A Project Report on

Orchestrating dynamically scalable container based lab environment for an Educational Institute

Submitted in partial fulfillment of the requirements for the award of the degree of

Bachelor of Engineering

in

Information Technology

by

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

Under the Guidance of

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

Academic Year 2022-2023

Approval Sheet

This Project Report entitled ***"Orchestrating dynamically scalable container based lab environment for an Educational Institute"*** Submitted by ***"Vedant Patil"-(19104065), "Mudra Limbasia"-(19104058), "Anvit Mirjurkar"-(19104059),*** is approved for the partial fulfillment of the requirenment for the award of the degree of ***Bachelor of Engineering*** in ***Information Technology*** from ***University of Mumbai***.

(Dr. Kiran Deshpande)
Guide

Dr. Kiran Deshpande
HOD, Information Technology

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

CERTIFICATE

This is to certify that the project entitled ***"Orchestrating dynamically scalable container based lab environment for an Educational Institute"*** submitted by ***"Vedant Patil" (19104065), "Mudra Limbasia" (19104058), "Anvit Mirjurkar" (19104059)*** 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
HOD, Information Technology

Dr. Uttam D.Kolekar
Principal

External Examiner(s)

1.

2.

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

Acknowledgement

We have great pleasure in presenting the report on **Orchestrating dynamically scalable container based lab environment for an Educational Institute**. We take this opportunity to express our sincere thanks towards our guide and HOD, **Dr. Kiran Deshpande** for providing the technical guidelines and suggestions regarding line of work. We would like to express our gratitude towards his constant encouragement, support and guidance throughout the development of project.

We thank **Prof. Sonal Jain** BE project co-ordinator, for being encouraging throughout the course and for guidance.

We also thank all the faculties of the Department for their invaluable help rendered during the course of this work. We wish to express our deep gratitude towards all our colleagues for their encouragement.

Vedant Patil
(19104065)

Mudra Limbasia
(19104058)

Anvit Mirjurkar
(19104059)

Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, We have adequately cited and referenced the original sources. We also declare that We have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We 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.

Vedant Patil - (19104065)

Mudra Limbasia - (19104058)

Anvit Mirjurkar - (19104059)

Date:

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 ensure their technical enablement. 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 can be used in the most valuable solutions.

The solution proposed makes use of Kubernetes cluster deployed over cloud platform, to orchestrate a dynamically scalable container based lab environment for educational institutes.

Contents

List of Figures

List of Abbreviations

IT:	Information Technology
URL:	Uniform Resource Locator
CPU:	Central Processing Unit
EKS:	Elastic Kubernetes Service
AKS:	Azure Kubernetes Service
GKE:	Google Kubernetes Engine
API:	Application Programming Interface
IP:	Internet Protocol
OS:	Operating System
CRI:	Command Runtime Interface
GCP:	Google Cloud Platform
VM:	Virtual Machine

Chapter 1

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 project is to orchestrate a dynamically scalable container-based lab environment that can resolve the problems above mentioned. This solution is implemented using Kubernetes cluster deployed over 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 deploy various technologies and software environments in 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 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.

A container image is a self-contained software package that contains all the necessary components such as dependencies, libraries, and other resources required to run a program. It includes the program's code, runtime, application, and system libraries, as well as any default settings necessary for its execution.

Containers allow running simple software processes or microservices as well as complex programs. They contain all the necessary executables, libraries, binary code, and configuration files in a single package. Unlike server or machine virtualization techniques, containers do not include operating system images, making them more lightweight and portable with lower overhead. Large application installations can be deployed using container clusters, which can be managed by a container orchestrator like Kubernetes. Containers can be deployed in a developer's local laptop, an on-premises data center over the cloud, simplifying the process of building, testing, deploying, and redeploying programs.

Advantages of containers include:

- **Less overhead:** Containers have a lower overhead compared to traditional or hardware virtual machine environments because they do not include operating system images. As a result, they consume fewer system resources.
- **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 applications running in containers will function the same wherever they are deployed.
- **Greater efficiency:** Applications can 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.

Container orchestration is the process of automating the deployment, management, scaling, and networking of containers having software environments required to be used in laboratories.

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.

These features are used to make our system efficient and improve productivity. Containers are used to include the dependencies of software. These containers are 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.

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 not only serves the purpose of experimenting but can also be extended for collaborative project management.

1.1 Objectives

After in depth literature survey, discussions, project meetings, the project 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 sessions.

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

Chapter 2

Literature Review

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 designing appropriate architecture and suitable feature extraction processes for getting efficient results.

- In Paper [1], The implementation of DevOps or the creation of a modern cloud infrastructure can be achieved with the revolutionary technologies of Docker and Kubernetes. Despite their differences, both these technologies bring together the processes of development and integration, making it possible to build any architecture. Docker is used for building, shipping and running applications on any platform, allowing for efficient use of available resources. By using containers, deployments can be made faster due to their smaller size, reliability and speed. Docker Swarm is used for managing the docker containers. On the other hand, Kubernetes is an automated platform for container management, deployment and scaling. By deploying containers on Kubernetes Engine through Google Cloud Platform, rapid application development and management can be achieved. With features like deployment, easy scaling and monitoring, Kubernetes plays a crucial role in modern cloud infrastructure development.

- In paper [2], When an enterprise adopts virtualization, they can consolidate several servers into fewer host servers, leading to reduced space, power, and administrative requirements. Containerization, which is a form of lightweight virtualization, enables workloads to be significantly reduced by sharing host operating system resources. Kubernetes is a popular tool for automatically deploying and scaling application containers. The scalability of application containers can be enhanced through various parameters, which is expected to improve performance and server response time for users without compromising server utility capabilities. This study focuses on applying scalability to Kubernetes and evaluating its performance in handling increasing numbers of concurrent users accessing academic data. Three computers were used, with one serving as the master node and the other two as worker nodes. Simulations were performed using an application that generates multiple user behaviors accessing various microservice URLs. Two scenarios were designed to evaluate CPU load on single and multiple servers. When scalability was enabled on multiple servers, container load was reduced due to the distribution of loads to containers scattered across many workers. In addition to measuring CPU load, the research also evaluated server response time in responding to user requests. Response time on multiple servers took longer than on a single server due to the overhead delay of scaling containers.

- In paper [3], The concept of microservices involves building an application from small, independent modules that can be developed and deployed separately, which brings benefits such as scalability and improved maintenance. Kubernetes is an open-source platform that supports containerized applications across a cluster of hosts, and it also provides mechanisms for healing and recovery to improve application availability. The goal of this paper is to explore the use of Kubernetes for building highly available microservices architectures. The study examines the default configuration of Kubernetes and evaluates its effectiveness in achieving high availability. Experimental results indicate that the service outage may be higher than expected, which highlights the need for further investigation and optimization.

- In paper [4], In recent years, container-based virtualization technologies have gained significant popularity across various cloud platforms, and this trend is expected to continue in the future. As a result, container orchestration technologies such as Kubernetes have become essential tools due to their robustness, maturity, and rich features. To make use of Kubernetes' functionalities without the hassle of configuring and maintaining complex infrastructures, major cloud providers now offer cloud-native managed Kubernetes alternatives. The objective of this research paper is to examine the performance of containers running in these hosted services. The study includes a series of experimental evaluations of containers to monitor the behavior of system resources, such as CPU, memory, disk, and network. For comparison, a baseline consisting of a manually deployed Kubernetes cluster was built. The research focuses on 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], To simplify deployment and reduce memory usage, cloud service providers are increasingly turning to container-based virtualization technologies like Docker and Kubernetes instead of hypervisor-based virtualization. While Kubernetes does offer basic replication for availability, its lack of strong consistency can result in application state corruption in the event of a fault. This paper proposes a state machine replication scheme for Kubernetes that ensures high availability and integrity with strong consistency. The replication scheme is provided as a service with lightweight coupling to applications, and experimental results demonstrate its feasibility.

Chapter 3

Project Design

3.1 Existing System

- Educational institutions catering educational services in engineering and technology domain often have multiple labs dedicated to teaching students about emerging technologies. However, setting up these labs manually can be a very arduous task for any institution. Deploying essential software environments in labs is a massive undertaking because manual installation with dependency resolution is time-consuming and labor-intensive. Educational institutes typically contain a variety of laboratories relevant to various technical areas 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.

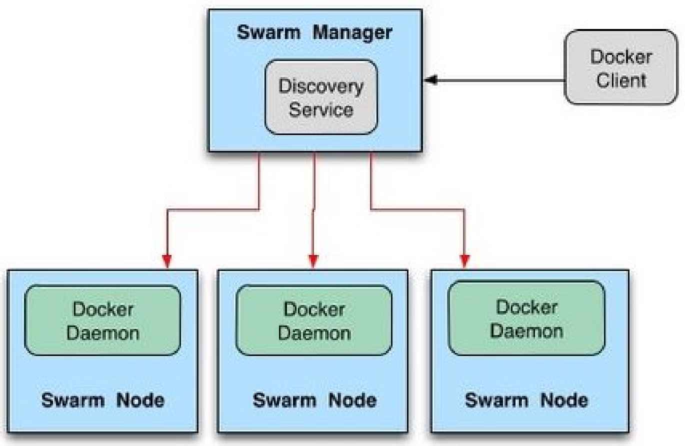


Figure 3.1: Container Orchestration through Docker Swarm

- Through in-depth literature review, we came across the following existing architecture to handle container orchestration. The existing system includes Docker Swarm, which is a container orchestration tool that enables users to manage multiple containers deployed across multiple host machines. While Docker Swarm is a useful tool for managing containerized applications, it has some limitations. Docker Swarm is limited to running on Docker-supported platforms and has fewer extensions and tools available.

- Kubernetes, on the other hand, offers a more advanced and feature-rich platform for container orchestration. It is better suited for large-scale deployments and provides more advanced features for managing scalability, portability, extensibility, multi-tenancy, and high availability. These features make Kubernetes an ideal choice for managing complex containerized environments at scale.

3.2 Proposed System

In recent years, there has been a growing adoption of container-based virtualization technologies across Cloud platforms, and this trend is expected to continue in the future. Consequently, container orchestration technologies are becoming increasingly important. Kubernetes has emerged as the industry standard due to its reliability, maturity, and extensive set of 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.

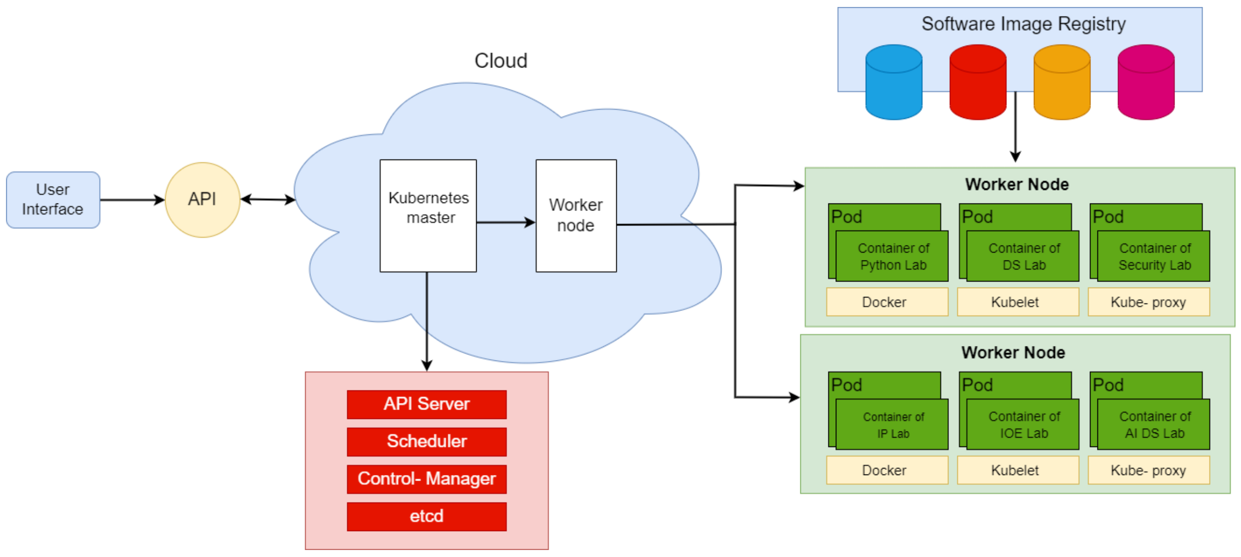


Figure 3.2: System Architecture

Users are provided with an interface where they are supposed to enter their credentials for authentication, which in turn will be connected through an API to the next interface where they are 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 is pulled and users are 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.

3.2.1 Critical Components of System Architecture

To design and develop the proposed system expected, we intend to integrate mainstream technologies like Kubernetes and Google Cloud Platform.

Kubernetes

Kubernetes is a popular open-source platform that helps to manage and orchestrate containers. It provides an easy and efficient way to deploy, manage, and scale containerized appli-

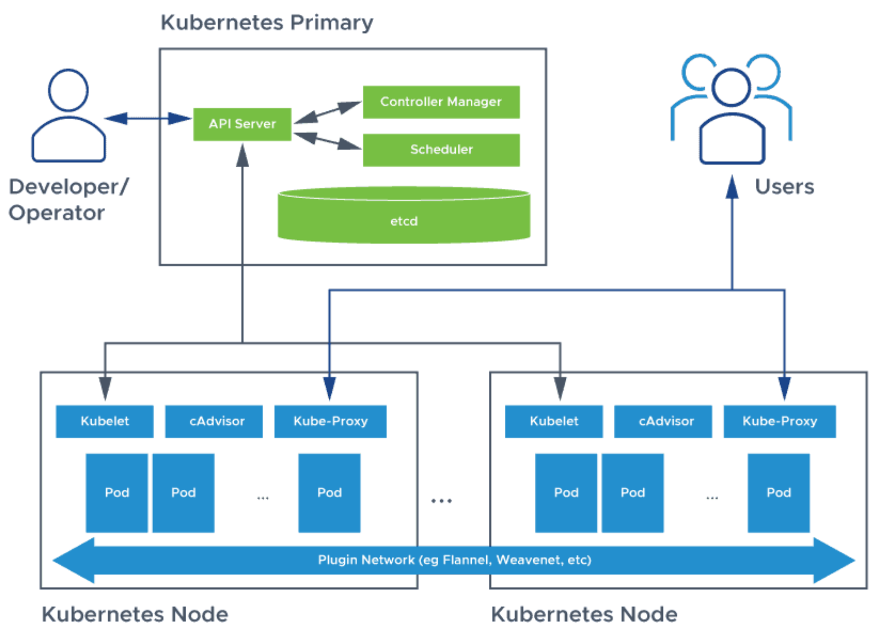


Figure 3.3: Kubernetes Architecture

cations. With Kubernetes, you can automate the deployment, scaling, and management of your applications, and it also helps you to abstract away the underlying infrastructure. In simple words, Kubernetes helps to make the deployment and management of containerized applications easier and more efficient.

I. 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 back up and restore the master node for the cluster and ensures that it is never subjected to an undue amount of load.

Key Components of Master Node

- **API Server:** User interface, API, and command line interface are the entry point to the Kubernetes cluster. The front-end of the control plane is represented by the API Server, which is the only component we interact with directly. Both internal system components and external user components utilize the same API for communication.

- **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, all of them are merged into a solitary binary and executed within a unified 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 condition of the cluster is maintained in a key-value storage system, which Kubernetes employs as the underlying storage for all cluster data due to its dependable and extensively accessible nature.

II. 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.

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 enables networking on Kubernetes nodes by implementing network regulations that facilitate communication between pods and entities located outside of the Kubernetes cluster. Kube-proxy either uses the operating system's packet filtering layer or simply forwards traffic.

III. Google Cloud Platform

Compute
Data Transfer
Storage & Database
Internet of Things
Networking
Google Cloud
Management Tools
Identify & Security
Big Data
Cloud AI

Figure 3.4: Google Cloud Platform

Google Cloud Platform (GCP) offers a complete suite of tools and services for deploying and managing Kubernetes clusters. This platform provides various functionalities that enable you to create, manage, and scale Kubernetes clusters quickly and easily.

GCP allows creation of Kubernetes cluster using the console or the gcloud command-line tool. One can configure the cluster by setting various parameters like the number and size of nodes, networking options, and security settings. Once the cluster is set up, you one can deploy applications to the cluster using Kubernetes manifests or Helm charts.

In addition to deployment, GCP also offers various monitoring and management tools such as Stackdriver logging and monitoring. One can use these tools to monitor the health of the cluster, troubleshoot issues, and analyze logs and can scale the cluster up or down based on resource utilization and application demands.

Moreover, GCP provides integrated billing and cost management features that help optimize costs and control spending. Users can easily estimate costs and create a budget, and GCP also provides alerts and reports to help you manage expenses.

Overall, GCP provides a robust and flexible platform for running Kubernetes, making it easy for creating and managing Kubernetes clusters and deploy applications in a scalable and cost-efficient manner.

IV. Google Kubernetes Engine

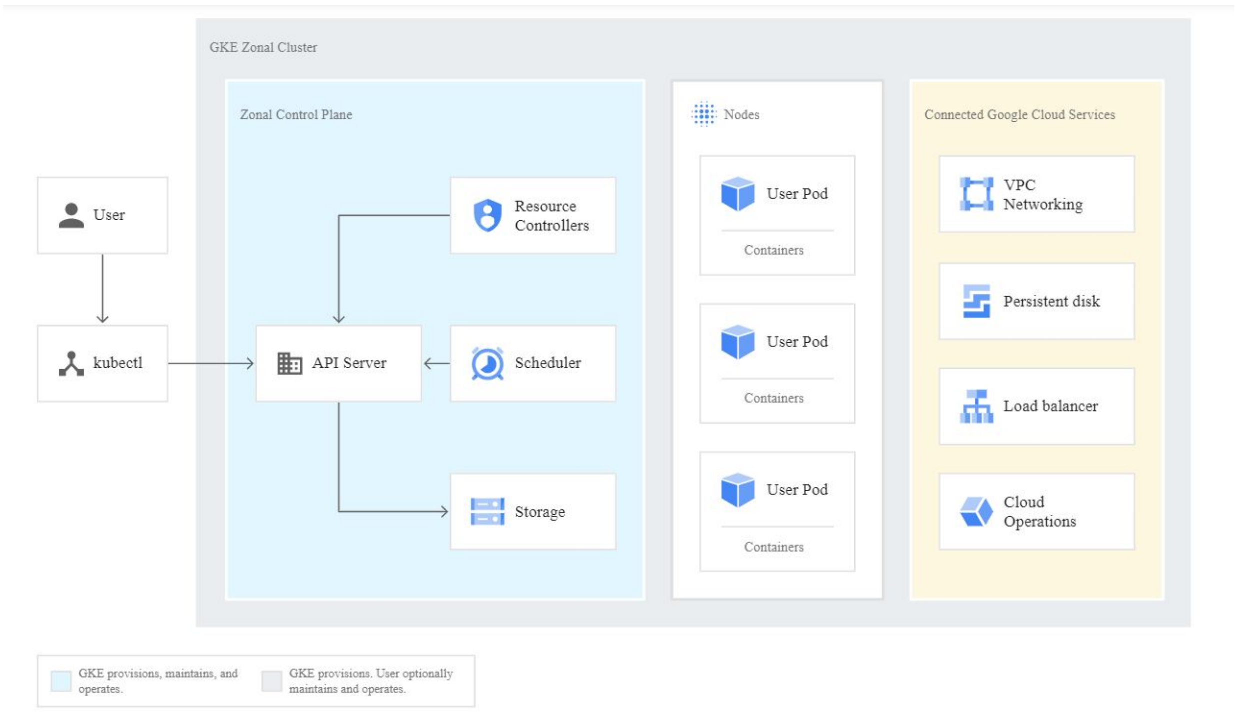


Figure 3.5: GKE - Architecture

The foundation of the Google Kubernetes Engine system is formed by clusters. Kubernetes objects are dependent on clusters, including containerized applications, which use the cluster as a platform. A cluster comprises of one or more Control Plane or cluster master machine(s) and multiple worker machines called nodes. These nodes are typically created by Google as Google Compute Engine VM instances once a cluster is created. The machine type can be specified at the time of cluster creation. Depending on the user's preferred level of control over the functioning, GKE clusters can be either in "standard mode" or "autopilot mode".

This project is implemented using the GKE cluster in "Standard Mode", where the cluster admin creates the cluster and sets the number of nodes and pods which are to be used by the end-user.

3.3 System Diagram

3.3.1 UML Diagram

UML (Unified Modeling Language) is a visual modeling language used to represent software systems. It consists of a set of diagrams and symbols that can be used to describe different aspects of a system, including its structure, behavior, and interactions. UML diagrams are commonly used in software engineering to facilitate communication and understanding among team members and stakeholders. There are several types of UML diagrams, including class diagrams, use case diagrams, sequence diagrams, activity diagrams, and more.

To give precise understanding about the system design and development of the project, following UML diagrams are drawn.

3.3.2 Activity Diagram

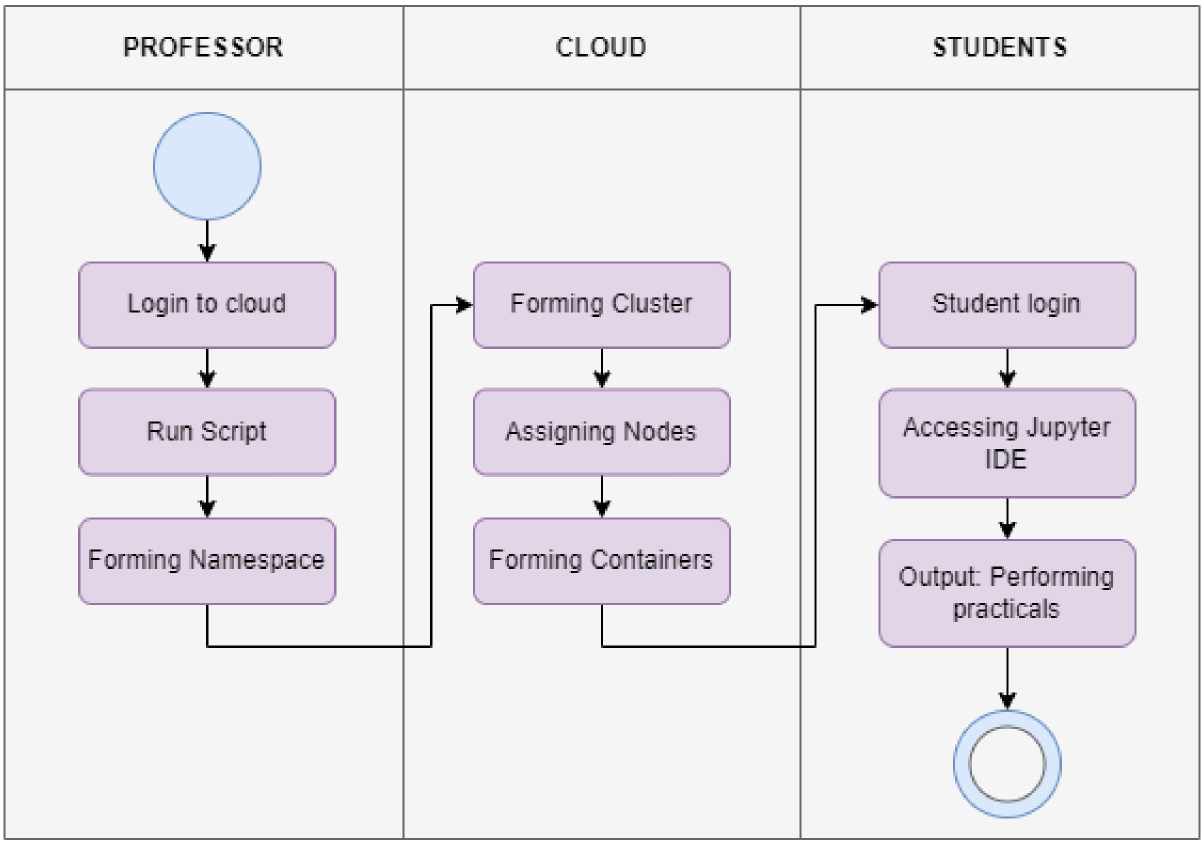


Figure 3.6: Activity Diagram for creating and using Lab Environment over GCP Cloud

The activity diagram shown is for the working of the proposed system. Firstly the professor logs into the cloud and runs the script which creates a namespace.

This namespace is used to store the clusters made by the GKE(Google Kubernetes Engine). The cluster assigns nodes, which have containers.

These containers are accessed by students. They contain the pod for the desired software, JupyterHub in our case. The student logs in to the JupyterHub and then will be accessed to perform practicals during lab sessions

3.3.3 Use Case Diagram

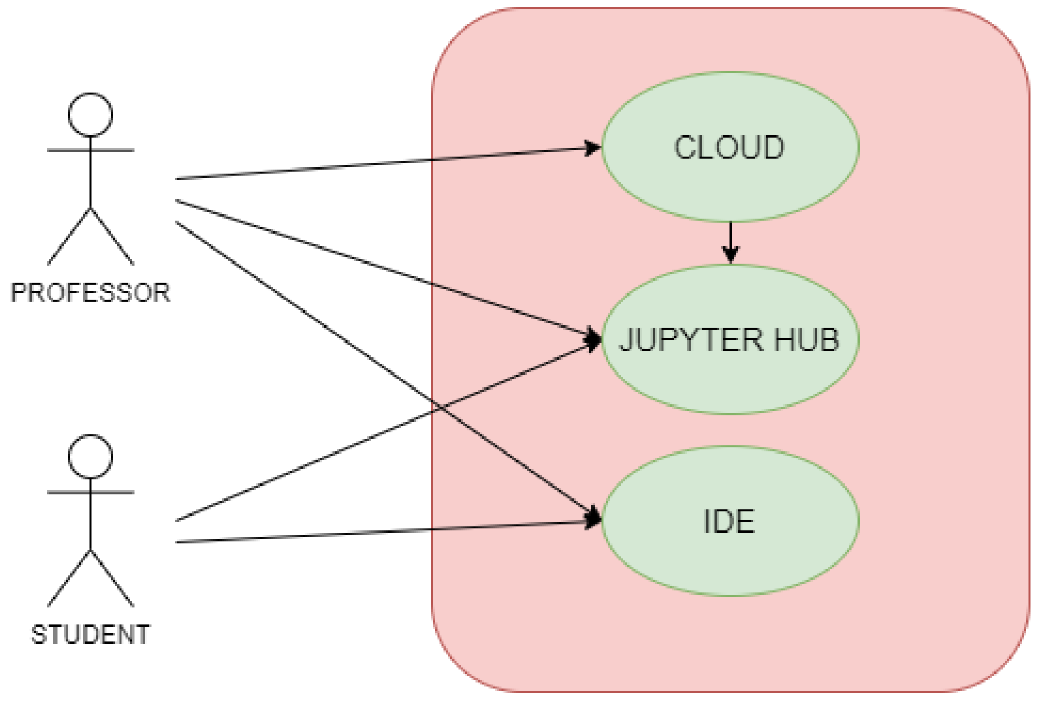


Figure 3.7: Use Case Diagram representing access of Lab Environment deployed over cloud

Use case diagrams are important tools for identifying and documenting the requirements of a system, including both internal and external factors. These requirements are primarily focused on the design aspects of the system and can be accompanied by other types of diagrams. The use case Diagram illustrates the different roles of users such as Professors and Students, their relationship with the system, and the actions they can perform with it. The system will serve as a platform for conducting laboratory experiments with various programming languages.

3.3.4 Sequence Diagram

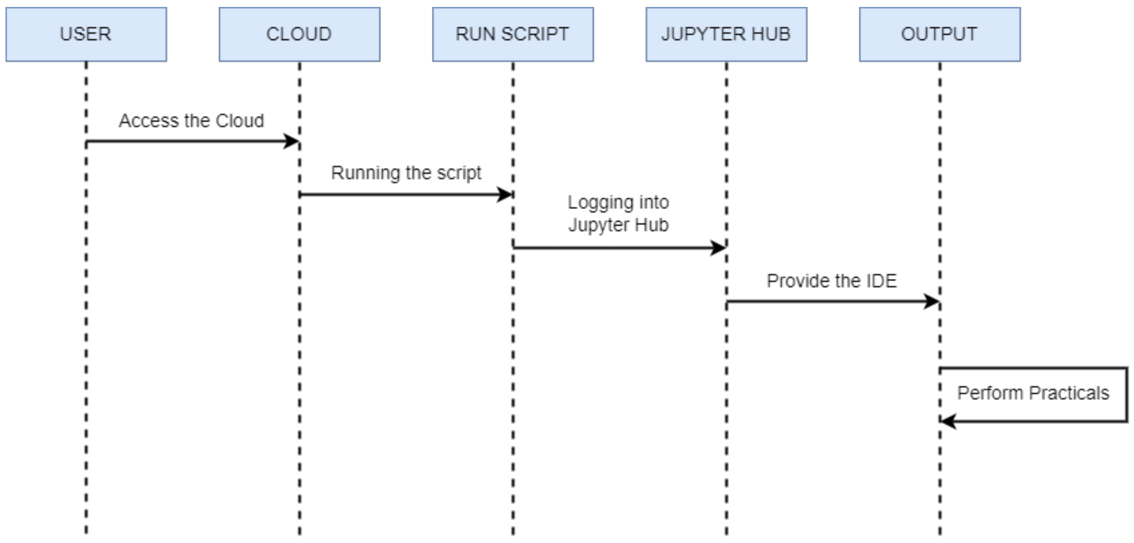


Figure 3.8: Sequence Diagram representing usage from user perspective

The sequence diagram depicts the interactions and sequence of events between objects in our system. Additionally, an activity diagram outlines the steps involved in the operation of our proposed system. The process begins with a professor logging into the cloud and initiating a script that generates a namespace. This namespace is utilized to store the clusters created by the Google Kubernetes Engine (GKE), which assigns nodes to contain the relevant containers. These containers, in turn, are accessible to students and contain the desired software pod, namely JupyterHub. Students then log in to JupyterHub and can subsequently carry out practical lab sessions.

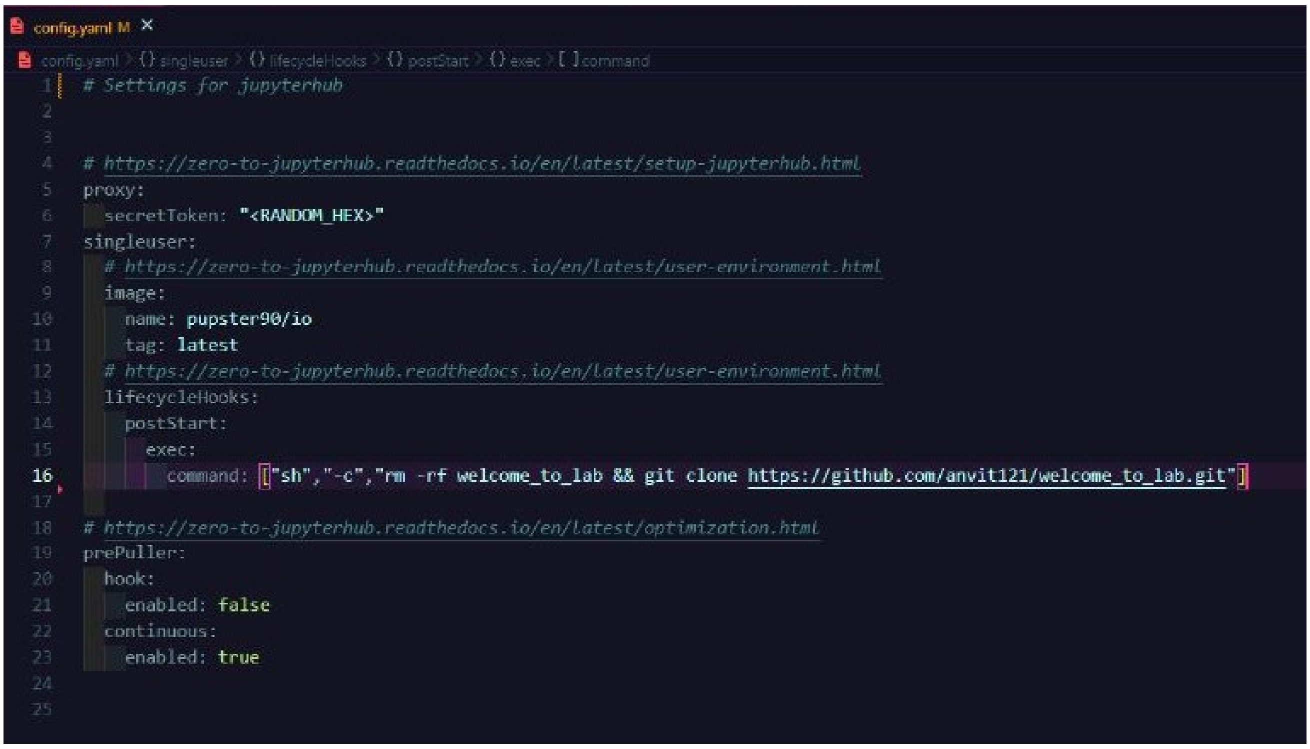
Chapter 4

Project Implementation

In this section, we will provide implementation snippets of the project to give you a better understanding of how the code works. These snippets will highlight some of the methods utilized to obtain the required outcomes while showcasing the project's main functionality. These snippets represent the flow of actions that the users will need to follow to access the system.

4.1 Code Snippets

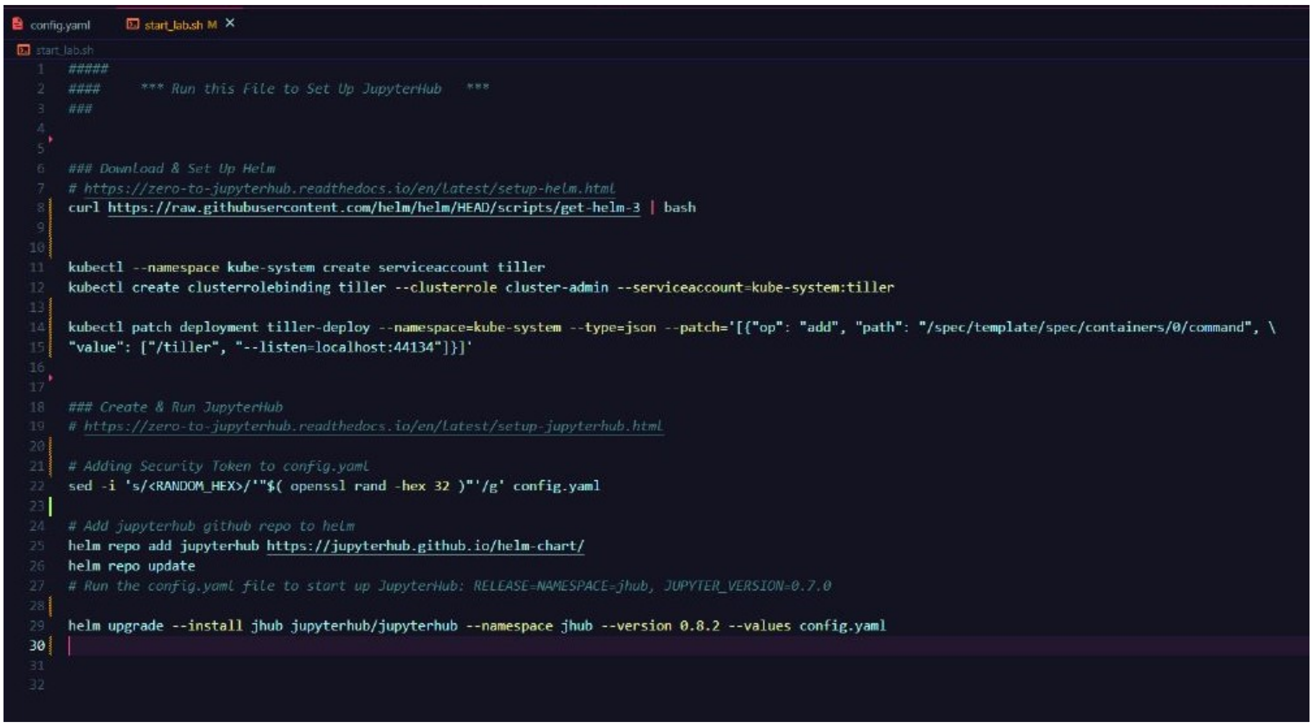
The configuration file of the system.


config.yaml M ×
config.yaml > {} singleuser > {} lifecycleHooks > {} postStart > {} exec > [ ] command

```yaml
# Settings for jupyterhub


# https://zero-to-jupyterhub.readthedocs.io/en/latest/setup-jupyterhub.html
proxy:
  secretToken: "<RANDOM_HEX>"
singleuser:
  # https://zero-to-jupyterhub.readthedocs.io/en/latest/user-environment.html
  image:
    name: pupster90/io
    tag: latest
  # https://zero-to-jupyterhub.readthedocs.io/en/latest/user-environment.html
  lifecycleHooks:
    postStart:
      exec:
        command: ["sh","-c","rm -rf welcome_to_lab && git clone https://github.com/anvit121/welcome_to_lab.git"]

# https://zero-to-jupyterhub.readthedocs.io/en/latest/optimization.html
prePuller:
  hook:
    enabled: false
  continuous:
    enabled: true
```

Figure 4.1: Configuration file showcasing automation and deployment

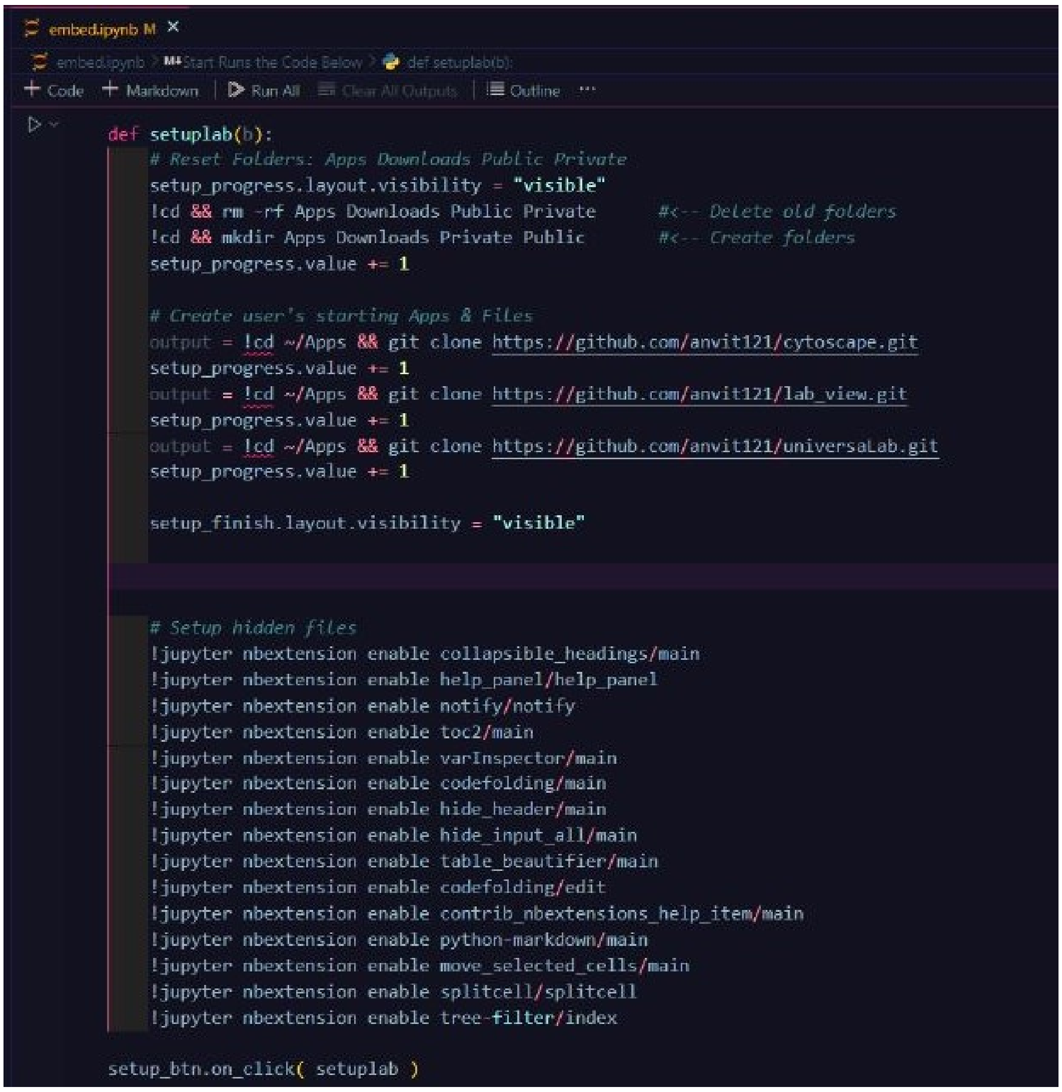
This code initializes the project and provides the user with the server's IP address to access the JupyterHub platform.



```bash
#####
####     *** Run this File to Set Up JupyterHub   ***
###


### Download & Set Up Helm
# https://zero-to-jupyterhub.readthedocs.io/en/latest/setup-helm.html
curl https://raw.githubusercontent.com/helm/helm/HEAD/scripts/get-helm-3 | bash


kubectl --namespace kube-system create serviceaccount tiller
kubectl create clusterrolebinding tiller --clusterrole cluster-admin --serviceaccount=kube-system:tiller

kubectl patch deployment tiller-deploy --namespace=kube-system --type=json --patch='[{"op": "add", "path": "/spec/template/spec/containers/0/command", \
"value": ["/tiller", "--listen=localhost:44134"]}]'


### Create & Run JupyterHub
# https://zero-to-jupyterhub.readthedocs.io/en/latest/setup-jupyterhub.html

# Adding Security Token to config.yaml
sed -i 's/<RANDOM_HEX>/'"$( openssl rand -hex 32 )"'/g' config.yaml

# Add jupyterhub github repo to helm
helm repo add jupyterhub https://jupyterhub.github.io/helm-chart/
helm repo update
# Run the config.yaml file to start up JupyterHub: RELEASE=NAMESPACE=jhub, JUPYTER_VERSION=0.7.0

helm upgrade --install jhub jupyterhub/jupyterhub --namespace jhub --version 0.8.2 --values config.yaml
```

Figure 4.2: Initializing the system designed

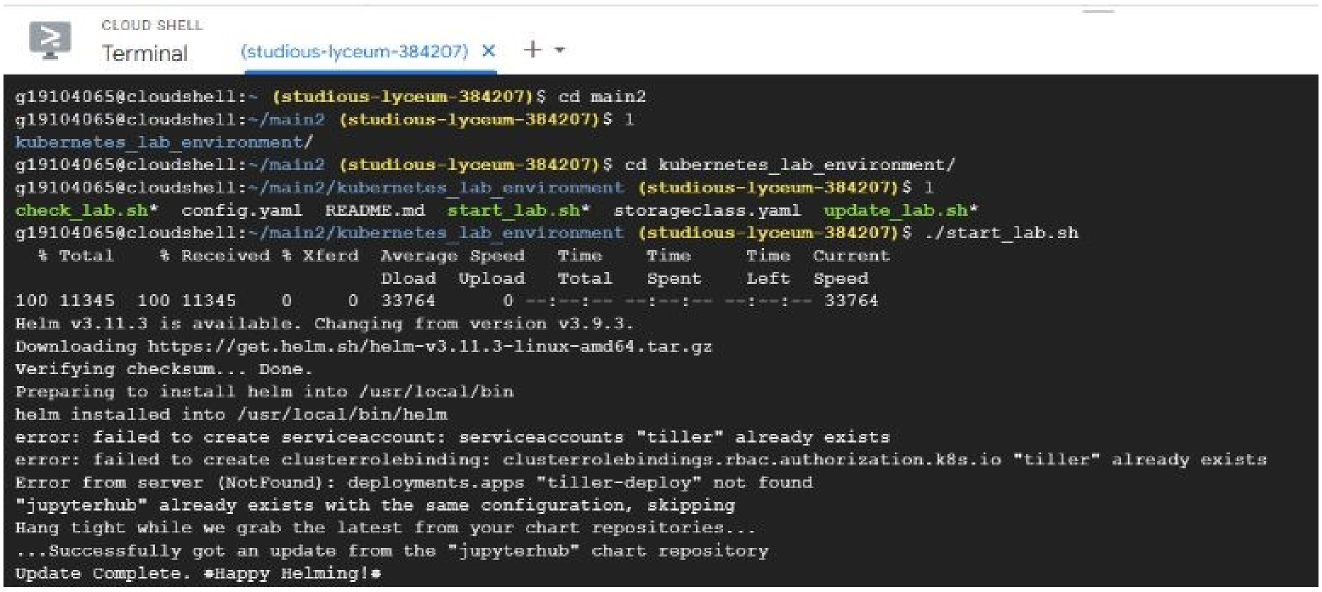
This code displays the folder structure in JupyterHub, which is a web-based interactive computing environment for Jupyter notebooks.


embed.ipynb M
embed.ipynb > M↓Start Runs the Code Below > def setuplab(b):
+ Code + Markdown | Run All | Clear All Outputs | Outline ···

```
def setuplab(b):
    # Reset Folders: Apps Downloads Public Private
    setup_progress.layout.visibility = "visible"
    !cd && rm -rf Apps Downloads Public Private      #<-- Delete old folders
    !cd && mkdir Apps Downloads Private Public       #<-- Create folders
    setup_progress.value += 1

    # Create user's starting Apps & Files
    output = !cd ~/Apps && git clone https://github.com/anvit121/cytoscape.git
    setup_progress.value += 1
    output = !cd ~/Apps && git clone https://github.com/anvit121/lab_view.git
    setup_progress.value += 1
    output = !cd ~/Apps && git clone https://github.com/anvit121/universaLab.git
    setup_progress.value += 1

    setup_finish.layout.visibility = "visible"



    # Setup hidden files
    !jupyter nbextension enable collapsible_headings/main
    !jupyter nbextension enable help_panel/help_panel
    !jupyter nbextension enable notify/notify
    !jupyter nbextension enable toc2/main
    !jupyter nbextension enable varInspector/main
    !jupyter nbextension enable codefolding/main
    !jupyter nbextension enable hide_header/main
    !jupyter nbextension enable hide_input_all/main
    !jupyter nbextension enable table_beautifier/main
    !jupyter nbextension enable codefolding/edit
    !jupyter nbextension enable contrib_nbextensions_help_item/main
    !jupyter nbextension enable python-markdown/main
    !jupyter nbextension enable move_selected_cells/main
    !jupyter nbextension enable splitcell/splitcell
    !jupyter nbextension enable tree-filter/index

setup_btn.on_click( setuplab )
```

Figure 4.3: Configuration file showcasing automation of JupyterHub Web-Interface

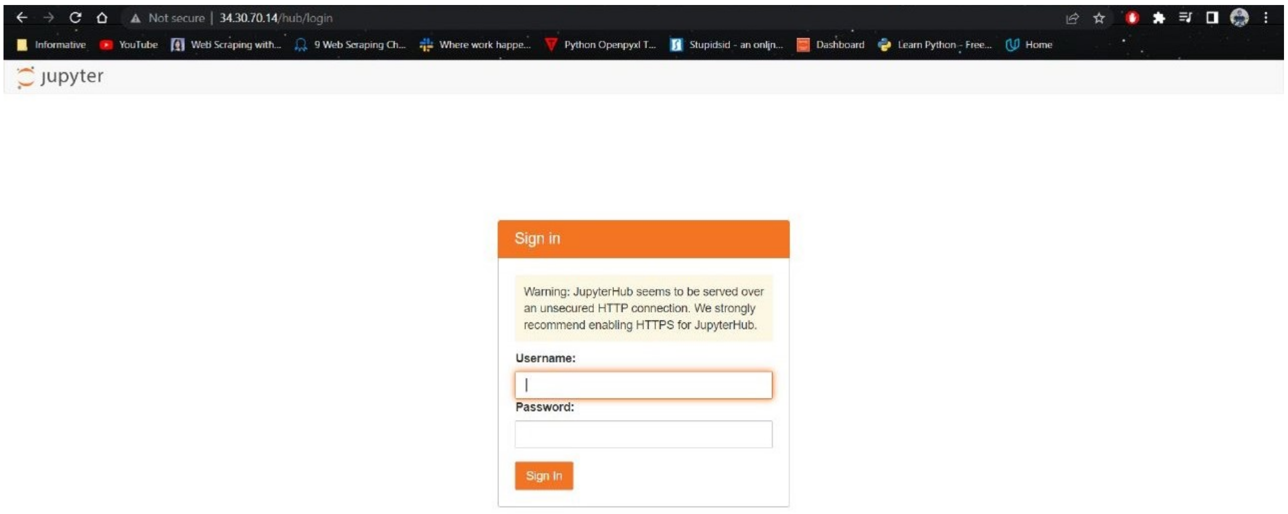
4.2 Steps to access the System



```
g191040650cloudshell:~ (studious-lyceum-384207)$ cd main2
g191040650cloudshell:~/main2 (studious-lyceum-384207)$ l
kubernetes_lab_environment/
g191040650cloudshell:~/main2 (studious-lyceum-384207)$ cd kubernetes_lab_environment/
g191040650cloudshell:~/main2/kubernetes_lab_environment (studious-lyceum-384207)$ l
check_lab.sh*  config.yaml  README.md  start_lab.sh*  storageclass.yaml  update_lab.sh*
g191040650cloudshell:~/main2/kubernetes_lab_environment (studious-lyceum-384207)$ ./start_lab.sh
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 11345  100 11345    0     0  33764      0 --:--:-- --:--:-- --:--:-- 33764
Helm v3.11.3 is available. Changing from version v3.9.3.
Downloading https://get.helm.sh/helm-v3.11.3-linux-amd64.tar.gz
Verifying checksum... Done.
Preparing to install helm into /usr/local/bin
helm installed into /usr/local/bin/helm
error: failed to create serviceaccount: serviceaccounts "tiller" already exists
error: failed to create clusterrolebinding: clusterrolebindings.rbac.authorization.k8s.io "tiller" already exists
Error from server (NotFound): deployments.apps "tiller-deploy" not found
"jupyterhub" already exists with the same configuration, skipping
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "jupyterhub" chart repository
Update Complete. *Happy Helming!*
```

Figure 4.4: GCP's Cloud Shell showcasing deployment of kubernetes cluster

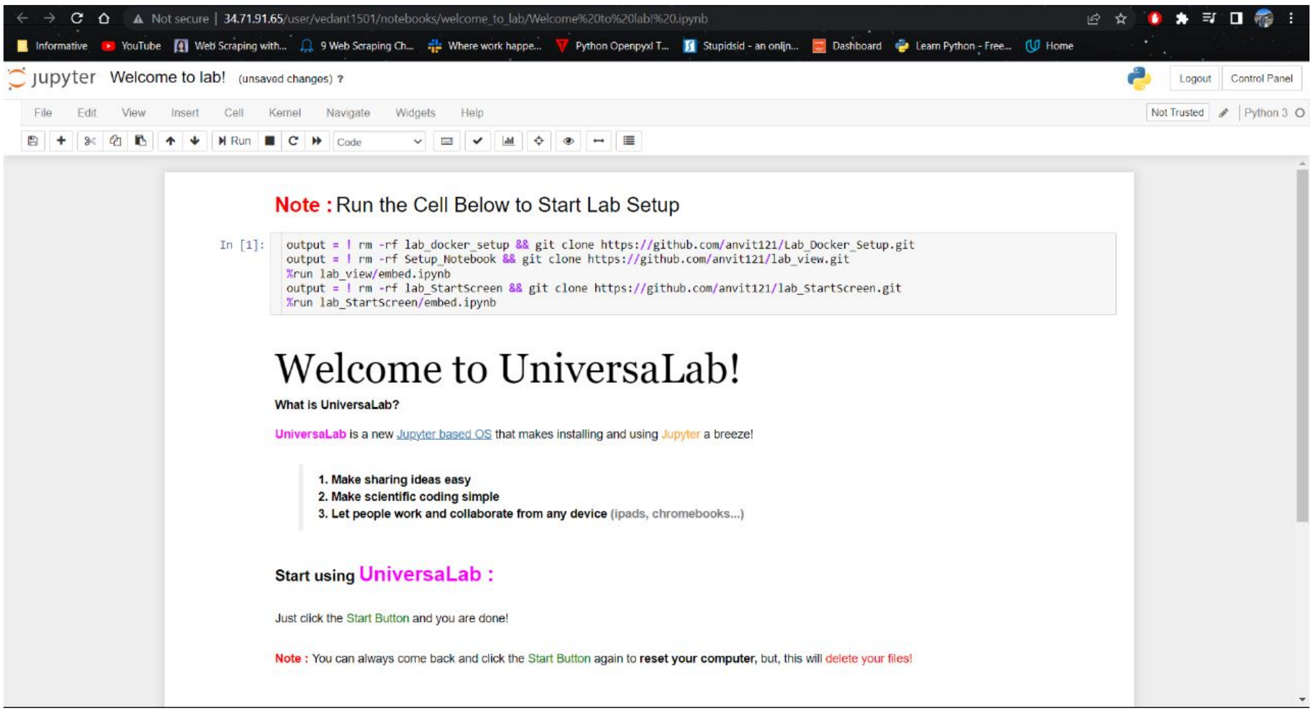
The image demonstrates a convenient way to initiate a lab on the Google Cloud Platform (GCP) using the Cloud Shell. By running the command **"./start_lab.sh"** within the Cloud Shell, users can quickly begin working on their lab without the need to set up a separate environment. This feature provides a hassle-free experience for users who want to start working on their projects immediately. With the Cloud Shell, users can easily access a preconfigured shell environment that is built on top of Google's infrastructure. This setup saves users the time and effort required to configure and maintain their own development environments, allowing them to focus on their coding and development tasks.



Warning: JupyterHub seems to be served over an unsecured HTTP connection. We strongly recommend enabling HTTPS for JupyterHub.

Figure 4.5: Accessing system designed through authentication

After executing the command (as shown in Figure 4.4), the user will be redirected to the login page of UniversaLab. Here, the user will be prompted to provide their login credentials, which will be used to authenticate their identity and log them into the platform. The login process typically requires the user to enter their email address or username and their password, which are verified against the platform's user database. Once the user is authenticated, user will be granted access to UniversaLab and can begin working on their projects within the platform's lab environment.


34.71.91.65/user/vedant1501/notebooks/welcome_to_lab/Welcome%20to%20lab!%20.ipynb
jupyter Welcome to lab! (unsaved changes)
Logout
Control Panel
File Edit View Insert Cell Kernel Navigate Widgets Help
Not Trusted
Python 3
Run
Code

Note : Run the Cell Below to Start Lab Setup

```
In [1]: output = ! rm -rf lab_docker_setup && git clone https://github.com/anvit121/Lab_Docker_Setup.git
        output = ! rm -rf Setup_Notebook && git clone https://github.com/anvit121/lab_view.git
        %run lab_view/embed.ipynb
        output = ! rm -rf lab_StartScreen && git clone https://github.com/anvit121/lab_StartScreen.git
        %run lab_StartScreen/embed.ipynb
```

Welcome to UniversaLab!

What is UniversaLab?

UniversaLab is a new Jupyter based OS that makes installing and using Jupyter a breeze!

> **1. Make sharing ideas easy**
> **2. Make scientific coding simple**
> **3. Let people work and collaborate from any device** (ipads, chromebooks...)

Start using UniversaLab :

Just click the Start Button and you are done!

Note : You can always come back and click the Start Button again to **reset your computer**, but, this will delete your files!

Figure 4.6: Snippet reflecting usage of software environment deployed over kubernetes pod

Once the user is logged in, they will be redirected to the landing page, which provides a brief introduction to UniversaLab and instructions on how to start using it. The landing page serves as a starting point for users to familiarize themselves with the platform's purpose, features, and benefits. It may include information on the lab's scope, the types of tasks that can be completed within the lab environment, and any other important details that users need to know.

Overall, the landing page plays a crucial role in helping users understand what UniversaLab is and how to use it effectively. By providing clear and concise instructions, users can quickly get up to speed and start working on their projects without any confusion or delay.

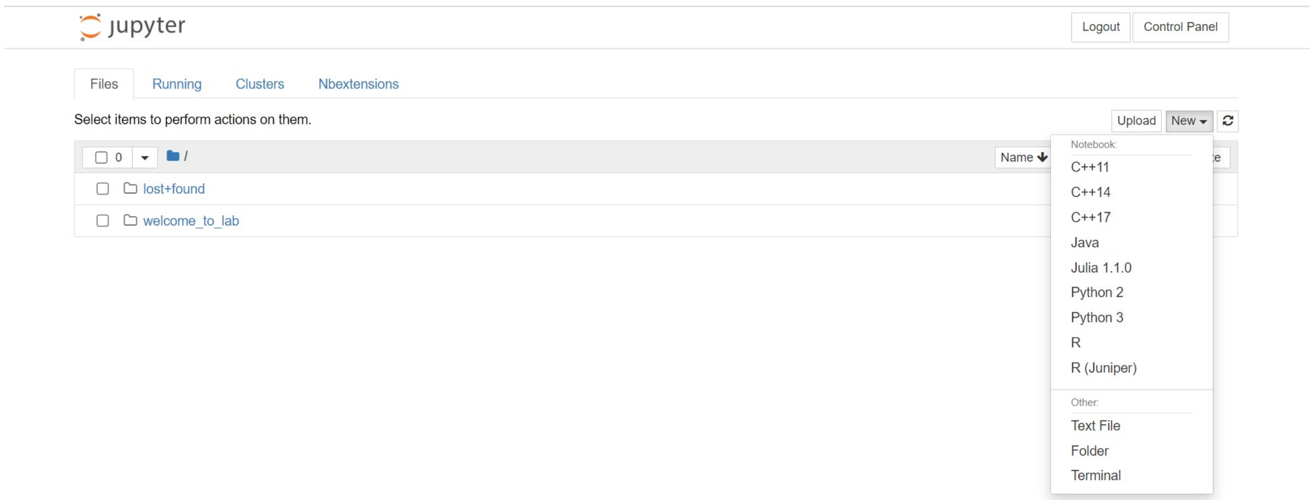
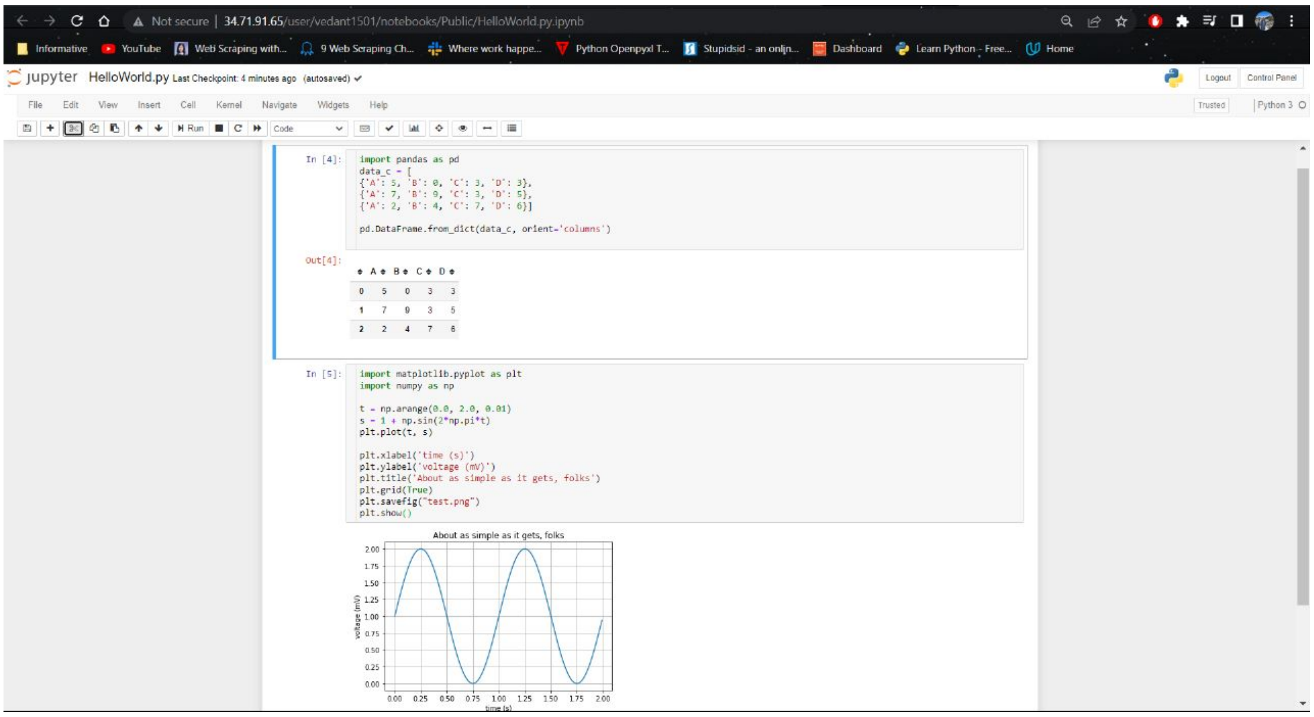


Figure 4.7: Snippet reflecting software environments available on JupyterHub deployed over kubernetes worker pod

This is the JupyterHub platform where the students can create folders and files and start coding. They can also choose from a variety of available labs to perform experiments for their practical lab sessions.

The platform supports multiple programming languages and provides access to essential libraries and frameworks for data science and machine learning. Additionally, JupyterHub enables collaborative coding, allowing students to work together on projects and share their work with their peers and instructors.

The JupyterHub IDE provides a user-friendly interface that simplifies the process of creating and managing files and folders. Additionally, it offers a seamless way for students to access and utilize the necessary resources and tools required for their lab experiments.


```
In [4]: import pandas as pd
        data_c = [
        {'A': 5, 'B': 0, 'C': 3, 'D': 3},
        {'A': 7, 'B': 9, 'C': 3, 'D': 5},
        {'A': 2, 'B': 4, 'C': 7, 'D': 6}]

        pd.DataFrame.from_dict(data_c, orient='columns')

Out[4]:
```

	A	B	C	D
0	5	0	3	3
1	7	9	3	5
2	2	4	7	6

```
In [5]: import matplotlib.pyplot as plt
        import numpy as np

        t = np.arange(0.0, 2.0, 0.01)
        s = 1 + np.sin(2*np.pi*t)
        plt.plot(t, s)

        plt.xlabel('time (s)')
        plt.ylabel('voltage (mV)')
        plt.title('About as simple as it gets, folks')
        plt.grid(True)
        plt.savefig("test.png")
        plt.show()
```

Figure 4.8: Snippet reflecting usage of Python coding environment

This is an example of Python code that can be written and executed in the JupyterHub IDE. Users can also write and execute code in various other programming languages, such as R and Java, and collaborate with colleagues. The platform provides a collaborative environment where multiple users can work on the same project simultaneously, making it easier for students and instructors to collaborate on practical lab sessions. The ability to share code and results in real-time makes it a valuable tool for collaborative research and learning.

Chapter 5

Testing

5.1 Software Testing

Software testing is a process of evaluating a software product to ensure that it meets the specified requirements and works correctly. The main goal of software testing is to identify defects or errors in the software and to ensure that the software meets the business and technical requirements, is reliable, and performs as expected. The testing process includes a series of activities that can be performed manually or using automated tools, and it typically involves testing the functionality, performance, security, usability, and compatibility of the software. The ultimate goal of software testing is to improve the quality of the software and to ensure that it meets the needs and expectations of the users.

5.2 Functional Testing

Integration testing is a software testing technique that verifies the interfaces between software components or modules work correctly when integrated with each other. This type of testing is performed after unit testing and before system testing to ensure that different software components work together as expected and the system as a whole functions correctly. The purpose of integration testing is to identify any issues or defects that may arise from the interaction between the integrated components.

Integration testing is a software testing technique that checks the compatibility between different modules of the software or system. It is performed to identify any defects that may occur when these modules are integrated with each other. Integration testing helps to verify the functional, performance, and reliability of the system by testing the interactions between different components.

In this type of testing, individual units of code are combined and tested as a group to ensure they work together seamlessly. One of the main benefits of integration testing is that it helps to detect defects early on in the development process, which reduces the cost of fixing these issues later on. By performing integration testing, developers can identify and fix any issues before the system is released to the end-users. Overall, integration testing is an essential part of the software testing process that helps to ensure the quality and reliability of the software system.

5.3 Applicability of functional testing into our project

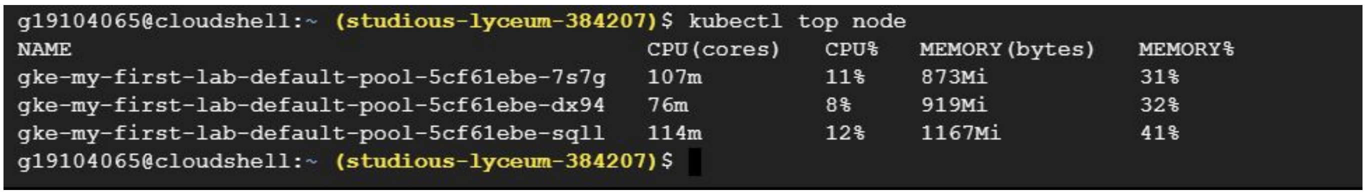
Integration testing can be used in the context of orchestrating a dynamically scalable container-based lab environment for an educational institute to ensure that all the different components of the system work together seamlessly. This can involve testing the integration between various components such as the container orchestration system, the cloud platform, the network configuration, and the monitoring and logging systems.

Manual testing can also be used to validate the integration of different components and to ensure that the system is functioning correctly from a user's perspective. This can involve conducting user acceptance testing to validate that the system meets the requirements of the educational institute and that it is easy to use and navigate.

Chapter 6

Result

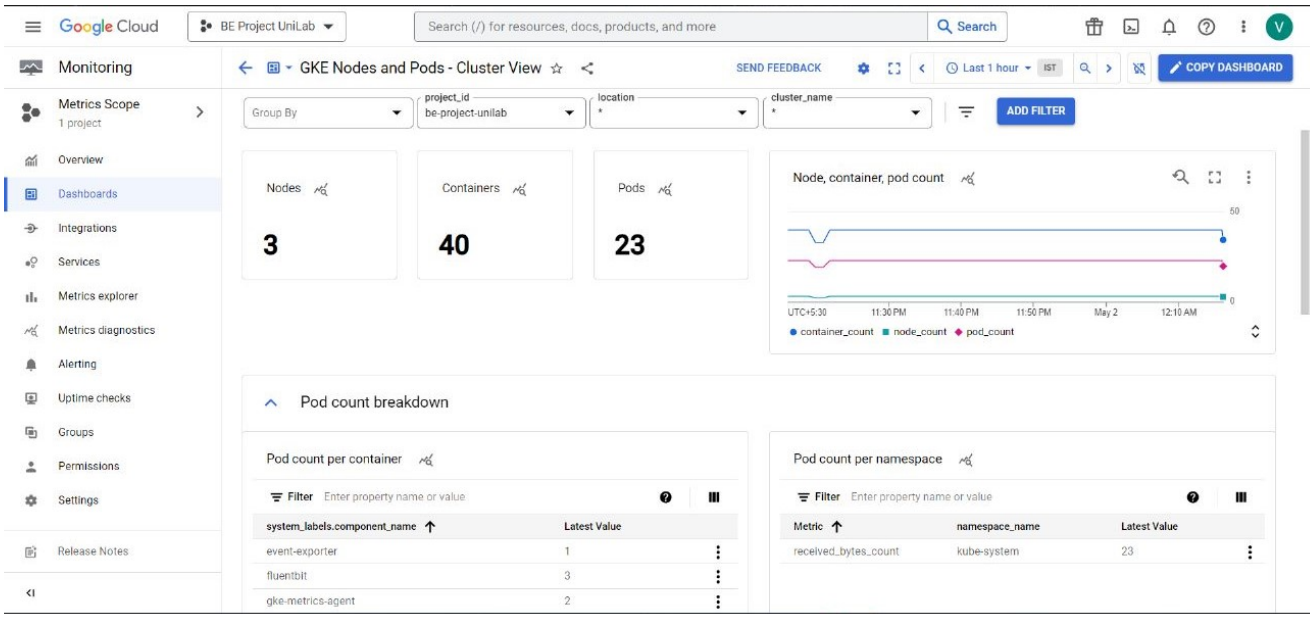
The implementation of scalable containerization using Kubernetes has provided several benefits that can be visualized through the dashboard snippets. The dashboard snippets display an overall view of how the pods and containers are working. Kubernetes provides a scalable platform for container orchestration that enables the deployment and management of containerized applications. The dashboard snippets illustrate how Kubernetes is capable of managing the containerized applications effectively by providing detailed insights into the health and performance of the containers and pods. The dashboard snippets enable administrators to monitor the resource utilization of the containers, detect and fix issues, and optimize the performance of the applications. With Kubernetes, containerized applications can be scaled up or down based on the workload demands, which helps in reducing the infrastructure cost and improving the efficiency of the application. Overall, the implementation of scalable containerization using Kubernetes has provided significant benefits in terms of scalability, flexibility, and efficiency of containerized applications.



```
g19104065@cloudshell:~ (studious-lyceum-384207)$ kubectl top node
NAME                                        CPU(cores)   CPU%   MEMORY(bytes)   MEMORY%
gke-my-first-lab-default-pool-5cf61ebe-7s7g   107m         11%    873Mi           31%
gke-my-first-lab-default-pool-5cf61ebe-dx94   76m          8%     919Mi           32%
gke-my-first-lab-default-pool-5cf61ebe-sqll   114m         12%    1167Mi          41%
g19104065@cloudshell:~ (studious-lyceum-384207)$
```

Figure 6.1: Snippet reflecting kubernetes cluster health deployed over GCP

Fig 6.1 displays crucial information about the health status of the nodes, including their CPU and memory utilization. It enables users to monitor the performance of each node, identify any potential issues, and take necessary actions to optimize the cluster's performance. The node health information is essential for ensuring the reliability and availability of applications running on the Kubernetes cluster. With this information, users can efficiently manage the resources, ensure optimal performance, and prevent any downtime or disruptions.



Nodes	Containers	Pods
3	40	23

Pod count per container

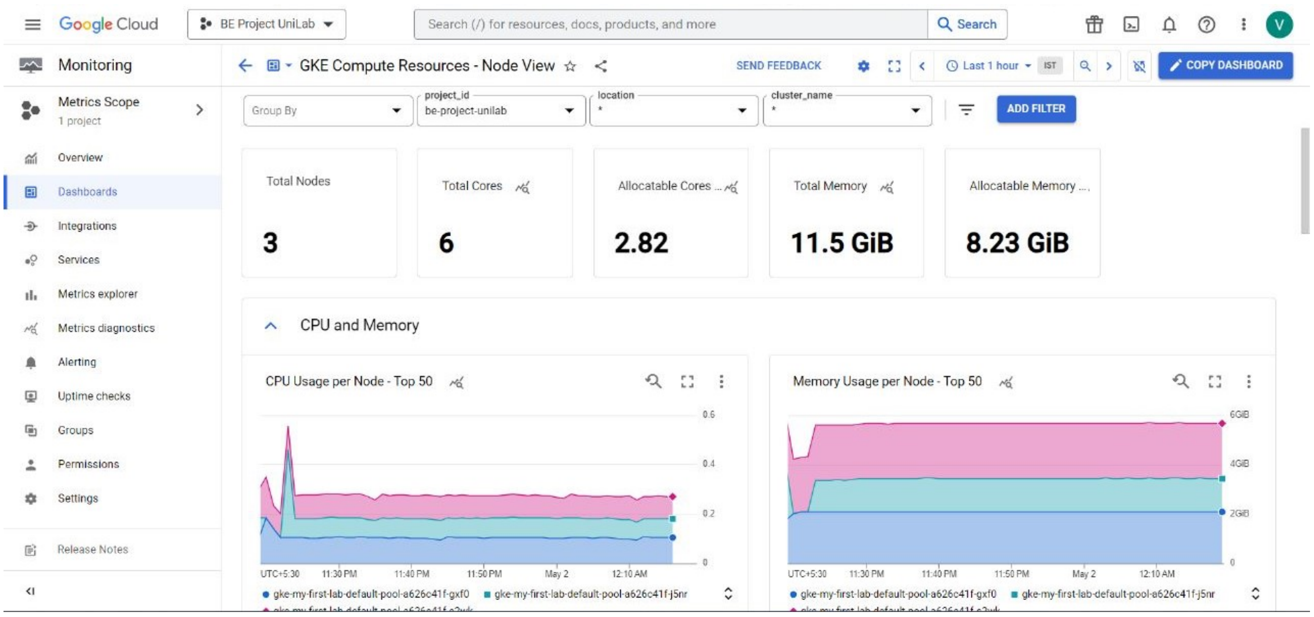
system_labels.component_name	Latest Value
event-exporter	1
fluentbit	3
gke-metrics-agent	2

Pod count per namespace

Metric	namespace_name	Latest Value
received_bytes_count	kube-system	23

Figure 6.2: GKE Nodes and Pods- Cluster View

Fig 6.2 provides a cluster view of Google Kubernetes Engine (GKE) nodes and pods, indicating the number of nodes, containers, and pods, and their timelines. This view is useful for monitoring and managing the performance of a cluster and identifying any issues that need to be addressed. It enables users to get an overview of the cluster's health and identify any resource constraints that need to be addressed to ensure optimal performance. The timeline feature also allows users to track changes and updates to the cluster over time, providing valuable insights into its operation.



Group By

project_id: be-project-unilab

location: *

cluster_name: *

Total Nodes	Total Cores	Allocatable Cores ...	Total Memory	Allocatable Memory ...
3	6	2.82	11.5 GiB	8.23 GiB

Figure 6.3: GKE Resource Usage

Fig 6.3 represents the GKE Compute Resources- Node view, which provides an overview of the computing resources available in the GKE cluster. It displays the number of nodes, the number of CPU cores, the total memory, and the allocatable memory of each node. This view helps administrators monitor the available resources and ensure that there is enough capacity to run all the applications in the cluster. Additionally, it enables them to identify any nodes that may be overutilized or underutilized, allowing them to optimize resource allocation and minimize costs.

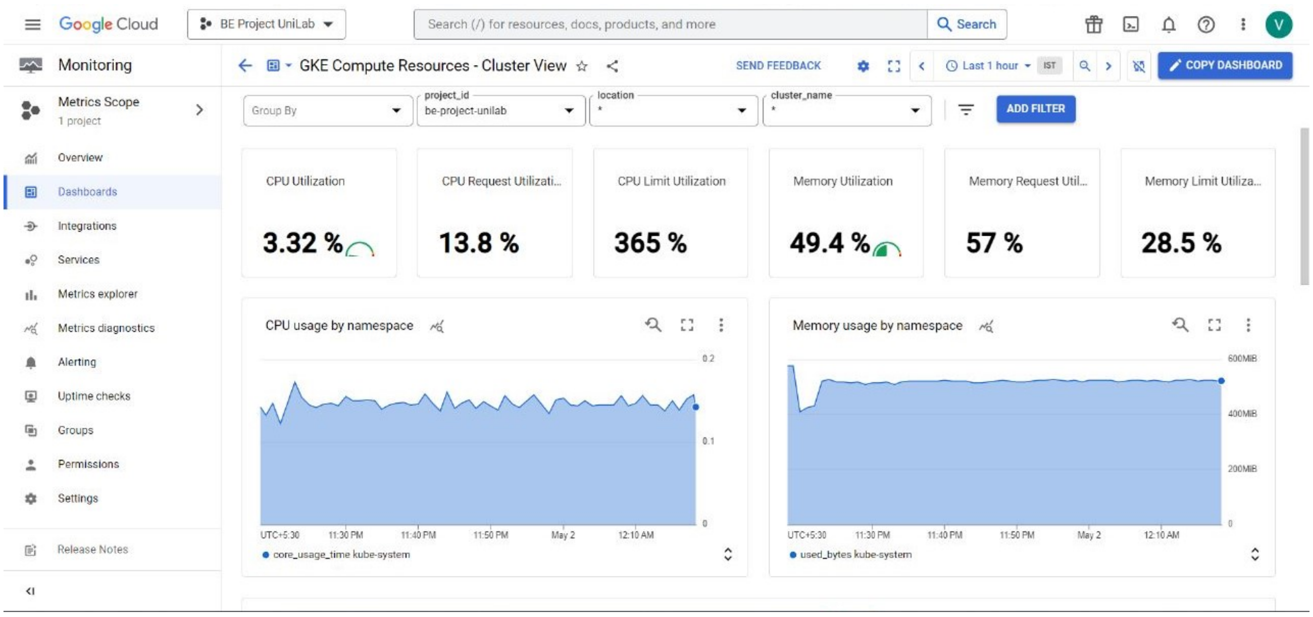
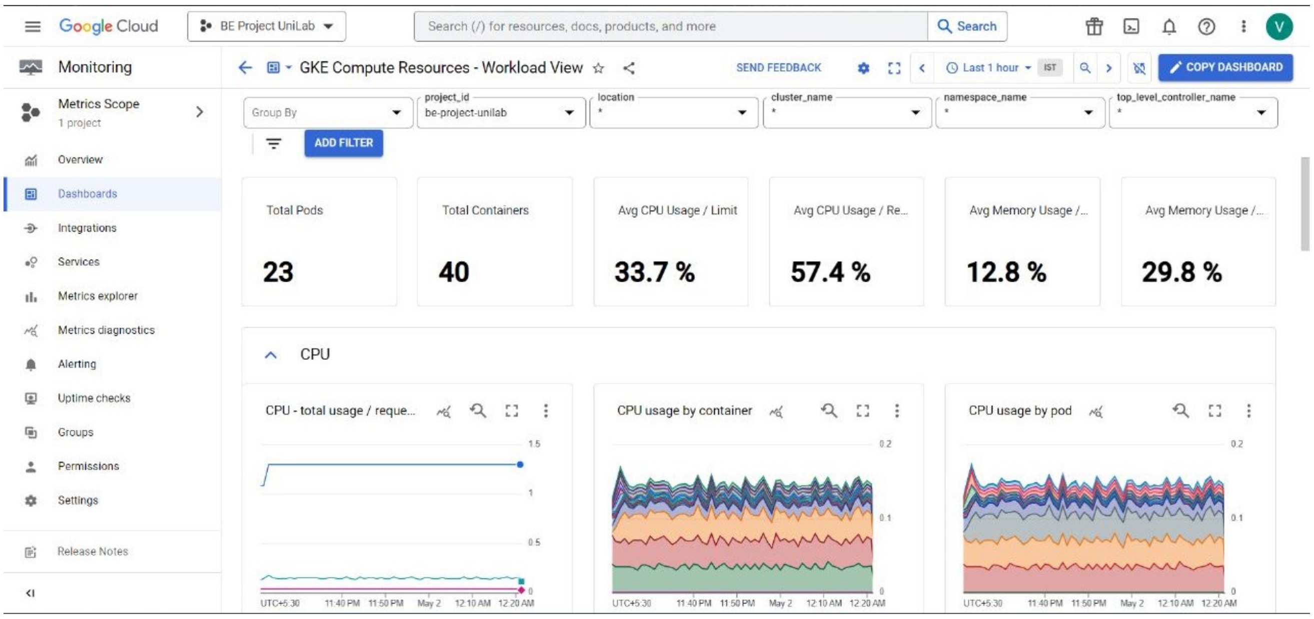


Figure 6.4: GKE Cluster view

In Fig 6.4 we can see the GKE Compute Resources in a cluster view, which provides an overview of the current state of the cluster. It displays the total number of nodes in the cluster, the total number of CPU cores available, and the memory utilization. Additionally, it shows the amount of memory allocatable to the pods and the actual usage of the memory. This view can help users to monitor the resource usage of their applications and ensure that the cluster is operating efficiently.



Total Pods 23
Total Containers 40
Avg CPU Usage / Limit 33.7 %
Avg CPU Usage / Re... 57.4 %
Avg Memory Usage /... 12.8 %
Avg Memory Usage /... 29.8 %

Figure 6.5: GKE Workload

Fig 6.5 depicts the GKE workload, which presents the total number of pods, average CPU and memory usage, and other related statistics. The image also offers a graphical representation of CPU usage by container and pod. This enables users to visualize and monitor the performance of their workloads and troubleshoot any issues related to resource utilization. The GKE workload view is a valuable tool for managing and optimizing the performance of Kubernetes clusters, ensuring the efficient allocation and utilization of resources, and enhancing overall productivity.

Chapter 7

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 faculties. Flexibility and Customization of container orchestration allow Educational Institutes to support a wide range of workloads, from traditional teaching to collaborative project management.

Thus, the system designed and implemented is an excellent solution for Educational Institutes that are looking to build and manage scalable and efficient lab environments.

Chapter 8

Future scope

The solution proposed can be extended to use this platform as a project collaboration environment, where all the necessary source 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.

In addition, experiments performed successfully by students can be saved and used by later batches as a reference to debug any difficulties faced during execution through the effective use of GitHub. Moreover, the system designed can be modified so that the students will only be able to access the lab environments that are mapped to their respective academic year and semester.

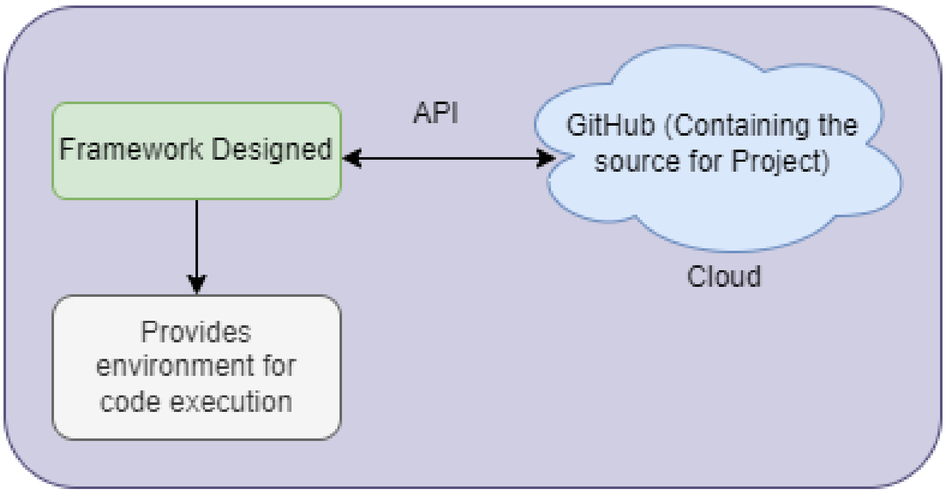

Framework Designed
API
GitHub (Containing the source for Project)
Cloud
Provides environment for code execution

Figure 8.1: Future Scope reflecting collaborative project management through system designed

Bibliography

[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/.

Appendices

Appendix- A: Google Cloud Platform Setup

1. Open GCP and login with your Google Account.

2. Create a Kubernetes Cluster with GKE.

3. Set the number of nodes and number of pods to be created with the cluster.

4. Set the disk space required for the virtual machine.

5. Create the cluster.

Appendix- B: Container Images

1. Enable the container registery api in the GCP Dashboard.

Appendix- C: Connect Virtual Machine to Cloud Console

1. Connect Virtual Machine to the cloud console.

2. Paste the command given by GCP in the console to connect the virtual machine.

Appendix- D: Pulling the required software dependencies from GitHub

1. Clone the
"https://github.com/anvit121/kubernetes_lab_environment" repository in the virtual machine

Appendix- E: Running the project

1. Use the script
"./start_lab.sh"
in the cloud console to run the project.

2. It provides the user with the IP address of the server where jupyterhub is working, this IP is to be pasted in the URL.

3. It provides the user with the JupyterHub interface which is to be used by the user for coding purposes.

Publication

Paper entitled **“Orchestrating dynamically scalable container-based lab environment for an Educational Institute”** is submitted at **“International Conference on Advanced Computing Technologies and Applications-2023”** by **“Vedant Patil, Mudra Limbasia, Anvit Mirjurkar, and Dr. Kiran Deshpande”.**