

A Project Report
on
CONTAINER MANAGEMENT WEB APPLICATION

*Submitted in partial fulfillment of the
requirement for the award of the degree of*

B.Tech In Computer Science and
Engineering



Under The Supervision of
C. Ramesh Kumar
Assistant Professor

Submitted By
Vishal Kumar 19SCSE1010126
Mayank Ahuja 19SCE1010595

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA
INDIA
March, 2023

Abstract

According to the systems which are going on or rather say the technologies which are being used in the systems nowadays are basically dependent on the commands, it means user needs to learn the docker commands to use the existing docker or container management web applications. The main issue or the problem statement is to learn the different commands to use the different container management systems, because it makes very difficult to learn the commands and to use such applications for a non-tech persons. Many already introduced systems such as Docker. It has all the containers, but to use it we should have the knowledge of some of the commands of it, which is really difficult for a non-tech person. Learning of docker commands is necessary in that application.

So, to overcome this situation our project idea comes in where the user does not require to learn any of the commands and it can be used by any of the user, such as in testing any of the softwares we require a container management system to test the software in every operating systems. Which will take much time by installing and then testing the softwares but through our proposed system the operating systems can be launched within seconds and can be used by the users easily. In our proposed system there is no use of learning the commands the user needs to only click the button and can simply launch any of the operating systems cause it easily on the command-line. The things that were more difficult to us were the managing the different ISO files of the different operating systems, because we were creating the button to simply click on it and it should work so all the files and the resources should be made exactly according to the plan. The main tools which would require to complete the whole project is HTML5, CSS and the basics of Javascript. We will launch our project on the web, So basically we are making, a web application to complete our project. At the back-end we will use the disk (iso images) of all the

containers and will make use of Javascript and HTML to concatenate it with the button which we will provide for the user. The basic knowledge of cloud is also important for the proposed project just to maintain the different ISO files and the types of the operating systems.

So, our project will be very beneficial for the future use, for saving the time because it can complete the work in very less time and there is no need of learning the commands to the users. Docker has been tipped as the future of virtualisation. Its popularity is definitely growing, especially with companies like Netflix, Spotify, PayPal and Uber using the containerisation system. Hyve provides hosting for Docker containers on our PrivateDocker platform.

Table of Contents

Title	Page No.
Abstract	I
Chapter 1 Introduction	1
1.1 Introduction	2
1.2 Formulation of Problem	2
1.2.1 Tool and Technology Used	
Chapter 2 Literature Survey/Project Design	3-4

CHAPTER-1

Introduction

Container management platforms facilitate the organization and virtualization of software containers, which may also be referred to as operating-system-level virtualizations. Developers use containers to launch, test, and secure applications in resource-independent environments. Containers house components of applications, libraries, or groups of source code that can be executed on demand. The management platforms help users allocate resources to optimize efficiency and balance system workloads. Containers provide a flexible, portable platform to organize, automate, and distribute applications. Companies use container management software to streamline container delivery to avoid the complexities of interdependent system architectures. The tools are scalable and can greatly improve the performance of widely distributed applications. Implementation and evaluation of a container management platform.

In the history of computing, Containers have unique recognition because of its importance in virtualization of infrastructure. Unlike traditional Hypervisor virtualization where one or more independent machines run virtually on physical hardware via an intermediate layer, containers run the user space on top of the operating system kernel. Containers provide the isolation between multiple user work space instances. Because of this unique feature container virtualization is often referred to as operating system level virtualization. Instead of starting a complete operating system on the host operating system containers shares the kernel with the operating system which eliminates the overheads and it also provide isolation between the applications. These features of the containers make it possible to ship the small container which acts as a complete operating system which encapsulates only those files which are needed to run our desired applications. This paper exclusively discusses about one of the containers technology which is currently being used in many production environments to package their

applications in isolated environment. This newly evolved containers are none other than Docker which has changed the perspective of deploying the applications in production environment. Docker is an open-source engine which was introduced by Docker Inc in 2013 under apache 2.0 license. The primary goal of the Docker is to provide fast and lightweight environment in which to run the developers code as well as the efficient workflow to get that from the Dev environment to test environment and then into production Environment. Docker containers are built from applicationimages which are stored and managed in Docker hub. Users can also create their own Docker registries to store their customized images which are created from a Docker file or from an existing container. These flexible functionality features of Docker have made it popular with in no time. In today's competitive environment using the available resources efficiently has becomeimperative for IT organizations. The traditional virtualization techniques which are being used to create virtual machines from few past years, has shown some degradation in the performance of applications which are deployed on those virtual machines. Data center size of these organizations have also been increasing because of these obsolete virtualizations techniques which in turn increasing the cost of infrastructure. Even though the public clouds like Amazon Web Services (AWS), Microsoft Azure have been evolved in past few years to solve these increasing size of private data centers but unfortunately using large number of servers on these public clouds has become an overhead for the organizations because of the monetary constraints. So, it has become a highly important concern for any organization to solve this problem to sustain in today's competitive environment.

There is also been a huge issue in developing and deploying applications in development and production environments. Applications working perfectly fine in Development environment have been showing glitches when they are deployed in production environment. This environment issues have widened the gap between Development and Operation Teams. Further, this issue has led to the slower delivery of the software with increasing the cost of

maintenance. This shows that there is an urgent need of a technology which can deliver the reproducible environments to avoid difference development and production environments. The proposed study uses the mix of Qualitative and Quantitative approach as we are comparing the security features and resource utilization of containers and virtual machines. Docker containers have been used to support the study and detailed explanation of these technology has been provided in the following paragraphs.

Before moving on to discussing how Docker works let us first understand how Docker are used in creating the containers with in no time and how its functionality is different from traditional Virtual machines. The primary objective of this study is to discuss the available alternative solutions for the specified problem statement. This paper introduces a new virtualization technique using Docker containers which is as an alternative solution for the traditional Virtual Machines for reproducible environments. This paper also compares the security and performance of the applications running in the Containers and Virtual Machines and their resource utilisation.

CHAPTER-2

Literature Survey

In “Using Docker to support reproducible Research” R. Chamberlain and J. Schommer[1] stated that reproducibility and sharing of an environment is imperative factor for an organization to make faster operations. A brute-force approach to achieve this reproducibility and sharing of an environment is through virtual machines. Virtual machines are safe and predictable way to share a complete computational environment. However, there are serious drawbacks in using Virtual machines for reproducibility and sharing of resources. Firstly, it is very hard for a user to do this reproducibility without the very high-level knowledge of Systems administration. Secondly, Virtual Machines consume lots of storage space irrespective of the applications or processes running on them. The below figure depicts how virtualization has been achieved using Virtual Machines and the problem Linux Containers (LXC) have been introduced to solve the problem of resource utilization which was created by virtual machines. These containers do not need a separate hypervisor to create an isolated environment. In a large-scale environment using Virtual Machines would mean you are probably running many duplicate instances of the same OS and many redundant boot files which are not required [2]. Containers are lightweight compared to VM's since they contain only the bootable files specific to that application. Since Containers decouple the applications from operating systems users can have a clean and minimal Linux operating system and they can run in one or more isolated containers.

These Linux Containers (LXC) are designed for operating system level virtualization method for running multiple Linux containers on the single Linux Host machine. Cgroups and Namespaces are the two primary features that make this Linux containers possible. Linux Cgroups are developed by google, which governs the isolation and usage of system resources like CPU and memory usage for a group of processes. Consider an example application which takes up a lot

of CPU cycles and memory we can put this application in a Cgroup to limit the usage of CPU and memory by that application.

Docker is an open source platform based on Linux Containers (LXC) which completely packages software applications. It is backed by a private company that focuses on providing a platform which is easy and scalable for hosting web applications. As we discussed in the above sections LXCs provide a completely operating system level virtualization which creates a sandboxed virtual environment in Linux that eliminates the overhead without creating a complete virtual machine. Docker extends this terminology of LXCs to make it user friendly and provides easy versioning, distribution, and deployment.

These Docker containers can be launched in a sub second, and then you can have a hypervisor that sits directly on top of the operating system. By this we can pack a lot of the containers on a single Physical or Virtual Machine. This gives an added advantage of effective usage of available resources.

Docker, allows there to be just one host operating system, and provides a layer of software at the top of the operating system that isolates multiple applications and their required supporting stacks of software from each other, and from the operating system. Docker are created to provide lightweight and fast environment in which to run users code with efficient workflow and get that code from user's laptop to test environment and then into production environment . Docker is very simple because one can run it on simple host which has nothing but a compatible Linux kernel and binary files. The mission of Docker is to provide following features:

Easy and lightweight way to model reality. Docker are so fast such that one

can easily containerize their applications within minutes. Users can modify their applications and dockerize their applications within no time. When a change is applied to an application a new

container will be created to run these modified applications. Unlike Virtual machines which uses hypervisor Docker containers takes only seconds to launch. Then the modified applications are packaged into the newly created containers.

Logical segregation of duties. With Docker, it has become easy for an organization to segregate the duties between Development and Operations teams. Development focuses on developing the applications inside the containers while operations team focuses on managing these containers. Docker enhances the consistency by providing the same environments in which Developers write the code and operations team deploy the code. This methodology removes the conflicts between Dev and Ops teams by resolving “worked in Dev, failed in Ops” problem.

Fast and efficient application lifecycle development. The downtime in the production environment can be drastically reduced by using Docker. They reduce the cycle time between code being written by developers and code being tested, deployed by the operations team into the production environment.

With the above features, Docker have resolved many challenges like Dependency Hell, imprecise documentations, tackling code-rot with image versions and barriers to adoption and research. Docker containers also enhances the security features of application in two ways. One is by providing isolation between application and another is providing isolation between application and host system. They also reduce the host surface area to protect both the host and co-located containers by restricting access to the host. Docker Containers also enhances security by providing Process restrictions, Device and file restrictions, application image security and open-source security and other Linux kernel security features. Inside the containers unprivileged users cannot be added to root group so that they won't have privileges like sudo. This improves the overall security of the applications and makes running applications inside the containers more secure.

CHAPTER-3

Methodology/Implementation

In cloud computing, PaaS (Platform as a Service) is a service mode that can provide users with software platforms and development environments [22–25]. Docker is an open-source project realized on the basis of Golang and the application of Docker can accelerate the realization of PaaS mode [26]. In software development, we usually encounter software reuse failures caused by development environment difference, while Docker can effectively solve such problems. Whether during the development stage or test stage of software, Docker can create the same environment. In other words, it intervenes in the container configuration to guarantee the consistency of the dependence of all configurations within the container and to enable that the development, test, and release of software all occur in the same environment. Docker has three advantages. First, it can use the image repository. Second, it can conduct continuous deployment test. Third, it possesses high level of resource utilization. In this paper, the role-based authority management system is deployed on the Docker platform. After the server is started and the images are obtained, the installation of operating systems, repositories, and application services will be completed, which greatly shortens the system development process. Just because Docker does not need to start the slave operating system, enough disk space is spared, abundant system resources are saved, and the utilization rate of system resources is improved. It can also be considered that the virtualization of virtual machines is manifested in hardware level and the virtualization of Docker is manifested in operating system level.

Basic Model Building.

Build the authority management system model on the basis of the role-based access control technology. Due to the large scale of the system, we can not conduct authority setting and role division for every user in traditional ways. i.e concept of user group should be introduced and users with the same authorities and characteristics should be divided into the same user group without changing the many-to-many relationship in role-based access control. One user can be divided

into multiple user groups and one user group relates to multiple users. This can not only reduce the heavy and complicated workload but also facilitate subsequent management. Take the task management system as an example. Users can be divided into two user groups: first, the project participant user group, and second, the project administrator user group. Different user groups have different authorities. For example, roles in the administrator user group have authorities to modify and delete the project while roles in the participant user group only have authorities to preview and edit the tasks but cannot modify or delete the project.

In user information management systems, authority control is one necessary link. The so-called authority means that users need to access authorized content in accordance with the security rules set by the system. Authority control can effectively identify user identity and avoid illegal invasion; therefore, authority control has been widely promoted in numerous user login systems. Current authority control technologies can fall into two categories: the system level security management, such as operating system level security management and database level authority management, etc., and the application-level security management which generally is closely related to specific system requirements. From the perspective of implementation, authority control mainly employs three types of models: (1) the discretionary access control model; (2) the mandatory access control model; and (3) the role-based access control model.

It means that the subject with control power can authorize the access right of the object to other subjects [27]. In such access management model, one user can have different authorities to different resource objects while different users can have different authorities to one same resource object; one user can authorize his authority to other users without any limitation. In the discretionary access control model, users can formulate corresponding protection strategies for the resource object to be protected according to system requirements. Discretionary access control has

advantages such as flexible authority distribution, simple and easy-to-use models, and strong expansibility; however, the discretionary pattern of authorization leads to low security level of the whole system while the high complexity of discretionary authorization is also a hard nut to crack.

There are four main internal components of docker, including Docker Client and Server, Docker Images, Docker Registries, and Docker Containers. The docker server gets the request from the docker client and then process it accordingly. The complete RESTful (Representational state transfer) API and a command line client binary are shipped by docker. Docker daemon/ server and docker client can be run on the same machine or a local docker client can be connected with a remote server or daemon, which is running on another machine [9].

Docker image creates a docker container. Containers hold the whole kit required for an application, so the application can be run in an isolated way. For example, suppose there is an image of Ubuntu OS with SQL SERVER, when this image is run with docker run command, then a container will be created and SQL SERVER will be running on Ubuntu OS.

There are two methods to build an image. The first one is to build an image by using a read-only template. The foundation of every image is a base image. Operating system images are basically the base images, such as Ubuntu 14.04 LTS, or Fedora 20. The images of operating system create a container with an ability of complete running OS. Base image can also be created from the scratch. Required applications can be added to the base image by modifying it, but it is necessary to build a new image. The process of building a new image is called “committing a change”. The second method is to create a docker file. The docker file contains a list of instructions when “Docker build” command is run from the bash terminal it follows all the instructions given in the docker file and builds an image. This is an automated way of building an image. Hypervisor is lying between host and guest operating systems. It is a virtual platform and it handles more than one operating system in the server. It works between the operating system and

CPU. The virtualization divides it into two segments: the first one is Para-Virtualization and the second one is Full Virtualization [3]. Figure 3 depicts the architecture of the Docker Container. Linux containers are managed by the docker tool and it is used as a method of operating system level virtualization. Figure 3 shows that in single control host there are many Linux containers, which are isolated. Resources such as Network, Memory, CPU, and Block I/O are allocated by Linux kernel and it also deals with cgroups without starting virtualization machine [8].

Hypervisor is lying between host and guest operating systems. It is a virtual platform and it handles more than one operating system in the server. It works between the operating system and CPU. The virtualization divides it into two segments: the first one is Para-Virtualization and the second one is Full Virtualization [3]. Figure 3 depicts the architecture of the Docker Container. Linux containers are managed by the docker tool and it is used as a method of operating system level virtualization. Figure 3 shows that in single control host there are many Linux containers, which are isolated. Resources such as Network, Memory, CPU, and Block I/O are allocated by Linux kernel and it also deals with cgroups without starting virtualization machine [8].

KEY DOCKER USECASES [10]

A. Simplifying Configuration - This is a basic use case; the Docker configurations can be used several times in a diversity of environments. VMs hold a plus when it comes to running any platform with its configurations, the same is included by the Dockers, but without any Virtual Machine overhead, and allows to put into the code your configurations and environment and deploy the code. The infrastructure requirements are disassociated from the application environment. When it comes to the real-life use case, it enables in accelerating the project setup in the organizations by allowing them to dive into development directly by skipping the repetitive job of environment settings and configuration procedures. **B.**

Code Pipeline Management - The prior use case majorly influences the code pipeline management. By the time the code that is written in a developer environment passes through various stages(each of different platforms/environments) and approaches the production stage, a few minor differences could be observed; However, Dockers provide a consistent environment along all the stages from development to production, facilitating an easy development and deployment pipeline. Also, the stable nature of the Docker image and the ease with which it can be started can add up in achieving the aforementioned pipeline management.

C. Docker for Development Productivity - Docker facilitates achieving the 2 goals in a developer environment - First is to keep a developer close to production, this is made achievable by the Docker as it has no or low overhead when it comes to working remotely. The second is to have a development environment active for interactive use; Docker facilitates this by making the application code accessible to the container from the container's host OS by its shared volumes. This fetches benefits like - enabling the developer to modify the source code from the platform of his choice and also observe the same.

D. Multi-Tenancy - Docker helps to prevent major application rewrites and hence is used in multi-tenant applications. Multi-tenant applications' codebases are considerably more complex, rigid, and obstinate to manage. Redesigning an application consumes a surplus of time and money. Docker usage eases the creation of confined environments to running multiple instances of applications for each tenant and makes the process economical. The easy-to use API provided by the Docker's enables to spin up containers programmatically.

E. Debugging Capabilities - Docker provides various tools that work skilfully with containers concept. To name a few of its provisions, one being the ability to checkpoint containers and its versions, to differentiate two containers, hence readily fixing applications whenever necessary.

In our project we basically made use of the front-end part in which we had used the HTML5 and the CSS technologies. Through the front-end part we made our basic block of our project which each and every user can see. So, there is a good environment-friendly design of our project , so all the users can use it very clearly. Even, a non-tech person can use our system as it is very environment friendly.

And for the back-end part of our project we had used Javascript which is the basic-building block of our project. With, the help of the Javascript only we can connect our project with the front-end part and the technology can work accordingly. In, our project we are making a Container-Management Web application which will provide different operating systems to work upon. Such as, if anyone has to check that if a software is compatible with every operating system or not, then they can use our software , so without using a single command they can use our project and can check their software accordingly. Then will simply have click on the button of the operating system which they want to use, and the operating system will be opened in the command prompt, and the users can use it.

So, I to make our project knowledge of Html,Css and Javascript is very important.

HTML:-

HTML stands for HyperText Markup Language. It is used to design web pages using the markup language. HTML is the combination of Hypertext and Markup language. Hypertext defines the link between the web pages and markup language defines the text document within the tag that define the structure of web pages.

HTML is used to create the structure of web pages that are displayed on the World Wide Web (www). It contains Tags and Attributes that are used to design the web pages. Also, we can link multiple pages using Hyperlinks. The basic structure of an HTML page is laid out below. It contains the essential building-block elements (i.e. doctype declaration, HTML, head, title, and body elements) upon which all web pages are created.

The HyperText Markup Language or HTML is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript. Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects such as interactive forms may be embedded into the rendered page. HTML provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes, and other items. HTML elements are delineated by tags, written using angle brackets. Tags such as `` and `<input />` directly introduce content into the page. Other tags such as `<p>` surround and provide information about document text and may include other tags as sub-elements. Browsers do not display the HTML tags but use them to interpret the content of the page.

HTML can embed programs written in a scripting language such as JavaScript, which affects the behavior and content of web pages. The inclusion of CSS defines the look and layout of content. The World Wide Web Consortium (W3C), former maintainer of the HTML and current maintainer of the CSS standards, has encouraged the use of CSS over explicit presentational HTML since 1997.[2] A form of HTML, known as HTML5, is used to display video and audio, primarily using the <canvas> element, in collaboration with java.

Hyper Text: HyperText simply means "Text within Text." A text has a link within it, is a hypertext. Whenever you click on a link which brings you to a new webpage, you have clicked on a hypertext. HyperText is a way to link two or more web pages (HTML documents) with each other.

Markup language: A markup language is a computer language that is used to apply layout and formatting conventions to a text document. Markup language makes text more interactive and dynamic. It can turn text into images, tables, links, etc.

Web Page: A web page is a document which is commonly written in HTML and translated by a web browser. A web page can be identified by entering an URL. A Web page can be of the static or dynamic type. **With the help of HTML only, we can create static web pages.**

Css:-

Cascading Style Sheets, fondly referred to as CSS, is a simply designed language intended to simplify the process of making web pages presentable. CSS allows you to apply styles to web pages. More importantly, CSS enables you to do this independent of the HTML that makes up each web page. It describes how a webpage should look: it prescribes colors, fonts, spacing, and much more. In short, you can make your website look however you want. CSS lets developers and designers define how it behaves, including how elements are positioned in the browser.

While html uses tags, css uses rule sets. CSS is easy to learn and understand, but it provides powerful control over the presentation of an HTML document.

- **CSS saves time:** You can write CSS once and reuse the same sheet in multiple HTML pages.
- **Easy Maintenance:** To make a global change simply change the style, and all elements in all the webpages will be updated automatically.
- **Search Engines:** CSS is considered a clean coding technique, which means search engines won't have to struggle to "read" its content.
- **Superior styles to HTML:** CSS has a much wider array of attributes than HTML, so you can give a far better look to your HTML page in comparison to HTML attributes.
- **Offline Browsing:** CSS can store web applications locally with the help of an offline cache. Using this we can view offline websites.

CSS is designed to enable the separation of content and presentation, including layout, colors, and fonts.[3] This separation can improve content accessibility; provide more flexibility and control in the specification of presentation characteristics; enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file, which reduces complexity and repetition in the structural content; and enable the .css file to be cached to improve the page load speed between

the pages that share the file and its formatting. Separation of formatting and content also makes it feasible to present the same markup page in different styles for different rendering methods, such as on-screen, in print, by voice (via speech-based browser or screen reader), and on Braille-based tactile devices. CSS also has rules for alternate formatting if the content is accessed on a mobile device.[4] The name cascading comes from the specified priority scheme to determine which style rule applies if more than one rule matches a particular element. This cascading priority scheme is predictable. The CSS specifications are maintained by the World Wide Web Consortium (W3C). Internet media type (MIME type) text/css is registered for use with CSS by RFC 2318 (March 1998). The W3C operates a free CSS validation service for CSS documents.[5]

In addition to HTML, other markup languages support the use of CSS including XHTML, plain XML, SVG, and XUL.

JavaScript:-

JavaScript (JS) is the world's most popular lightweight, interpreted compiled programming language. It is also known as a scripting language for web pages. It can be used for Client-side as well as Server-side developments. JavaScript often abbreviated as JS, is a programming language that is one of the core technologies of the World Wide Web, alongside HTML and CSS. As of 2022, 98% of websites use JavaScript on the client side for webpage behavior, often incorporating third-party libraries. All major web browsers have a dedicated JavaScript engine to execute the code on users' devices. JavaScript is a high-level, often just-in-time compiled language that conforms to the ECMAScript standard.[10] It has dynamic typing, prototype-based object-orientation, and first-class functions. It is multi-paradigm, supporting event-driven, functional. It has application programming interfaces (APIs) for working with text, dates, regular expressions, standard data structures, and the Document Object Model (DOM). The ECMAScript standard does not include any input/output (I/O), such as networking, storage, or graphics facilities. In practice, the web browser or other runtime system provides JavaScript APIs for I/O. JavaScript engines were originally used only in web browsers, but are now core components of some servers and a variety of applications. The most popular runtime system for this usage is Node.js.

Although Java and JavaScript are similar in name, syntax, and respective standard libraries, the two languages are distinct and differ greatly in design.

JavaScript can be added to your HTML file in two ways:

- **Internal JavaScript:** We can add JS code directly to our HTML file by writing the code inside the `<script>` tag. The `<script>` tag can either be placed inside the `<head>` or the `<body>` tag according to the requirement.
- **External JavaScript File:** We can create a file with `.js` extension and paste the JS code inside it. After creating the file, add this file in `<script src="file_name.js">` tag inside `<head>` tag of the HTML file.

NodeJs:-

Node.js is an open-source and cross-platform runtime environment built on Chrome's V8 JavaScript engine for executing JavaScript code outside of a browser. You need to recollect that NodeJS isn't a framework, and it's not a programming language. It provides an event-driven, non-blocking (asynchronous) I/O and cross-platform runtime environment for building highly scalable server-side applications using JavaScript.

Node.js is an open-source server environment. Node.js is cross-platform and runs on Windows, Linux, Unix, and macOS. Node.js is a back-end JavaScript runtime environment. Node.js runs on the V8 JavaScript Engine and executes JavaScript code outside a web browser. Node.js lets developers use JavaScript to write command line tools and for server-side scripting. The functionality of running scripts server-side produces dynamic web page content before the page is sent to the user's web browser. Consequently, Node.js represents a "JavaScript everywhere" paradigm, unifying web-application development around a single programming language, rather than different languages for server-side and client-side scripts.

Node.js has an event-driven architecture capable of asynchronous I/O. These design choices aim to optimize throughput and scalability in web applications with many input/output operations, as well as for real-time Web applications (e.g., real-time communication programs and browser games). The Node.js distributed development project was previously governed by the Node.js Foundation,[8] and has now merged with the JS Foundation to form the OpenJS Foundation. OpenJS Foundation is facilitated by the Linux Foundation's Collaborative Projects program.

Advantages of NodeJs:-

1. Easy Scalability
2. Real time web apps
3. Fast Suite
4. Easy to learn and code
5. Advantage of Caching
6. Data Streaming
7. Hosting
8. Corporate Support

Front-end Code For The Project:-

```
<div class="div2 div"><!-------div2----->
```

```
<label class="label">Select an Image to download:</label>
```

```
<select id="img" name="img_name" class="select button ">
```

```
<option>ubuntu</option>
```

```
<option>centos</option>
```

```
<option>postgres</option>
```

```
<option>redis</option>
```

```
<option>alpine</option>
```

```
<option>node</option>
```

```
<option>mysql</option>
```

```
<option>traefic</option>
```

```
<option>busybox</option>
```

```
<option>python</option>
```

```
<option>nginx</option>
```

```
<option>openjdk</option>
```

```
<option>golang</option>
```

```
<option>httpd</option>
```

```
<option>mongo</option>
```

```
<option>debian</option>
```

```
<option>consul</option>
```

```
<option>ruby</option>
```



```
<option>amazonlinux</option>
```

```
<option>mariadb</option>
```

```
<option>elasticsearch</option>
```

```
</select>
```

```
</div>
```

```
<div class="div3 div" style="display:inline-flex"><!-------
```

```
div3----->
```

```
<label class="label d5"> To Download Image: </label>
```

```
<button onclick="downloadImg()" class="button d5 to1"> click here</button>
```

```
</div> <div id="dwnld_status"></div>
```

```
<hr width="100%"/>
```

```
<div class="div4 div">
```

```
<label class="label">Select an Image to Delete</label>
```

```
<select id="imag" name="imag_name" class="select1 button" >
```

```
<option>centos</option>
```

```
<option>ubuntu</option>
```

Back-end Code For The Project:-

```
const express=require("express")

const app=express()

const { exec }= require("child_process")

app.listen(999, () => {console.log("Server started..")})

app.use('/assets',express.static('assets'))

app.get("/", (req,res) => {

    res.sendFile(__dirname+"/frontend.html");

})

app.get("/run", (req,res) =>{

    const os_name=req.query.os_name;

    const image_name=req.query.image_name;

    exec("docker run -dit --name "+os_name+" "+image_name, (err, stdout, stderr) => {

        res.send("<pre>"+stdout+"</pre>")

    })

})

})
```

```

app.get("/list", (req,res) =>{

    exec("docker ps | tail -n +2", (err,stdout,stderr) =>{

        let a=stdout.split("\n");

        res.write("<table border='2px solid blue'>")

        res.write("<tr><th>OS Name</th><th>OS Id</th><th>OS Status</th></tr>")

        a.forEach( (details) => {

            let info=details.trim().split(/\s+/)

            res.write("<tr>"+<td>"+info[10]+</td>"+<td>"+info[0]+</td>"+<td>"+info[6]+</
td>"+</tr>")

            //    res.write(details+"<br>")

        })

        res.write("</table>")

        res.send()

    })

})

```

```

app.get("/rm", (req,res) =>{

    const osid=req.query.del_os;

    exec("docker rm -f "+osid, (err,stdout,stderr)==>{

        res.send("<pre>"+stdout+"</pre>")
    })
}

```

```
  })
```

```
})
```

```
app.get("/img", (req,res) =>{
```

```
  exec("docker images | tail -n +2", (err,stdout,stderr)=>{
```

```
    let a=stdout.split("\n")
```

CHAPTER-4

Result Analysis

After the deployment is completed, the system is tested to verify the sound interaction between the cloud computing platform and the role authority management system. The interaction service is carried out at the entrance of the user access system, which enables users to visually learn about the real-time status of the project and the requests of users can be processed accordingly. Details are as shown in Table 2. It can be seen that the cloud computing platform based role authority management system has satisfactory test results consistent with expected outputs as well as strong robustness. A part of the implementation interface of the cloud platform based role authority management system test is presented as shown in Figure 5. It can be seen that the cloud computing platform based role authority management system has complete functions and can effectively disclose member information. The cloud computing platform and role authority management system have good interaction, which confirms the feasibility of the design of this paper.

Advantages of Docker Container :-

The demand and the advancement of Linux containers can be seen in the last few years. Docker has become popular very quickly, because of the benefits provided by docker container. The main advantages of docker are speed, portability, scalability, rapid delivery, and density.

Speed:- Speed is one of the most exceedingly touted advantages of Containers. When the benefits of using docker are highlighted, it would be incredible not to mention about the speed of docker in the conversation (Chavis & Architect, 2015). The time required to build a container is very fast because they are really small. Development, testing, and deployment can be done faster as containers are small. Containers can be pushed for testing once they have been built and then from there, on to the production environment [12].

Portability :- Those applications that are built inside docker containers are extremely portable. These portable applications can easily be moved as a single element and the performance remains the same [12].

Scalability:- Docker has the ability that it can be deployed in several physical servers, data servers, and cloud platforms. It can also be run on every Linux machine. Containers can easily be moved from a cloud environment to local host and from there back to cloud again at a fast pace. Adjustments can easily be done; the scale can simply be adjusted by the user according to the need [5].

Density:- Docker uses the resources that are available more efficiently because it does not use a hypervisor. This is the reason that more containers can be run on a single host as compared to virtual machines. The performance of a Docker Containers is higher because of higher density and no overhead wastage of resources [5].

Disadvantages of Docker Container:-


Complete virtualization is not provided by a docker because it depends on the Linux kernel which is provided by the local host.

- • Currently, docker does not run on older machines. It only supports 64-bit local machines.
- • The complete virtualized environment must be provided by the docker container for Windows and Mac machines. Even though the boot2docker tool fills this gap, but still, it should be checked whether it makes obstructions to acceptance by users of these systems or the integration and performance with the host machine's operating system are adequate [4].
- • It is necessary that the possibility of security issues should be evaluated. Building off trusting binaries could be made easier by digitally signing docker images, for future support.
- • An important concern is to check if the teaching community or scientific researcher will significantly think of adopting docker.

- In this section, the performance of application virtualization and the performance of the docker container will be discussed, and the evaluation of other containerize technology will be compared and reviewed. Seo et al. (2014) summarize that there is no guest OS of docker in the cloud, so the storage and the wastage of CPU resources are less [8]. The images are not disturbed; boot time is faster and the time of generating the images is short. These are the benefits of docker cloud in comparison with VM Cloud. They used two similar servers with the same configuration in the cloud environment. One server was used for docker.


Project Design:-

```
ec2-user@ip-172-31-42-167:~$ cat /etc/apt/sources.list.d/option.list
<option>openjdk</option>
<option>golang</option>
<option>httpd</option>
<option>mongo</option>
<option>debian</option>
<option>ruby</option>
<option>elasticsearch</option>
client_loop: send disconnect: Connection reset by peer
bell@vikas MINGW64 ~/downloads (master)linux</option>
$ ssh -i KeyPairRheI051.pem ec2-user@43.205.120.237
Last login: Tue Aug 16 13:55:12 2022 from 49.36.177.218
[ec2-user@ip-172-31-42-167 ~]$
[ec2-user@ip-172-31-42-167 ~]$
[ec2-user@ip-172-31-42-167 ~]$
[ec2-user@ip-172-31-42-167 ~]$
[ec2-user@ip-172-31-42-167 ~]$ tyle="display:inline-flex"><!-------div3-
[ec2-user@ip-172-31-42-167 ~]$ >
[ec2-user@ip-172-31-42-167 ~]$ e1 d5"> To Download Image: </label>
<button onclick="downloadImg()" class="button d5 to1"> click here</button>
</div> <div id="dwnld_status"></div>
<hr width="100%" />
```




459,7 71%

```
root@ip-172-31-42-167:~$ sudo su - root
Last login: Tue Aug 16 13:55:13 UTC 2022 on pts/0
[root@ip-172-31-42-167 ~]#
```



7:31 PM 8/16/2022


```
root@ip-172-31-42-167 ~# ContainerManagement
[ec2-user@ip-172-31-42-167 ~]$ sudo su - root
Last login: Tue Aug 16 13:55:13 UTC 2022 on pts/0
[root@ip-172-31-42-167 ~]# cd ContainerManagement/
[root@ip-172-31-42-167 ContainerManagement]# ls
frontend.html  index.js  node_modules  package-lock.json  version1.html
[root@ip-172-31-42-167 ContainerManagement]#
```



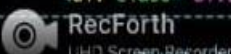
Type here to search

28°C Cloudy 7:31 PM 8/16/2022

```
root@ip-172-31-42-167 ~# ContainerManagement
<option>postgres</option>
<option>redis</option>
<option>alpine</option>
<option>node</option>
<option>mysql</option>
<option>traefic</option>
<option>busybox</option>
<option>python</option>
<option>nginx</option>
<option>openjdk</option>
<option>golang</option>
<option>httpd</option>
<option>mongo</option>
<option>debian</option>
<option>consul</option>
<option>ruby</option>
<option>amazonlinux</option>
<option> mariadb</option>
<option>elasticsearch</option>

</select>
</div>

<div class="div3 div" style="display:inline-flex"><!-------div3-
RecForth
UHD Screen Recorder
<label class="label d5"> To Download Image: </label>
'frontend.html" 666L, 22511C 459,7 69%
```



Type here to search

28°C Cloudy 7:31 PM 8/16/2022

```
test@ip-172-31-42-167:~/ContainerManagement
<div class="div4 div">
  <label class="label">Select an Image to Delete</label>
  <select id="imag" name="imag_name" class="select1 button" >
    <option>centos</option>
    <option>ubuntu</option>
    <option>python</option>
    <option>nginx</option>
    <option>openjdk</option>
    <option>golang</option>
    <option>httpd</option>
    <option>mongo</option>
    <option>debian</option>
    <option>consul</option>
    <option>ruby</option>
    <option>amazonlinux</option>
    <option>mariadb</option>
    <option>elasticsearch</option>
    <option>busybox</option>
    <option>postgres</option>
    <option>redis</option>
    <option>alpine</option>
    <option>node</option>
    <option>mysql</option>
    <option>traefic</option>
  </select>
</div>
RecForth
UHD Screen Recorder
503,7 75%
```

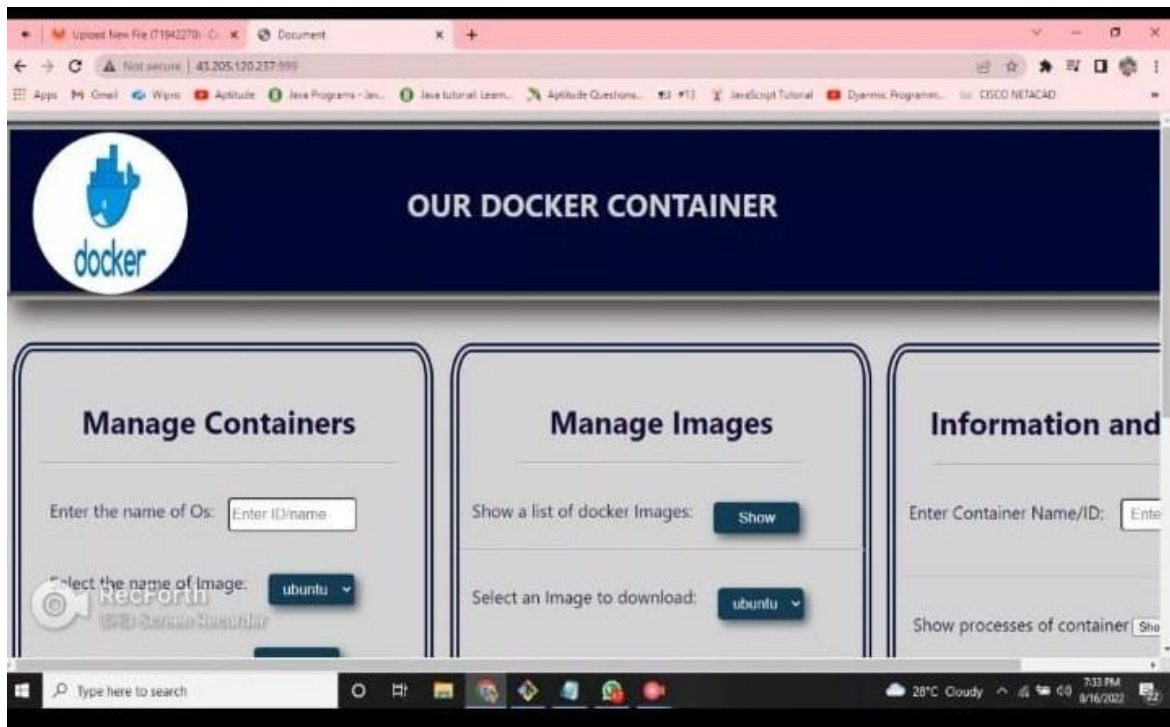
```
test@ip-172-31-42-167:~/ContainerManagement
<div class="inputElement 11 div7">
  <label class="label " > Enter Container Name/ID:</label>
  <input id="procss" name="con_id" class="ip ip3 b1" placeholder="Enter Container ID/name"
size="22px" />
</div><hr width="100%" />
<div id="show_stats" class="inputElement 12" >
  <div class="div8 c1">
    <label class="label 12">Show processes of container</label>
    <button id="stats" class="show" onclick="showStats()">Show </button>
  </div> <div id="stats_area"></div><hr width=" 100%" />
</div>
</div>
</section> <!-------end of bigcontainer----->
```

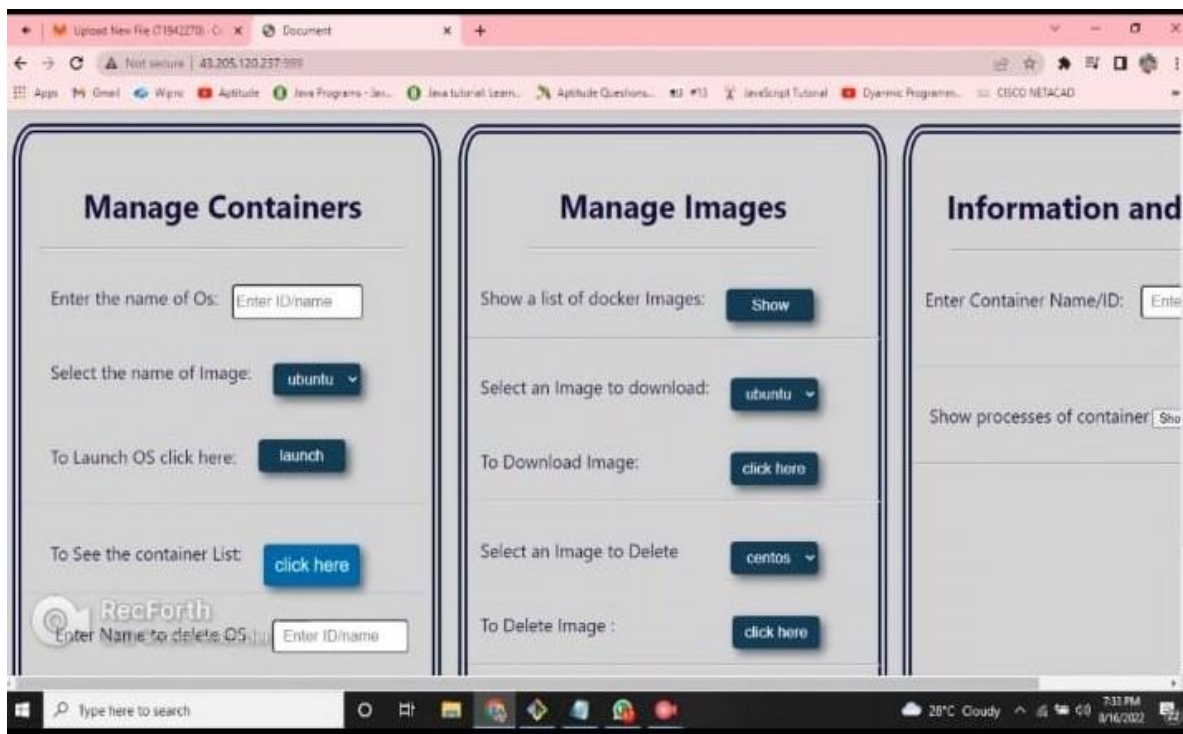
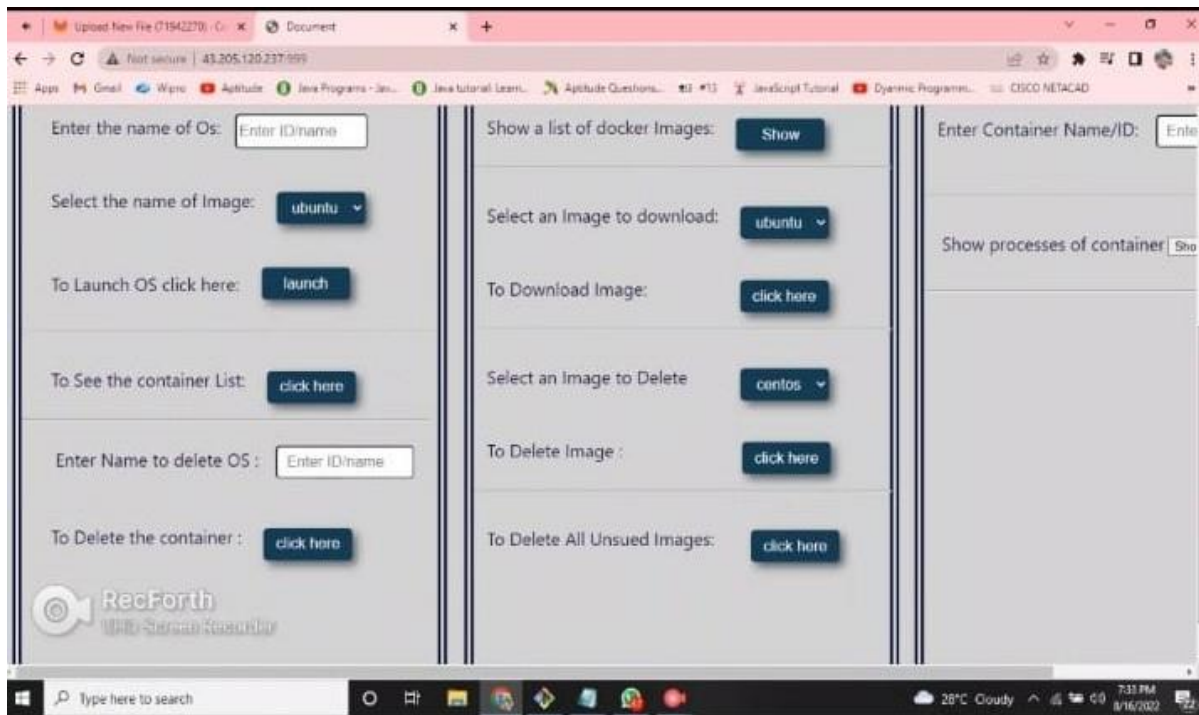
```
root@ip-172-31-42-167:~/ContainerManagement
[root@ip-172-31-42-167 ContainerManagement]# vim index.js
[root@ip-172-31-42-167 ContainerManagement]# client_loop: send disconnect: Connection reset by peer

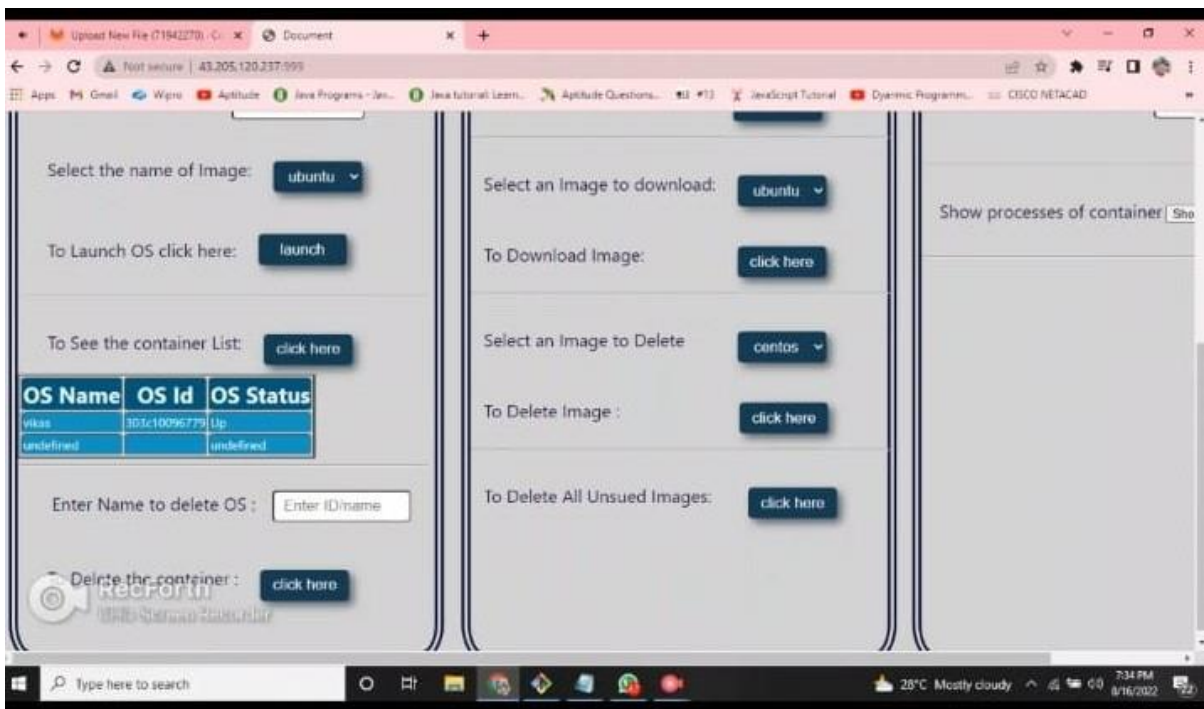
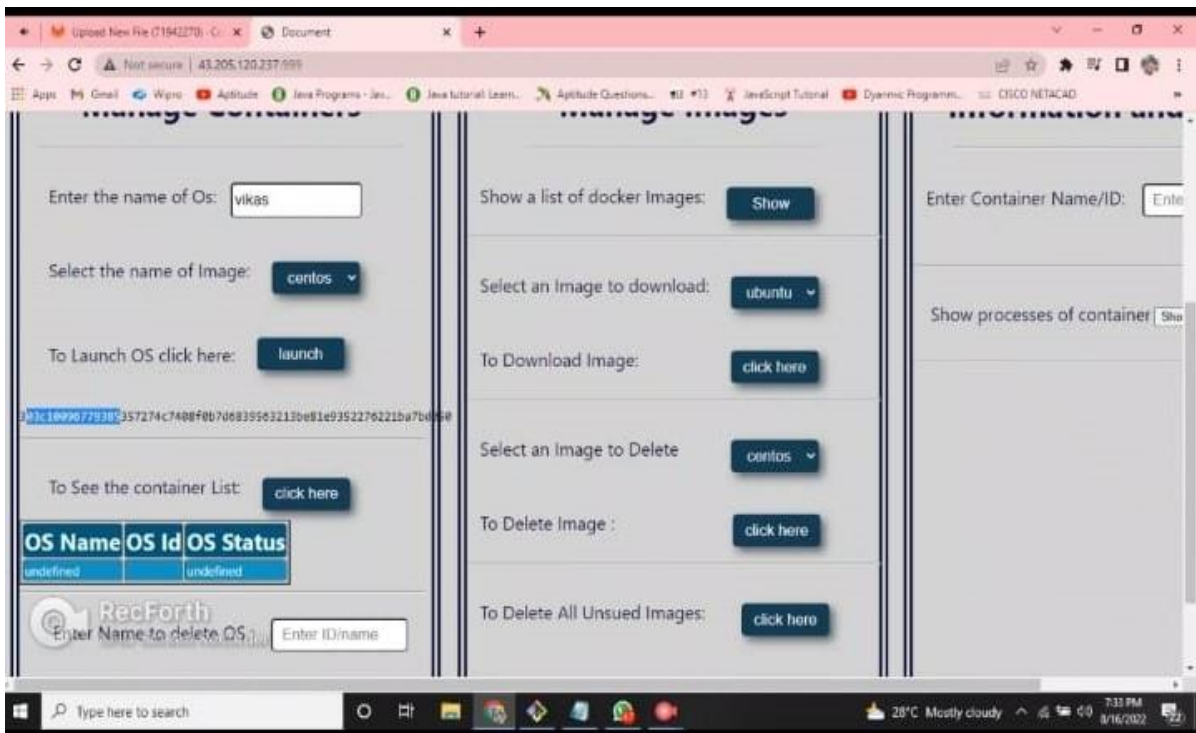
vikas@vikas MINGW64 ~/downloads (master)
$ ssh -i KeyPairRhelOS1.pem ec2-user@43.205.120.237
Last login: Tue Aug 16 13:04:31 2022 from 49.36.177.218
[ec2-user@ip-172-31-42-167 ~]$ sudo su - root
Last login: Tue Aug 16 13:04:35 UTC 2022 on pts/0
[root@ip-172-31-42-167 ~]# cd ContainerManagement/
[root@ip-172-31-42-167 ContainerManagement]# ls
Frontend.html index.js node_modules package.json package-lock.json version1.html
[root@ip-172-31-42-167 ContainerManagement]# vim frontend.htm
[root@ip-172-31-42-167 ContainerManagement]# node index.js
Server started..
^C
[root@ip-172-31-42-167 ContainerManagement]# client_loop: send disconnect: Connection reset by peer

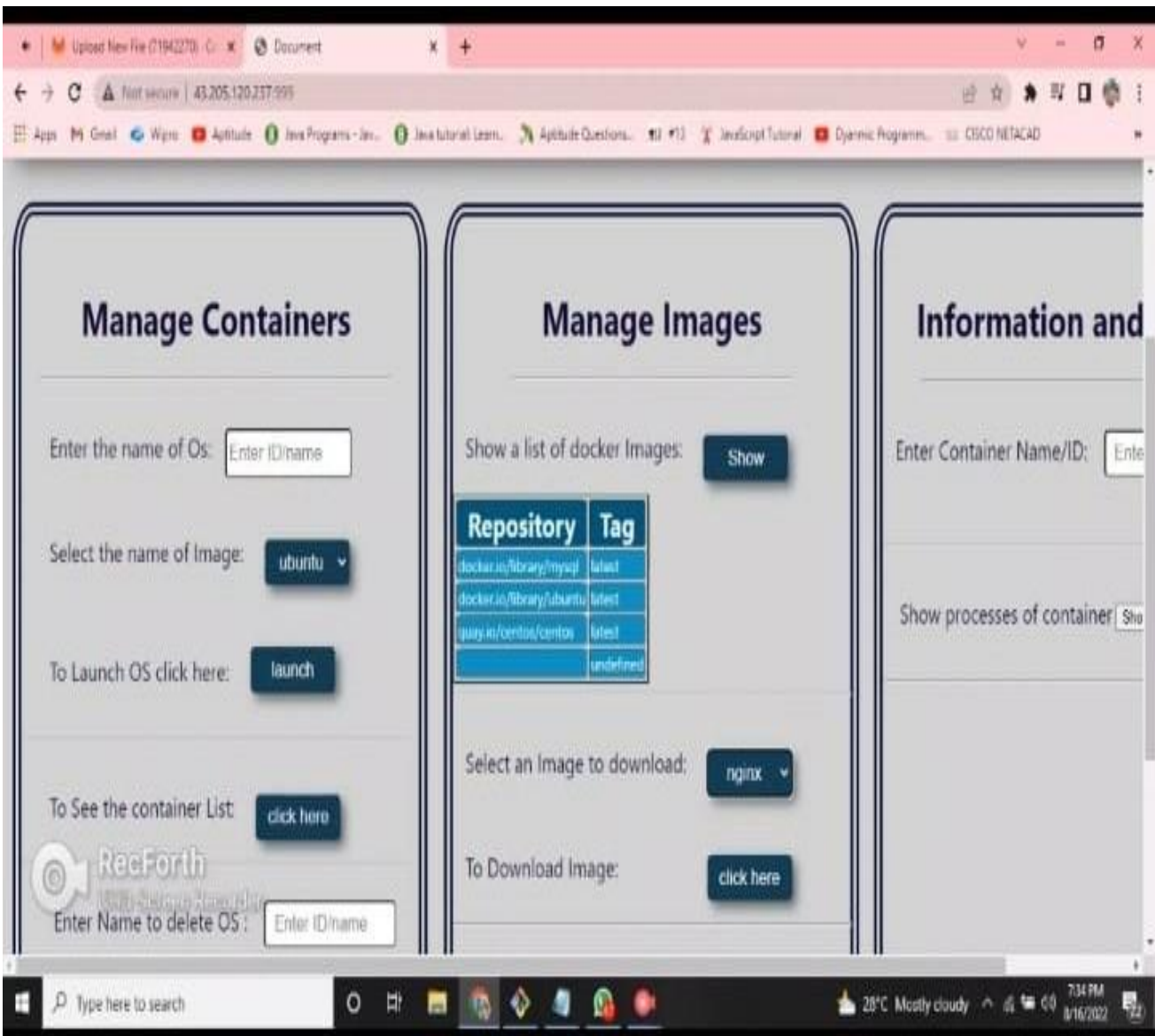
vikas@vikas MINGW64 ~/downloads (master)
$ ssh -i KeyPairRhelOS1.pem ec2-user@43.205.120.237
Last login: Tue Aug 16 13:43:20 2022 from 49.36.177.218
[ec2-user@ip-172-31-42-167 ~]$ sudo su - root
Last login: Tue Aug 16 13:43:22 UTC 2022 on pts/0
[root@ip-172-31-42-167 ~]# cd ContainerManagement/
[root@ip-172-31-42-167 ContainerManagement]# ls
Frontend.html index.js node_modules package.json package-lock.json version1.html
[root@ip-172-31-42-167 ContainerManagement]# vim frontend.htm
[root@ip-172-31-42-167 ContainerManagement]# vim index.js
[root@ip-172-31-42-167 ContainerManagement]# node index.js
Server started..
```

OUR INTERFACE WILL LOOK LIKE THIS:-









CHAPTER-5

Conclusion & References

Conclusion:-

It gives an overview of the research in this field, analyzes authority control models, Docker, and other related technologies, determines the role authority control model based on the basic framework of Docker, and designs the role authority management system on this basis. Finally, the cloud computing platform is deployed, its interaction with the role authority management system is completed, and the core authority function is realized. Through the test, the function of the cloud computing platform based role authority management system is fully verified, and the feasibility of the model design is confirmed. With the continuous research and development as well as optimization of the model, the basic framework can completely replace traditional role authority management modes and then promote the development of role authority management. Docker is a lightweight virtual technology with advantages of shorter deployment time and higher resource utilization rate than virtual machines. i.e deployment of the project on Docker enables the system to realize the purpose of multiple operations at one time, which greatly shortens development and testing time, improves work efficiency, and is highly practical. However, this technology also has some problems such as poor isolation performance and serious waste of storage resources, which affects the performance of the design architecture in this paper and indicates the direction for future research.

REFERENCES:-

1. Boettiger, C. (2015). An introduction to Docker for reproducible research. *ACM SIGOPS Operating Systems Review*, 49(1), 71-79.
2. Bui, T. (2015). Analysis of docker security. arXiv preprint arXiv:1501.02967.
3. Felter, W., Ferreira, A., Rajamony, R., & Rubio, J. (2014). An updated performance comparison of virtual machines and linux containers. *technology*, 28, 32.
4. Harji, A. S., Buhr, P. A., & Brecht, T. (2013). Our troubles with Linux Kernel upgrades and why you should care. *ACM SIGOPS Operating Systems Review*, 47(2), 66-72.
5. Joy, A. M. (2015). Performance comparison between Linux containers and virtual machines. Paper presented at the Computer Engineering and Applications (ICACEA), 2015 International Conference on Advances in.
6. Russell, B. (2015). Passive Benchmarking with docker LXC, KVM & OpenStack.
7. Scheepers, M. J. (2014). Virtualization and containerization of application infrastructure: A comparison.
8. Seo, K.-T., Hwang, H.-S., Moon, I.-Y., Kwon, O.-Y., & Kim, B.-J. (2014). Performance Comparison Analysis of Linux Container and Virtual Machine for Building Cloud.
9. Turnbull, J. (2014). *The Docker Book: Containerization is the new virtualization*. [10] Van der Aalst, W., Weijters, T., & Maruster, L. (2004). Workflow mining: Discovering process models from event logs. *Knowledge and Data Engineering, IEEE Transactions on*, 16(9), 1128-1142.
11. Varghese, B., Subba, L. T., Thai, L., & Barker, A. (2016). Container-Based Cloud Virtual Machine Benchmarking. arXiv preprint arXiv:1601.03872.
12. Vase, T. (2015). Advantages of Docker.
13. Waldspurger, C. A. (2002). Memory resource management in VMware ESX server. *ACM SIGOPS Operating Systems Review*, 36(SI), 181-194.