Cloud Monitoring Tools: A Comparative Study on Different Cloud Platforms

Ayushi Trivedi, Omkar Swami, Nishad Vyas and Thomas Courtney
Department of Electrical and Computer Engineering,
University of Ottawa, Ontario, Canada
*Contact: atriv057@uottawa.ca, phone +13435580409

***Abstract*— Cloud monitoring technologies are becoming increasingly vital in guaranteeing efficient and effective cloud resource management. The requirement for powerful and dependable monitoring solutions has grown crucial as more enterprises employ cloud-based services. Numerous studies have been conducted on various elements of the Cloud, such as its characteristics, features, virtualization technologies, security, and privacy. However, there appears to be a dearth of in-depth examination of Cloud monitoring in these studies. To fill this void, this study provides a comprehensive assessment of Cloud monitoring. Our goal is to analyse and assess the most popular cloud monitoring programmes in terms of functionality, performance, and convenience of use. The research will involve an in-depth examination of each tool's major features and functions, such as monitoring, alerting, and notification, data visualisation and analysis, and interaction with other tools and technologies. The research will also evaluate the simplicity of use, degree of customization and setup necessary for each tool, as well as the quality of support and documentation provided. Finally, after the benchmark results are in, this project will recommend a cloud monitoring tool as well as its implementation.**

I. INTRODUCTION

As cloud computing grows more popular, the necessity for good cloud monitoring tools becomes even more critical. With the advent of dynamic and sophisticated Cloud settings, real-time insight into the performance, availability, and security of Cloud resources is critical. There are several Cloud monitoring solutions on the market, but choosing the correct one might be difficult. As a result, comparing and analysing Cloud monitoring technologies may assist organisations in making educated decisions regarding the selection and use of these tools. We want to present a thorough analysis and comparative research of prominent Cloud monitoring tools, including their features, functions, strengths, and limits, in this paper. By studying and comparing various products, we intend to give important insights that will assist organisations in making more educated decisions when picking the best Cloud monitoring tool for their unique needs.

Cloud Monitoring: A brief Overview

Cloud computing is defined as a "Model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (network, servers, storage, applications, services) that can be rapidly provisioned and released with minimal management effort or service provider interaction" by the National Institute of Standards and Technology (NIST) [1]. Commercial cloud providers such as Amazon Web Services (AWS), Microsoft Azure, Salesforce.com, Google App Engine, and others provide cloud consumers with the ability to deploy their applications across an infinite resource pool with little to no capital investment and low operating costs proportional to usage. Amazon EC2 cloud, for example, supports around 500,000 physical servers, each of which hosts several virtual machines that may be dynamically activated or withdrawn [2]. Cloud monitoring is essential in terms of capacity and resource planning, capacity and resource management, data center management, SLA management, billing, troubleshooting, performance management and security management. The two cloud monitoring tools we have studied are as follows:

CloudWatch[3]: CloudWatch is one of the most widely used commercial cloud monitoring solutions. It is given by Amazon to allow its customers to monitor their EC2 resources. As a result, multi-cloud infrastructure monitoring is not supported. CloudWatch's technological techniques to data collection are implicit and not disclosed to consumers. CloudWatch is restricted in its ability to monitor resources across cloud levels. However, an API is given for users to gather metrics at any cloud layer, but users must write extra code to do so. Amazon CloudWatch is AWS's native monitoring solution, providing customers with a single view of their AWS resources, applications, and services running on AWS and on-premises servers.

CloudWatch gives users with the observability and insights they require across their entire AWS ecosystem, assisting in the improvement of operational performance and resource optimisation.

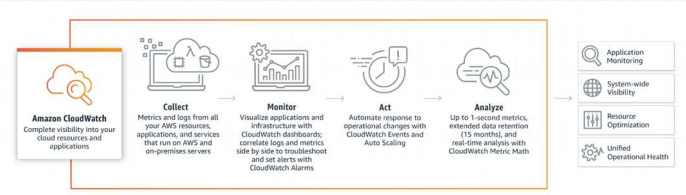

Figure 1: Representation on how CloudWatch works[5]

Azure Monitor[4]: Azure Monitor is Microsoft Azure's native monitoring tool, designed to assist users maximise the performance and availability of their apps and cloud services throughout their Azure ecosystem.

To do this, Azure Monitor gives customers with comprehensive visibility into their cloud and on-premises environments, gathering and analysing data from a range of sources before storing it for subsequent cost and performance optimisation.

Azure Monitor provides insight into how your apps are operating and proactively discovers issues. It improves the availability and performance of your applications and services by providing end-to-end solutions for collecting, analysing, and responding on telemetry from the cloud and on-premises environments.

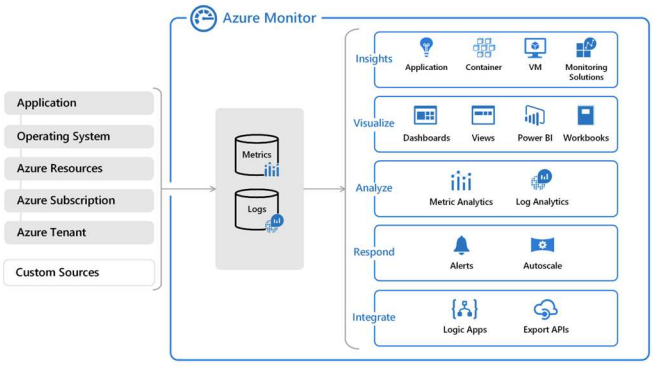

Figure 2: High level representation on how Azure monitor works[5]

The main contribution of this paper are summarized as follows:

- In this research, we have compared cloud monitoring tools of AWS platform and Azure platform in terms of their capabilities, performance and ease of use.
- We have also performed an in-depth analysis of the key features and functionalities of each tool, including monitoring, alerting and notification, data visualization and analysis along with integration with other tools and technologies.
- We would further benchmark the results of different tools and suggest a cloud monitoring tool along with its implementation.

The remainder of the paper is structured in the following manner. Section II focuses on literature reviews and the research we did based on existing work. Section III describes the overall architecture of a basic cloud monitoring scenario. Section IV focusses on in-depth discussion on implementation. Section V provides the comparative analysis between these cloud monitoring tools based on the evaluation of several factors. Section V concludes our study and helps us come up with the best cloud monitoring tool.

III. LITERATURE REVIEW

The paper "Cloud Infrastructure Monitoring Using Elastic Stack" by T. Patil and S. Khan proposes a solution for monitoring cloud infrastructure using Elastic Stack[8]. The authors highlight the importance of monitoring cloud infrastructure to ensure its availability, reliability, and performance. They argue that traditional monitoring tools are not suitable for monitoring cloud infrastructure due to their inability to handle the large-scale and dynamic nature of the cloud. The authors propose a solution that uses Elastic Stack, which is an open-source stack that includes Elasticsearch, Logstash, and Kibana. Elasticsearch is used to store and index data, Logstash is used to collect and parse data, and Kibana is used for visualization and analysis. The authors explain the architecture of their solution and the various components involved, including the agents for data collection, the indexing pipeline for data processing, and the visualizations for data analysis. They also discuss the benefits of their solution, including its scalability, flexibility, and ease of use. The authors conclude that their solution provides an effective way of monitoring cloud infrastructure using Elastic Stack. They argue that their solution offers several advantages over traditional monitoring tools, including better scalability, real-time monitoring, and easy customization. However, the authors do not provide any experimental results to validate their claims, which is a limitation of the paper. Overall, the paper provides a good overview of the proposed solution for cloud infrastructure monitoring using Elastic Stack. However, more research is needed to validate the effectiveness of the solution and compare it with other existing monitoring tools.

The paper "A Comparative Study of Cloud Monitoring Tools: Challenges and Opportunities" by T. Kaur and N. Kumar[9] explores various cloud monitoring tools available in the market and their challenges and opportunities. The study aims to identify the limitations of existing cloud monitoring tools and provide insights for the development of future tools. The authors conducted a comparative study of several cloud monitoring tools such as Amazon CloudWatch, Google Stackdriver, Microsoft Azure Monitor, and Prometheus. The paper analyzed the features and capabilities of each tool and compared them based on various criteria such as cost, scalability, ease of use, and compatibility with different cloud environments. The paper identifies several challenges that exist in cloud monitoring tools such as lack of support for multi-cloud environments, limited scalability, and high costs. The authors propose various opportunities for future research, including the development of tools that can support multi-cloud environments, the integration of artificial intelligence and machine learning for improved monitoring, and the use of blockchain for enhanced security. Overall, the paper provides a comprehensive analysis of various cloud monitoring tools and highlights the challenges and opportunities for the development of future tools. It serves as a valuable resource for researchers and practitioners interested in cloud monitoring and management.

IV. ARCHITECTURE

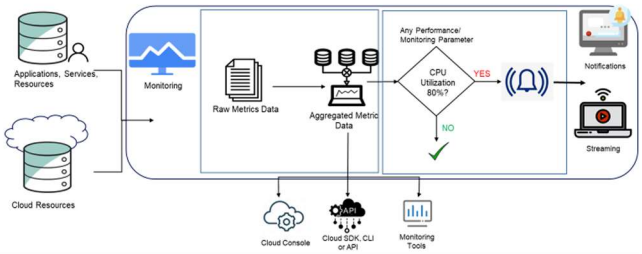

Figure 3: Architecture of cloud monitoring

The Metrics and Alarms capabilities of AWS cloud monitoring and Azure monitoring service allow you to actively and passively monitor your cloud resources. The AWS and Azure monitoring services helps to monitor the resources using metrics and alarms. A metric is a data point related to health, capacity or performance of a given resource.

We have divided this architecture into three separate parts.

1. Input: The input is raw metric data that we receive from different cloud services, application and resources. An application can be a node.js, website or a database. A resource can be an EC2 instance that is running on top of that application. This raw data is used to process and generate useful metrics.
2. Processing: This involves processing the raw data and filtering the important things and generating aggregated data that is further used to plot graphs, display useful insights about the platform.
3. Output: We display this aggregated results on Amazon cloud console in form of different graphs and alerts. We also used inbuilt monitoring tools provided by cloud platform.

IV. IMPLEMENTATION

Implementing cloud monitoring tools using AWS and Azure is a crucial aspect of ensuring that a cloud-based infrastructure and applications are running efficiently and effectively. Both AWS and Azure offer a variety of monitoring tools and services that can be used to monitor different aspects of a cloud infrastructure and applications.

To implement cloud monitoring tools using AWS, you can use services such as CloudWatch, which provides monitoring and logging services for AWS resources and applications. CloudWatch can be used to monitor metrics such as CPU utilization, network traffic, and disk usage, as well as to set alarms and trigger actions based on thresholds. AWS also offers services such as AWS Config, which provides a detailed inventory of all AWS resources and configurations and can be used to monitor changes to the resources over time.

Similarly, Azure offers a range of monitoring tools and services such as Azure Monitor, which provides monitoring and logging services for Azure resources and applications. Azure Monitor can be used to monitor metrics such as CPU utilization, memory usage, and network traffic, as well as to set alerts and trigger actions based on thresholds. Azure also offers services such as Azure Log Analytics, which provides advanced analytics and visualization capabilities for monitoring and troubleshooting applications.

To implement cloud monitoring tools using AWS or Azure, you can follow a similar methodology as outlined in the text. You can begin by setting up the required infrastructure, such as virtual machines and storage resources, and configuring them to meet the specific requirements of an application. Once the infrastructure is set up, you can deploy an application using the platform-specific tools and services, and then configure the monitoring tools and services to monitor the application's performance and health. To ensure consistency and accuracy in monitoring results, it is important to use the same application with the same functionalities across all platforms. This ensures that any differences in performance are not due to differences in the data used. Additionally, it is important to regularly analyze and optimize monitoring results to identify any issues and improve the performance and scalability of an application. Overall, implementing cloud monitoring tools using AWS and Azure is a critical step in ensuring that a cloud-based infrastructure and applications are running smoothly and efficiently. By following a structured methodology and using the right tools and services, you can effectively monitor and manage a cloud infrastructure and applications to achieve optimal performance and scalability.

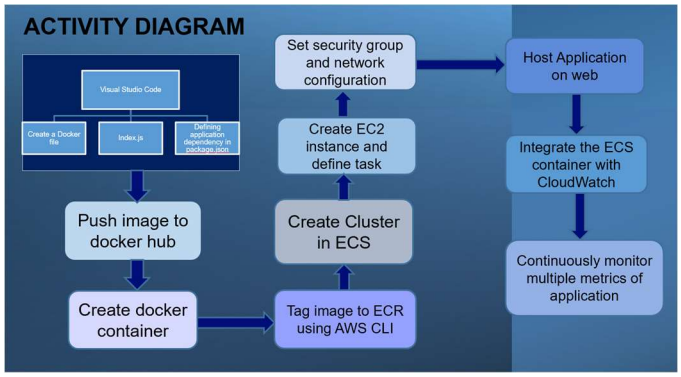

Figure 4: Activity flow diagram for AWS

Figure 4 shows an activity flow diagram on AWS platform. We used following steps for Implementation on AWS Platform:

- The implementation of a containerized application involves several key steps that are essential to ensure its smooth deployment and operation. In this section, we will discuss in detail the various steps involved in deploying a containerized application on AWS Elastic Container Service (ECS).
- The first step in the process is creating the application itself. This typically involves writing the code for the application and packaging it into a Docker container. To create a Docker container, we need to define a Dockerfile and create an index.js file that contains the application code. The Dockerfile specifies the various dependencies and configurations needed to build the container, while the index.js file contains the actual application code.
- Once the Docker container is ready, the next step is to push it to a Docker image repository, such as Docker Hub. This is done using the "docker push" command, which uploads the container to the repository.

- The next step is to create an Amazon Elastic Container Registry (ECR) repository and tag the image to the repository using the AWS CLI. This step is essential for the successful deployment of the application in ECS.
- After creating the ECR repository and tagging the image, we need to create a cluster in Amazon ECS. A cluster is a logical grouping of container instances and tasks that can be managed as a single unit. It provides a central resource to manage and monitor the containers running on the AWS infrastructure.
- To run the container on an EC2 instance in the ECS cluster, we need to create an EC2 instance and define a task. The task definition specifies the Docker container to be used, along with the resources needed to run it, such as CPU and memory.
- To ensure the security of the container and the application, we need to set up a security group and network configuration. A security group is a virtual firewall that controls inbound and outbound traffic to the container instances. The network configuration, on the other hand, specifies the network settings for the container, such as the IP address and port number.
- Once the container is up and running, we can host the application on the web by exposing it to the internet. This is typically done by configuring the container's load balancer and domain name system (DNS) settings.
- To monitor the performance of the containerized application, we need to integrate it with Amazon CloudWatch, a monitoring and logging service provided by AWS. This allows us to monitor the container's resource utilization, application logs, and other metrics.
- Finally, we need to continuously monitor the application's metrics to ensure its availability and performance. This involves configuring alerts and notifications based on predefined thresholds and metrics.

In conclusion, the successful deployment of a containerized application on Amazon ECS involves several key steps, including creating the application, pushing the Docker container to a repository, creating an ECR repository, creating a cluster, defining tasks, setting up security groups and network configuration, hosting the application on the web, integrating it with CloudWatch, and monitoring its metrics continuously.

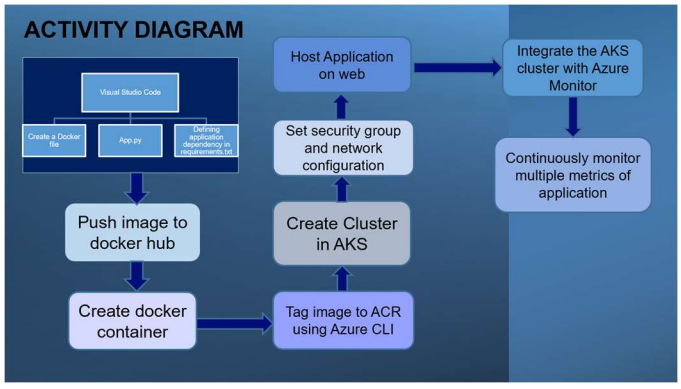

Figure 5: Activity flow diagram for Azure

Figure 5 shows an activity flow diagram on Azure platform. We used following steps for Implementation on Azure Platform:

- The process of deploying and managing containerized applications has become increasingly popular in recent years due to the benefits of containerization, such as improved portability, scalability, and reliability. We will discuss the steps involved in deploying and managing containerized applications using Docker and Azure Kubernetes Service (AKS), along with the tools and techniques used for monitoring and securing the application.
- The first step in the process is to push the Docker image of the application to a container registry such as Docker Hub. This ensures that the image is available for distribution and deployment. Once the image is available in the registry, it can be pulled and instantiated as a Docker container.
- The next step is to tag the Docker image with the Azure Container Registry (ACR) repository name using Azure CLI. This enables the Docker image to be stored in ACR, which is a managed container registry provided by Microsoft. ACR offers a secure and scalable way to store and manage container images, and it can be integrated with other Azure services.
- After tagging the Docker image with ACR, the next step is to create a cluster in AKS. AKS is a fully managed Kubernetes service provided by Microsoft, which automates the deployment, scaling, and management of containerized applications. Creating a cluster in AKS involves defining the number of nodes, specifying the cluster location, and configuring the authentication and authorization settings.
- Once the AKS cluster is created, the next step is to set up the security group and network configuration. This involves defining the firewall rules and network policies that control the traffic to and from the AKS cluster. The security group and network configuration help to ensure that the application is secure and accessible only to authorized users.
- The next step is to host the application on the web. This involves configuring the container and exposing the application endpoints to the internet. Once the application is hosted on the web, it can be accessed by users across the internet.
- To monitor the application, the AKS cluster can be integrated with Azure Monitor. Azure Monitor is a cloud-based monitoring and analytics solution provided by Microsoft, which offers a range of tools for monitoring various metrics such as CPU and memory usage, response times, and error rates. By integrating the AKS cluster with Azure Monitor, it is possible to continuously monitor the multiple metrics of the application, detect and troubleshoot issues, and optimize the application performance.

V. RESULTS

We discovered that CPU utilisation followed a diurnal pattern, with peak usage during work hours and reduced usage at night. CloudWatch gave us with extra information, such as the amount of CPU credits utilised by the instance, in addition to the standard CPU utilisation metrics, which might be valuable in managing EC2 instances that utilise burstable performance. We also utilised CloudWatch alarms to create CPU utilisation targets and receive warnings when the threshold was exceeded.

Overall, our research confirmed CloudWatch's and Azure monitoring's use in monitoring CPU utilisation in the AWS cloud and Azure cloud platform. We were able to obtain insights into the performance of our application and identify possible performance bottlenecks thanks to the measurements given by CloudWatch and Azure Monitor. These insights may be utilised to improve the overall performance of cloud-based applications by optimising resource use.

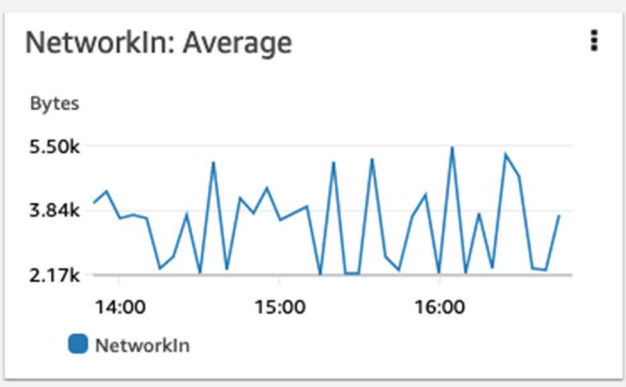

Figure 6: NetworkIn Average

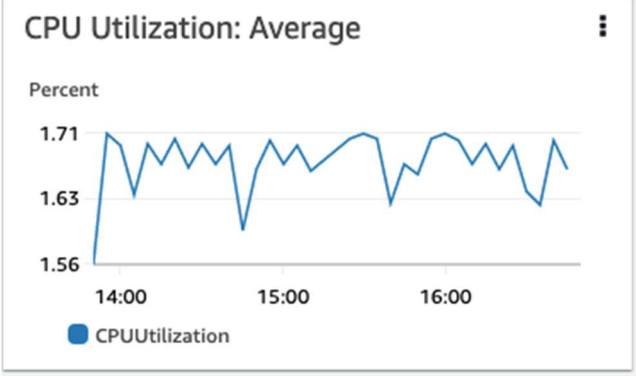

Figure 7: CPU Utilization Average

Platform	Scalability	Elasticity	Adaptability	Timeliness	Autonomicity	Comprehensiveness	Extensibility	Intrusiveness	Resilience	Reliability	Availability	Accuracy
CloudWatch [95]		✓		✓			✓					
AzureWatch [137]	✓		✓		✓		✓					
CloudKick [96]	✓		✓									
CloudStatus [48]				✓								
Nimsoft [97]	✓					✓						
Monitis [99]						✓						
LogicMonitor [100]	✓	✓				✓						
Aneka [101]	✓	✓										
GroundWork [129]						✓						

Figure 6: Key properties of commercial cloud monitoring platform

CONCLUSION

Both cloud monitoring solutions provide an extraordinary variety of capabilities and services for customers on their respective native platforms, so comparing one against the other is difficult—you won't need Amazon CloudWatch if you're just running on Azure (and vice versa for Azure Monitor).

However, as companies increasingly use many clouds to support their digital transformation initiatives, cloud teams will find it incredibly challenging to manage all their cloud services, resources, and tools dispersed across various public clouds. In reality, according to a recent analysis based on actual cloud customer data, over half of organisations have embraced a multi-cloud approach, with the majority of those utilising a mix of AWS and Azure.

REFERENCES

[1] Mell P, Grance T (2011) The NIST definition of cloud computing (draft). NIST Spec Publ 800:145

[2] Shin S, Gu G (2012) CloudWatcher: network security monitoring using openflow in dynamic cloud networks (or: How to provide security monitoring as a service in clouds?). In: 2012 20th IEEE international conference on network protocols (ICNP), pp 1–6

[3] CloudWatch(2014) http://awsdocs.s3.amazonaws.com/AmazonCloudWatch/latest/acw-dg.pdf.

[4] Sahay, Rahul, and Rahul Sahay. "Azure Monitoring." Microsoft Azure Architect Technologies Study Companion: Hands-on Preparation and Practice for Exam AZ-300 and AZ-303 (2020): 139-167.

[5] https://blogs.vmware.com/cloudhealth/cloud-services-aws-cloudwatch-azure-monitor/

[6] "Cloud Infrastructure Monitoring Using Elastic Stack" by T. Patil and S. Khan in 2020 International Conference on Inventive Research in Computing Applications.

[7] T. Patil and S. Khan, "Cloud Infrastructure Monitoring Using Elastic Stack," 2020 International Conference on Inventive Research in Computing Applications (ICIRCA), Coimbatore, India, 2020, pp. 1-4, doi: 10.1109/ICIRCA48451.2020.9270418.

[8] T. Kaur and N. Kumar, "A Comparative Study of Cloud Monitoring Tools: Challenges and Opportunities," 2020 IEEE International Conference on Computing, Power and Communication Technologies (GUCON), Greater Noida, India, 2020, pp. 1114-1119, doi: 10.1109/GUCON48538.2020.9219779.

[9] Z. Zhang, H. Zhang, Y. Chen, X. Jia and B. Wu, "Design and Implementation of a Cloud Monitoring System Based on OpenStack," 2018 International Conference on Networking and Network Applications (NaNA), Xi'an, China, 2018, pp. 51-54, doi: 10.1109/NaNA.2018.86444.