Virtualization Using Docker Container For Reproducible Desktop Environment

Dr.Manikandan.J[1]
ECE
Sri Sairam Engineering College
Chennai, India
manikandan.ece@sairam.edu.in

* Kirubakaran. K[2]
ECE
Sri Sairam Engineering College
Chennai, India
sec20ec098@sairam.edu.in

Yuvanesh. N[3]
ECE
Sri Sairam Engineering College
Chennai, India
sec20ec099@sairamtap.edu.in

Dhino Bilgates J[4]
ECE
Sri Sairam Engineering College
Chennai, India
sec20ec054@sairamtap.edu.in

Abstract— **To overcome the old and traditional method of sharing data by sending a file through a hard drive or even sending a file through the internet needs a 3rd party application or a medium(hardware), to achieve what we want. In order to eliminate the intermediate route that we use to send and receive data (Hardware and software for file sharing), we are steaming a desktop to the client's browser, so they can access a common OS among them, with the help of Docker Container. The desktop that is being steamed to our browser (client side) Is stored in a private container, that can be accessed only through the credentials that we hold(host name, password), The hardware components can be configured to our wish like increasing the RAM size, Storage size and Container Lifetime. By this we can commercialize it as product which holds all the peripheral software's and can be deployed whereever we want at instance. Simply put, our project is inspired from Google suite, but instead of deploying webapp's, we are trying to stream an entire Operating System streaming via browser. Which will have enough storage needed as per the client's request.**

Index Terms—Docker container, RAM, Webapp, Google suite, Hardware, Software

I.INTRODUCTION

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 share the kernel with the operating system which eliminates the overheads and it also provides 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 one of the container technologies which is currently being used in many production environments to package their applications in isolated environments. This newly evolved containers are none other than Docker which has changed the perspective of deploying the applications in the 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 application images 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.

II. RELATED WORKS

The Operating System level virtualization supports the failover and load balancing mechanisms in the docker swarm area and the stack plays a role in distributing web server services from node manager to node worker as a backup webserver service [1]. A number of digital signage clients and Internet of things devices are virtualized and managed by a container-based middleware. Each container-based middleware is responsible to manage and process data for a cluster of digital signage clients and corresponding Internet of things devices to reduce load to server and improve the service performance [5].

III. BASIC THEORY

A. *Docker daemon*

Docker Daemon listens for Docker API requests from Client and manages Docker images, containers ,networks, and volumes. When a request from Client has been received to create a container, Docker Daemon pulls the specified image from the Docker registry or local image registry and then creates the container from the image. A Daemon can also communicate with other Daemons to manage Docker services.

B. *Virtualization*

Virtualization is the process of migrating physical environment into virtual environments. This virtual environment can include anything from virtual operating systems to virtual servers. Many companies have already adopted virtualization because they reduce the overheads like maintaining the hardware which is included in large rooms or data centers occupied with large number of devices and cables. Although Virtualization did not completely solve the problem of using bulky hardware but it got succeeded in reducing the usage of unnecessary bulky and costly hardware which was a burden to most of the organizations

C. *Hypervisor*

A hypervisor, also known as a virtual machine monitor or VMM, is software that creates and runs virtual machines (VMs). A hypervisor allows one host computer to support multiple guest VMs by virtually sharing its resources, such as memory and processing. The main difference between Type 1 vs. Type 2 hypervisors is that Type 1 runs on bare metal and Type 2 runs on top of an operating system

IV.SOLUTION METHODOLOGY

To overcome the above the problem we mentioned, our proposed methodology to avoid the use of external hardware and software is by steaming contents into the clients browser, with the help of docker containers simply say, two persons cannot work on a single computer at the same time, it can be accessing the keyboard to viewing the screen it is merely impossible for 2 or 3 persons to work on a computer at once. But using docker, we can install an OS in a container and virtually steam it to our browser as output. Like Google slides and other Google Suite, which is versatile, cross platformed and very efficient to use. It can steam only web application, while we are trying to steam an entire desktop (OS).

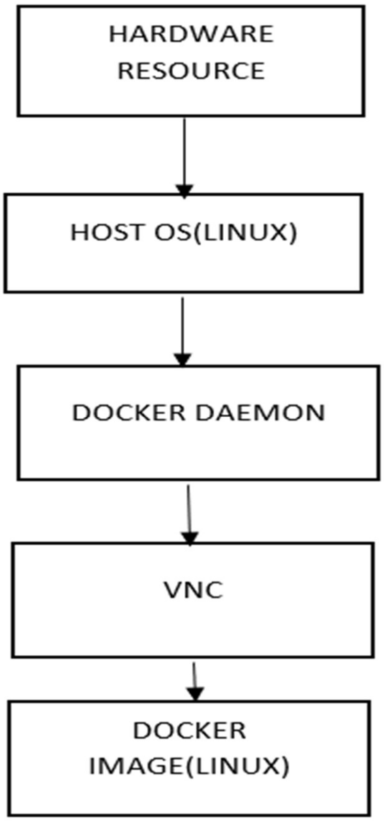
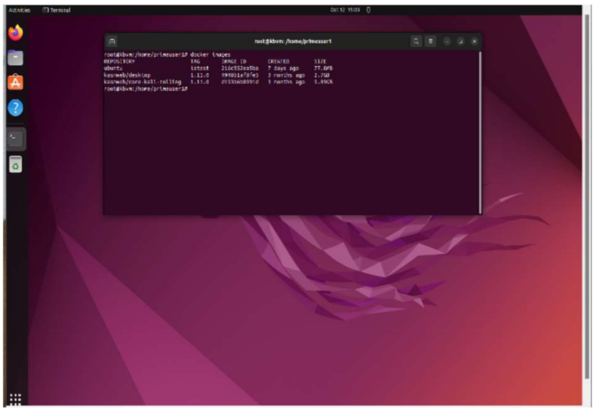


Fig. 1. Flowchart

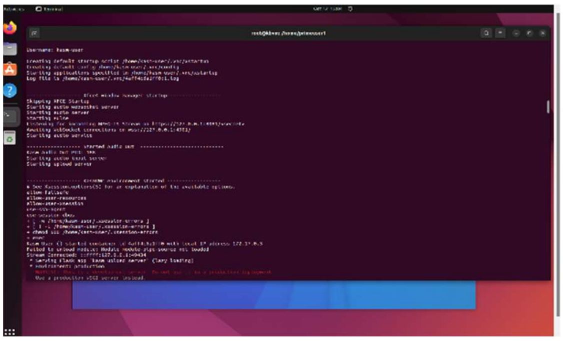
V. RESULTS AND DISCUSSION

The Output shows the virtualization of OS through the browser using VNC and Docker Container.



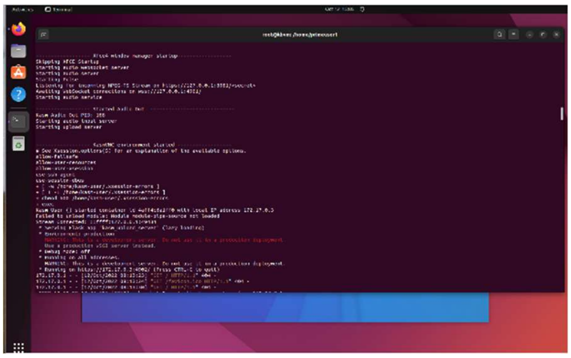
```
root@kbvm:/home/primeuser1# docker images
REPOSITORY                  TAG       IMAGE ID       CREATED        SIZE
ubuntu                      latest    216c552ea5ba   7 days ago     77.8MB
kasmweb/desktop             1.11.0    494801ef8fe3   3 months ago   2.7GB
kasmweb/core-kali-rolling   1.11.0    d193b6d8091d   3 months ago   1.89GB
root@kbvm:/home/primeuser1#
```

Fig. 1. Runs the image in a docker using the command



```
Username: kasm-user

Creating default startup script /home/kasm-user/.vnc/xstartup
Creating default config /home/kasm-user/.vnc/config
Starting applications specified in /home/kasm-user/.vnc/xstartup
Log file is /home/kasm-user/.vnc/4aff4cda2ff0:1.log

---------------- Xfce4 window manager startup ----------------
Skipping XFCE Startup
Starting audio websocket server
Starting audio server
Starting pulse
Listening for incoming MPEG-TS Stream on https://127.0.0.1:4901/<secret>
Awaiting WebSocket connections on wss://127.0.0.1:4901/
Starting audio service

---------------- Started Audio Out ----------------
Kasm Audio Out PID: 166
Starting audio input server
Starting upload server

---------------- KasmVNC environment started ----------------
# See Xsession.options(5) for an explanation of the available options.
allow-failsafe
allow-user-resources
allow-user-xsession
use-ssh-agent
use-session-dbus
+ [ -e /home/kasm-user/.xsession-errors ]
+ [ ! -L /home/kasm-user/.xsession-errors ]
+ chmod 600 /home/kasm-user/.xsession-errors
+ exec
Kasm User () started container id 4aff4cda2ff0 with local IP address 172.17.0.3
Failed to unload module: Module module-pipe-source not loaded
Stream Connected: ::ffff:127.0.0.1:40434
 * Serving Flask app 'kasm_upload_server' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
```

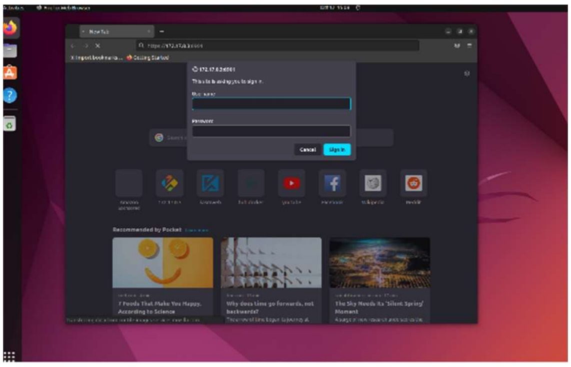
Fig. 2a



```
---------------- Xfce4 window manager startup ----------------
Skipping XFCE Startup
Starting audio websocket server
Starting audio server
Starting pulse
Listening for incoming MPEG-TS Stream on https://127.0.0.1:4901/<secret>
Awaiting WebSocket connections on wss://127.0.0.1:4901/
Starting audio service

---------------- Started Audio Out ----------------
Kasm Audio Out PID: 166
Starting audio input server
Starting upload server

---------------- KasmVNC environment started ----------------
# See Xsession.options(5) for an explanation of the available options.
allow-failsafe
allow-user-resources
allow-user-xsession
use-ssh-agent
use-session-dbus
+ [ -e /home/kasm-user/.xsession-errors ]
+ [ ! -L /home/kasm-user/.xsession-errors ]
+ chmod 600 /home/kasm-user/.xsession-errors
+ exec
Kasm User () started container id 4aff4cda2ff0 with local IP address 172.17.0.3
Failed to unload module: Module module-pipe-source not loaded
Stream Connected: ::ffff:127.0.0.1:40434
 * Serving Flask app 'kasm_upload_server' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on all addresses.
   WARNING: This is a development server. Do not use it in a production deployment.
 * Running on https://172.17.0.3:4902/ (Press CTRL+C to quit)
172.17.0.1 - - [12/Oct/2022 08:15:12] "GET / HTTP/1.1" 404 -
172.17.0.1 - - [12/Oct/2022 08:15:14] "GET /favicon.ico HTTP/1.1" 404 -
172.17.0.1 - - [12/Oct/2022 08:15:14] "GET / HTTP/1.1" 404 -
```

Fig. 2b

Fig. 2a and 2b shows the IP for the login



172.17.0.3:6901

This site is asking you to sign in.

Username

Password

Cancel Sign in

Fig. 3. Username and password for login

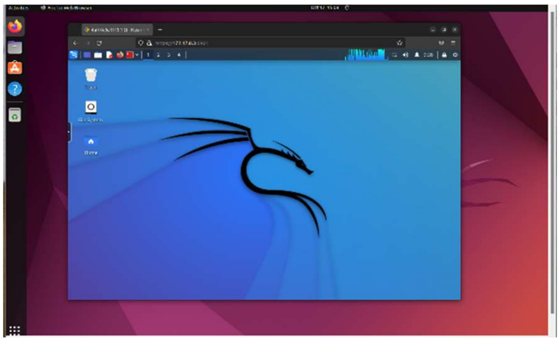


Fig. 3. Streaming the OS in the browser

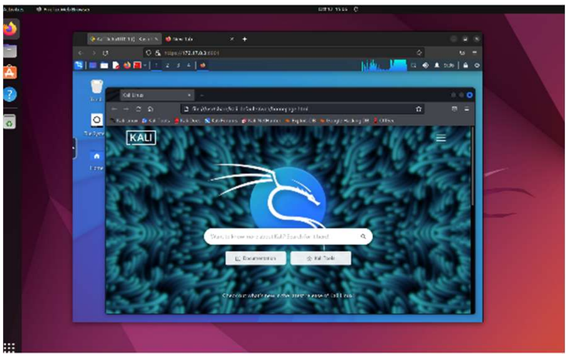


Fig. 4. Opening browser in streaming OS

VI.CONCLUSION

We have explained in the paper a procedural way to use a Desktop which is common for all users and it can be in both in CLI (Command Line Interface) and in GUI (Graphical User Interface) which is basically focused for users who haven't exposed to CLI mode of computing. As steaming of GUI interfaces through remote desktop will be useful in cloud-computing and cloud-gaming, our very first integration in based on cloud-computing.

For future works, consolidation of the existing docker engine will be replaced, as integration of pod- man which is better than docker in terms of handling and easy do deploy containers, so probably deployment will be a bit of a hassle, if once rectified it can be delivered as and finished product. As our theme or one of the principle for this is to fetch maximum computing power using less hardware resource.

References

1] H. Nurwarsito and V. B. Sejahtera, “Implementation of Dynamic Web Server Based on Operating System-Level Virtualization using Docker Stack,” 2020 12th International Conference on Information Technology and Electrical Engineering (ICITEE), 2020.

[2] N. Zhao et al., “Large-Scale Analysis of Docker Images and Performance Implications for Container Storage Systems,” in IEEE Transactions on Parallel and Distributed Systems, vol. 32, no. 4, pp. 918-930, 1 April 2021.

[3] S. Wang, L. Zhu and M. Cheng, “Docker-based Web Server Instructional System,” 2019 IEEE/ACIS 18th International Conference on Computer and Information Science (ICIS), 2019.

[4] W. M. C. J. T. Kithulwatta, K. P. N. Jayasena, B. T. G. S. Kumara and R. M. K. T. Rathnayaka, “Performance Evaluation of Docker-based Apache and Nginx Web Server,” 2022 3rd International Conference for Emerging Technology (INCET), 2022.

[5] Youngki Park, Hyunsik Yang, Thanh Dinh and Younghan Kim “Design and implementation of a container-based virtual client architecture for interactive digitalsignage systems”.

[6] K. H. Anun and M. S. Almhanna, "Web Server Load Balancing Based on Number of Client Connections on Docker Swarm," 2021 2nd Information Technology to Enhance e-learning and Other Application (IT-ELA), 2021.

[7] H. Nurwarsito and M. Fadhil, "Implementation of Dynamic Web Server Cluster Based on Operating System-Level Virtualization using Docker Swarm," 2020 10th Electrical Power, Electronics, Communications, Controls and Informatics Seminar (EECCIS), 2020

[8] M. Iurian, A. Ivanciu and V. Dobrota, "Couchbase Server in Microsoft Azure Cloud: A Docker Container Approach," 2020 International Symposium on Electronics and Telecommunications (ISETC), 2020.

[9] C. Cerin, T. Menouer, W. Saad and W. B. Abdallah, "A New Docker Swarm Scheduling Strategy," 2017 IEEE 7th International Symposium on Cloud and Service Computing (SC2), 2017.

[10] R. Rahmansyah, V. Suryani, F. Arif Yulianto and N. Hidayah Ab Rahman, "Reducing Docker Daemon Attack Surface Using Rootless Mode," 2021 International Conference on Software Engineering & Computer Systems and 4th International Conference on Computational Science and Information Management (ICSECS-ICOCSIM), 2021.