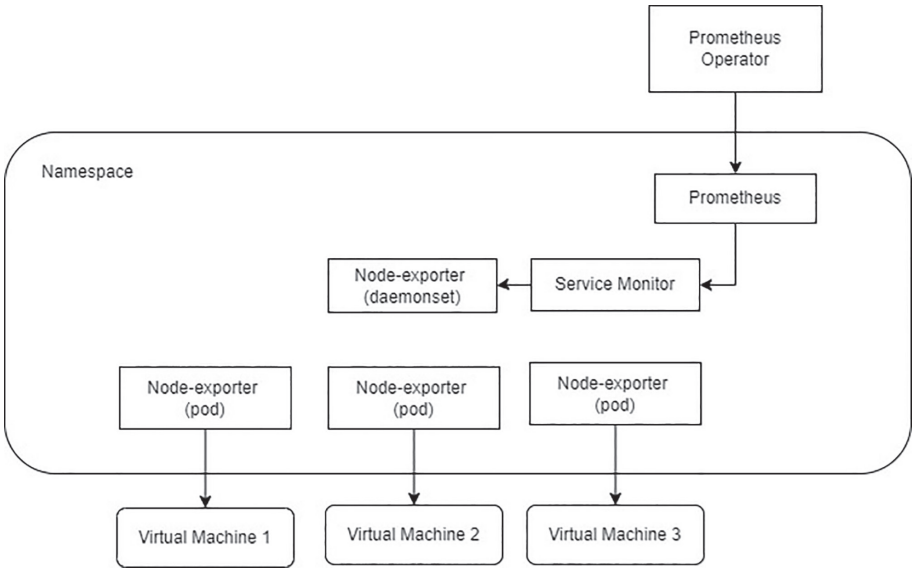

Fig. 2. Prometheus Node Exporter

Tesla Inc. experienced a security breach as a result of the absence of password protection on their Kubernetes administrative console. In a separate incident, Capital One's inadequate AWS firewall definitions led to the exposure of 30 GB of credit application data affecting 106 million customers. These events highlight the significance of placing security as a top priority alongside the deployment of Kubernetes. There are numerous threats to Kubernetes workloads including the potential for attackers to gain control of the entire cluster by breaching the control plane. Additionally, attackers can compromise individual pods or the physical host running Kubernetes pods as well as any unsecured open connections to the public making network connections a possible entry point. Finally, containers are vulnerable to threats due to misconfigurations or backdoors in the container image, which can allow attacker's access to the physical host.

In [3] researchers described the most important Kubernetes security best practices that were reported by practitioners including:

— authorization and authentication to verify API requests, enforce non-root user access only, and disable anonymous login;
— adoption of network security policies and pod policies to regulate (restrict) traffic and ensure contextual security of pods within the cluster;
— keeping the version up-to-date and applying the latest security patches;
— access restriction and implementation of encryption techniques to secure sensitive data stored in etcd;
— implementation of resource requests and limitations;
— enabling SSL/TLS support and creating separate namespaces.

Related work

In the study conducted by Yu et al. [6], the focus was on the investigation of security concerns related to service communications in microservice-enabled fog applications. A comprehensive literature search was conducted with a specific focus on the security of microservice architecture. As a result of the literature search, 66 important papers were identified and analyzed accordingly. The authors centered their analysis on four major categories of security issues in microservice communication: data issues, container issues, network issues, and authorization issues.

One important aspect of Kubernetes security monitoring is the detection of anomalous behavior in the cluster. The existing approaches fall into three categories: log-based anomaly detection, virtual network monitoring, and container security measures.

Several studies have focused on log-based anomaly detection using methods such as source code analysis and information retrieval to create composite features for automatically detecting runtime problems in the system [7] Other studies have investigated the use of log message grouping and counting as well as the inherent linear characteristics of normal program workflows, to detect anomalies in logs. However, these methods are limited in their ability to detect anomalies caused by external attacks, such as web service attacks and Common Vulnerabilities and Exposures (CVE) attacks.

Another area of focus in Kubernetes security monitoring is the monitoring of the virtual network. Researchers have explored the use of network flow analysis, intrusion detection systems [8], and network segmentation to enhance the security of container-based networks [9]. These studies are limited to detecting anomalies at the network layer and do not account for the fact that malicious behavior can be disguised as normal login and installation activities.

In recent studies, researchers have introduced anomaly-based detection mechanisms to address security concerns in container environments. Various machine learning technics are used to analyze real-time performance data [6], including CPU utilization, memory utilization, and network metrics, disk read/write rate, network receive/transmit rate, and container management information to detect the anomaly in containers. These characteristics do not differ significantly if the system is in a normal state and during a cyberattack.

In [10], KubAnomaly is proposed, a system that provides security monitoring capabilities for anomaly detection on the Kubernetes orchestration platform. Authors have designed a container monitoring module for Kubernetes and applied neural network techniques to construct classification models that enhance its capability of detecting abnormal behaviors. The described agent service collects only events from monitor logs from Docker-based containers with Sysdig and Falco. Sysdig is a diverse and complex system offering complete monitoring suite including visualization and alerting. It might be used for the metrics collection, but there are more light-weighted tools that suits that task more, like Prometheus.

The security of containers is an important topic of discussion. As highlighted in the SANS12 report, several tools have been developed to enhance container security. For instance, AppArmor is a policy-based Linux kernel security module that allows system administrators to restrict process capabilities, such as network access and file read/write permissions, by using security profiles.

A monitoring system that leverages Kubernetes for dynamic resource provisioning in cloud environments was presented by Chang et al. [11]. Furthermore, Shah and Dubaria [12] compared the management features of Swarm, Docker, Kubernetes, and Google Cloud Platform, and noted that Kubernetes provides features, such as deployment, monitoring, and ease of scalability.

In the study conducted by Sultan et al. [1], a comprehensive review of the literature related to container security was performed. The authors presented the threats for four utilized cases of securing containers and elaborate on security threats and solutions to each security risk concerning each use case. The use cases outlined were securing containers from applications, protecting containers from each other and host, and finally, protecting the host from containers.

Muralidharan et al. [13] introduced a Kubernetes-based system for monitoring and managing the Internet of Things (IoT) in smart cities. In [14], Burns et al. documented the development of container management frameworks at Google and described the evolution of two internal systems, Brog and Omega, into Kubernetes.

Approach to feature collection for anomaly detection in Kubernetes environment and novel agent for metrics collection from Kubernetes nodes

In this section, we present our implemented system for monitoring and collecting metrics directly from nodes (and not from logs) operating within the Kubernetes environment.

Our system addresses the design challenges associated with the collection and export of data from nodes for use in anomaly classification models. Our novel agent service has been tailored for Kubernetes allowing for easy deployment within the Kubernetes environment. The agent service must be installed on every node in order to collect metrics for the entire cluster, which are then transmitted as time series data to a central database (DB) for easy analysis and processing.

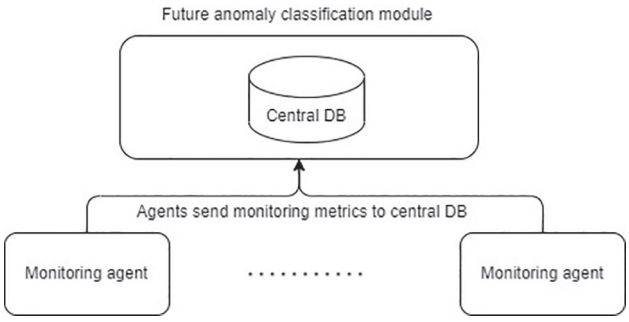

Fig. 3. Architecture of the system with proposed new monitoring agent in the Kubernetes environment for security purposes

Novel agent service for metrics collection from Kubernetes nodes

In our study, we have designed a novel agent service within the Kubernetes environment for collecting monitoring metrics from the nodes[1]. Fig. 3 presents the architecture of the proposed system. The objective of this deployment is to understand the behavior of all nodes, which is achieved through the deployment of the agent as a pod on every node within the cluster. To reduce the development workload, the docker image is based on the node-exporter service which is specifically designed for the collection of metrics and monitoring of the host system. Our agent utilizes the features of the node-exporter service to gather data and process it into a useful format for further analysis.

The new agent deployment consists of two Kubernetes components: the exporter service which accesses the exporter API to understand the metrics and data to be collected, and the Daemonset which is a Kubernetes workload that ensures the provisioning of the agent pods on each node in the cluster.

The agent also requires a prepared Postgres database to connect to and export our data. The number of records and the interval at which they are to be recorded are specified as environment variables in the agent's deployment definition and made available to the pods within the deployment. These variables are utilized by the agent to determine its operational behavior and tasks.

The selection of metrics from nodes is a challenge in the implementation of our system. An excessive collection of metrics would cause an overload on the nodes and negatively impact performance. To address this challenge, we have selected the most relevant and useful metrics for our security-related analysis. This approach not only captures the behavior of the nodes but also ensures optimal performance.

Feature collection for anomaly detection in Kubernetes environment

Our agent exports metrics to the Postgres database. The collected metrics are arguments we pass to the deployment

[1] ghadeerda/Kubernetes-monitoring-agent. Available at: https://github.com/ghadeerda/Kubernetes-monitoring-agent/tree/main (accessed: 10.04.2023).

Table 1. The proposed metrics and the feature set to construct anomaly detection model

Collected features (metric name)	Description
cpu_sec_idle	Seconds the CPUs spent in idle mode
disk_av_per	Filesystem space available percent to non-root users
disk_read	The total number of bytes read successfully
disk_write	The total number of bytes written successfully
net_receive	Network device statistic receive bytes
mem_pressure	/proc/vmstat information field pgmajfault
mem_av_per	Memory space available percent
forks_total	The total number of forks
intr	The total number of interrupts serviced
load1	1 minute load average
Load5	5 minutes load average
Load15	15 minutes load average
receive_drop	Network device statistic received and dropped
receive_errs	Network device statistic received with errors
transmit_packets	Network device statistic transmitted packets
tcp_sock_alloc	The number of TCP (Transmission Control Protocol) sockets in state “allocate”
tcp_sock_inuse	The number of TCP sockets in state “in use”
tcp_sock_mem	The number of TCP sockets in state “memory”
udp_sock_inuse	The number of UDP (User Datagram Protocol) sockets in state “in use”
udp_sock_mem	The number of UDP sockets in state “memory”
ipv4_sock_inuse	The number of IPv4 sockets “in use”
est_conn	TCP connections in the state “established”
lis_conn	TCP connections in the state “listen”
open_fds	The number of open file descriptors

of pods. It includes current events listed in the Table 1, that includes the collected feature to construct anomaly classification model.

The agent service scraps the metrics listed in the table above and exports them to the central database at an interval of 10 seconds; the interval is an environment variable that is exported to the pods. This interval was chosen to strike a balance between monitoring accuracy and performance efficiency. Based on research conducted by Akamai [10], approximately 900 attacks can occur within a 10-second time frame. Thus, a 10-second collection duration was deemed appropriate for our purposes. The collected metrics from central database are then utilized to develop a classification model for detecting anomalies in container behavior. This enables us to monitor the behavior of containers and gather monitoring metrics effectively.

The central database is structured to include a separate table for each node, wherein each record comprises of a time-series value and an attack indicator. This attack indicator column is intended to be utilized in future for the purpose of identifying attack records for machine learning modules.

Testing environment

The testing of our monitoring agent was performed in a real Kubernetes environment hosted on Microsoft Azure cloud using Azure Kubernetes Service (AKS). The cluster consisted of two nodes and a public load balancer with a default security group, and utilized Kubernetes version 1.22.

The docker image was built using prom/node-exporter: v1.4.0 as builder layer and pushed the new image to our container registry in Azure cloud. For the agent deployment we specify 0.5 CPU and 1G RAM, and its service was exposed on port 9100 for testing.

The result of our experiment is stored in Postgres: 14-alpine database which is running in the AKS and had an Azure Disk (5G) attached to it to store our data. To test the metrics obtained from nodes under different conditions, a new pod was deployed with the “stress” utility to subject the deployment to maximum usage over a prolonged period of time. This CPU load test was designed to push the processing power to its limits and generate different data values based on the stress load.

Results

Upon executing our agent on various nodes, we have observed the creation of a table (Table 2) in our database for each node, which displays the collected metrics accurately indicating the status of each node before and after the stress load test was conducted. Our defined metrics were collected successfully and are presented in a manner that will facilitate the development of future machine learning modules.

Table 2. Example of records in the resulting table in central database, collected metrics for one node with an interval of 10 seconds

id	date	time	cpu_sec_idle	disk_av_per	disk_read $\times 10^8$	disk_write $\times 10^9$	net_receive	mem_pressure	mem_av_per
1	2023-02-12	12:33:32	754.765	80.5205	6.66	6.75	3,461	4474	83.9407
2	2023-02-12	12:33:42	762.78	80.5205	6.66	6.75	5,428	4474	83.8183
3	2023-02-12	12:33:52	771.51	80.5205	6.66	6.75	6,913	4474	83.7209
4	2023-02-12	12:34:02	780.2	80.5205	6.66	6.75	8,398	4474	83.6702
5	2023-02-12	12:34:12	788.77	80.5205	6.66	6.75	9,883	4474	83.7633
6	2023-02-12	12:34:22	797.555	80.5205	6.66	6.75	11,368	4474	83.6889
7	2023-02-12	12:34:32	806.21	80.5204	6.66	6.75	12,895	4474	83.83
8	2023-02-12	12:34:42	814.2	80.5202	6.66	6.76	14,880	4474	83.537
9	2023-02-12	12:34:52	822.91	80.5203	6.66	6.76	16,435	4474	83.5568
10	2023-02-12	12:35:02	831.535	80.5203	6.66	6.76	17,920	4474	83.5106
11	2023-02-12	12:35:12	839.535	80.5203	6.66	6.76	19,475	4474	83.6359
12	2023-02-12	12:35:22	848.29	80.5203	6.66	6.76	21,002	4474	83.5364
13	2023-02-12	12:35:32	856.925	80.5202	6.66	6.76	22,487	4474	83.5117
14	2023-02-12	12:35:42	865.21	80.5202	6.66	6.76	24,042	4474	83.5375
15	2023-02-12	12:35:52	873.925	80.5202	6.66	6.76	25,527	4474	83.5383

The collected metrics are separated by periods of time considering the env variable (INTERVAL) we defined in the deployment.

After we apply z-score normalization to understand the probability of a score occurring within the distribution of the data, z-score is calculated by dividing the difference between the observed value and the sample mean by the sample standard deviation. Results are shown in the Table 3.

If z-score is close to zero it means that the data point is close to the average, a positive z-score means that the data point is above average, and a negative means the data point is below average.

Z-score data then could be used to build anomalies detection model. A data point can be considered anomaly or unusual if its z-score is above threshold values. The optimal threshold should be investigated. If z-score variance is close to the normal distribution, recommended optimal threshold is between –2.0 and +2.0. Otherwise in the case of z-score variance is more than 1.0, using ordinary threshold cannot point out anomaly.

Conclusion and future work

Kubernetes has emerged as the most widely adopted orchestration platform for Docker containers, and

Table 3. Part of dataset after z-score normalization

id	date	time	cpu_sec_idle	disk_av_per	disk_read	disk_write	net_receive	mem_pressure	mem_av_per
1	2023-12-02	12:33:32	–1.76304	0.63778	–0.70363	–0.63980	–1.73033	–0.70616	0.64287
2	2023-12-02	12:33:42	–1.74976	0.63778	–0.70363	–0.63970	–1.71260	–0.70616	0.60106
3	2023-12-02	12:33:52	–1.73529	0.63778	–0.70363	–0.63959	–1.69922	–0.70616	0.56779
4	2023-12-02	12:34:02	–1.72089	0.63778	–0.70363	–0.63938	–1.68583	–0.70616	0.55048
5	2023-12-02	12:34:12	–1.70670	0.63778	–0.70363	–0.63931	–1.67244	–0.70616	0.58228
6	2023-12-02	12:34:22	–1.69214	0.63778	–0.70363	–0.63925	–1.65906	–0.70616	0.55686
7	2023-12-02	12:34:32	–1.67780	0.63775	–0.70363	–0.63918	–1.64529	–0.70616	0.60506
8	2023-12-02	12:34:42	–1.66456	0.63768	–0.70363	–0.63862	–1.62740	–0.70616	0.50498
9	2023-12-02	12:34:52	–1.65013	0.63772	–0.70363	–0.63856	–1.61338	–0.70616	0.51174
10	2023-12-02	12:35:02	–1.63584	0.63772	–0.70363	–0.63844	–1.59999	–0.70616	0.49596
11	2023-12-02	12:35:12	–1.62259	0.63772	–0.70363	–0.63833	–1.58597	–0.70616	0.53876
12	2023-12-02	12:35:22	–1.60808	0.63772	–0.70363	–0.63810	–1.57221	–0.70616	0.50478
13	2023-12-02	12:35:32	–1.59377	0.63768	–0.70363	–0.63792	–1.55882	–0.70616	0.49634
14	2023-12-02	12:35:42	–1.58005	0.63768	–0.70363	–0.63782	–1.54481	–0.70616	0.50515
15	2023-12-02	12:33:32	–1.76304	0.63778	–0.70363	–0.63980	–1.73033	–0.70616	0.64287

is extensively utilized for application deployment and microservice creation. With advancements in containerization technology, information technology organizations are leveraging Kubernetes for managing their systems and reporting benefits during the deployment process.

However, security concerns have been raised in the Kubernetes environment as potential vulnerabilities can be exploited by hackers to cause damage to company resources.

In this study, we have developed a new agent service that collects metrics from the Kubernetes nodes and transmits them to a central database. The exported data encompass the most critical metrics for defining the nodes behavior and the attack baselines.

Furthermore, we aim to investigate the implementation of an anomaly detection plugin for detecting these anomalies and predict anomalies in the deployed system, or extrapolate the approach for use in another system.

References

1. Sultan S., Ahmad I., Dimitriou T. Container security: Issues, challenges, and the road ahead. *IEEE Access*, 2019, vol. 7, pp. 52976–52996. https://doi.org/10.1109/ACCESS.2019.2911732
2. Shamim Md.S.I., Bhuiyan F.A., Rahman A. XI Commandments of kubernetes security: A systematization of knowledge related to kubernetes security practices. *Proc. of the 2020 IEEE Secure Development (SecDev)*, 2020, pp. 58–64. https://doi.org/10.1109/SecDev45635.2020.00025
3. Darwesh G., Hammoud J., Vorobeva A.A. Security in kubernetes: best practices and security analysis. Bulletin of the Ural Federal District. *Security in the Information Sphere*, 2022, vol. 22, no. 2, pp. 63–69. https://doi.org/10.14529/SECUR220209
4. Mondal S.K., Pan R., Kabir H.M.D., Tian T., Dai H.N. Kubernetes in IT administration and serverless computing: An empirical study and research challenges. *Journal of Supercomputing*, 2022, vol. 78, no. 2, pp. 2937–2987. https://doi.org/10.1007/s11227-021-03982-3
5. Shamim S.I. Mitigating security attacks in kubernetes manifests for security best practices violation. *Proc. of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE)*, 2021, pp. 1689–1690. https://doi.org/10.1145/3468264.3473495
6. Yu D., Jin Y., Zhang Y., Zheng X. A survey on security issues in services communication of Microservices-enabled fog applications. *Concurrency and Computation: Practice and Experience*, 2019, vol. 31, no. 22, pp. e4436. https://doi.org/10.1002/CPE.4436
7. Lou J.-G., Fu Q., Yang S., Xu Y., Li J. Mining invariants from console logs for system problem detection. *Proc. of the USENIX Annual Technical Conference*, 2010, pp. 1–14.
8. Lin C.H., Tien C.W., Pao H.K. Efficient and effective NIDS for cloud virtualization environment. *Proc. of the 4th IEEE International Conference on Cloud Computing Technology and Science Proceedings*, 2012, pp. 249–254. https://doi.org/10.1109/cloudcom.2012.6427583
9. Gomez M.E. *Full Packet Capture Infrastructure Based on Docker Containers*. Tech. rep. SANS Institute InfoSec Reading Room. 2016.
10. Tien C.-W., Huang T.-Y., Tien C.-W., Huang T.-C., Kuo S.-Y. KubAnomaly: Anomaly detection for the Docker orchestration platform with neural network approaches. *Engineering Reports*, 2019, vol. 1, no. 5, pp. e12080. https://doi.org/10.1002/eng2.12080
11. Chang C.-C., Yang S.-R., Yeh E.-H., Lin P., Jeng J.-Y. A Kubernetes-based monitoring platform for dynamic cloud resource provisioning. *Proc. of the GLOBECOM 2017 — 2017 IEEE Global Communications Conference*, 2017, pp. 1–6. https://doi.org/10.1109/GLOCOM.2017.8254046
12. Shah J., Dubaria D. Building modern clouds: Using Docker, Kubernetes & Google Cloud Platform. *Proc. of the 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*, 2019, pp. 0184–0189. https://doi.org/10.1109/CCWC.2019.8666479
13. Song M., Zhang C., Haihong E. An auto scaling system for API Gateway based on Kubernetes. *Proc. of the 2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS)*, 2018, pp. 109–112. https://doi.org/10.1109/ICSESS.2018.8663784
14. Burns B., Grant B., Oppenheimer D., Brewer E., Wilkes J. Borg, Omega, and Kubernetes. *Queue*, 2016, vol. 14, no. 1, pp. 70–93. https://doi.org/10.1145/2898442.2898444

Литература

1. Sultan S., Ahmad I., Dimitriou T. Container security: Issues, challenges, and the road ahead // IEEE Access. 2019. V. 7. P. 52976–52996. https://doi.org/10.1109/ACCESS.2019.2911732
2. Shamim Md.S.I., Bhuiyan F.A., Rahman A. XI Commandments of kubernetes security: A systematization of knowledge related to kubernetes security practices // Proc. of the 2020 IEEE Secure Development (SecDev). 2020. P. 58–64. https://doi.org/10.1109/SecDev45635.2020.00025
3. Darwesh G., Hammoud J., Vorobeva A.A. Security in kubernetes: best practices and security analysis // Вестник УРФО. Безопасность в информационной сфере. 2022. Т. 22. № 2. С. 63–69. https://doi.org/10.14529/SECUR220209
4. Mondal S.K., Pan R., Kabir H.M.D., Tian T., Dai H.N. Kubernetes in IT administration and serverless computing: An empirical study and research challenges // Journal of Supercomputing. 2022. V. 78. N 2. P. 2937–2987. https://doi.org/10.1007/s11227-021-03982-3
5. Shamim S.I. Mitigating security attacks in kubernetes manifests for security best practices violation // Proc. of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE). 2021. P. 1689–1690. https://doi.org/10.1145/3468264.3473495
6. Yu D., Jin Y., Zhang Y., Zheng X. A survey on security issues in services communication of Microservices-enabled fog applications // Concurrency and Computation: Practice and Experience. 2019. V. 31. N 22. P. e4436. https://doi.org/10.1002/CPE.4436
7. Lou J.-G., Fu Q., Yang S., Xu Y., Li J. Mining invariants from console logs for system problem detection // Proc. of the USENIX Annual Technical Conference. 2010. P. 1–14.
8. Lin C.H., Tien C.W., Pao H.K. Efficient and effective NIDS for cloud virtualization environment // Proc. of the 4th IEEE International Conference on Cloud Computing Technology and Science Proceedings. 2012. P. 249–254. https://doi.org/10.1109/cloudcom.2012.6427583
9. Gomez M.E. Full Packet Capture Infrastructure Based on Docker Containers: Tech. rep. SANS Institute InfoSec Reading Room, 2016.
10. Tien C.-W., Huang T.-Y., Tien C.-W., Huang T.-C., Kuo S.-Y. KubAnomaly: Anomaly detection for the Docker orchestration platform with neural network approaches // Engineering Reports. 2019. V. 1. N 5. P. e12080. https://doi.org/10.1002/eng2.12080
11. Chang C.-C., Yang S.-R., Yeh E.-H., Lin P., Jeng J.-Y. A Kubernetes-based monitoring platform for dynamic cloud resource provisioning // Proc. of the GLOBECOM 2017 — 2017 IEEE Global Communications Conference. 2017. P. 1–6. https://doi.org/10.1109/GLOCOM.2017.8254046
12. Shah J., Dubaria D. Building modern clouds: Using Docker, Kubernetes & Google Cloud Platform // Proc. of the 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC). 2019. P. 0184–0189. https://doi.org/10.1109/CCWC.2019.8666479
13. Song M., Zhang C., Haihong E. An auto scaling system for API Gateway based on Kubernetes // Proc. of the 2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS). 2018. P. 109–112. https://doi.org/10.1109/ICSESS.2018.8663784
14. Burns B., Grant B., Oppenheimer D., Brewer E., Wilkes J. Borg, Omega, and Kubernetes // Queue. 2016. V. 14. N 1. P. 70–93. https://doi.org/10.1145/2898442.2898444

Authors

Ghadeer Darwesh — PhD Student, ITMO University, Saint Petersburg, 197101, Russian Federation, https://orcid.org/0000-0003-1116-9410, ghadeerdarwesh32@gmail.com
Jaafar Hammoud — PhD Student, ITMO University, Saint Petersburg, 197101, Russian Federation, SC 57222044000, https://orcid.org/0000-0002-2033-0838, hammoudgj@gmail.com
Alisa A. Vorobeva — PhD, Associate Professor, Saint Petersburg, 197101, Russian Federation, SC 57191359167, https://orcid.org/0000-0001-6691-6167, Alice_w@mail.ru

Авторы

Дарвиш Гадир — аспирант, Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация, https://orcid.org/0000-0003-1116-9410, ghadeerdarwesh32@gmail.com
Хаммуд Жаафар — аспирант, Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация, SC 57222044000, https://orcid.org/0000-0002-2033-0838, hammoudgj@gmail.com
Воробьева Алиса Андреевна — кандидат технических наук, доцент, Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация, SC 57191359167, https://orcid.org/0000-0001-6691-6167, Alice_w@mail.ru

Received 25.11.2022
Approved after reviewing 20.02.2023
Accepted 16.05.2023

Статья поступила в редакцию 25.11.2022
Одобрена после рецензирования 20.02.2023
Принята к печати 16.05.2023