A Synopsis on

Orchestrating dynamically scalable container based lab environment for an Educational Institute

Submitted in partial fulfillment of the requirements
of the degree of

Bachelor of Engineering

in

Information Technology

by

Vedant Jitendra Patil (19104065)
Anvit Dinesh Mirjurkar (19104059)
Mudra Dinesh Limbasia (19104058)

Dr. Kiran Deshpande

Department of Information Technology
NBA Accredited
A.P. Shah Institute of Technology
G.B.Road,Kasarvadavli, Thane(W), Mumbai-400615
UNIVERSITY OF MUMBAI
2022-2023

CERTIFICATE

This is to certify that the project Synopsis entitled ***"Orchestrating dynamically scalable container based lab environment for an Educational Institute"*** Submitted by ***"Vedant Jitendra Patil, Anvit Dinesh Mirjurkar, Mudra Dinesh Limbasia"*** for the partial fulfillment of the requirement for award of a degree ***Bachelor of Engineering*** in ***Information Technology*** to the University of Mumbai, is a bonafide work carried out during academic year 2022-2023

Dr. Kiran Deshpande
Guide

Dr. Kiran Deshpande
Head of Department Information Technology

Dr. Uttam D.Kolekar
Principal

External Examiner(s)

1.

2.

Place: A.P.Shah Institute of Technology, Thane
Date:

Declaration

I declare that this written submission represents my ideas in my own words. I have adequately cited and referenced the original sources wherever other's ideas or words have been included. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

_____

(Signature)

Vedant Jitendra Patil(19104065)
Anvit Dinesh Mirjurkar(19104059)
Mudra Dinesh Limbasia(19104058)

Date:

Abstract

How to effectively manage the lab environments of an Educational Institute with respect to the ever growing market of various programming languages is a great challenge for educational institutes. The most valuable solutions use sophisticated automation and orchestration systems. Containers are the upcoming technology, they provide rapid and flexible deployment, fine-grained resource sharing, and lightweight performance isolation. 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 in laboratories ready for student operation. The task of preparing the labs is a colossal exercise for large institutes, manual installation with dependencies are time consuming and tedious.

The aim of this project is to provide an orchestrating cloud environment for educational institutes to create containers which incorporate various lab dependencies that contain all the necessary files needed for a perfectly working laboratory, and can be used whenever required.

Index Terms —Container, Containerization, Cloud, Docker, Kubernetes, Deployment.

Introduction

Our system will help educational institutes incorporate a smooth and dynamically salable 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 in individual systems, hence making it effortless for professors as well as the students.

Container orchestration automates the deployment, management, scaling, and networking of containers. Container orchestration might be useful for businesses that must install and oversee hundreds of Linux® servers and containers. Any environment where you employ containers can benefit from container orchestration. Deploying the same programme across many settings might save you from having to reinvent it.

The perfect application deployment unit and self-contained execution environment are provided by containers for your micro service-based projects. They enable autonomous operation of various app components in the form of micro services on the same hardware with significantly improved control over individual components and life cycles. Using orchestration to manage the lifetime of containers benefits DevOps groups who incorporate it into CI/CD processes.

The majority of the operational work necessary to execute containerized workloads and services is automated using container orchestration. This comprises a variety of tasks that operational teams must perform in order to manage the lifespan of a container, such as provisioning, deployment, scaling (both up and down), networking, load balancing, and other activities. Running containers in production can easily turn into a huge effort due to their light weight and ephemeral nature. When designing and running any large-scale system, a containerized application may translate into operating hundreds or thousands of containers, which typically run in their own containers.

If handled manually, this can bring up a tremendous amount of complexity. Container orchestration, which offers a declarative method of automating much of the effort, is what makes that operational complexity bearable for, or DevOps teams. With just a few pieces of software, a few libraries, and the bare essentials of an OS, containers can function and complete the goal for which they were created. In large and dynamic systems, orchestration solutions like Docker and Kubernetes are routinely used to deploy and manage multiple connected virtual machines.

To make the project easy to use by all- professors as well students, a user friendly interface for pulling required dependencies for performing respective lab experiments is created using docker and kubernetes. This lab environment does not only serve the purpose of experimenting, but is also useful in project collaboration and management.

Objectives

In this project implementation we intend to fulfill the following objectives:

1. To model and orchestrate container based environment for IT laboratories to provide hassle free lab environments to students.

2. To provide dynamic scalability to container based lab environment through Kubernetes cluster deployed over cloud.

3. To model and build user friendly interface for using container based lab environment created which will provide ease to professors and students during lab hours.

4. To extend use of dynamically scalable container based lab environment created for collaborative project management and assessment.

Literature Review

The purpose of literature review is to gain an understanding of the existing technologies and research of 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.

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

2. In paper [2], L. P. Dewi, A. Noertjahyana, H. N. Palit and K. Yedutun, "Server Scalability Using Kubernetes," 2019 4th Technology Innovation Management and Engineering Science International Conference (TIMES-iCON), 2019, pp. 1-4, doi: 10.1109/TIMES-iCON47539.2019.9024501.Abstract: 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 behaviors 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.

3. In paper [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. 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.

4. In paper [4], 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. 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.

5. In paper [5] 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. 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 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 feasibility.

Problem Definition

Tech companies face the issue of making each and every system available with the right software to every employee. The manual installation of the softwares is profoundly tiresome.

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

Manual setup of laboratories is a tedious task for any institute. The application of docker and kubernetes provides the functionality of using containers which deals with the drawback of traditional systems which involves manual installation of software on each and every system.

Proposed Solution

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, 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 programming environments in laboratories of educational institutes can be tackled by the adoption of Cloud Computing in Educational Institutes. Cloud Computing can be utilized by uploading the required programming 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.

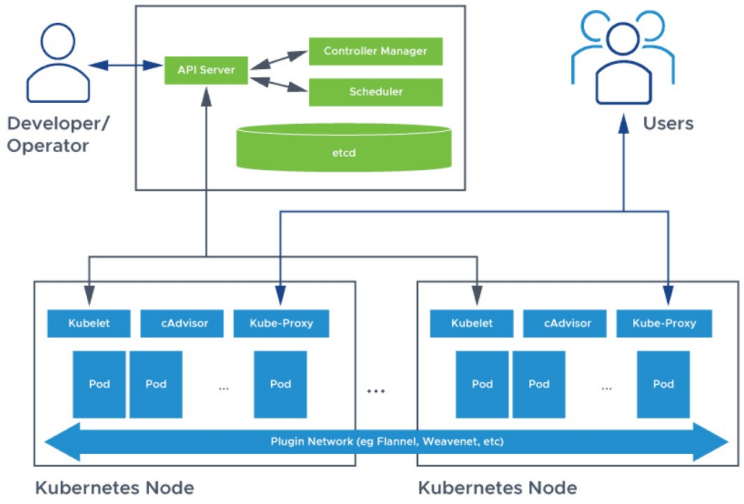

Figure 1: Kubernetes Architecture

Proposed System Architecture

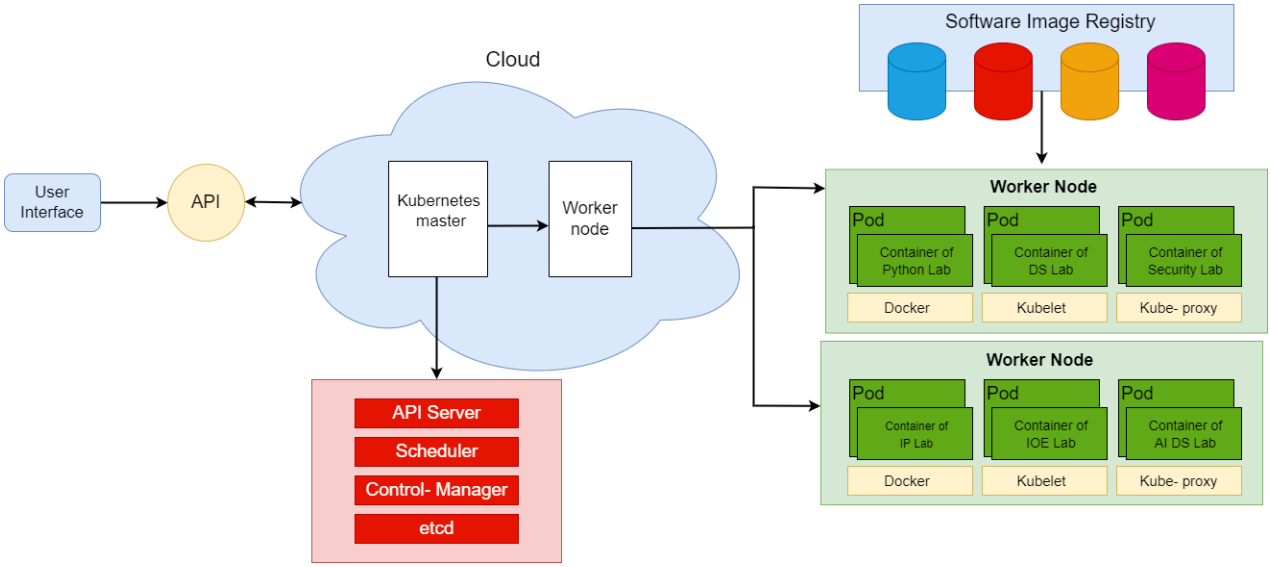

Figure 2: Cloud & Container Architecture

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

Key Features of Master Node:

- **API Server:** User interface, API, and command line interface are the entry points 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 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.

Worker Node

It controls the containers and pods. The Kubernetes cluster uses worker nodes to execute containerized apps 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.

Key Features 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 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:** There are several pods per worker node.

  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.

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] 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.

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

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

[7] "What Are Containers?", Rancher, [Online]. Available: https://www.suse.com/c/rancher_blog/what-are-containers/.

[8] 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.

[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: https://www.cncf.io/blog/2018/08/29/cncf-survey-use-of-cloud-native-technologies-in-production-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-and-master-nodes.

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

[16] "Kubernetes Components", Kubernetes.io, [Online]. Available: https://kubernetes.io/docs/concepts/overview/components/.

[17] "Service", Kubernetes.io, [Online]. Available: https://kubernetes.io/docs/concepts/services-networking/service/.

[18] "Ingress", Kubernetes.io, [Online]. Available: https://kubernetes.io/docs/concepts/services-networking/ingress/.

[19] "ReplicaSet", Kubernetes.io, [Online]. Available: https://kubernetes.io/docs/concepts/workloads/controllers/replicaset/.

[20] "Namespaces", Kubernetes.io, [Online]. Available: https://kubernetes.io/docs/concepts/overview/working-with-objects/namespaces/.

[21] "Network Policies", Kubernetes.io, [Online]. Available: https://kubernetes.io/docs/concepts/services-networking/network-policies/.

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

1 Publication

Paper entitled **"Paper Title"** is presented at **"International Conference/Journal Name"** by **"Author Name".**