Original Article

Estimating the Time to Deploy Containerized Application using Novel Kubernetes based Microservice Architecture over VMware Workstation based Virtualization Architecture

Sai Vimal Kumar V[1], K Malathi[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 containerized applications using novel kubernetes based microservice architecture over VMware workstation based virtualization architecture. **Materials and Methods:** Sample groups that are considered in the project can be classified into two each has 20 samples, one for Microservice architecture and other for Virtualization architecture, which are tested using 0.80 for G-power to determine the sample size and for t-test analysis. 20 containerized applications have been used in each group for estimating deployment time. **Results:** The azure kubernetes based feature of Microservice architecture with efficient accuracy of 60%, which by far seems to be better than the Virtualization architecture which gives around 54.40%. The significance is around 0.482 ($p<0.05$) and therefore there is a statistical insignificant difference among the study group. **Conclusion:** Novel Kubernetes based microservice architecture seems to be better in deploying containerized applications over the VMware workstation based virtualization architecture.

Keywords: VMware Workstation, Novel Kubernetes, Microservice Architecture, Containers, Virtualization Architecture, Containerized application, Deployment Time.

DOI:10.47750/pnr.2022.13.S04.183

INTRODUCTION

In the industry, the microservices architectural style has gotten a lot of attention, and the transition to this design is well underway. The creation of software applications as a composition of loosely connected and independently deployable microservices is possible with this architectural style ("MICROSERVICES - Lightweight Service Descriptions for REST Architectural Style" 2010). Microservices are small applications with a single business purpose that communicate with other microservices via APIs (De Santis et al. 2016). In Traditional monolithic architectures, where the application is a big and complex code base, can be overcome by the microservices architectural approach (Daya et al. 2016). To get the benefits of the microservices architecture style, one must employ technologies that are compatible with microservices' characteristics. There are many applications of containerization, mainly it is a common deployment technique for microservices, and Docker is the most popular container platform for deploying and running microservices (i.e., code, libraries, settings, etc.) (Arundel and Domingus 2019). Containers are unaware of each other since they are segregated. As a result, an orchestration platform to control container deployment is required. Novel Kubernetes is the leading platform for containerized application deployment, scaling, and management. It frees developers from the burden of integrating robustness in their applications, allowing them to concentrate on the business logic (Vayghan et al. 2019). As a result, it has grown in popularity as a platform for deploying microservice-based applications.

In order to speed up procedures and lower the cost of developing and deploying containerized applications, many architectures and paradigms are being investigated. Container-based virtualization is becoming a feasible option for application deployment (Soh et al. 2020). 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" . Kubernetes offers a variety of options for microservice-based applications. Deployment controllers can be used to deploy microservice-based applications. In any case, the state information is assumed to be maintained on permanent storage that is handled outside of Kubernetes. A virtualization-based application can be deployed in a variety of ways. Virtualization was first used to deploy numerous server

applications on a single physical device. For example, a Citrix server, a Web server, and an Oracle database server can all be run on the same physical server by using three virtual servers (Xiang, He, and Wen 2022). Server virtualization was also necessitated by programmes that couldn't make use of the servers' increased memory availability, necessitating the hosting of several application instances, one per virtual server (Olzak et al. 2010). Organizations used to run apps on physical servers. There was no way to specify resource boundaries for programmes on a physical server, which caused resource allocation concerns. When multiple apps are running on a physical server, for example, one programme may absorb the majority of the resources, leading the other apps to function poorly. One possibility is to run each software on its own physical server. However, this did not expand since resources were underutilized and maintaining a large number of physical servers was costly (Kleppmann 2017). Virtualization was introduced as a solution. It enables the operating of several Virtual Machines (VMs) on the CPU of a single physical server. Virtualization allows programs to be isolated between virtual machines (VMs) and provides a level of security by preventing one application's information from being freely accessed by another. Virtualization improves resource efficiency on a physical server, improves scalability by allowing applications to be quickly added or changed, reduces hardware costs, and much more. It can offer a group of physical resources as a cluster of disposable virtual machines using virtualization (Pearson and Yee 2012). On top of the virtualized hardware, each VM is a full machine with all of its components, including its own operating system. At the app layer, containers are an abstraction that bundles code and dependencies together. Multiple containers can share the same operating system and execute on the same machine.

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 applications. In the research we compare the kubernetes and vmware to identify which platform provides best deployment and management services at least cost and easy maintenance. In order to identify the best platform we took the same set of applications for each platform and compared the deployment time and accuracy of the applications to identify the efficient platform. The kubernetes uses a microservice architecture and vmware uses a virtualization architecture. Microservice architecture allows a large application to be broken down into smaller, self-contained components, each with its own set of responsibilities. Virtualization architecture allows several consumers and organizations to share a single physical instance of a resource or application at the same time. This is accomplished by giving a physical storage a logical name and giving a pointer to that physical resource on demand. The main aim of our project is to identify which is the efficient platform for deploying applications among kubernetes and VM ware workstation, by deploying the same set of applications and calculating the deployment time. Thus the goal is to find an efficient way of deploying the applications in cloud microservice architecture.

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 proposed work contains two groups. Group 1 is taken as Microservice Architecture Based novel Kubernetes and Group 2 as Virtualization Architecture Based Kubernetes. The Microservice Architecture Based novel Kubernetes and Virtualization Architecture Based Kubernetes were executed and evaluated a different number of times with a sample size of 20 (Buelta 2019) . The minimum power analysis for G-Power calculation is fixed at 0.8 and the maximum accepted error is fixed at 0.05. Same set of 20 containerized applications 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 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 VMware workstation based virtualization architecture

Step-1: KIND Set-up

- **Docker:** Docker must be installed in order for “kIND” to work. On Linux,use the package manager that comes with your operating system, such as apt on Ubuntu to install docker.
- **Kubectl:** Once the cluster is up and running, you'll need to use the kubectl command to interface with it. With your Linux distribution's packages manager,find kubectl install instructions, including methods to install it.
- **Kind:** Finally KIND can be installed using the package manager of the linux operating systemz

Step-2: Creating your cluster
After you've installed all of these components, you're ready to start building your local Kubernetes cluster. Kind uses a Docker container to deploy a Kubernetes instance. It's preferable to stop any other containers that are operating on your system because they may conflict with the ports.

Step-3: Deploy an application
To execute a process now that the cluster is up and running. All workloads in Kubernetes are described in a simple yaml format file called a "manifest." So, in order to run something on the cluster, first create a yaml file that describes what the system wants to do.

Step-4: Expose the service
The process is ongoing. Kubernetes offers a scalable service layer for routing connections to the containers it manages. It must specify the ports that Pod will map onto the container when executing it. Then construct a 'Service' Kubernetes resource that will route requests to the processes operating in your Pods.
Algorithm for deploying application using Vmware Workstation based kubernetes is given below:

Input: Application A to deploy

```
A=application
C=Cluster(*C)
S=Service(*S)
E=expose()
P=port
Kindsetup(A)    //Setup kind Service for deployment
        Enable Docker()
        Kubectl(get nodes)
Create cluster(*C)       //Function to create cluster
        Create cluster(*C)=new cluster(*C)
Create service(*S)       //Funcion to create Service
       Create service(*S)=new service(*S)
configure(*CF)
       Configuration of(Cluster,Service,Port)
expose(*S)
       Expose service(*S)=Service deployment.Kubectl()
if( kindsetup(A) == true)
Create cluster(C)     //Create Cluster
Configure(P)          //Configure ports
Create Service(S)    //Create Service
Expose(Service (S) )       //Expose Service
End if
Else
    "Error while setting up Kind"
End Else
```

Output:

```
Application Deployment
Deployment time
```

Procedure for Microservice Architecture Based 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, you'll be asked to name it and give it a location. This is the address:

- The place where your resource group metadata is stored.
- Specify another region during resource creation, your 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 your Kubernetes cluster.
- Using the kubectl get command, verify the connection to your 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 your 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 your 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=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)
```

Output:

Application Deployment
Deployment time

Statistical Analysis

Statistical software used in the study is IBM SPSS version 26 (George and Mallery 2018). 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 Microservice architecture based novel kubernetes appears to be better than the Vmware virtualization based kubernetes. Microservice architecture based kubernetes 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 Microservice architecture based kubernetes deployment and Vmware virtualization based kubernetes deployment respectively. The mean, standard deviation and standard error mean for Microservice architecture based kubernetes deployment are 60.00, 9.301, 4.159 respectively. The mean, standard deviation and standard error mean for Vmware virtualization based kubernetes deployment are 54.40, 9.317, 4.167 respectively. It is inferred that the mean accuracy for novel kubernetes is 60 which is greater than the mean accuracy of comparison architecture which is 54. Moreover, the mean accuracy value of Microservice architecture based kubernetes is around 60.00 which seems to be superior to the Vmware virtualization based kubernetes.

In Table 3, it was observed that the Levens test for equality of variance and its significance for Microservice architecture based kubernetes is 0.001 and 0.0482 respectively and standard error difference and confidence interval are lower than Vmware virtualization based kubernetes.

Fig. 1 represents the architecture of the proposed system. Fig. 2 represents the deployed application in the kubernetes. Fig. 3 shows the bar graph analysis based on the accuracy of two architectures. Microservice architecture based kubernetes deployment seems to appear better for the deployment of containerized applications.

Discussion

The proposed Microservice architecture based kubernetes deployment provides better containerized application deployment with less deployment time compared to Vmware virtualization based kubernetes deployment. The mean, standard deviation and standard error mean for Microservice architecture based kubernetes deployment are 60.00, 9.301, 4.159 respectively. The mean, standard deviation and standard error mean for Vmware virtualization based kubernetes deployment are 54.40, 9.317, 4.167 respectively. It is inferred that the mean accuracy for novel kubernetes is 60 which is greater than the mean accuracy of comparison architecture which is 54.

A systematic review on containerized application deployment techniques presents around 32 studies, the analysis of various papers shows that novel Microservice architecture based kubernetes deployment is the most used technique for containerized application deployment. Yearly deployment time and maintainability is compared to the previous years with different engines and architectures. Since most of the deployment times are related to Microservice architecture based kubernetes deployment, the technique provides the best accuracy for deployment of containerized applications (Modi 2017).

In the industry, the microservices architectural style has gotten a lot of attention, and the transition to this design is well underway. The creation of software applications as a composition of loosely connected and independently deployable microservices is possible with this architectural style (Buelta 2019). Microservices are small applications with a single business purpose that communicate with other microservices via APIs (Richardson 2018). Traditional monolithic architectures, where the application is a big and complex code base, can be overcome by the microservices architectural approach (Thota 2021). To get the benefits of the microservices architecture style, one must employ technologies that are compatible with microservices' characteristics. Containerization has become a common deployment technique for microservices (Rosso et al. 2021). Microservice architecture based kubernetes deployment seems to appear better for the deployment of containerized applications. 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(Burns, Beda, and Hightower 2019). 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 Microservice architecture based kubernetes deployment technique deploys the containerized application with better deployment time and maintenance compared to Vmware virtualization based kubernetes deployment technique.

Declarations

Conflict of Interests

No conflict of interest in this manuscript.

Author Contribution

Author SVKV was involved in data collection, data analysis, manuscript writing. Author KM 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
We thank the following organizations for providing financial support that 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.

REFERENCES

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. Buelta, Jaime. 2019. Hands-On Docker for Microservices with Python: Design, Deploy, and Operate a Complex System with Multiple Microservices Using Docker and Kubernetes. Packt Publishing Ltd.
3. Burns, Brendan, Joe Beda, and Kelsey Hightower. 2019. Kubernetes: Up and Running: Dive into the Future of Infrastructure. "O'Reilly Media, Inc."
4. Daya, Shahir, Nguyen Van Duy, Kameswara Eati, Carlos M. Ferreira, Dejan Glozic, Vasfi Gucer, Manav Gupta, et al. 2016. Microservices from Theory to Practice: Creating Applications in IBM Bluemix Using the Microservices Approach. IBM Redbooks.
5. De Santis, Sandro, Luis Florez, Duy V. Nguyen, Eduardo Rosa, and I. B. M. Redbooks. 2016. Evolve the Monolith to Microservices with Java and Node. IBM Redbooks.
6. George, Darren, and Paul Mallery. 2018. IBM SPSS Statistics 25 Step by Step: A Simple Guide and Reference. Routledge.
7. Kleppmann, Martin. 2017. Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems. "O'Reilly Media, Inc."
8. "MICROSERVICES - Lightweight Service Descriptions for REST Architectural Style." 2010. Proceedings of the 2nd International Conference on Agents and Artificial Intelligence. https://doi.org/10.5220/0002720105760579.
9. Modi, Ritesh. 2017. Azure for Architects: Implementing Cloud Design, DevOps, IoT, and Serverless Solutions on Your Public Cloud. Packt Publishing Ltd.
10. Olzak, Thomas, James Sabovik, Jason Boomer, and Robert M. Keefer. 2010. Microsoft Virtualization: Master Microsoft Server, Desktop, Application, and Presentation Virtualization. Syngress.
11. Pearson, Siani, and George Yee. 2012. Privacy and Security for Cloud Computing. Springer Science & Business Media.
12. Richardson, Chris. 2018. Microservices Patterns: With Examples in Java. Simon and Schuster.
13. Rosso, Josh, Rich Lander, Alex Brand, and John Harris. 2021. Production Kubernetes. "O'Reilly Media, Inc."
14. Soh, Julian, Marshall Copeland, Anthony Puca, and Micheleen Harris. 2020. "Developing and Deploying Azure-Based Applications." Microsoft Azure. https://doi.org/10.1007/978-1-4842-5958-0_20.
15. Thota, Narendranath Reddy. 2021. Mastering Kubernetes Automation: Automate Deployments Using Helm , Kubernetes Operators and Kubernetes Client Libraries. Narendranath Reddy Thota.
16. Vayghan, Leila Abdollahi, Mohamed Aymen Saied, Maria Toeroe, and Ferhat Khendek. 2019. "Microservice Based Architecture: Towards High-Availability for Stateful Applications with Kubernetes." 2019 IEEE 19th International Conference on Software Quality, Reliability and Security (QRS). https://doi.org/10.1109/qrs.2019.00034.
17. Xiang, Qian, Yuntao He, and Donghai Wen. 2022. "Adaptive Deep Learning-Based Neighborhood Search Method for Point Cloud." Scientific Reports 12 (1): 2098.

Tables and Figures

Table 1. Application deployment time and accuracy for Microservice architecture based kubernetes deployment technique and Vmware virtualization based kubernetes deployment technique.

ITERATION NO (n)	**Novel kubernetes based Microservice Architecture**		**Vmware Workstation based Virtualization architecture**	
	Time (in Sec)	**Accuracy(%)**	**Time (in Sec)**	**Accuracy(%)**

1	38	62	41	59
2	29	71	34	66
3	43	57	49	51
4	54	46	59	41
5	36	64	45	55

Table 2. Group statistical analysis of Novel Kubernetes with Mean Value Of 60.00 and VMware Workstation Kubernetes with mean value of 54.40 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 kubernetes based Microservice Architecture**	20	60.00	9.301	4.159
	Vmware workstation Kubernetes based Virtualization architecture	20	54.40	9.317	4.167

Table 3. Independent Sample T-test Results with confidence interval of 95% and level of significance greater than 0.05 (Azure Microservice architecture based kubernetes deployment seems to appear better for the deployment of containerized application).

		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**	.001	0482	-.951	8	.001	5.600	5.887	-7.976	19.176
	Not Assumed			-.951	8.000	.002	5.600	5.887	-7.976	19.176

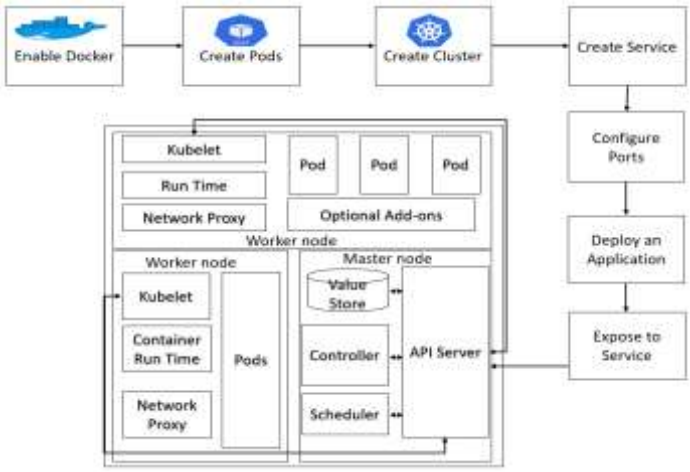

Fig. 1. Architecture for Deploying a Containerized application in kubernetes using different platforms. Docker, Cluster, Service are the important components in the architecture. Nodes are used to configure ports and store status in the architecture.

Fig. 2. Deployed application in kubernetes using Microservice architecture based kubernetes deployment and Vmware virtualization based kubernetes deployment respectively.

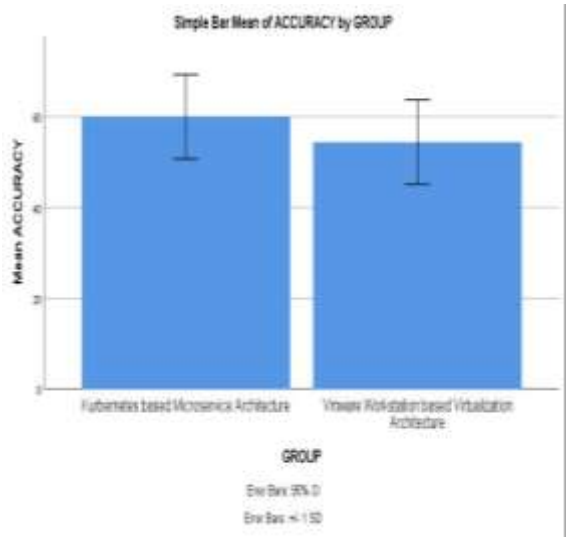

Fig. 3. Bar graph analysis of Novel Kubernetes based microservice architecture and VMware workstation based virtualization architecture. Graphical representation shows the mean Accuracy of 60% and 54.40% for the proposed platform (Microsoft Azure) and Vmware respectively. X-axis : MZ vs VM, Y-axis : Mean Accuracy ± 1 SD.