Building Modern Clouds: Using Docker, Kubernetes & Google Cloud Platform

Jay Shah
Telecommunication and Network Engineering
Southern Methodist University
Dallas, TX. USA
jayashoks@smu.edu

Dushyant Dubaria
Telecommunication and Network Engineering
Southern Methodist University
Dallas, TX. USA
ddubaria@smu.edu

***Abstract*—To develop and build a modern cloud infrastructure or DevOps implementation than both Docker and Kubernetes have revolutionized the era of software development and operations. Although both are different, they unify the process of development and integration, it is now possible to build any architecture by using these technologies. Docker is used to build, ship and run any application anywhere. Docker allows the use of the same available resources. These containers can be used to make deployments much faster. Containers use less space, are reliable and are very fast. Docker Swarm helps to manage the docker container. Kubernetes is an automated container management, deployment and scaling platform. Using Google Cloud Platform to deploy containers on Kubernetes Engine enabling rapid application development and management. Kubernetes provides key features like deployment, easy ways to scale, and monitoring.**

Keywords—Cloud, Docker, Kubernetes, Google Cloud Platform, Containers, Orchestration

I. Introduction

Docker and Kubernetes have revolutionized the way of DevOps consulting [1] and both are leading container orchestration tools today [2]. There is always a challenge to control an increase in the demand of scaling and auto-healing of the network and virtual instances. Managing the containers is always as task for any company because microservices which are running on the containers do not communicate with each other. They work independently as a separate entity. This is where kubernetes steps in [13]. Kubernetes is nothing but a platform to manage containers. These containers can be docker containers or any other alternative containers. Kubernetes orchestrates, manages and forms a line of communication between these containers. This paper is divided into three sections. The first Section covers Docker: Containerization Using Docker, Docker for networking [4], The Docker File and hosting a WordPress website using Docker. In the Second Section, we have covered Kubernetes: The Role and Architecture of Kubernetes, basic concepts, features, Kubernetes clusters, scaling and deploying applications and hosting a Web Server Using Helm [10]. In the Final Section, we have discussed Google Cloud Platform, its compatibility, services, features and how Kubernetes and GCP go hand in hand [9]. To summarize we have mentioned the difference between the two-orchestration tool [24].

II. Docker

A. Containerization Using Docker

Containerization is a way of running multiple software applications on the same machine. Each of which is run in an isolated environment called container. A container is a closed environment for the software. It bundles all the files and libraries that the application needs to function correctly. Multiple containers can be deployed on the same machine and share the resources. [2] Docker uses images to spawn containers. Multiple containers can be created using an image. Images are indexed and saved in an online repository managed by Docker [6] [16]. Here is the example where we will know the power of Docker: We have a web application running on Ubuntu 15. We need to upgrade your OS to the latest Ubuntu 17. But then the software will not run as it depends on libraries that are only available in Ubuntu 15. Using a Docker image to run this software, we can upgrade Ubuntu to the latest version and ensure that the application will continue to run in complete isolation of the OS current libraries [3].

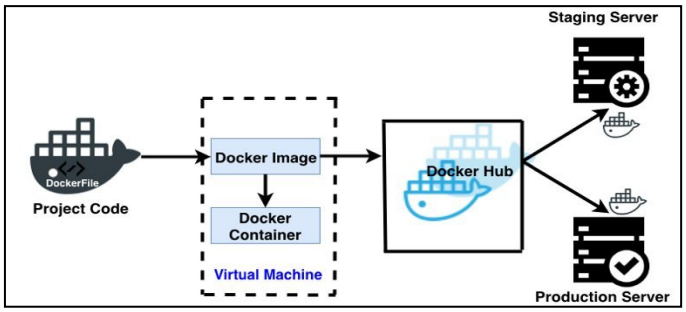

Fig. 1. Docker Containerization.

The DockerFile is an instructions file used for building containers [5]. Building an image is an incremental process [2]. We can download the Apache web server image (httpd) and use a DockerFile to install components on top of it like a specific version of PHP [17]. We can add new packages, change configuration files, create new users and groups, and even copy files and directories to the image.

All commands that deal with Docker containers and images start with docker command followed by subcommands and switches. Let's pull an image, called Busybox from the repository. Busybox is a tiny Linux kernel that has very basic

978-1-7281-0554-3/19/$31.00©2019 IEEE

functionality [3]. Here is the command: docker run --detach –name server Busybox.

B. Docker for Networking

The Docker container is not a virtual machine. This means that it uses the existing network interface of the host [17]. Once a container is created, a private loopback interface is available for the container for internal communication. It is totally isolated from the outside traffic [2]. Docker also creates a bridge interface called docker0 that is responsible for carrying traffic from and to the container. The docker0 interface acts as a router for the container [5]. It can be used for internal communications between different containers and the host. It can also be used for routing traffic from and to the outside network. [4] Docker provides three modes of networking: closed, bridged and open.

Docker supports three methods of storage: volume, bind and tmpfs mounts:

- A volume is just a mount point to a directory inside the container where files can be shared with the host.
- Docker uses UFS-Union File System to work. The image may be a combination of several images, each of which has its own file systems [6]. Those file systems are layered together so that common files and libraries can be reused [19].
- A bind mount refers to a mount point on the host that targets some destination directory on the container.

C. Hosting a web server using Docker

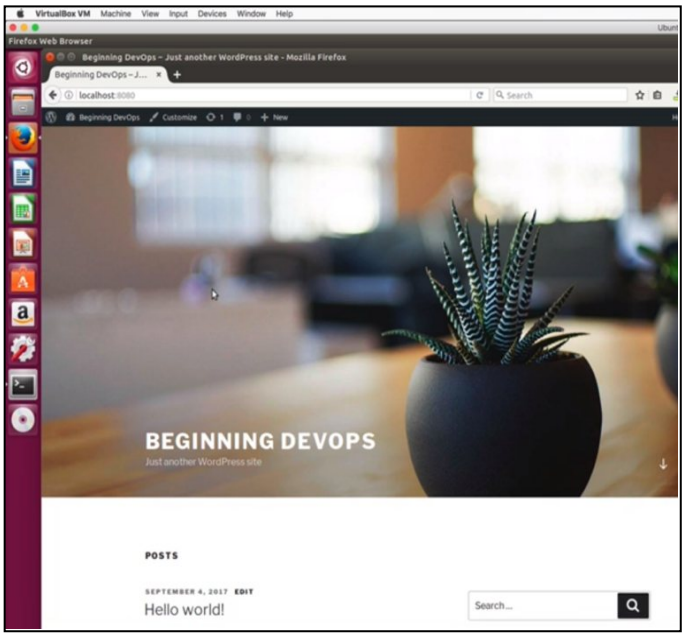

Fig. 2. WordPress using Docker.

The following command will create a web server container based on the httpd image [6] [15]. It will bind-mount the current directory on the host to the web directory of the container so that we can add HTML files: docker run -d --name webserver - p 8080:80 -v.

The other docker commands are as follows [6] [22]:

- docker kill
- docker create
- docker start
- docker stop
- docker ps
- docker rm
- docker login
- docker tag
- docker exec
- docker image
- docker push
- docker pull
- docker logs

III. KUBERNETES

Kubernetes is all about orchestration of containers. Kubernetes is also written as K8s which means a pilot in Greek. So, it is the captain of the ship with all the containers running inside [5]. Its container management responsibilities include container deployment, scaling & descaling of containers & container load balancing. Kubernetes is an open-source platform for managing containerized workloads, services and for automating deployment, scaling and orchestration of containerized applications [19]. Kubernetes is all about running multiple connected containers all organized in pods [1] [2]. Just like docker is the de-facto standard for containers, kubernetes is the de-facto standard for orchestration of containers. Kubernetes does not limit the supported application types. Kubernetes has the power to support a variety of workloads like stateful, stateless and data processing workloads [4]. If the container is able to run the application, kubernetes can definitely run the application. They have new release every 3 months so it is rapidly evolving. The main application of kubernetes is for orchestration of containers. We can deploy containers using kubectl as well deploying web app. We can create ingress routing and run stateful services on kubernetes. We can also use kubernetes to manage secrets and passwords.

A. Features of Kubernetes

1) Automatic Bin-Packing: Kubernetes automatically packages the application and schedules the containers based on the requirements [5]. To ensure complete utilization and save unused resources, Kubernetes balances between critical and best-effort workloads [10].

2) Load Balancing and Service Discovery: With Kubernetes, there is no need to worry about networking and communication because Kubernetes will automatically assign IP addresses to containers and a single DNS name for a set of containers, that can load-balance traffic inside the cluster [18].

3) Storage Orchestration: With Kubernetes, you can mount the storage system of your choice [9] [17]. You can either opt for local storage, or choose a public cloud provider such as GCP or AWS, or perhaps use a shared network storage system such as NFS, iSCSI, etc.

4) Self-Healing: Kubernetes can automatically restart containers that fail during execution and kills those containers that don't respond to user-defined health checks [16]. But if nodes itself die, then it replaces and reschedules those failed containers on other available nodes.

5) Secret and configuration Management: Kubernetes can help you deploy and update secrets and application configuration without rebuilding image and without exposing secrets in your stack configuration [2] [9].

6) Automatic Rollbacks and Rollouts: Kubernetes progressively rolls out changes and updates to your application or its configuration, by ensuring that not all instances are worked at the same instance [1] [12]. Even if something goes wrong, Kubernetes will roll back the change for you.

B. Kubernetes Architecture

1) Master Node: The master node is responsible for the management of Kubernetes cluster [13]. It is mainly the entry point for all administrative tasks. the master node is hosting the controlling processes that are available for the entire environment. On the master node, there are few services that are available and are explained below.

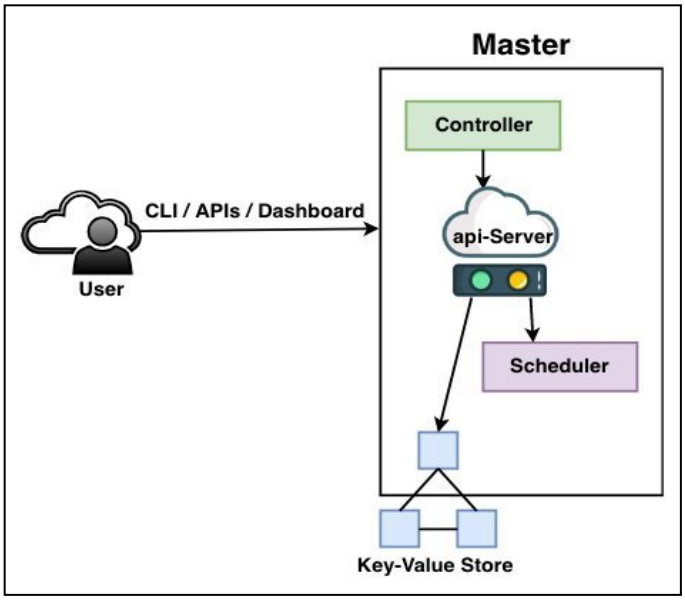

Fig. 3. Kubernetes Architecture (Master Node).

a) API Server: There is an API server which is exposing functionality to external users. We manage kubernetes using a kubectl command and basically, we are talking to the API server so the commands are usually received by the API server [18].

b) Controller Manager: Then there is a controller manager which manages the process and these services are running together on the kubernetes master

c) Scheduler: There is a scheduler which ensures that the kubernetes pods or the deployments are scheduled somewhere in the environment.

d) ETCD: There is etcd which is the backend database with the key value pair stored within the database [16].

2) Worker/Slave nodes: Other than the master, there are worker nodes, so there can be many worker nodes, it depends on how big you want to create your architecture. They are also called slave nodes. [12].

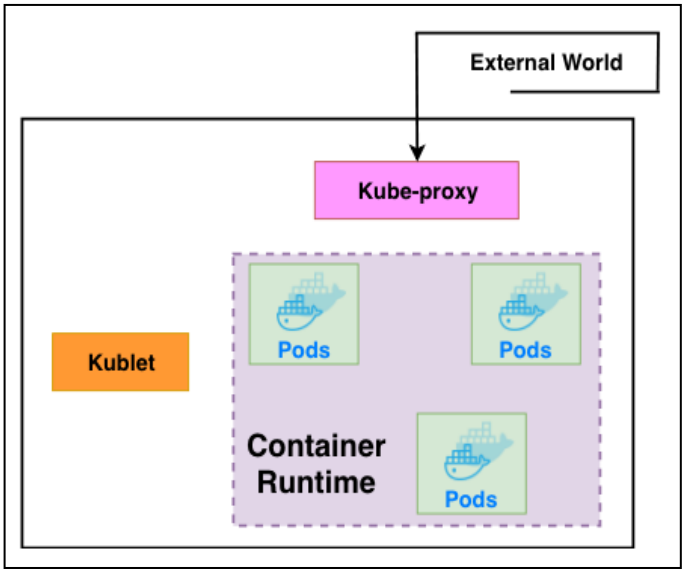

Fig. 4. Kubernetes Architecture (Worker/Slave Node)

a) Kubelet: Now going towards the worker nodes, the most essential part of the worker node is the CE that is the Container Engine and it is the one that makes it possible to run containers [18]. Now to run the containers, kubernetes needs something to talk to and that's the kubelet and it is an essential component that works on the worker nodes.

b) Kube-proxy: there is also a service called proxy which makes sure that whatever is happening in the container is made available [19]. Now, how the communication works in kubernetes is: controller manager in master node communicates with the kubelet that are working on different worker nodes so controller manager is the one that makes sure the work is being done. This controller manager is fed by the scheduler and the API server which is you running the different commands telling Kubernetes what to do.

c) Pods: In kubernetes, there are different hosts and kubernetes takes care of managing all these hosts in a cluster. Joining all the hosts in a cluster is beneficial as kubernetes can run containers and these will run in what we call a Pod. Pods can be replicated, and it is mainly for redundancy and scalability [2]. There will be group of containers in pods that makes up an application. Within the pod, there is also a reference to the storage, storage can be inside or outside the pod and containers can connect to the storage that is made available. Kubernetes makes sure all these pods are monitored and if any problem shoots up, new pods are scheduled.

3) Kubernetes Cluster: Cluster is a group of other servers where the containers can be deployed. These clusters can be managed by using the API called "Kubelet". [7] [13].

```
apiVersion: apps/v1beta2 # for versions before 1.9.0 use apps
kind: Deployment
metadata:
  name: redis-master
spec:
  selector:
    matchLabels:
      app: redis
      role: master
      tier: backend
  replicas: 1
  template:
    metadata:
      labels:
        app: redis
        role: master
        tier: backend
    spec:
      containers:
      - name: master
        image: k8s.gcr.io/redis:e2e  # or just image: redis
        resources:
          requests:
            cpu: 100m
            memory: 100Mi
        ports:
        - containerPort: 6379
```

Fig. 5. Kubernetes Architecture (Worker/Slave Node)

The master kubernetes which is also known as worker can communicatewith other cluster nodes using the Kubelet API [7] [15]. Kubernetes can handle deployments and make it easier for the user. You can specify how much CPU, Memory, Storage you want to give your containers.

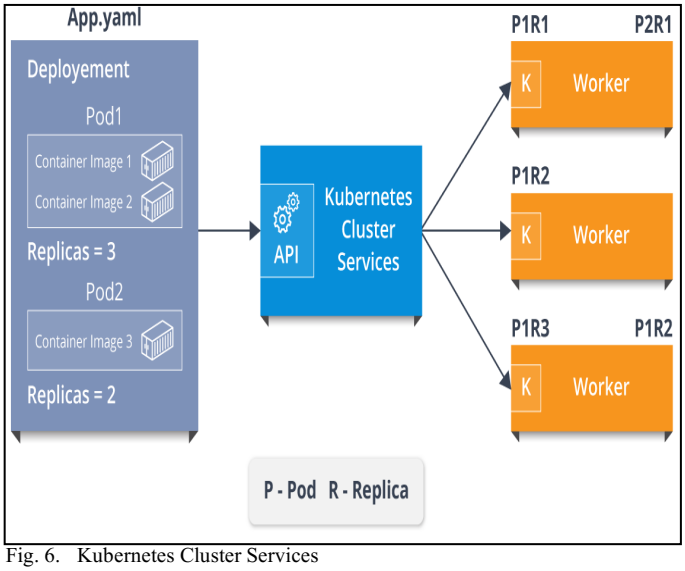

Fig. 6. Kubernetes Cluster Services

Kubernetes would spin a container for you. This is known as "Desired state Management" If you want to provide monitoring for your containers Kubernetes allows you to create an APP which will constantly monitor and have auto recovery featuresin it [21] [25]. Kubernetes allows usto write a code in yml file. These yml file contains pods. Pods are the smalles deployment module. It is the responsibility of the K8 cluster to deploy these pods [1] [3]. You can also configure what TCP port or servers the pods are running on.

C. Deploying WordPress Using Helm

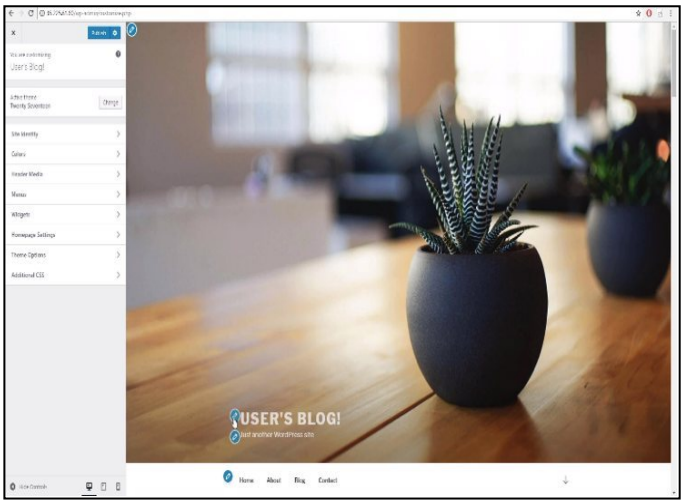

Fig. 7. WordPress Using Kubernetes and Helm

Helm is a package management tool. Helm can be used to add packages, services, etc. to the APPs which are deployed using Kubernetes. Helm is nothing but a package manager for kubernetes [7]. Helm-Charts helps the user to install, upgrade any applications of kubernetes. Even the most complex one's. The advantages of Helm-Chart are that they are easy to create, is faster, easy to version and easy to publish and share [14]. Helm-Charts describe even the most complex Kubernetes apps, they provide reliable application installation, and they serve as a single point of authority.

Here are the few Kubernetes Commands [15]:

- List pods with *"kubectl get pods"* command
- List deployment with *"kubectl get deployments"* cmd
- List services with the *"kubectl get services"* command

To run WordPress with helm on Kubernetes cluster, we need to install the software using the helm package manager [20] [21] [22]. First, we must start installing and upgrading the helm on our local machine. To install we use "brew install kubernetes-helm" and if the package is already installed, we can use "brew upgrade kubernetes-helm". Then initialize the helm & configure tiller by using "helm init". If tiller on our cluster is already installed the just enter "helm init --upgrade" [20]. In order to allow tiller to install & configure the required resources on my cluster, we should create a new service account with the name "tiller" in the System-Wide "kube-system" NameSpace [22]. Now, install the WordPress by creating a NameSpace on Kubernetes with the command kubectl create namespace varmywordpress, then on next line-

namespace "varmywordpress" is created. This helps in configuring all the all the required application for the WordPress web application to run including WordPress, elastic load balancing and elastic block store volume for managing data for MariaDB [20] [21]. We can now connect to our own WordPress site using the login credentials.

IV. Google Cloud Platform (GCP)

The purpose is not only to install WordPress but to learn more about the Google Cloud Platform Console [9] [25]. In addition, learn how to use some of the resources such as Compute Engine and Cloud SQL. This will give you a good basic understanding of Google's cloud offering [11].

A. Setting us a Cloud Solution Environment

1) Installing and Configuring the CLI: Specifically the Cloud SDK, the Cloud SDK is the command line interface used for managing Google Cloud. It consists of several components including, but not limited to:

a) gcloud - Google Cloud CLI

b) gsutil - Cloud Storage CLI

c) bq - BigQuery CLI

d) kubectl - Kubernetes CLI

B. Planning and Configuring a Cloud Solution

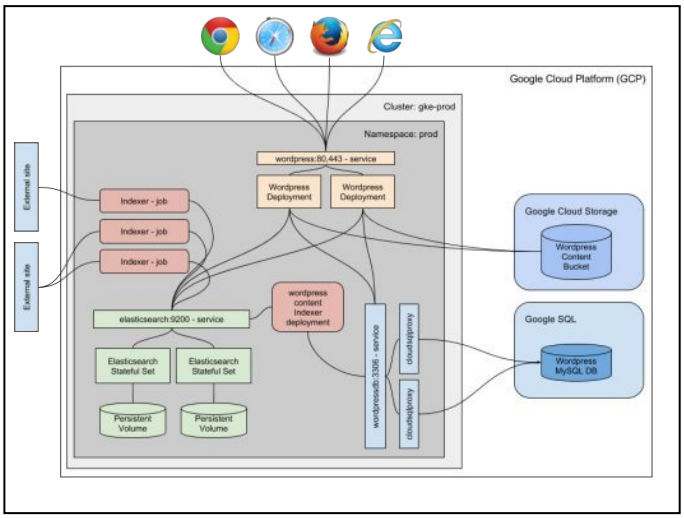

Fig. 8. Architecting for Kubernetes: WordPress Application

1) Planning and Configuring Data Storage Options: Considerations from data storage include [8]:

a) Product choice (e.g., Cloud SQL, BigQuery, Cloud Spanner, Cloud Bigtable)

b) Cloud SQL is used for a traditional SQL database

c) Spanner is used for horizontal SQL database scaling.

d) Bigtable is a sparse table, used when you have massive amounts of keyed data

e) BigQuery is a data warehouse service that can query massive data sets quickly.

f) Datastore is a NoSQL database built on top of Bigtable

C. Planning and Configuring Network Resources

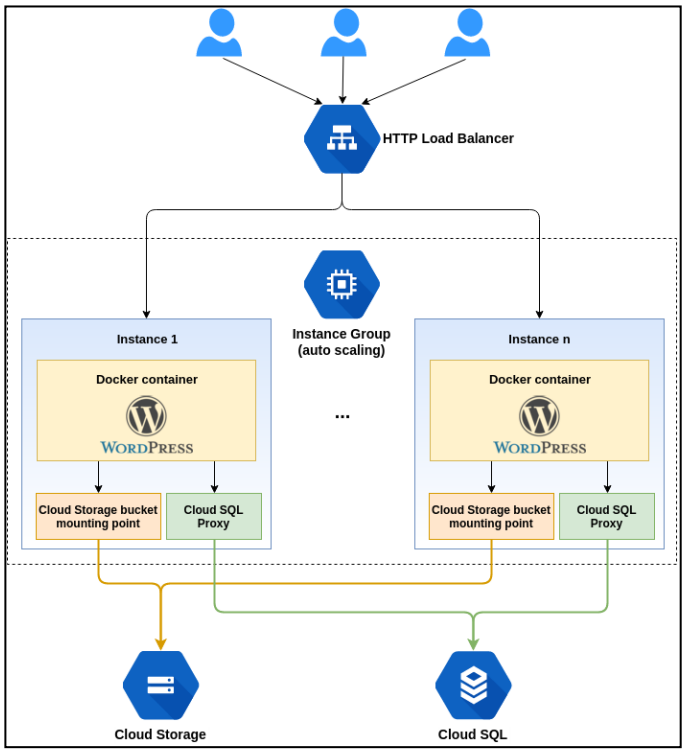

Fig. 9. WordPress Cluster using Docker and Google Cloud Platform

Google offers global and regional load balancers. Selecting the option that's right for your application depends on the traffic type [11].

1) Global load balancing options :

a) HTTP(S)

b) SSL Proxy

c) TCP Proxy

2) Regional load balancing options:

a) Network TCP/UDP load balancing

b) Internal TCP/UDP load balancing

D. Deploying and Implementing App Engine and Cloud Functions Resources Solution

1) App Engine provides traffic splitting to perform:

a) A/B testing

b) Canary deployments

c) Rollbacks

2) App Engine also provides multiple scaling mechanisms:

a) Automatic

b) Manual

c) Basic (Standard environments only)

V. KUBERNETES, DOCKER OR BOTH

TABLE I. COMPARISON OF DOCKER & KUBERNETES

Features	Kubernetes	Docker Swarm
Installation & Cluster Configuration	Installation is complicated, but once set up, the cluster is very strong	Installation is very simple, but the cluster is not very strong [23]
GUI	GUI is the Kubernetes Dashboard [9]	There is no GUI
Scalability	Highly scalable & scales fast [14] [23]	Highly scalable & scales 5x faster than Kubernetes
Auto-Scaling	Kubernetes can do auto-scaling	Docker Swarm cannot do auto-scaling
Load Balancing	Manual intervention needed for load balancing traffic between different containers in different Pods [8]	Docker Swarm does auto load balancing of traffic between containers in the cluster
Rolling Updates & Rollbacks	Can deploy Rolling updates & does automatic Rollbacks	Can deploy Rolling updates, but not automatic Rollbacks
Data Volumes	Can share storage volumes only with other containers in the same Pod	Can share storage volumes with any other container [8]
Logging & Monitoring	In-built tools for logging & monitoring [10]	3rd party tools like ELK should be used for logging & monitoring

A. *Usage of Kubernetes*

1) For deployment and monitoring option [12] [25].
2) For fast and reliable response times.
3) Deploying a big cluster.
4) Developing and managing largre number of containers.

B. *Usage of Docker*

1) Docker creates virtual containers. Its container management tool can consistently run software as long as system exits [23].

2) With the use of docker we can use single VM and can run many containers above that. It uses much less memory.

3) Container is a part of operating system that is responsible running a software. That makes the process faster.

VI. CONCLUSION

Docker and Kubernetes work at different levels [24]. Docker used in the IT software, makes use of the Containerization technology that helps to generate the different types of Linux containers. The Docker technology make use of the of the Linux kernel and the different and various features of Linux, because in Linux operating system the entire system is been separated from the other running applications, which makes the process or working of the Docker faster and accurate Kubernetes is the most popular container orchestration system and it was designed specifically with Google Cloud Platform integration in mind [8] [23]. The major advantage of using Kubernetes is orchestrating between many applications. It is also very useful for scaling many applications at very less time. By using this we can save our cost and time. So, it gives us management capabilities, saving cost, proper used of all resources. Kubernetes allows us to deploy and manage of the application running multiple host using the docker. Stateful and stateless applications are used on a large scale. Stateful involves the WordPress and MySQL whereas stateless involves the deploying PHP guestbook application.

REFERENCES

[1] 'DevOps for Networking' by Steve Armstrong, PackT Publications, first edition, 2016

[2] 'DevOps; Puppet, Docker and Kubernetes – Learning path' by Thomas Uphill, Arundel, Khare, Saito, Lee and Carol Hsu, Packt Publications, First Edition, 2017

[3] 'DevOps Troubleshooting' by Kyle Rankin, 2013

[4] 'Bringing DevOps for Networking with Ansible' by RedHat Enterprise

[5] Documentation of 'PaaS and Container clouds' by John Rofrano, IBM, NY University, 2015

[6] 'Projects in DevOps: Build real-world Processes' by Edunoix Learning Solutions, Online Web Courses

[7] Lecture slides of 'DevOps for NetOps' by Rick Sherman

[8] Lecture Slides of' DevOps for Networking' by Prof. Widhalm, SMU, 2018

[9] 'Cloud Native Applications- The Intersection of Agile Development and Cloud Platforms' by Douglas Bourgeois, David Kelly, Thomas Henry members of Deloitte Touche Tohmatsu Limited,2016

[10] 'Kubernetes from the ground up, deploy and scale performant and reliable containerized applications with Kubernetes' by Level Up Kubernetes Program, Basit Mustafa, Tao W, James Lee, Stefan Thorpe, 2018

[11] Article on 'Top cloud providers 2018: How AWS, Microsoft, Google stack up' by Larry Dignan in ZDNet

[12] Article on 'How does Kubernetes Networking Work' by Level Up Programming

[13] 'Kubernetes Networking' by Sahiti Kappagantula

[14] 'AWS vs Azure vs Google' By Mathew Finnegan and Scott Carey in ComputerWorld

[15] Article on 'How to Install Kubernetes Cluster on Ubuntu' by Hemant Sharma

[16] Article on 'What is Docker Container?' by Saurabh Kulshrestha

[17] 'Docker EE and CE forum and community' from official Docker Hub website

[18] 'Production-Grade container orchestration document' from Official website of Kubernetes

[19] Publication of 'Kubernetes Fundamentals' by The Linux Foundation Training, 2017

[20] GitHub account of Anh Tu Ngyuen

[21] Kubernetes Course on INE by David Coronel

[22] GitHub account of Dushyant Dubaria

[23] 'The Best architecture with Docker and Kubernetes- myth or reality?' by Dmitriy Paunin

[24] Lecture slides of 'Cloud Computing for Network Engineers' by Dr. Scott Kingsley, SMU, 2018

[25] 'Kubernetes Engine' document from official Google Cloud website