International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 3
***Article Received:** 10 January 2023 **Revised:** 12 February 2023 **Accepted:** 04 March 2023*

Microservices Container Security Orchestration Framework within Kubernetes and Docker for Business-Critical Applications within Digital Transformation

Amarjeet Singh[1], Alok Aggarwal[2]

[1,2] School of Computer Science, University of Petroleum and Energy Studies, Dehradun, India

Abstract - Container virtualization technology facilitates the creation of microservices-based systems through continuous integration. Container-based apps can be deployed more easily when they use orchestration systems like Kubernetes, which has become the de facto standard. It can be difficult to create effective and precise orchestration systems, nevertheless. The scheduler, a crucial orchestrator task that allocates physical resources to containers, is the subject of this article. Scheduling strategies are developed using several Quality-of-Service metrics.

The CI in CI/CD stands for continuous integration. Continuous integration drives the automation in the development and delivery of the code and developers frequently apply code changes. It's an automated process that allows multiple developers to contribute software components to the same project without integration conflicts. CI also triggers the process of testing the applications automatically upon code commit into the repository. Container virtualization technology facilitates the creation of microservices-based systems through continuous integration. Container-based apps can be deployed more easily when they use orchestration systems like Kubernetes, which has become the de facto standard. It can be difficult to create effective and precise orchestration systems, nevertheless. The scheduler, a crucial orchestrator task that allocates physical resources to containers, is the subject of this article. Scheduling strategies are developed using several Quality of Service metrics.

Keywords: Microservice, Microservice Security, Kubernetes, Container, Cloud, Distributed Systems, Micro-services

I. INTRODUCTION

Emphasizing collaboration between development and security teams at an early stage brings numerous advantages over time. DevSecOps facilitates enhanced operational efficiency across different departments, resulting in direct improvements. The implementation of DevSecOps leads to swifter responses from security teams, earlier identification of code vulnerabilities, and increased reliability of the products. Primarily, DevSecOps empowers organizations to deliver more secure products to consumers at a faster pace. Less gridlock during the application of late-stage security practices can make a major difference in freeing up time for DevSecOps engineers to make improvements during other segments of the product development cycle. Considering these significant advantages, it becomes evident why an growing number of companies and organizations are opting to incorporate DevSecOps principles into their development processes.

The main focus of the Kubernetes platform is optimization; it streamlines software developers' tasks by automating numerous laborious DevOps procedures. What then is the key to the platform's popularity? Load balancing is provided by Kubernetes services, which also streamline container administration across several hosts. They facilitate the easy scalability, flexibility, portability, and productivity of an enterprise's software.

Figure: A python script to spin up Kubernetes cluster

The integration of containers and the availability of storage resources from several cloud providers facilitate development, testing, and deployment. Compared to virtual machine (VM) images, container images are easier to produce and contain all the components needed for a program to run. All of this results

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 3
Article Received: *10 January 2023* ***Revised:*** *12 February 2023* ***Accepted:*** *04 March 2023*

in more rapid development and effective release and deployment schedules. Kubernetes should be implemented as early in the development lifecycle as possible to enable developers to test code early and prevent expensive mistakes later. Applications built with microservices consist of separate functional components that communicate with one another through APIs. Out of all version control system existed in development world, now a days 79% version control systems are on Git and it's growing. Many organizations have already left older version control system and adopted Git as their VCS. Although there are some migration tools available in the software industry, their options are somewhat lacking, especially when it comes to pre-migration checks such as empty directories, failover capability and reporting as steps of post-migration.

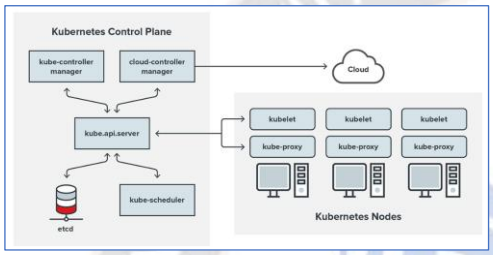

Figure: A holistic containerization approach with Kubernetes.

Docker containers offer the advantage of eliminating the requirement for developers to have an identical production environment setup. Instead, they can utilize their own systems to run Docker containers on VirtualBox. The versatility of Docker becomes evident when the same container can be executed on Amazon EC2 instances. When implementing upgrades during a product release cycle, Docker facilitates seamless modification, testing, and application of changes to existing containers. This flexibility stands out as a key benefit of using Docker. Similar to conventional deployment and integration processes, Docker enables the construction, testing, and release of images deployable across multiple servers. Even in the case of a new security patch, the procedure remains consistent – applying the patch, conducting tests, and releasing it to production.

II. RELATED WORK

According to a poll conducted in 2021, 88% of firms said they were already utilizing Kubernetes for container orchestration. Kubernetes is now the industry standard platform for microservices and container-based task orchestration because of its core capabilities for abstracting cluster resource provisioning. Despite the fact that Kubernetes makes deployment simpler, its distributed ecosystem presents difficulties with regard to cost control and monitoring cluster consumption data. This article delves into the intricacies of a Kubernetes cluster, the difficulties in handling expenses as a result of these inherent complexities, and optimal approaches to enhance cost optimization. In the end, an effective application security program has the potential to position a company more competitively compared to other market players that neglect to prioritize application security in their environments.

III. Containerization Aspects

While migrating from a legacy version control system to distributed version control system, various aspects of the migration work are explored below within the sort of problems and their respective solutions. Four migration aspects - Project structure validation and self-resiliency capabilities, Sensing the SVN & Git SSH network connectivity using NodeMCU IoT platform, SVN users and Git author mapping pattern validation, Remote Git server filing system & space pre-validation - are covered in this work along with possible solutions.

It lets you use a Docker container as a full-featured development environment. You can open any folder within (or attached to) a container, leveraging the complete feature set of Visual Studio Code. The presence of a devcontainer.json file in your project guides VS Code on accessing (or generating) a development container with a clearly defined tool and runtime stack. This container serves the purpose of running an application or isolating tools, libraries, and runtimes necessary for working with a specific codebase.

Workspace files are either mounted from the local file system, copied, or cloned into the container. Extensions are then installed and operated within the container, affording them complete access to the tools, platform, and file system. Consequently, you have the flexibility to effortlessly transition your entire development environment by connecting to a different container.

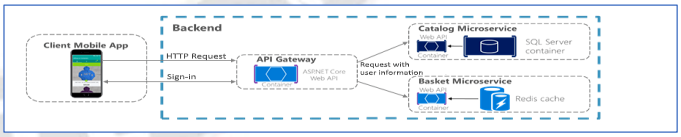

Figure: Centralized Authentication of API Gateway

Mitigating the risk of a CSRF attack involves implementing sophisticated validation techniques, particularly for users interacting with pages on your site, particularly on social media or community platforms. A key strategy is the use of CSRF tokens, also known as anti-CSRF tokens, designed to thwart CSRF attacks. These tokens, often consisting of a unique, lengthy string of numbers, are distinct to both the user and their session. This complexity makes it significantly more challenging for attackers to guess the correct token necessary to generate a valid request.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 3
Article Received: *10 January 2023* ***Revised:*** *12 February 2023* ***Accepted:*** *04 March 2023*

By implementing CSRF tokens in your form submissions and side-effect URLs, you can better ensure that every form submission or request is tied to an authenticated user and shielded from a potential CSRF attack. In cases involving highly sensitive operations, OWASP notes that you may also want to consider implementing a user interaction based protection (either re-authentication/one-time token along) along with token based mitigation techniques.

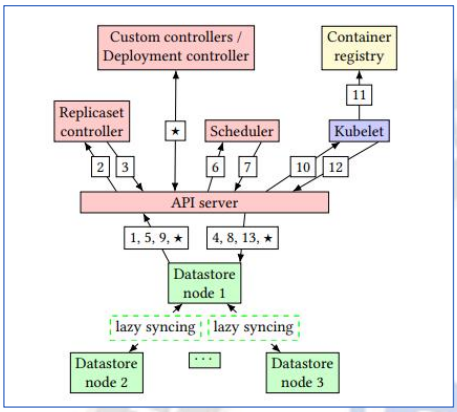

Figure: Scheduling the Kubernetes pods in automated fashion

In the past, you would deploy an application to a virtual machine (VM) and point a DNS server (Domain Name System) to that VM. Today, among the many other advantages of Kubernetes, it allows workloads to exist in one cloud or be easily distributed across many cloud services. Kubernetes clusters make it easy and fast to migrate containerized apps from an on-premises infrastructure to a hybrid deployment across any cloud providers' public cloud (PaaS) or private cloud (PaaE) infrastructure without sacrificing any of an application's functionality or performance. This allows you to move workloads into a closed or private system without locking them in.IBM Cloud, AWS, Google Cloud Platform, and Microsoft Azure provide easy integrations with Kubernetes based apps.

Figure: A holistic approach using Jenkins to spin up pod automatically

Multi-Cloud Platforms

One of Docker's greatest benefits is portability. In recent years, major cloud computing providers, such as Amazon Web Services (AWS) and Google Compute Platform (GCP), have widely embraced Docker's accessibility, providing individualized support. Docker containers are compatible with various hosting environments, including Amazon EC2 instances, Google Compute Engine instances, Rackspace servers, or VirtualBox, as long as the host operating system supports Docker. This flexibility allows seamless migration of containers between different environments, ensuring consistency and functionality. Consequently, Docker provides a level of abstraction from the underlying infrastructure. Beyond AWS and GCP, Docker seamlessly integrates with other Infrastructure as a Service (IaaS) providers like Microsoft Azure and OpenStack. Furthermore, it is compatible with various configuration management tools such as Chef, Puppet, and Ansible, expanding its versatility in different deployment scenarios.

Consistency

Containers offer developers the ability to create consistent and reproducible environments that are isolated from one another and can encompass all necessary dependencies. Since the container image defines the foundational dependencies, there's a high assurance that what runs on a developer's machine will also run consistently across various production environments and on any host operating system. This results in development teams spending less time troubleshooting personal infrastructure issues and more time enhancing applications.

In terms of security and stability, while containers facilitate communication between services through API calls, service meshes, and other discovery methods, they also provide isolation of critical resources. This includes the container's access to underlying CPU, memory, storage, and network resources, preventing individual containers from consuming excessive resources and mitigating potential security risks. Docker, known for its rapid evolution recognized by Gartner, ensures that applications running in containers are entirely segregated and isolated from one another, offering precise control over traffic flow and management. Each container, from an architectural perspective, is allocated its own set of resources spanning processing to network stacks, and no Docker container can inspect processes running inside another container.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 3
Article Received: *10 January 2023* ***Revised:*** *12 February 2023* ***Accepted:*** *04 March 2023*

Figure: A snippet of Pod template

As a means of tightening security, Docker uses host OS sensitive mount points (e.g., '/proc' and '/sys') as read-only mount points and uses a copy-on-write filesystem to make sure containers can't read each other's data. It also limits system calls to your host OS and works well with Selinexor and AppArmor. Additionally, Docker images that are available on Docker Hub are digitally signed to ensure authenticity. Since Docker containers are isolated and resources are limited, even if one of your applications is hacked, it won't affect applications that are running on other Docker containers.

Team Focus

Kubernetes is widely used in industry with 59% of large organizations using Kubernetes in production [15]. The flexibility of Kubernetes allows for Functions as a Service (FaaS), storage orchestration [13] and public-cloud integration [7], among other things. This adoption and flexibility make Kubernetes one of the most attractive platforms for edge deployments on team improvement.
Kubernetes relies on etcd [8], a highly consistent distributed, key-value store, for truth across control-plane components, making it a critical factor in the journey of all requests'. various securities systems proposed for version control migration
Various DevOps systems proposed for version control migration are discussed below.

- Increased Attack Surface
- The public cloud environment has become a large and highly attractive attack surface for hackers who exploit poorly secured cloud ingress ports in order to access and disrupt workloads and data in the cloud. Malware, zero-day exploits, account takeovers, and various other malicious threats have become commonplace in our day-to-day reality.
- Lack of Visibility and Tracking
- In the IaaS model, the cloud providers have full control over the infrastructure layer and do not expose it to their customers. The lack of visibility and control is further extended in the PaaS and SaaS cloud models. Cloud customers often cannot effectively identify and quantify their cloud assets or visualize their cloud environments.

- Ever-Changing Workloads
- Cloud assets are provisioned and decommissioned dynamically—at scale and at velocity. Traditional security tools are simply incapable of enforcing protection policies in such a flexible and dynamic environment with its ever-changing and ephemeral workloads.

- DevOps, DevSecOps and Automation
- Organizations that have embraced the highly automated DevOps CI/CD culture must ensure that appropriate security controls are identified and embedded in code and templates early in the development cycle. Making security-related changes after a workload has been deployed in production can compromise the organization's security posture and extend the time to market.
  Granular Privilege and Key Management: Cloud user roles are frequently configured with loose permissions, providing extensive privileges beyond what is necessary or intended. A prevalent example is assigning database delete or write permissions to untrained users or those without a legitimate business need to modify or add database assets. At the application level, improperly configured keys and privileges can expose sessions to security risks.

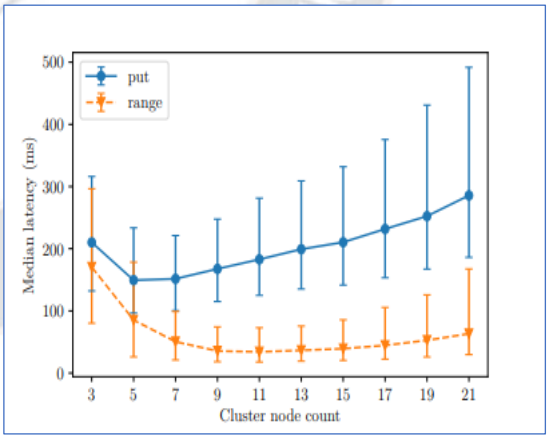

Figure: Latency in k8 by using Splunk monitoring

- Complex Environments
- Managing security in a consistent way in the hybrid and multicloud environments favored by enterprises these days requires methods and tools that work seamlessly across public cloud providers, private cloud providers, and

on-premises deployments—including branch office edge protection for geographically distributed organizations.

- Cloud Compliance and Governance
- All the leading cloud providers have aligned themselves with most of the well-known accreditation programs such as PCI 3.2, NIST 800-53, HIPAA and GDPR. However, customers are responsible for ensuring that their workload and data processes are compliant. Given the poor visibility as well as the dynamics of the cloud environment, the compliance audit process becomes close to mission impossible unless tools are used to achieve continuous compliance checks and issue real-time alerts about misconfigurations.

V. RESULTS AND DISCUSSION

In this paper, we've highlighted the heavy reliance on Kubernetes on etcd and what's causing lower availability and delay in scheduling. We've seen that etcd is a bottleneck to cluster scalability and impacts scheduling latency and the availability of the whole system because of its limited scalability at scale. Based on our results, this supports our core observation that relying on strong consistency in the datastore limits Kubernetes performance, scalability and availability. We propose building a decentralized and eventually consistent store specifically designed for Kubernetes to address these issues.

This redesign also opens up the possibility of rearchitecting Kubernetes edge for increased performance, scalability, and scalability, potentially leading to lower latency, larger scale at the edge deployments. We hope to inform future work on orchestration platforms with the goal of

Kubernetes orchestrates the operation of multiple containers in harmony together. It manages areas like the use of underlying infrastructure resources for containerized applications such as the amount of computer, network, and storage resources required. Orchestration tools like Kubernetes make it easier to automate and scale container-based workloads for live production environments.

References

[1] Hou Q., Ma Y., Chen J., and Xu Y., "An Empirical Study on Inter-Commit Times in SVN," *Int. Conf. on Software Eng. and Knowledge Eng.,"* pp. 132–137, 2014.

[2] O. Arafat, and D. Riehle, "The Commit Size Distribution of Open Source Software," *Proc. the 42nd Hawaii Int'l Conf. Syst. Sci. (HICSS'09),* USA, pp. 1-8, 2009.

[3] C. Kolassa, D. Riehle, and M. Salim, "A Model of the Commit Size Distribution of Open Source," *Proc. the 39th Int'l Conf. Current Trends in Theory and Practice of Comput. Sci. (SOFSEM'13),* Czech Republic, pp. 52–66, 2013.

[4] L. Hattori and M. Lanza, "On the nature of commits," *Proc. the 4th Int'l ERCIM Wksp. Softw. Evol. and Evolvability (EVOL'08),* Italy, pp. 63–71, 2008.

[5] P. Hofmann, and D. Riehle, "Estimating Commit Sizes Efficiently," *Proc. the 5th IFIP WG 2.13 Int'l Conf. Open Source Systems (OSS'09),* Sweden, pp. 105–115, 2009.

[6] Kolassa C., Riehle, D., and Salim M., "A Model of the Commit Size Distribution of Open Source," *Proceedings of the 39th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM'13),* Springer-Verlag, Heidelberg, Baden-Württemberg, p. 5266, Jan. 26-31, 2013.

[7] Arafat O., and Riehle D., "The Commit Size Distribution of Open Source Software," *Proceedings of the 42nd Hawaii International Conference on Systems Science (HICSS'09),"* IEEE Computer Society Press, New York, NY, pp. 1-8, Jan. 5-8, 2009.

[8] R. Purushothaman, and D.E. Perry, "Toward Understanding the Rhetoric of Small Source Code Changes," IEEE Transactions on Software Engineering, vol. 31, no. 6, pp. 511–526, 2005.

[9] A. Singh, V. Singh, A. Aggarwal and S. Aggarwal, "Improving Business deliveries using Continuous Integration and Continuous Delivery using Jenkins and an Advanced Version control system for Microservices-based system," *2022 5th International Conference on Multimedia, Signal Processing and Communication Technologies (IMPACT),* Aligarh, India, 2022, pp. 1-4, doi: 10.1109/IMPACT55510.2022.10029149.

[10] A. Alali, H. Kagdi, and J. Maletic, "What's a Typical Commit? A Characterization of Open Source Software Repositories," *Proc. the 16th IEEE Int'l Conf. Program Comprehension (ICPC'08),* Netherlands, pp. 182-191, 2008.

[11] A. Hindle, D. Germán, and R. Holt, "What do large commits tell us?: a taxonomical study of large commits," Proc. the 5th Int'l Working Conf. Mining Softw. Repos. (MSR'08), Germany, pp. 99-108, 2008.

[12] V. Singh, M. Alshehri, A. Aggarwal, O. Alfarraj, P. Sharma et al., "A holistic, proactive and novel approach for pre, during and post migration validation from subversion to git," *Computers, Materials & Continua*, vol. 66, no.3, pp. 2359–2371, 2021.

[13] Vinay Singh, Alok Aggarwal, Narendra Kumar, A. K. Saini, "A Novel Approach for Pre-Validation, Auto Resiliency & Alert Notification for SVN To Git Migration Using Iot Devices," *PalArch's Journal of Arch. of Egypt/Egyptology*, vol. 17 no. 9, pp. 7131 – 7145, 2020.

[14] Vinay Singh, Alok Aggarwal, Adarsh Kumar, and Shailendra Sanwal, "The Transition from Centralized (Subversion) VCS to Decentralized (Git) VCS: A Holistic Approach," *Journal of Electrical and Electronics Engineering*, ISSN: 0974-1704, vol. 12, no. 1, pp. 7-15, 2019.

[15] Ma Y., Wu Y., and Xu Y., "Dynamics of Open-Source Software Developer's Commit Behavior: An Empirical Investigation of Subversion," *Proceedings of the 29th Annual ACM Symposium on Applied Computing (SAC'14),* pp. 1171-1173, doi: 10.1145/2554850.2555079, 2014.

[16] M. Luczak-R¨osch, G. Coskun, A. Paschke, M. Rothe, and R. Tolksdorf, "Svont-version control of owl ontologies on the concept level." GI Jahrestagung (2), vol. 176, pp. 79–84, 2010.

[17] E. Jim´enez-Ruiz, B. C. Grau, I. Horrocks, and R. B. Llavori, "Contentcvs: A cvs-based collaborative ontology engineering tool." in SWAT4LS. Citeseer, 2009.

[18] I. Zaikin and A. Tuzovsky, "Owl2vcs: Tools for distributed ontology development." in OWLED. Citeseer, 2013.