

Cloud Native Computing: Deploying and Managing Applications with Kubernetes

Dr. Uma Pujeri¹, Dr. Trupti Baraskar², Rasi Bansal³, Preeti Savant⁴ and Omkar Pujeri⁵
¹⁻⁵Dr. Vishwanath Karad MIT World Peace University, School of Computer Engineering and Technology, Kothrud
Email: uma.pujeri@mitwpu.edu.in

Abstract— In a globalized world at the present time, the realization of improving efficiency and minimizing costs during different stages of the application lifecycle, have confused software engineers on how they should deploy their applications and workload. It is noticed that many native applications of cloud computing and services comprises a lot of micro-services into the bargain traditional monolithic services. These services have almost been deprecated in application life-cycle mostly because of problems like maintaining, scaling, and other dynamics associated with it. Furthermore, developers focus their attention on containers. Machine resources are more effectively used by containers than a virtual machine (VM) and reduces the overhead of using a VM. Moreover, containers facilitate developers to create, deploy or delete applications without much effort. In order to manage multiple container applications, some of the container orchestration platforms like Kubernetes, Docker Swarm are introduced by cloud providers. In this paper, a study is conducted on how to deploy and manage applications using containerization with Kubernetes, from a cloud administrator's perspective

Index Terms— Containers, Kubernetes, Orchestration, Docker, Micro- services, Virtual Machine, Cloud-native, Deployment

I. INTRODUCTION

Kubernetes, which is also referred to as k8s, is used to automate software deployments. It was originally designed by Google and now managed by CNCF. It is popular due to its open-source nature and containerization benefits. Applications that run in isolated environments (also known as containers) are called containerized applications. These isolated environments wrap up an application which also contains all the application dependencies, including config files, several binaries and also language system libraries.

We can scale containerized applications with Kubernetes, manage services declaratively, ensuring that deployed applications are always up and running.

II. LITERATURE REVIEW

In this review, we will review some of literature and papers related to Kubernetes and are published in the years 2017-2022. The summary of all the mentioned papers will be presented in a table towards the end of this section. V. Model used a Net-based model for analyzing how Kubernetes performance varies. The purpose of this paper is to understand the overheads associated with the containers inside kubernetes. They also focused on analysis of abstraction in nodes and pods and various k8s based performance models. Q. Nguyen proposed a proxy which was

named as Resource Adaptive Proxy or RAP for edge computing environments which are k8s based. handling of requests locally and monitoring status of worker resources during the operation of load balancing became possible as a result of their work. This paper was majorly focused on architecture of Edge Computing which is k8s based and the workflow algorithm (RAP). S. Fiori proposed Kubernetes - RT. Their primary focus was on container based deployments of software components. They primarily targeted architecture of how scheduling takes place in a container. D. Jorge-Martinez proposed KCCS algorithm, which stands for k8s container scheduling strategy. This algorithm makes use of Machine Learning and Artificial Intelligence which is used for enhancing the efficiency of how a container is scheduled inside kubernetes. S. Taherizadeh's focus was to improve how auto-scaling works inside Kubernetes.

Various factors that were targeted to different patterns for different workloads were also discussed and possible solutions were proposed. L. Abdollahi Vayghan et al proposed a state controller with high-availability in their paper. The primary focus of the paper is to improve the already existing state of the availability of kubernetes and to handle failures of pod and enhancing the feature of auto recovery.

even if they have been submitted for publication, should be cited as "unpublished" [4]. Papers that have been accepted for publication should be cited as "in press" [5]. In a paper title, capitalize the first word and all other words except for conjunctions, prepositions less than seven letters, and prepositional phrases.

For papers published in translated journals, first give the English citation, then the original foreign-language citation [6]. All the listed references must cite in the text.

Authors	Title	Methodology
Quang Minh Nguyen et al	Load-Balancing of Kubernetes-Based Edge Computing Infrastructure Using Resource Adaptive Proxy	Resource Adaptive Proxy
Stefano Fiori's et al	RT-kubernetes: containerized real-time cloud computing	Real time Container Scheduling architecture
Diaz Jorge-Martinez's et al	Artificial Intelligence-Based Kubernetes Container for Scheduling Nodes of Energy Composition	Kubernetes container Scheduling strategy/ (KCSS) based on Artificial Intelligence.(AI)
Victor Medel et al	Characterising resource management performance in Kubernetes	Petri Net-based Performance model
Nawailuo Zhou et al	Container orchestration on HPC systems through Kubernetes	Torque - Operator
Salman Taherizadeh et al	Key influencing factors of the Kubernetes auto-scaler for computing-intensive microservice-native cloud-based applications	Auto Scaling Factor
Dylan Bannon et al	DeepCell Kiosk: scaling deep learning-enabled cellular image analysis with Kubernetes	Deep Cell Kiosk
Leila Abdollahi Vayghan et al	A Kubernetes controller for managing the availability of elastic microservice-based stateful applications	High Availability State Controller
Nguyen Dinh Nguyen	Balanced Leader Distribution Algorithm in Kubernetes Clusters	Balanced Leader Distribution(BLD) Algorithm

Figure 1 : Paper on Kubernetes with Methodology

III. KUBERNETES ARCHITECTURE

A Kubernetes cluster is the primary component of the Kubernetes (k8s) architecture. It consists of a control plane (can be one or many, but at least one), and one or more worker nodes. Both the control planes and nodes can be instances or VM's in the cloud

Control Plane (Master Node): The control plane or the master node (as it is rightly called) is responsible for managing the pods and scheduling of Kubernetes. In real-world environments, the master node is typically run through multiple hosts / machines, and a cluster can run multiple nodes, which provides high availability, zero downtime, and auto scaling. The control plane gets the input from a cloud CLI, GUI, or API request.

A Node: Also known as worker node or compute node. It is a physical or virtual machine that provides the services necessary to run applications inside containers. A cluster requires a minimum of one worker node, but normally has a couple of them. They contain pods, which are the components of the workload. They are scheduled and managed to run on nodes. You can scale up and scale down clusters by adding and removing nodes.

Kubernetes Control Plane Components: The components of K8s control plane are responsible to take universal decisions about the k8s cluster, and also detect, respond and react to events that happen in a cluster. Kubernetes relies on the control plane for several administrative services running on it. These services are responsible to

manage cluster component communication, workload scheduling, cluster state persistence, self-healing, auto scaling, and load balancing.

API Server (*kube-apiserver*): The API server is a server which expose the k8s API to the outside world. It is the entry point for REST API or the kubectl CLI command. It is responsible for tracking the state of cluster components and also managing the interaction between them. It scales horizontally. It takes as input, YAML or JSON manifest files. It is also responsible for validating and processing the requests made via the Kubernetes API.

Etc (key-value store): etcd is known as the brain of Kubernetes. It is distributed, consistent and stores value in key-value pair type. It is stateful just like other databases and has persistent storage that stores all data of a k8s cluster (cluster state and config mostly). It is also known as the source of truth for the cluster.

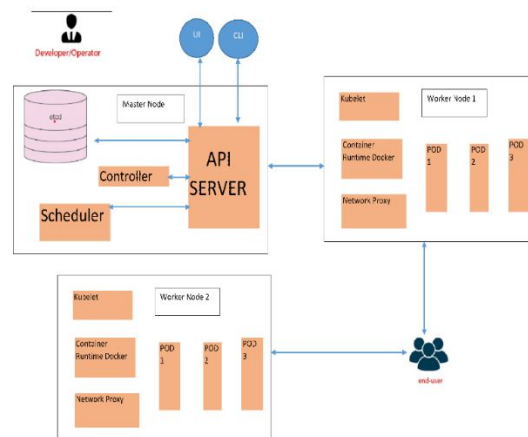


Figure 2: Kubernetes Architecture Kubernetes model workflow

Scheduler (*kube-scheduler*): A scheduler's responsibility is to schedule pods on worker nodes. It mainly watches the Kubernetes api-server for pods which have been created recently and have no node assigned to them, and chooses a healthy node for them and schedules the pods on that specific node. If there are no suitable nodes for the pods to schedule on, the pods go in a pending state till a worthy node is found to schedule the pod on. It also watches API Server for new worker tasks.

Controller Manager (*kubernetes-controller-manager*): The job of the CM is to keep watching the state of the components via the k8s API server. It makes sure that the current state of the components matches the desired state. If not, it's the responsibility of the CM to run a reconcile loop and follow the declarative approach to run the controller process.

Cloud Controller Manager (CCM): The cloud controller manager integrates with the underlying cloud technologies in a kubernetes cluster when the cluster is up and running in a cloud environment. CCM only runs controllers that are specific to the cloud provider you are using. It lets you link the cluster to the cloud provider's API.

kubelet: Kubelet is run on every node of the network. It acts as a layer or bridge between the node and the k8s api server. The responsibility of kubelet is to verify that the applications in isolated environments running on a node are healthy.

kube-proxy: It is the networking component of a cluster that plays an important role in the networking of services. It is a proxy that should run on each and every node in the kubernetes cluster. It maintains network rules on all the nodes present. These network rules can allow network communication to pods. It ensures each pod gets a unique IP address and makes it possible that all containers in a pod to share a single IP. It facilitates Kubernetes networking services and load-balancing across all pods in a service. It is also responsible to deal with individual host sub-netting and making sure that the services are available and exposed to external users.

Container runtime: A container runtime like Docker is software that is responsible for running containers. Each worker node includes a container runtime engine to run the containers, which pulls images from an image registry and is responsible for starting and stopping containers.

Container Registry: A container registry stores images which the Kubernetes can rely on. It is a self-configurable registry or can also be a third-party registry.

IV DEPLOYING APPLICATIONS TO A KUBERNETES CLUSTER

Let's discuss the earlier app deployment architecture to understand the importance of Kubernetes.

Traditional Deployment Era – In traditional deployment architecture the application was running on a single physical server. This caused problem for allocating/ defining resources for an applications running on physical server causing resource allocation issues. In traditional deployment of an application where many applications were running on physical server then there can be instance such that one application would utilize most of the resources thus resulting the other application to underperform because other applications may not have sufficient resources left in physical server. The solution for this problem was to have many different physical server and each application can run on different physical server. But this solution was expensive and organization needed lot of cost, resources to maintain these physical servers. This was not efficient also because many time resources were underutilized.

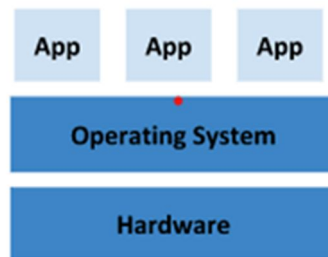


Figure 3 – Traditional Deployment of an APP.

Virtualized Deployment Era- As we have seen in traditional deployment architecture there was problem of resource allocation to all running app on physical server and solution for this problem was to have multiple physical server which was an expensive solution.

This problem was overcooked in virtualization deployment era. In this architecture multiple virtual machines were running on a single physical server. Virtualization increases the security as the application running on one virtual machine cannot freely access the data or information of an application running on other virtual machine.

Virtualization advantages are

- Better resources utilization
- Scalability of application added
- Application can be easily added, updated or removed
- Reduction in hardware cost
- Increase in security.

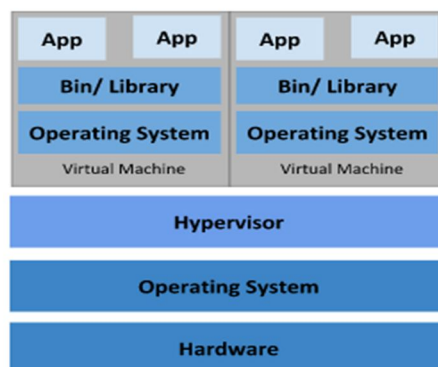


Figure 4 – Virtualized Deployment of an APP.

Container Deployment Era – Container are light weight similar to virtual machine, container have their own file system, container share CPU, processes space memory etc. Containers are decoupled from the underlying infrastructure, they are portable across the clouds and OS distribution. Containers have following benefits

- Application centric management

- Agile application creation and deployment
- High resource utilization
- Resource isolation to predict application performance
- Continuous development, integration and deployment

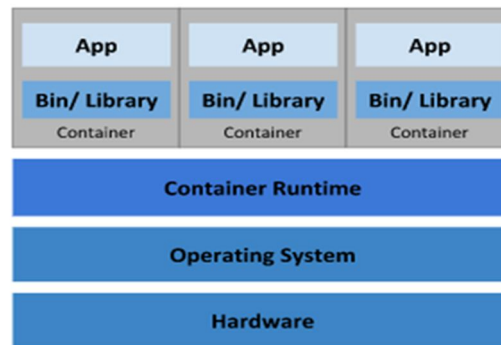


Figure 5 – Container Deployment of an APP.

Kubernetes Uses container deployment era and provides the following benefits

Advantages:

- Service discovery and Load Balancing – Kubernetes is able to balance load if traffic to the container is high making deployment of an app stable.
- Increased Security and Configuration management –
- Helps Developers / DevOps Engineers design applications for Scalability.
- Used to containerize Application Components.
- Containers and Pods scope to be calculated.
- Liveness Probes and implementing Readiness can be implemented.
- Managing Scale and Availability by using deployments.
- Managing Access to layers of an application by creating various ingress rules and services. Using Declarative Syntax instead of Imperative Syntax for managing Kubernetes State

Drawbacks:

Kubernetes usage can sometimes be very overwhelming especially if we consider the local developments and simple stateless applications. This means it could also affect efficiency and have a negative impact on the preset timelines of the organization as a whole. Learning curve for k8s is quite steep. Kubernetes requires experience and rigorous training for its debugging and troubleshooting deployments and managing applications. The benefits of Kubernetes are in abundance, but reaching that level might consume a lot of time, effort, and resources. Teams in organizations have to re-iterate how their time will be used and familiarize themselves with new processes and the complex workflow.

V. PROPOSED MODEL

An Ingress controller, is a dedicated load balancer that abstracts away Kubernetes networking complexity and creates a bridge between services in a Kubernetes cluster and those outside of it, reducing the complexity barrier of Kubernetes.

Ingress controllers for Kubernetes accept traffic coming from outside the Kubernetes platform and load balance it to the platform's active pods (containers). Services that must communicate with other services outside of a cluster can manage egress traffic within a cluster are set up to deploy "Ingress Resources" objects using the Kubernetes API.

When pods are added to or removed from a service, automatically update the load-balancing rules by keeping track of the Kubernetes pods that are operating.

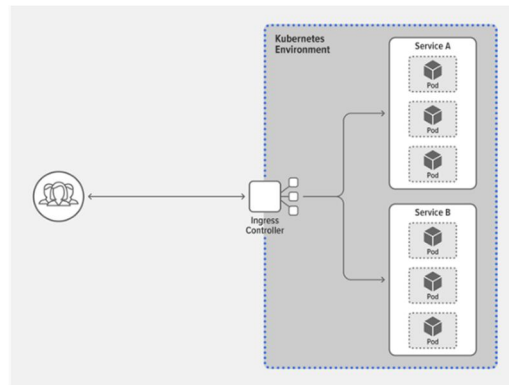


Figure 6: Proposed System architecture

VI. CONCLUSIONS

Kubernetes' perks lie in its generality and modularity. No matter what is the kind of application, it can be easily fit within k8s, and can be deployed using Kubernetes. Applications can also be extended, adjusted or tuned as per the requirements of different organizations. This doesn't mean that Kubernetes is the way to move ahead for all teams or organizations. Everything comes at a cost. The cost here is the complexity Kubernetes comes with, and the effort and time required to understand it fully. The most important part is to understand how various components of k8s work together, and how the kubectl API works. This is essential for software developers so that they can unleash the full potential of the tool. This will also help them to make their application orchestration simple and more reliable. Another critical aspect to master is to understand how the tool links up with a huge variety of external configurations, best practices to use it and how essentially it works in real-world scenarios. Kubernetes is not only about changing the way we deploy and run applications but is more about the modernization of the architecture. Earlier, organizations were worried about the management of applications because of a lack of knowledge and tools. However, that is set to change now. They can now make use of innovative and easy-to-use methods across the world. Since the launch of k8s, it seems like an eternity. This is just the start of a new era with many new ways to comply with technologies like ML, Edge Computing, etc. The only thing remaining is to take the initiative and innovate with the already existing ecosystem which will help award the efforts of every developer who is catering to maintain and scale the technology.

ACKNOWLEDGMENT

We would like to thank Mr Haardik Dharma third year student of Dr. Vishwanath Karad MIT World Peace University for his minor contribution in the work

REFERENCES

- [1] Nguyen, Q. M., Phan, L. A., & Kim, T. (2022). Load-Balancing of Kubernetes-Based Edge Computing Infrastructure Using Resource Adaptive Proxy. *Sensors*, 22(8), 2869. <https://doi.org/10.3390/s22082869>
- [2] Stefano Fiori, Luca Abeni, and Tommaso Cucinotta. 2022. RT-kubernetes: containerized real-time cloud computing. In *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing (SAC '22)*. Association for Computing Machinery, New York, NY, USA, 36–39. <https://doi.org/10.1145/3477314.3507216>
- [3] Bannon, D., Moen, E., Schwartz, M. et al. DeepCell Kiosk: scaling deep learning-enabled cellular image analysis with Kubernetes. *Nat Methods* 18, 43–45 (2021). <https://doi.org/10.1038/s41592-020-01023-0>
- [4] Jorge-Martinez, D., Butt, S.A., Onyema, E.M., Chakraborty, C., Shaheen, Q., De-la-Hoz-Franco, E., & Ariza-Colpas, P.P. (2021). Artificial intelligence-based Kubernetes container for scheduling nodes of energy composition. *International Journal of System Assurance Engineering and Management*, 1-9
- [5] Nguyen, N. D., & Kim, T. (2021). Balanced Leader Distribution Algorithm in Kubernetes Clusters. *Sensors*, 21(3), 869.
- [6] Vayghan, L. A., Saied, M. A., Toeroe, M., & Khendek, F. (2021). A Kubernetes controller for managing the availability of elastic microservice based stateful applications. *Journal of Systems and Software*, 175, 110924.
- [7] Zhou, N., Georgiou, Y., Pospieszny, M., Zhong, L., Zhou, H., Niethammer, C., ... & Hoppe, D. (2021). Container orchestration on HPC systems through Kubernetes. *Journal of Cloud Computing*, 10(1), 1–14.
- [8] Taherizadeh, S., & Grobelsnik, M. (2020). Key influencing factors of the Kubernetes auto-scaler for computing-intensive microservice-native cloud-based applications. *Advances in Engineering Software*, 140, 102734

- [9] Barriga, J. J., Sulca, J., León, J. L., Ulloa, A., Portero, D., Andrade, R., & Yoo, S. G. (2019). Smart parking: A literature review from the technological perspective. *Applied Sciences*, 9(21), 4569.
- [10] Medel, V., Tolosana-Calasan, R., Bañares, J. Á., Arronategui, U., & Rana, O. F. (2018). Characterising resource management performance in Kubernetes. *Computers & Electrical Engineering*, 68, 286–297.
- [11] Uma Pujeri , Utkarsh Rodge , Abhijeet Pawar , Anand Panure , Yaqoob Peerzade2, “CLOUD ENABLED ADVANCED HOME AUTOMATION SYSTEM”, *International Journal of Computer Engineering and Applications*, Volume XII, Special Issue, March 18, www.ijcea.com ISSN 2321-3469.
- [12] Sayalee Narkhede and Trupti Baraskar, Debajyoti Mukhopadhyay (2014), ‘Analyzing Web Application Log Files to Find Hit Count Through the Utilization of Hadoop MapReduce in Cloud Computing Environment’, *CSI BIG 2014 Conference on IT in Business, Industry & Govt*, 08-09, March, 2014 Organized by CSI Indore Chapter, in Association with Sri Aurobindo Institute of Technology, Indore, MP, 978-1-4799-3064-7.