Orchestrating dynamically scalable container-based lab environment for an Educational Institute

Vedant J Patil
Department of Information Technology
A.P. Shah Institute of Technology
Thane, INDIA
19104065@apsit.edu.in

Mudra Limbasia
Department of Information Technology
A.P. Shah Institute of Technology
Thane, INDIA
19104058@apsit.edu.in

Anvit Mirjurkar
Department of Information Technology
A.P. Shah Institute of Technology
Thane, INDIA
19104059@apsit.edu.in

Dr. Kiran Deshpande
Department of Information Technology
A.P. Shah Institute of Technology
Thane, INDIA
kbdeshpande@apsit.edu.in

***Abstract*—Educational institutes in the field of Engineering and technology are liable to have various labs in order to tutor students with new and upcoming technologies. It is a laborious task for any institute to manually set up labs.**

Educational institutes typically contain a variety of laboratories relevant to various technical domains in which students learn about emerging technologies that will prepare them for employment. Educational institutions are required to have software environments in laboratories that are ready for student use. Effective lab environment deployment and administration in light of ever-changing IT trends is a significant problem for educational institutions. Deploying essential software environments in labs is a massive undertaking because manual installation with dependency resolution is time-consuming and requires more technical manpower. Advanced automation and orchestration systems are used in the most valuable solutions.

The solution proposed makes use of Kubernetes along with a cloud platform, which is used to orchestrate a dynamically scalable container-based lab environment for educational institutes.

***Index Terms*—Container orchestration, Cloud, Kubernetes**

I. Introduction

Educational institutes tend to have various laboratories pertaining to diverse technical aspects by which students learn the upcoming technologies which make them industry apt. Educational institutes are obliged to have the software environments in laboratories ready for student operation. Effective deployment and management of the lab environments with respect to ever-growing IT trends is a great challenge for educational institutes. The task of deploying required software environments in labs is a colossal exercise since manual installation with dependency resolution is time-consuming and tedious. The most valuable solutions use sophisticated automation and orchestration systems.

The aim of this paper is to orchestrate a dynamically scalable container-based lab environment that can resolve the problems above mentioned. This solution can be implemented using Kubernetes along with a cloud platform which would assist in deploying the containers so that they can be accessed whenever required during lab sessions.

Our system will help educational institutes incorporate a smooth and dynamically scalable container-based environment to operate and employ various technologies and software in IT/Computer laboratories. This will eliminate the demand of manually installing, and updating software environments in individual systems, hence making it effortless for professors as well as students.

A container image is a software package that is ready to use and includes all of the components required like libraries, dependencies, etc., to run a program, including the code, any necessary runtimes, application, and system libraries, and any necessary default values.

From a simple software process or microservice to a more complex program, anything can be run inside of a single container. The executables, libraries, binary code, and configuration files are all contained inside a container. However, operating system images are absent from containers as compared to server or machine virtualization techniques, and hence, they become substantially more portable and lightweight with less overhead. A container cluster or numerous clusters of containers can be deployed in bigger application installations. A container orchestrator like Kubernetes might be used to manage these clusters. A developer's local laptop, an on-premises data center, or even the cloud can all be used as environments for containers, which simplify the building, testing, deployment, and redeployment of programs. Container advantages include:

- **Less overhead:** As they don't contain operating system images, containers use fewer system resources than conventional or hardware virtual machine environments.
- **Increased portability:** Applications that are running in containers can be quickly deployed to a variety of hardware platforms and operating systems.
- **More consistent operation:** DevOps teams are aware that apps running in containers will function the same wherever they are deployed.

- **Greater efficiency:** Applications may be scaled, patched, or deployed more quickly.
- **Better application development:** Agile and DevOps initiatives to speed up the development, test, and production cycles are supported by containers.

These features will be used to make our system efficient and improve productivity. Containers will be used to include the dependencies of software. These containers will be used by the students to perform lab experiments. However, due to the increasing load on the containers, efficient load balancing must take place, which would include the creation of a new container in order to maintain the proper functioning of the system. Container orchestration is the process of automating the deployment, management, scaling, and networking of containers having software environments required to be used in laboratories.

Any environment where you employ containers can benefit from container orchestration. Deploying the same program across many settings might save you from having to reinvent it. A container is a software that provides a method of executing a piece of software without having to install any physical dependency with the help of using a virtually configured system. Containers can be used to easily implement a software that needs to be configured on a large scale, which would result in decreasing the time taken to install the software while managing the system configuration of the host machine. Kubernetes provides the facility of automating the task of scaling containers, which is also referred to as "Container Orchestration." Container orchestration is the process of automating container deployment, administration, scaling, and networking.

The majority of the operational work necessary to execute containerized workloads and services is automated using container orchestration. Running containers in production can easily turn into a huge help due to their lightweight and ephemeral nature. When designing and running any large-scale system, a containerized application may translate into operating hundreds or thousands of containers.

In large and dynamic systems, orchestration solutions like Docker swarm and Kubernetes are routinely used to deploy and manage multiple connected virtual machines.

To enhance the accessibility of the scalable laboratory environment created, a user-friendly interface is designed and developed for the deployment of required containers for performing respective lab sessions. Dynamically scalable container-based lab environments created will not only serve the purpose of experimenting but can also be extended for collaborative project management

II. Background and related work

The purpose of the literature review is to gain an understanding of the existing technologies and research related to Container Orchestration and debates relevant to the area of study. The literature review helped in selecting appropriate algorithms and suitable feature extraction processes for getting efficient results.

- In Paper [1], To develop and build a modern cloud infrastructure or DevOps implementation than both Docker and Kubernetes have revolutionized the era of software development and operations. Although both are different, they unify the process of development and integration, it is now possible to build any architecture by using these technologies. Docker is used to build, ship and run any application anywhere. Docker allows the use of the same available resources. These containers can be used to make deployments much faster. Containers use less space, are reliable and are very fast. Docker Swarm helps to manage the docker container. Kubernetes is an automated container management, deployment and scaling platform. Using Google Cloud Platform to deploy containers on Kubernetes Engine enabling rapid application development and management. Kubernetes provides key features like deployment, easy ways to scale, and monitoring.
- In paper [2], An enterprise that has implemented virtualization can consolidate multiple servers into fewer host servers and get the benefits of reduced space, power, and administrative requirements. Sharing their hosts' operating system resources, containerization significantly reduces workloads, and is known as a lightweight virtualization. Kubernetes is commonly used to automatically deploy and scale application containers. The scalability of these application containers can be applied to Kubernetes with several supporting parameters. It is expected that the exploitation of scalability will improve performance and server response time to users without reducing server utility capabilities. This research focuses on applying the scalability in Kubernetes and evaluating its performance on overcoming the increasing number of concurrent users accessing academic data. This research employed 3 computers: one computer as the master node and two others as worker nodes. Simulations are performed by an application that generates multiple user behaviours accessing various microservice URLs. Two scenarios were designed to evaluate the CPU load on single and multiple servers. On multiple servers, the server scalability was enabled to serve the user requests. Implementation of scalability to the containers (on multiple servers) reduces the CPU usage pod due to the distribution of loads to containers that are scattered in many workers. Besides CPU load, this research also measured the server's response time in responding user requests. Response time on multiple servers takes longer time than that on single server due to the overhead delay of scaling containers.
- In paper [3], Microservices represent a new architectural style where small and loosely coupled modules can be developed and deployed independently to compose an application. This architectural style brings various benefits such as maintainability and flexibility in scaling and aims at decreasing downtime in case of failure or upgrade. One of the enablers is Kubernetes, an open-source platform that provides mechanisms for deploying, maintaining, and scaling containerized applications across a cluster

of hosts. Moreover, Kubernetes enables healing through failure recovery actions to improve the availability of applications. As our ultimate goal is to devise architectures to enable high availability (HA) with Kubernetes for microservice based applications, in this paper we examine the availability achievable through Kubernetes under its default configuration. We have conducted a set of experiments which show that the service outage can be significantly higher than expected.

- In paper [4], Container-based virtualisation technologies are gaining more and more traction in recent years across Cloud platforms and this will likely continue in the coming years. As such, containers orchestration technologies are becoming indispensable. Kubernetes has become the de facto standard because of its robustness, maturity and rich features. To free users of the burden of having to configure and maintain complex Kubernetes infrastructures, but still make use of its functionalities, all major Cloud providers are now offering cloud-native managed Kubernetes alternatives. The goal of this paper is to investigate the performance of containers running in such hosted services. For this purpose, we conduct a series of experimental evaluations of containers to monitor the behavior of system resources including CPU, memory, disk and network. A baseline consisting of a manually deployed Kubernetes cluster was built for comparison. In particular, we consider the Amazon Elastic Container Service for Kubernetes (EKS), Microsoft Azure Kubernetes Service (AKS) and Google Kubernetes Engine (GKE). The Australia-wide NeCTAR Research Cloud was used for the baseline.
- In paper [5], Container-based virtualization technologies such as Docker and Kubernetes are being adopted by cloud service providers due to their simpler deployment, better performance, and lower memory footprint in relation to hypervisor-based virtualization. Kubernetes supports basic replication for availability, but does not provide strong consistency and may corrupt the application state in case there is a fault. This paper presents a state machine replication scheme for Kubernetes that provides high availability and integrity with strong consistency. Replica coordination is provided as a service, with lightweight coupling to applications. Experimental results show the solution's feasibility.

III. Objectives

This paper intends to achieve the following objectives.

- To model and orchestrate a container-based environment for IT laboratories to provide hassle-free lab environments to students.
- To provide dynamic scalability to container-based lab environment through Kubernetes cluster deployed over the cloud.
- To model and build a user-friendly interface for using a container-based lab environment created which will provide ease to professors and students during lab hours.
- To extend the use of a dynamically scalable container-based lab environment created for collaborative project management and assessment.

IV. Proposed System Architecture

Container-based virtualization technologies are gaining more and more traction in recent years across Cloud platforms and this will likely continue in the coming years. As such, container orchestration technologies are becoming indispensable. Kubernetes has become the de facto standard because of its robustness, maturity, and rich features.

The problem of manually incorporating various lab environments in laboratories of educational institutes can be tackled by the adoption of Cloud Computing and Kubernetes clustering. Cloud Computing can be utilized by uploading the required lab environments on a container, which would then be uploaded on a cloud platform for easy access for the faculty as well as for the students.

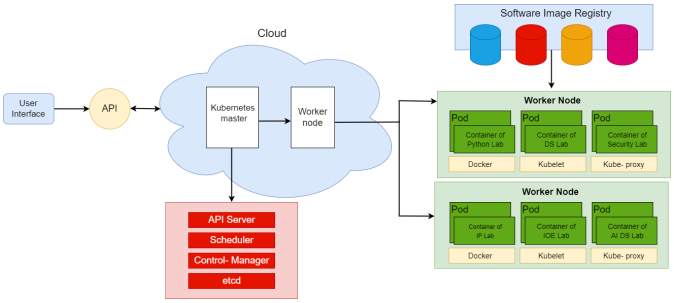

Fig. 1. System Architecture

The user will be provided with a login page where they will have to enter their login credentials, which in turn will be connected through an API to the next interface where they will be shown all the labs that are present in their current academic year.

When the user will select a particular lab, the environment of the respective lab will be pulled and the user will be provided with an environment where they can perform their lab experiments.

Once the experiments are completed, the user can save them on the cloud platform.

A. *Master Node*

Key Kubernetes processes that run and manage the cluster. There can be more than one for high availability. The master node is in charge of overseeing the Kubernetes cluster and providing the API needed to manage and configure its resources. Components of the Kubernetes master node can operate inside Kubernetes as a group of containers inside a specific pod. Typically, the cluster's master node is also set up as a worker node. As a result, the kubelet service, container runtime, and kube proxy service are also operated by the master node, along with other common node services. Note that a node can become corrupted to prevent workloads from running on the wrong node. No additional workloads or containers are

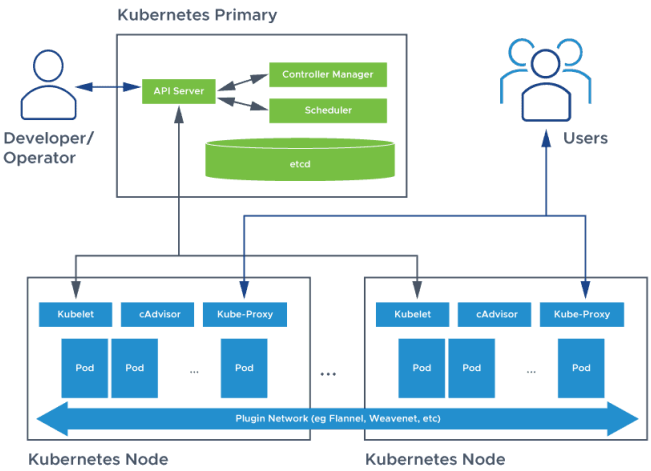

Fig. 2. Kubernetes Architecture

allowed to operate on the master node since the kubeadm function immediately taints it. This makes it easier to back up and restore the master node for the cluster and ensures that it is never subjected to an undue amount of load.

*1) **Key Components of Master Node**:*

- **API Server:** User interface, API, and command line interface are the entry point to the Kubernetes cluster. The API Server is the control plane's front-end and the sole component with which we interface directly. The same API is used for communication between internal system components and external user components.
- **Controller Manager:** It controls and keeps track of the cluster's containers. Logically, each controller should operate in its own process, but for simplicity's sake, they are all compiled into a single binary and executed in a single process.
- **Scheduler:** According to the schedule of the application, it decides which worker nodes will be employed and when. Based on events that happen on etcd, it organizes tasks for the worker nodes. It also stores the node's resource plan in order to decide.
- **Etcd:** The state of the cluster is stored in a key-value store. Kubernetes uses it as the backing store for all cluster data because it is a reliable and highly available key-value store.

B. Worker Node

It controls the containers and pods. The Kubernetes cluster uses worker nodes to execute containerized applications and manage networking so that traffic between applications inside the cluster and from outside the cluster may be appropriately facilitated. Any operations initiated by the Kubernetes API that are performed on the master node are carried out by the worker nodes. The worker nodes continue uninterrupted with the execution of container applications even if the master node is temporarily unavailable.

*1) **Key Components of Worker Node**:*

- **Kubelet:** Each node has a running instance of the Primary Kubernetes "agent." An API server is used for communication with the Master. It manages communication with the master and registers messages. It is the agent that enables the communication between each worker node and the master node's running API Server. Additionally, this agent is in charge of configuring the needs for pods, including mounting volumes, running containers, and reporting status.
- **Pods:** It is the Kubernetes component that runs several containers. Each worker node has several pods. When a pod is started and stopped for load-balancing, its IP address can change, making it difficult to communicate with the pod directly. Each pod has a service that enables contact with it instead.
- **Containers:** It operates within the pods. Along with the OS and other resources required for the application to function, it is the location where the application operates. In a containerized application architecture, a container repository provides storage for container images. Private container repositories allow you to share images with internal teams and approved parties, whilst public repositories allow you to store and share container images with a closed community or the general public.
- **Container Runtime:** Container runtime software, such as Docker, is responsible for operating containers on the node. The fundamental lifecycle management of containers, including stopping and starting them, is not handled by Kubernetes. Any container engine that implements the Container Runtime Interface (CRI) may communicate with the kubelet, which sends commands to the engine based on the requirements of the Kubernetes cluster.
- **Kube- proxy:** kube-proxy supports networking on Kubernetes nodes by implementing network rules that allow communication between pods and entities outside the Kubernetes cluster. Kube-proxy either uses the operating system's packet filtering layer or simply forwards traffic.

V. Conclusion

Container orchestration provides a powerful tool for Educational Institutes to build and manage dynamically scalable lab environments. By automating the deployment, management, and scaling of containerized workloads, container orchestration can help Educational Institutes save time and reduce costs while providing a secure and reliable infrastructure for students and teachers. The flexibility and customization of container orchestration allow Educational Institutes to support a wide range of workloads, from traditional teaching to research and development projects.

Thus, container orchestration is an excellent solution for Educational Institutes that are looking to build and manage scalable and efficient lab environments.

VI. FUTURE SCOPE

The solution proposed can be extended to use this platform as a project collaboration environment, where all the necessary dependencies and files needed for the project will be stored on the platform. These files can be pulled onto any machine thereby creating a suitable environment that will enable the project to be run on any desired machine, without having to install all the dependencies again on a new machine.

The experiments performed successfully by students can be saved and used by later batches as a reference to debug any difficulties faced during execution. Moreover, the system can be updated so that the students will only be able to access the lab environments that are mapped to their respective academic year and semester.

REFERENCES

[1] J. Shah and D. Dubaria, "Building Modern Clouds: Using Docker, Kubernetes and Google Cloud Platform," 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC), 2019, pp. 0184-0189, doi: 10.1109/CCWC.2019.8666479."

[2] Lily Puspa Dewi, Agustinus Noertjahyana, Henry Novianus Palit and Kezia Yedutun, "Server Scalability Using Kubernetes", IEEE Xplore Digital Library 2019. [Online]. Available: https://ieeexplore.ieee.org/document/9024501.

[3] L. Abdollahi Vayghan, M. A. Saied, M. Toeroe and F. Khendek, "Deploying Microservice Based Applications with Kubernetes: Experiments and Lessons Learned," 2018 IEEE 11th International Conference on Cloud Computing (CLOUD), 2018, pp. 970-973, doi: 10.1109/CLOUD.2018.00148.

[4] H. V. Netto, A. F. Luiz, M. Correia, L. de Oliveira Rech and C. P. Oliveira, "Koordinator: A Service Approach for Replicating Docker Containers in Kubernetes," 2018 IEEE Symposium on Computers and Communications (ISCC), 2018, pp. 00058-00063, doi: 10.1109/ISCC.2018.8538452.

[5] A. Pereira Ferreira and R. Sinnott, "A Performance Evaluation of Containers Running on Managed Kubernetes Services," 2019 IEEE International Conference on Cloud Computing Technology and Science (CloudCom), 2019, pp. 199-208, doi: 10.1109/CloudCom.2019.00038.

[6] "Use containers to Build, Share and Run your applications", Docker, [Online]. Available: https://www.docker.com/resources/what-container/.

[7] "Use containers to Build, Share and Run your applications", Docker, [Online]. Available: https://www.docker.com/resources/what-container/.

[8] "What is container orchestration?", RedHat, [Online]. Available: https://www.redhat.com/en/topics/containers/what-is-container-orchestration/.

[9] "Overview of Docker Compose", Docker Documentation, [Online]. Available: https://docs.docker.com/compose/.

[10] "Swarm mode overview", Docker Documentation, [Online]. Available: https://docs.docker.com/engine/swarm/.

[11] "Docker Compose vs Docker Swarm", linuxhint, [Online]. Available: https://linuxhint.com/docker compose vs docker swarm/.

[12] "CNCF Survey: Use of Cloud Native Technologies in Production Has Grown Over 200%", CLOUD NATIVE COMPUTING FOUNDATION 2018. [Online]. Available: ps://www.cncf.io/blog/2018/08/29/cncf-survey-use-of-cloud-native-technologies-inproduction-has-grown-over-200-percent/.

[13] "Kubernetes", En.wikipedia.org, [Online]. Available: https://en.wikipedia.org/wiki/Kubernetes. [14] M. Ashir, "What is Kubernetes cluster?", Educative.io, 2023. [Online]. Available: https://www.educative.io/answers/what-is-kubernetes-cluster-what-are-worker-andmaster-nodes.

[14] "Concepts", Kubernetes.io, [Online]. Available: https://kubernetes.io/docs/concepts/.

[15] "Install and Set Up kubectl", Kubernetes.io, [Online]. Available: https://kubernetes.io/docs/tasks/tools/install-kubectl/.