Docker Container-Based Big Data Processing System in Multiple Clouds for Everyone

Nitin Naik
Defence School of Communications and Information Systems
Ministry of Defence, United Kingdom
Email: nitin.naik100@mod.gov.uk

Abstract*—Big data processing is progressively becoming essential for everyone to extract the meaningful information from their large volume of data irrespective of types of users and their application areas. Big data processing is a broad term and includes several operations such as the storage, cleaning, organization, modelling, analysis and presentation of data at a scale and efficiency. For ordinary users, the significant challenges are the requirement of the powerful data processing system and its provisioning, installation of complex big data analytics and difficulty in their usage. Docker is a container-based virtualization technology and it has recently introduced Docker Swarm for the development of various types of multi-cloud distributed systems, which can be helpful in solving all above problems for ordinary users. However, Docker is predominantly used in the software development industry, and less focus is given to the data processing aspect of this container-based technology. Therefore, this paper proposes the Docker container-based big data processing system in multiple clouds for everyone, which explores another potential dimension of Docker for big data analysis. This Docker container-based system is an inexpensive and user-friendly framework for everyone who has the knowledge of basic IT skills. Additionally, it can be easily developed on a single machine, multiple machines or multiple clouds. This paper demonstrates the architectural design and simulated development of the proposed Docker container-based big data processing system in multiple clouds. Subsequently, it illustrates the automated provisioning of big data clusters using two popular big data analytics, Hadoop and Pachyderm (without Hadoop) including the Web-based GUI interface *Hue* for easy data processing in Hadoop.

***Keywords*—Docker Container; Docker Swarm; Big Data Processing System; Cloud; Hadoop; Hue; Pachyderm**

I. Introduction

Big data is high-volume, high-velocity and/or high-variety information assets that cannot be handled and processed by using the traditional IT infrastructure and tools [1]. Earlier, it was the requirement of major businesses and organisations but due to the rapid growth of data, ordinary users are looking to use big data processing options for their large volume of data, which cannot be processed by using traditional IT infrastructure [2]. For ordinary users, the significant challenges are the requirement of the powerful data processing system and its provisioning, installation of complex big data analytics and difficulty in their usage. Therefore, they require an economical, user-friendly, easy to design and develop data processing system.

Cloud-based big data processing systems are the most efficient and established infrastructure to fulfil the big data analysis requirements. Now, most businesses and users are shifting towards multi-cloud infrastructure for reducing their vendor dependent risk and achieving best services and resources for performance optimisation [3], [4]. Virtualization is one of the key technologies of cloud computing, and most cloud-based systems are based on virtualization. However, the requirement of significant and redundant resources, issues of interoperability and deployment, load balancing and migration complexities make it unattractive for various types of big data analyses for ordinary users [5]. Docker is a container-based virtualization technology and it has recently introduced Docker Swarm for the development of various types of multi-cloud distributed systems, which can be helpful in solving all above problems related to big data analysis for ordinary users [5], [6], [7]. However, Docker is predominantly used in the software development industry, and less focus is given to the data processing aspect of this container-based technology.

This paper proposes the Docker container-based big data processing system in multiple clouds for everyone, which explores another potential dimension of Docker for big data analysis. This Docker container-based system is an inexpensive and user-friendly framework for everyone who has knowledge of basic IT skills. Additionally, it can be easily developed on a single machine, multiple machines or multiple clouds. This paper demonstrates the architectural design and simulated development of the proposed Docker-based big data processing system in multiple clouds. This simulation of the big data processing system is based on a single machine consisting of Docker Swarm, VirtualBox and Mac OS X. However, the same big data processing system can be easily created on any of the Docker supported cloud by just selecting the appropriate driver name (see Fig. 6) such as Amazon Web Services, Microsoft Azure, Digital Ocean, Google Compute Engine, Exoscale, Generic, OpenStack, Rackspace, IBM Softlayer, VMware vCloud Air [8]. Before developing this system on the above clouds, it must require a valid subscription on those clouds. Subsequently, this paper illustrates the easy and automated provisioning of big data clusters using two popular big data analytics, Hadoop and Pachyderm (without Hadoop) including the Web-based GUI interface *Hue* for easy data processing in Hadoop. The automated provisioning of Hadoop cluster is demonstrated using Apache Ambari and SequenceIQ Ambari Shell. Whereas, the complete installation of Pachyderm requires four components: Docker, Kubectl (Kubernetes CLI), Pachyderm Command Line Interface and FUSE (optional). This proposed Docker-based system is also a dependable system due to the implicit support of dependability in Docker Swarm [9].

The remainder of this paper is organised as follows: Section II explains the theoretical background of big data analysis, Docker Container and Docker Swarm; Section III illustrates the architectural design of a Docker-based big data processing system in multiple clouds; Section IV presents the simulated development of this Docker-based big data processing system; Section V illustrates the provisioning of two popular big data analytics for the Docker platform, Hadoop and Pachyderm (without Hadoop); Section VI concludes the paper and suggests some future areas of extension.

II. Theoretical Background

A. Big Data Analysis

Big data analysis is the process of mining and extracting meaningful patterns from massive input data for decision making, prediction, and other inferencing [2], [10]. Traditional data analysis is the process of applying standard statistical methods such as factor analysis, cluster analysis, correlation analysis, and regression analysis to explore the cleaned first-hand data of limited amount [11]. This analysis is usually limited to testing a small number of hypotheses that we define well before the data collection [12]. However, big data analysis can be based on traditional statistical methods or enhanced computational models and is used to analyse unstructured and unclean data of massive amount. A big data analytic is not a single tool/technology but a combination of multiple tools/technologies that are combined as a system/platform/framework and used to perform various operations in the entire big data analysis process such as data collection, data cleaning, data modelling and visual interpretation of data [13].

B. Docker Container and Docker Swarm

Containerization or container-based virtualization is moderately different technique from virtualization, where an isolated environment (container) is created similar to a virtual machine (VM) but without virtual hardware emulation [3]. A container is a very old technique in Unix and Linux but now it is reintroduced at commercial level due to its benefits as compared to a VM. Containerization can be considered as an OS-level virtualization because containers are created in the user space above the OS kernel [14]. Multiple containers can be created in multiple user spaces on a single host but with very fewer resources than VMs [14].

Docker container is an instance of containerization. Docker is a container-based technology for an easy and automated creation, deployment and execution of applications by employing containers [14]. It facilitates an isolated environment (container) similar to a VM but without having its own OS, therefore all containers share the same OS kernel via Docker Engine as shown in Fig. 1. However, a container consists of all the binary and library files required to run an application. If Docker container is used on Linux then Linux OS act as a default Docker Host, but when it is used on a non-Linux machine then this Docker Host needs to be installed separately as shown in Fig. 2. This Docker Host is a lightweight VM and needs minimum resources as compared to the actual VM in Virtualization. Docker has given the name *default* to this Docker Host because it comes with the default installation and requires to run the Docker Engine.

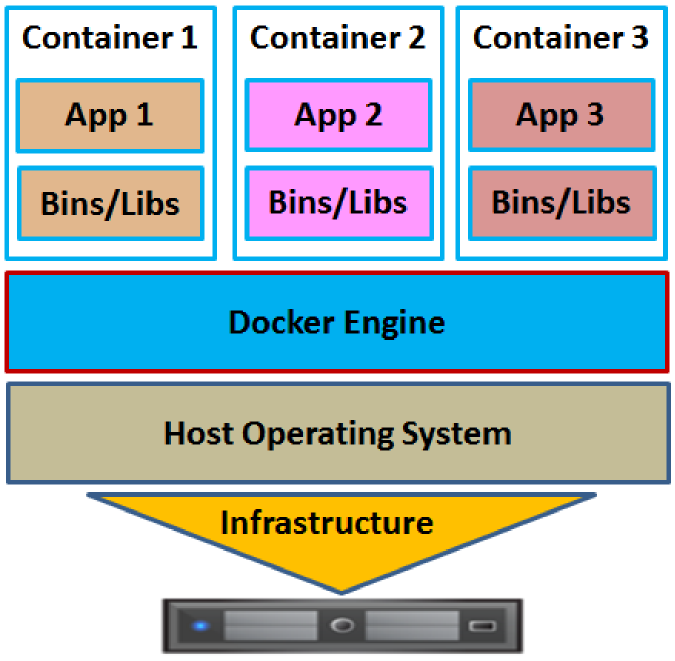


Fig. 1. Docker Containers on Linux Host

Docker Swarm is a cluster management and orchestration tool that connects and controls several Docker nodes to form a single virtual system [15]. It is an enhancement of Docker container technology for designing distributed systems in multiple clouds. It offers several unique features to the Swarm cluster such as availability, reliability, security, maintainability and scalability, which is an added advantage to the standard container technology and makes the container-based system a dependable system.

III. Architectural Design of a Docker Container-based Big Data Processing System

Fig. 4 shows the architectural design of Docker container-based big data processing system (including 3 Managers and 2 Workers) on multiple Docker supported clouds [8]. In Docker Swarm, the manager is responsible for the entire cluster and manages the resources of multiple Docker hosts at scale [16]. Managers are responsible for orchestrating the cluster, serving the Service API, scheduling tasks (containers) and addressing containers that have failed health checks [17]. A primary manager (leader) is the main point of contact within the Docker Swarm cluster. In Docker Swarm, there could be one primary manager (leader) and multiple secondary managers (reachable managers) in case the primary manager fails [16]. Primary manager works as a leader of the system and all the secondary managers contact with it regarding services and information. It is also possible to talk to secondary managers (replica instances) that will act as backups. However, all requests issued on a secondary manager are automatically proxied to the primary manager. If the primary manager fails, a secondary manager takes away the lead. Therefore, it facilitates a highly available and reliable cluster [16]. Worker nodes serve only simpler functions such as executing the tasks to spawn containers and routing data traffic intended for specific containers [17]. The complete breakdown and workflow are

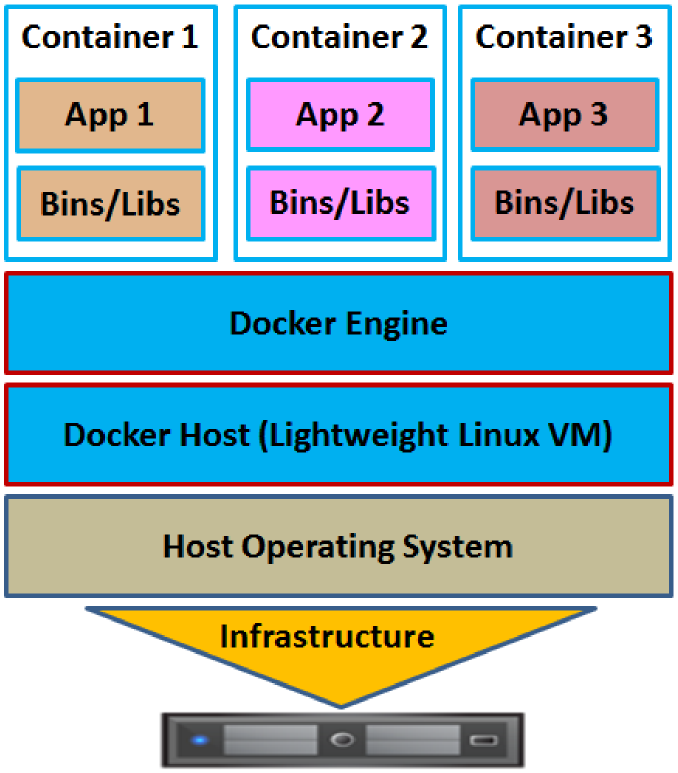
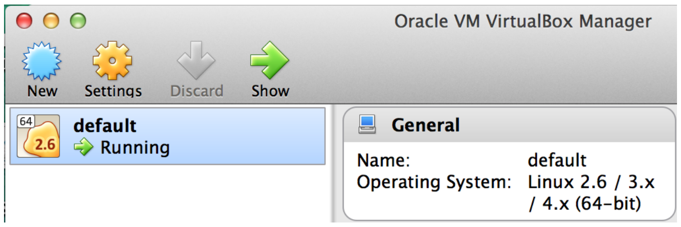


Fig. 2. Docker Containers on Non-Linux Host require an additional Docker Host (lightweight Virtual Machine) component



default
Running

Name: default
Operating System: Linux 2.6 / 3.x / 4.x (64-bit)

Fig. 3. Docker Host "default" (lightweight Virtual Machine) in VirtualBox

shown in Fig. 5. Secure connectivity of Docker Swarm nodes across multiple clouds can be provided using identity and access management protocols [18], [19], [20], [21], which is not covered in this paper.

IV. Simulated Development of a Docker Container-based Big Data Processing System

This section demonstrates the simulated development of a Docker container-based big data processing system into two steps: building a Docker cluster for the processing of data and building data volume containers for the management of data.

A. Building a Docker Cluster for the Processing of Data

This experimental simulation of the big data processing system is based on Docker Swarm, VirtualBox and Mac OS X. Here, the big data system is developed as a cluster of five Swarm Nodes (3 Managers and 2 Workers) by creating five lightweight VMs in VirtualBox on the same host computer (Mac OS X) as shown in Fig. 6. However, all these lightweight VMs and, subsequently, Docker Swarm Nodes

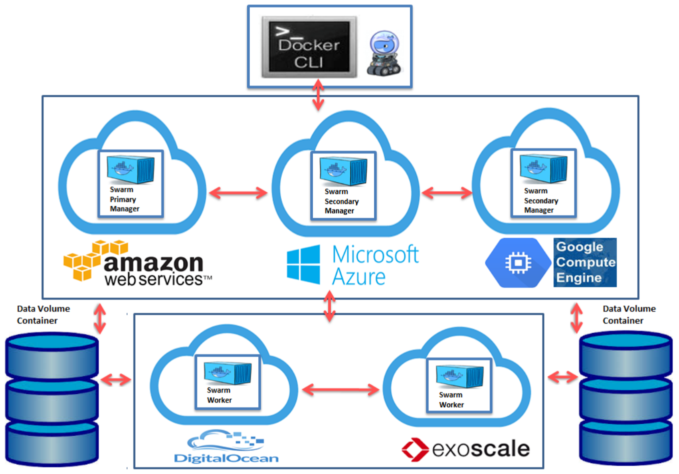


Fig. 4. Architectural Design of Docker Container-based Big Data Processing System in Multiple Clouds

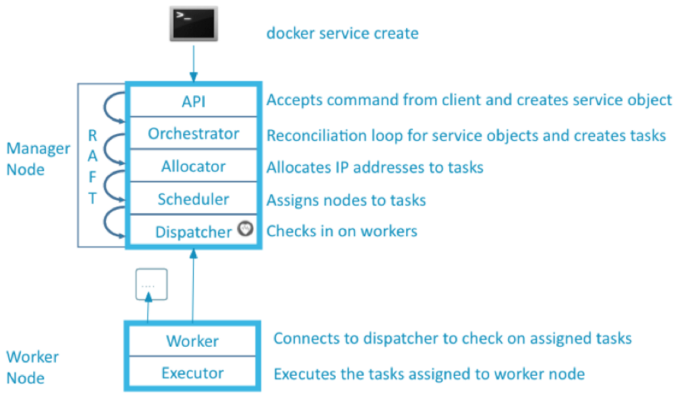
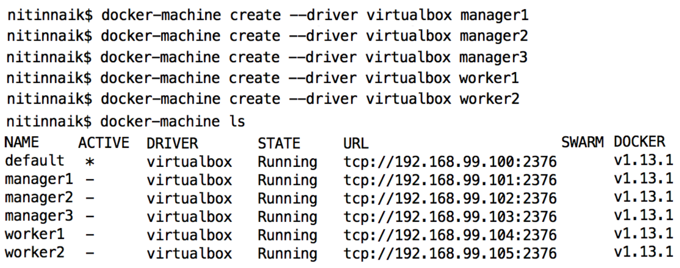

docker service create
Manager Node
R A F T
API
Accepts command from client and creates service object
Orchestrator
Reconciliation loop for service objects and creates tasks
Allocator
Allocates IP addresses to tasks
Scheduler
Assigns nodes to tasks
Dispatcher
Checks in on workers
Worker Node
Worker
Connects to dispatcher to check on assigned tasks
Executor
Executes the tasks assigned to worker node

Fig. 5. Docker Swarm Node Breakdown and Workflow [17]

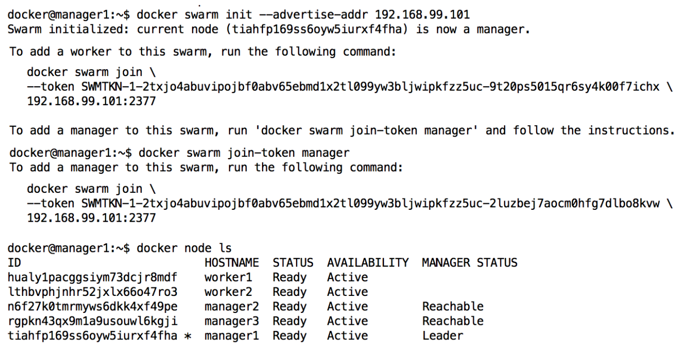
can be created on different clouds (shown in Fig. 4) by just changing the driver name from *–driver virtualbox* to *–driver amazonec2/azure/google/digitalocean/exoscale* in Fig. 6. The only requirement for doing this is to have a valid subscription account on the desired cloud. Fig. 6 shows the process of creation of five Docker Machines (lightweight VMs) with different private IP addresses and standard Docker Port 2376 using the most recent version *v1.13.1* of Docker at the time of this experiment.

Subsequently, these Docker Machines (lightweight VMs) are used to create a cluster of five Docker Swarm Nodes,



```
nitinnaik$ docker-machine create --driver virtualbox manager1
nitinnaik$ docker-machine create --driver virtualbox manager2
nitinnaik$ docker-machine create --driver virtualbox manager3
nitinnaik$ docker-machine create --driver virtualbox worker1
nitinnaik$ docker-machine create --driver virtualbox worker2
nitinnaik$ docker-machine ls
NAME      ACTIVE   DRIVER       STATE     URL                         SWARM   DOCKER
default    *       virtualbox   Running   tcp://192.168.99.100:2376           v1.13.1
manager1   -       virtualbox   Running   tcp://192.168.99.101:2376           v1.13.1
manager2   -       virtualbox   Running   tcp://192.168.99.102:2376           v1.13.1
manager3   -       virtualbox   Running   tcp://192.168.99.103:2376           v1.13.1
worker1    -       virtualbox   Running   tcp://192.168.99.104:2376           v1.13.1
worker2    -       virtualbox   Running   tcp://192.168.99.105:2376           v1.13.1
```

Fig. 6. Creating Docker lightweight Virtual Machines (VMs) for building a Swarm Cluster



```
docker@manager1:~$ docker swarm init --advertise-addr 192.168.99.101
Swarm initialized: current node (tiahfp169ss6oyw5iurxf4fha) is now a manager.

To add a worker to this swarm, run the following command:

   docker swarm join \
   --token SWMTKN-1-2txjo4abuvipojbf0abv65ebmd1x2tl099yw3bljwipkfzz5uc-9t20ps5015qr6sy4k00f7ichx \
   192.168.99.101:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.

docker@manager1:~$ docker swarm join-token manager
To add a manager to this swarm, run the following command:

   docker swarm join \
   --token SWMTKN-1-2txjo4abuvipojbf0abv65ebmd1x2tl099yw3bljwipkfzz5uc-2luzbej7aocm0hfg7dlbo8kvw \
   192.168.99.101:2377

docker@manager1:~$ docker node ls
ID                           HOSTNAME  STATUS  AVAILABILITY  MANAGER STATUS
hualy1pacggsiym73dcjr8mdf    worker1   Ready   Active
lthbvphjnhr52jxlx66o47ro3    worker2   Ready   Active
n6f27k0tmrmyws6dkk4xf49pe    manager2  Ready   Active        Reachable
rgpkn43qx9m1a9usouwl6kgji    manager3  Ready   Active        Reachable
tiahfp169ss6oyw5iurxf4fha *  manager1  Ready   Active        Leader
```

Fig. 7. Creating Docker Swarm cluster with 3 Manager and 2 Worker Nodes on above Docker lightweight VMs

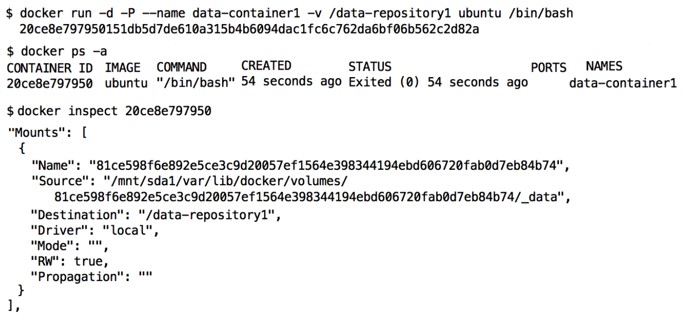
where 3 Swarm Managers are created on manager1, manager2 and manager3 Docker Machines; and 2 Swarm Workers are created on worker1 and worker2 Docker Machines as shown in Fig. 7. The manager1 is the primary manager (leader) as it is created first but this can be easily changed and reassigned. When the node is assigned the responsibility of a manager, it joins a *RAFT Consensus group* to share information and perform leadership election. The leader is the primary manager that maintains the state, which includes lists of nodes, services and tasks across the swarm in addition to making scheduling decisions [17]. This state is circulated across the each manager node through a built-in RAFT store. Consequently, managers have no dependency on an external key-value store such as *etcd or Consul*. Non-leader managers function as hot spares and forward API requests to the current elected leader [17]. In a Docker Swarm cluster, all the commands should be run on the manager's node.

B. Building Data Volume Containers for the Management of Data

This subsection presents the process of creation of data volumes/data volume containers to manage data inside and between Docker containers. A data volume is a specially-designated directory within one or more containers that bypasses the Union File System (UFS). Data volumes are designed to persist data, independent of the container's life cycle. Therefore, Docker never automatically deletes these volumes when the user removes a container, nor the garbage collector removes volumes that are no longer referenced by a container [22]. Here, two data volume containers are created for the previous Docker Swarm cluster, which can be shared among all the nodes of the cluster. There are two different ways to create data volume containers, which are demonstrated in Figs. 8 and 9.

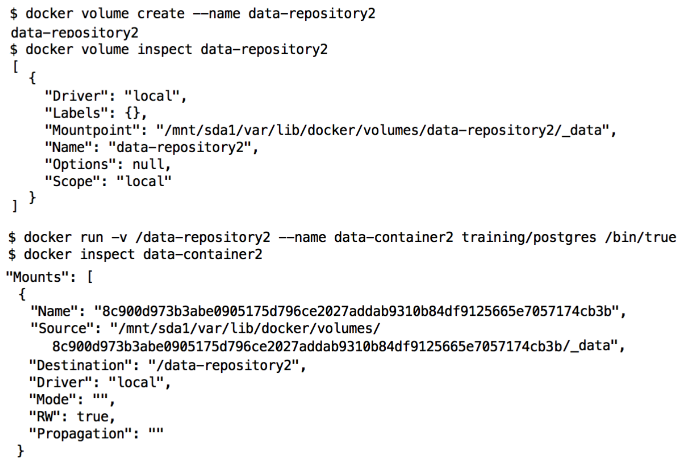
V. Big Data Analytics (with and without Hadoop) on Docker-based Big Data Processing Plateform for Everyone

This section illustrates the automated provisioning of big data clusters using two popular big data analytics, Hadoop and Pachyderm (without Hadoop). The automatic provisioning of Hadoop cluster is demonstrated using Apache Ambari and SequenceIQ Ambari Shell. Whereas, the complete installation of Pachyderm requires four components: Docker, Kubectl



```
$ docker run -d -P --name data-container1 -v /data-repository1 ubuntu /bin/bash
  20ce8e797950151db5d7de610a315b4b6094dac1fc6c762da6bf06b562c2d82a

$ docker ps -a
CONTAINER ID  IMAGE   COMMAND     CREATED          STATUS                             PORTS   NAMES
20ce8e797950  ubuntu  "/bin/bash" 54 seconds ago Exited (0) 54 seconds ago              data-container1

$ docker inspect 20ce8e797950
"Mounts": [
  {
    "Name": "81ce598f6e892e5ce3c9d20057ef1564e398344194ebd606720fab0d7eb84b74",
    "Source": "/mnt/sda1/var/lib/docker/volumes/
       81ce598f6e892e5ce3c9d20057ef1564e398344194ebd606720fab0d7eb84b74/_data",
    "Destination": "/data-repository1",
    "Driver": "local",
    "Mode": "",
    "RW": true,
    "Propagation": ""
  }
],
```

Fig. 8. Creating Data Volume1 with Container1 and their descriptions



```
$ docker volume create --name data-repository2
data-repository2
$ docker volume inspect data-repository2
[
  {
     "Driver": "local",
     "Labels": {},
     "Mountpoint": "/mnt/sda1/var/lib/docker/volumes/data-repository2/_data",
     "Name": "data-repository2",
     "Options": null,
     "Scope": "local"
  }
]

$ docker run -v /data-repository2 --name data-container2 training/postgres /bin/true
$ docker inspect data-container2
"Mounts": [
  {
    "Name": "8c900d973b3abe0905175d796ce2027addab9310b84df9125665e7057174cb3b",
    "Source": "/mnt/sda1/var/lib/docker/volumes/
       8c900d973b3abe0905175d796ce2027addab9310b84df9125665e7057174cb3b/_data",
    "Destination": "/data-repository2",
    "Driver": "local",
    "Mode": "",
    "RW": true,
    "Propagation": ""
  }
```

Fig. 9. Creating Data Volume2 and later linking with Container2

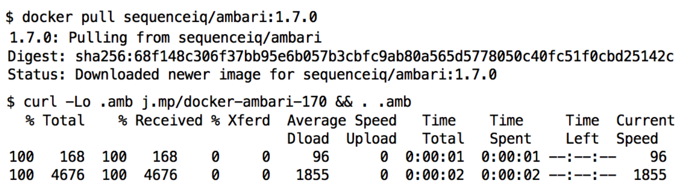
(Kubernetes CLI), Pachyderm Command Line Interface and FUSE (optional).

A. Hadoop on Docker for Everyone: Automated Provisioning of a Hadoop Cluster using Apache Ambari and SequenceIQ Ambari Shell

This subsection illustrates the easy provisioning of a Hadoop cluster on Docker, which avoids the tedious installation of every single component from the Hadoop Ecosystem. Once it is created, it can be used any number of times and anywhere; additionally, it offers consistent data processing, development, testing, integration and deployment functionalities. The Apache Ambari project is developed for making Hadoop management simpler by developing software for provisioning, managing, and monitoring of Apache Hadoop clusters. Ambari provides an intuitive, easy-to-use Hadoop management Web UI backed by its RESTful APIs [23].

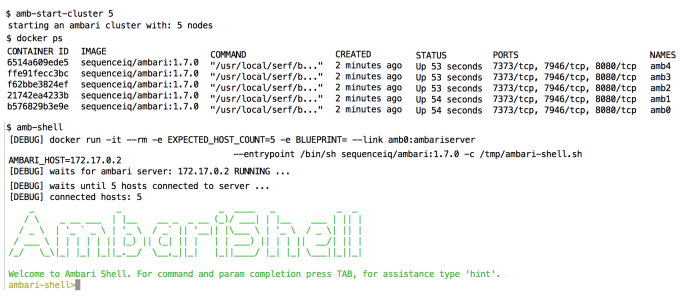
For the easy provisioning of Hadoop cluster, Ambari image and function can be downloaded from the SequenceIQ cloud repository as shown in Fig. 10. After obtaining the Ambari image for Docker, an Ambari cluster can be created as shown in Fig. 11. Here, a 5 node Ambari cluster is created in which all the containers are preconfigured and the Ambari agents are running. SequenceIQ has made Hadoop provisioning process relatively simple by introducing Ambari Shell (Apache Ambari + Spring Shell) [24], which can be started as shown in Fig. 11.

In this Ambari shell, Hadoop (HDP) cluster (see Fig. 12)



```
$ docker pull sequenceiq/ambari:1.7.0
 1.7.0: Pulling from sequenceiq/ambari
Digest: sha256:68f148c306f37bb95e6b057b3cbfc9ab80a565d5778050c40fc51f0cbd25142c
Status: Downloaded newer image for sequenceiq/ambari:1.7.0

$ curl -Lo .amb j.mp/docker-ambari-170 && . .amb
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   168  100   168    0     0     96      0  0:00:01  0:00:01 --:--:--    96
100  4676  100  4676    0     0   1855      0  0:00:02  0:00:02 --:--:--  1855
```

Fig. 10. Downloading Ambari Image and Function from Cloud Repository



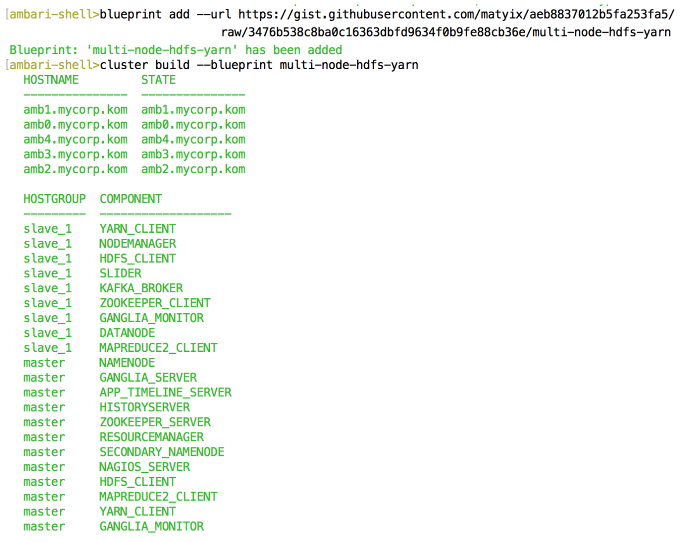
```
$ amb-start-cluster 5
 starting an ambari cluster with: 5 nodes
$ docker ps
CONTAINER ID   IMAGE                     COMMAND                   CREATED          STATUS          PORTS                              NAMES
6514a609ede5   sequenceiq/ambari:1.7.0   "/usr/local/serf/b..."   2 minutes ago    Up 53 seconds   7373/tcp, 7946/tcp, 8080/tcp       amb4
ffe91fecc3bc   sequenceiq/ambari:1.7.0   "/usr/local/serf/b..."   2 minutes ago    Up 53 seconds   7373/tcp, 7946/tcp, 8080/tcp       amb3
f62bbe3824ef   sequenceiq/ambari:1.7.0   "/usr/local/serf/b..."   2 minutes ago    Up 53 seconds   7373/tcp, 7946/tcp, 8080/tcp       amb2
21742ea4233b   sequenceiq/ambari:1.7.0   "/usr/local/serf/b..."   2 minutes ago    Up 54 seconds   7373/tcp, 7946/tcp, 8080/tcp       amb1
b576829b3e9e   sequenceiq/ambari:1.7.0   "/usr/local/serf/b..."   2 minutes ago    Up 54 seconds   7373/tcp, 7946/tcp, 8080/tcp       amb0

$ amb-shell
 [DEBUG] docker run -it --rm -e EXPECTED_HOST_COUNT=5 -e BLUEPRINT= --link amb0:ambariserver
                                --entrypoint /bin/sh sequenceiq/ambari:1.7.0 -c /tmp/ambari-shell.sh
 AMBARI_HOST=172.17.0.2
 [DEBUG] waits for ambari server: 172.17.0.2 RUNNING ...
 [DEBUG] waits until 5 hosts connected to server ...
 [DEBUG] connected hosts: 5

 Welcome to Ambari Shell. For command and param completion press TAB, for assistance type 'hint'.
 ambari-shell>
```

Fig. 11. Creating Ambari Cluster of 5 Nodes and Starting Ambari Shell

can be easily created by anyone. Here Hadoop cluster is created in the easiest way by using Blueprint as shown in Fig. 12 [25]. Finally, Hadoop cluster of 5 nodes is created as shown in Fig. 13, where the master and slaves are assigned automatically. After the installation of Hadoop Cluster, it can be monitored by using *Ganglia and Nagios*.

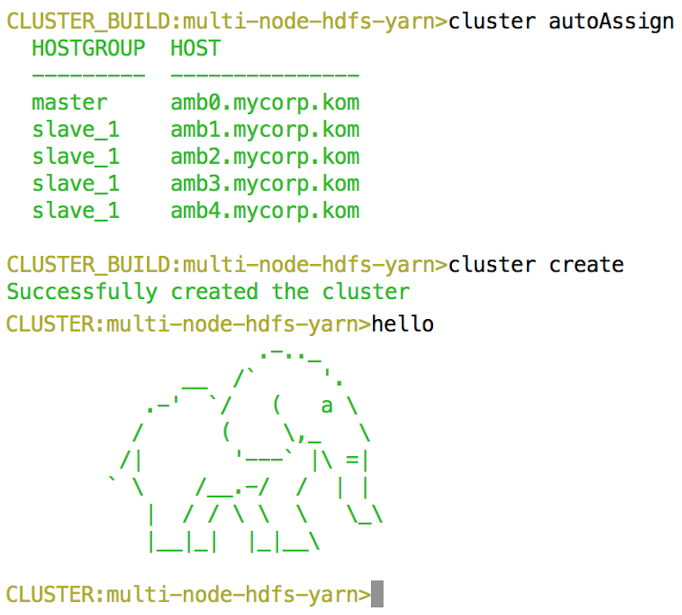
B. GUI Hadoop on Docker for Everyone: Big Data Processing in Hadoop from the Browser by using Hue (Hadoop User Experience) Web Interface

Hue is an open-source Web interface (lightweight Web server) that supports Apache Hadoop and its ecosystem, and offers user-friendly Hadoop big data processing facility directly from the Web browser to a non-technical user [26]. Hue is simply a view on top of any Hadoop distribution and can be installed on any machine. The easiest way to install and use Hue is by using Docker, which offers several benefits to users such as a lighter and more robust system than a VM, gives administrative permissions and quick starting of data processing with much faster execution.



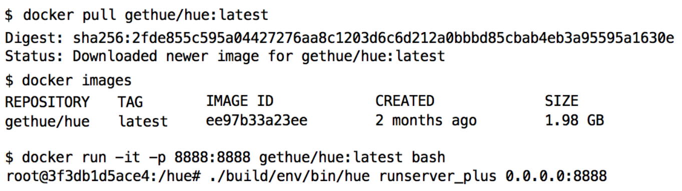
```
ambari-shell>blueprint add --url https://gist.githubusercontent.com/matyix/aeb8837012b5fa253fa5/
                                raw/3476b538c8ba0c16363dbfd9634f0b9fe88cb36e/multi-node-hdfs-yarn
 Blueprint: 'multi-node-hdfs-yarn' has been added
ambari-shell>cluster build --blueprint multi-node-hdfs-yarn
  HOSTNAME         STATE
  ---------------  ---------------
  amb1.mycorp.kom  amb1.mycorp.kom
  amb0.mycorp.kom  amb0.mycorp.kom
  amb4.mycorp.kom  amb4.mycorp.kom
  amb3.mycorp.kom  amb3.mycorp.kom
  amb2.mycorp.kom  amb2.mycorp.kom

  HOSTGROUP  COMPONENT
  ---------  --------------------
  slave_1    YARN_CLIENT
  slave_1    NODEMANAGER
  slave_1    HDFS_CLIENT
  slave_1    SLIDER
  slave_1    KAFKA_BROKER
  slave_1    ZOOKEEPER_CLIENT
  slave_1    GANGLIA_MONITOR
  slave_1    DATANODE
  slave_1    MAPREDUCE2_CLIENT
  master     NAMENODE
  master     GANGLIA_SERVER
  master     APP_TIMELINE_SERVER
  master     HISTORYSERVER
  master     ZOOKEEPER_SERVER
  master     RESOURCEMANAGER
  master     SECONDARY_NAMENODE
  master     NAGIOS_SERVER
  master     HDFS_CLIENT
  master     MAPREDUCE2_CLIENT
  master     YARN_CLIENT
  master     GANGLIA_MONITOR
```

Fig. 12. Using Blueprint for the Creation of Hadoop Cluster



```
CLUSTER_BUILD:multi-node-hdfs-yarn>cluster autoAssign
  HOSTGROUP  HOST
  ---------  ---------------
  master     amb0.mycorp.kom
  slave_1    amb1.mycorp.kom
  slave_1    amb2.mycorp.kom
  slave_1    amb3.mycorp.kom
  slave_1    amb4.mycorp.kom

CLUSTER_BUILD:multi-node-hdfs-yarn>cluster create
Successfully created the cluster
CLUSTER:multi-node-hdfs-yarn>hello
                   .-.._
             __  /`     '.
          .-'  `/   (   a \
         /      (    \,_   \
        /|       '---` |\ =|
       ` \    /__.-/  /  | |
          |  / / \ \  \   \_\
          |__|_|  |_|__\

CLUSTER:multi-node-hdfs-yarn>
```

Fig. 13. Creating Hadoop Cluster of 5 Nodes and automatically assigning a Master and Slaves



```
$ docker pull gethue/hue:latest
Digest: sha256:2fde855c595a04427276aa8c1203d6c6d212a0bbbd85cbab4eb3a95595a1630e
Status: Downloaded newer image for gethue/hue:latest
$ docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
gethue/hue    latest    ee97b33a23ee   2 months ago   1.98 GB

$ docker run -it -p 8888:8888 gethue/hue:latest bash
root@3f3db1d5ace4:/hue# ./build/env/bin/hue runserver_plus 0.0.0.0:8888
```

Fig. 14. Downloading Hue Image from Cloud Repository and starting its development version in Docker Container as an administrator

Hue can be easily configured in Docker container by pulling the latest Hue image from the Hue repository as shown in Fig. 14. It can be started in Docker container as a bash to the root of the project as shown in Fig. 14. The last command in Fig. 14 will start the development version of Hue.

Hue graphical experience can be started in the browser by using localhost (http://192.168.99.100:8888), which is usually the default Docker IP 192.168.99.100 on the port 8888. Hue offers all the most important big data processing tool related to Hadoop as shown in Fig. 15. Here, an ordinary user can start big data processing without knowing the complexities of all these tools.

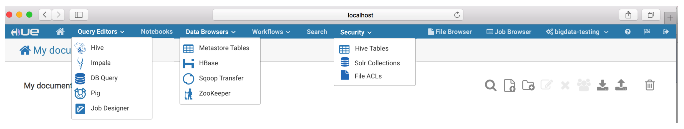


Fig. 15. Web-based GUI Dashboard of Hue and its available Hadoop Big Data Processing Tools

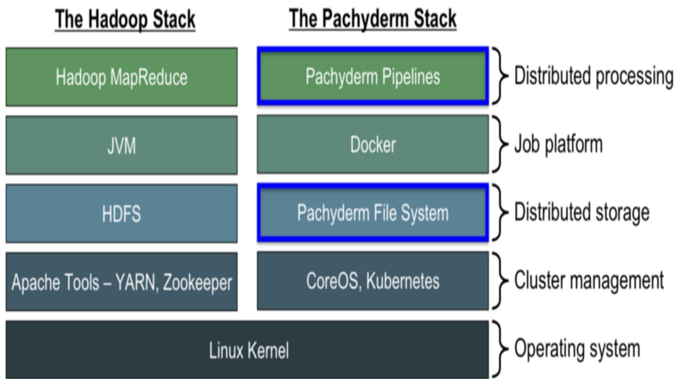
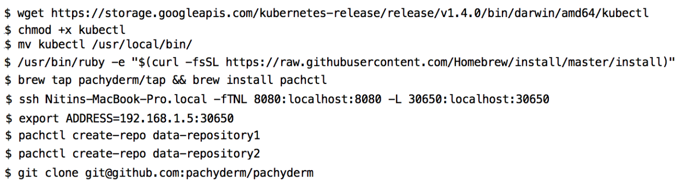


Fig. 16. Comparison between Hadoop Stack and Pachyderm Stack [27]



```
$ wget https://storage.googleapis.com/kubernetes-release/release/v1.4.0/bin/darwin/amd64/kubectl
$ chmod +x kubectl
$ mv kubectl /usr/local/bin/
$ /usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"
$ brew tap pachyderm/tap && brew install pachctl
$ ssh Nitins-MacBook-Pro.local -fTNL 8080:localhost:8080 -L 30650:localhost:30650
$ export ADDRESS=192.168.1.5:30650
$ pachctl create-repo data-repository1
$ pachctl create-repo data-repository2
$ git clone git@github.com:pachyderm/pachyderm
```

Fig. 17. Installation procedure of Pachyderm and Kubernetes on Docker

C. Non-Hadoop Data Analytic on Docker for Everyone: Pachyderm - A Docker Container-based Data Analytic

Pachyderm is another elephant in the room when it comes to big data analytics due to the weaknesses of Hadoop. In Hadoop, MapReduce jobs are specified as Java classes, which requires specialist Java programmers who write MapReduce jobs or hiring a third party such as *Cloudera* but this is difficult for ordinary users [27]. This typically means that big data initiatives require a lot of coordination internally and require resources that are beyond the reach of even large enterprises who do not have that kind of expertise. Therefore, most ordinary users want big data processing without incurring the complexity of Hadoop and MapReduce, and one of the solutions is a Pachyderm data analytic tool. Pachyderm allows programmers to implement an HTTP server inside a Docker container, then use Pachyderm to distribute the job [28], [29]. This has the potential to allow *sysadmins* to run large scale MapReduce jobs swiftly and easily to make product level decisions, without the knowledge of MapReduce [30]. The Pachyderm stack uses Docker containers, CoreOS and Kubernetes for cluster management. It replaces Hadoop file system HDFS with its file system called PFS (Pachyderm File System) and Hadoop processing framework MapReduce with its processing framework called Pachyderm Pipelines as shown in Fig. 16 [31]. The core features of Pachyderm are reproducibility, data provenance and, most importantly, collaboration, which has been missing from the big data world (Hadoop). Pachyderm is a promising big data analytic and has an intention of replacing Hadoop completely. The complete installation of Pachyderm requires four components: Docker, Kubectl (Kubernetes CLI), Pachyderm Command Line Interface and FUSE (optional). This installation procedure is also dependent on the operating system employed, therefore, this installation procedure is based on OS X, which is shown in Fig. 17.

VI. Conclusion

This paper proposed the Docker container-based big data processing system in multiple clouds for everyone, which explored another potential dimension of Docker. It demonstrated the architectural design and simulated development of the proposed Docker container-based big data processing system in multiple clouds. Subsequently, it illustrated the automated provisioning of big data clusters using two popular big data analytics, Hadoop and Pachyderm (without Hadoop) including the Web-based GUI interface *Hue* for easy data processing in Hadoop. This Docker container-based big data processing system is an inexpensive and user-friendly framework for everyone who has the knowledge of basic IT skills. Additionally, it can be easily developed on a single machine, multiple machines or multiple clouds. This proposed framework showed that Docker has a potential to develop a big data processing system for everyone. However, it is a new approach and still in the early stage of development with the support of only a few selected cloud service providers. Thus, it requires further testing and refinement to become a mature technique and wider acceptance in the cloud industry. In the future, it may be worthwhile to develop and evaluate this simulated big data processing system in multiple clouds.

References

[1] D. Fisher, R. DeLine, M. Czerwinski, and S. Drucker, “Interactions with big data analytics,” *Interactions*, vol. 19, no. 3, pp. 50–59, 2012.

[2] N. Naik, P. Jenkins, N. Savage, and V. Katos, “Big data security analysis approach using computational intelligence techniques in R for desktop users,” in *IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2016.

[3] N. Naik, “Building a virtual system of systems using Docker Swarm in multiple clouds,” in *IEEE International Symposium on Systems Engineering (ISSE)*. IEEE, 2016.

[4] ——, “Connecting Google cloud system with organizational systems for effortless data analysis by anyone, anytime, anywhere,” in *IEEE International Symposium on Systems Engineering (ISSE)*. IEEE, 2016.

[5] ——, “Migrating from Virtualization to Dockerization in the cloud: Simulation and evaluation of distributed systems,” in *IEEE 10th International Symposium on the Maintenance and Evolution of Service-Oriented and Cloud-Based Environments, MESOCA 2016*. IEEE, 2016.

[6] C. Anderson, “Docker [Software Engineering],” *IEEE Software*, no. 3, pp. 102–c3, 2015.

[7] D. Merkel, “Docker: lightweight linux containers for consistent development and deployment,” *Linux Journal*, vol. 2014, no. 239, p. 2, 2014.

[8] Docker.com. (2016) Supported drivers. [Online]. Available: https://docs.docker.com/machine/drivers/

[9] N. Naik, “Applying computational intelligence for enhancing the dependability of multi-cloud systems using Docker Swarm,” in *IEEE Symposium Series on Computational Intelligence (SSCI)*, 2016.

[10] M. M. Najafabadi, F. Villanustre, T. M. Khoshgoftaar, N. Seliya, R. Wald, and E. Muharemagic, “Deep learning applications and challenges in big data analytics,” *Journal of Big Data*, vol. 2, no. 1, pp. 1–21, 2015.

[11] M. Chen, S. Mao, Y. Zhang, and V. C. M. Leung, “Big data analysis,” in *Big Data*. Springer, 2014, pp. 51–58.

[12] V. Mayer-Schönberger and K. Cukier, *Big data: A revolution that will transform how we live, work, and think*. Houghton Mifflin Harcourt, 2013.

[13] A. Trnka, “Big data analysis,” *European Journal of Science and Theology*, vol. 10, no. 1, pp. 143–148, 2014.

[14] J. Turnbull, *The Docker Book: Containerization is the new Virtualization*. James Turnbull, 2014.

[15] Docker.com. (2016) Docker swarm. [Online]. Available: https://www.docker.com/products/docker-swarm

[16] ——. (2016) High availability in docker swarm. [Online]. Available: https://docs.docker.com/swarm/multi-manager-setup/

[17] ——. (2016, July 28) Docker built-in orchestration ready for production: Docker 1.12 goes ga. [Online]. Available: https://blog.docker.com/2016/07/docker-built-in-orchestration-ready-for-production-docker-1-12-goes-ga/

[18] N. Naik and P. Jenkins, "Securing digital identities in the cloud by selecting an apposite federated identity management from saml, oauth and openid connect," in *2017 11th International Conference on Research Challenges in Information Science (RCIS)*. IEEE, 2017, pp. 163–174.

[19] N. Naik, P. Jenkins, and D. Newell, "Choice of suitable identity and access management standards for mobile computing and communication," in *2017 24th International Conference on Telecommunications (ICT)*. IEEE, 2017, pp. 1–6.

[20] N. Naik and P. Jenkins, "An analysis of open standard identity protocols in cloud computing security paradigm," in *14th IEEE International Conference on Dependable, Autonomic and Secure Computing (DASC 2016)*. IEEE, 2016.

[21] ——, "A secure mobile cloud identity: Criteria for effective identity and access management standards," in *2016 4th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*. IEEE, 2016, pp. 89–90.

[22] Docker.com. (2017) Manage data in containers. [Online]. Available: https://docs.docker.com/engine/tutorials/dockervolumes/

[23] Apache. (2017, February 9) Apache Ambari. [Online]. Available: https://ambari.apache.org/

[24] K. Horvath. (2014, May 26) Ambari Shell. [Online]. Available: http://blog.sequenceiq.com/blog/2014/05/26/ambari-shell/

[25] J. Matyas. (2014, December 4) Multinode cluster with ambari 1.7.0 - in docker. [Online]. Available: http://blog.sequenceiq.com/blog/2014/12/04/multinode-ambari-1-7-0/

[26] Gethue.com. (2017) How to configure hue for your hadoop cluster. [Online]. Available: http://gethue.com/how-to-configure-hue-in-your-hadoop-cluster/

[27] J. Zwicker. (2015, February 10) Let's build a modern hadoop. [Online]. Available: https://medium.com/pachyderm-data/lets-build-a-modern-hadoop-4fc160f8d74f#.ltxyi7mvu

[28] N. Naik, P. Jenkins, P. Davies, and D. Newell, "Native web communication protocols and their effects on the performance of web services and systems," in *16th IEEE International Conference on Computer and Information Technology (CIT)*. IEEE, 2016, pp. 219–225.

[29] N. Naik and P. Jenkins, "Web protocols and challenges of web latency in the web of things," in *2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN)*. IEEE, 2016, pp. 845–850.

[30] D. Sayers. (2015, June 15) 5 Must-see Docker big data use cases that show Docker's processing power. [Online]. Available: http://www.midvision.com/blog/5-must-see-docker-big-data-use-cases-that-show-dockers-processing-power

[31] S. Hall. (2016, May 10) Pachyderm challenges hadoop with containerized data lakes. [Online]. Available: http://thenewstack.io/pachyderm-aims-displace-hadoop-container-based-collaborative-data-analysis-platform/