SETTING UP HADOOP CLUSTER NETWORK USING DOCKER

[1]Patil Pankaj Shantaram, [2]Prof. R. B. Wagh
[1]PG Student, [2]Assistant Professor
Department of Computer Engineering,
R. C. Patel Institute of Technology, Shirpur, India.

Abstract : Apache Hadoop is a popular big data framework that is being used a lot in the software industry. As a distributed system, Hadoop runs on clusters ranging from one single node to thousands of nodes. If any want to test out Hadoop, or don't currently have access to a big Hadoop cluster network, then they can set up a Hadoop cluster on their own computer, using Docker. Docker is a popular independent software container platform that allows you to build and ship applications, along with all its environments, libraries and dependencies in containers. The containers are portable, so one can set up the exact same system on another machine by running some simple Docker commands. Docker is easy to build, share and run applications anywhere, without having to depend on the current operating system configuration.

IndexTerms **- Docker, VMware.**

I. INTRODUCTION

1.1. Introduction of Docker

Docker is a containerization platform which is used to packages applications and all its dependencies together in the form of a docker container to ensure that application works properly in any environment. Docker is designed to benefit both Developers and System Administrators, making it a part of many DevOps tool chains. Developers can write their code without worrying about the testing or the production environment and system administrators need not worry about infrastructure as Docker can easily scale up and scale down the number of systems for deploying on the servers.

Also docker consist of three important terms as follow:

a) Docker Image

Docker Image can be compared with a template which is further used to create Docker Containers. Also they can called as the building blocks of a Docker Container. Such Docker Images are created using the build command. These type of Read only templates are used for creating containers by using the run command. Docker commands are explore in depth of the "Docker Commands blog". Docker lets people or companies to create and share software through Docker images. Therefore Docker container can always run the software in a Docker image.

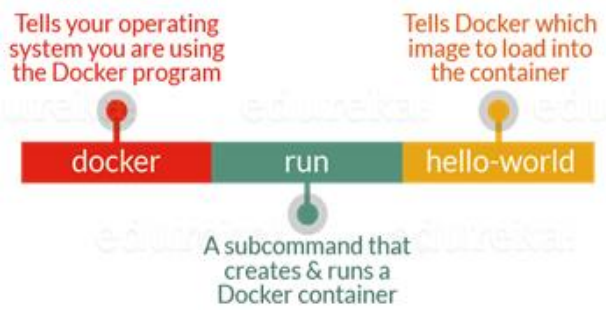

figure 1: docker image

b) Docker Container

Docker Containers are standardized unit which can be created on the fly to deploy a particular application or environment. To full-fill the requirement from an operating system Docker container could be an Ubuntu container, CentOs container, etc. Also, like CakePHP container or a Tomcat-Ubuntu container which are application oriented container. Containers are the ready applications created from Docker Images or one can say a Docker Container is a running instance of a Docker Image and they hold the entire package needed to run the application which is nothing but a ultimate utility of Docker.

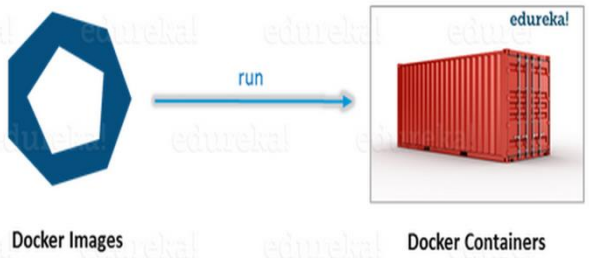

figure 2: docker container

c) Docker Registry

Docker Images are stored in Docker Registry, which can be either a user's local repository or a public repository. It is similar as Docker Hub allowing multiple users to collaborate in building an application. By uploading containers into the Docker Hub, multiple teams from same organization are allow to exchange or share containers. Docker Hub is Docker's very own cloud repository similar to GitHub.

1.2. Docker Architecture

Docker Architecture consist of Docker client which is used to trigger Docker commands, in which Docker Host is running the Docker Daemon and a Docker Registry for storing Docker Images. The Docker Daemon running within Docker Host is responsible for the images and containers. To build a Docker Image, CLI (client) is used to issue a build command to the Docker Daemon (running on Docker Host). The Docker Daemon will then build an image based on given inputs and save it in the Registry, which can be either Docker hub or a local repository. If one do not want to create an image, then just pull an image from the Docker hub, which would have been built by a different user. Finally, for creating a running instance of my Docker image, it is necessary to run command from the CLI, which will create a Docker Container.

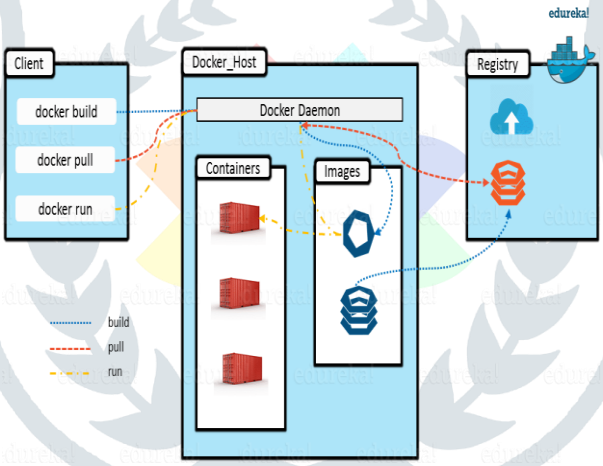

figure 3: docker architecture

1.3. Docker vs VMware

Virtual machines have a full OS with its own memory management installed with the associated overhead of virtual device drivers. Valuable resources in virtual machine are emulated for the guest OS and hypervisor, which are responsible to run many instances of one or more operating systems in parallel on a single machine or host. Every other kind of OS runs as an individual entity from the host system. On the other hand rather than the hypervisor, Docker containers are executed with the Docker engine. In such case containers are smaller than Virtual Machines and enable faster start up with better performance, less isolation and greater compatibility possible due to sharing of the host's kernel.

Containers are looks like "light" version of virtual machines and also they are different. Therefore containers and virtual machines can solve two different problems. Initially, virtual machines can solve a problem with service use, in which they abstract hardware from the operating system and create a separate environment for operating systems and applications. Consequently, it is possible to consolidate many virtual machines in a single physical server.

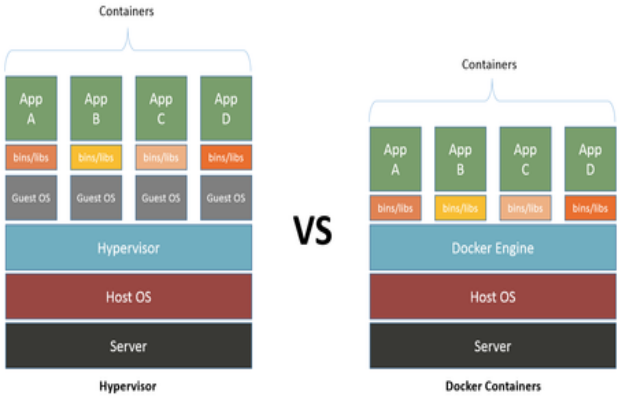

figure 4: docker vs vmware

Containers are used to abstract applications from the operating system and also they can make applications more portable. In comparison them though, containers can be more efficient than virtual machines at one or several levels. Such as containers can start in fractions of a second rather than minutes. Also containers are take up MB rather than GB on the disk. Also same workstation can run 10 or 100 times more containers than virtual machines.

In this section Project risks and the approach to managing them are discussed.

- Easy data ingestion:

Copying data to and from the cloud is as simple as copying data to a standard file system using Direct Access NFS. Applications can therefore ingest data into the cloud server in real time without any staging areas or separate clusters just to ingest data.

- Existing applications work:

Due to the unix platform's POSIX compliance, any python application works directly on cloud without undergoing code changes. Existing toolsets, custom utilities and applications are good to go on day one.

- Multi-tenancy:

It support multiple user groups, any and all enterprise data sets, and multiple applications in the same cluster. Data modellers, developers and analysts can all work in unison on the same cluster without stepping on each other's toes.

- Business continuity:

To protect against both hardware failure as well as site-wide failure, cloud provides integrated high availability (HA), data protection, and disaster recovery (DR) capabilities.

- High scalability:

Scalability is key to bringing all data together on one platform so the analytics are much more nuanced and accurate. cloud is the only platform that scales all the way to a trillion files without compromising performance.

- High performance:

The Distribution for cloud was designed for high performance, with respect to both high throughput and low latency. Also fraction of servers are required for running cloud distributions, leading to architectural simplicity and lower capital and operational expenses.

II. RISK IDENTIFICATION

Security Information and Event Management (SIEM): Analyze and correlate large amounts of real-time data from network and security devices to manage external and internal security threats, improve incident response time and compliance reporting.

- Application Log Monitoring:

Improve analysis of application log data to better manage system resource utilization, security issues, and diagnose and pre-empt production application problems.

- Network Intrusion Detection:

Monitor and analyze network traffic to detect, identify, and report on suspicious activity or intruders.

- Fraud Detection:

Use pattern and anomaly recognition on larger volumes and variety of data to detect and prevent fraudulent activities by external or internal parties.

- Risk Modelling:

Improve risk assessment and associated scoring by building sophisticated machine learning models on cloud that can take into account hundreds or even thousands of indicators.

III. RISK ANALYSIS

The risks for the Project can be analyzed within the constraints of time and quality.

table 1: risk table

ID	Risk Description	Probability	Impact		
			Schedule	Quality	Overall
1	Security Information and Event Management (SIEM)	Low	Low	Low	Low
2	Application Log Monitoring	High	High	High	High
3	Network Intrusion Detection	Low	High	Low	Low
4	Fraud Detection	Low	Low	Low	Low
5	Risk Modeling	High	High	High	High

table 2: risk probability definitions

Probability	Value	Description
High	Probability of occurrenceis	>75%
Medium	Probability of occurrenceis	26−75%
Low	Probability of occurrenceis	<25%

table 3: risk impact definitions

Impact	Value	Description
Very high	>10%	Schedule impact or Unacceptable quality
High	5−10%	Schedule impact or Some parts of the project have low quality
Medium	<5%	Schedule impact or Barely noticeable degradation in quality Low Impact on schedule or Quality can be in corporated

Following are the details for each risk.

table 4: risk description for id 1

Risk ID	1
Risk Description	Analyze and correlate large amounts of real-time data from network and security devices to manage external and internal security threats, improve incident response time and compliance reporting.
Category	Security Information and Event Management.
Source	Cloud Cluster
Probability	Low
Impact	Low
Response	Mitigate
Strategy	High Performance
Risk Status	Rarely Occurred

table 5: risk description for id 2

Risk ID	2
Risk Description	Improve analysis of application log data to better manage system resource utilization, security issues, and diagnose and pre-empt production application problems.
Category	Application Log Monitoring
Source	Processing Server
Probability	High
Impact	High
Response	Continuous
Strategy	Multi Tenancy
Risk Status	Occurred

table 6: risk description for id 3

Risk ID	3
Risk Description	Monitor and analyze network traffic to detect, identify, and report on suspicious activity or intruders.
Category	Network Intrusion Detection
Source	NameNode
Probability	Low
Impact	Low
Response	Mitigate
Strategy	High Scalability
Risk Status	Identified

table 7: risk description for id 4

Risk ID	4
Risk Description	Use pattern/anomaly recognition on larger volumes and variety of data to detect and prevent fraudulent activities by external or internal parties.
Category	Fraud Detection
Source	Web Server - API
Probability	Low
Impact	Low
Response	Mitigate
Strategy	Easy Data Ingestion
Risk Status	Identified

table 8: risk description for id 5

Risk ID	5
Risk Description	Improve risk assessment and associated scoring by building sophisticated machine learning models on Hadoop that can take into account hundreds or even thousands of indicators.
Category	Risk Modeling
Source	Client Interface
Probability	High
Impact	High
Response	Continuous
Strategy	Existing Application Work.
Risk Status	Occurred

IV. PESUDOCODE

This algorithm uses Conatiner selection and greedy approach, which is used to distributes the load over Containes to achieve efficient performance in heterogeneous cloud computing environment. The algorithm depends on current resource allocation count.

- Input:

List of Conatiners in Containers_List(), it is used to maintain an index table of Containers with current allocation count for every ConatinerCl_Table(C_id) , K where K is the number of Conatiners that will be selected randomly. Conatinerids(), it is used to maintain the index of selected node randomly with its current load, TempCid is a temp Cid that selected randomly.

- Output:

Cid is the Conatiner id that is selected to assign the load.

0. Distribute the Conatiners over the Hosts according to the host's qualification (Conatiner provisioning).

1. Initialize, *Cl_Table(*0..n-1*)* ← **0** At start all Coatiners's have zero allocation**.**, *K*← *m, C_id*←*-1* , *Cids()=-1,i*← *0,* currCount← 0, minCount← *Max_Value*, *TempCid*← **-1**;

2. Parses C_List() to LoadBalancer:

3. ***For i← 0 to k*** *//SelectContainer randomly*

4. *TempCid← random* ***(****C_List()****)****.***

5. *C_id← TempCid*

6. ***If C_id Exist in Cl_Table(****C_id****) then***

7. currCount← *Cl_Table(C_id)*

8. ***Else***

9. currCount← ***0***

10. **C***ids()* ← *(C_id,* currCount*).*

11. ***End for***

12. *TempCid←* ***-1***

13. currCount← ***0***

14. ***For i ← 1 to k***

15. *TempCid ← i*

16. currCount← ***C****ids(TempCid)*

17. ***If*** **currCount<minCount then**

18. minCount= currCount

19. **C**_*id← TempCid*

20. ***End if***

21. ***End for***

V. SCOPE

Containers have reduce the overhead required for deployment and accepted widely. In the minds of developers, administrators, and planners technology of Docker is set and they are using it to increase the performance of the network. Also containers are quickly provide developers a clean environment in which they can experiment freely.

Now a days 44% of enterprises are looking to adopt a DevOps within their organization. Also such cultural shift is reducing the traditional barrier between Developer teams and IT operations teams. Goal of developers team and IT operation team is to help enable DevOps within the enterprise via the Docker platform. To secure and manage entire environment, IT operations teams manages the security features such as role-based access controls, image signing and image scanning. The applications is build in a self service manner by developers which is enabled by Docker CaaS platform and select from image content that the IT operations team has seemed okay for developer use. In such way developers can build new applications, quickly and securely using these images.

VI. INNOVATIVENESS

While writing, testing and deployment of application inside containers is in processing, the environment should not change at different parts of the delivery chain, which makes environment consistent. Which also makes collaboration between different teams like developers, testers and administrators, easier because they all are working with the same containerized environment.

For application updates on a constant, streamlined basis the software requires continuously delivery of software. In such case containers can help, which make it easy to apply updates to applications. When application is distributed into multiple micro services, each one hosted in a separate container, then update one part of the application by restarting the container without interrupting the rest of the applications.

VII. USEFULNESS

Support for multiple frameworks. DevOps are having the ability to switch easily between different programming frameworks or deployment platforms is easy. Containers are relatively agnostic toward programming languages and deployment platforms. Any type of application can be run inside a container, regardless of the language it is written in. Containers can also move easily between different types of host systems. Such as, if user want to switch from Red Hat to Ubuntu, then they can do that very quickly with containers. Also user can't move a containerized application from a Linux to a Windows server, or vice versa, because Docker on Windows is very different from Docker on Linux.

VIII. MARKET POTENTIAL AND COMPETITIVE ADVANTAGE

a) Inventory

At start Docker will help in understanding the problems associated with containers. Developers work on containers and in the end have a finished product in which the desired application runs smoothly and usually work with a finished image, To develop a web application based on Ubuntu 16.04, should start with a container in which Ubuntu and Apache are already installed before adding another application. If the application is running, the developer can create an image from container and forward it to the administrator, after making a few adjustments to the container.

The administrator works on IT infrastructure operations. In this administrators loads the container onto a Docker host and puts it into operation. Immediately afterward, the container service is available on the network. The developer and administrator can give themselves a path on the back and remove it from their to-do list. It only becomes apparent later that the container could be a problem.

b) Update Problem

When software needs to be up to date, problem becomes evident also involving a security update. Some errors in the SSL standard library and the problem in the C library resolver illustrate this problem. Also administrator has to update multiple systems at the same time. Distributors provide a suitable update to provide the solution for physical machines, the system's update function or an automated solution installs the appropriate packages.

To solve the update problems, container have to enable automatic updates and also built in line with common standards so that updates works properly. It seems to be a complicated with Docker container, in which administrators update their capabilities which is based on third-party provider's basic image. Also locally installed C library can be updated directly in the container. In other hand developers also have to recreate the container with the application again for necessary security fix from the beginning.

c) Development vs Operations

In development task, improvements are made in existing applications, also new features are added and adapting the container to different conditions. Also administration team concerns itself with stability, security, and maintainability. Developers are using existing component of a container in application when they are providing new features.

If new version of the software is appeared then it must operate within the framework. In such work administrators are having totally different interest with regard to the operation of a platform. Due to which application will fits into the existing infrastructure. Also administrators should resist temptation, if container is a temptation and looks feasible at first while deploying the original development.

Containers are completely providing benefits that's why companies can't forget them. The problems are not caused due to use of containers but by the problematic way in which they are used, and also administrators are having different ways to employ containers without the concomitant frustration.

IX. TIME COMPLEXITY

In this work two metrics are used to measure the performance as follow:

a) Response Time

It is the time interval between sending a request and receiving its response. Also it can minimize the response time in order to enhance the system performance. The total response time can be obtained as follow:

Total response time = The Docker request processing delay + Network delay

b) Processing Time

Average processing time is the amount of time actually needed to process a particular task.

X. CONCLUSION

Docker technology is starting point which meets the needs of the development of container cloud, the performance and service of Docker in the distributed platform. Also it solves the shortcomings of the current Docker container only for single node application, and setting up the hadoop cluster network to improve the development deployment operation and maintenance efficiency.

REFERENCES

[1] Liu Lijuan. 2017. Research and implementation of Docker performance service in distributed platform. International Journal Of Engineering And Computer Science, ISSN:2319-7242, Page No. 23072-23076.
[2] Varghese, B., Subba. 2016. Container-Based Cloud Virtual Machine Benchmarking, 1601.03872.
[3] Turnbull J.. 2014. The Docker Book: Containerization is the new virtualization.
[4] B B Rad, H J Bhatti, M Ahmadi. 2017. An Introduction to Docker and Analysis of its Performance. International Journal of Computer Science and Network Security, Vol.17, No. 3.
[5] Qi Zhang, Ling Liu, Calton Pu, Qiwei Dou, Liren Wu, Wei Zhou. 2018. A Comparative Study of Containers and Virtual Machines in Big Data Environment, 1807.01842.
[6] Shaun McCullough. 2017. Using Docker to Create Multi-Container Environments for Research and Sharing Lateral Movement Scenarios. The SANS Institute.