Original Article

Estimating the Deployment Time for Cloud Applications using Novel Google Kubernetes Cloud Service over Microsoft Kubernetes Cloud Service

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 time to deploy cloud applications using novel google kubernetes cloud service over microsoft kubernetes cloud service. **Materials and Methods:** Sample groups that are considered in the project can be classified into two each has 20 samples, one for Novel google kubernetes cloud service and other for Microsoft kubernetes cloud service, which are tested using 0.80 for G-power to determine the sample size and for t-test analysis. 20 cloud applications have been used in each group for estimating deployment time. **Results:** The novel google kubernetes cloud service with efficient accuracy of 83.40%, which by far seems to be better than the Microsoft kubernetes cloud service which gives around 61.60%. The significance is around 0.222 ($p>0.05$) and therefore there is a statistical insignificant difference among the study group. **Conclusion:** Novel Google kubernetes cloud service seems to be better in deploying cloud applications over the Microsoft kubernetes cloud service.

Keywords: Microsoft Kubernetes, Novel Google Kubernetes, Container, Cloud application, Deployment Time.

DOI:10.47750/pnr.2022.13.S04.186

INTRODUCTION

GKE (Google Kubernetes Engine) is a controlled environment for deploying, maintaining, and scaling cloud applications on Google infrastructure. Multiple computers (particularly, Compute Engine instances) are brought together to form a cluster in the GKE environment. The Kubernetes open source cluster management technology powers GKE clusters. Kubernetes offers the tools that are needed to communicate with the clusters. To deploy and manage the apps, execute administrative activities, define policies, and check the health of the deployed workloads, It utilizes Kubernetes commands and resources (Sabharwal and Pandey 2020; Garbarino 2019). Kubernetes uses the same architectural principles as Google's famous services and offers the same benefits as automated administration, monitoring, and liveness probes for application containers, automatic scaling, and rolling upgrades, among other things (Riti 2018). When using a cluster to run the apps, It will use Google's 10+ years of expertise running production workloads in containers. Containerized apps are supported by GKE. These are apps that are bundled into platform-agnostic, isolated user-space instances, such as using Docker. These containers, whether for apps or batch operations, are referred to as workloads in GKE and Kubernetes (Hightower, Burns, and Beda 2017). First bundle a workload into a container before deploying it on a GKE cluster. Docker containers are supported by GKE. To help develop and serve application containers, Google Cloud provides continuous integration and continuous delivery technologies (Nguyen and Kim 2021; Burns, Beda, and Hightower 2019). Cloud Build may be used to create container images (such as Docker) from a variety of source code repositories, and Artifact Registry or Container Registry can be used to store and serve them. The method of operation in GKE is determined by the degree of flexibility, responsibility, and control the demand for clusters. Autopilot, which manages the whole cluster and node infrastructure, is one of the two modes of operation for GKE clusters (Ifrah 2020; Rose 2020). Autopilot gives a hands-off Kubernetes experience so it can concentrate on the workloads and only pay for the resources that need to operate the apps. Autopilot clusters are pre-configured with a production-ready cluster configuration. The second option is Standard, which gives complete control over the clusters and node architecture as well as node configuration freedom. When the Standard option chooses to

construct a cluster, it chooses the configurations needed for the production workloads and pays for the nodes utilized.

There are around 23 IEEE papers and 17 google scholar papers have been published over the past 5 years. The article with the most citations is "Deploying Microservice Based Applications With Kubernetes" . The Microsoft Azure Kubernetes Service is a container orchestration service based on the open source Kubernetes technology (Nguyen and Kim 2021). AKS can be used by an enterprise to manage key tasks like deploying, scaling, and managing Docker containers and container-based applications. AKS was released to the public in June 2018 and is primarily utilized by software developers and IT operations personnel (Salvaris, Dean, and Tok 2018). Kubernetes is the open source container orchestration tool, but it comes with a lot of cluster management overhead. AKS assists in the management of a large portion of the overhead, decreasing the complexity of deployment and management duties. AKS is for enterprises who wish to use the Azure architecture to build scalable apps utilizing Docker and Kubernetes (Salvaris, Dean, and Tok 2018; Chawla and Kathuria 2019). The Azure command-line interface (CLI), an Azure portal, or Azure PowerShell can all be used to set up an AKS cluster. Azure Resource Manager templates can also be used to build template-driven deployment choices (Riti 2018). A control plane is automatically established and configured when a user creates an AKS cluster. The control plane is a managed Azure resource that the user does not have direct access to (Riti 2018; Poniszewska-Marańda and Czechowska 2021). The Azure platform configures secure communication between the nodes and the control plane based on the size and number of nodes specified by the user. There is at least one node in an AKS cluster (Luksa 2017). The node resources utilized to assist the node work as part of a cluster are the central processing unit and memory. Node pools are created when nodes with similar configurations are clustered together. Two resource categories are also covered by AKS deployments. The Kubernetes service resource is in one group, while the node resource group is in the other. To establish and administer additional Azure resources, It will need a service principal or managed identity (Garbarino 2019).

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 deploying cloud applications. The main aim of our project is to identify the efficient platform for deploying cloud applications by deploying the same set of cloud applications in different platforms and calculating the deployment time.

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 google kubernetes and group 2 as Microsoft kubernetes. The novel google kubernetes and Microsoft kubernetes were executed and evaluated a different number of times with a sample size of 15 (Poniszewska-Marańda and Czechowska 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 cloud applications (Biggs, García, and Salanova 2019) are used to calculate the deployment time, management and scalability of the applications for each architecture to get the accuracy of each architecture.

Testing setup for this proposed system used novel google Kubernetes and Microsoft Azure. Vmware workstation is used to create a guest OS to deploy cloud 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 Google Kubernetes

Step-1: Create a repository

A container repository is a type of tech that stores and manages container images. A container registry is a set of container repositories that includes authentication, authorization, and storage management techniques.

Step-2: Building the container image

Docker images are accepted as an application deployment format by GKE. Don't forget to bundle the application source code into a Docker image before deploying it to GKE. The source code and a Dockerfile are required to create a Docker image. A Dockerfile specifies how the image should be created.

Step-3: Pushing the Docker image to Registry

The container image must be uploaded to a registry so that the GKE cluster may download and run it. Push the image to the repository after configuring credentials and tagging the local image.

Step-4: Creating a GKE cluster
Create a GKE cluster to execute the application using the Docker image saved in Artifact Registry. A GKE cluster is made up of a collection of Compute Engine VM instances that are all running Kubernetes, the open source cluster orchestration engine that runs GKE.

Step-5: Deploying the sample app to GKE
Applications are represented by Kubernetes as Pods, which are scalable units that hold one or more containers. In Kubernetes, the Pod is the smallest deployable unit. Pods are often deployed as a collection of replicas that may be scaled and distributed across the cluster. A Kubernetes Deployment is one approach to deploy a group of replicas.

Step-6: Exposing the sample app to the internet
While each pod has its own IP address, such addresses can only be accessed from inside the cluster. GKE Pods are also ephemeral, beginning and terminating based on scaling requirements. When a Pod fails due to an issue, GKE redeploys it automatically, each time issuing a new Pod IP address. Both of these issues are addressed by Kubernetes Services. Services combine Pods into a single static IP address that may be accessed from any Pod in the cluster. GKE additionally gives the static IP a DNS hostname.

Algorithm for deploying application using novel google kubernetes is given below:
Input: Application A to deploy

A=application
C=Cluster (*C)
S=Service (*S)
E=expose ()
P=port
Create cluster (*C)
Create cluster (*C) = new cluster (*C)
Create service (*S)
Create service (*S)= new service (*S)
Configure (*CF)
Configuration of (Cluster,Service,Port)
expose (*S)
Expose service (*S) = Service deployment.Kubectl ()
If (Deploy(A) == true)
Create cluster (C)
Configure (P)
Create Service (S)
Expose (Service (S))
End if
Else
"Error while setting up Kind"
End Else

Output:

Application Deployment
Deployment time

Procedure for Microsoft Kubernetes in Azure
Step-1: Create a resource group
A logical group in which Azure resources are deployed and managed is called an Azure resource group. When creating a resource group, It will be asked to name it and give it a location. This is the address:

- The place where the resource group metadata is stored.
- Specify another region during resource creation, The resources will operate in Azure.

Step-2: Enable cluster monitoring

Check the connection to see if Microsoft.OperationsManagement and Microsoft.OperationalInsights are active. Register Microsoft.OperationalInsights and Microsoft.OperationalManagement if they aren't already.

Step-3: Create AKS cluster
Create an AKS cluster using the --enable-addons monitoring parameter to enable Azure Monitor for containers using the az aks create command.

Step-4: Connect to the cluster

- Using the az aks install-cli command, install kubectl locally.
- Using the az aks get-credentials command, configure kubectl to connect to the Kubernetes cluster.
- Using the kubectl get command, verify the connection to the cluster. A list of cluster nodes is returned by this command.

Step-5: Deploy the application
A Kubernetes manifest file specifies the desired state of a cluster, such as which container images should be used. Using the kubectl apply command, deploy the application and enter the name of the YAML manifest.

Step-6: Test the application
A Kubernetes service exposes the application front end to the internet when it operates. It may take a few minutes to finish this process. Open a web browser to the external IP address of the service to view the deployed application in action.

Algorithm for deploying application using Microservice architecture based kubernetes in azure is given below:

Input: Application A to deploy

```
A=state_application
R=ResourceGroup (*G)
ResourceGroup (*G)
      Create ResourceGroup (*G) = new ResourceGroup (*G)
Enable Cluster monitor ()
      Register Microsoft.OperationsManagement ()
      Register Microsoft.OperationalInsights ()
K=AKS_Cluster (*AK)
     AKS_Cluster (*AK) =new AKS_Cluster (*AK)
Connect (R,K)
     Enable Kubectl ()
          Register az_aks.kubectl ()
     Configure Kubectl ()
          P=port
          Configure (R,K,P)
          if  (Configure (R,K,P) ==true)
               Return Deploy ()
          Else
               Error ()
Deployment (A)
    Create Deployment D=kubectl apply (A)
G=Availability (A)
Availability ()
    Check (Reaction Time)
              RT=Get (Reaction time)
    Check (Repair Time)
              RET=Get (Repair time)
    Check (Recovery Time)
              RCT=Get (Recovery time)
    Check (Outage Time)
             OT=Get (Outage time)
    compare (RT,RET,RCT,OT)
```

return (G)

Output:
Application Deployment
Deployment time

Statistical Analysis
IBM SPSS version 26 was used to conduct the research. The computation of the independent sample T-test for examining equal variance, standard error, and Levene's test is assessed. Platform, Application number, and Deployment Time are independent factors, whereas accuracy is a dependent variable. For testing the accuracy, an independent sample T-test was used.

Results

It was observed that Novel google kubernetes looks to be superior to Microsoft kubernetes in this suggested system. Microsoft kubernetes enables the continuous delivery and deployment of large, complex cloud 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 Google Kubernetes deployment and Microsoft Kubernetes deployment respectively. The mean, standard deviation and standard error mean for Novel Google Kubernetes deployment are 83.40, 4.159, 1.860 respectively. The mean, standard deviation and standard error mean for Microsoft kubernetes deployment are 61.60, 11.971, 5.354 respectively. It is inferred that the mean accuracy for novel google kubernetes is 83.40 which is greater than the mean accuracy of comparison which is 61.60. Moreover, the mean accuracy value of Novel Google Kubernetes is around 83.40 which seems to be superior to the Microsoft Kubernetes. In Table 3, it was observed that the Levens test for equality of variance and its significance for Novel Google Kubernetes is 1.758 and 0.222 respectively and standard error difference and confidence interval are lower than Microsoft Kubernetes. Figure 1 represents the architecture of the proposed system. Architecture for Deploying a Cloud application in kubernetes 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 represents the deployed cloud application in the kubernetes. Fig. 3 shows the bar graph analysis based on the accuracy of two architectures. Novel Google Kubernetes deployment seems to appear better for the deployment of cloud applications.

Discussion

The proposed Novel google kubernetes deployment provides better cloud application deployment with less deployment time compared to microsoft kubernetes deployment. The mean, standard deviation and standard error mean for novel google kubernetes deployment are 83.40, 4.159, 1.860 respectively. The mean, standard deviation and standard error mean for microsoft kubernetes deployment are 61.60, 11.971, 5.354 respectively. It is inferred that the mean accuracy for novel google kubernetes is 83.40 which is greater than the mean accuracy of comparison microsoft kubernetes which is 61.60. It was observed that the Levens test for equality of variance and its significance for Novel google kubernetes is 1.758 and 0.222 respectively and standard error difference and confidence interval are lower than Microsoft kubernetes.
A systematic review on cloud application deployment techniques presents around 32 studies, the analysis of various papers shows that novel google kubernetes deployment is the most used technique for cloud application deployment. Yearly deployment time and maintainability is compared to the previous years with different engines and architecture (Sabharwal and Pandey 2020). Since most of the deployment times are related to novel google kubernetes deployment, the technique provides the best accuracy for deployment of containerized applications. In the industry, the microservices architectural style has gotten a lot of attention, and the transition to this design is well underway (Ifrah 2020). The creation of software applications as a composition of loosely connected and independently deployable microservices is possible with this architectural style. Microservices are small applications with a single business purpose that communicate with other microservices via APIs (Sayfan 2018). Traditional monolithic architectures, where the application is a big and complex code base, can be overcome by the microservices architectural approach (Burns and Tracey 2018). To get the benefits of the microservices architecture style, one must employ technologies that are compatible with microservices' characteristics (Arundel and Domingus 2019). Containerization has become a common deployment technique for microservices. Microservice architecture based kubernetes deployment seems to appear better for the deployment of containerized applications (Poniszewska-Marańda and Czechowska 2021). The limitation of this research work is, it is limited to deploy containerized applications. Currently, it is not programmed to embed with other stateful

applications. Further, this research work can be improved by deploying a model that deploys more applications in less time so that wait will be less and it can be easily manageable and scalable as in this research.

Conclusion

The Novel google kubernetes deployment technique deploys the cloud application with better deployment time and maintenance compared to Microsoft kubernetes deployment technique.

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. Arundel, John, and Justin Domingus. 2019. Cloud Native DevOps with Kubernetes: Building, Deploying, and Scaling Modern Applications in the Cloud. "O'Reilly Media, Inc."
2. Biggs, John, Vicente Herrera García, and Jose Luis Calvo Salanova. 2019. Building Intelligent Cloud Applications: Develop Scalable Models Using Serverless Architectures with Azure. "O'Reilly Media, Inc."
3. Burns, Brendan, Joe Beda, and Kelsey Hightower. 2019. Kubernetes: Up and Running: Dive into the Future of Infrastructure. "O'Reilly Media, Inc."
4. Burns, Brendan, and Craig Tracey. 2018. Managing Kubernetes: Operating Kubernetes Clusters in the Real World. O'Reilly Media.
5. Chawla, Harsh, and Hemant Kathuria. 2019. Building Microservices Applications on Microsoft Azure: Designing, Developing, Deploying, and Monitoring. Apress.
6. 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.
7. 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.
8. Garbarino, Ernesto. 2019. Beginning Kubernetes on the Google Cloud Platform: A Guide to Automating Application Deployment, Scaling, and Management. Apress.
9. Hightower, Kelsey, Brendan Burns, and Joe Beda. 2017. Kubernetes: Up and Running: Dive Into the Future of Infrastructure. "O'Reilly Media, Inc."
10. Ifrah, Shimon. 2020. Getting Started with Containers in Google Cloud Platform: Deploy, Manage, and Secure Containerized Applications. Apress.
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. Luksa, Marko. 2017. Kubernetes in Action. Simon and Schuster.
13. 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.
14. Nguyen, Nguyen Dinh, and Taehong Kim. 2021. "Balanced Leader Distribution Algorithm in Kubernetes Clusters." Sensors 21 (3). https://doi.org/10.3390/s21030869.
15. 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.
16. 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.
17. 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.
18. Poniszewska-Marańda, Aneta, and Ewa Czechowska. 2021. "Kubernetes Cluster for Automating Software Production Environment." Sensors 21 (5). https://doi.org/10.3390/s21051910.
19. Riti, Pierluigi. 2018. Pro DevOps with Google Cloud Platform: With Docker, Jenkins, and Kubernetes. Apress.
20. Rose, Richard. 2020. Hands-On Serverless Computing with Google Cloud: Build, Deploy, and Containerize Apps Using Cloud Functions, Cloud Run, and Cloud-Native Technologies. Packt Publishing Ltd.
21. Sabharwal, Navin, and Piyush Pandey. 2020. Pro Google Kubernetes Engine: Network, Security, Monitoring, and Automation Configuration. Apress.
22. Salvaris, Mathew, Danielle Dean, and Wee Hyong Tok. 2018. Deep Learning with Azure: Building and Deploying Artificial Intelligence Solutions on the Microsoft AI Platform. Apress.
23. 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.
24. Sayfan, Gigi. 2018. Mastering Kubernetes: Master the Art of Container Management by Using the Power of Kubernetes, 2nd Edition. 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.

TABLES AND FIGURES

Table 1. Application deployment time and accuracy for Novel Google Kubernetes Cloud Service deployment and Microsoft Kubernetes Cloud Service deployment.

Platform	Iteration no(n)	Deployment Time(in Sec)	Accuracy (in %)
Novel Google Kubernetes Cloud Service	1	21	86%
	2	17	89%
	3	24	83%
	4	29	79%
	5	25	82%
Microsoft Kubernetes Cloud Service	1	38	62%
	2	29	79%
	3	43	57%
	4	54	46%
	5	36	64%

Table 2. Group statistical analysis of Novel Google Kubernetes Cloud Service with mean value of 83.40 and Microsoft Kubernetes Cloud Service with mean value of 61.60 and similarly the results of Standard Deviation and Standard Error Mean are given.

T-Test:

Group Statistics

GROUP		N	Mean	STD Deviation	STD Error mean
ACCURACY	Novel Google Kubernetes Cloud Service	20	83.40	4.159	1.860
	Microsoft Kubernetes Cloud Service	20	61.60	11.971	5.354

Table 3. Independent Sample T-test Results with confidence interval of 95% and level of significance greater than 0.05 (Novel google kubernetes deployment technique seems to appear better for the deployment of cloud application).

		Levene's Test for Equality of Variance		Levene's Test for Equality of Variance						
									95% Confidence Interval of the Difference	
	Equal Variances	F	Sig.	t	df	Sig.(2-tailed)	Mean Difference	Std. Error Differ -ence	Lower	Upper
ACCURACY	Assumed	1.578	.222	3.847	8	.005	21.800	5.667	8.731	34.869
	Not Assumed			3.847	4.952	.012	21.800	5.667	7.189	36.411

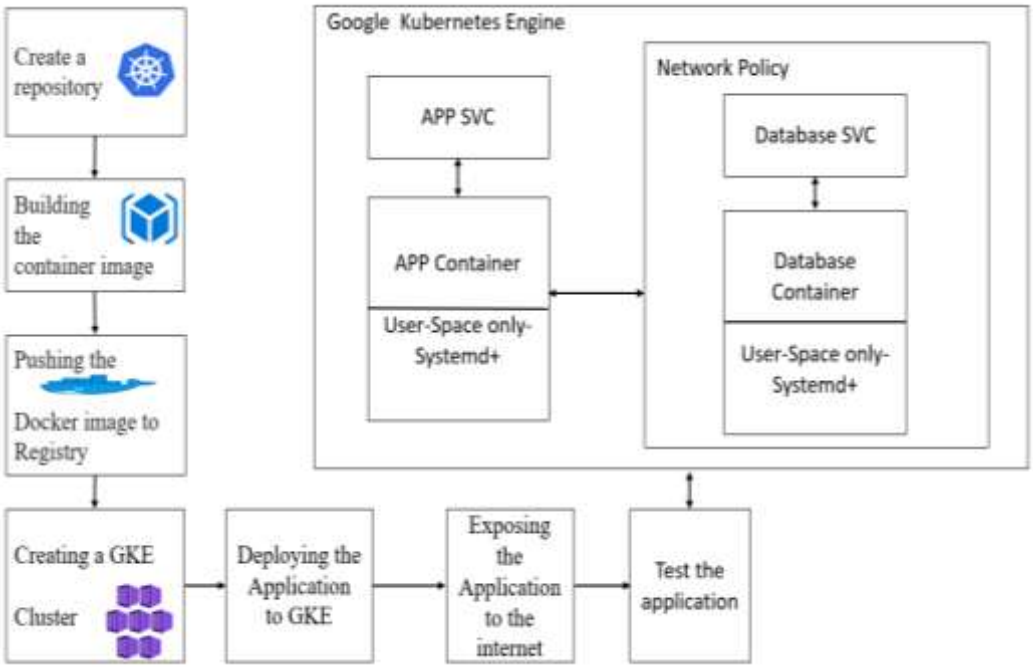

Fig. 1. Architecture for Deploying a Cloud application in kubernetes using different platforms. Docker, Cluster, Service are the important components in the architecture. Nodes are used to configure ports and store status in the architecture.

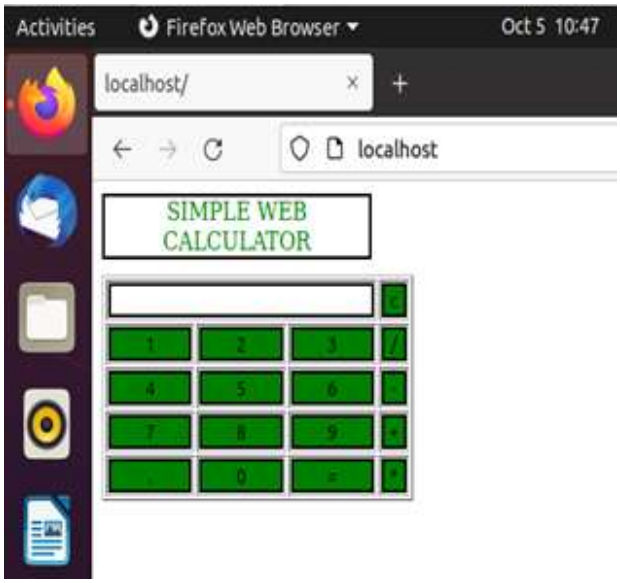

Fig. 2. Deployed application in kubernetes using Novel google kubernetes deployment and Microsoft kubernetes deployment respectively.

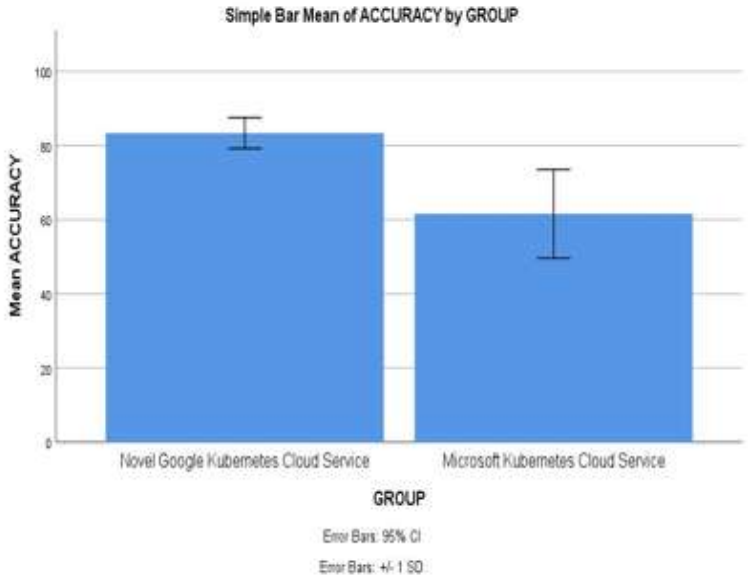

Fig. 3. Bar graph analysis of Novel Google Kubernetes deployment and Microsoft Kubernetes deployment. Graphical representation shows the mean Accuracy of 83.40% and 61.60% for the proposed platform (Google Kubernetes) and Microsoft Kubernetes respectively. X-axis : MZ vs VM, Y-axis : Mean Accuracy ± 1 SD.