Container Orchestration in Modern Devops

Hareesh Kumar Rapolu

hareeshkumar.rapolu@gmail.com

Abstract

The following research project has examined the effective implications of container orchestration in modern DevOps. It has emphasised the vital role in automating the management of containerised applications. At the same time, understanding the key benefits such as scalability with simplified automated deployment followed by high availability and developing resource utilisation has helped to improve resource utilization. Integration of key technologies such as Docker Swarm with Apache Mesos and Kubernetes has posed to get adhered with the agile workflows and thus is supported to modern DevOps. This has been done by lowering the challenges with strategies such as the integration of CI/CD pipelines and daily monitoring.

Keywords: Container orchestration, DevOps, Automation, Kubernetes, Docker Swarm, Apache Mesos, Microservices.

I. INTRODUCTION

The research project will define the importance of container orchestration in modern DevOps. Container orchestration is defined as a DevOps practice which seeks to automate the overall management of containerised applications. At the same time, it will be used to simplify the main aspects of constructing a container prolifically. The research project will evaluate the key benefits of container orchestration in DevOps and will discuss the key technologies needed for container orchestration, Furthermore, the research project will also be considered to be crucial for identifying the challenges at the initial stages while applying container orchestration and will come up with necessary strategies that will help to mitigate the challenges in a simplified manner. As a result, this will pose to be crucial in terms of streamlining agile or DevOps workflows and thus serving to be flexible with a moderate pace to support modern hybrid multi-cloud architecture.

II. PROVIDING A BRIEF OVERVIEW OF CONTAINER ORCHESTRATION

The following section provides a brief overview of the term container orchestration. It is identified as the automatic provisions along with deployment and scalability which helps to manage the lifecycle of containerised applications. Containers are lightweight and executable application components which are combined with the application source code. This is done with the help of all the operating systems also abbreviated as “OS” libraries and dependencies. These are needed to run the code in any environment. A suitable example states that a consistent integration or continuous delivery (CI/CD) or rather DevOps pipeline is impossible without container orchestration. This is because it tendsto automate the operational tasks concerning deploying and running containerised applications and services[1]. Moreover, as organisations leverage their applications, managing several containers thus becomes complex in regards to application performance monitoring (APM). This complexity is managed by the advanced role of orchestration within service-levelagreements. It has the power to handle critical tasks like service discovery

and health monitoring. Therefore, it is evident that orchestration tools stand to be responsible for enhancing operational efficiency and lowering the chances of downtime errors.

Figure 1: Depicting Container Orchestration

III. DESCRIBING THE BENEFITS OF CONTAINER ORCHESTRATION IN DevOps

This section describes the key benefits of container orchestration in DevOps in an intricate fashion. This tends to increase the deployment speed and frequency. These benefits are mentioned below.

Scalability: In terms of scalability, container orchestration allows for seamless scaling of applications either up or down which is fully based on demand and effective utilisation of cloud resources.

Simplified Automation Deployment: Container orchestration aids in automating the process of deploying followed by scaling and managing the containerised applications. It helps to minimise the chances of manual interventions and potential errors[2].

High Availability: Container orchestration ensures complete application resilience and automatically restarts failed containers on different nodes. This helps to rescue the possibilities of downtime and aligns with optimal performance during peak and off-peak times[3].

Developed Resource Utilization: It is evident that the implicit use of container orchestration aids in optimising effective resource allocation by scheduling containers dynamically on the basis of available infrastructure. This seeks to lower the possibilities of waste.

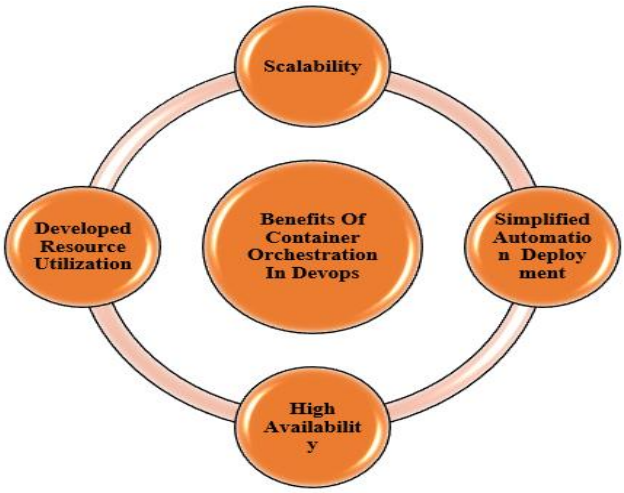

Figure 2: Demonstrating the Benefits of Container Orchestration in DevOps

IV. ILLUSTRATING THE KEY TECHNOLOGIES REQUIRED IN CONTAINER ORCHESTRATION

The following sedition deeply delves into the key technologies which are needed in container orchestration in DevOps. The first key technology required in containment orchestration is the Docker Swarm. It caters for a simpler orchestration solution that is meant for Docker users and nurtures with simple integration with the Docker CLI and a straightforward setup process. It mainly resonates in simplicity with enabling a balanced load and providing service discovery. Another necessary key technology involved with container orchestration is Apache Mesos[4]. This technology is used to manage both containers and non-containerized workloads. The third key technology that is associated with container orchestration is Kubernetes which is recognised for its solidified characteristics. This contains self-heating followed by service discovery and scaling. As a result, this enables effective orchestration of containerised applications and this allows Kubernetes to have robust community support with a charismatic ecosystem of advanced tools and extensions.

V. IDENTIFYING THE CHALLENGES IN IMPLEMENTING CONTAINER ORCHESTRATION

This section highlights the challenges that are observed while implementing container orchestration. These challenges are mentioned below.

Security: In terms of security, sometimes containers pose security-related issues such as vulnerability in images along with inter-container communication and exposure to service. However, protected configuration and aligning in various environments at times can be challenging[5].

Complexity: Container orchestration at times creates problems because of the need to manage diversified elements with networking and service dependencies. As a result, this type of criticalness results in misconfigurations and performance-related challenges that cannot be ignored.

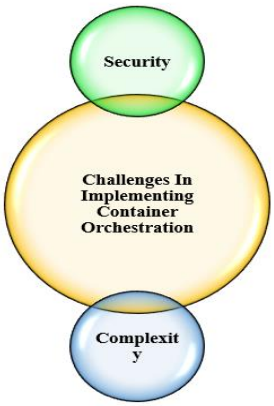

Figure 3: Identifying the Challenges While Implementing Container Orchestration

VI. PROPOSING STRATEGIES TO MITIGATE THE CHALLENGES

The following section elucidates about implementing the best strategies for mitigating the challenges identified while applying container orchestration. These strategies are explained below.

Conducting Monitoring daily: Container orchestration needs to be thoroughly monitored dailyto keep a track record of the performance with resource utilisation and security events. This uses necessary tools such as Prometheus and Grafana[6]. This can nurture valuable insights to prevent any sort of issues.

Integration of CI/CD: The integration of CI/CD pipelines helps to augment the lifecycle. This is used to render positive outcomes and deployment of containers[7]. As a result, this seeks to harness consistency thereby making it easier to manage and scale up the applications cohesively.

Figure 4: Discussing Strategies to Mitigate the Challenges

VII. CONCLUSION

This research project has provided a detailed explanation of container orchestration in modern DevOps. It has signified the importance of container orchestration which has been used to automate deployment and thus improve overall operational efficiency by addressing challenges such as complexity and security. Furthermore, considering the best practices such as conducting monitoring daily and amalgamation of CI/CD pipelines has navigated into sustainable results to limit the challenges and optimise the workflows. As a result, this is strictly adhered to by fulfilling the ongoing demands and needs of hybrid cloud architectures. Therefore, this has determined its usefulness for microservices architectures and modern DevOps.

Abbreviations and Acronyms

- CI/CD- Continuous Integration/ Continuous Deployment
- OS- Operating Systems
- Docker CLI- Docker Command Line Interface
- SLA- Service Level Agreement
- APM- Application Performance Monitoring

Units

- Response time is calculated in seconds
- Resource usage is measured in bytes

Equations

- Throughput = [Number of Transactions / Time Period]
- Latency = [Response Time - Processing Time]

REFERENCES

[1] A. M. Beltre, P. Saha, M. Govindaraju, A. Younge, and R. E. Grant, "Enabling HPC Workloads on Cloud Infrastructure Using Kubernetes Container Orchestration Mechanisms,"*2019 IEEE/ACM International Workshop on Containers and New Orchestration Paradigms for Isolated Environments in HPC (CANOPIE-HPC),* Nov.2019,doi:https://doi.org/10.1109/canopie-hpc49598.2019.00007.

[2] A. Wiedemann, M. Wiesche, H. Gewald, and H. Krcmar, "Understanding how DevOps aligns development and operations: a tripartite model of intra-IT alignment," *EuropeanJournalofInformationSystems*,vol.29,no.5,pp.1–16,Jul.2020,doi:https://doi.org/10.1080/0960085x.2020.1782277.

[3] D. Trihinas, A. Tryfonos, M. Dikaiakos, and G. Pallis, "DEPARTMENT: View from the Cloud DevOps as a Service: Pushing the Boundaries of Microservice Adoption TakingthePulseofDevOpsintheCloud,"Jun.2018.Available:http://linc.ucy.ac.cy/assets/files/publications/pdfs/dtrihinas_IC2018.pdf

[4] D. Weerasiri, M. C. Barukh, B. Benatallah, Q. Z. Sheng, and R. Ranjan, "A Taxonomy and Survey of Cloud Resource Orchestration Techniques," *ACM Computing Surveys*, vol. 50, no. 2, pp. 1–41, May 2017, doi: https://doi.org/10.1145/3054177.

[5] G. Chen, "White Paper Modernizing Applications with Containers in the Public Cloud Sponsored by: Amazon Web Services IDC OPINION," Jun. 2019. Available: http://d1.awsstatic.com/analyst-reports/ModernAppContainersCloud.pdf

[6] J. Kousa, "'Teaching Container-based DevOps Practices in Higher Education Context.,''*ComputerScience47(2020): 0.Helsinki.fi*,Feb.2020.https://helda.helsinki.fi/bitstreams/629abf0e-36a2-40c5-83ca-deb737d9fa2d/download

[7] S. Ravula and M. Tapio Tolvanen, "Achieving Continuous Delivery of ImmutableContainerizedMicroserviceswithMesos/Marathon,"May2017.Available:https://aaltodoc.aalto.fi/bitstream/handle/123456789/27061/master_Ravula_Shashi_2017.pdf?sequence=1