Journal of Theoretical and Applied Information Technology
15th May 2022. Vol.100. No 9

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

IMPROVED PERFORMANCE OF CLOUD ENERGY USING CONTAINERIZATION WITH MULTIPLE MANAGERS AND CONSENSUS ALGORITHM

MS. VASANTHA KUMARI N[1], DR. SHANMUGARATINAM [2]

[1]Assistant Professor, Presidency College/Presidency University, Department Of Computer Applications,India

[2]Associate Professor, Presidency University ,Department Of Computer Science And Engineering,India

Email:[1]vasantha.kn@gmail.com,[2]shanmugaratinam@presidencyuniversity.in

Abstract

One of the favored topics in data centers and web application developers today is Containerization. Within the path of reducing the energy in cloud data centers virtualization is one among the important solution for minimization of energy in cloud. In recent trends of software data centers, Docker container is hottest tool used for Virtualization. In this paper, main concern is to construct the cluster node and schedule for docker containers. With the adverse use of containers by IT administrators and developers, cluster nodes are often established as one virtual machine. Docker is an open source engine which was introduced by Docker Inc. in 2013 under apache 2.0 license. The most objective of the Docker is to make an environment where programmers develop the code efficiently. A recent trend in docker technology with the introduction of tool called Swarm which extends its feature to numerous swarms in multiple clouds. Swarms of Docker containers with multiple managers help guarantee high service availability and improve election time during execution using Consensus Algorithm. Swarm with multiple managers, performance is improved with the algorithm. When there are increased numbers of managers any failures of the node are often recovered without downtime. This paper also explains about model of constructing a virtual system on multiple clouds in distributed systems. It also discusses about novel methodology for traditional virtualization.

Keywords—*Docker, Swarm, Cloud, Container, Virtual Machine*

1 INTRODUCTION

The container technologies have widely been accepted for building next-generation Cloud systems. In green cloud technology, virtualization has become a major concern for reducing the energy [1]. Container based technology using docker has evolved as a predominant for developing different types of software and also become familiar in IT field [2]. Virtualization refers to partitioning of various virtual machines by allowing many operating systems and design without the need of independent systems [3]. Swarm based technology in docker is a novel technique in cloud IT [4].Dockers containers are used over virtual machine as they are light weight. Performance of existing method is compared with the proposed methodology to improve the request time. With the use of multiple managers, docker swarm load balancer will distribute the load amongst the containers. Dockers are required for developers to run and deploy applications. Containers are required for the following purposes like:

A) Convenient to operate: It is an open source tool which can be installed in any system with an operating system that supports virtual box like Docker for Mac/Windows.

b) Efficient use of resources: Containers operate more efficient than virtual machines as it uses very less instances.

In fig1 explain about the architecture of Docker Client Server Architecture in which docker daemon generates docker images which are recorded by daemon. The docker client communicates with docker daemon which also does building, running, and dispenses docker containers.

Journal of Theoretical and Applied Information Technology
15th May 2022. Vol.100. No 9

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Docker Daemon:

Docker daemon pay attention to API needs and accomplishes images, containers, networks, etc. A daemon talks to another daemons to perform different services.

Docker Client:

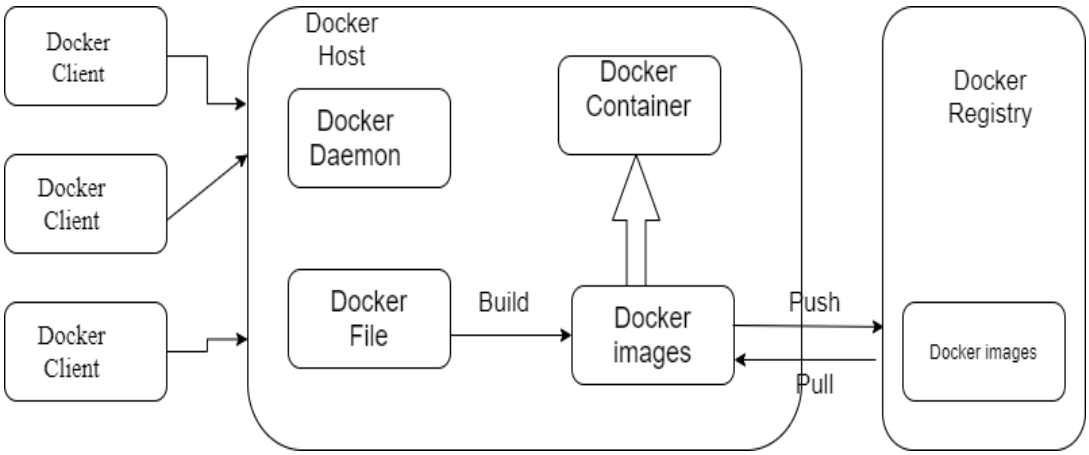


Figure.1 Docker Client-Server Architecture

Docker client plays an important role through which docker users communicate with docker. There is also possibility to have docker client or docker daemon in the same host or in the remote machine also.

Docker Registry

The main purpose of docker registry is to load docker images in which images are used to generate containers. Docker hub is one of the registries used by the users to load thousands of images. Commands like ***docker pull*** or ***run*** are used to pull the images from registries.

Images can be created by our own or others. To construct our own image, *docker file* needs to be created by explaining about the steps to create an image and execute it. [5].

Docker Containers:

A docker Container is an instance of an image. Docker Command line interface is used to create, delete or start a container. The best example is docker run command which attaches container to local command line session if no network options are specified.

Overall, our work makes three fold contributions to the docker container system:

1. Our comparative analysis and results can help both the researchers and developers to better understand about containers and docker swarm and also setting up the environment for creating a swarm cluster.

2. This method will help to build simulation of docker swarm based for distributed system development process on cloud. The simulation is based on Docker swarm, virtual box, nginx, etc.

3. The performance can be improvised with Raft algorithm than the existing approach which improves Latency by analyzing the response time, MTBF and election time.

The remainder of this paper is organized as follows:
Section 2 briefly explains about setting up the environment for containers. Section 3 summarizes the related work done and evaluation performance of different methods. Section 4 explains about building container images and execution. Section 5 and 5.1 highlights about the proposed methodology used in this paper with its simulation of building container of cluster nodes. Section 6 compares virtualization technology with containers and virtual machines. Section 7 analyses the best approach for virtualization by considering different parameters. Conclusions and future work are discussed in section 8.

2 SETTING UP THE ENVIRONMENT FOR DOCKER CONTAINERS:

Installing Docker for Windows:

Docker provides an environment for developing applications. Use docker desktop installer.exe file for installation. To run successfully on windows 10 hardware requirements are 64-bit processor and 4GB RAM. After installation enable Hyper-V option. There are constituents connected with docker .In few installations, BIOS setup manipulation would be required. There are different versions of docker available for windows can be used based on the requirement. Different terminal emulators are used with different features.

Verifying Docker Install:

Containers are the basic component of Docker toolkit. Nginx is a basic web server which is easy to use. The important consideration is to check whether docker is installed properly. Docker terminal need to be checked for working of docker commands. If the choice is Mac, then Terminal or iTerm2 can be used. On the other hand, if windows operating system is used then Power shell. Instead if docker toolbox is used, then Terminal called Quick Start Terminal is available with it.

***Docker version*:** It is a command which checks version of the docker is working.

Docker command line will communicate to the server on machine and return its values.*'sudo docker version'* is the command used if Linux operating system is used. To know more about the configuration and setup information for our engine. It details about number of containers that are running or expectable number of images that can be stored in docker. Docker command line structure has old and new format but old way format still works. Docker is generating a new model of the management commands with space command. For example, '*docker run*' can also be executed as '*docker container run*' which is the new of execution.

3 BACKGROUND AND RELATED WORK

In current IT, both the technologies containerization and virtualization have the equal prominence. The variance between these two is how it attains isolation between various machines. Both of these technologies use concepts which can be used conforming to the demand. Containers are more friendly to the developers in deploying and also during boot up. With more workloads containers performs better than virtual machines in terms of space and processing [6].Containers hosts on a single machine whereas virtual machine has a host OS which runs on multiple guest operating systems. Container and its usage lead to low redundancy than virtual machines [7].Unlike virtual machines; containers have moderate limits to resource usage. The time period estimation for building containers is faster than virtual machines. They are very scalable as they can move from cloud environment to the independent machine and also can be put back [8].

Many works has been accomplished earlier with regards to swarm node and its creation. Containerization using virtualization is acquiring strength. There are different container based technologies like Docker, etc. used in recent trends. Virtualization is one of the method which favors migration from physical to virtual circumstances. Docker is an open source software available formulated on Linux Containers which is a set of software usage [9].In paper [10], author says that virtualization over cloud using docker is performed. In this paper, docker containers are exactly deployed for the different applications with technique of containerization which results in low economy in spite of virtual machines. Author explains in [11] that applications are deployed and their problems are articulated. Author Nitin explains in the paper [12] how docker swarm and cabernets used in cloud computing environment. It tells about how work has been distributed between the hosts with regards to that hosts will not be overloaded. It gives proper explanation of how containers are used in code development. In [13], organization of containers has made the placing the containerized applications on a cluster easier. It also provides precise explanation about various job scheduling policies implemented by the managers of the cluster. Paper [14] presents model of constructing a virtual system of systems for the scattered on several clouds. Modelling is based on swarm, virtual box, nginx, etc. These modelling are very useful in constructing software systems. The more explanation about how docker works is available with development tools in [15] and also docker swarm tool is available which does the function of clustering and scheduling which converts collection of nodes into a cluster. Docker only supports the questions with relation to observe the performance for a container and also few testing [16] and analysis also done on it. Application has been developed and the information is stored using mango dB database. [17] Application has been deployed using 3 node swarm cluster. Through this cluster nodes will communicate with each other. Docker approach differs completely from traditional method of virtualization [18].This paper considers number of

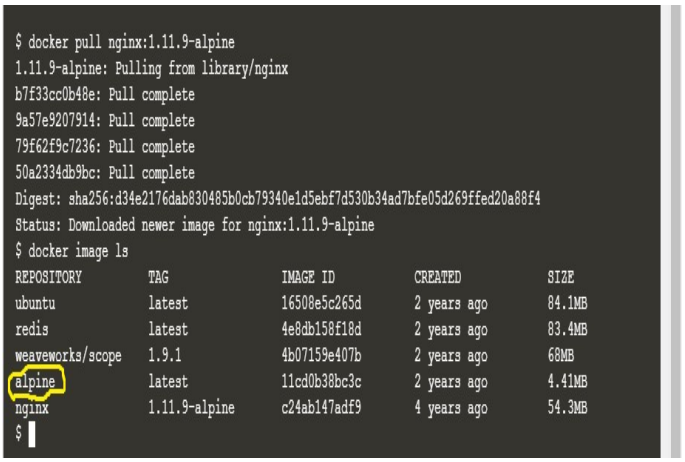
GA islands in clouds. The research work former to this states about execution of cloud using Genetic Algorithm (GA) [19]. The research which started having executed the GA in cloud has been experimented by Dziurzanski a. The more explanation is shown about service and task to execute on the manager and worker nodes. Docker engine executes in swarm mode using different command formats [20].Micro service architecture is explained in detail to overcome the drawbacks of monolithic architecture which separates complex applications and light weight applications [21].In prior research work, cloud has been organized at different levels using occopus. Occopus is a context which is used to dispense applications on individual or many clouds and also management of nodes [22].Service like docker machine is used to deploy application with docker swarm and nodes on variety of multiple clouds for which docker machine driver is available [23]. For arranging the nodes in a cluster, it's required for docker objects like images, containers and volumes [24].Particle Swarm optimization is suggested for the implementation of node with different intentions of saving power while nodes are connected. By contemplating connectivity of different nodes and its arrangement, a novel based algorithm for PSO is proposed to reduce the energy in fully connected cluster node [25].

4 BUILDING CONTAINER IMAGES AND ITS EXECUTION

In this section, more discussion is about what an image is, how to build them and run containers from it. Images of a docker are constructed from docker file. A docker file has proper defined steps about creating an image with the structure of an application. Built images can have a move between different environments and also that if it works in one environment then it will definitely work in another environment. An image is a collection of root file system and also parameters required for execution during run time of a container. Inside a there is no whole operating system [26]. It's just the binaries that application requires as host provides the kernel. This one feature differs containers from virtual machine. It can be small binary file or big as Ubuntu with its package manager. The instance of NGINX can be generated in docker container from docker hub [27].Images are not certainly named but they are tagged.

```
$ docker image ls
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
ubuntu              latest              16508e5c265d        2 years ago         84.1MB
redis               latest              4e8db158f18d        2 years ago         83.4MB
weaveworks/scope    1.9.1               4b07159e407b        2 years ago         68MB
alpine              latest              11cd0b38bc3c        2 years ago         4.41MB
nginx               1.11-alpine         bedece1f06cc        3 years ago         54.3MB
$ docker pull nginx
Using default tag: latest
latest: Pulling from library/nginx
45b42c59be33: Pull complete
8acc495f1d91: Pull complete
ec3bd7de90d7: Pull complete
19e2441aeeab: Pull complete
f5a38c5f8d4e: Pull complete
83500d851118: Pull complete
Digest: sha256:f3693fe50d5b1df1ecd315d54813a77afd56b0245a404055a946574deb6b34fc
Status: Downloaded newer image for nginx:latest
$
```

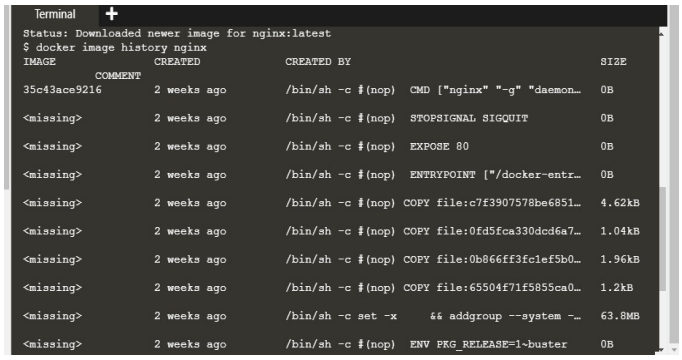
Command 'docker image ls' lists all the images so far worked in the machine. There is a tag 'latest' which is a special tag. It doesn't guarantee the latest commit in the repository but it generally means latest version of the product [28].



If we need to download the standard version of alpine 1.11.9-alpine and then do 'docker ls'. It will display different versions with same image ID because image id depends upon cryptographic SHA (secure hash algorithm) of each image in docker hub. Custom Nginx version can be created as it is an open source.

4.1 Layers of an Image:

Images are planned with the concept of union file system of creating layers. Every image starts with a blank layer called scratch



Layers of docker image are just a file which is created by executing few commands. All the layers of an image can be viewed with 'docker history' command [29].When we start a new image, it will have one layer. Every layer will have its own SHA that has the feature of finding whether one layer is identical to another layer.

5 PROPOSED METHODOLOGY: DOCKER SWARM NODES AND SERVICES

Docker swarm can be defined as container arrangement machine. Applications executing on multiple nodes in a cluster should make use of resources efficiently with minimum percentage of down time. To do this, the setup of the architecture should be able to balance the load on traffic. There is a concept of container orchestration which removes the issues by deploying the application on multiple nodes depending on the traffic which are under execution. It is an orchestration tool which performs multiple tasks at lowest down time [30]. Swarm helps developers to manage the load by scaling the containers up and down. The main element of docker swarm is a ***node***. It is a separate docker engine involving in a swarm. Two or three nodes can be executed on a single physical machine or on a cloud server. There are different types of nodes in docker swarm [31]. The essential nodes are *manager nodes* and *worker nodes.* Manager nodes maintain states of a cluster, schedule the services. Docker always suggests having odd number of nodes in a cluster. In docker swarm N manager will stand the failure of max of (N-1)/2 managers. Worker nodes also part of the swarm which is considered as element of the docker engine which is also necessary to execute containers [32].In a machine with a single node, '*docker service create'* command can be used. A node can be promoted or demoted based on the requirements using relevant commands.

IP addresses has to be assigned to the nodes on the operating system which can be accomplished using the command 'ip a s'. Docker has a privilege of API for communicating with the daemon process and creates service objects for each task. Tasks are then assigned to the nodes for execution. The services in the swarm are visible to the users through ports and addresses.

Requisite of Swarm:

Integration of cluster and docker engine: Application services are deployed using docker engine command Line Interface (CLI) but no additional programming package is required to manage the swarm.

Scaling of container: For the number of tasks executing on a machine, swarm manager spontaneously scales up or down by removing tasks to maintain the desired condition [33].

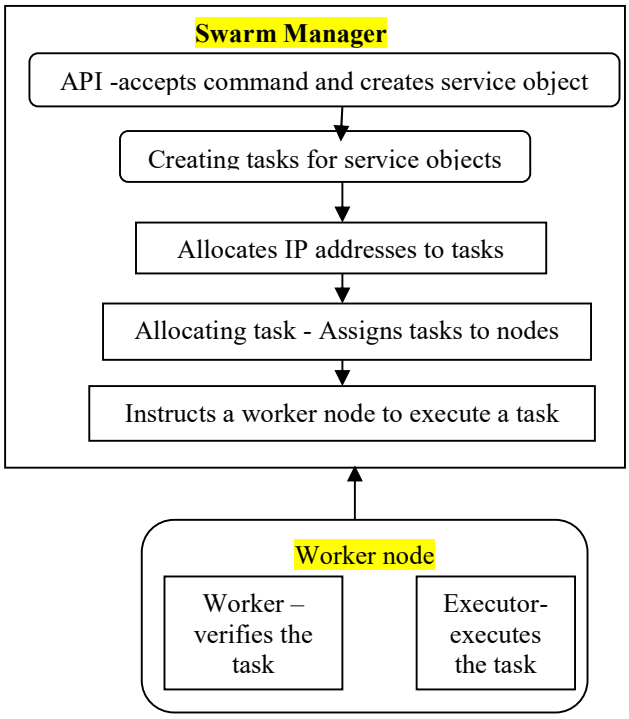


Fig 2. Work Flow Of Docker Swarm Node Cluster

Overlay network: Network can be formed depending on the services. Swarm manager assigns addresses to the containers in an overlay network [34].

When the docker engine executes in swarm mode, Raft Consensus algorithm is implemented by manager nodes. It is used to ensure that all the manager nodes are having worker nodes. IP address of the manager node is required.TCP and UDP ports 2377 should be open for communication.

Load balancing: Tasks are distributed between the containers nodes. The mechanism concentrates on balancing the workload traffic between the nodes [35].

Working of nodes: Multi manager

Docker engine has the feature of creating a cluster of nodes with multiple managers and workers. Swarm cluster consists of few nodes which has docker engine. A manager node carries the task of handling cluster management like managing the state of cluster, Arrangement of services, etc. cluster with one manager also be created but if the manager goes to state of failure the services which are under execution can be sustained only when there is another cluster to make progress. To improve the performance swarm fault tolerance docker proposes to implement odd number of nodes according to the request of the client. The advantage of having multiple managers is that disaster of a node can be improved without down time. Tolerance of a cluster manager depends on number of managers. Cluster with three managers endures with damage of single manager.
Similarly with five managers stands with damage of two managers Nodes. If managers fail, it will be difficult to manage the swarm. Worker cannot be regarded as a manager until particular node is promoted.

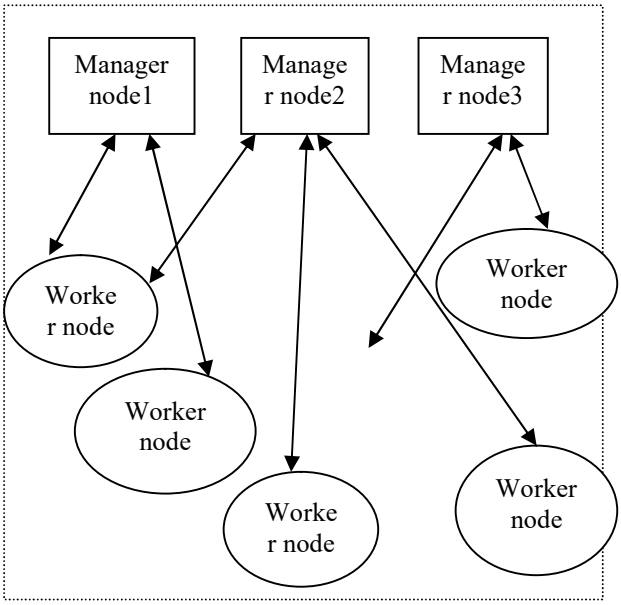


Fig3. Swarm Cluster With Multiple Managers And Workers.

Using manager node, worker nodes are managed and updated. Services are the jobs that are to be executed on manager node. Jobs are defined as tasks that are executed on manager and worker nodes. Manager node is responsible for accepting commands and creating service objects and executes a task. It also controls the workflow in worker node. Worker nodes perform the execution of assigned tasks in the container. Docker swarm must have a manager that can assign tasks to worker nodes. By using more than one manager node, developers ensure that the system can continue to function even if the manager node fails. Docker recommends a maximum of seven manager nodes per cluster.

5.1 Simulation: Building Containers Using Docker Swarm Cluster Nodes with Multiple Managers

More than one manager can be set up in a swarm with multiple nodes. One acting as primary and other as a secondary manager. The following command syntax is being used to up the container machines:

docker-machine create -d virtualbox --swarm \

--swarm-master \

--swarm-opt replication \

--swarm-discovery hostname ://ip-addr \

--engine-opt cluster-store=hostname://ip-addr \

manager-0

In the similar manner, manager 1 should also be executed for start-up. eval command is used to connect to any of these managers.

#Docker –machine stop manager-0 is used to stop the manager 0. `docker ps` command will diplay all the containers which are under execution. To work with any of the containers, `docker exec –it container-name bash` syntax format is used.

Before creating the swarm cluster of nodes, testing should be done to know whether swarm feature is taking place or not. This can be done with the command `docker info`. Node can be executed with init for initialization of docker swarm. Initially create primary manager and its nodes and then secondary manager and its nodes. When primary manager goes to off mode, then secondary manager allocates work to the subordinate node.

```
docker@manager1:~$ docker node ls
ID                            HOSTNAME            STATUS              AVAILABILITY
qd6jx4m2f4f8ehqya6qhct5y7 *   manager1            Ready               Active
wgjiyvdmrnjouq0nr3gic5mzy     worker1             Ready               Active
ja2z7bmsrh37jnkvy6drux4m0     worker2             Ready               Active
docker@manager1:~$
```

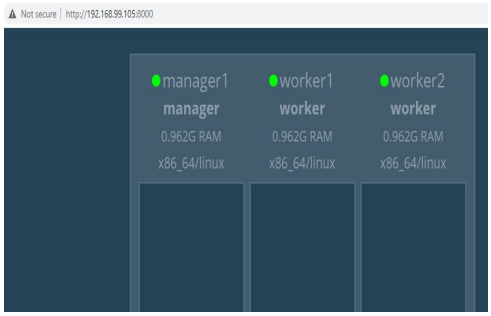
```
docker@manager2:~$ docker node ls
ID                            HOSTNAME            STATUS              AVAILA
o5zhdvd9czb8yw2ykk7pvkxxi *   manager2            Ready               Active
4m3ib1uw6qsfzeqpidcmve1xg     worker3             Ready               Active
jrklhh8tft43hv6gjcl1gbbh6     worker4             Ready               Active
docker@manager2:~$
```

There are different options for creating set of cluster of nodes like using playwithdocker .com which has built-in docker in it.It has a disadvantage that all the data will be wiped off every four hours. Second way is by using docker machine which is a command line tool which actually can be mounted on operating systems like windows or MAC. This tool provisions set of virtual machines on a local machine. Hence we require a machine with sufficient amount of RAM. Another option is to use Digital Ocean. After using any of these options for installation, testing is required whether it's working properly or not. It can be started through simple docker commands and later with `***docker machine create***` command to create set of nodes. After which there is a need to ssh into a particular node using the command '***docker-machine ssh***`.

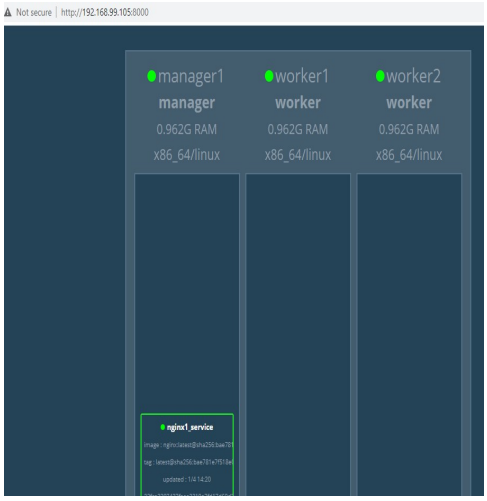
Using the set of commands, manager nodes and worker nodes are created. On the docker daemon, pull the docker swarm visualizer and create a container for it which provides a graphical where we can see the nodes and what are the services running in the node. There are other tools also available for monitoring the swarm. Visualizer can be accessed through host ip address 8000.

Journal of Theoretical and Applied Information Technology
15th May 2022. Vol.100. No 9

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195



It can be visualized here with three nodes that are manager1, worker1 and worker2 and nothing can be seen as services are not yet created. New services can be created with the command `***docker service create***`. Service is created with the name nginx1_service and mapped to the port 8001 from the host and 80 from the container and the image in NGINX.



Services or containers can be created based on the requirement. Services will be executed on different nodes. Replica of a service can be created of same container. ` docker service ls` lists all services which are created.

```
docker@manager1:~$ docker service ps nginx2_service
ID             NAME              IMAGE          NODE      DES
IRENT STATE          ERROR          PORTS
xknz72bakerg   nginx2_service.1  nginx:latest   worker1   Run
ning 6 minutes ago
myxxzcwkp7sv   nginx2_service.2  nginx:latest   worker2   Run
ning 6 minutes ago
ejs3ig4go8tt   nginx2_service.3  nginx:latest   manager1  Run
ning 8 minutes ago
i5kh5gomdju3   nginx2_service.4  nginx:latest   worker1   Run
ning 6 minutes ago
```

The above example shows service nginx2_service running on different worker and manager nodes. By default docker has load balance the containers on different nodes. Another feature of docker is to scale the services. Command is used to scale up and scale down the containers.nginx2_service is scaled up from 4to 6.

```
docker@manager1:~$ docker service scale nginx2_service=6
nginx2_service scaled to 6
overall progress: 6 out of 6 tasks
1/6: running
2/6: running
3/6: running
4/6: running
5/6: running
6/6: running
verify: Service converged
docker@manager1:~$
```

Viualizer shows that nginx_2 service is increased and running on different swarm nodes.

The availability of node can be changed. That is, node can be drained using `drain` command. In the following example, worker2 node is in drain condition which makes the services running on worker node2 are migrated to other nodes.

```
docker@manager1:~$ docker node update --availability drain worker2
worker2
docker@manager1:~$ docker node ls
ID                            HOSTNAME        STATUS          AVAILABILITY     MANAGER
ON
qd6jx4m2f4f8ehqya6qhct5y7 *   manager1        Ready           Active           Leader
wqjjyvdmrnjquq0nr3qtc5mzy     worker1         Ready           Active
ja2z7bmsrh37jnkvy6drux4m0     worker2         Ready           Drain
docker@manager1:~$
```

The benefit of this if any updates need to be done on the server or nodes require any maintenance then that particular can be changed from availability to drain mode. Docker swarm automatically balance the load of containers running on one node to other nodes.

Journal of Theoretical and Applied Information Technology
15th May 2022. Vol.100. No 9

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

The above visualizer shows that worker2 node is drained and load is migrated to other manager and worker nodes.Nodes which are drained can also be made available again.

6 VIRTUALIZATION WITH CONTAINERS THAN VIRTUAL MACHINES:

Both containers and virtualization are destined with same objectives[36].Every Virtual machine has to execute with independent RAM,memory or other resources.Hence when the number of virtual machine increases then there can be possibility of draining of resources.This drawback can be overcome by solving through technology like containerization.One of the technology is docker which has concept of Operating system layers of abstarction which facilitates more number of containers with support of same underlying operating system.Starting an docker container is more fatser than strting a virtual machine container.containers provide an feature of suppleness and handiness for working with multiple cloud environment.containers have many advantages over virtual machines like development,deployment,etc.Containers are chosen in environment where huge number of applications need to be deployed with few number of servers[37].

6.1 Steps For Creating A Swarm In Docker Container:

Step1: Create a Docker Swarm nodes with one manager and required number of worker nodes.

Step2: check machine created successfully with ‘docker-machine ls’

Step 3: Use SSH command to connect to the docker machine

Step 4: Initialize the docker swarm

Step 5:Join workers in the swarm executing ‘docker swarm join’

Step 6: On manager node, execute and check for number of nodes connected.

Step 7: Execute the containers and check the services running on all the nodes.

Step 8: Scale the services up and down on manager node and each individual node can be inspected for workload.

Step 9: Nodes in the swarm can be drained using ‘docker node update’

Step 10: Service can be updated or removed from the docker machine

Container orchestration ensures that all containers are running on all systems and that the containers are scaled up or down depending on the load. As a single service, orchestration maintains and controls numerous Docker containers. Many tools are available, including Docker Swarm, Kubernetes, Apache Mesos, and others**.**

6.2 Deploying Docker Swarm Cluster:

Deployment file which contains following snippet:

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: nginx-deployment
spec:  selector:
   matchLabels:
    app: nginx
 replicas: 2
 template:
  metadata:
   labels:
    app: nginx
  spec:
   containers:
   - name: nginx
    image: nginx:1.17.3
    ports:
  - containerPort: 80
```

Logtsash is an server sidedata processing pipeline that ingests data from a multitude of source simultaneously transforms it.

```
logstash:
   image: docker.elastic.co/logstash/logstash:5.6.1
   Environment:
    - xpack.security.enabled=false
    - LOGSPOUT=ignore
   deploy:
    resources:
     reservations:
      memory: 400M
     limits:
      memory: 800M
    placement:
     constraints: [node.role == worker]
   configs:
    - source: logstash.conf
     target:
/usr/share/logstash/pipeline/logstash.conf
     uid: '1000'
     gid: '1000'
     mode: 0664
   command: logstash -f
/usr/share/logstash/pipeline/logstash.conf
```

Monitoring tool is used to retrieve the CPU usage, Memory usage and other alerts. Heap memory size has been set to maximum of 50% of the main memory. It indicates when the usage of CPU and memory is more than 80%. Logs are collected from different nodes using logging tool.

Docker Swarm and Docker:

This paper relies on Docker Swarm and docker. To replicate an application that is scaled, Docker Images are used[38]. In addition to running multiple instances simultaneously, it also runs other services like a web proxy and database using Docker Containers. Meanwhile, Docker Swarm enables services to communicate with each other and scale easily [39]. Docker also facilitates scaling to new machines. Docker Swarm and its concept of multiple managers can be used that decide where new containers are to be executed and when they are to be.

6.3 Scaling Of Containers:

A Docker Swarm can either comprise static or dynamic nodes in an auto-scaling algorithm. Nodes that are static won't be removed if your application scales down. Static nodes provide benefits that allocate nodes to swarm. Dynamic nodes can be taken away from the swarm or added. In the scenario of execution, three static nodes are used which will act as a NGINX server, node which executes scaling application and another as a Swarm manager.

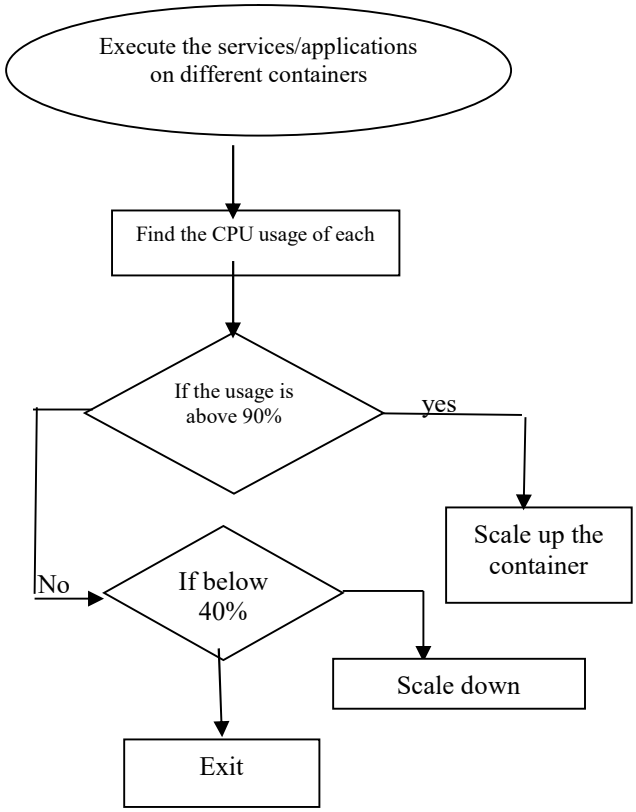


Fig4. Scaling up and down containers

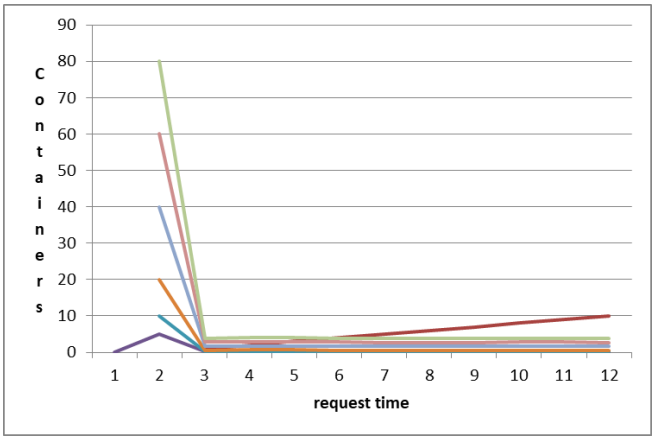


Fig 5. Request time for 1,2,…10 containers with request times between 10 and 80 per second

7 COMPARISON OF EXISTING APPROACH WITH PROPOSED METHOD:

Docker swarm with one manager provides limited functionality. High availability and fault recovery abilities are limited. With the use of multiple managers, Load balancing is done automatically. On every node, Docker Swarm's load balancer can distribute requests across all the containers on any host within the cluster. Without NGINX or NGINX Plus, the Swarm load balancer processes inbound client requests along with internal requests. Raft Consensus Algorithm is used to manage swarm state by swarm manager nodes. In Docker swarm mode, consensus is used to ensure that the manager nodes responsible for managing and scheduling tasks in the cluster are all storing the same consistent state. When the Leader Manager, who schedules tasks in the cluster, dies unexpectedly, any other Manager can take over task scheduling and rebalance tasks accordingly. Raft algorithm controls swarm up to (N-1)/2 failures of nodes

Some of the features which improve the performance of multi manager nodes are:

a. **High-Availability:** No outage or downtime is guaranteed.

b. Load Balancing: If a node fails in the cluster, automatically allocate resources and requests to other nodes.

C.De-centralized: The cluster is never dependent on a single manager node in a production environment because multiple nodes run the manager.

Swarm with three managers can indulge loss of at least one manager, Swarm with five managers can indulge failure of two nodes of manager and N manager cluster can stand (N-1)/2 managers.

7.1 Testing Fault Tolerance:

Setting up an environment by connecting two managers to execute the services/applications. Bringing down one manager will alert the manager2 as a leader which uses Raft algorithm for selecting the manager. If there are three managers, and one manager fails then remaining managers can lead the role. Credentials can be retrieved by adding nodes by using 'docker swarm join-token' which contains whether it's a manager or worker. All the containers on a swarm can be stopped and start executing on another swarm.

Steps to provide high availability in swarm cluster:

1. Static IP address has to be used for the swarm cluster nodes.

2. Managers which failed to be replaced at once.

3. Start the containers and services on respective leader nodes.

According to the size of the swarm cluster, following table shows how the instances available:

Swarm size	Fault tolerance
1	0
2	0
3	1
4	1
5	2

Table1: Swarm cluster

7.2 Raft Algorithm:

Docker in swarm mode uses Raft consensus algorithm for managing the nodes in a cluster. This is better than Paxos algorithm in terms of Latency. In a System of multiple processes (nodes), each of them will be a candidate, leader or follower. Initially each process will be a follower, if they are not selected then they can become candidate. Candidates in turn can requests for the votes from other process/nodes for becoming a leader. Median value differs by 1 microsecond between Paxos and Raft.

Timing requirement of Raft is calculated by:

Response time, MTBF (Mean Time between Failures) and election time out.

Method	Response time	Election time
Paxos	NA	NA
Raft	0.5ms to 20ms	10ms to 100ms

There is no election time and response time calculation for Paxos algorithm .It has only safety and fault tolerance properties.

8 CONCLUSION AND FUTURE WORK:

Two important technologies are in comparison based totally on few parameters like portability, security, and so forth. Both of those serve identical reason of virtualization. Boxes do not require many resources in comparison to virtual machines. Selecting considered one of them is the duty of a researcher based totally at the requirement of the studies. Non-stop integration and non-stop delivery gear automate the deployment with the aid of integrating the code. There are specific docker models available for variety of operating system.

Docker packing containers are brought asset to the present day technology. They're greater relaxed than virtual machines in deployment, checking out, and so on. Dockers plays quicker than digital machines as it makes use of less wide variety of sources and additionally guest running system does not exist. In addition, there may be a competition for all researchers in the region of strength efficiency in cloud and consolidating services with workloads.

Bins carry out higher to serve the quantity of requests and yield better throughput examine to virtual machines. Using bins has been growing dramatically inside the cloud computing community for providing cloud offerings in recent years. Consequently, green management of cloud assets at run time through field scheduling has emerge as of top importance in cloud computing. Docker is a project which brings the containers towards developers and system administrators. It comes with an smooth to apply interface for container management and presents a wholesome ecosystem for sharing the containerized applications images.

As a future work, energy management algorithms can be implemented using docker containers. Many models related to resource management can also be developed using multi managers.

REFERENCES:

[1] Borah, Amlan Deep, et al. "Power saving strategies in green cloud computing systems." International Journal of Grid Distribution Computing 8.1 (2015): 299-306.
[2] What Is Docker? Available online: https://www.docker.com/what-docker (accessed on 8 May 2018).
[3] Li, X., Garraghan, P., Jiang, X., Wu, Z., Xu, J., 2017. Holistic virtual machine scheduling in cloud datacenters towards minimizing total energy. IEEE Trans. Paral. Distrib. Syst. 29 (6), 1317–1331.
[4] Docker, “Swarm mode key concepts,” 2018. [Online]. Available: https://docs.docker.com/engine/swarm/key-concepts/. [Accessed: 30- Jun-2018].
[5] Z. Kozhirbayev and R. O. Sinnott, “A performance comparison of container-based technologies for the Cloud,” Future Gen eration Computer Systems, no. 68, pp. 175–182, March 2017, https://doi.org/10.1016/j.future.2016.08.0256.
[6] Zhang, Qi, et al. "A comparative study of containers and virtual machines in big data environment." 2018 IEEE 11th International Conference on Cloud Computing (CLOUD). IEEE, 2018.
[7] . Yadav, Anuj Kumar, and M. L. Garg. "Docker containers versus virtual machine-based virtualization." Emerging Technologies in Data Mining and Information Security. Springer, Singapore, 2019. 141-150.
[8] . Rad, Babak Bashari, Harrison John Bhatti, and Mohammad Ahmadi. "An introduction to docker and analysis of its performance." International Journal of Computer Science and Network Security (IJCSNS) 17.3 (2017): 228.
[9] . Kumar, Krishan, and Manish Kurhekar. "Economically efficient virtualization over cloud using docker containers." 2016 IEEE international conference on cloud computing in emerging markets (CCEM). IEEE, 2016.
[10] Chelliah, Pethuru, and Amit Mangalvedkar. "IEEE CCEM 2016 Program Details."
[11] Wan, Xili, et al. "Application deployment using Microservice and Docker containers: Framework and optimization." Journal of Network and Computer Applications 119 (2018): 97-109.
[12] Marathe, Nikhil, Ankita Gandhi, and Jaimeel M. Shah. "Docker swarm and kubernetes in cloud computing environment." 2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI). IEEE, 2019.
[13] Rodriguez, Maria A., and Rajkumar Buyya. "Container-based cluster orchestration systems: A taxonomy and future directions." Software: Practice and Experience 49.5 (2019): 698-719.
[14] https: //www.docker.com/products/docker-swarm
[15] Lijuan, Liu. "Research and implementation of Docker performance service in distributed platform." Int. J. Eng. Comput. Sci. 6.11 (2017).
[16] Jurenka, Vladimir. "Virtualization using Docker Platform." Faculty of Informatics Masaryk University (2015).
[17] Dziurzanski, Piotr, et al. "Scalable distributed evolutionary algorithm orchestration using Docker containers." Journal of Computational Science 40 (2020): 101069.
[18] Docker Swarm Strategies. https://docs.docker.com/swarm/scheduler/strategy/
[19] Wan, Xili, et al. "Application deployment using Microservice and Docker containers: Framework and optimization." Journal of

Network and Computer Applications 119 (2018): 97-109.
[20] Kovács, József, Péter Kacsuk, and Márk Emődi. "Deploying docker swarm cluster on hybrid clouds using occopus." Advances in Engineering Software 125 (2018): 136-145.
[21] https://geekflare.com/docker-architecture/
[22] da Silva Fré, Gabriel Lobão, et al. "Particle swarm optimization implementation for minimal transmission power providing a fully-connected cluster for the Internet of Things." 2015 International Workshop on Telecommunications (IWT). IEEE, 2015.
[23] Cito, Jürgen, Vincenzo Ferme, and Harald C. Gall. "Using docker containers to improve reproducibility in software and web engineering research." International Conference on Web Engineering. Springer, Cham, 2016.
[24] https://www.nginx.com/blog/deploying-nginx-nginx-plus-docker/
[25] da Silva Fré, Gabriel Lobão, et al. "Particle swarm optimization implementation for minimal transmission power providing a fully-connected cluster for the Internet of Things." 2015 International Workshop on Telecommunications (IWT). IEEE, 2015.
[26] Cito, Jürgen, Vincenzo Ferme, and Harald C. Gall. "Using docker containers to improve reproducibility in software and web engineering research." International Conference on Web Engineering. Springer, Cham, 2016.
[27] Meadusani, Srinath Reddy. "Virtualization Using Docker Containers: For Reproducible Environments and Containerized Applications." (2018).
[28] https://medium.com/@jessgreb01/digging-into-docker-layers
[29] https://www.aquasec.com/cloud-native-academy/docker-container/docker-swarm/
[30] Stelly, Christopher, and Vassil Roussev. "SCARF: A container-based approach to cloud-scale digital forensic processing." Digital Investigation 22 (2017): S39-S47.
[31] Wu, Shuangyan, et al. "Auto-scaling web application in docker based on gray prediction." 2018 International Conference on Network, Communication, Computer Engineering (NCCE 2018). Atlantis Press, 2018.
[32] https://docs.docker.com/engine/swarm/
[33] Ala'Anzy, Mohammed, and Mohamed Othman. "Load balancing and server consolidation in cloud computing environments: A meta-study." IEEE Access 7 (2019): 141868-141887.
[34] https://www.replex.io/blog/virtualization-vs-containers-scalability-portability-and-resource-utilization
[35] https://www.backblaze.com/blog/vm-vs-containers/
[36] Jurenka, Vladimir.(2015).Virtualization using Docker Platform.https://edisciplinas.usp.br/pluginfile.php/318402/course/section/93668/thesis_3.pdf
[37] J. Turnbull, "The Docker Book," 2014, https://dockerbook.com/#toc.
[38] Adolfsson, Henrik. "Comparison of Auto-Scaling Policies Using Docker Swarm." (2019).
[39] Ho, H., S. Fong, and Z. Yan. "user acceptance testing of mobile payment in various scenario. IEEE inetrnational conference on e-business engineering." *IEEE computer sciety* (2008).