Scientific Journal of Impact Factor (SJIF): 4.14

e-ISSN (O): 2348-4470
p-ISSN (P): 2348-6406

International Journal of Advance Engineering and Research Development

Volume 3, Issue 2, February -2016

RESOURCE MONITORING OF DOCKER CONTAINERS

Akshay Kumar Soam[1], Anish Kumar Jha[2], Amit Kumar[3], Vivek Kumar Thakur[4], Praveen Hore[5]

[1,2,3,4,5]Department of Computer Engineering, Army Institute of Technology, Pune, India

Abstract*—Docker is a new technology developed to overcome the shortcomings of traditional virtual machines. Virtual machines have a complete operating system with its own memory management installed with the overhead associated with virtual device drivers. In a virtual machine, valuable resources are emulated for the operating system and client hypervisor, which allows to run many cases, one or more operating systems in parallel on a single machine (or host). Each guest OS operates as an individual entity host system.*

Moreover, Docker containers executed with the Docker motor instead of the hypervisor. The containers are therefore smaller than the virtual machines and allow faster startup with better performance, less isolation and more compatible as possible by sharing the hosts kernel.

The proposed solution involves monitoring Docker containers in a cluster and in turn provide suggestion system administrator about the use of resources to improve the overall performance of the proposed system. The system calculates the different type of resources and stores the data only if the standard deviation is outside a certain limit, it reduces the size of the database and improves the performance

Keywords—Hypervisor; Docker; API; Swarm; Virtual Machine

I. INTRODUCTION (PROBLEM DEFINITION)

A centralized system of resource monitoring to analyze the performance of Docker container is to be developed. Here solution will include CPU usage, memory usage, number of demands, the number of HTTP errors. Also trying to develop alert rules if the resource of a particular container Docker is overused. And the solution is not limited to the monitoring of local machines, but also for many different machines hosted on different groups. It is made using Docker Swarm API that is the support of the official API provided for communication.

II. LITERATURE SURVEY

A. What is Docker

Docker containers wrap a piece of software in one complete file system that contains everything it needs to function: code execution, system tools, system libraries anything can install on a server. This ensures that it will always run the same regardless of the environment it is executed.

Docker is an open-source project that automates the deployment of applications within software containers, providing an extra layer of abstraction and automation - Operating System- high level virtualization on Linux, Mac OS and Windows.According to industry analyst firm 451 Research, "Docker is a tool that can package an application and its dependencies in a virtual container that can run on a Linux server. This support allows flexibility of where the application can operate, either on site, public cloud, private cloud.

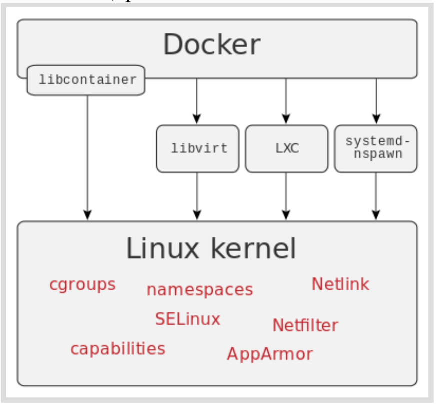

Figure 1: Docker Layer Architecture

B. How Docker Is Different From Virtual Machine

Containers enjoy isolation and allocation similar to virtual machine resources but a different architectural approach allows them to be much more portable and effective..

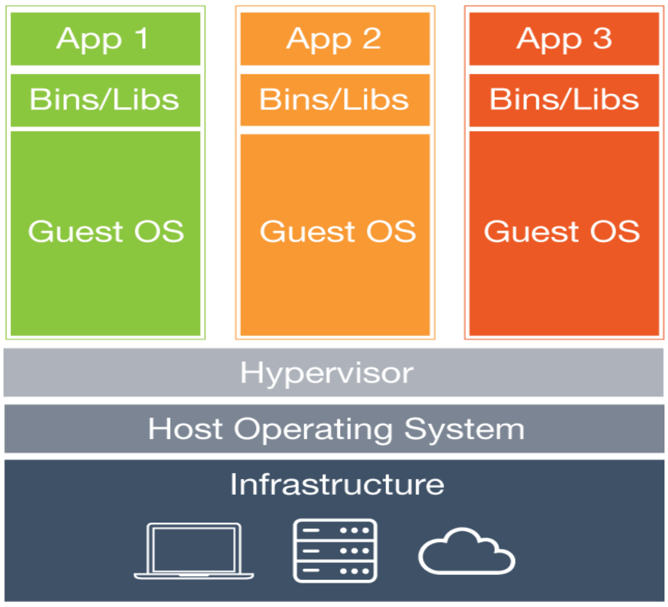

Figure 2: Virtual Machine Architecture

In Virtualiazation each virtual machine includes the application binaries and libraries needed and everything for a guest operating system - all of which can be tens of GB in size. .

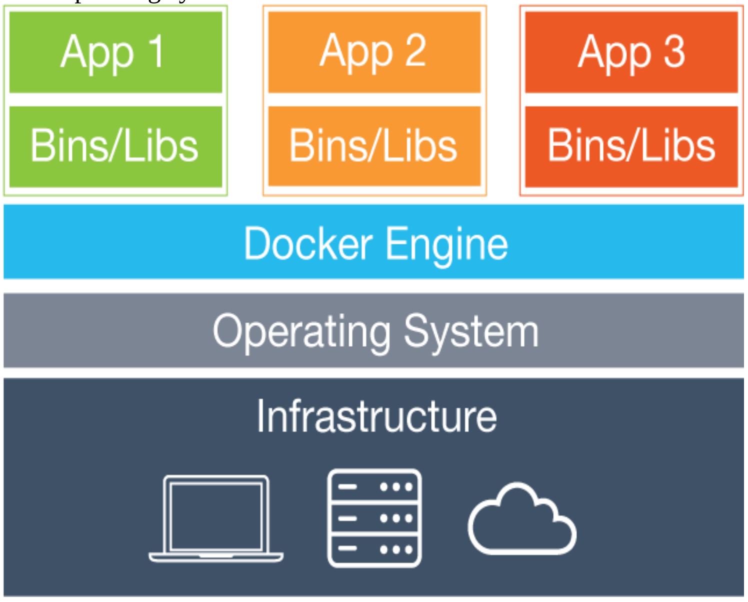

Figure 3: Docker Containers

Containers include the application and all its dependencies , but share the kernel with other containers. They function as an isolated process in user space on the host operating system. They are not related to any specific infrastructure Docker containers run on any computer, over any infrastructure , and in any cloud [1] .

C. *Kubernetes*

Kubernetes at its basic level, is an application management system on a containerized node cluster [1] . In many ways, Kubernetes was designed to address the disconnect between the way that a modern infrastructure and is designed cluster and some of the assumptions that most applications and services have on their environment.

D. *Dockersh*

Dockersh is designed to be used as a login shell on machines with several interactive users. When a user calls dockersh , it will establish a Docker container (if not already running) and then create a new interactive shell in the namespace of the container. Dockersh can be used as a shell in / etc / passwd or as an SSH ForceCommand.This allows a single ssh process on the normal ssh port site user sessions in their own individual containers docker in a secure environment and locked manner.

E. *Docker Swarm API*

Docker Swarm is the native grouping for Docker. It allows to create and access a host group Docker using the complete set of tools Docker. Because Swarm Docker. Docker uses the standard API, any tool that already communicates with a daemon can use Docker Swarm transparent to scale to multiple hosts. Docker Swarm following the exchange, the plug and play principle. As the initial development sets in, will develop an API to allow pluggable backend. This means that swap on programming backend Docker Swarm uses out of thebox with a backend. Swarms replaceable design offers an extraordinary experience smooth casing for most use cases, and allows large-scale production deployments to exchange more powerful backend. .

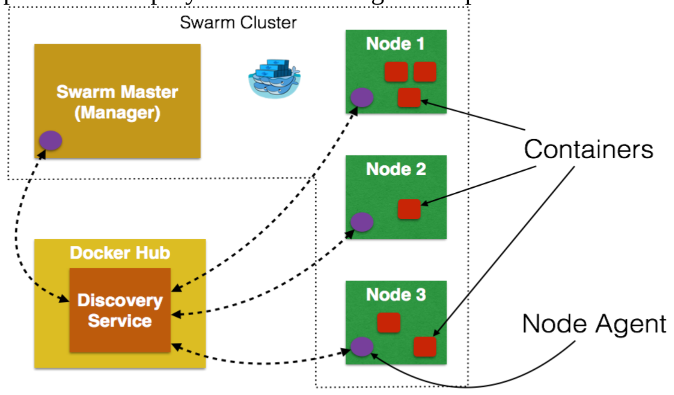

Figure 4: Docker Layer Architecture

III. SWARM API

A. Pull the swarm image and create a cluster.

1. 1 Pull the swarm image.
   ```
   $ docker pull swarm
   ```
2. Create a Swarm cluster using the docker command.
   ```
   $ docker run —rm swarm create
   6856663cdefdec325839a4b7e1de38e#
   ```

The **create** command returns a unique cluster ID (cluster id). You'll need this ID when starting the Docker Swarm agent on a node.

B. Create swarm nodes

Each Swarm node run a node agent . The agent records the Docker demon referenced, the monitors and updates the backend discovery with the status of the node.

1. Starting docker daemon with -H flag.
   ```
   $ dockerdaemon-H tcp://192.168.188.198:4243 -Hunix :///var/run/docker.sock
   ```
2. Registering swarm agent by swarm manager with the help of node's ip and port number in unique clusters
   ```
   $ docker run -d swarm join —addr =<node_ip:2375>token://cluster_id
   ```

C. Configure a manager

After nodes connected to swarm manager , manger controlling the agents,accesing information about the agent's nodes using docker remote api.Setting swarm manager on specific port.

```
$ docker run -d -p <port>:2375 swarm manage token://cluster_id
```
For getting information about swarm agent's using below command.
```
$ docker -H tcp://<ip:port> info
```

IV. PROPOSED SOLUTION

The first step is to store the resource monitoring parameters into the database.
Here the problem arises that, we cannot store the entire result (of every minute) into the database.

The solution is that we compute the standard deviation of the resource monitoring parameters.
Standard deviation measures how spread out the values in a data set are around the mean. In our case the dataset is the resource monitoring parameters i.e. CPU usage and Memory usage.

If the standard deviation is very small then we do not store the results in the database and if it crosses the certain limit, we store the result in database.

$$s = \sqrt{\frac{1}{N-1} \sum_{i=1}^{N} (x_i - \overline{x})^2}$$

Acknowledgment

We would like to express our deep gratitude to our Guide, PROF. PRAVEEN HORE, Faculty of the Department of Computer Engineering, Army Institute of Technology for all the valuable guidance and intellectual stimuli that he provided during the progress of this work.It was a privilege to work under the his valuable guidance and supervision.We are also grateful to Prof. M.B LONARE, Faculty of the Department of Computer Engineering for helping us from start,Prof. Sunil Dhore, HOD of the Department of Computer Engineering and other faculty members of Department of Computer Engineering for timely guidance and encouragement to complete this work.

We thank BhumikaBayani and Amol Tambe ,Great Software Laboratory , for giving us a good understanding of Docker System. It was a unique privilege to work under their valuable guidance and supervision. We are grateful for their timely guidance encouragement to complete this work.

We would also like to thank the library staff and Computer Centre for all the help and facilities that they extended during this project work.

References

1. Bernstein, David. Containers and Cloud: From LXC to Docker to Kubernetes.
2. Marcus Daniels. *Swarm and COM*. Integration workshop at the GIS/EM4 Conference on September 02, 2000.
3. Paul Johnson , Alex Lancaster. *Swarm User Guide*. Published 10 April 2000.
4. Raja, A.R, Kakadia. Virtualisation vs Containerisation to Support PaaS Dua, R.. VMWare, Bangalore, India
5. P. Mell and T. Grance. The NIST Definition of Cloud Computing.. Recommendations of the National Institute of Standards and Technology.
6. J. Petazzoni. Linux Containers (LXC), Docker, and Security. 2014
7. B. Butler. Containers: Buzzword du Jour, or Game-Changing Technology? Network- World, 2014