Design of User Management and security Tools for Centaurus

Shreya Mukesh Dubey (dubeyshreya98@gmail.com)
VIT

Short Report

Keywords: AWS, Kubernetes, multi-tenancy, cloud, user interface, Centaurus clusters

Posted Date: April 8th, 2022

DOI: https://doi.org/10.21203/rs.3.rs-1517240/v1

Design of User Management and security Tools for Centaurus

Shreya Mukesh Dubey
dubeyshreya98@gmail.com
VIT, Vellore, Tamil Nadu, India

Abstract - Centaurus is an open-source software that facilitates the effective development of distributed computing applications in 5G cloud environment. This is necessary for efficient application development since Kubernetes is lacking multi-tenant capability and also in the provision of security. In this scenario, Centaurus can be used to build these gaps. In this project work, a new user and security management tool has been developed for Kubernetes platform using Centaurus. The major tools developed in this project work are User management tool using login feature, tool for clustering and role creation, Multi-Tenant Management tool, Namespace management and virtual machine management tool and a role-based access control tool for enhancing the security of distributed applications. Moreover, for enhancing the security of applications developed using Centaurus, a new Role Based Access Control tool has been developed in this project work. This Role Based Access Control tool provides facilities for creating roles, assigning privileges, securing applications based on user privileges and finally the facility for revoking the privileges in order to enhance the security of the system. The major advantages of the system developed in this project work include the increase in Usability, flexibility and security of the applications.

Keywords: AWS, Kubernetes, multi-tenancy, cloud, user interface, Centaurus clusters.

INTRODUCTION

Kubernetes is an open-source container orchestration platform hosted by Cloud Native Computing Foundation (CNCF). This open-source container orchestration platform is used to automate the management, scaling and deployment of applications on the cloud. Container Orchestration, specified above, is the process of automating the management, scaling and networking of the containers. Containers can be defined as the software packages that can be used by a developer over the cloud to run their applications without making any kind of changes in the environment. Kubernetes can be termed as an excellent technology to work with for the developers as well as the end-users. But, also, on the other hand is far more complex than its alternatives to let the developers and the end-users to work with it. Hence, working to resolve such limitations of Kubernetes, this project came into action. With the increased adoption of cloud technology over the recent years, cloud computing has now become the standard for developing, storing, deploying, and running current and future applications, networks, infrastructures, in context of AI and 5G networks, as well as Open Edge Architecture. The cloud infrastructure platform requires a solution to compute and perform network-related activities throughout data centers in terms of managing these resources. There still are times when one is required to containerize the environment, manage infrastructure, latency, numerous nodes, edge computing, edge core, and edge network, among other things. Centaurus is an open-source project for constructing cloud infrastructure platforms that may be used to build and maintain public or private clouds, edge computing, and edge device datacenters. It is a next-generation upcoming cloud for the telecom sector. It's a solution for large-scale cloud concerns like system scalability, resource efficiency, multi-tenancy, edge computing, and native support for rapidly growing modern workloads like containers and serverless operations. Centaurus aids in the creation of several nodes, infrastructure management, containerization of the environment, pod management, and many other tasks. Centaurus assists telecom companies with edge computing, which brings on the technological resources closer to customers as then the data can be accelerated regularly without any kind of interruption. It addresses issues such as latency, bandwidth constraints, expensive data transmission costs, and data privacy. Centaurus is among the most useful projects for the telecom industry, as it aids infrastructure management, network connectivity, AI computing, and 5G technology. Centaurus is designed to fulfil the infrastructural demand for various new types of cloud workloads such as 5G, AI, Edge, and IoT applications, with advancements in high-performance cloud network solutions, unified runtime environment, and hyper-scale cluster

administration. The Centaurus project aims to accomplish the following goals:

- Unified infrastructure for inherently administering VMs, containers, serverless, bare-metal computers, and other cloud resources.
- Cloud network data plane with high performance for extremely low latency network traffic forwarding and routing.
- In a single region control plane, hyper-scale compute cluster management enables 300K+ hosts and 10M+ network endpoint deployment.
- Edge cloud is a cloud extension that allows you to manage compute and network resources at edge sites from the cloud.

Kubernetes is a container deployment and management platform. While Kubernetes provides some foundational functionality on which users can rely on deploying containerized apps or developing extensions, it does not provide support for the whole cloud platform capabilities. Whereas, Centaurus cloud platform, is a large-scale cloud infrastructure platform which not only supports the configuration and management of containerized workloads but also offers IaaS Provisioning & Management features at a very large-scale. As a result, the Centaurus platform is self-contained and it does not rely on a third-party IaaS layer. Evolved from Kubernetes dashboard – v.2.0.0-beta4, it can be proven to be a more modified, simpler and user-friendly version of Kubernetes dashboard. The dashboard designed will help the common end-users to easily and conveniently manage and interact with the resources within the cluster. This paper specifies the enhancements of the dashboard user interface that allows users to manage Centaurus Clusters, Tenants, Users and Quotas in an intuitive way.

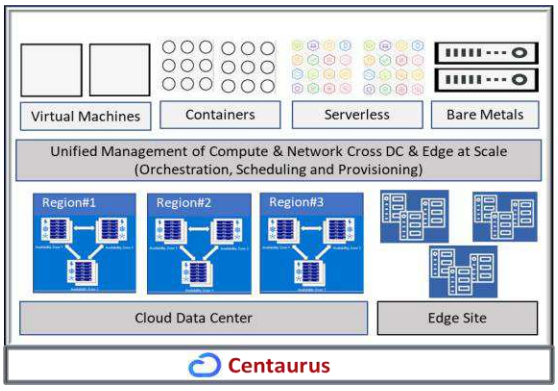
LITERATURE SURVEY

There are some works implemented in the field of Docker and Kubernetes. Here, we have reviewed some of the relative works on the same, presented by the experts in the past. [1] In the cloud-native era, container technologies have been expanding at a rapid pace. Kubernetes has indeed been demonstrated to be able to successfully manage containerized applications in on-premises datacentres as a production-grade container orchestration framework. Dedicated clusters are needed to be in virtualized environment to provide service to many users, i.e. tenants, because Kubernetes lacks suitable multi-tenant capability by architecture. This constraint severely reduces the advantages of cloud computing and makes multi-tenant Software as a Service (SaaS) solutions more challenging to develop using Kubernetes. In this paper, a novel multi-tenant framework also known to be virtual clusters, that adds multi-tenant capabilities to Kubernetes. Virtual cluster, in essence, isolates the control and data planes while distributing the fundamental compute resources between tenants. By avoiding modifying Kubernetes fundamental components, the new conceptual framework ensures API compatibility. As a result, it becomes simple enough to let the existing Kubernetes use cases be integrated. The overheads introduced by virtual cluster in terms of response time and performance are moderate, according to the results. [2] Distributed computing is a technique that adds on a new computing stack based on asset virtualization. With the most current trend of developing applications on cloud, which allows customers to use it in an increasing fashion, the heap on the servers is rapidly growing. The resources are not enough efficiently used due to this notion of extending the stacks on the servers. So, that is why it has been introduced in this period of time. The key goal of this research is to balance the workload on each and every node. This will aid in the distribution of work across multiple nodes by ensuring that no hubs are overwhelmed. Thus, in this paper, the topics like: what docker and container are, and how these simple concepts can help us comprehend docker swarm and Kubernetes technologies are been discussed. Furthermore, the paper demonstrates the ways services are accessed by nodes in a cluster using docker swarm and Kubernetes, as well as the differences between the two. [3] The primary purpose of this study is to examine current Docker container orchestration choices. Presently, three orchestrators are extensively used: Docker Swarm, Kubernetes, and OpenShift. All of the three orchestrators are been studied and discussed in this study, hence highlighting their benefits and drawbacks, and comparing them with each other. [4] Because "cloud" providers deliver resources on demand over the Internet, the Information Technology (IT) sector is migrating from physical storage to cloud storage in today's world of tremendous technical innovation. Cloud computing is a successful and constantly changing technology, with new capabilities and features released on a regular basis. The approach is termed as "pay as you go," and it allows businesses to go to the cloud. As a

result, the confidentiality of such data has become a concern. One of several primary concerns of cloud clients is the security and integrity of cloud-based services. Accessibility, Secrecy, and Consistency are the three cornerstones of cloud security. The use of Container Clustering is one of the most efficient methods. Container clustering can be accomplished in a variety of methods. The analysis, similarities and difference between the two such systems, namely Docker Swarm and Kubernetes, has been discussed about in this research.

ARCHITECTURE

Centaurus is basically, an open source project that has been planned to develop the cutting-edge cloud framework to meet 5G, AI, IOT, etc… technological needs. Centaurus assists with cost and cloud management, disaster recovery, security, and multi-tenancy by merging standards of Infrastructure as a Service (IaaS) and Platform as a Service (PaaS) layers in a single infrastructural platform. It lets you supply and control virtual machines, containers, serverless, and other cloud resources using the same API. At a local scale, it combines the coordination, network provisioning, and administration of cloud compute and network resources. It lets one provision and administer virtual servers, containers, serverless, and other cloud resources using the same Application Programming Interface (API). Centaurus integrates traditional IaaS and PaaS layers into a single infrastructure platform that can enable one to maintain and manage the cloud more efficiently and save money.

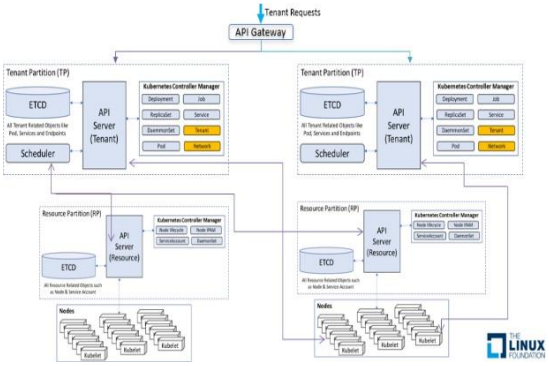


This project is broken down into two sub-projects that are as follows:

- **Arktos** (for Compute) – an extension of Kubernetes

Arktos pushes computing and storage resources much more closer to the end users, to the extent wherein data is being generated instantaneously in a fast virtuous cycle. This data can indeed be regularly linked with the cloud. Arktos addresses a variety of virtualized issues such as latency, bandwidth constraints, high data transit costs, and data privacy concerns. It's an open-source solution for the cloud related difficulties held by the companies that work on and with cloud on a large-scale. It has advantages in terms of speed and latency, as well as security, cost savings, increased reliability, and scalability.

This project uses an algorithm known as: Raft consensus algorithm. It is used by etcd, the key-value store, to assure data store uniformity throughout all the nodes in a cluster, which is essential for a fault-tolerant distributed system. Raft achieves this consistency by electing a master node to oversee replication for the cluster's subordinate nodes, known as followers. Client requests are accepted by the master, that subsequently transmits them to the following nodes. When the master determines that a large percentage of follower nodes have saved all newly received requests as a log entry, it implements the entry to its local state machine and delivers the outcome of that implementation 'write'—to the user. If followers face any kind of breakdown or network packets go missing, the master tries again until all followers have reliably stored all log entries.



The below specified are the architectural components: The **API server** can be defined as, the control plane component of Kubernetes that is responsible to expose out the Kubernetes API. The Kubernetes control plane's front end is the API server. The front-end of the Kubernetes control plane that is seen by every other component within the Kubernetes architecture including users, is the API server. kube-apiserver is the most common as well as the most important Kubernetes API server

implementation. kube-apiserver is built to expand horizontally, which means it can grow by adding more instances. One can start many instances of kube-apiserver and distribute traffic among them.

The **Controller** is in charge of the majority of the collectors, which govern the cluster's state and perform a task accordingly. It can be thought of as a daemon that operates in an infinite loop and is responsible for gathering and sending data to the API server. It aims to obtain the cluster's shared state and then make some adjustments to get the server's present status to the required ideal state. The replication controller, endpoint controller, namespace controller, and service account controller are the some of the most important controllers. To handle the nodes, endpoints, and other things, the controller manager executes several types of controllers.
etcd is the component that keeps track of the configuration files that each node in the cluster can access. It's a key-value store with high availability that can be deployed and distributed across several nodes. Because it may contain some sensitive information as well, Kubernetes API server is the only component in the architecture that is allowed to access the etcd. It's a publicly accessible distributed key-value store.

RESULT AND DISCUSSION

This project has been built using Angular 8 for the frontend and Go language for the backend. The dashboard of this project consist of various components some of which the major ones are:- Tenant Management, Namespace, Cluster Management, User Management and Quota. The results for the project show the improvements of the dashboard user interface that permits clients to oversee Clusters, Tenants, Users and Quotas in an intuitive way.

On logging in to the Cluster-admin account only the Cluster Management and the User Management tabs can be seen by the cluster-admin. Cluster Management consist of cluster monitoring as well as tenants. Cluster monitoring tab consist of the list of Resource Partition that further consist of the list of nodes and its details and the Tenant Partition that consist of the list and details of the tenants present.

Tenant Management consist of tenant monitoring, roles, cluster roles and quota tab under it, each containing the list and details of the respective fields.

Cluster roles and Roles, respectively, describe the steps that a user can execute within a cluster or namespace. With role bindings and cluster role bindings, users may allocate certain roles to Kubernetes subjects (users, groups, or service accounts). Somewhere within a particular namespace only may a Role be used to grant access to resources. A ClusterRole can be responsible to provide the similar privileges as a Role as well, but because it is cluster-scoped, it could also give it access to: cluster-scoped resources and non-resource endpoints. A ResourceQuota object defines quota limits that restricts the aggregate resource consumption per namespace. It can also restrict the number of objects which can be generated in a namespace as per type, as well as the total amount of 3computational resources that resources in that namespace can utilize.

Resource quotas work like this:

- Various teams work in various namespaces. This is now optional, but ACLs will be used to make it mandatory in the coming future.
- For every namespace, the administration produces a Resource Quota.
- The quota system can track the utilization to verify that it does not surpass hard resource limits established in a Resource Quota and the users are responsible to build resources (pods, services, etc.) in the namespace.
- If a quota constraint is broken while adding or modifying a resource, the request will fail with HTTP status code 403 FORBIDDEN and a message detailing the limitation that was violated.
- Users should set demands or limitations for compute resources like cpu and memory if quota is applied on a namespace; or else, the quota system may refuse pods formation.

When multiple teams or projects utilize a shared cluster, namespaces are a useful method to divide clusters into virtualized sub-clusters. Within a cluster, any number of namespaces can be maintained, each logically distinct from one another yet capable of communicating with one another. Together within, namespaces cannot be nested. Any resource that exists in one of two namespaces: the default namespace or a namespace created by the cluster admin. Outside of the namespace, only nodes and persistent

storage volumes exist; these low-level resources are always visible to every namespace in the cluster.

CONCLUSION

Centaurus' outstanding services for next-generation cloud telecom are summarized in this paper. It presents the decentralized cloud's enhanced version. When tried to compare to other cloud resources, it provides a more comprehensive solution with integrated orchestration, multi-tenant, high performance, and cloud-native networking. Centaurus assists with the implementation and administration of software-defined networking technologies across their entire lifecycle. It handles the virtual machines, bare-metal compute nodes, and containers natively with integrated orchestration.

DECLARATIONS

● Ethics approval and consent to participate

Not applicable

● Consent for publication

Not applicable

●Availability of data and materials

Not applicable

● Competing interest

Not applicable

● Funding

Not applicable

● Author's contribution

Being the only author of the paper, I hereby ensure that, the work done within the project and the paper written is completely done by me.

● Acknowledgement

I would like to express my special thanks of gratitude to my guide under the project who completely supported me during this.

REFERENCES

[1] C. Zheng, Q. Zhuang and F. Guo, "A Multi-Tenant Framework for Cloud Container Services," 2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS), 2021, pp. 359-369, doi: 10.1109/ICDCS51616.2021.00042.

[2] N. Marathe, A. Gandhi and J. M. Shah, "Docker Swarm and Kubernetes in Cloud Computing Environment," 2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI), 2019, pp. 179-184, doi: 10.1109/ICOEI.2019.8862654.

[3] A. Tesliuk, S. Bobkov, V. Ilyin, A. Novikov, A. Poyda and V. Velikhov, "Kubernetes Container Orchestration as a Framework for Flexible and Effective Scientific Data Analysis," 2019 Ivannikov Ispras Open Conference (ISPRAS), 2019, pp. 67-71, doi: 10.1109/ISPRAS47671.2019.00016.

[4] Mondal, S.K., Pan, R., Kabir, H.M.D. et al. Kubernetes in IT administration and serverless computing: An empirical study and research challenges. J Supercomput (2021). https://doi.org/10.1007/s11227-021-03982-3

[5] D. Bernstein, "Containers and Cloud: From LXC to Docker to Kubernetes," in IEEE Cloud Computing, vol. 1, no. 3, pp. 81-84, Sept. 2014, doi: 10.1109/MCC.2014.51.

[6] P. Townend et al., "Invited Paper: Improving Data Center Efficiency Through Holistic Scheduling In Kubernetes," 2019 IEEE International Conference on Service-Oriented System Engineering (SOSE), 2019, pp. 156-15610, doi: 10.1109/SOSE.2019.00030.

[7]https://www.strongdm.com/blog/kubernetes-rbac-role-based-access-control

[8]*https://kubernetes.io/docs/concepts/overview/working-with-objects/namespaces/*

[9] *https://kubernetes.io/docs/concepts/policy/resource-quotas/*

[10] Joseph Heck. Kubernetes for Developers : Use Kubernetes to Develop, Test, and Deploy Your Applications with the Help of Containers. Packt Publishing; 2018. Accessed January 26, 2022.

[11] Y. Pan, I. Chen, F. Brasileiro, G. Jayaputera and R. Sinnott, "A Performance Comparison of Cloud-Based Container Orchestration Tools," 2019 IEEE International Conference on Big Knowledge (ICBK), 2019, pp. 191-198, doi: 10.1109/ICBK.2019.00033.