INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

ISSN: 2454-132X
Impact Factor: 6.078
(Volume 7, Issue 3 - V7I3-2052)
Available online at: https://www.ijariit.com

A survey on docker container and its use cases

Abha Anand B.
abhaanandb.is17@rvce.edu.in
RV College of Engineering, Bengaluru, Karnataka

Darunya B. C.
darunyabc.is17@rvce.edu.in
RV College of Engineering, Bengaluru, Karnataka

Anisha B. S.
anishabs@rvce.edu.in
RV College of Engineering, Bengaluru, Karnataka

S. G. Raghavendra Prasad
raghavendrap@rvce.edu.in
RV College of Engineering, Bengaluru, Karnataka

ABSTRACT

Docker is an open platform for rapidly building, delivering, and executing a variety of applications. It allows for faster software delivery by separating applications from infrastructure. Docker techniques encourage rapid shipping, testing, and deployment of code anywhere, reducing the time between code development and production deployment. It assists both developers and system administrators by letting developers write code without having to worry about the system on which it will eventually run. It also can minimize the number of systems and provide operations workers more flexibility. The principles of Docker containers are discussed in this paper and recent research on the subject. The document also contains significant use cases, as well as the pros and challenges of using it.

Keywords: *Docker, Container, Virtual Machine, Docker Image, Docker Use Case, Docker Techniques, Open Platform*

1. INTRODUCTION

Before Docker containers, huge corporations like JPMorgan Chase, ThoughtWorks, Walmart, Target, and that relied on technology employed servers and added many of them, resulting in over-allocation, to manage the growing number of requests from customers. The disadvantage of over-allocating servers was that it was incredibly expensive, and if the servers did not scale effectively, the business would perish After a few years have passed, VMware developed the concept of virtualization, which allowed multiple operating systems to run on the same host, essentially allowing any application to run in isolation on the same server and infrastructure, giving the impression of running an entirely separate computer on the same computer, which was a game-changer for many industries. Virtualization came with a lot of benefits, but it was also highly expensive. To begin with, there are numerous kernels for each guest operating system that will run on the infrastructure; also, resources must be allotted to each guest operating system that is introduced to the infrastructure. Furthermore, virtualization is costly because, despite the absence of actual hardware, virtual hardware consumes resources, including the guest operating system space and RAM allocation required for this operating system. Moving forward with current technology known as Containerization, which is defined as a sort of operating system (OS) virtualization in which applications are executed in separated user areas while sharing the same operating system. Container-based virtualization, such as Docker, can be used instead of virtual machines for faster functionality because starting and shutting down a container is significantly faster. **[13]As**as a containerization platform, Dockers encapsulate all of an application's dependencies in a Docker Container, ensuring that the app runs smoothly in any environment. Docker allows for much denser server consolidation than VMs and to meet the first goal each flexibility to share additional usable RAM amongst instances. **[9]** It facilitates Rapid Deployment; it is fast, lightweight because it does not boot a separate OS per VM, making it quick to start/stop; it requires less disc space because common layers are shared across images; incremental deployments of new app versions are smaller and thus faster than VMs, and it shares a kernel across containers and thus uses less memory. Docker files can be used to configure any computer instantly, making it portable. Dockers without a hypervisor have a significant benefit because they don't require a separate kernel. Theycentralized uses the same resources as the host OS. They take advantage of namespaces and control groups to more efficiently utilize these resources.

2. LITERATURE SURVEY

The paper titled "Workload-aware Resource Management for Energy Efficient Heterogeneous Docker Containers" [5] The proposed technology demonstrates how energy usage can be lowered effectively. Workload-aware Energy Efficient Container (WEEC) brokering system is proposed to align with energy spent by running container applications across various cloud servers. Essentially, the system divides input requests into several server racks based on the containers' resource use patterns. The WEEC brokering system is divided into four sub-

components1) The Table Manager for Power Consumption Per Application (PPA) is in charge of assessing the initial energy efficiency consumed by the heterogeneous server in containers.2) EWMA (Exponential Weighted Moving Average) forecasts workload for future input requests. 3) DRS (Dynamic Right-Sizing) aids in the management of active servers.4) In Docker containers, the Request Allocation Manager oversees the assignment of input requests to each specified server. The work establishes a benchmark by assessing the power and performance of multiple heterogeneous Docker containers with benchmark programs and Yocto-Watt power monitoring devices.

The paper titled "Model-Driven Management of Docker Containers" [3] focuses on docker container administration, particularly where users encounter low-level system concerns, and illustrates how modeling Docker containers aids in the deployment and administration of Docker containers in a long-term manner. Although the Docker system offers more benefits than cloud computing platforms such as Azure and Amazon, the Docker containers suffer from synchronization issues between deployed and intended containers. focuses on docker container administration, particularly where users encounter low-level system concerns, and illustrates how modeling Docker containers aids in the deployment and administration of Docker containers in a long-term manner. Although the Docker system offers more benefits than cloud computing platforms such as Azure and Amazon, the Docker containers suffer from synchronization issues between deployed and intended containers. The connector establishes a link between the Docker model and the Execution environment, providing tools to efficiently run the Docker model by creating particular artifacts in response to changes in the Execution environment. This paper introduced a model-driven method to verification and synchronization, and future research will focus on atomic modifications in managed container architecture. **II.**

The paper titled "Evaluation of Docker as Edge Computing Platform" [2] explains how to deal with issues including high latency, network bottlenecks, and network congestion. Edge computing will be able to lower application response time for a better user experience by transitioning from a centralized to a decentralized model. Docker, a container-based technology platform with more advantages than VM-based Edge computing, is used to allow Edge computing. This article focuses on the two most important requirements for EC: 1) Deployment and Termination, which defines the platform that makes it simple to manage to install and configure services for low-end devices. 2) Resource and Service Management, which enables consumers to use services even when resources are depleted. . 3) Fault Tolerance, which is based on the user's high availability and reliability. 4) Caching helps users to enjoy higher speed by caching Docker images at the edge. One such example is the use of the Docker idea on Hadoop Streaming, which reduces setup time and configuration issues. Overall, several places may be improved, but it still offers flexibility and outstanding performance.

The paper titled "OpenStack and Docker: building a high-performance IaaS platform for interactive social media applications" [1] discusses the Nova-Docker plugin, which allows for the quick and effective provisioning of computer resources that can be used as a Hypervisor to manage the expansion of application users. This is made possible via OpenStack IaaS, which allows cloud computing data centers to be controlled. Three key roles are present in the OpenStack standard architecture: Nova is in charge of computing, whereas Cinder is in charge of storage. Neutron is in charge of all networking resources across many data centers. NUBOMEDIA is another approach that uses the cloud to enable (PaaS) interactive social media. Kurento Media Server (KMS), a WebRTC media server that offers interactive communications, is one of the main technologies used. Using Docker containers, OpenBaton maintains the lifecycle of media server capabilities. OpenShift Origin is activated to host apps that use media server capabilities. Docker containers are preferred by developers and administrators over kernel-based virtual machines due to their fast boot time, direct access to containers, and the fact that they can be run on any hardware that supports Linux. Docker containers are small and light, which reduces the amount of bandwidth required for deployment [8].

The paper titled "Docker container-based analytics at IoT edge" [4] deals with internet-connected devices that use sensors and generate a large amount of data. Compute and process the data becomes a difficult effort. As a remedy, this paper focuses on how to use Docker, a lightweight virtualization technology, to facilitate application deployment at the IoT end. The research primarily displays the use case for a video surveillance feed developed in the United Kingdom touses analyze and detect impending incidents using Deep learning. The above use case's implementation mostly entails components such as the Raspberry Pi 3, which serves as a gateway and collects the video feed from the CCTV. The stream is analyzed using many deep learning frameworks, and the MQTT client is used to alert cloud-based services about the presence of threads in the frames using Docker Swarm. This method also serves as a standard for achieving efficient high-resolution frame processing. When contrasted to bare metal deployment, this study reveals that deep learning may be combined with Docker containers on single-based machines, resulting in low CPU processing overhead.

3. KEY DOCKER USE CASES [10]

A. Configuration Made Simple - Without the overhead of a virtual machine, Docker lets every platform with its configuration to run on top of any infrastructure. Docker allows you to embed configuration files in code, pass over env variables for multiple environments, and deploy it. As a result, a single docker image can be utilized in several contexts. This separates the application's infrastructure requirements from the application's requirements.

B. Management of the CodePipeline- The code pipeline management is greatly aided by the simplification of configuration. There are many different contexts that code must pass through on its way from the developer's machine to production. Along the route, each of these may have small variations. The simplicity of configuration considerably aids code pipeline management. On its trip from the developer's machine to production, code must transit through several distinct environments. Each of these may vary somewhat along the way.

C. Docker for Development Productivity - We primarily need to achieve two primary goals in the development environment. We want the development environment to be as near to production as feasible, and we also want it to be quick enough for interactive use. Each service must run on its virtual machine (VM) to meet the first goal, just like the production application does. Because of Docker's low overhead, a few dozen services can easily operate within separate containers in a development environment with limited RAM. The second requirement is to have an active development environment for interactive use; Docker makes this possible by making the application code accessible to the

container from the host OS via shared volumes. This has advantages such as allowing the developer to alter the source code from his preferred platform while also keeping track of it.

D. ***Isolation of App***s - ought to Server consolidation for cost reduction or a plan to break a monolithic application into independent components are examples of application isolation. Run two REST API servers, one for each Apache installation. They all, however, use a slightly different version of Apache and have distinct system requirements. Running these API servers in separate containers is a simple method to get around this problem.

E. ***Consolidating Servers*** - Docker's application isolation capabilities allow you to consolidate many servers to save money by avoiding the memory footprint of various OSes and sharing unused memory between instances. In comparison to VMs, Docker allows for significantly more server consolidation.

F. ***Debugging Capabilities*** **-** Docker offers several tools that are adept at working with the concept of containers. To name a few of its features, one is the ability to checkpoint containers and their versions, as well as the ability to distinguish between two containers, allowing for quick application fixes.

G. ***Multi-tenancy***- Docker can also be used in multi-tenant systems to prevent major rewrites. For instance, creating multi-tenancy for an IoT application quickly and easily. Multi-tenant codebases are significantly more sophisticated, strict, and difficult to manage. Re-architecting an application takes time and money.

H. ***Rapid Deployment*** - In milliseconds, Docker containers can be constructed and launched. Containers achieve this by not booting up an operating system and instead of performing the application process. Furthermore, the immutable nature of Docker images assures you that things will continue to work as they have in the past.

4. DOCKER USAGE

A. WHEN TO USE?

Docker cannot be the best solution always; few cases are as described below [10] -

- ***Teams of software developers*** **-** When developers are working with diverse settings, Docker makes it easy to establish local development environments that nearly mirror the production environment without having to ssh into a remote box.
- ***App isolation*** - If it is necessary to run many applications on one server, placing the components of each application in distinct containers will help with dependency management.
- ***Getting acquainted with new technologies*** - Docker offers a disposable and isolated environment that allows you to get started with a new tool without having to spend a lot of effort on setup and configuration. Several projects maintain docker images of previously installed and configured programs.

5. CASE STUDY

In this case study, we'll explore how to use the TensorFlow Object-detection API in a Docker container to handle real-time (webcam) and video post-processing. With the help of OpenCV with python3 multiprocessing and multithreading libraries.

Motivation - The first reason for this project was to just investigate the real-time object recognition difficulty, which led to the investigation of the Python multiprocessing module to boost FPS. To take things a step further, the project will be integrated into a Docker container to improve portability. The biggest challenge here was dealing with video streams entering and exiting the container.

Data Science with Docker: TensorFlow's classic ssd mobilenet v2 coco model can be used for speed performance. Simply copy the model and the corresponding label map locally to keep the option of utilizing a personal model later. In today's world where new AI or ML algorithms are released every week and installing all that can cause OS crashes. Docker containers can be used to avoid this. The process of creating an image is time-consuming and takes several minutes. Then it's only a matter of putting it to use.

Object Detection in Real-Time: First, apply object detection to the webcam stream. Getting the webcam stream into the docker container and recovering the output stream such that the X11 server could display it was the challenge.

Stream video into a Docker container:

- Devices are found in the /dev/ dir in Linux and can be managed in the same way that files are. In most cases, the "0" device is your laptop webcam. When launching the docker image, utilize the device argument to transmit its stream into the container.
- The best way to launch a Docker container on Windows is to utilize Virtual Box.

Recover the video stream from the container.

- To render to the correct display, first expose your xhost so that the container may read and write using the X11 UNIX socket. Begin by granting docker access to the X server host's rights.
- After that, reset the access controls to their default values and generate the XSOCK and XAUTH environment variables. The first is an X11 Unix socket, while the second is a properly permission X authentication file.
- Finally, make sure the docker run command is up to date. Then, using the DISPLAY environment variable to forward the DISPLAY environment variable, mount a drive for the X11 Unix socket, and use the XAUTHORITY environment variable to link to the X authentication file, use the DISPLAY environment variable to forward the DISPLAY environment variable, mount a drive for the X11 Unix socket, and use the XAUTHORITY environment variable to link to the X authentication file.

Video Processing- To execute the object-detection API in real-time with a webcam, use the threading and multiprocessing python packages. A thread is used to read the camera stream. A line of frames is arranged and processed by a group of individuals (in which TensorFlow object-detection is running).

- Because all video frames must be read before workers can apply object detection to the first ones in the input queue, threading is not allowed for video processing. When the input queue is full, frames that are read are lost. Perhaps many workers and long queues will solve the problem (with a prohibitive computational cost).
- If the input queue is not yet full, the next frame from the video stream is read and added to it. If a frame is not received from the input queue, nothing is done.

This case study demonstrates how to use Docker to build a real-time object identification project using TensorFlow. Docker is the most secure approach to test new data science tools and package the solutions we provide to customers.

III.

IV.

6. REFERENCES

[1] Alin Calinciuc, Cristian Constantin Spoiala, Corneliu Octavian Turcu, Constantin Filote, "OpenStack and

Docker: building a high-performance IaaS platform for interactive social media applications", May 19-21, 2016.

[2] Bukhary Ikhwan Ismail, Ehsan Mostajeran Goortani, Mohd Bazli Ab Karim, Wong Ming Tat, Sharipah Setapa, Jing, Yuan Luke, Ong Hong Hoe, "Evaluation of Docker as Edge Computing Platform"., 2015 Advanced Computing Lab

[3] Fawaz Paraiso, Stephanie Challita, Yahya Al-Dhuraibi, Philippe Merle, "Model-DrivenManagement of Docker Containers"., University of Lille & Inria Lille - Nord Europe 2016.

[4] Pankaj Mendki, "Docker container-based analytics at IoT edge"., Senior Principal Engineer, Member of R&D 2018.

[5] Dong-Ki Kang, Gyu-Beom Choi, Seong-Hwan Kim, II-Sun Hwang, and Chan-Hyun Youn, "Workload-aware Resource Management for Energy Efficient Heterogeneous Docker Containers"., School of Electrical Engineering.

[6] Containers vs. VMs: What's the difference? http://searchservervirtualization.techtarget.com/answer/ Containers-vs-VMs-Whats-the-difference.

[7] Understanding the architecture https://docs.docker.com/engine/understanding-docker/.

[8] M. Raho, A. Spyridakis, M. Paolino, D. Raho, "KVM, Xen, and Docker: a performance analysis for ARM-based NFV and Cloud computing," IEEE 3rd Workshop on Advances in Information, Electronic and Electrical Engineering (AIEEE), pp. 1–8, November 2015.

[9] R. R. Yadav, E. T. G. Sousa, and G. R. A. Callou, Docker Containers Versus Virtual Machine-Based Virtualization: Proceedings of IEMIS 2018.

[10] Deploy Docker Open Source, or Enterprise for High Performing Systems, https://www.flux7.com/tech/ container-technology/docker/

[11] Chao Zheng and Douglas, Integrating Containers into Workflows: A Case Study Using Makeflow, Work Queue, and Docker.

[12] Control Desk existing solution: A containerization case study with Docker https://developer.ibm.com/ technologies/ containers/articles/containerization-docker-case-study.

[13] Preeth E N, F. J. P. Mulerickal, B. Paul and Y. Sastri, "Evaluation of Docker containers based on hardware utilization," 2015 International Conference on Control Communication & Computing India (ICCC), 2015, pp. 697-700, DOI: 10.1109/ICCC.2015.7432984.

