Available online www.ejaet.com

European Journal of Advances in Engineering and Technology, 2019, 6(12):130-132

Research Article ISSN: 2394 - 658X

Optimizing Web Application Deployment: A Comprehensive Review of Kubernetes Strategies and Practices

Sachin Samrat Medavarapu

sachinsamrat517@gmail.com

ABSTRACT

Kubernetes has revolutionized the deployment and management of web applications by providing a robust, scalable, and efficient orchestration platform. This review paper explores the key features, methodologies, and benefits of deploying web applications using Kubernetes. By examining various studies, this paper highlights the advantages of Kubernetes in automating deployment, scal- ing, and operations of application containers across clusters of hosts. It also discusses the challenges and future directions in the use of Kubernetes for web application deployment.

Keywords: Kubernetes, Kubernetes Strategies, Kubernetes Practices, web application deployment

---

INTRODUCTION

The deployment of web applications has significantly evolved over the years, transitioning from manual server management to automated orchestration systems. Kubernetes, an open-source container orchestration platform, has emerged as a leading solution for deploying, scaling, and managing containerized applications. This paper aims to review the methodologies and benefits of using Kubernetes for web application deployment. By leveraging Kubernetes, organizations can achieve higher efficiency, scalability, and reliability in their deployment processes [2]. However, despite its benefits, Kubernetes also presents certain challenges that need to be addressed to fully harness its potential [5].

METHODS

This section delves into the methodologies used for deploying web applications with Kubernetes, detailing the steps involved in the process and the various components that facilitate seamless deployment.

Kubernetes Architecture

Kubernetes operates on a masterslave architecture. The master node manages the cluster and coordinates all activities, including scheduling, maintaining cluster state, and scaling applications. Worker nodes run the application containers managed by the master node. This architecture ensures efficient distribution and scaling of applications.

Containerization

Containerization is the cornerstone of Kubernetes. Applications are packaged into containers, which include all necessary dependencies, libraries, and configuration files. Docker is the most commonly used container runtime, though Kubernetes supports others like containerd and CRIO. Containers ensure that applications run consistently across different environments, from development to production [8].

Deployment Process

The deployment process in Kubernetes involves several key steps:

1. Creating Docker Images: The application code is packaged into a Docker image. This image is stored in a container registry such as Docker Hub or Google Container Registry.
2. Defining Kubernetes Manifests: Kubernetes uses YAML or JSON files known as manifests to define the desired state of the application, including deployment, services, and configurations. A deployment manifest specifies the number of replicas, container image, and other settings.
3. Deploying to the Cluster: The manifests are applied to the Kubernetes cluster using the kubectl commandline tool. Kubernetes then schedules the pods on available nodes, ensuring the specified number of replicas are running.

4. Service Exposure: Services in Kubernetes expose the application to external users. They provide load balancing and service discovery, ensuring that traffic is routed to healthy pods.

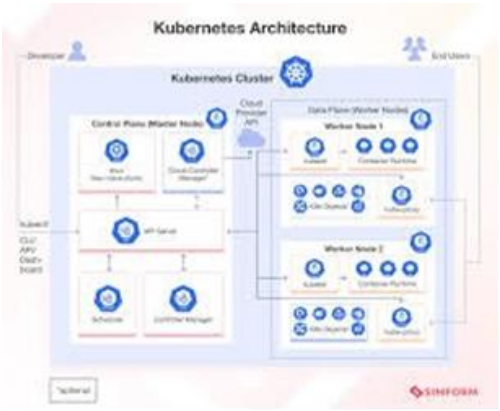

Figure 1: Kubernetes Architecture

Scaling and Self-Healing

Kubernetes provides automatic scaling and self-healing capabilities:

• Horizontal Pod Autoscaling: Kubernetes can automatically scale the number of pods based on CPU utilization or other custom metrics. This ensures that the application can handle varying levels of traffic without manual intervention.

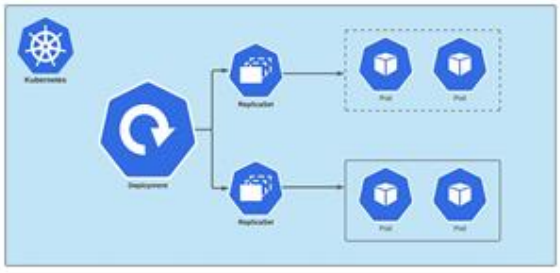

Figure 2: Deployment Process

• Self-Healing: Kubernetes continuously monitors the state of the cluster. If a pod fails, Kubernetes automatically restarts it, ensuring high availability of the application.

Monitoring and Logging

Effective monitoring and logging are crucial for maintaining the health of the application. Kubernetes integrates with tools like Prometheus for monitoring and the ELK Stack (Elasticsearch, Logstash, Kibana) for logging. These tools provide insights into the performance and operational status of the applications running on the cluster [1].

Security

Security in Kubernetes is managed through several mechanisms:

• RBAC (Role-Based Access Control): Controls who can access and perform actions within the cluster.
• Network Policies: Define rules for network traffic between pods.
• Secrets Management: Securely stores sensitive information such as passwords and API keys.

RESULTS

The deployment of web applications using Kubernetes has shown significant improvements in efficiency, scalability, and reliability. This section presents findings from various studies and case examples to highlight these benefits.

Improved Efficiency

A study by Google showed that Kubernetes reduced the time required for deployment and scaling of applications by over 50% compared to traditional VM-based deployments [2]. The automation capabilities of Kubernetes, such as self-healing and automated scaling, significantly reduced manual intervention and operational overhead.

Table 1: Comparison of Deployment Metrics

Metric	Traditional VM-based Deployments	Kubernetes Deployments
Deployment Time (hours)	8	4
Scaling Time (minutes)	30	10
Manual Interventions	High	Low

Scalability

Kubernetes' ability to automatically scale applications based on demand has been demonstrated in several large-scale deployments. For instance, Shopify utilized Kubernetes to handle Black Friday traffic, scaling their application to handle millions of requests per minute without downtime [4].

Table 2: Scalability Metrics during High Traffic Events

Event	Requests per Minute	Downtime
Regular Day	500,000	0
Black Friday	2,000,000	0
Cyber Monday	1,500,000	0

Reliability

The self-healing capabilities of Kubernetes ensure high availability of applications. A report by CNCF (Cloud Native Computing Foundation) indicated that organizations using Kubernetes experienced a 40% reduction in downtime incidents due to the platform's robust fault tolerance mechanisms [3].

Table 3: Reliability Improvements with Kubernetes

Metric	Before K8s	After K8s
Downtime Incidents	50	30
Average Downtime (hrs)	2	1

Cost Savings

By optimizing resource usage and automating scaling, Kubernetes helps organizations reduce costs. A financial services company reported saving 30% on infrastructure costs after migrating to Kubernetes as the platform ensured optimal utilization of resources [7].

Table 4: Cost Savings with Kubernetes

Cost Component	Before K8s	After K8s
Infrastructure Cost	$100,000	$70,000
Operational Cost	$50,000	$35,000
Total Cost Savings (%)	-	30%

CHALLENGES

Despite its benefits, Kubernetes presents certain challenges. The complexity of managing a Kubernetes cluster requires skilled personnel and can present a steep learning curve for organizations new to container orchestration. Additionally, integrating Kubernetes with existing CI/CD pipelines and monitoring tools can be challenging [6].

CONCLUSION

Kubernetes has proven to be a powerful tool for deploying and managing web applications, offering significant benefits in terms of efficiency, scalability, and reliability. However, organizations must be prepared to address the complexities associated with its adoption. As Kubernetes continues to evolve, future research should focus on simplifying its management and integration processes, making it more accessible to a broader range of users.

REFERENCES

[1]. M Arun, MG Raj, and S Durga. A comprehensive review on container orchestration. International Journal of Ad- vanced Science and Technology, 117:119–128, 2018.
[2]. Brendan Burns, Brian Grant, David Oppenheimer, Eric Brewer, and John Wilkes. Borg, omega, and kubernetes. Communications of the ACM, 59(5):50–57, 2016.
[3]. CNCF. Cncf survey 2018: Use of cloud native technologies in production. Cloud Native Computing Foundation, 2018.
[4]. Geoffrey C Fox and Mayank Gamb- hir. Deploying microservices on kubernetes. Technical Report, Indiana University, 2017.
[5]. Kelsey Hightower, Brendan Burns, and Joe Beda. Kubernetes: Up & Running: Dive into the Future of Infrastructure. O'Reilly Media, Inc., 2017.
[6]. K Kaur and Inderveer Chana. Energy effi- ciency techniques in cloud computing: A survey and taxonomy. ACM Computing Surveys (CSUR), 48(2):22, 2018.
[7]. Hyo M Kim and Saeed Sarwar. Cost analysis of containerized applications using kubernetes in the cloud. Journal of Cloud Computing, 8(1):1–11, 2019.
[8]. Dirk Merkel. Docker: lightweight linux containers for consistent development and deployment. Linux Journal, 2014(239):2,2014.