Original Article

Multi Cluster Monitoring for Fault Detection Using Novel Kubernetes with Prometheus over Docker Container

Sai Vimal Kumar V[1], Malathi K[2]

[1]Research Scholar, Department of Computer Science and Engineering, Saveetha School of Engineering, Saveetha Institute of Medical and Technical Sciences, Saveetha University, Chennai, Tamilnadu. India, Pincode: 602105.
[2]Project Guide, Corresponding Author, Department of Computer Science and Engineering, Saveetha School of Engineering, Saveetha Institute of Medical and Technical Sciences, Saveetha University, Chennai, Tamilnadu. India, Pincode: 602105.

Abstract

Aim: The aim of the study is to estimate the accuracy of Multi-Cluster Monitoring for fault detection using novel kubernetes with prometheus over docker containers. **Materials and Methods:** Sample groups that are considered in the project can be classified into two each has 20 samples, one for novel kubernetes with prometheus and other for docker containers, which are tested using 0.80 for G-power to determine the sample size and for t-test analysis. 5 Clusters have been used in each group for estimating the efficient accuracy. **Results:** The novel kubernetes with prometheus with efficient accuracy of 87.60%,which by far seems to be better than the docker containers which gives around 84.20%. The significance is around 0.034 ($p<0.05$) and therefore there is a statistical insignificant difference among the study group. **Conclusion:** Novel kubernetes with prometheus seem to be better in Multi-Clustermonitoring for fault detection over the docker containers.

Keywords: Novel Kubernetes with Prometheus, Multi Cluster, Monitoring, Docker Container, Metrics.

DOI:10.47750/pnr.2022.13.S04.185

INTRODUCTION

Nowadays, the world of distributed systems is rapidly shifting to cloud-native concepts such as containers, microservices, service meshes, and serverless applications, which provide features such as built-in service discovery and load balancing, automated rollouts and rollbacks, and self-healing capabilities (Arundel and Domingus 2019). All of these new concepts are aimed at abstracting network and hardware issues from developers, allowing development teams to focus more resources on the product itself (Burns and Tracey 2018). The Cloud Native Computing Foundation was established with the goal of empowering enterprises to design and execute scalable applications in dynamic environments using various techniques of service selection in cloud environments (Madushan 2021). Use Prometheus metrics and alerting capabilities to solve the challenge of observability in distributed systems, specifically the problem of fault detection in multi clustering. Fault detection can aid developers and operations engineers in diagnosing issues, app performance regression, identifying resource misuse, gaining insight into customers, detecting data breaches, and preventing faults (Madushan 2021; Ahmadi, n.d.). Forecasting metrics will allow us to spot faults earlier and more reliably, reducing the need for human intervention. Kubernetes and Prometheus are projects that have graduated from the Cloud Native Computing Foundation. Kubernetes is an open-source framework for automating containerized application deployment, scaling, and administration (Brewer 2015). Customers may use Kubernetes to access private, public, and hybrid cloud infrastructure. Prometheus is an open source solution that enables monitoring and alerting features in applications. Prometheus metrics are built-in to the Kubernetes monitoring system (Brewer 2015; Saleh and Karslioglu 2021).

There are around 23 IEEE papers and 17 google scholar papers have been published over the past 5 years. The most cited article is “Observability in Kubernetes Cluster: Automatic Anomalies Detection using Prometheus”. A Docker Swarm is a collection of physical or virtual computers that have been configured to come together in a cluster and run the Docker application. Execute the Docker commands used to after a set of computers has been clustered together, but they'll be handled by the machines in the cluster (Mouat 2015). The activities of the cluster are overseen by a swarm manager, and machines that have joined the cluster are called the nodes. Docker Swarm is a container orchestration tool, which means it manages numerous containers across different host computers. One of the most important advantages of running a docker swarm is the high level of application availability it

provides (Schenker et al. 2019). A docker swarm generally consists of numerous worker nodes and at least one manager node, which is responsible for efficiently managing the worker nodes' resources and ensuring that the cluster runs smoothly (Krochmalski 2017). There are two sorts of services in Docker Swarm replicated and global. Swarm mode replicated services work by defining a number of replica jobs for the swarm management to allocate to available nodes (Soppelsa and Kaewkasi 2016). Global services work by scheduling one job to each available node that fulfills the services limitations and resource needs utilizing the swam manager.
Our institution is passionate about high quality evidence based research and has excelled in various fields (Parakh et al. 2020; Pham et al. 2021; Perumal, Antony, and Muthuramalingam 2021; Sathiyamoorthi et al. 2021; Devarajan et al. 2021; Dhanraj and Rajeshkumar 2021; Uganya, Radhika, and Vijayaraj 2021; Tesfaye Jule et al. 2021; Nandhini, Ezhilarasan, and Rajeshkumar 2020; Kamath et al. 2020). In the existing research they didn't identify the efficient platform for fault detection monitoring in multi clusters. The main aim of our project is to identify the efficient platform for fault detection monitoring in multi clusters by deploying the same set of clusters in different platforms and calculating the accuracy for fault detection.

Materials and Methods

The research work was performed in the Cloud Computing Laboratory, Department of Computer Science and Engineering, Saveetha School of Engineering, SIMATS (Saveetha Institute of Medical and Technical Sciences). The research work contains two groups. Group 1 is taken as novel kubernetes with prometheus and group 2 as docker containers. The novel kubernetes with prometheus and docker containers were executed and evaluated a different number of times with a sample size of 15 (Madushan 2021). The minimum power analysis for G-Power calculation is fixed at 0.8 and the maximum accepted error is fixed at 0.5. Same set of 20 containerized applications (Ifrah 2021) are used to identify the fault detection in multi clusters for each architecture to get the accuracy of each architecture.
Testing setup for this proposed system used a VMware workstation and Microsoft Azure. VMware workstation is used to create a guest OS to deploy applications. Hardware configuration for this proposed system is Intel core i5 8th gen processor and requires 4GB random access memory and 256GB Solid state drive used. The configuration of the system is the Windows 10 operating system.

Procedure for novel kubernetes with prometheus
Step-1: Connect to the Kubernetes Cluster
To establish cluster roles, connect to Kubernetes cluster and make sure to have admin capabilities.

Step-2: Create a Namespace & ClusterRole
We'll start by building a Kubernetes namespace for all of our monitoring components. All Prometheus kubernetes deployment objects are deployed to the default namespace if there is no dedicated namespace.

Step-3: Create a Config Map To Externalize Prometheus Configurations
The "prometheus.yaml" file contains all Prometheus settings, whereas "prometheus.rules" contains all Alertmanager alert rules. Don't have to rebuild the Prometheus image every time customize to change or delete a setting by externalizing Prometheus configs to a Kubernetes config map. To implement the updated settings, It must update the config map and restart the Prometheus pods.

Step-4: Create a Prometheus Deployment
The latest official Prometheus image from the Docker Hub is used in this deployment. Also, because this is a minimal configuration, do not use any permanent storage volumes for Prometheus storage. When configuring Prometheus for production use cases, make sure to include persistent storage.

Step-5: Setting Up Kube State Metrics
Many metrics will be provided through the Kube state metrics service that are not available by default. Please ensure that Kube state metrics are deployed to monitor all Kubernetes API objects, such as deployments, pods, jobs, and cron jobs.

Step-6: Connecting To Prometheus Dashboard
The deployed Prometheus dashboard may be seen in three distinct ways.

- Using Kubectl port forwarding
- Exposing the Prometheus deployment as a service with NodePort or a Load Balancer.
- Adding an Ingress object and having an Ingress controller deployed.

Algorithm for novel kubernetes with prometheus is given below:

Input: Application A to deploy

A=application
C=Cluster (*C)
N=Namespace
CR=Cluster role
Create Cluster (*C) = new Cluster (*C)
Connect to Cluster (*C)
Create Namespace (*N) = new Namespace (*N)
Create Cluster role (*CR)= new Cluster role (*CR)
Configure (*CF)
    Configuration of (Cluster,Namespace,Cluster role)
P=Prometheus
D=Deployment
Deployment (P)
    Create Deployment (*P) = new Deployment (*P)
        Connect to Cluster (*C)
        Create Namespace (*N)
        Create Cluster role (*CR)
        Configure (*CF)
E=Expose service
Expose service (P)
    Create ingress service (*E)=new ingress service (*E)
    Ingress Service (P)

Output:

Fault detection monitoring

Procedure for Docker Swarm

Step-1: Setting up a Docker Swarm runtime environment

Docker Swarm uses the Raft method to ensure high availability for management nodes, which requires an odd number of nodes to be registered as manager (3, 5, 7 and so on). This method chooses a leader from among the several management nodes. A cluster with three managers has a failure tolerance of one node, while a cluster with five nodes has a fault tolerance of up to two nodes.

Step-2: Setting up the registry using self-signed certificates

The registry server should be kept separate from the management and worker node servers.

Step-3: Building a Docker image for a LoopBack

Once the microservice is complete, use npm to package API and produce a deployable image that can be launched in a Docker Swarm environment.

Step-4: Adding Docker Swarm visualizer tool

Docker Swarm has a visualization tool that displays the cluster's structure as well as container statistics. The visualization tool runs as a separate container that can launch as a new Docker Swarm service.

Step-5: Adding a health check for LoopBack application

Docker Swarm has a health check method for each container. Swarm launches a new container to replace the terminated one whenever a container is terminated following a health check. Because a microservice operates in its own process and Docker Swarm does not communicate with processes running within containers, perform a service health check by adding a new endpoint to the API and without exposing it to API consumers in the Swagger file. This endpoint may be added to a LoopBack app by creating a new component, as shown below. This REST API is only accessible by the cluster.

Algorithm for Docker Swarm is given below:

Input: Application A to deploy

A=application
Enable Docker ()
SC=Swarm cluster
Create Swarm cluster (*SC) = new Swarm cluster (*SC)
C=self signed certificates

```
Registry (C)
        Create Directory (D)
        Current Directory ()
                Make Current Directory (D) = new Current Directory (D)
i=image
Create image (*i) = new image (*i)
Enable Docker Swarm Virtualization tool ()
        V=virtualization
        Create Service (*V) = new Service (*V)
        Expose Service (V)
H=Health check
Add Health Check to (Swarm Cluster (*SC))
```

Output:

Fault detection monitoring

Statistical Analysis

Statistical software used in the study is IBM SPSS version 26. The independent sample T-test calculation for analyzing equal variance, standard error, and levene's test are evaluated. Attributes like platform,Application number form the independent variables, Deployment Time, accuracy are dependent variables. Independent sample T-test has been carried out for evaluating the accuracy.

Results

In this proposed system it was observed that novel kubernetes with prometheus appear to be better than the docker container. Novel kubernetes with prometheus enables the continuous delivery and deployment of large, complex applications. Table 1 represents the outcome of the deployment. Table 2 shows the statistical calculation such as mean, standard deviation and standard error mean for Novel kubernetes with prometheus and Docker container respectively. The mean, standard deviation and standard error mean for Novel kubernetes with prometheus are 87.60, 3.647, 1.631 respectively. The mean, standard deviation and standard error mean for Docker containers are 84.20, 3.701, 1.655 respectively.

It is inferred that the mean accuracy for novel kubernetes is 87.60 which is greater than the mean accuracy of comparison architecture which is 84.20. Moreover, the mean accuracy value of Novel kubernetes with prometheus is around 87.60 which seems to be superior to the Docker container. In Table 3, it was observed that the Levens test for equality of variance and its significance for Novel kubernetes with prometheus is 0.036 and 0.036 respectively and standard error difference and confidence interval are lower than Docker containers.

Figure 1 represents the architecture of the proposed system. Architecture for fault detection in Multi-Clustermonitoring using different platforms. Docker, Cluster, Service are the important components in the architecture. Nodes are used to configure ports and store status in the architecture.

Figure 2 shows the bar graph analysis based on the accuracy of two architectures. Novel kubernetes with prometheus seems to appear better for the Multi-Cluster Monitoring for fault detection.

Discussion

The proposed Novel kubernetes with prometheus deployment provides better Multi-Cluster Monitoring for fault detection compared to Docker containers. The mean, standard deviation and standard error mean for Novel kubernetes with prometheus deployment are 87.60, 3.647, 1.631 respectively. The mean, standard deviation and standard error mean for Docker containers deployment are 84.20, 3.701, 1.655 respectively. It is inferred that the mean accuracy for novel kubernetes is 87.60 which is greater than the mean accuracy of comparison architecture which is 84.20. It was observed that the Levens test for equality of variance and its significance for Novel kubernetes with prometheus is 0.036 and 0.854 respectively and standard error difference and confidence interval are lower than Docker containers.

A systematic review on Multi-Cluster Monitoring for fault detection techniques presents around 21 studies, the analysis of various papers shows that Novel kubernetes with prometheus deployment is the most used technique for Multi-Clustermonitoring for fault detection. Yearly deployment time and maintainability is compared to the previous years with different engines and architectures. Since most of the techniques used prometheus the Novel kubernetes with prometheus deployment technique provides the best accuracy for Multi-Cluster Monitoring for fault detection.

Use Prometheus metrics and alerting capabilities to solve the challenge of observability in distributed systems, specifically the problem of fault detection in multi clustering (Burns, Beda, and Hightower 2019). Fault detection can aid developers and operations engineers in diagnosing issues, app performance regression, identifying resource misuse, gaining insight into customers, detecting data breaches, and preventing faults (Schenker et al. 2019). Forecasting metrics will allow us to spot faults earlier and more reliably, reducing the need for human intervention. Kubernetes and Prometheus are projects that have graduated from the Cloud Native Computing Foundation (Shkuro 2019). Kubernetes is an open-source framework for automating containerized application deployment, scaling, and administration (Poniszewska-Marańda and Czechowska 2021). Customers may use Kubernetes to access private, public, and hybrid cloud infrastructure. Prometheus is an open source solution that enables monitoring and alerting features in applications (Shkuro 2019; Ibryam and Huß 2019). Prometheus metrics are built-in to the Kubernetes monitoring system. Novel kubernetes with prometheus seems to appear better for the Multi-Cluster Monitoring for fault detection (Bastos and Araújo 2019).

The limitation of this research work is, it is limited to fault detection. Currently, it is not programmed to embed with other detections. Further, this research work can be improved by deploying a model that detects both fault and anomaly so that wait will be less and it can be easily manageable and scalable as in this research.

Conclusion

The Novel kubernetes with prometheus deployment technique monitor the Multi-Cluster for fault detection with better accuracy compared to Docker containers technique. Novel kubernetes with prometheus seems to appear better for the Multi-Cluster Monitoring for fault detection.

Declarations

Conflict of Interests

No conflict of interest in this manuscript.

Author Contribution

Author VSVK was involved in data collection, data analysis, manuscript writing. Author MK was involved in conceptualization, guidance and critical review of manuscript.

Acknowledgments

The authors would like to express their gratitude towards Saveetha School of Engineering, Saveetha Institute of Medical and Technical Sciences (Formerly known as Saveetha University) for providing the necessary infrastructure to carry out this work successfully.

Funding

Authors are grateful to the following organizations for their financial contributions, which enabled us to complete the study.

1. Renaissance Technologies, Chennai.
2. Saveetha University
3. Saveetha Institute of Medical and Technical Sciences.
4. Saveetha School of Engineering.

REFERENCE

1. Ahmadi, Seyed Hossein. n.d. “Fault Detection Automation in Distributed Control Systems Using Data-Driven Methods: SVM and KNN.” https://doi.org/10.36227/techrxiv.15029739.v1.
2. Arundel, John, and Justin Domingus. 2019. Cloud Native DevOps with Kubernetes: Building, Deploying, and Scaling Modern Applications in the Cloud. “O’Reilly Media, Inc.”
3. Bastos, Joel, and Pedro Araújo. 2019. Hands-On Infrastructure Monitoring with Prometheus: Implement and Scale Queries, Dashboards, and Alerting across Machines and Containers. Packt Publishing Ltd.
4. Brewer, Eric A. 2015. “Kubernetes and the Path to Cloud Native.” Proceedings of the Sixth ACM Symposium on Cloud Computing. https://doi.org/10.1145/2806777.2809955.
5. Burns, Brendan, Joe Beda, and Kelsey Hightower. 2019. Kubernetes: Up and Running: Dive into the Future of Infrastructure. “O’Reilly Media, Inc.”

6. Burns, Brendan, and Craig Tracey. 2018. Managing Kubernetes: Operating Kubernetes Clusters in the Real World. O'Reilly Media.
7. Devarajan, Yuvarajan, Beemkumar Nagappan, Gautam Choubey, Suresh Vellaiyan, and Kulmani Mehar. 2021. "Renewable Pathway and Twin Fueling Approach on Ignition Analysis of a Dual-Fuelled Compression Ignition Engine." Energy & Fuels: An American Chemical Society Journal 35 (12): 9930–36.
8. Dhanraj, Ganapathy, and Shanmugam Rajeshkumar. 2021. "Anticariogenic Effect of Selenium Nanoparticles Synthesized Using Brassica Oleracea." Journal of Nanomaterials 2021 (July). https://doi.org/10.1155/2021/8115585.
9. Ibryam, Bilgin, and Roland Huß. 2019. Kubernetes Patterns: Reusable Elements for Designing Cloud-Native Applications. O'Reilly Media.
10. Ifrah, Shimon. 2021. "Deploy Containerized Applications with." Getting Started with Containers in Google Cloud Platform. https://doi.org/10.1007/978-1-4842-6470-6_4.
11. Kamath, S. Manjunath, K. Sridhar, D. Jaison, V. Gopinath, B. K. Mohamed Ibrahim, Nilkantha Gupta, A. Sundaram, P. Sivaperumal, S. Padmapriya, and S. Shantanu Patil. 2020. "Fabrication of Tri-Layered Electrospun Polycaprolactone Mats with Improved Sustained Drug Release Profile." Scientific Reports 10 (1): 18179.
12. Krochmalski, Jaroslaw. 2017. Docker and Kubernetes for Java Developers. Packt Publishing Ltd.
13. Madushan, Dhanushka. 2021. Cloud Native Applications with Ballerina: A Guide for Programmers Interested in Developing Cloud Native Applications Using Ballerina Swan Lake. Packt Publishing Ltd.
14. Mouat, Adrian. 2015. Using Docker: Developing and Deploying Software with Containers. "O'Reilly Media, Inc."
15. Nandhini, Joseph T., Devaraj Ezhilarasan, and Shanmugam Rajeshkumar. 2020. "An Ecofriendly Synthesized Gold Nanoparticles Induces Cytotoxicity via Apoptosis in HepG2 Cells." Environmental Toxicology, August. https://doi.org/10.1002/tox.23007.
16. Parakh, Mayank K., Shriraam Ulaganambi, Nisha Ashifa, Reshma Premkumar, and Amit L. Jain. 2020. "Oral Potentially Malignant Disorders: Clinical Diagnosis and Current Screening Aids: A Narrative Review." European Journal of Cancer Prevention: The Official Journal of the European Cancer Prevention Organisation 29 (1): 65–72.
17. Perumal, Karthikeyan, Joseph Antony, and Subagunasekar Muthuramalingam. 2021. "Heavy Metal Pollutants and Their Spatial Distribution in Surface Sediments from Thondi Coast, Palk Bay, South India." Environmental Sciences Europe 33 (1). https://doi.org/10.1186/s12302-021-00501-2.
18. Pham, Quoc Hoa, Supat Chupradit, Gunawan Widjaja, Muataz S. Alhassan, Rustem Magizov, Yasser Fakri Mustafa, Aravindhan Surendar, Amirzhan Kassenov, Zeinab Arzehgar, and Wanich Suksatan. 2021. "The Effects of Ni or Nb Additions on the Relaxation Behavior of Zr55Cu35Al10 Metallic Glass." Materials Today Communications 29 (December): 102909.
19. Poniszewska-Marańda, Aneta, and Ewa Czechowska. 2021. "Kubernetes Cluster for Automating Software Production Environment." Sensors 21 (5). https://doi.org/10.3390/s21051910.
20. Saleh, Aly, and Murat Karslioglu. 2021. Kubernetes in Production Best Practices: Build and Manage Highly Available Production-Ready Kubernetes Clusters. Packt Publishing Ltd.
21. Sathiyamoorthi, Ramalingam, Gomathinayakam Sankaranarayanan, Dinesh Babu Munuswamy, and Yuvarajan Devarajan. 2021. "Experimental Study of Spray Analysis for Palmarosa Biodiesel-diesel Blends in a Constant Volume Chamber." Environmental Progress & Sustainable Energy 40 (6). https://doi.org/10.1002/ep.13696.
22. Schenker, Gabriel N., Hideto Saito, Hui-Chuan Chloe Lee, and Ke-Jou Carol Hsu. 2019. Getting Started with Containerization: Reduce the Operational Burden on Your System by Automating and Managing Your Containers. Packt Publishing Ltd.
23. Shkuro, Yuri. 2019. Mastering Distributed Tracing: Analyzing Performance in Microservices and Complex Systems. Packt Publishing Ltd.
24. Soppelsa, Fabrizio, and Chanwit Kaewkasi. 2016. Native Docker Clustering with Swarm. Packt Publishing Ltd.
25. Tesfaye Jule, Leta, Krishnaraj Ramaswamy, Nagaraj Nagaprasad, Vigneshwaran Shanmugam, and Venkataraman Vignesh. 2021. "Design and Analysis of Serial Drilled Hole in Composite Material." Materials Today: Proceedings 45 (January): 5759–63.
26. Uganya, G., Radhika, and N. Vijayaraj. 2021. "A Survey on Internet of Things: Applications, Recent Issues, Attacks, and Security Mechanisms." Journal of Circuits Systems and Computers 30 (05): 2130006.

Table 1. Fault detection time and accuracy for Novel Kubernetes With Prometheus technique and Docker Container.

Platform	Iteration no(n)	Fault Detection(in Sec)	Accuracy (in %)
Novel Kubernetes With Prometheus	1	19	89%
	2	15	91%
	3	21	86%
	4	27	82%
	5	18	90%
Docker Container	1	21	86%
	2	17	89%

	3	24	83%
	4	29	79%
	5	23	84%

Table 2. Group statistical analysis of Novel Kubernetes With Prometheus with mean value of 87.60 and Docker Container with mean value of 84.20 and similarly the results of Standard Deviation and Standard Error Mean are given.

T-Test:

Group Statistics

GROUP		**N**	**Mean**	**STD eviation**	**STD Error mean**
ACCURACY	**Novel Kubernetes With Prometheus**	20	87.60	3.647	1.631
	Docker Container	20	84.20	3.701	1.655

Table 3. Independent Sample T-test Results with confidence interval of 95% and level of significance greater than 0.05 (Novel kubernetes with prometheus seems to appear better for the Multi-Cluster Monitoring for fault detection).

		Levene's Test for Equality of Variance		**Levene's Test for Equality of Variance**						
	Equal Variances	**F**	**Sig.**	**t**	**df**	**Sig.(2-tailed)**	**Mean Difference**	**Std. Error Differ -ence**	**95% Confidence Interval of the Difference**	
									Lower	**Upper**
ACCURACY	**Assumed**	.036	.034	1.463	8	.182	3.400	2.324	-1.959	8.759
	Not Assumed			1.463	7.998	.182	3.400	2.324	-1.959	8.759

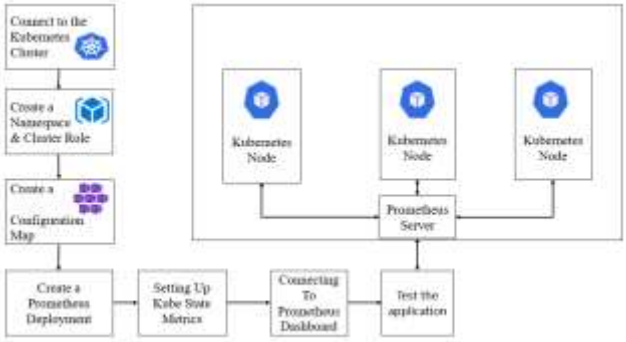

Fig. 1. Architecture for fault detection in Multi-Clustermonitoring using different platforms. Docker, Cluster, Service are the important components in the architecture. Nodes are used to configure ports and store status in the architecture.

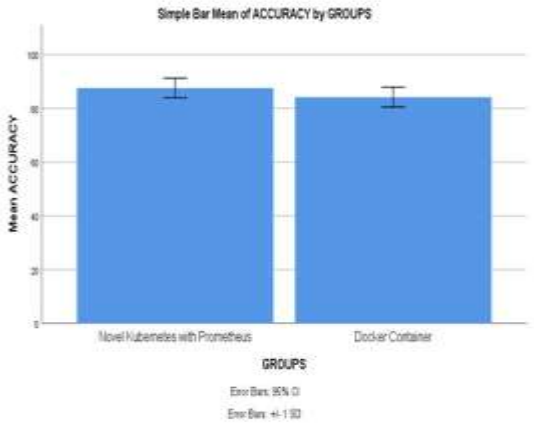

Fig. 2. Bar graph analysis of Novel Kubernetes With Prometheus and Docker Container. Graphical representation shows the mean Accuracy of 87.60% and 84.20% for the proposed platform (Kubernetes) and Docker respectively. X-axis: MZ vs VM, Y-axis : Mean Accuracy ± 1 SD.